[[1]](#footnote-1)

Digiceipt

Tyler Adams, CSE; Enoc Flores, EE; George Papageorgiou, EE; Edward Townsend, EE

*Abstract*—At the end of every transaction the customer has the option of receiving a receipt. What we’ve done in our project is give the customer the option of receiving a digital receipt on to their Android phone. Digiceipt consists of a computer, receiving data from a barcode scanner, which then sends digital receipts to our Android app through an NFC transceiver. The Android app has cloud storage capabilities that allow users to save their receipts wirelessly without ever having to worry about losing their digital receipts. This new option allows customers to organize all their transactions and help the environment by decreasing the usage of paper and ink.

*Index Terms*: GPIO, I2C, NDEF, NFC, SPI, UART

# INTRODUCTION

E

very day, millions of items are purchased; the result: millions of receipts. With all the receipts being produced every day enormous amounts of resources are being consumed as well as an enormous amount of waste being produced. A study done by Clinton Global Initiative University estimates that, 250 million gallons of oil, 10 million trees, and 1 billion gallons of water are consumed each year for the production of receipts along with 1.5 billion pounds of waste being produced. With today’s economic crisis, resources need to be saved in any way that they can. The study also shows that each receipt that is printed can cost about 1 cent. With each receipt costing about 1 cent that means that huge corporations can lose large amounts of money annually every year for an outdated system that can be improved. [1]

Another issue with receipts is that they can be a hassle to deal with from the customer's point of view.  Receipts can take up a lot of room wherever they are being kept so having a system where they are condensed would help people who like to keep all their finances well organized. There is also a health concern regarding some types of receipts. Thermal receipts, which are used by superheating the receipt to get the information to show rather than using ink, have BPA/BPS chemicals in them which are linked to causing cancer. [2] Thermal receipts also fade over time so it is almost impossible to keep them for personal records for long amounts of time. Technology is constantly being updated and upgraded to help be more energy and economically efficient as well as safer for the general public. Some examples would be the invention of electric cars, advances in airbag technology, solar panels to produce energy, and the introduction of fluorescent and LED
light bulbs. With all kinds of technology being updated to save resources, energy, and lives it’s only obvious that it was time to update our outdated and wasteful receipt system.

A solution to the inefficient paper receipt system has already been attempted with the advent of email receipts. Digital receipts are sent to a customer’s email after a transaction instead of having the register print one out at the point of sale. This allows customers to keep their digital receipts as long as they need to. Unfortunately, this solution relies on the email’s system structure which is limited to keeping track of email messages. This structure lacks the advanced consolidation and financial tracking tools needed for managing one’s finances. The Digiceipt platform offers the convenience of the digital receipts while providing the tools necessary to simplify the management of one’s finances.

# Design

|  |  |
| --- | --- |
| Dimensions | 95mm x 127.7mm x 25mm |
| Wireless Transfer Protocol | NFC |
| NFC Range | 2cm |
| NFC Data Exchange (terminal to phone) | NDEF |
| Receipt Transfer Time | <1s |
| Weight | 1lb, or ~7lbs with optional battery pack |
| Ports | Ethernet, 4 USB 2.0, Power |
| Display | LCD Screen |

## Overview

**Table 1: Specification Table**

Our block diagram that was created for our project can be seen in *Figure 1*. We were able to break down our project into seven different blocks. The first block is the inputs which include the barcode scanner and the product data that it receives. The center block is the computer which deals with the four interfaces: the USB barcode scanner, the LCD monitor, the NFC transceiver, and the power source. The computer is hooked up to the LCD display monitor and NFC transceiver creating one compact system that converts information into receipts that the NFC transceiver can write on to Android devices. Finally, the mobile phone will incorporate the app and cloud storage that brings everything together by storing and receiving digital receipts.

The purpose of the barcode scanner is to send product data to the computer through a USB connection. The function of the barcode scanner will not change from the traditional barcode scanner that one sees at stores. The data that is received from the barcode scanner is looked up instantly on an online UPC database which then sends the looked-up information to the computer including: the name of the item being purchased, the establishment where the goods are being sold, the amount, and the date and time of the transaction.

All the information received from the online UPC database is converted into a digital receipt once it is sent to the computer. A Raspberry Pi 2 model B is what we are using for our computer. The purpose of the Raspberry Pi B is to build the receipt and send it over to the NFC transceiver in a form that it can be written on to our Android app. The computer is connected to the NFC transceiver via UART. The display monitor is connected to the computer via SPI connection. The display monitor displays a GUI so all the information being read in can clearly be seen by the user.

Finally, the Android phone has the official Digiceipt app with cloud storage capabilities that stores and sorts receipts. With the Android app, users are able to easily navigate and view previously saved receipts or receive new receipts. The receipts are organized by date and can easily be searched. In addition to the app, the cloud storage feature assures that data is never lost even when phones are lost or broken. The app then, by default, will have user accounts that will allow users to access their receipts from any phone through the cloud.



**Figure 1: Block Diagram**

## Inputs

The goal of Digiceipt is to integrate into today’s checkout systems and be the standard for all future checkouts. This block is where all the barcode data is scanned and sent to the Raspberry Pi. This block also acts as a mock point of sale. In other words, we are mimicking the process of grocery items having their barcodes scanned and having their information being transferred to the rest of the system.

Much of this is realized through the interfacing of a SOLUX USB barcode scanner to a Raspberry Pi B. [3] This specific barcode scanner was chosen due to its ability to read UPC barcodes, which are the universal standard for grocery items in the United States, and its compatibility with Linux, which is the operating system utilized by the Raspberry Pi B board. The Raspberry Pi B interprets the barcode scanner as another keyboard. When a barcode is read, it’s similar to as if someone were to type an input. This similarity was exploited with a Python script that read in the data being transmitted to the Raspberry Pi B. These inputs are then saved in a local variable which is used for the NFC transmission. In order to verify the information being scanned and stored in the Raspberry Pi is accurate, there is a display monitor showing exactly what is being stored. The information displayed can then be compared with the information on the barcode that was scanned to verify proper operation.

## Computer

The computer for Digiceipt is a Raspberry Pi 2 Model B computer. The Raspberry Pi was chosen because it was a system that can be programmed efficiently to interface with multiple inputs and outputs. Apart from being able to interface with inputs and outputs the Raspberry Pi is also small and powerful enough for what we are were trying to accomplish. The Raspberry Pi’s dimensions are 85x56x17mm and it has a quad-core ARM Cortex-A7 processor that lets the CPU run at 900MHz to go along with 1GB of RAM. To get the Raspberry Pi operational, the NOOBS Raspian OS, which is the standard operating system used for Raspberry Pis, was downloaded onto a 16GB micro SD card. The Raspberry Pi was then connected to a display monitor through a HDMI to DVI cable along with having a keyboard and mouse attached via USB. It was then powered by standard 5V micro USB cable through a wall outlet. The installation of the Raspian operating system was then carried out by the Raspberry Pi which was observed by the display monitor. [4]

Once the Raspberry Pi was setup and running with the operating system it was connected via Ethernet cable to the internet through the on board Ethernet port on the Raspberry Pi. From there commands were run in the terminal to update and upgrade all the standard packages. The updates and upgrades were essential because they were used in the NFC and Linux libraries later on. At this point extensive research was put into which connection type: SPI, UART, or I2C would benefit the interface between the Raspberry Pi and the PN532 NFC board the best. The conclusion was that the UART connection was the best fit because the PN532 NFC board had a simple input/output pin interface. [5] The Raspberry Pi by default has its general input/output pins dedicated to SPI connections so the kernel had to be changed to free up UART for use with the libnfc library. To free up the UART a simple configure command was run in the terminal which led to a system screen where the option to change the serial setting to disable shell and kernel messages via UART. After the kernel changes were in place, the board was rebooted to implement the changes. [6]

The next step in the process was the installation of the libnfc standard open source library. This library was necessary for the communication between the PN532 NFC board and the Raspberry Pi. First the tar.gz file containing the libnfc version 1.7.0 library was downloaded onto the Raspberry Pi and extracted and placed in a created libnfc directory. A file that is associated with the PN532 NFC board located in the libnfc library had to be modified to allow serial port probing between the two devices. Following a configuration of the libnfc files, the library was able to be utilized. The library and its components were installed onto the Raspberry Pi. To check that everything was installed correctly the PN532 NFC board was connected to the Raspberry Pi with jumper cables via the UART connection interfaces between the two. A command was then run in the terminal to read tags that were embedded into NFC cards that came with the PN532 NFC board. The tags were read correctly proving that the installation had been completed correctly.

The last step to setting up the Raspberry Pi was to install the nfcpy library. The nfcpy library is a collection of functions and data that allows python code to be run on the Raspberry Pi board that is connected to an NFC device. The first step to installing the nfcpy library was to install a bzr client and then using a branch with the bzr to download and install nfcpy. The Raspberry Pi was then rebooted to complete the installation. To verify the installation a command to read was used and when the NFC card was placed over the PN532 NFC board its tag ID was read and displayed in the terminal.

Pictured below is a snapshot of an NFC tag being read, verifying that nfcpy was properly installed.

**Figure 2: Screengrab of NFC tag being read.**

 The Python code on the Raspberry Pi board was updated to run a Graphical User Interface (GUI) on the LCD Display alongside the NFC device code to create a more appealing user experience. This code unfortunately, is only capable of transferring on receipt before the program needs to reset. The program also needs to be run “with the Terminal” since the terminal window needs to be selected for the GUI to function properly.

## Display

The display serves as a substitute for the computer monitor that was originally being used to access and initiate commands on the Raspberry Pi B board during initial setup and debugging. The display we chose was a 480x320 3.5’’ touchscreen liquid crystal display (LCD) screen also known as the PiTFT.[7] The addition of this hardware component brought the whole system together. With the mini LCD monitor implemented the Raspberry Pi B board was no longer restricted to a bulky computer monitor. The touch screen capabilities of the PiTFT made it possible to completely eliminate the keyboard and mouse from the system. A significant part of our Senior Design Project to make it transportable and easy to use; therefore, capitalizing on the fact that this system is extremely efficient not only in regards to the environment, economy, but in its availability to users will become more appealing to vendors and customers. The feature of transportation is exactly what the PiTFT allowed our system to do. One of the main functions that the PiTFT screen did for us was that it allowed us to run our code straight from the board. The implementation of the PiTFT on to the Raspberry Pi B board is straightforward; it is connected directly through the general-purpose input/output also known as the GPIO.

The GUI was an essential part of our project because it allows the user to interact with the whole system, and it brings every component of the project together. The GUI, a separate screen from the main terminal, runs once the program starts. The GUI screen displays the items that are being scanned (e.g. potato chips, brownies etc.), the price of each item, and the total cash amount. The phone can then tap the NFC Transceiver once those items are displayed on the GUI. After the phone has received the digital receipt the contents on the GUI are erased. The GUI brings every component of the project together in the way that it gives the user live feedback that can be understood. Items being displayed on the GUI, from the display on the Raspberry PI, are coming from the scanner, those barcodes are being fetched from a UPC database via WiFi, the prices are coming from the program, and finally the mobile phone will reset the contents on the GUI once they are transferred over to the app via NFC; therefore, the GUI is incorporating all components of Digiceipt.

## Power Supply

 The main component of my power supply consisted of a switching voltage regulator circuit that outputs 5V and 3A at any input voltage above 5V. This is the chosen output of the circuit because it is what is needed to power the rest of our system. To give power to the power supply, we used 16 D cell NiMH batteries in series. Each of these batteries are rated for 1.2V and have an exceptionally large energy charge of 10,000mAh. This is need to ensure the voltage regulator can be powered for at least 9 hours without the batteries having to be recharged.

 The circuit diagram for the voltage regulator is shown below in Figure 3. The circuit is made using a LM2576HV voltage regulator chip. Connected to the input of the chip is the input voltage and a 100μF capacitor while the output of the chip is connected to a 1N5822 Schottky Diode, 100μH inductor, and a 1000μF capacitor as well as the Raspberry Pi system. [8]



**Figure 3: Switching Voltage Regulator Circuit Diagram**

The power supply system was encased in a 3D printed case with dimensions of 5”x8”x3”. This allowed the power supply to have a more aesthetic look to it as well as keep the circuitry safe.

## NFC Transceiver

The NFC transceiver consists of a PN532 breakout board. This board was chosen due to the ample documentation available for it as well as the ability to easily interface it with a Raspberry Pi board. The initial steps for setting up the PN532 involved soldering the selector and UART pins into their respective ports. To enable UART on the PN532, and make sure SPI and I2C weren’t being enabled, jumpers were placed on the selector pins, setting both of them to the off position. An off position for both selector pins correlates to a UART interface. Next four jumper wires were used to connect the PN532 to the Raspberry Pi board. A wire connecting the 5V powers as well as the grounds were connected, as well as jumper wires connecting the TXD and RXD to their respective pins. The TXD and RXD pins are used specifically transmitting and receiving data respectively for serial interfaces. This connection was verified by running read and write commands as stated earlier in the computer subsection.

Getting the PN532 to transmit data to the mobile phone involved writing a python script to create and send NDEF messages. The script works by taking the local variable holding the barcode information on the Raspberry Pi and sends that information via TXD to the PN532. This information is then formatted into a readable NDEF format. The data is stored there until a mobile phone attempting to read NDEF messages is placed over the antenna. From here, the NDEF message is transmitted using NFC to the mobile phone to be stored and displayed by the Android app. [8]

To verify that this subsystem is completely working, you can write the NDEF message containing the barcode information onto a writable NDEF tag. This tag can later be read, and its information can be compared to that of the expected barcode data. If they are identical, the PN532 is working properly. Figure 4.1 displays the results we received when testing the transmission range of our NFC board. We ran these tests for both consistency and reliability, with each range measurement in the Trials signifying the range at which data is reliably transferred.

**Figure 4.1: NFC Range and Reliability**

Figure 4.2 displays the command line execution of the NFC transfer system.

**Figure 4.2.: Code & Execution of Data Transfer**

## Mobile Android App

The mobile application portion of our system is used to store and retrieve the digital receipts transmitted by the NFC chip. The application uses the Android platform due to the larger availability of phones with NFC chips in comparison to other platforms such as iOS and Windows. The Android platform has also supported NFC technology within its system (for Android 2.2+, which supports over 85% of phones) while the iOS platform only supports phones past their iPhone 6 models. The mobile application also uses cloud storage for account management as well as data backups for the user's data.

 In order to build this mobile application, we developed an environment which supports Android as well as a method in which to test and run our application. We learned how to configure our devices to support Android debugging and/or run an Android device emulator to run our code. We also learned the framework under which Android code operates since we have already learned Java (the language on which Android is run on) from our previous courses. In addition to the Android framework, we familiarized ourselves with the NFC library included in the Android development kit. [10]

 Of course, to test the development of the mobile application, we ran it on our physical Android devices. In order to verify the working code within the Android environment, we programmed a read/write mode toggle. This toggle told the application whether it should be configured to receive NFC messages or send them based on the automatic triggers of the 2 phone's NFC chips which detected when they were in proximity with another chip. We also tested the mobile application's functionality with the NFC chip connected to the Raspberry Pi B board in order to ensure the interface between the 2 devices is working correctly. We verified the results of these 2 tests by displaying the data transmitted to the receiving device within the mobile application itself. Figure 5 exemplifies the transfer of the barcode data from the NFC chip to the Android application.



**Figure 5.1: The barcode data is displayed on the mobile application**

After communication between the app and the Raspberry Pi board was verified, the user experience became the focus for our updates. A login interface was developed alongside a login backend structure allowing people using the app to have accounts attached to the receipts they received at the point-of-sale. We also created a more refined view for grouping receipts on the main screen and a single-receipt view which is IRS compatible.



**Figure 5: The single-receipt view, which contains all the data necessary for IRS verification**

## Cloud Server

The Cloud server portion of our system is primarily used to store backups of digital receipts received by users. The server is also used to authenticate and manage accounts for our users. After searching through a number of Cloud storage services, we settled on using the Parse service due to its ease of integration and low cost. The Parse SDK includes an Android development kit which has allowed us to interface with the server using native Android code.

# Project Management

Each member of Digiceipt played an important role in its development. Tyler and Enoc, having expertise in Android development, were mainly responsible for creating the mobile app and making sure it could read in NDEF messages. They were also responsible for creating the GUI on the LCD monitor that allowed the customer to see what information is being processed. George and Edward, who had expertise with microcontrollers as well as the Linux operating system, worked mainly with the Raspberry Pi and power source. Their job was to make sure the Raspberry Pi was able to take in accurate data from the USB barcode scanner and forward it to the antenna of the PN532 NFC board. From here they needed to make sure that the data was transmitted in an NDEF format so that it could be read by the mobile app. For the power source, they needed to design a system that would output 5V and 3A for any given input voltage. This power source must also be able to run for at least 9 hours without pause. Although everyone had primary parts they were working on, everyone worked together and helped each other out. This helped with debugging as well as ensuring the different subsystems would be able to communicate with each other. Weekly meetings were held with Professor Soules to talk about where we are in the process and what we still needed to do.

Pictured below is a Gantt Chart of our progress over the past semester as well as what we project to have done for CDR. In purple is what we did as a team, green is work done by Edward and George, and red is work done by Tyler and Enoc.

|  |  |
| --- | --- |
| Deliverable | Completion |
| Ability to run for 9+ hours without fail | 100% Completed |
| Data transfer only works within 3cm | 100% Completed |
| Receipts have no size limit | 100% Completed |
| Receipts are searchable within app | 100% Completed |
| User can create their own account on Android app | 100% Completed |
| GUI displays barcode information and prices | 100% Completed |
| Receipts are IRS compatible | 100% Completed |

**Table 2: System Deliverables**

# Conclusion

Digiceipt has provided a full product presentation given the results of our FPR initiative and Demo Day demonstration. The proof of concept that each independent component of the system can interface with another allowed us to complete the end-to-end data transfer of our digital receipts. By being able to test and verify each independent component individually, we were able to verify our individual implementations worked before joining each one together.

From CDR, we developed our product on the skeleton of the current system we had. By improving the efficiency of NFC transfers from the Raspberry Pi, improving the usefulness and presentation of our mobile application, and integrating the system we’ve developed with current point-of-sale systems, we created a product which will simplify the task of tracking financial history for our users.

As we developed this product, we ensured the reliability and security of our system. The transfer process is quick (< 2 seconds for receipt transfers) and usable by our users. This meant making sure the load on our hardware is small and the mobile application has a simple-to-use interface. The communication we had as teammates and the feedback we got on our design were key tools in making sure we met both our team’s goals and real world expectations.

# Appendix

## Acknowledgment

We would like to thank our advisor, Baird Soules, and our advisors, Do-Hoon Kwon and David Irwin, for their invaluable advice and support.

References and Footnotes

[1] W. Hines, “Going Paperless: The Hidden Cost of a Receipt,” *The Huffington Post*. [Online]. Available at: http://www.huffingtonpost.com/will-hines/going-paperless-the-hidde\_b\_3008587.html. [Accessed: 25-Jan-2016].

[2] “Plastics chemicals BPA, BPS linked to altered brain development,”*NaturalNews*. [Online]. Available at: http://www.naturalnews.com/052634\_bisphenol\_s\_brain\_development\_plastics\_chemicals.html. [Accessed: 25-Jan-2016].

[3] S. O. L. U. X., “Solux Barcode Scanner Manual,” Solux Barcode Scanner Manual. Solux. [Accessed: 25-Jan-2016].

[4] “Raspberry Pi 2 Model B Specifications,” Raspberry Pi Raspberry Pi 2 Model B Comments. [Online]. Available at: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Accessed: 2015].

[5] “PN532/C1 NFC controller,” *PN532/C1 NFC controller*, 31-Mar-2011. [Online]. Available at: https://www.adafruit.com/datasheets/pn532ds.pdf. [Accessed: 25-Jan-2016].

[6] K. Townsend, “Adafruit NFC/RFID on Raspberry Pi,” Overview.
[Online]. Available at: https://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi?view=all. [Accessed: 2015].

 [7] “PITFT - ASSEMBLED 320X240 2.8’ TFT TOUCHSCREEN FOR RASPBERRY PI.” [Online]. Available at: https://www.adafruit.com/products/1601. [Accessed: 2015].

[8] "LM2576/LM2576HV Series SIMPLE SWITCHER ® 3A Step-Down Voltage Regulator", *Texas Instruments*, 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm2576hv.pdf. [Accessed: 03- Feb- 2016].

[9] “Near Field Communication,” *Near Field Communication*. [Online].
 Available at: http://developer.android.com/guide/topics/connectivity
 /nfc/index.html. [Accessed: Feb-2015].

[10] “IDC: Smartphone OS Market Share,” www.idc.com. [Online]. Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 2015].

[11] “Getting Started,” Getting Started. [Online]. Available at:
http://developer.android.com/training/index.html. [Accessed: Apr-2015].

1. T. A. Author from Brookfield, CT (tsadams@umass.edu).

E. F. Author, from El Paso, TX (flores@umass.edu).

G. P. Author from Dracut, MA (gpapageo@umass.edu).

E. T. Author from Falmouth, ME (ejtownse@umass.edu). [↑](#footnote-ref-1)