

NFC-Enabled Menu Ordering System

Final Report

ECE 445

TA: Lydia Majure

Yau Chan, Patrick Ding, Patric Takagi

Team #47

4/30/2013

Abstract

The Near Field Communication (NFC) menu system is designed to ease interaction with restaurant staff. The system uses NFC to allow customers to intuitively point and select items programmed with affordable NFC tags. After an order is complete, a checkout signal is selected (via NFC), allowing the entire cart to be sent via radio frequency (RF) to the kitchen for staff use. Hardware used includes an Arduino Mega 2560 R3, a XBEE Series 1 Wire Antenna, and a PN532 RFID NFC chip. The device is portable and powered by a 9V battery, making this device convenient and easy to use.

Contents

- 1 Introduction 1
 - 1.1 Title: NFC Menu 1
 - 1.2 Project Description..... 1
 - 1.3 Goals, Functions, Benefits and Features..... 2
- 2 Design..... 3
 - 2.1 Block Diagrams..... 3
 - 2.2 Block Descriptions..... 4
 - 2.2.1 Overall 4
 - 2.2.2 Menu Module..... 4
 - 2.2.3 Receiver Module 5
 - 2.3 Schematics 6
 - 2.4 Process Flow Chart..... 7
 - 2.5 Simulations and Calculations 8
- 3 Results, Graphs, and Measurements 11
- 4 Cost Analysis 13
- 5 Conclusions 14
 - 5.1 Accomplishments..... 14
 - 5.2 Uncertainties..... 14
 - 5.3 Ethics..... 14
 - 5.4 Future Work..... 15
- 6 References 17
- 7 Appendix 19
 - 7.1 Requirements & Verification Table..... 19
 - 7.2 Arduino Code 23

1 Introduction

1.1 Title: NFC Menu

1.2 Project Description

Customers at restaurants desire prompt service, convenient ordering, and great food. We have no control over the quality of food served; however, our project is designed to increase customer satisfaction by taking advantage of the first two goals and making the restaurant experience as quick and painless as possible.

Another problem that affects restaurants is the hectic queue due to a big lunch or dinner rush. Checks may be misplaced in the kitchen, and customers will be forced to wait for their misplaced order to be prepared. Also, there is an innate bottleneck when only waiters can place orders. A table may be ready to order, but their waiter may be serving food or cleaning up after another table. We wanted to find a solution that would allow patrons to place an order whenever they were ready.

Our idea is to help restaurants solve these problems by implementing a low-cost solution to increasing customer and restaurant staff satisfaction. The idea is creating a menu using NFC technology. NFC tags will be affixed to the restaurant menu with a specifically-designed NFC reader hard-wired to the menu. Customers can use the reader to select menu items by hovering over the appropriate NFC tags on their menus. Since NFC-tags are so low profile (almost as small as a sticker!), restaurants can keep the current design/layout of their menus and integrate our new technology with low switching costs.

The menu will contain an RF module that will send the order directly to the kitchen, reducing the work that waiters need to do. This will allow the restaurant to hire less waiters and will also improve any miscommunication between server and patron. Imagine if you are in a foreign country where you do not speak the native language. Our solution will allow you to intuitively point and select the items you want, keeping difficult communication between server and guest minimal.

Of course, our product will not be attractive to every restaurant. Some restaurants thrive through the friendly service that their staff brings to the table. Our product is designed for fast-casual type restaurants where you are solely interested in getting your food and eating.

1.3 Goals, Functions, Benefits and Features

Goals & Functions:

- Improve customer and staff restaurant experience
- Up to 256 menus can be used at once
- Up to 256 items available for purchase on item catalogue (menu)
- Up to 128 unique selections can be transmitted in each order
- Small NFC Reader to detect and send NFC tags to microcontroller memory
- RF communication link between menu and kitchen
- Simple yet powerful computer user interface in kitchen for staff use

Benefits:

- Low-cost compared to other electronic menu options
- Reduction in wait staff required for dine-in customers
- Quick and accurate processing of take-out orders
- Enables precise queuing in the kitchen
- Simple integration with current menu (no significant redesign required)
- Eliminates need for cumbersome communication between server and guest

Features:

- Sleek menu with intuitive point-and-select interface
- Fast wired communication between reader and menu microcontroller
- LCD display to aid order selection/confirmation
- Backlit LCD module for use in dim restaurants
- Quick RF communication between menu and kitchen

2 Design

2.1 Block Diagrams

Figure 2.1: Top Level Block Diagram

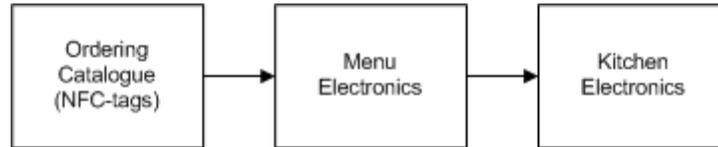
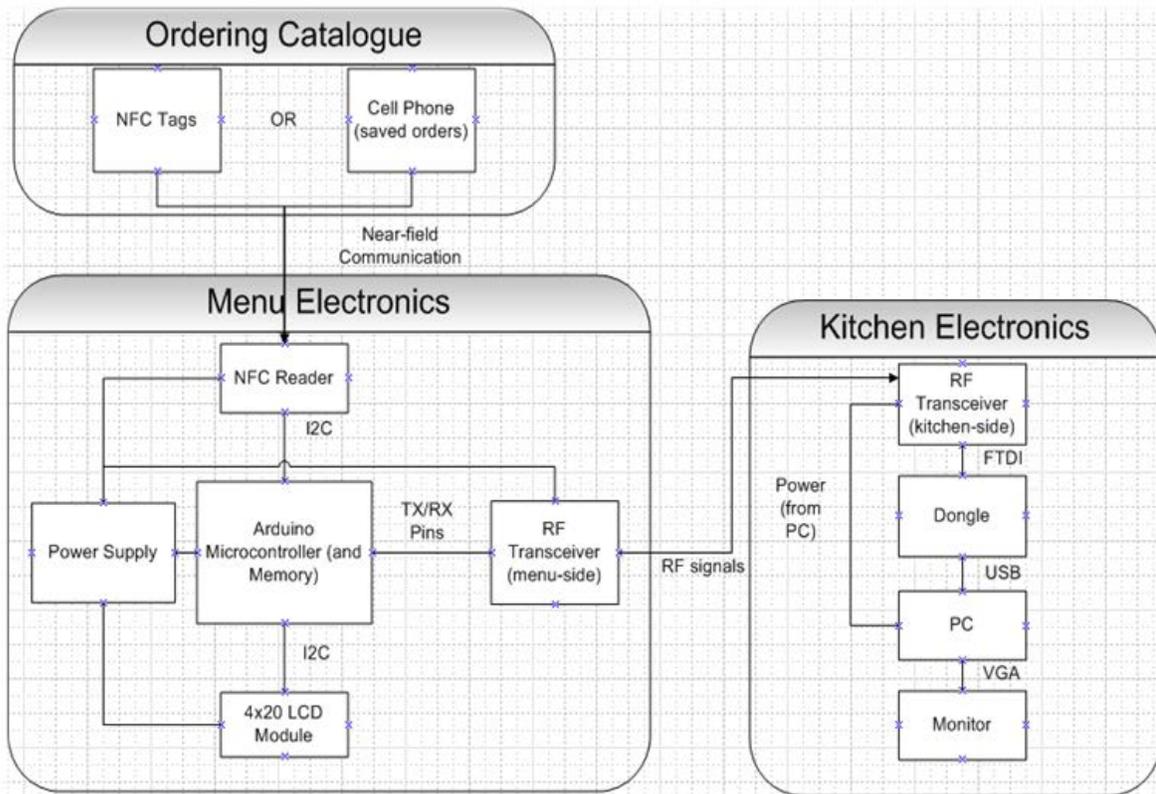


Figure 2.2: Detailed Block Diagram



2.2 Block Descriptions

2.2.1 Overall

To create this NFC ordering system we used a menu that acted as an interface for the user. The menu had a RF transmitter that wirelessly transmitted data of a customer's order to a receiver which transmitted the data to a computer via USB port. For our prototype, the computer translated the data into English and displayed the order.

NFC Tags:

These tags are small and easily programmable via computer software. The tags do not require power and will be stuck inside the menu to represent various items-to-order. When a customer scans a picture of the item they want to order, they will actually be scanning the NFC tag underneath the picture. When scanned by the NFC reader, the NFC tags will transmit the data that represents an item to the reader using near field communication technology.

2.2.2 Menu Module

Power Supply:

This will supply power to all the components that require it. The only component that does not require the power supply to function is the NFC tags. The power supply will be connected to microcontroller in parallel with a battery indicator circuit. The microcontroller will distribute power to the components that it is connected to. The microcontroller takes 9V, so the power supply will consist of a 9V battery and a battery box IM361 that will connect the 9V battery to the microcontroller.

NFC Reader:

NFC uses magnetic induction between two loop antennas. One antenna belongs to the initiator (active) and the other, the target (passive). The initiator generates a carrier field in which the target responds by modulating the existing field. For our case, our reader acts as the initiator while the NFC tags act as the target. When we transmit our menu items from a smartphone onto the reader, the smartphone will act as the initiator instead while the reader acts as the target. The NFC reader we will be using is NXP's PN532 transceiver module that is soldered onto a NFC RFID module made by elechouse. This module comes with its own library and has I²C set as the default interface which is what we will use. It is be hardwired to the microcontroller. The range of this reader 4-6 cm requires 5V TTL for I²C. This interface only requires two bi-directional lines, Serial Data Line (SDA) and Serial Clock (SCL) which are pulled up with resistors. Data transfer is initiated with a start bit when the SDA is pulled low while SCL is high. Then, SDA sets the transferred bit while SCL is low and the data is received when SCL is an active high. When transfer is complete, a stop bit is sent by releasing the data line to allow it to be pulled up while SCL is constantly high.

RF Transceiver:

The transceiver will take the data that was picked up by the reader after the order is verified through the microcontroller. Once the order is confirmed, the transceiver will wirelessly transmit the data to another transceiver component which will be in the kitchen of the restaurant. This transceiver will also receive a confirmation from the kitchen transceiver to notify customers that the order has been received. The transceiver that will be used is the XBEE 1mW series 1 transceiver. This chip will draw 3.3 V at 50 mA from the microcontroller and features 250 kbps max data rate in the 2.4 GHz frequency band. The range for this device will be around 100 meters.

4x20 LCD Module:

This will display the order and acts as an interface that allows the customer to view and then confirm or cancel their order. The display will be a backlit LCD module that receives the data to display from the microcontroller. We will be using the HD44780U (LCD-II) as the display. This display requires 2.7V - 5.5 V of power and can display 80 max characters on the screen at once.

Arduino Microcontroller:

The microcontroller will take the data it receives from the NFC reader and will translate the data into the desired item to display on the display module. It will also be able to receive input from the customer in order to confirm or cancel the order. If confirmed, the microcontroller will transmit the data from the NFC reader to the transmitter. Some memory will be required to store up to 32 unique selections made by the user.

Battery Indicator:

This is a circuit consisting of a few basic components, particularly one red LED that will only light up when the voltage accepted from the battery drops below ~6.9V (the threshold voltage before the exponential plunge in the discharge rate curve for a 9V alkaline battery).

2.2.3 Receiver Module

RF Transceiver:

This will receive the data from the transmitter and will be transferred into a computer via a USB port. The transceiver will be placed in the kitchen to allow communication from the customer directly to the kitchen staff. When this transceiver has successfully received the order from the customers, it will send confirmation to the menu transceiver. The transceiver used is the XBEE 1mW series 1. This is the same transceiver being used as in the menu.

Dongle:

This is the device which interfaces the kitchen transceiver to the USB which will allow data to be transmitted into a computer. We will be using the XBEE explorer dongle. This will allow the transceiver

to attach to the usb port perfectly. The dongle will regulate the 5V supply from the USB to the 3.3V that the XBEE transceiver needs to run. It also eliminates the need for cables and allows for a small compact unit that can be kept plugged into a computer display.

2.3 Schematics

Figure 2.3: Menu Schematic

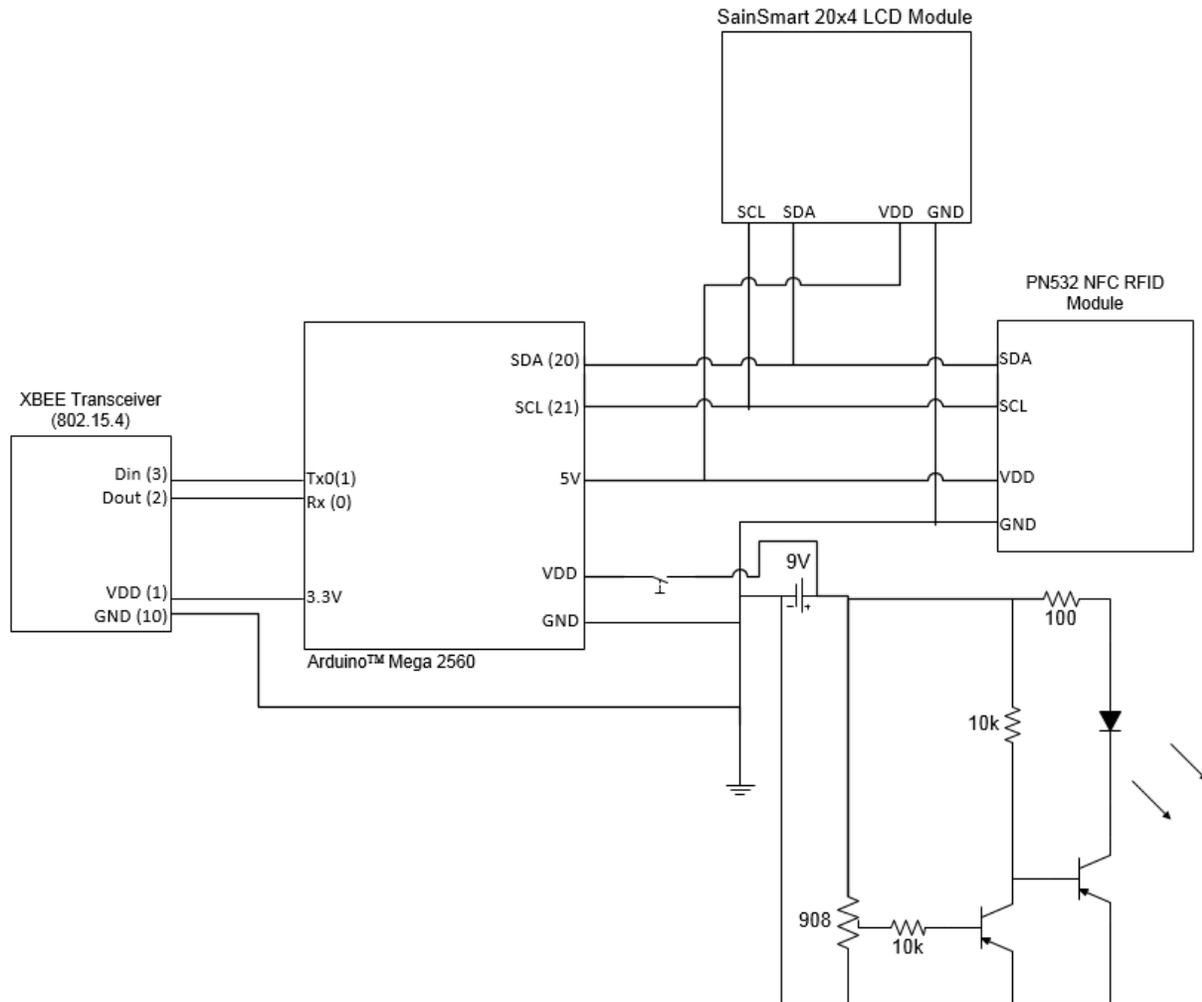
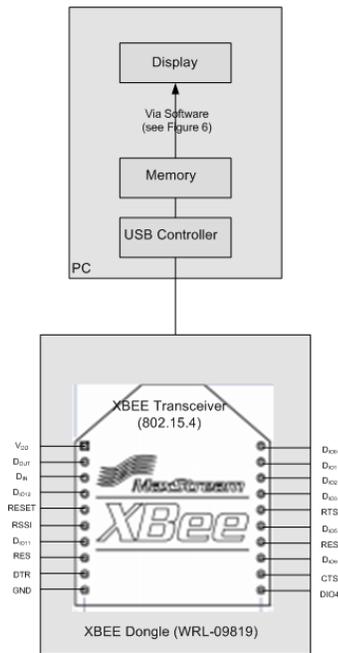
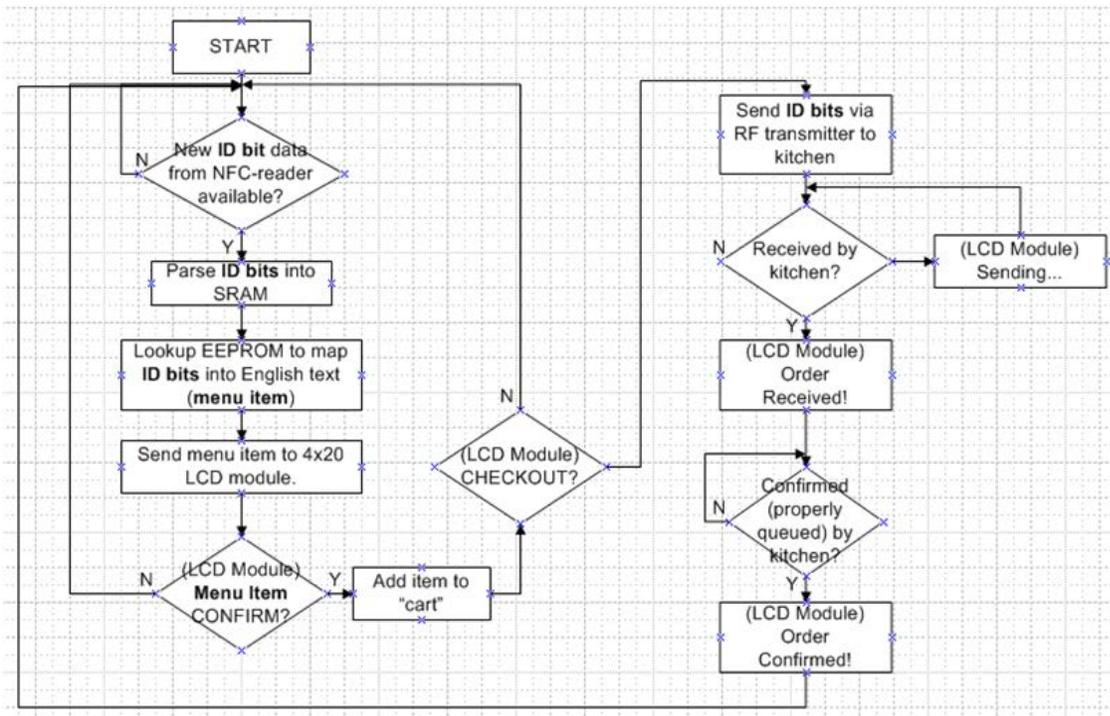


Figure 2.3: Receiver Schematic



2.4 Process Flow Chart:

Figure 2.4: Menu Flow Chart



2.5 Simulations and Calculations

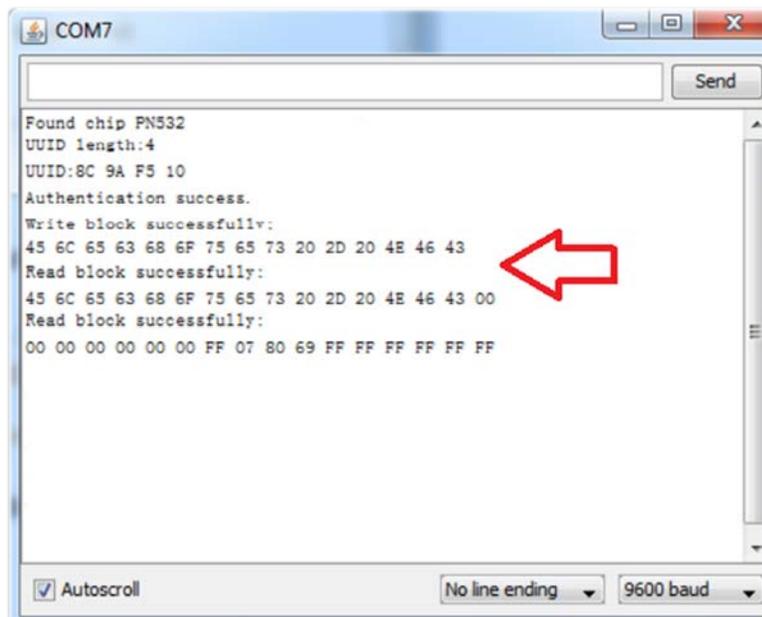
NFC Reader Preliminary Simulation

The following figures (Figures 7-8) show the expected results from scanning/receiving NFC information. We will be using a smartphone and our NFC chip to test these results and make sure the NFC part of the project works without any hitches. The software is an IDE (Integrated Development Environment) created by Arduino. The pseudo-codes for the programs are shown as well¹:

```
//Example code to write data
strcpy((char*)block, "Testing - NFC");
sta = nfc.MifareWriteBlock(blocknum, block); //Mifare is NXP's trademark for several of their chips, particularly the PN532
if(sta) //transceiver that is central to our NFC reader as well as many other
    Serial.println("Write block successfully:"); //NFC-enabled smartphones
```

```
//Example code to read data
//read block #4
Sta = nfc.MifareReadBlock(blocknum, block);
if(sta){
    Serial.println("Read block successfully:");
    Nfc.puthex(block,16);
    Serial.println();
}
```

Figure 2.5: Read/Write Example from Above Code

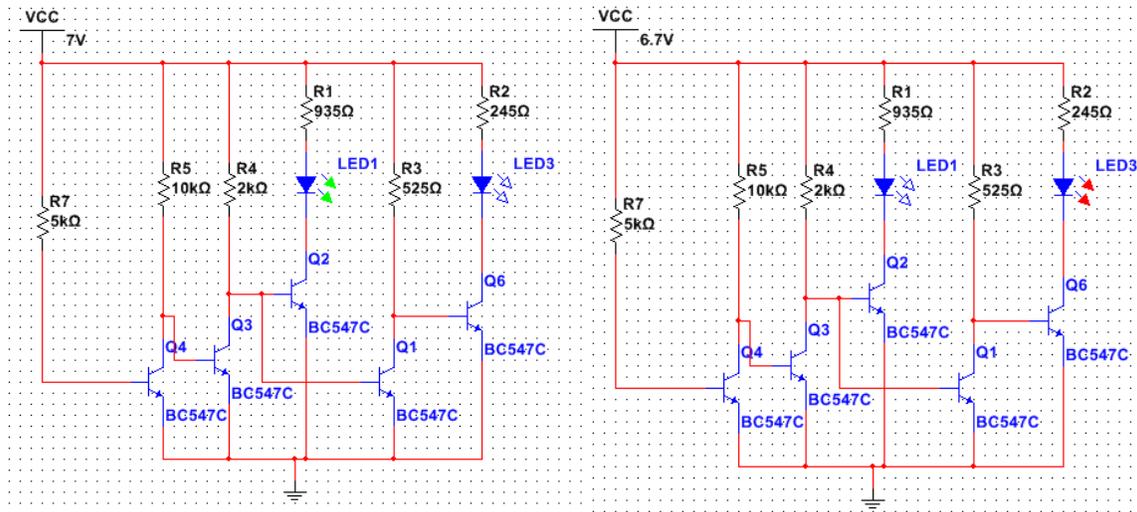


```
COM7
Found chip PN532
UUID length:4
UUID:8C 9A F5 10
Authentication success.
Write block successfully:
45 6C 65 63 68 6F 75 65 73 20 2D 20 4E 46 43
Read block successfully:
45 6C 65 63 68 6F 75 65 73 20 2D 20 4E 46 43 00
Read block successfully:
00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
```

¹ This code was only used in the NFC-Reader verification and was edited afterwards. Please view the attached code in the appendix to see the modifications.

Battery Indicator Preliminary Simulation

Figure 2.6: Battery Indicator Circuit Prior to Reductions



Shown in Figure 2.6 is the schematic of the battery life indicator circuit prior to a component reduction. The two SPICE schematics show the status of the LEDs above an approximately 6.7-6.9 voltage threshold (green LED on) and below the threshold when the green LED is off and the red LED is on.

The green LED, (LED1) is the power indicator; it is on when at ~7V or above. The red LED (LED3) is on when below the ~7V voltage threshold. These LEDs will have ~2V across when lit.

The BJTs (BC547C, NPN) all have a base-emitter voltage of approximately 0.7V and serve as switches in our case.

A potentiometer (variable resistor) was used in determining the left most resistor value for the threshold that we wanted (6.7-6.9V range) and afterwards, replaced by the equivalent valued resistor to reduce cost.

Using KVL at the nodes relevant to the LEDs, we see that for red:

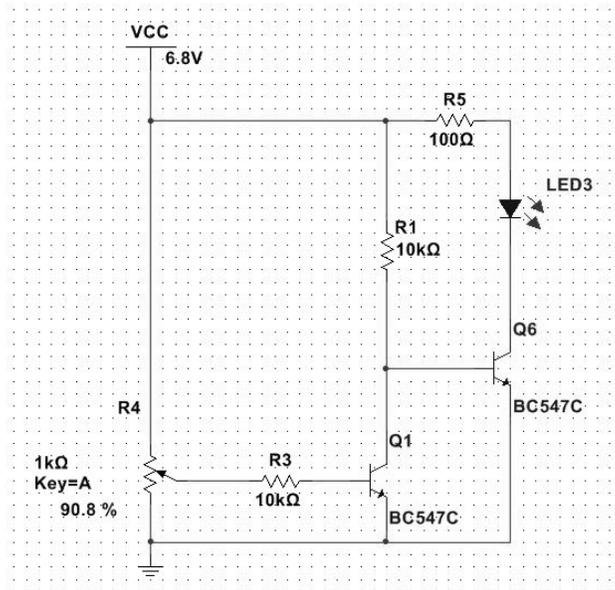
$$\begin{aligned} V_{CC} &= I_2 \cdot R_2 + V_{LED3} + V_{CE} \\ 6.7 &= (0.012 \cdot 245)A + V_{LED3} + 1.939V \\ V_{LED3} &= 1.821V \end{aligned}$$

For the green LED:

$$\begin{aligned} V_{CC} &= I_1 \cdot R_1 + V_{LED1} + V_{CE} \\ 6.7 &= 0.005 \cdot 935 + V_{LED1} + (-2.407mV) \\ V_{LED1} &= 2.027V \end{aligned}$$

Although this circuit worked very well. We decided that a green LED was a hindrance on our goal of minimizing size and power consumption. So we removed this LED and reduced the number of transistors so our final circuit ended up as shown in Figure 2.7.

Figure 2.7: Final Battery Indicator Circuit



We added a potentiometer, reduced the number of transistors, resistors, and LED. The potentiometer was used primarily in controlling the voltage of Q1 since the BJT required 0.6V-0.7V between base and emitter. When Q1 is not conductive LED3 is off. When Q1 is conductive, Q6 turns on and enough current flows through the LED to light it up.

Baud Rate Calculations

Arduino quartz oscillator operates at 16 MHz.

NFC Reader operates between 106 - 400 kbps bitrate:

@424 kHz: $16 \text{ MHz} / 424 \text{ kHz} = 37.7$

Frequency of NFC Reader stepped down by $\sim 38x$

Each item = 1 byte; $1 \text{ byte} / (424E3/8) = \mathbf{0.0000189s \text{ per scan}}$

@212 kHz: $16 \text{ MHz} / 212 \text{ kHz} = 75.7$

Frequency of NFC Reader stepped down by $\sim 76x$

$1 \text{ byte} / (212E3/8) = \mathbf{0.0000377s \text{ per scan}}$

@106 kHz: $16 \text{ MHz} / 106 \text{ kHz} = 151$

Frequency of NFC Reader stepped down by $151x$

$1 \text{ byte} / (106E3/8) = \mathbf{0.000075s \text{ per scan}}$

RF Transceiver operates between 1.2 - 250 kbps bitrate:

- @ **250kHz**: $16\text{MHz}/250\text{kHz} = 64$
 Frequency for transceiver will step down by 64x
 Full order of 128 items = 128 bytes. $128 \text{ bytes} / (250\text{E}3/8) = \mathbf{0.004 \text{ seconds per order}}$
- @ **125kHz**: $16\text{MHz}/125\text{kHz} = 128$
 Frequency for transceiver will step down by 128x
 $128 \text{ bytes} / (125\text{E}3/8) = \mathbf{0.008 \text{ seconds per order}}$
- @ **12.5kHz**: $16\text{MHz}/12.5\text{kHz} = 1280$
 Frequency for transceiver will step down by 1280x
 $128 \text{ bytes} / (12.5\text{E}3/8) = \mathbf{0.082 \text{ seconds per order}}$
- @ **1.2kHz**: $16\text{MHz}/1.2\text{kHz} = 13,333.33$
 Frequency for transceiver will step down by 13,333.33x
 $128 \text{ bytes} / (1.2\text{E}3/8) = \mathbf{0.853 \text{ seconds per order (little slow, but still manageable)}}$

3 Results, Graphs, and Measurements

Battery Life Calculations

Since menus are vital to a restaurant’s operation, we needed to include calculations from battery life. The system drew a whopping 320 mA when first completed. Knowing this was too high, our group attempted to limit this power draw by limiting the amount of current that the communication components and LCD backlight drew.

Figure 3.1: Reduction of Current Draw

Component	Before	After
Arduino	70mA	50mA
NFC Reader	90mA	40mA
LCD Module	120mA	60mA
XBEE transmitter	40mA	30mA
Total:	320mA	180mA

Figure 3.2: Hours of Expected Use

Battery	Capacity (mAh)	Power Draw (mA)				
		50	100	180	200	320
Zinc-Carbon	400	8.00	4.00	2.22	2.00	1.25
Alkaline	565	11.30	5.65	3.14	2.83	1.77
Lithium	1200	24.00	12.00	6.67	6.00	3.75

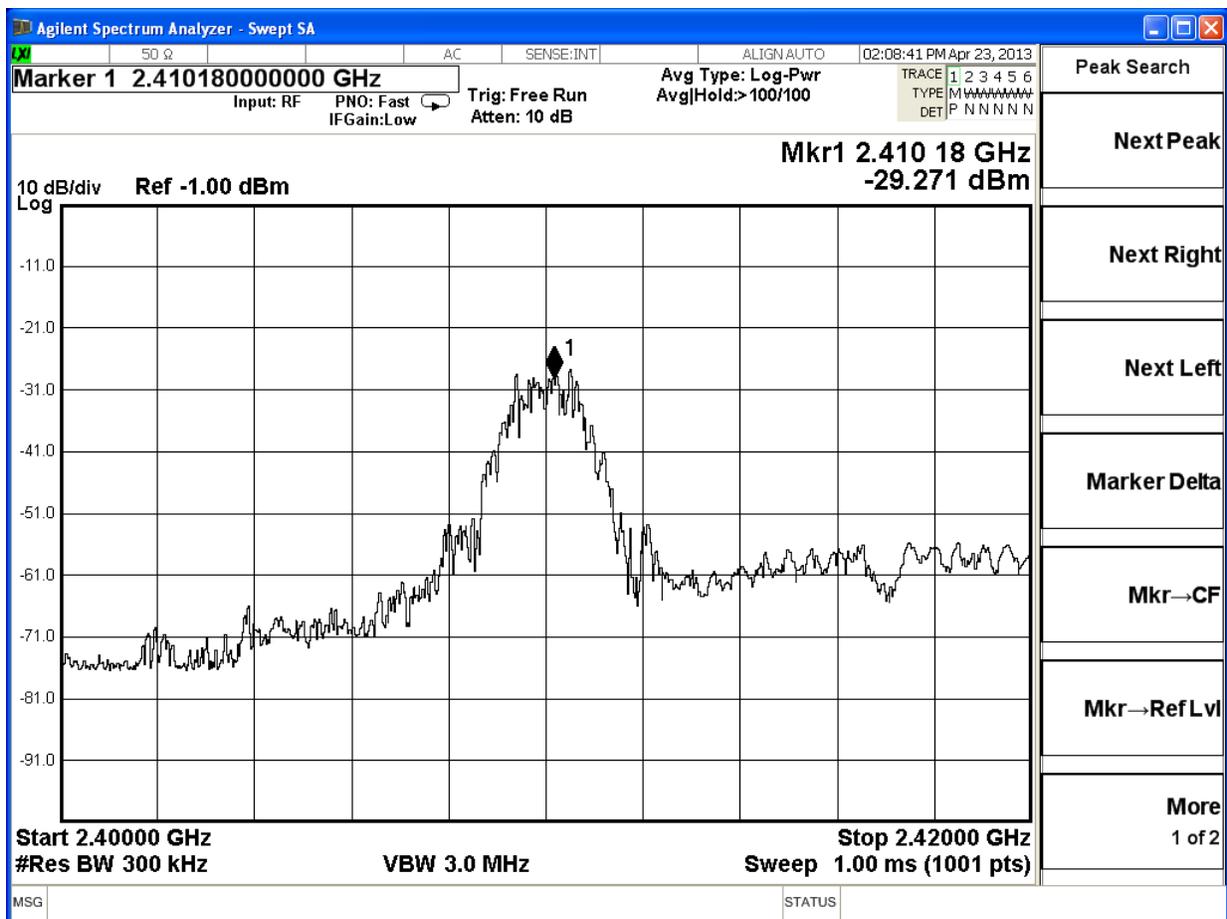
From Figure 3.1, NFC reader and XBEE transmitter’s power draws were limited when we changed the baud rate of those components. Since these components do not need to poll as quickly, power

consumption was reduced. The LCD Module also dropped its current draw by half when we added a potentiometer to limit the backlight brightness.

The red column in Figure 3.2 shows the expected life (in hours) of various batteries BEFORE we did any power limiting engineering. The green column shows the increased battery lifetime after we made several modifications to our design.

We estimate that users will spend about 10 minutes learning the interface and selecting each order. After our modifications and using a standard 565 mAh capacity battery, the menu would last roughly 3 hours. This would allow 18 orders to be placed, or about a night’s worth of orders for a busy restaurant. More power considerations are discussed in future work.

Figure 3.3: XBEE Transmission Frequency



We used a spectrum analyzer from the lab to measure the frequency of the XBEE’s transmission. The spectrum analyzer was attached with a small antenna to measure the XBEE’s transmission and find its frequency. The XBEE was connected to a computer using the USB dongle and sent transmissions using the program, X-CTU. The spectrum analyzer was set to “hold max” mode to measure the short transmission sent from the XBEE. Figure 3.3 shows that the frequency of the transmission is within the ISM 2.4 GHz frequency band

4 Cost Analysis

Labor

Name	Hourly Rate	Total Hours Invested	Total = Hourly Rate x 2.5 x Total Hours Invested
Patrick Ding	\$35.00	180	\$15,750
Yau Chan	\$35.00	180	\$15,750
Patric Takagi	\$35.00	180	\$15,750
Total Labor Costs		540	\$47,250

Parts

Item (P/N)	Unit Cost	Quantity	Total Cost (\$)
LED	\$0.42	1	\$0.42
PCB	\$0.00	1	\$0.00
Battery Box (IM361)	\$6.50	1	\$6.50
Resistors, Transistors	\$3.17	Total	\$3.17
Backlit LCD Display (LCD-00790)	\$15.00	1	\$15.00
RF Transceiver (WRL- 10534)	\$22.95	1	\$22.95
NFC Tags (NTAG203)	\$0.60	10	\$6.00
NFC-Reader	\$34.60	1	\$34.60
Dongle	\$12.95	1	\$24.95
Total Parts Costs			\$113.59

Grand Total

Section	Total
Labor	\$47,250.00
Parts	\$113.59
Total	\$47,363.59

5 Conclusions

5.1 Accomplishments

All of our project's verifications passed as proposed, and each of the modules were engineered to work properly. After we verified all modules, we then connected the entire system together with little interface hitches. The PCB was the last roadblock to a complete working system; luckily for us, soldering was a simple task and continuity was clean. We created a 3D-Printed enclosure for the system as a final polishing step. Our demo to the ECE445 staff was successful.

5.2 Uncertainties

We were uncertain on how to further reduce power, after limiting LCD backlight and communication baud rate. We have discussed further steps in the "future work" section of the paper (5.4).

Another uncertainty we have is how the system would work with multiple menus sending data at the same time. According to the XBEE datasheet, the receiver should be able to handle ~10 signals at the same time. However, further testing needs to be completed before we can support these claims.

A final uncertainty was the range of the RF transceivers. Although the system worked up to ~50 meters, we were not able to consistently receiver signals past that range. We believe that obstacles played a major role in hindering the RF signals. To counter this problem, we could use a higher model XBEE chip or increase antenna size.

5.3 Ethics

The purpose of this project is to provide a user with a more convenient way to order and enjoy food at a restaurant. In addition, the project should make it easier for restaurant staff to manage their orders during peak demand times. Since this product will be used by both customers and staff, proper safety is a must. If any unsafe bugs are discovered, we must work to develop a solution and recall the faulty products. This follows the first code of the IEEE Code of Ethics:

1. To accept responsibility in making decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

The menu will also be marketed to restaurants around the country, so it must be able to deliver on all of the features that we advertise. Data should not and will not be fabricated to deceive potential customers. This correlates to the third code of the IEEE Code of Ethics:

3. To be honest and realistic in stating claims or estimates based on available data;

Our team needed to learn and study how both near-field and radio frequency communication works before attempting to design the system. We also needed to learn how an Arduino behaves in order to succeed in our design. We believe this relates to codes 5 and 6 of the IEE Code of Ethics:

5. *To improve the understanding of technology, its appropriate application, and potential consequences;*
6. *To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;*

In this senior design class, we will be expected to interact with our peers and instructors technically and professionally as well (in peer reviews, weekly TA meetings, etc). If needed, we will not hesitate to seek advice from (or give advice to) the people that we find necessary. This correlates to the seventh, eighth, and tenth codes of the IEEE Code of Ethics:

7. *To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;*
8. *To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;*
10. *To assist colleagues and co-workers in their professional development and to support them in following this code of ethics;*

Finally, we are representing ourselves, our peers, our professors, and our school by working on this project. We will need to be sure to act in a way that honors all of these stakeholders. This follows the ninth code of the IEEE Code of Ethics.

9. *To avoid injuring others, their property, reputation, or employment by false or malicious action;*

5.4 Future Work

Although our project was a success in the scope of this class, there is still much future work to be done for the system to be completely marketable.

a.) Shrink Size of Menu

Our final product, while fairly small, was still a bit bulky to use around a restaurant. By shrinking the size and improving the efficiency of the PCB, we hope to make the entire system a more manageable size. This would allow greater convenience for users.

b.) Multiple Page Considerations

One problem with the menu is that when multiple pages are used (with tags on each page), the reader seems to randomly detect either Tag A or Tag B (on different pages). Future work would require some sort of shielding between each page to make sure that the correct tag is detected.

c.) Communication from Kitchen to Menu

One feature we would like to implement is communication from the kitchen back to the menu. Customers would be able to see the progress of their order, including expected wait time.

d.) Reduction in Power Consumption

Our attempts to limit power consumption took us from a current draw of 320 mA to 180 mA. This current draw is still too high for use in a restaurant application. Ryan May suggested that we could add solar cells to the menu to trickle charge the system. He also suggested a sleep mode or current regulators in the Arduino system to further limit power draw. An optimistic estimate would be ~60mA power draw for the system. With a typical 565 mAh capacity battery, this would bring us up to a reasonable 9 hours of battery life.

6 References

Battery Indicator Circuit:

"9v Battery Status Indicator Circuit." *Instructables*. N.p., n.d. Web. 6 Mar. 2013.
<<http://www.instructables.com/id/9v-battery-status-indicator-circuit/>>.

Ethics:

United States. IEEE. 7.8. IEEE Code of Ethics. Washington DC: , 2013. Web.

LCD Module:

Elger. "I2C LCD Display." *Projects:lcd_module*. N.p., n.d. Web. 23 Feb. 2013.
<http://elger.org/wiki/projects/lcd_module>.

"SainSmart IIC/I2C/TWI Serial 2004 20x4 LCD Module Shield For Arduino UNO MEGA R3."
SainSmart. N.p., n.d. Web. 21 Feb. 2013.
<<http://www.sainsmart.com/sainsmart-iic-i2c-twi-serial-2004-20x4-lcd-module-shield-for-arduino-uno-mega-r3.html>>.

Microcontroller:

"Arduino - ArduinoBoardMega2560." *Arduino - ArduinoBoardMega2560*. N.p., n.d. Web. 01 Oct. 2012. <<http://arduino.cc/en/Main/ArduinoBoardMega2560>>.

Fried, Limor. "Arduino Tutorial - Learn Electronics and Microcontrollers Using Arduino!" *Ladyadanet Blog*. N.p., 27 Apr. 2012. Web. 19 Feb. 2013.
<<http://www.ladyada.net/learn/arduino/index.html>>.

NFC:

"PN532 NFC RFID Module Kits -- Arduino Compatible." *Elechouse.com*. N.p., n.d. Web. 19 Feb. 2013.
<http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=90_93&products_id=2205>.

Jahn, Rene. "NFC/RFID for Beagleboard Xm with Java." *Blog SIB Visions RSS*. N.p., 26 Nov. 2012. Web. 19 Feb. 2013. <<http://blog.sibvisions.com/2012/11/26/nfc-rfid-for-beagleboard-xm-with-java/>>.

Power:

"Nine-volt battery." *Wikipedia, The Free Encyclopedia*. *Wikipedia, The Free Encyclopedia*, 27 Feb. 2013. Web. 28 Feb. 2013.
<http://en.wikipedia.org/w/index.php?title=Nine-volt_battery&oldid=540922147>

Transceiver:

XBee. "WRL-08664." XBee 1mW Chip Antenna. N.p., n.d. Web. 18 Feb. 2013.
<<https://www.sparkfun.com/products/8664>>.

XBee. "WRL-09819." XBee Explorer Dongle. N.p., n.d. Web. 18 Feb. 2013.
<<https://www.sparkfun.com/products/9819>>.

7 Appendix

7.1 Requirements & Verification Table

Requirement	Verification	Status (C-Complete or F-Failed)
<p>1. Menu Power Supply (9V Battery):</p> <ul style="list-style-type: none"> • Must be able to supply a voltage within the range of 7V-10V (safe range) • Must be able to supply a minimum current rating of 25mA to a maximum current rating of 100mA at load ranges between 360 and 90 ohms 	<p>1.</p> <p>a. Use a DMM to monitor the voltage</p> <p>b. While testing with loads between 360 and 90 ohms, we will use a DMM to monitor the current</p>	C
<p>2. Menu NFC Tags (Passive Element):</p> <ul style="list-style-type: none"> • Must be able to properly store messages • Must be able to transmit messages to the NFC Reader (functionality test) 	<p>2.</p> <p>a. We will use the smartphone application called TagWriter (by NXP) and compare the contents read by the smartphone with the content programmed into the NFC tag through Arduino's IDE</p> <p>b. We will write a plain text message to the tag through Arduino's IDE</p>	C
<p>3. Menu NFC Reader:</p> <ul style="list-style-type: none"> • Must be able to properly receive stored messages from the target • Must be supplied 5V (± 0.5 V) • Must be able to forward the received messages to the microcontroller at 9600 bit/s with 0.1% bit-error 	<p>3.</p> <p>a. Use TagWriter to receive plain text messages</p> <p>b. Probe the 5V feed from the microcontroller with a DMM</p> <p>c. Use Arduino's IDE in relay messages and spot for 0.1% bit-error</p>	C
<p>4. Menu RF Transceiver:</p> <ul style="list-style-type: none"> • Must be supplied with 2.8-3.4V • Must operate at ISM (industrial, scientific, and medical radio band) 2.4 GHz frequency band. • Must work at a range of approximately 100 meters. 	<p>4.</p> <p>a. Using a DMM, we will probe the voltage supplied in order to verify that it falls within the desired range at all times</p> <p>b. For a frequency range test, we will use a spectrum analyzer and</p>	C

<ul style="list-style-type: none"> ● Must transmit data to and from kitchen transceiver and microcontroller at 9600 bps 	<p>run a frequency sweep with 1 MHz step size to find the frequency range (This must be between 2.4-2.5 GHz)</p> <p>c. For distance testing, we will manually guess and check starting from 100 meters and observe for accuracy on our computer display (i.e. submit a selection and observe the response from the GUI)</p> <p>d. Use X-CTU to set the baud rate and create a test program to have a message from the menu XBEE connected to the Arduino transmitted to the kitchen transceiver connected to the dongle. Message will be displayed on the computer screen and must have 0.1% bit-error.</p>	
<p>5. Menu Display:</p> <ul style="list-style-type: none"> ● Must be able to properly display messages that were read from the NFC Reader 	<p>5.</p> <p>a. Will be given a test message from the microcontroller (typed from Arduino's IDE) and spot for 0.1% bit-error</p>	C
<p>6. Menu Microcontroller:</p> <ul style="list-style-type: none"> ● Must operate from a 9V battery power supply (7V-12V safety range). ● Must output data to LCD display correctly. ● Must receive correct data from NFC reader. ● Must transmit correct data to menu transceiver. ● Must be able to hold 2560 bits of memory for menu items. ● Must be able to transmit and receive data from transceiver at 9600bps. ● Must be able to regulate voltage from the power supply and provide other components with power. 	<p>6.</p> <p>a. While the microcontroller is powered, we can use a DMM to monitor the voltage across and the current drawn from the battery</p> <p>b. Test code will be developed (using provided library's from Arduino) to interface between the LCD and microcontroller which will properly display the messages read from the NFC reader (which can be tested with a TagWriter as mentioned in #2 Verif.)</p> <p>c. Using TagWriter to transmit a message to the NFC reader and to</p>	C

	<p>the microcontroller, we will use Arduino's IDE to view the data and spot for 0.1% bit-error</p> <p>d. Test code will be written to display (or TagWriter will be used) the data that the microcontroller will send to the transceiver while also spotting for 0.1% bit-error</p> <p>e. Dummy memory will be written to the first 3 MB of EEPROM, and then read to an output file. Input/output test files will be Vimdiff'd to prove that they are identical</p> <p>f. Create test code to send message to the kitchen transceiver connected to a computer which will be set at 9600 baud rate. Check the test message being sent using X-CTU</p> <p>g. We will use a DMM to probe voltage outputs from the power supply and the feeds</p>	
<p>7. Receiver RF Transceiver:</p> <ul style="list-style-type: none"> ● Must be supplied with 2.8-3.4V ● Must operate at ISM (industrial, scientific, and medical radio band) 2.4 GHz frequency band. ● Must work at a range of approximately 100 meters. ● Must transmit data to and from kitchen transceiver and microcontroller at 9600 bps 	<p>7.</p> <p>a. We will use a DMM to probe the regulated voltage from the dongle</p> <p>b. For a frequency range test, we will use a spectrum analyzer and run a frequency sweep with 1 MHz step size to find the frequency range. This must be between 2.4-2.5 GHz</p> <p>c. For distance testing, we will manually guess and check starting from 100 meters and observe for accuracy on our computer display (i.e. submit a selection and observe the response from the GUI)</p>	<p>C</p>

	d. Use X-CTU to set the baud rate and create a test program to have a message from the kitchen XBEE connected to the Arduino transmitted to the menu transceiver connected to the dongle. Message will be displayed on the computer screen and must have 0.1% bit-error.	
8. Dongle <ul style="list-style-type: none"> • Must regulate voltage of 5V from USB to 2.8-3.4V for the transceiver • Must allow for transfer of data from kitchen transceiver to USB 	8. <p>a. Probe the VCC pin (pin1) of the dongle with a DMM to verify that the voltage supplied is within the correct range</p> <p>b. Also create test code on computer display to verify the contents from the transceiver are being transmitted to computer with 0.1% bit-error</p>	C
9. Receiver Computer Display (software) <ul style="list-style-type: none"> • Must be able to parse the signal and display the appropriate messages from the kitchen transceiver 	9. <p>a. Software will test corner cases/simultaneous entry and show the data on the computer screen</p>	C
10. Menu Battery Life Indicator: <ul style="list-style-type: none"> • Accurate LED Indication (Red LED lit below 6.9V Vcc) • Low power consumption such that it does not interrupt the battery's ability to power a load 	10. <p>a. Use a DC power supply and set the input voltage to below and above the 7V threshold</p> <p>b. Use a DMM on a 9V power source to observe the power consumed from a matched load (replicated with resistors)</p>	C

7.2 Arduino Code

```
// This is the root program to run for NFC to work
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <stream>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include "nfc.h"
#include "Wire.h"

#define I2C_ADDR      0x3F //I2C Address
#define BACKLIGHT_PIN 3
#define En_pin       2
#define Rw_pin       1
#define Rs_pin       0
#define D4_pin       4
#define D5_pin       5
#define D6_pin       6
#define D7_pin       7

NFC_Module nfc;

byte incomingByte;
int cartNum = 1; // initialize cart position
int orderNum = 1;
char menuNum = 'B';
int myCart[64];
boolean resetCartFlag = true;

LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

void setup()
{
  clearCart();

  Serial.begin(9600);

  nfc.begin();

  lcd.begin (20,4,LCD_5x8DOTS);
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE); //initialize backlight

  lcd.clear();
  lcd.setBacklight(HIGH);
  nfc.SAMConfiguration();

  // Welcome screen
  printCenter(10,1);
  lcd.print("Welcome to");
  printCenter(17,2);
```

```

lcd.print("MIKE'S MILKSHAKES");
delay(1500);
wipeLines();

}

void loop()
{

wipeLines();

if (resetCartFlag == true)
{
clearCart();
resetCartFlag = false;
}

printCenter(10,0);
lcd.print("May I take");
printCenter(11,1);
lcd.print("your order?");
printCenter(15,2);
lcd.print("Please select a");
printCenter(10,3);
lcd.print("menu item.");

//
// If new data is available, then load item information
//

//while (Serial.available() == 0) {}//Wait for keyboard input

incomingByte = pollNFC();

wipeLines();
delay(500);

byte MenuID = incomingByte;

if ((incomingByte > 0) && (incomingByte < 255)) // If data falls within spec
{
if (incomingByte == 254)
{
checkoutQuestion();
checkout();
return;
}

printCenter(16,0);
lcd.print("You have chosen:");
displayItem(incomingByte, 1);

//Ask for Confirmation

```

```

printCenter(12,3);
lcd.print("ADD TO CART?");

delay(500);

boolean YesNo = yesOrNo();
    if (YesNo == 1)
        //User wants to add to cart
        {
            wipeLines();

            addToCart(MenuID, cartNum);

            printCenter(17,1);
            lcd.print("Item # ");
            lcd.print(MenuID);
            lcd.print(" has been");
            printCenter(13,2);
            lcd.print("added to cart");
            delay(1000);
        }

    else
        //User does not want to add to cart
        {
            wipeLines();
            return; //Go back and ask for new swipe
        }
}
else
{
    printCenter(14,1);
    lcd.print("Invalid Entry");
    delay(1000);
    return; // go back to top of loop() function.
}
}

void checkout()
{
    boolean YesNo = yesOrNo();

    if (YesNo == 1)
    {
        myCart[0] = cartNum;
        wipeLines();
        printCenter(20,0);
        lcd.print("Items in your cart: ");
        delay(500);

        printCenter(20,1);
        int count = 0;
    }
}

```

```

//   for (int z = 0; z <= myCart[0]/4; z++)
//   {
for (int j = 1; j <= myCart[0]; j++)
{
    displayItem(myCart[j], j%4);
    delay(300);
    count++;
    if (count == 3)
    {
        count = -1;
        delay(2000);
        wipeLines();
    }
}

    delay(2000);
    wipeLines();
}

else
{
    wipeLines();
    return;
}

    wipeLines();
    printCenter(16,0);
    lcd.print("Are you sure you");
    printCenter(14,1);
    lcd.print("want to submit");
    printCenter(11,2);
    lcd.print("your order?");
    printCenter(20,3);
    lcd.print("NO WILL CANCEL ORDER");

    YesNo = yesOrNo();

    if (YesNo == 1)
    {
        sendOrder();
    }
    else
    {
        resetCartFlag = true;
        wipeLines();
        printCenter(14,1);
        lcd.print("Order has been");
        printCenter(10,2);
        lcd.print("CANCELLED!");
        delay(1000);
        return;
    }
}

```

```

void checkoutQuestion()
{
    printCenter(9,0);
    lcd.print("CHECKOUT?");
    printCenter(18,1);
    lcd.print("NO will allow more");
    printCenter(18,2);
    lcd.print("items to be placed");
    printCenter(13,3);
    lcd.print("into the cart");
}

//
//
//

//
// Begin Helper Functions
//

//
// Function to print item on the LCD
//

void displayItem(int newByte, int row)
{
    String stringPtr;
    switch (newByte)
    {
        case 1: stringPtr = "Grasshopper";
            break;
        case 2: stringPtr = "Tin Roof";
            break;
        case 3: stringPtr = "Oatmeal Cookie";
            break;
        case 4: stringPtr = "Strawberry Shortcake";
            break;
        case 5: stringPtr = "Guinness";
            break;
    }
    int strLength = stringPtr.length();
    printCenter (strLength, row);
    lcd.print(stringPtr);
}

//
// Function to +1 item to cart
// x = Item #; i = cart position

void addToCart(int x, int i)
{
    myCart[i] = x;
}

```

```

    cartNum = cartNum++;
}

//
// Helper function to figure out if user swiped YES or NO tag
//

boolean yesOrNo()
{
    byte newByte = pollNFC();
    while (newByte != 255 && newByte != 253)
    {
        newByte = pollNFC();
    }

    if (newByte == 255)
    {
        return true;
    }
    else if (newByte == 253)
    {
        return false;
    }
}

//
// Function to empty contents of cart
//

void clearCart()
{
    for (int i = 0; i < 64; i++)
    {
        myCart[i] = 0;
    }
    cartNum = 1;
}

void wipeLines()
{
    for (int y = 0; y < 4; y++)
    {
        for (int x = 0; x < 20; x++)
        {
            lcd.setCursor (x,y);
            lcd.print(" ");
            delay(5);
        }
    }
}

// Send order to Kitchen

```

```

void sendOrder()
{
    wipeLines();
    printCenter(10,1);
    lcd.print("Sending...");
    sendRF(myCart);
    delay(1000);
    printCenter(15,2);
    lcd.print("Order Received!");
        orderNum++;
    resetCartFlag = true;
    delay(1000);
}
// Helper for sendOrder.
// Sends contents of cart to kitchen.

void sendRF(int* myCart)
{
    Serial.println("*****");
    Serial.println("*****BEGIN ORDER*****");
    Serial.println("*****");

    Serial.print("Menu ");
    Serial.print(menuNum);
    Serial.print(":");
    Serial.println();

    Serial.print("Order #");
    Serial.print(orderNum);
    Serial.print(":");
    Serial.println();

    for (int i = 1; i<myCart[0]; i++)
    {
        Serial.print("Item ");
        Serial.print(i);
        Serial.print(": ");
        Serial.print(myCart[i]);
        Serial.println();
    }

    Serial.println("*****");
    Serial.println("*****END OF ORDER*****");
    Serial.println("*****");
    Serial.println();
}

// Function to poll NFC and read next block

u8 pollNFC()
{

```

```

u8 sta, buf[32];

sta = nfc.InListPassiveTarget(buf);

//if (sta && buf[0] == 4)
while (sta == 0)
{
    sta = nfc.InListPassiveTarget(buf);
    if (sta == 1)
        break;
}

u8 key[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

sta = nfc.MifareAuthentication(0,4,buf+1,buf[0],key);
if (sta)
{
    u8 block[16];
    sta = nfc.MifareReadBlock(4,block);
    if (sta)
    {
        return block[0];
    }
}

//
// Function to put cursor in center
// x = stringLength, y = row

void printCenter(int x, int y)
{
    switch (x)
    {
    case 1:
        lcd.setCursor(10, y);
        return;
    case 2:
        lcd.setCursor(9, y);
        return;
    case 3:
        lcd.setCursor(9, y);
        return;
    case 4:
        lcd.setCursor(8, y);
        return;
    case 5:
        lcd.setCursor(8,y);
        return;
    case 6:
        lcd.setCursor(7, y);
        return;
    case 7:

```

```
    lcd.setCursor(7, y);
    return;
case 8:
    lcd.setCursor(6, y);
    return;
case 9:
    lcd.setCursor(6, y);
    return;
case 10:
    lcd.setCursor(5, y);
    return;
case 11:
    lcd.setCursor(5, y);
    return;
case 12:
    lcd.setCursor(4, y);
    return;
case 13:
    lcd.setCursor(4, y);
    return;
case 14:
    lcd.setCursor(3, y);
    return;
case 15:
    lcd.setCursor(3, y);
    return;
case 16:
    lcd.setCursor(2, y);
    return;
case 17:
    lcd.setCursor(2, y);
    return;
case 18:
    lcd.setCursor(1, y);
    return;
case 19:
    lcd.setCursor(1, y);
    return;
case 20:
    lcd.setCursor(0, y);
    return;
}
}
```