

# Digicept Midterm SDP Report

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 plan to do in our project is give the customer the option of receiving a digital receipt on to their Android phone. Digicept will consist of a computer, receiving data from a barcode scanner, which sends digital receipts to an Android app through an NFC transceiver. The Android app will have cloud storage capabilities that will allow users to never lose their digital receipts. This new option will allow 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

## I. INTRODUCTION

Every day, millions of items are purchased; the result: millions of receipts. With all the receipts being produced every day an enormous amount 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costing about 1 cent that means that huge corporations can lose large amounts of money annually every year for an outdated system that can be improved.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 are 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's 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 Digicept platform offers the convenience of the digital receipts while providing the tools necessary to simplify the management of ones finances.

## II. DESIGN

### A. Overview

Target Dimensions	95mm x 127.7mm x 25mm
Wireless Transfer Protocol	NFC
NFC Range	4cm-6cm
NFC Data Exchange (terminal to phone)	NDEF
Receipt Transfer Time	2s
Weight	<3lbs
Ports	Ethernet
	USB 2.0
Display	LCD Screen

**Table 1: Specification Table**

A block diagram was created in order to fully understand how our project is going to create digital receipts and can be seen in *Figure 1*. We were able to break down our project into six different blocks. The first block is the barcode scanner that will receive product data. Secondly, a computer will be hooked up to a display 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 that will bring everything together by storing and receiving digital receipts.

The main 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. Information that will

T. A. Author from Brookfield, CT ([tsadams@umass.edu](mailto:tsadams@umass.edu)).

E. F. Author, from El Paso, TX ([flores@umass.edu](mailto:flores@umass.edu)).

G. P. Author from Dracut, MA ([gpapageo@umass.edu](mailto:gpapageo@umass.edu)).

E. T. Author from Falmouth, ME ([eitownse@umass.edu](mailto:eitownse@umass.edu)).

come from the barcode will consist of the name of the establishment selling goods or providing a service, the amount, the date, and time of transaction.

The information from the scanner will be converted into a digital receipt once it is sent to the Raspberry Pi B. The Raspberry Pi B will be what we are using for our computer. The main 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 will be connected to the NFC transceiver via UART. The display monitor will directly connect to the computer via GPIO. The display monitor will inform the user when to begin scanning items, when the information is being processed by the computer, once the mobile phone is ready to receive the receipt, and finally when the receipt was successfully sent to the phone.

Finally, the Android phone will have an app with cloud storage capabilities that will store and sort receipts. The Android app will be easy to navigate through previously saved receipts and receive receipts. These receipts will be organized and encrypted to protect users and vendors. In addition, the app will have cloud storage features to assure 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.

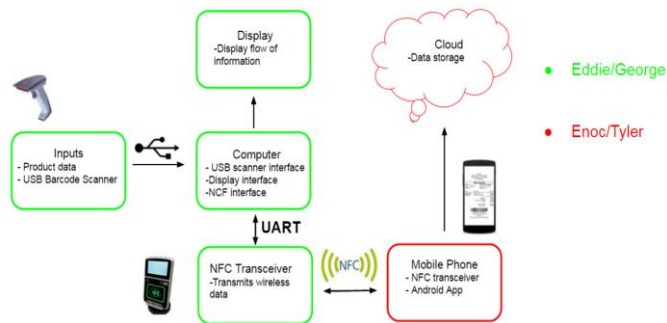


Figure 1: MDR Block Diagram

### B. Inputs

The goal of Digiceipt is to integrate into today's checkout systems and be the standard for all future checkouts. The point of this block is to be a mock point-of-sale system. 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.

### C. 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 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]

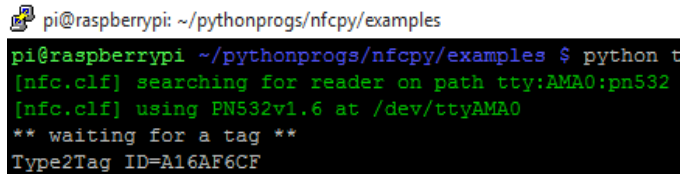
Once the Raspberry Pi was setup and running with the operating system it was connected via Ethernet cable to the internet through the on board Ethernet port on the Raspberry Pi. From there commands were run in the terminal to update and upgrade all the standard packages. The updates and upgrades were 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 is 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 bazaar client and then using a branch with the bazaar 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.



```

pi@raspberrypi: ~/pythonprogs/nfcpy/examples
pi@raspberrypi ~/pythonprogs/nfcpy/examples $ python t
[nfc.cif] searching for reader on path tty:AMA0:pn532
[nfc.cif] using PN532v1.6 at /dev/ttyAMA0
** waiting for a tag **
Type2Tag ID=A16AF6CF
  
```

Figure 2: Screenshot of NFC tag being read.

#### D. Display

The display will serve as a substitute for the screen monitor that is currently being used to access and initiate commands on the Raspberry Pi B board. Currently, we have an Assembled 320x240 2.8" Liquid Crystal Display (LCD) Screen also known as the PiTFT with touch screen capabilities which utilizes thin-film transistors. [7] The addition of this hardware component will bring the whole system together. The Raspberry Pi B board will not be restricted to a screen monitor that is hooked up to a mouse and a keyboard after the LCD Screen is implemented. The touch screen capabilities of the PiTFT will completely eliminate the keyboard and mouse. A significant part of our Senior Design Project is 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 allow our whole system to do. The implementation of the PiTFT on to the Raspberry PI B board is straight-forward. The PiTFT is directly connected through the general-purpose input/output also known as the GPIO.

One of the main functions that the PiTFT screen will do is allow us to run our code on the board. As well, the LCD Screen will guide and inform the user of its functions. After the code has been run on the board the PiTFT will inform the user when the scanner is ready to scan items. Then, after items have been scanned the LCD screen will notify the user that the digital receipt is being created in the Raspberry Pi B board and shortly being sent to the NFC transceiver. After the NFC transceiver has received the digital receipt it will update its information stating that the NFC transceiver is ready to engage the phone to send the receipt over to the Android app. Once the receipt is successfully sent to the mobile phone the PiTFT will update appropriately.

#### E. 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 involves writing a python script to create and send an NDEF message. Taking the local variable holding the barcode information on the Raspberry Pi, the script then send this 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 3.1 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.

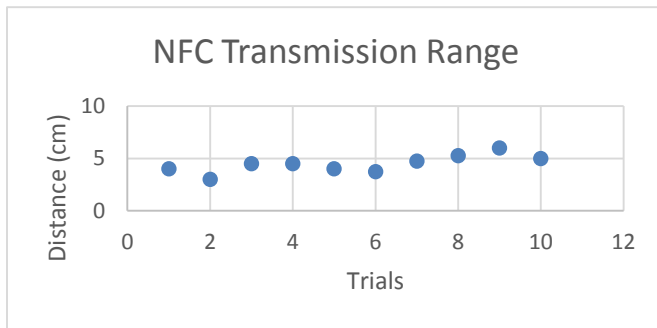


Figure 3.1: NFC Range and Reliability

Figure 3.2 displays the command line execution of the NFC transfer system.

```

pi@raspberrypi: ~/Desktop/digiceipt
pi@raspberrypi:~/Desktop/digiceipt$ python send_receipt.py --device tty:AMA01pn32 scan text "a"
waiting for RDM...
06450000206
input value: 06450000206
sending message
[INFO:lib1] searching for reader on path tty:AMA01pn32
[INFO:lib1] using PN532v1.4 at /dev/ttyAMA0
[INFO:ncp.nfc] using NDEF message found to read 4 NDEFs (104, 104, 104, 104) will accept up to 1048576 bytes NDEF message
[INFO:ncp.nfc] running as NFC-DEF Initiator 424f active mode NID=031 NMT=0.038464
[INFO:llcp.llc] LLCP Link established as NFC-DEF Initiator
[INFO:llcp.llc] sending message
[INFO:llcp.llc] LLCP Version: 1.1
[INFO:llcp.llc] Link Timeout: 300 ms
[INFO:llcp.llc] Max DFU Mail: 2175 bytes
[INFO:llcp.llc] Service List: 0000000000010011
[INFO:llcp.llc] Remote LLCP Settings
[INFO:llcp.llc] LLCP Version: 1.4
[INFO:llcp.llc] Link Timeout: 1000 ms
[INFO:llcp.llc] Max DFU Mail: 2175 bytes
[INFO:llcp.llc] Service List: 00000000010011
[INFO:llcp.llc] sending message
[INFO:llcp.llc] accepting COMMAND from RSP 02
[INFO:ncp.nfc] sending message 02 on remote esp 32
[INFO:llcp.llc] ProtocolError('unrecoverable NFC-DEF error in retransmission request',)
[INFO:ncp.nfc] Stop NFC-DEF Initiator 424f active mode NID=031 NMT=0.038464, packets sent/cvcd INF 17/17 ACK 0/0 NACK 0/0
pi@raspberrypi:~/Desktop/digiceipt$

```

Figure 3.2.: Code & Execution of Data Transfer

#### F. Mobile Android App

The mobile application portion of our system will be used to store and retrieve the digital receipts transmitted by the NFC chip. The application will use 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) for while the iOS platform only supports phones past their iPhone 6 models. The mobile application will also use cloud storage for account management as well as data backups for the user's data.

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

Of course, to test the development of the mobile application, we'll need to run it on our physical Android

devices. To verify the working code within the Android environment, we will program a read/write mode toggle. This toggle will tell 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 detect when they are in proximity with another chip. We will also need to 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 will verify the results of these 2 tests by displaying the data transmitted to the receiving device within the mobile application itself. Figure 4 exemplifies the transfer of the barcode data from the NFC chip to the Android application.

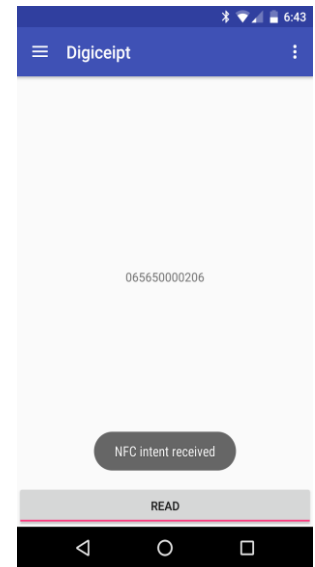


Figure 4: The barcode data is displayed on the mobile application

#### G. Cloud Server

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

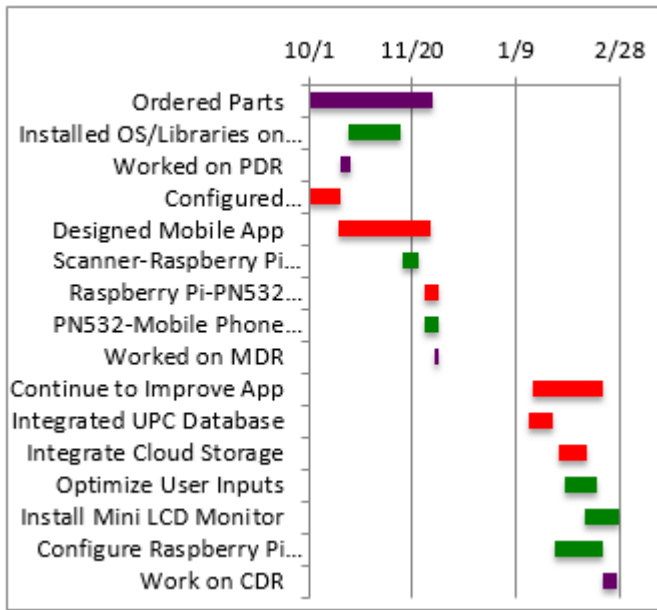
### III. PROJECT MANAGEMENT

For the future, we're looking to turn our barebones system into a more robust user experience. We'll start by allowing the transfer of multiple barcodes over the Raspberry Pi B – NFC chip system as well as displaying the data to the user at the point-of-sale. The Raspberry Pi board will also be integrated with the UPC database for live data on the products we scan. We will also be developing the mobile application to have a more developed user interface as well as login and data retrieval features. We're also looking into methods to present our system in a way that integrates with real world cash registers. The application will also provide cloud access for data backups.

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. George and Edward, who had expertise with microcontrollers as well as the Linux operating system, worked mainly with the Raspberry Pi. 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. 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.



GOAL	STATUS
Scanner interfaces with Raspberry Pi in a way that raw data is transmitted	100% Completed
Raspberry Pi is able to transmit the raw data to NFC transceiver	100% Completed
Transceiver can communicate with Android App and sends raw data via NFC	100% Completed
Android App receives raw data and displays it	100% Completed

Table 2: MDR Deliverables

#### IV. CONCLUSION

Digicept is well on track to provide a full product presentation given the results of our MDR initiative. 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 here we will develop our product on the skeleton of the current system we have. 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 plan on creating a product which will simplify the task of tracking financial history for our users.

As we develop this product, we'll need to ensure the reliability and security of our system. The transfer process will need to be quick (< 2 seconds for receipt transfers) and usable by our users. This will mean making sure the load on our hardware is small and the mobile application has a simple-to-use interface. The communication we have as teammates and the feedback we get on our design will be key tools in making sure we meet both our team's goals and real world expectations.

#### V. 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](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\\_developme nt\\_plastics\\_chemicals.html](http://www.naturalnews.com/052634_bisphenol_s_brain_developme nt_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/raspberrypi-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] "Near Field Communication," *Near Field Communication*. [Online]. Available at: <http://developer.android.com/guide/topics/connectivity/nfc/index.html>. [Accessed: Feb-2015].
- [9] "IDC: Smartphone OS Market Share," [www.idc.com](http://www.idc.com). [Online]. Available at: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. [Accessed: 2015].
- [10] "Getting Started," Getting Started. [Online]. Available at: <http://developer.android.com/training/index.html>. [Accessed: Apr-2015].