Chapter 1

Introduction

1.1 Problem Statement

The vending machines currently used at Stellenbosch University (SU) exclusively make use of cash transactions. These vending machine systems are currently the de facto standard throughout the world, but they do have one drawback: they require a hard cash transaction to take place. In a world moving away from cash transactions and towards online payments, e-transactions and mobile payments, this may become a problem to potential customers.

With that in mind, a need has been identified at SU for a vending machine that accepts cashless transactions.

1.2 Existing Solutions

Currently there are cashless payment solutions being used by the general public. These include debit and credit cards, Radio Field Identification (RFID) cards, Unstructured Supplementary Service Data (USSD) based systems and, more recently, Near Field Communication (NFC) payments.

These aspects will be further discussed in this section.

1.2.1 Credit and Debit Cards

A well-known and widely-used alternative to cash payments are the debit and credit cards most modern adults carry on their person. This is especially true in developed countries with mature banking and reliable financial institutions.

The great advantages these cards hold over the other cashless solutions are that they are relatively easy to obtain from a bank and that they have become very reliable and simple to use.

A possible disadvantage is that such systems may become costly and complicated to implement because of each banks different security and transfer protocols. This is especially true for a simple system such as the one developed for this project.

1.2.2 Radio Field Identification

Radio Field Identification (RFID) is a technology which was first patented in 1983 [Walton (1983)]. Since then, the technology has grown and matured into a very reliable identification and payment platform.

Examples of where this is used for making payments is the payments made to new parking meters with a contactless card.

The advantage of this technology is its great convenience: a customer only needs to tap the card against a receiver and it is not required that a password be entered.

However, this leads to some security concerns, for example if the card gets stolen or cloned, the thief can use the money on the card for her own benefit. Fortunately, these cards most commonly work with pre-paid money. Therefore, provided that there was not too much money loaded onto the card, the theft victim will not suffer a big financial loss.

Therefore, an RFID card has the same risk as regular cash.

1.2.3 Unstructured Supplementary Service Data

Unstructured Supplementary Service Data (USSD) is a communication standard used by cellphones to exchange data with a service provider's servers. If the service provider allows it, USSD may be used by a customer to make financial transfers.

An example of this is the M-Pesa mobile money service in Kenya, which is based on the use of USSD. It allows for a customer to pay for goods ranging from milk to bread and even the monthly rent. It is currently regarded as the most advanced and popular mobile payment platform in the world [Jack, William and Suri, Tavneet (2011)].

An advantage of implementing such a system is that it has been proven to be reliable and is usable by almost any cellphone.

The disadvantage is that it requires third party vendors, such as the mobile service providers, to provide systems and services. This may add unnecessary costs to the system.

1.2.4 Near Field Communication

Near Field Communication (NFC) payments have recently come to the fore as a prominent method of making cashless transactions. Especially in Europe and North America, there have been significant advances in making this payment method a

more attractive solution. Google has been making the largest contribution with the addition of NFC protocols to its Android platform[Google Android Team (2009)].

Some examples of NFC based payments are the London public transport system, which makes provision for NFC payments [Weinstein (2009)], as well as some retail vendors which accept payments made via Google's Wallet application.

1.3 Goal of the Final System

Although hard cash still remains the largest contributor to global financial transactions, standing at 59% of the 37 billion transactions that took place in 2012 [Humphrey (2004), Pasquali, Valentina and Bedell, Denise (2013)]. However, mobile and card transactions are expected to surpass cash as the leading payment method by 2015 [Wollop, Harry (2010)].

To this end, mobile payments, i.e. payments made with a cellphone, has chosen as the medium to facilitate cashless payments for this vending machine. Therefore, the final goal of this project would be to deliver a vending machine that will be used on SU's campus and will allow anyone to buy products from the vending machine using only their cellphones.

1.4 System Objectives

The system objectives are:

- The system must make provision for both NFC and Quick Response Code-based (QR Code) payments.
- The system must make use of a web server based in the cloud, i.e. it must be accessible by anyone across the world.
- An Android application must be made that will allow transactions to be completed using NFC.
- A demonstration model vending machine must be designed and constructed.
- All the data transfers between the user's cellphone and the server must be encrypted.
- Extra layers of security must be added to prevent theft and product losses.

1.5 Report Structure

In this report, Chapter 2 will give background information on all the technology, concepts and programs used in this project. Thereafter, in Chapter 3 the overall system design is discussed, which will be followed by a discussion on the detailed design of the software and hardware aspects of the entire system in Chapters 4 and 5. Afterwards, in Chapter 6 the system test results will be discussed and analysed, which will be followed by a discussion on the complete system and finally the conclusion on the project report in Chapter 7.

Chapter 2

Background Study

This chapter contains background information on the software, services and algorithms used in this project. They are divided up into Quick Response Codes (QR Codes), Near Field Communication (NFC), the web server and the encryption algorithms used.

2.1 Quick Response Codes

Quick Response Codes (QR Codes) are two dimensional bar codes that were initially used in Japanese car factories to allow computers to track the progress of an item on a production line [Soon, Tan Jin (2008)]. The technology has since evolved and matured and is today widely used in the media industry for storing data, such as a web address or a phone number. See Figure 2.1 for an example of a QR code.

QR Codes can store up to 7089 alphanumeric characters [Soon, Tan Jin (2008)], which are accessible by scanning the code with either a laser or a digital camera.

Figure 2.1: Example of a QR Code.

Scanning a QR Code requires a camera that can produce a digital image at a resolution that is at least twice that of the QR Code. This image is then processed by a QR Code library, e.g. the ZXing library (see section 2.1.1 for more detail on the ZXing library), which decodes the picture and extracts the data embedded inside the code. Cellphones are commonly used today because of their portability, increasingly powerful hardware and QR Code technology's simplicity. However, an image with an embedded QR Code can be decoded by any computer with the appropriate hardware and libraries installed, e.g. a webcam and the ZXing library.

2.1.1 Zebra Crossing Library

The Zebra Crossing Library (ZXing for short) is a QR Code coding and decoding library [ZXing Team (2013)]. It is commonly built into smart phone applications to decode QR Codes embedded inside a static image or a video stream. A desktop version of the library, called ZBar, is also available and works in a similar manner.

The Barcode Scanner app is made by the team that made the ZXing library. It is freely available on multiple cellphone platforms, such as BlackBerry OS, Apple's iOS and Google's Android. To date there have been at least 50 million downloads of ZXing's Barcode Scanner app on the Android platform alone, and it currently lies 98^{th} in the top 100 of the Google Play Store's most downloaded list [Google Play (2013)]. This shows the extent to which QR Code technology and the ZXing library has evolved to be used by millions of people.

2.2 Near Field Communication

Near Field Communication (NFC) is a relatively new communication standard in the world of wireless technology. It allows two NFC-enabled devices to wirelessly transmit data by bringing them close to one another, typically around 4 centimeters.

Most mainstream cellphone manufacturers, with the major exception of Apple, have added NFC capabilities to their flagship models, and more recently to some of their budget models[NFC World (2013)]. However, the technology has been ported to other platforms, such as the desktop computer and Arduino. This adds a new dimension to wireless inter-device communication and makes projects such as this more feasible.

2.2.1 libnfc

Libnfc is an open-source library for Linux systems [Libnfc Team (2013*b*)]. It allows a desktop computer to communicate with a NFC device which is based on the Phillips PN53 series of NFC chips [Libnfc Team (2013*a*)]. Recent versions have

made provision for the use of a PN532 breakout board that can be used by a Raspberry Pi.

It is currently in version 1.7 and is classified as a 'mature' library by the open-source community.

nfcpy

Nfcpy is a Python interface for the libnfc library and it allows for peer-to-peer communication between a cellphone and desktop-based NFC controller using the NFC Data Exchange Format (NDEF), the Simple NDEF Exchange Protocol (SNEP) and the Logical Link Control Protocol (LLCP). These standards and protocols have been set by the NFC standard governing body, the NFC Forum [NFC Forum (2013)], to simplify and standardise data exchange between different platforms and to make the user experience more pleasant.

Nfcpy is an open-source program, written and maintained by Stephen Tiedemann [Tiedemann, Stephen (2013)].

2.2.2 Android

Google's Android operating system is currently the most widely used cellphone operating system world wide, with an estimated 80% market share [Etherington, Darrel (2013)]. Other platforms, such as the Blackberry OS and Windows Phone, have also added NFC to their latest phones, but they do not have the market penetration that Android currently has and it was found that app development on the Android platform is relatively simple and free.

Android is also the platform which most actively promotes the use of NFC as an alternative payment option in modern retail outlets, with applications such as Google Wallet [Balaban, Dan (2012)] being heavily promoted by Google.

2.2.3 Radio Field Identification and Stellenbosch University Student Cards

NFC and Radio Frequency Identification (RFID) work in a similar manner: when two devices (e.g. cellphone, RFID tag, MiFare card, etc.), equipped with an antenna tuned to a certain frequency, for example 13.56MHz, come into close proximity, they transmit some form of data to one another.

However, there are some important difference between the two technologies. For example, a NFC system is an active system, meaning that the device's antenna is always powered and runs off its own power supply. NFC devices also have peer-to-peer (p2p) capabilities, meaning that the two devices can communicate with one another by both sending and receiving data. RFID systems on the other hand,

work by having one device act as a listener and the other as a sender [Chandler, Nathan (2012)] (e.g. the current SU's student entry control system).

2.3 Web Server

The web server is responsible for handling all the data transfers and transactions that take place when a customer buys a product.

In this section, some background information will be given on the key features of the web server that was implemented for this project.

2.3.1 Django Web Framework

Django is a Python web server framework which focuses on easy setup and simple design. Here are some of its features:

- Fully handles Hypertext Transfer Protocol (HTTP) GET and POST requests.
- Integrates with SQL databases, e.g. MySQL, SQLite3, etc.
- Supports Hypertext Markup Language (HTML) template design.
- Makes provision for the execution of Python scripts.
- Is fully scalable to commercial level servers.
- Has an offline server debugging function available.

This framework is expandable to commercial size servers that are accessible across the globe. For example large websites such as Instagram and Pinterest are based on the Django framework [Django Software Foundation (2013*a*)].

To make it easier to program, read and debug, the original Django developers designed Django to split its websites into multiple so-called 'applications'. These applications typically contain a single web-page with its own script and database handling functionality. These applications can communicate with one another, meaning that one script from application X may execute a script or call a function in application Y.

Django was initially developed by web programmers Adrian Holovaty and Simon Willison, from the newspaper Lawrence Journal World [Django Software Foundation (2013*b*)]. It was first released in 2005 under the Berkeley Software Distribution (BSD) license and is completely free to use.

2.3.2 Elastic Compute Cloud

Elastic Cloud Compute (EC2) is a cloud computing service offered by Amazon Web Services (AWS)[Amazon Web Services (2013)]. It allows a user to rent a cloud-based virtual computer from AWS from which to run their own applications. These applications are commonly web-based, in other words the virtual machines run a web server that is accessible by anyone around the world.

AWS offers new users the option to create a virtual machine for free. These free virtual machine instances are limited to the least powerful machine tier available, but functionally the free and paid tiers are the same.

2.3.3 Apache

Apache is a popular web server application available on most operating system platforms [Apache Software Foundation (2013*b*)]. A very notable feature of Apache is that it was designed to be easily configurable in order to be compatible and fully scalable with various web frameworks, such as Python's Django, PHP's cgiapp and C++'s Poco.

Apache is the most widely-used web server framework in use today, with an estimated 53.4% of the world's servers running on it [Netcraft (2013)].

It was initially released by Robert McCool in 1995 under the Apache Licence, which makes it free to use in any way. It is currently being maintained by the Apache Software Foundation [Apache Software Foundation (2013*a*)].

2.4 Encryption

Encryption is the act of encoding some data into seemingly unreadable garbage. This is done so that unwanted parties cannot read or access the data, but authorised parties can. This is most commonly done with an encryption key and an algorithm which specifies how the data was encoded and how it can be decoded.

Encryption is commonly used where sensitive information is being transmitted between two remote parties, e.g. banking codes, personal e-mails, etc.

There are two main encryption schemes, namely symmetric and asymmetric encryption. Both are discussed in this section.

2.4.1 Symmetric Encryption

In symmetric encryption, both parties, i.e. the sender and receiver, have to agree to a common encryption key prior to the data transmission. In other words, the sender and receiver use the same key to encrypt and decrypt the data [Tyson, Jeff

(2001)]. A famous example of symmetric encryption is the Enigma cipher machine used by Nazi Germany during the Second World War [Stripp, Alan (1999)].

Unfortunately, due to the increase in knowledge and understanding around this type of encryption and the increase in modern computing power, various code cracking methods have been developed since 1945 that can break the most commonly used symmetric encryption methods, such as known and chosen plain-text attacks[Biham, Eli (1994)]. Methods, such as the One-Time Pad (OTP) encryption, are still being used today and is considered to be unbreakable [Rijmenants, Dirk (2011)]. However, the difficulty of securely exchanging the keys between the communicating parties is difficult [Rijmenants, Dirk (2011)].

2.4.2 Asymmetric Encryption

Another encryption scheme is asymmetric encryption, or public-key encryption. It involves the use of a public and private key pair that can be used to securely encrypt and sign a data package on the sender's side and to decrypt and verify the data and its origin on the receiver's side.

These private and public keys are mathematically related to one another according to the algorithm in use (e.g. ElGamal or RSA. See sections 2.4.2 and 2.4.2 for more information on these algorithms). The public key is used to encrypt data and may be publicly distributed. However, in practise the keys are only distributed to trusted parties to increase security. The great advantage of asymmetric encryption is that even if the public half of the key is available, it is still very difficult, and sometimes impossible, to get the private half of the key from the public key alone.

The private key allows one to decrypt the data encrypted with the public half of the key.

The encryption is most easily explained with a postal analogy:

Imagine that two people, Alice and Bob, want to send each other secret messages through the public mail. In other words, Alice wants to send Bob a secret message and she expects a secret reply from Bob, and vice versa.

In an asymmetric scheme, Bob can lock his letter to Alice with a padlock to which only she has a key (she keeps this on her person at all times and does not show it to anyone, which includes Bob). This open padlock represents the public key half of Alice's key. This means that anyone can send Alice a secure message with a public key, which is easy to get from Alice, and only Alice can unlock the message with her private key half of the key pair. Similarly, Alice can lock her letter with Bob's padlock which only Bob can open.

The great advantage that this has over symmetric encryption is that the decryption keys never have to be exchanged between parties. This neutralises the risk of a middle-man attack, analogous to a nosy postal worker called Eve who likes to read other people's mail, who then intercepts the message and steals the key. Also,

if for example Bob has been careless and allowed Eve to see his key, his messages to Alice will be compromised. However, the messages from anyone else (including Bob) to Alice will remain as secure as it was before Bob lost his key.

The data source can also be signed and verified by using this key scheme. Referring once again to the postal analogy:

To show Alice that it was indeed Bob who sent her the message, and not Eve for example, he can send an extra message along with the original message. This extra message is locked with Bob's key that he shares with no one (i.e. his private key). However, Bob has sent out his public key to everyone who wants it. These keys can *only* be used to unlock the messages locked with Bob's own private key. Therefore, if Alice, who received one of Bob's keys, can unlock this extra message with that key, she knows that as long as Bob has not given anyone his private key, it can only be his message. The reverse is also true if Alice wants to prove to Bob that it was indeed she who sent him a message.

ElGamal

The ElGamal encryption algorithm is an alternative to the more widely used Ron Rivest, Adi Shamir and Leonard Adleman (RSA) asymmetric encryption algorithm. ElGamal's security stems from the 'difficulty of computing discrete logarithms in a large prime modulus'. [Leon, Jeffrey S. (2008)]

The main advantage that the ElGamal algorithm has over RSA is that, firstly, a smaller key can be used for a data string of the same length and secondly, due to the mathematics behind the algorithm, it is almost certain that a different ciphertext will be generated each time a string is encrypted.

However, a fairly large drawback of this algorithm is that the key needs to be at least twice as long as the plain-text string that is being encrypted [ElGamal, Taher (1985)].

The algorithm was developed by Taher ElGamal in 1984 and is free to use under the GNU license.

RSA

The Ron Rivest, Adi Shamir and Leonard Adleman (RSA) asymmetric encryption algorithm is a widely used encryption standard. Its security is based on 'the difficulty of factoring large integers' [Leon, Jeffrey S. (2008)].

The main advantage of the RSA algorithm is its encryption and decryption speed. Also, the encryptor has some measure of control over how long the produced ciphertext is going to be, because the ciphertext will be as long as the encryption key used, provided the key is long enough.

The RSA algorithm was developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1978 and has been widely used since 1993.

2.4.3 PyCrypto

PyCrypto is a Python cryptography toolkit which contains various encryption algorithms and key schemes, such as ElGamal, MD5 and RSA. It is currently registered under the Python License and is available in the public domain. It is maintained by the PyCrypto Team [PyCrypto Team (2013)].

2.4.4 Base64 Encoding

Base64 encoding is a scheme which represent arbitrary data as alphanumeric characters. This is often used where the output of encrypted text is human-unreadable binary data and ASCII characters are preferred because they are easier to read by humans and simpler to transmit via HTTP.

It is also important to note that the the output of base64 encoding is approximately 33% longer than the input string.

Here is an example of a base64 encoded string:

Original text:
Hi, I'm a base64 encoded string!
Base64 encoded output:
SGksIEknbSBhIGJhc2UgNjQgZW5jb2RlZCBzdHJpbmch

Chapter 3

System Design

In this chapter, the overall system design is discussed. The chapter is divided up into six sections with each section focusing on a different aspect of the complete system. These six sections are:

1. The vending machine central controller.
2. The Near Field Communication controller.
3. The Quick Response Code camera.
4. The product dispensing mechanism.
5. The vending machine unit.
6. The encryption scheme used.

Each section explains which technology or service was used in the final component design.

Where two or more available services or technologies were available, a brief discussion and explanation is given as to why the particular technology or service was used in the final design of the vending machine.

3.1 System Overview

Figure 3.1 gives a diagrammatical layout of the complete system. It shows the interactions between the different sub-components of the complete system.

The components used in these subsystems are discussed in the subsequent sections of this chapter.

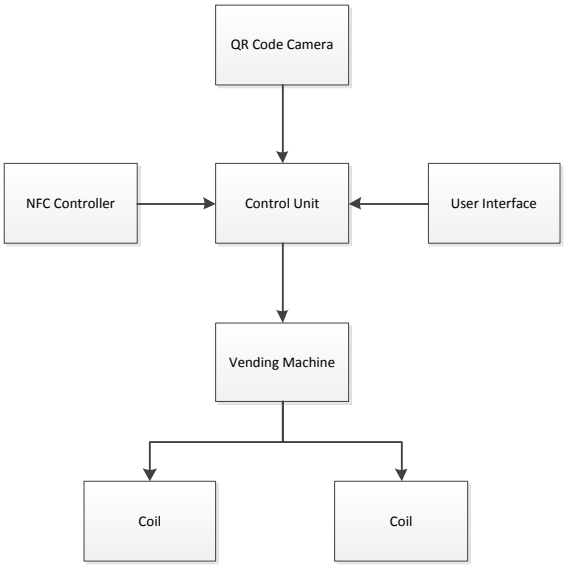

Figure 3.1: System overview from the control unit's perspective.

3.2 Central Control Unit

To be able to handle the image processing that QR Code decoding and NFC handling requires, a relatively powerful central controller is required. Although there are many controllers capable of this, only two main alternatives were considered in this project. They are the Raspberry Pi microcomputer and the Arduino Uno microcontroller.

These controllers are discussed in this section.

3.2.1 Arduino Uno

The Arduino Uno is popular open-source microcontroller (see Figure 3.2). It is based on an 8-bit Atmel ATmega328 ARM microprocessor. Its official specifications are given in Table 3.1.

Because of its open-source design, there are a multitude of peripheral devices and expansion boards (known as 'shields'), along with all their libraries and drivers, available locally. The Arduino's programming language of choice is a modified version of C and comes with its own Integrated Development Environment (IDE). This, along with its relatively low cost and specifications, makes the Arduino Uno an attractive option for this project.

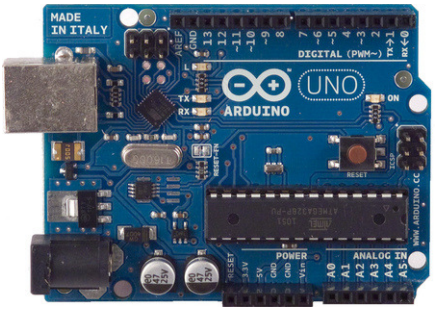

Figure 3.2: Picture of an Arduino Uno microcontroller [Arduino (2013)].

Table 3.1: Arduino Uno Specs[Arduino (2013)].

Operating Voltage	5 V
Processor	Atmel ATmega328 (clocked at 16 MHz)
GPIO Pins	14 (6 of which are Pulse Width Modulator enabled)
Memory	32 kB Flash, 2kB SRAM, 1kB EEPROM
Communication	i^2c, UART, SPI
Price	R310.00

Table 3.2: Raspberry Pi Specifications [eLinux (2013)].

Operating Voltage	3.3 V
Processor	ARM 6 clocked at 700 MHz
GPIO Pins	28 Pins
Memory	512 MB RAM
Communication	SPI, UART, USB, i^2c, Ethernet
Price	R400.00
Video	HDMI Output

3.2.2 Raspberry Pi

The Raspberry Pi is a Debian Linux-based microcomputer designed and manufactured by a UK-based charity called the Raspberry Pi Foundation, for the purpose of educating and familiarising young children with programming. However, its low price and respectable specifications makes it a strong choice as a control unit for the vending machine.

The Pi was designed with the focus on Python as its main programming language, which makes running scripts and controlling the board relatively simple. It also runs on a modified version of Debian Linux called Raspbian.

Its main specifications are given in Table 3.2.

3.2.3 Design Choice

The Raspberry Pi was chosen to be used as the central controller of this project and controls the hardware connected to it via a Universal Serial Bus (USB) connection, or one of its General Purpose Input Output (GPIO) pins.

The Pi was chosen ahead of the Arduino, mainly because of its video stream processing capabilities and that libnfc can be installed on it. This allows the Pi to decode QR Codes and to communicate with an NFC-enable cellphone. Furthermore, the Pi can interface with desktop peripheral hardware, such as a mouse and keyboard, which made development and prototyping much simpler.

After development and optimisation has been completed for this project, work can begin on porting it to a micro-controller, such as an Arduino. However, for the development phase of this project, the Pi is much better suited platform.

3.3 NFC Controller

The NFC controller that was selected is the PN532 NFC shield from Adafruit Industries [Adafruit Industries (2012)]. It is based on the Phillips PN532 chip, which is a widely used NFC chip. The main reason it was selected instead of other NFC controllers was that it has a large support base in the open-source community and is fully compatible with the libnfc open-source NFC library [Libnfc Team (2013*a*)]. The manufacturer, Adafruit, inc., also provides comprehensive documentation and guides on how to set up and configure the controller [Townsend, Kevin (2012)].

The main purpose of this component is to add the option of sending or receiving data through a NFC connection. This component is also capable of reading Radio Frequency Identification (RFID) cards, such as student or staff cards, since NFC and RFID transmit similar types of data. This adds the option of paying for the products with any NFC-capable smart phone running Google's Android operating system, or with a SU staff or student card.

3.4 QR Code Camera

To decode QR Codes, the vending machine needs to take pictures of a code so that it can be decoded by a QR Code library, such as the ZBar library. A PlayStation 2 EyeToy was chosen and added to the system to facilitate this. It was chosen for the following reasons:

- Its drivers are freely available for Linux systems [ov511 (2013)].
- Interfaces easily with the USB ports on the Pi.

Figure 3.3: Example of a vending machine coil system.

- One was readily available.

There is currently a camera add-on available for the Raspberry Pi, but this is relatively expensive (approximately $30 [Adafruit (2013)] versus the Pi's cost of $35 and the PS2 EyeToy's cost of $10) and at the time of writing, unavailable in South Africa.

3.5 Product Dispensing

3.5.1 Coils

To be able to effectively dispense bought products to the user, a coil mechanism is used. Such systems are the most familiar and simple methods of dispensing goods. See Figure 3.3 for an example.

These coils are designed and made in such a manner that one rotation of the coil will drop one product. The turning motion is made by attaching a DC motor to the base of the coil (see section 3.5.2 for a more detailed description).

3.5.2 DC Motors

The motors attached to the base of the coils are two 12 V DC motors from Faulhaber [Faulhaber (2013)]. Although these motors are rated for 12 V, it is possible to run them from a lower voltage. This will cause the motor to turn slower, and therefore be easier to control.

The motors are switched on by a 12 V relay switch controlled by the Raspberry Pi. See section 3.5.3 for more detail about the switch.

3.5.3 Relay Switch

A relay is a type of electronic switch, which means that it acts like a normal switch, but requires a voltage across it to open or close it. With this, it is possible to control when the DC motors turn (after a successful transaction) and when they are standing still.

However, the relays used here are 12 V, because they were readily available. The Raspberry Pi can deliver a maximum voltage of 5 V. Therefore it was decided that the relay will be permanently connected to a 12 V DC supply, but will be switched by a 2N2222 transistor, which is controlled directly from the Pi's GPIO pins (see Section 5.1 on page 37 for a detailed discussion on the switch).

This allows the Pi to directly control the motors and due to the circuit's construction, the Pi is protected from the relatively high voltages and currents involved in the working of the motor and relay.

3.6 Vending Machine Unit

The vending machine unit houses all the components (i.e. the Raspberry Pi, the NFC Shield, webcam, switches, motors and the product coils). See Appendix A for detailed manufacturing drawings.

3.7 Web Server

A web server in the computing cloud is used to process and authenticate the transactions that take place in the vending machine. Arguably, this could have been done by the vending machine itself. However, the use of a web server allows for the system to be expanded beyond a single vending machine and standardises the transaction across all of the vending machines connected to the server.

The server is used to communicate with a cellphone over the internet via Hypertext Transfer Protocol (HTTP) requests and Universal Resource Locators (URLs).

How the web server is used to process and authenticate each payment method is described in Section 3.8.

3.8 Transaction Design

Two different transaction processes were designed to accommodate for NFC, RFID and QR Code payments. However, all three of the payment options have some common elements to them.

Firstly, each product in the vending machine is assigned a unique, 4-letter Hexadecimal number. This is done to fit in with the security code scheme described in Section 4.1 on Page 21. When a customer selects a product to buy, this unique code is used identify which product to dispense. To simplify the demonstration system, only two codes are used in this project.

The second common element between the payment options is that all the transactions require authentication from the central server. At this stage, this excludes RFID card payments. The reason for this is explained in Section 3.8.3.

3.8.1 QR Code Transactions

3.8.2 NFC Transactions

3.8.3 SU Card Transactions

3.9 Encryption Scheme Design

To prevent the system from being hacked, an asymmetric encryption scheme was implemented (see Section 2.4.2 on page 10 for more information on asymmetric encryption). This is a secure method to exchange data between two sources and in the event that one of the vending machine's keys are cracked, the system will still remain secure for the other vending machines.

Two different schemes were implemented for the Android NFC app and the QR Code payment option. These two schemes are discussed in this section.

3.9.1 Android NFC App

For the Android NFC app, it was decided to use a 1024-bit key based on the Ron Rivest, Adi Shamir and Leonard Adleman (RSA) encryption algorithm. A 1024-bit key was chosen because it gives a good balance between security andencryption speed.

It was decided to base the app's encryption on RSA, because it is already included in the Android Development Kit (ADK) and is therefore the simplest to implement and distribute with the app.

3.9.2 QR Code

For the QR Code payment option, a 384-bit key based on the ElGamal algorithm was chosen.

The reason for choosing this key size was to produce smaller, less complex output which leads to a more readable QR Code. A smaller key size also reduces the time it takes to encrypt a data string.

It was found that the ElGamal algorithm in the PyCrypto module allows for key sizes of any bit length. Therefore the encryption used is based on the ElGamal encryption algorithm.

Chapter 4

Software Detail Design

In this chapter, the software design aspects of this project are discussed in detail.

This chapter is divided up into four sections, with each section explaining what the program being discussed is responsible for, what third party programs it uses and how it interacts with the rest of the system.

These four sections are:

1. The security scheme used.
2. The web server that was created.
3. The vending machine's control program.
4. The Android app that was created.

4.1 Security Scheme

In addition to the asymmetric encryption used on all data transfer to and from the server to its clients, two more layers of security were added to prevent repeated use of the same code and to make it harder for hackers to crack the encryption key.

4.1.1 Random Character String

The product code transmitted to the server is embedded inside a random 16-character string of hex values, i.e. numbers from 1 to 9 or characters from A to F. The product code is a 4 character hex string, but this is saved on the database and can therefore not be random. A hex string was chosen to make it harder for a hacker to see that they have sucessfully broken through the security. For example, if the hackers manage to break through the security, they will see a string like this one:

A2FB3291CFF

If the product code were alphanumeric, for example '$coke_3 50ml'$, $thehackerswillimmediatelyseeth$

This scheme does not make the code uncrackable, but it does make brute-force attacks harder to do.

4.1.2 Challenge and Response Code

The second layer of security added is a challenge/response system. Such a system works by having party A generate a challenge (this can be a string or a number). Party B then takes this challenge and puts it through a previously agreed-upon process, e.g. makes the letters capital or adds the numbers together. This is the response. Party B then sends the back the response to party A, which then checks to see if its a valid response to party A's original challenge.

In the case of the vending machine, a 16 character hex string is being generated. It was decided to use 4 characters of this random string as the vending machine's challenge. After receiving this challenge, the server then takes out the agreed-upon 4 characters and adds it to the server's response code. When the vending machine scans the customer's response QR Code, the vending machine checks to see if the 4 character response was part of its original code the vending machine generated.

This system is used to prevent customers from only buying one product and using the same code again to get another product for free. Thus, the challenge/response code system makes each code only valid for one transaction.

4.2 Web Server Program Design

The web server that was made for this project is based on the Django web framework for Python (see Section 2.3.1 on page 8 for some background information on Django). The Django server was then configured to run on top of an Apache web server located on an Amazon Web Service (AWS) Elastic Compute Cloud (EC2) cloud computer instance (see Section 2.3.2 on page 9 for some background on EC2 and Section 2.3.3 on page 9 for some background on Apache).

The server is responsible for handling all the data and financial transactions that will take place during the vending machine's operation.

This section stipulates the design of the complete server and how the different components interact with one another.

4.2.1 Django Server

The Django server is responsible for all of the scripting and database work that the server performs. The server is divided up into a total of six applications. Each

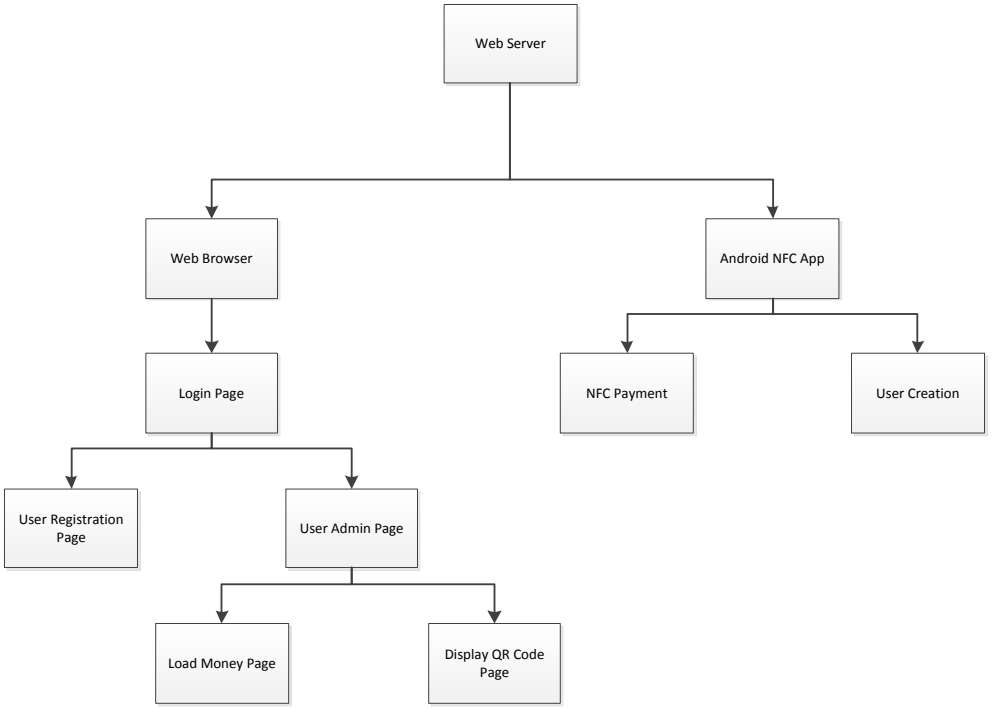

Figure 4.1: The web server app structure.

of these is responsible for either displaying a single web page or to handle data requests from the Near Field Communication (NFC) Android app (see Section 4.4 on page 32 for more details on the Android app).

The website apps and their interactions with the real world and one another are given in Figure 4.1.

display_qrcode

This app forms the core of the Quick Response Code (QR Code) payment handling part of the server. See Figure 4.2 for the process flow of this app.

As seen from Figure 4.2, the app first extracts the code containing the product code and vending machine's signature from the Uniform Resource Locator (URL) that the customer visited with his cellphone's web browser. The app then proceeds to decode the the code from the base64 encoded format it was sent in (see Section 2.4.4 on page 12 for some background information on base64 encoding).

After successfully decoding the data, the app proceeds to decrypt the data and signature with the ElGamal algorithm using the server's private key and the

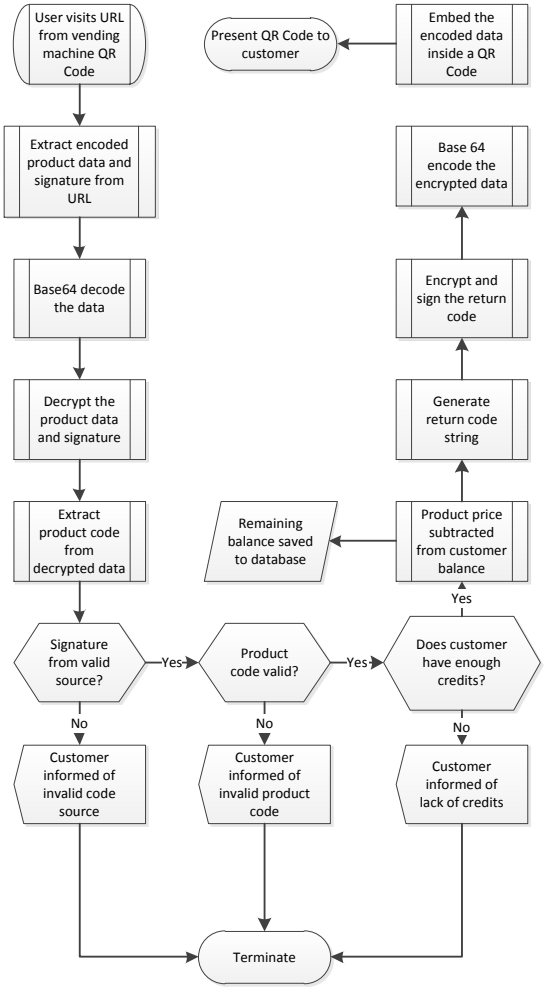

Figure 4.2: The display_qrcode app process flow.

vending machine's public key (see Section 2.4.2 on page 10 for more detail on public and private keys) and, following the security code scheme described in Section 4.1, extracts the product code from the decrypted string.

The app then checks to see if the signature comes from a valid source (i.e. one of the vending machines using this system), if the product code is valid (i.e. the product is in the database) and if the customer has enough credits loaded loaded onto his account. If either of these checks return false, an appropriate error message is shown to the customer explaining what went wrong and what the customer should do next.

If the checks were passed, the app proceeds to subtract the product cost from the user's remaining balance. Using the security code scheme, the app then generates

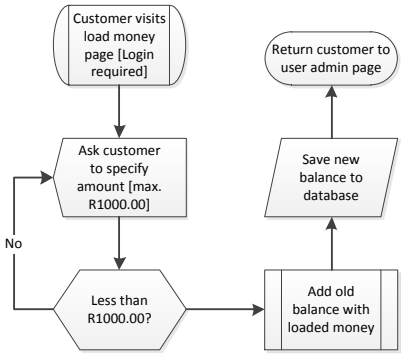

Figure 4.3: The load_money app process flow.

the correct return code, encrypts and signs it with the vending machine's public key and the server's private key, and encodes it with base64. After this is completed, the app embeds this data into a QR Code, which is returned to the customer's cellphone screen.

load_money

This app allows the customer to load money onto his account. At the moment it makes use of faux money, meaning that the money loaded has no real-world value. The customer can load a maximum of R1000.00 onto his account.

See Figure 4.3 for the process flow of this app.

nfc

This app forms the core of the NFC payment handling part of the server. See Figure 4.4 for a detailed process flow diagram.

As seen from Figure 4.4, the server first extracts the encoded and encrypted customer login details and product code from the URL the NFC app sends to the server. These codes are then all decoded and decrypted using the Ron Rivest, Adi Shamir and Leonard Adleman (RSA) algorithm and the server's private key.

The user login details are then checked and verified using the server's user database. If this check fails, the customer is given an appropriate error message and informed to create a user profile.

If the check is passed, the server then checks to see if the customer has enough money loaded onto his account. If this check fails, the customer is informed of his lack of funds and is instructed to load more money. If the check is passed, the server subtracts the product cost from the customer's balance and encrypts and encodes

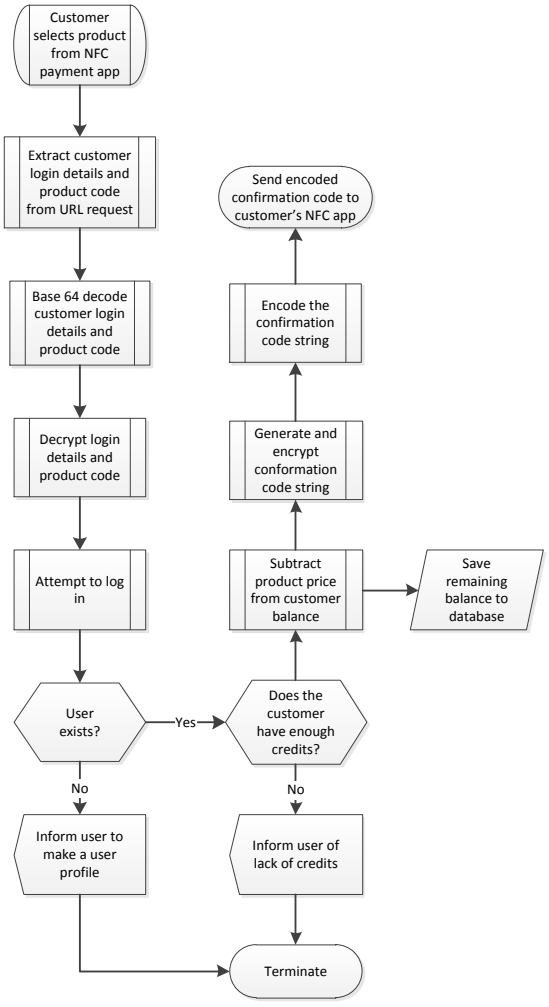

Figure 4.4: The nfc app process flow.

a confirmation code, according to the security code scheme specified in Section 4.1, which is then sent to the NFC app.

nfc_add_user

This app allows a customer using the NFC app to create a user profile for himself/herself. The server extracts the customer's login details from the URL request that the NFC app send to the server. The user name and email are kept in plain text, but the password is decrypted to plain text with the RSA algorithm and the server's private key.

These details are then saved to the database and is immediately available to be used by the new customer.

register

This app allows a new customer to register. This app is only accessible by a web browser and not by the NFC app. However, a user registered with this app will be able to use the same login details for the NFC app.

The app presents the user with a simple registration page which asks for a user name, email and password. Using Django's built-in form support. This allows the server to handle the POST request that is generated when the customer presses the 'Continue' button. When this is done, the login data contained within the POST request is extracted and saved into the user database.

4.2.2 EC2

The AWS EC2 server provides the platform on which the Apache server runs. The EC2 server instance was configured to run Ubuntu 12.10, 'Quantal Quetzal'. This was done because most of the server development was done on Ubuntu 12.10, and the server code will therefore require minimal adaptation to be able to run on the EC2 server.

After the server instance was created, the following packages and programs were installed for the server to be able to run properly:

- Apache2: Installs the Apache server framework.
- libapache2-mod-wsgi: An Apache module that allows Apache to work with Python wsgi scripts.
- python-pip: Allows Ubuntu to install Django from the Python Package Index (PyPI) [Python Package Index (2013)].
- Django: Installs the all the Django packages that will be used by the server.

Because the server's database uses SQLite3, and Ubuntu 12.10 comes with it by default, no external database programs were needed to be installed.

After this was completed, the server is fully capable of serving Django web pages.

4.2.3 Apache

The Apache server framework provides the foundation on which the Django server runs. It had to be configured to be able to run the Python scripts that Django contains. To do this, the steps described in Nick Polet's blog post was followed [Polet, Nick (2013)]. It describes in detail how to configure Apache to serve a Django website.

For Apache to be able to serve Django sites, it had to be configured to run the Web Server Gateway Interface (wsgi.py) script located within the Django server folder. This was done by adding the following code to the Apache's httpd.conf configuration file:

```
WSGIScriptAlias / /home/ubuntu/srv/server_site/server_site/wsgi.py
WSGIPythonPath /home/ubuntu/srv/server_site

<Directory /home/ubuntu/srv/server_site/server_site>
<Files wsgi.py>
Order deny,allow
Allow from all
</Files>
</Directory>
```

The following line was also needed to be added to Apache's apache2.conf file:

```
Include httpd.conf
```

4.3 Vending Program

The vending machine's program runs the vending machine. It is responsible for allowing the customer to select a product, to create a QR Code for that takes the customer to the web server, to scan the customer's response QR Code, to scan the customer's NFC request and to dispense the product after a successful transaction.

The whole program is based on Python scripts and designed to be used by a Raspberry Pi microcomputer (see Section 3.2.2 on page 15 for more details on the Raspberry Pi). The whole transaction process is given in Figure 4.5.

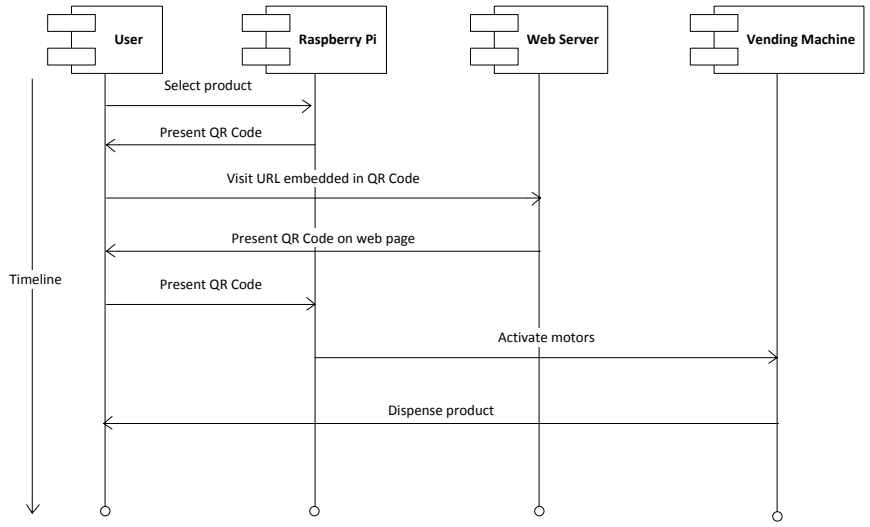

Figure 4.5: The vending machine transaction process.

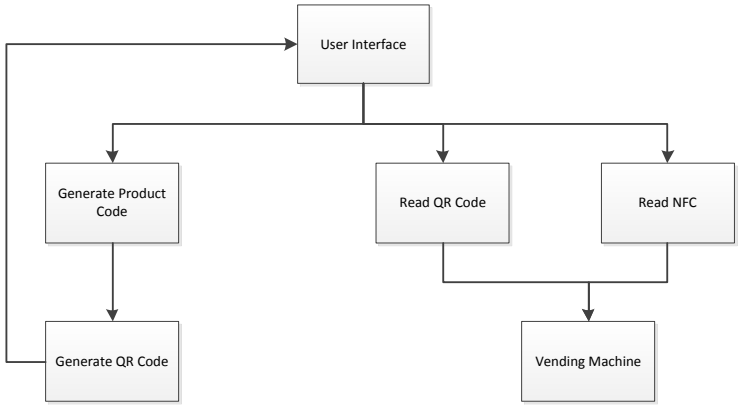

Figure 4.6: The vending machine's program structure.

To simplify the program, its split up into separate sub-programs. These sub-programs, called scripts, are discussed in this section.

The program structure can be seen in Figure 4.6.

4.3.1 User Interface

To allow the customer to select a product, a Graphical User Interface (GUI) was created. The GUI was made using the WX Python GUI toolkit [WX Python Team (2013)]. See Figure 4.7 for a screenshot of the GUI.

Figure 4.7: A screenshot of the user interface.

As can be seen from Figure 4.6, the GUI script is responsible for calling the encryption script and the QR Code generation script. It is also responsible for displaying the QR Code, to handle transactions from the Android NFC app and to activate the correct motor inside the vending machine.

4.3.2 Generating a Product Code

After the customer selects which product to buy from the GUI, the encrypt_elgamal script is called. This script is responsible for generating the random hex character string, in accordance with the security scheme described in Section 4.1, encrypting, signing and encoding the string in base64 and embedding the random string inside a Uniform Resource Locator (URL) that points to the web server. This URL is then sent to the generate_qrcode script described in Section 4.3.3.

See Figure **??** for a detailed process flow diagram.

4.3.3 Generating a QR Code

After the encrypt_elgamal script has been run, the generate_qrcode script is called. This script is responsible for embedding the URL received from the encrypt_elgamal script into a QR Code. This is done by using a qrcode module for Python, called qrcode [Loop, Lincoln (2013)].

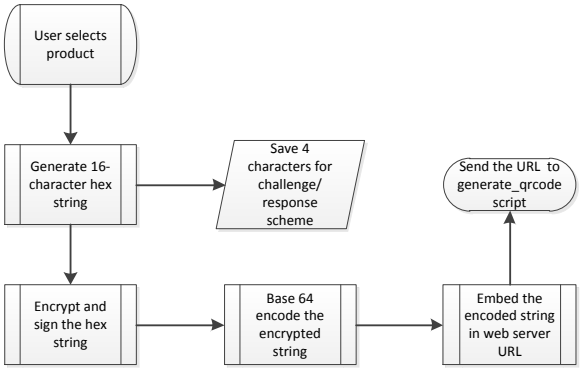

Figure 4.8: The GUI process flow

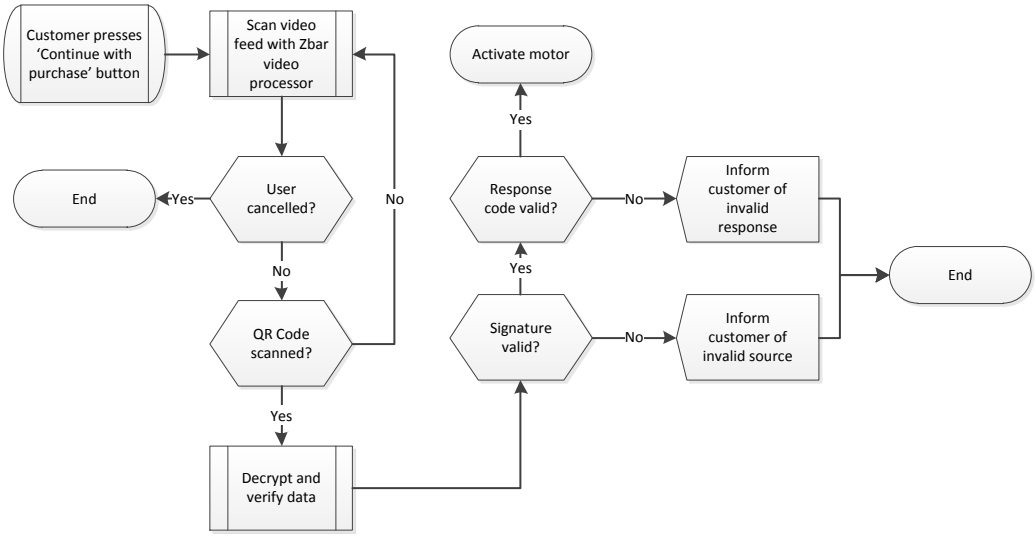

Figure 4.9: The generate_qrcode script process flow.

4.3.4 Reading a QR Code

After the customer has received his QR Code from the server verifying the transaction, the customer may press the button called 'Continue with purchase'. When this is done the read_qrcode script is run.

This script is responsible for reading the customer's QR Code via a web cam, extract the data from the scanned image, decrypting the data and verifying the transaction. See Figure 4.9 for a detailed process flow diagram.

As seen from Figure 4.9, as soon as the 'Continue with purchase' button is

pressed, the script creates a ZBar image processor which scans a web cam video feed for a QR Code (see Section 2.1.1 on page 6 for more information on ZBar). This image processor runs until the user cancels the process or it scans a QR Code.

After the ZBar processor has scanned a QR Code, it sends the retrieved data to be decrypted and verified with the ElGamal algorithm and the vending machine's private key and the server's public key. If the signature is valid and the data contains a valid response and product code, the script activates the correct motor and the customer receives his product.

4.3.5 Near Field Communication

If the customer opts to purchase a product with the Android Near Field Communication (NFC) app, the NFC script is run. This script is based on an example script included in the nfcpy package (see Section 2.2.1 on page 7 for more detail) and is written by nfcpy's creator, Stephen Tiedemann [Tiedemann, Stephen (2013)]. This example script, called snep-test-server, does the following:

- It connects the Raspberry Pi to the NFC controller chip.
- It polls the NFC controller chip for a Simple NFC Data Exchange Format Exchange Protocol (SNEP) message.
- When a SNEP message is read, it extracts the data in the message and presents it to the programmer for further processing and manipulation.

Using this example script allows the vending machine to extract SNEP messages from any source that follows the NFC Forum's standards [NFC Forum (2013)]. For this project, an Android NFC app was written specifically for this purpose (see Section 4.4 for more detail).

After the data is extracted from the NFC source, the script decrypts the data using the RSA algorithm and the vending machine's private key. The vending machine then checks the NFC message's response code.

If the response code is valid, the script then activates the correct motor to dispense the product to the customer.

4.4 Android Application

An Android NFC app was made for this project. It allows a customer to buy a product from the app's product menu and to complete the purchase by swiping his phone across the vending machine's NFC receiver.

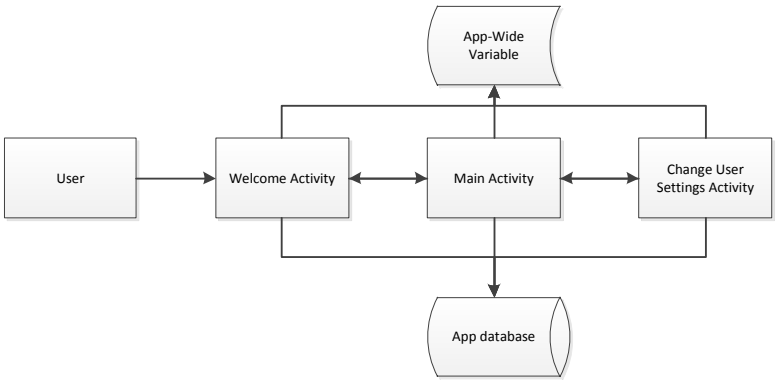

Figure 4.10: The Android NFC app structure.

The app is divided up into three activities (Android's technical term for what is essentially a different window of the app).

These activities, their design and significance are discussed in this section.

See Figure 4.10 for a structure diagram of the app.

4.4.1 Welcome Screen

This activity is the activity that is called when the app is opened. This activities process flow can be seen in Figure 4.11. Figure 4.12 shows a screenshot of the welcome screen.

As seen from Figure 4.11, the activity first checks to see if its the first time the user opens the app. If its not, the activity goes on to the Main Activity. Otherwise, it allows the user to either sign in with an existing profile or create a new one.

When the user signs into an existing profile, the login details he/she provides is stored in a database that is only accessible by this app. When this is done, it makes the login information persistent throughout the lifetime of the app.

These details are also saved into an app-wide variable. This variable is only accessible by the app and is only active while the app is running in the foreground or background. This variable is used to make the app more efficient by not having to read the database every time the user wants to use the app. This variable is only active while the app is running in the foreground or background

The login details saved here are used later by the app's other activities.

4.4.2 Change User Settings

The Change User Settings activity allows the user to change his login details that are saved in the database and the app-wide variable. See Figure 4.13 for a detailed

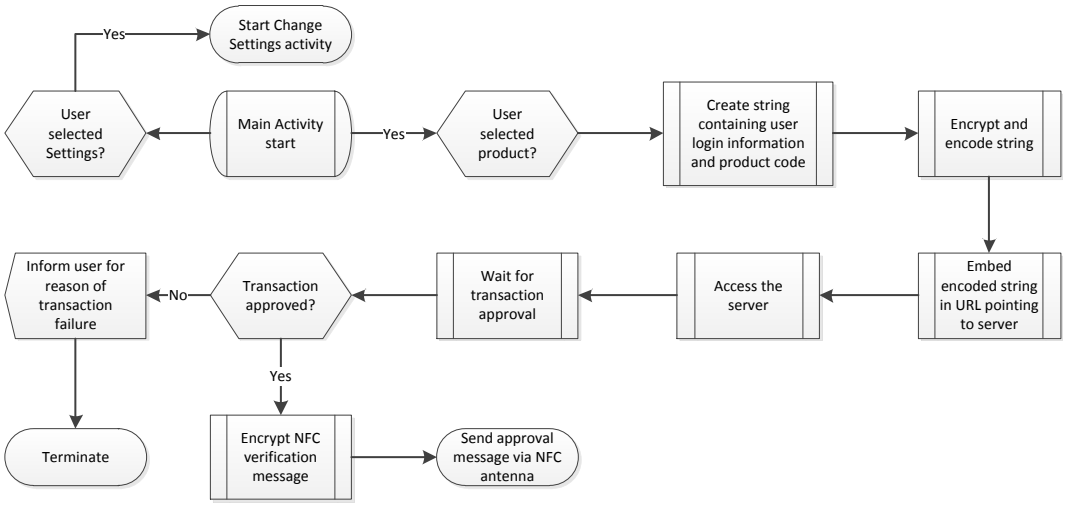

Figure 4.11: The Android NFC app structure.

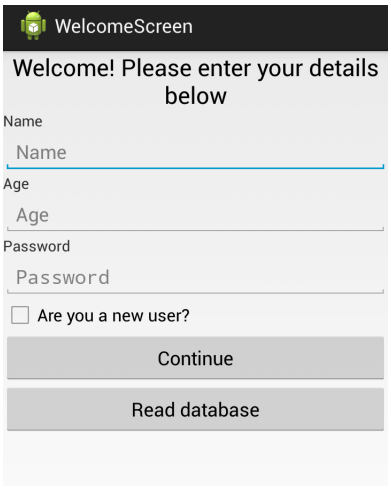

Figure 4.12: A screenshot of the app's welcome screen.

process flow diagram for this activity. Figure 4.14 shows a screenshot of this activity.

When the app enters this activity, it prompts the user to enter his user name and password. When the user presses the 'continue button', the old database entry and app-wide variable is overwritten with the new values the user entered.

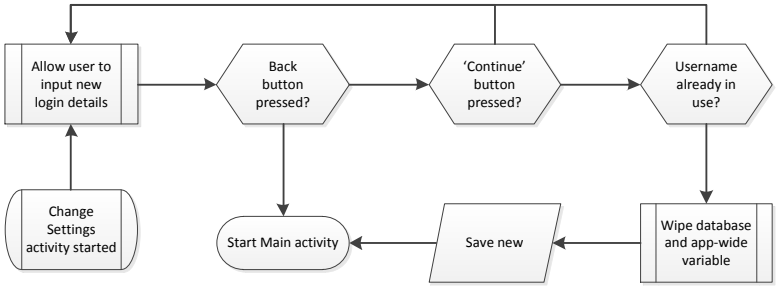

Figure 4.13: The process flow of the Change Settings activity.

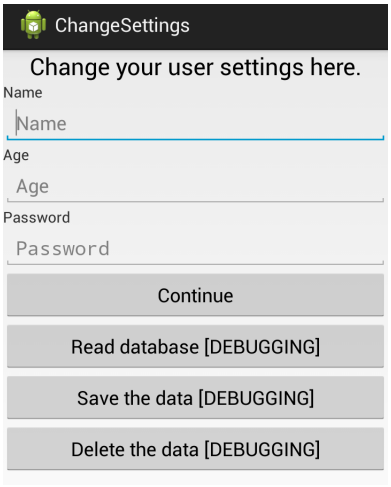

Figure 4.14: A screenshot of the app's change settings activity.

4.4.3 Main Activity

The main activity is the activity which is responsible for encrypting and sending the purchase requests to the web server, receiving and decrypting the purchase approval codes and sending the NFC messages to the vending machines NFC receiver. See Figure 4.15 for a process flow diagram and Figure 4.16 for a screenshot of this activity.

As seen from Figure 4.16, the activity presents the user with a list of products. When the user selects a product to buy, the activity forms a data string by adding the product code, the user's login name and password together. This data string is then encrypted with the server's public key half and then encoded with base64. This encoded string is then embedded into a Uniform Resource Locator (URL) which points to the web server.

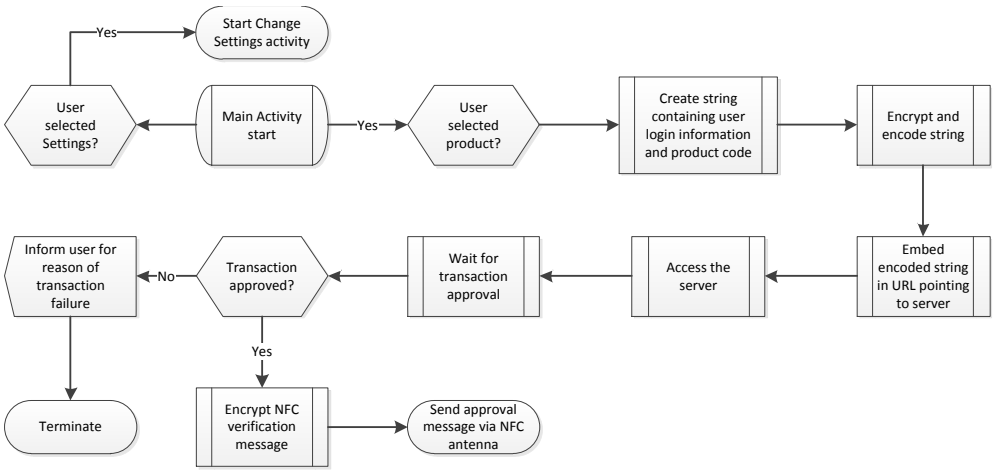

Figure 4.15: The process flow of the Main activity.

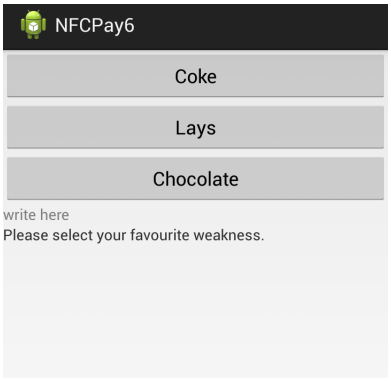

Figure 4.16: A screenshot of the app's main activity activity.

The activity then goes to this URL in the background which prompts the server to process the transaction (see Section 4.2.1 for more details on the server's NFC processes). The server then tells the activity if the transaction has been approved or denied. If it has been denied, the user is informed what was wrong.

If it is approved, the activity activates the cellphone's NFC antennae which transmits an encrypted approval message to the vending machine's NFC receiver.

Chapter 5

Hardware Detail Design

This chapter gives a detailed discussion on the hardware design and configuration of this system.

5.1 Relay Switch Circuit

As explained in Section 3.5.3 in page 18, a relay switch (the Mantech NT72C 12V DC relay [Mantech (2013)]), in conjunction with a 2N2222 Bipolar Junction Transistor (BJT) [ST Electronics (2013)], is used by the Raspberry Pi to switch the motor on and off. See Figure 5.1 for the circuit diagram.

The relevant parameters for these components are [Mantech (2013), ST Electronics (2013)]:

$$P_r = 0.36W$$

$$V_r = 12V$$

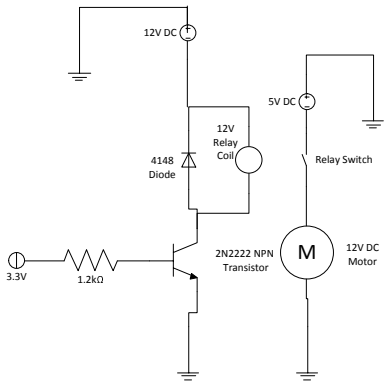

Figure 5.1: 12V relay transistor switch.

$$\beta \approx 10$$

$$V_p = 3.3V$$

From the relays power dissipation and voltage, its current draw is found by

$$I_r = \frac{P_r}{V_r}$$

which gives a current draw of

$$I_r = 0.03A$$

Taking the BJT's current amplification as roughly 10, the current draw from the Pi to the base of the BJT is given by

$$I_b = \frac{I_r}{\beta} = \frac{I_c}{\beta}$$

This gives a current draw of

$$I_b = 0.003A = 3mA$$

The maximum current draw from the GPIO pins is 16mA [elinux.org (2013)], though this is not recommended as the Pi does not have any current limiting or over-current protection. Therefore, a current draw of 3 mA is completely safe.

To limit the current draw from the Pi, a base resistor must be added between the Pi's GPIO pin and the BJT's base. With a current draw of $3mA$ and a voltage of $3.3V$, the resistor size is found by Ohm's law as follows

$$R_b = \frac{V_p}{I_p}$$

which gives

$$R_b = 1.1k\Omega$$

Compensating for tolerances by adding 10%, the resistor size is set to

$$R_b = 1.2k\Omega$$

which draws a current of

$$I_b = 2.75mA$$

which is still within the acceptable range.

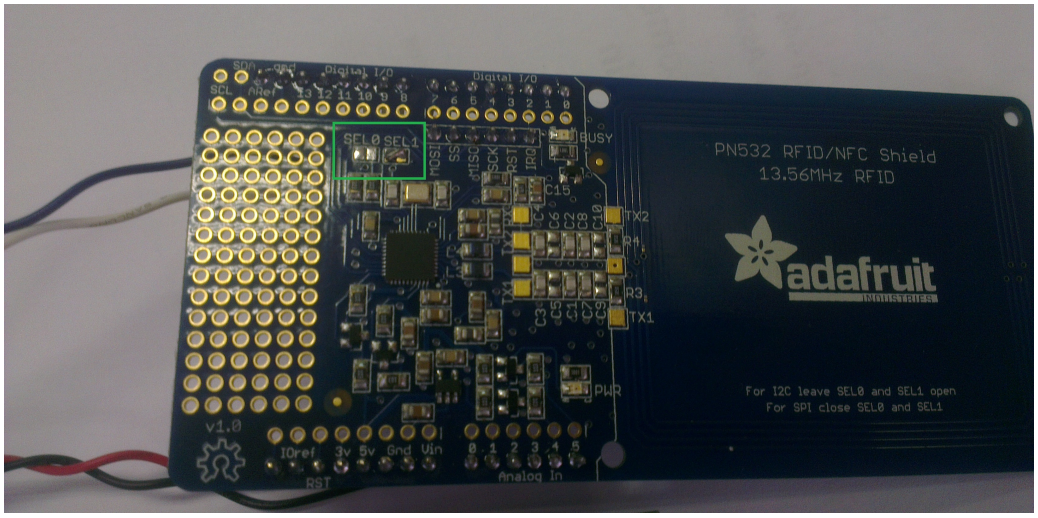

Figure 5.2: The location of the SEL1 pads (highlighted in green).

5.2 NFC Chip

The Near Field Communication (NFC) chip used, is the Adafruit NFC shield for an Arduino Uno microcontroller [Adafruit Industries (2012)]. See Section 3.3 on page 16 for more details on the controller.

The shield was designed and built to be used by an Arduino Uno microcontroller. Therefore, some modification to the chips connections had to be made before it would be able to communicate with the Pi.

By default, the chip was made to communicate with an Arduino with the Inter Integrated Circuit (I^2C) communication protocol. However, the component manufacturers have added connection pads to the chip that, when soldered, allows the chip to communicate via Serial Peripheral Interface (SPI) or Transistor Transistor Login (TTL).

The libnfc library is currently only configured to allow communication via an Universal Asynchronous Receiver Transmitter (UART). Therefore if the chip was modified to communicate via its TTL interface, which is compatible with a UART.

To do this, the 'SEL1' pads were soldered closed (see Figure 5.2). With this done, the chip can serially communicate with the Raspberry Pi's UART interface.

The connections between the Pi and the NFC controller are as follows:

5.3 libnfc Setup on the Raspberry Pi

Before libnfc could be built and configured, the communication between the NFC controller and the Pi needed to be finalised. To do this, the Pi's UART needed to

Table 5.1: Connections between the Raspberry Pi and the NFC Controller chip.

NCF Controller Pin	Raspberry Pi Pin
5V pin	5V (pin 4)
Ground pin	Ground pin (pin 6)
SS	UART0 TXD (pin 8)
MOSI	UART0 RXD (pin 10)

be freed up. By default, the Raspberry Pi uses its UART to serially write out its booting information. Therefore, to allow the Raspberry Pi to communicate with the NFC controller via its UART0 interface, it was necessary to modify some of its configuration files. To do this, the Adafruit tutorial was followed [Townsend, Kevin (2012)].

To do this, the file '/boot/cmdline.txt' and '/etc/inittab' had to be edited to contain the following lines of code:

cmdline.txt

```
dwc_otg.lpm_enable=0 console=tty1
```

inittab

```
#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

After this has been done, the libnfc package could be configured and installed and configured to work with the Pi's UART interface.

Thereafter, the Pi is ready to receive NFC messages from the NFC shield.

5.4 Vending Machine Unit

A small vending machine unit was constructed for demonstration purposes. It is designed to house all the vending machine components, i.e. the two DC motors, the two coils, the Raspberry Pi central controller, the NFC shield and the webcam.

Four 10mm vent holes were added at the sides to improve air circulation and to provide external wire access, while a larger 60mm vent hole was also added to allow for an external 60mm desktop computer fan to be added.

The unit is made from 1.6mm thick mild steel and the components were cut and bent by Fabrinox, Paarl and welded together by the Electric and Electronic Engineering Department of Stellenbosch University's workshop.

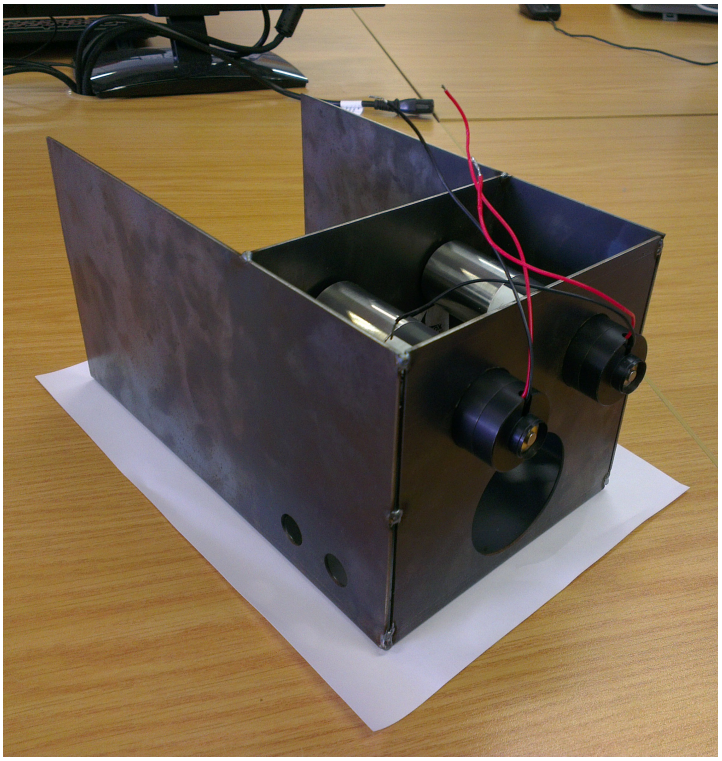

Figure 5.3: The complete vending machine unit.

See Figure 5.3 for a picture of the complete vending machine unit with the two DC motor inserted. See Appendix A for detailed design drawings of the vending machine unit.

5.5 Webcam

As discussed in Section 3.4, a Sony PS2 Eye Toy webcam was attached to the Raspberry Pi. This allows the vending machine to scan a live video feed for a QR Code. Its already compatible with the Raspberry Pi and therefore requires minimal configuration to begin working.

However, the camera needs to be plugged into a Universal Serial Bus (USB) port on the Pi itself and not into a USB hub. This is done in order to prevent the hardware timing issues that are introduced to the system when a USB hub is used.

5.6 Motor and Coil

MOET NOG GEDOEN WORD.

Chapter 6

System Tests

In this chapter, the tests conducted to measure how well the system measures to the original goals and objectives set out in Section 1.

The tests conducted are the voltage and current limits of the relay switches and user stories where scenarios are created and the response of the system is given.

6.1 Transistor Switch

MOET NOG GEDOEN WORD

6.1.1 Current and Voltage Limits

MOET NOG GEDOEN WORD

6.2 User Tests

The functionality of the system is tested by exposing it to different user scenarios and seeing what the outcome is. These tests are performed and discussed in this section.

6.2.1 Test 1

In this user story, the customer decides to pay with a Quick Response Code (QR Code). He/she does not yet have a user account on the database and has no credits laoded. Figure 6.1 shows the user story.

The process is as follows: The user first selects which product to buy from the Raspberry Pi's User Interface (GUI). The Pi then generates a Quick Response Code

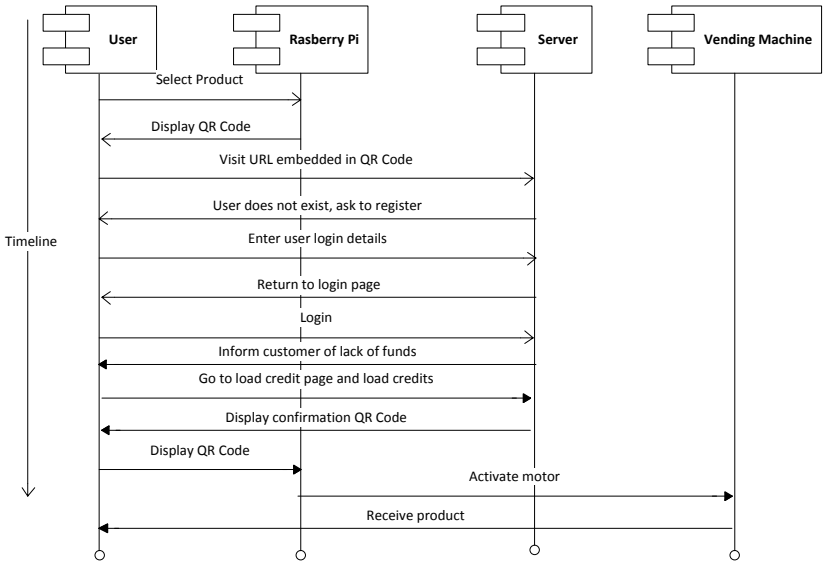

Figure 6.1: The first user scenario.

(QR Code) that the user scans with any barcode scanning application. The Universal Resource Locator (URL) embedded in the QR Code then takes the customer to a web page located on the server.

Not having a user account, the user clicks on the lick to create a new user profile. After entering valid user information, the customer is returned to the login page where he/she can log in with his new login credentials.

After logging in, the server displays to the customer that he/she does not yet have any credits loaded onto his account. The customer then goes to the load credit page where they load some money. When this is done, the server displays a confirmation QR Code on the customer's cellphone screen, verifying the transaction.

The customer then presses the continue button on the GUI, which starts scanning the web cam feed for the customer's QR Code. After this is done, the Pi activates the correct motor and dispenses the product.

6.2.2 Test 2

In this scenario, a customer tries to use a QR Code from a previous transaction. Figure 6.2 shows the user story.

To begin, the customer selects a product to buy, after which the Raspberry Pi displays a QR Code. However, instead of scanning the QR Code and paying for a new product, the customer tries to use an old QR Code that the server sent him/her after a successful transaction previously.

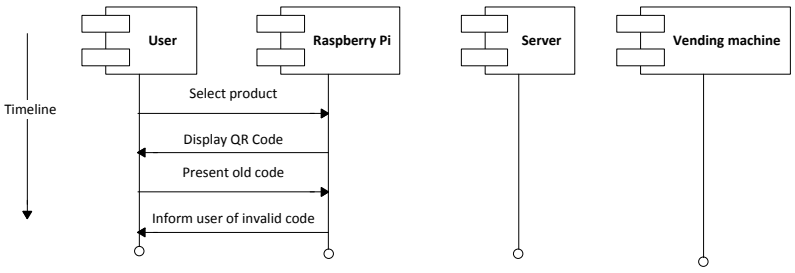

Figure 6.2: The second user scenario.

The customer tells the Raspberry Pi to scan his old QR Code. The Raspberry Pi does this, but the old QR Code does not contain a valid response to the Raspberry Pi's challenge. Therefore, the transaction is valid and the customer is informed of his invalid QR Code.

6.2.3 Test 3

In this scenario, the user opts to buy a product using the Android Near Field Communication (NFC) app. However, a new user profile must first be created. The user story can be seen in Figure 6.3.

After the user opens the app, he/she is presented with a login screen. The user enters his login credentials and checks the box that confirms that he/she is a new user. The app then contacts the server with a user creation request. Provided that the username is not already in use, the server sends the app a confirmation.

The app then returns the user to the main menu where the user can pick a product to buy. After picking a product, the app contacts the server with a purchase request. If the transaction is successful, the server sends the app a confirmation code which the app transmits via the cellphone's NFC antenna.

The user then swipes his phone across the vending machines NFC receiver. The vending machine then activates the motor and the user receives his product.

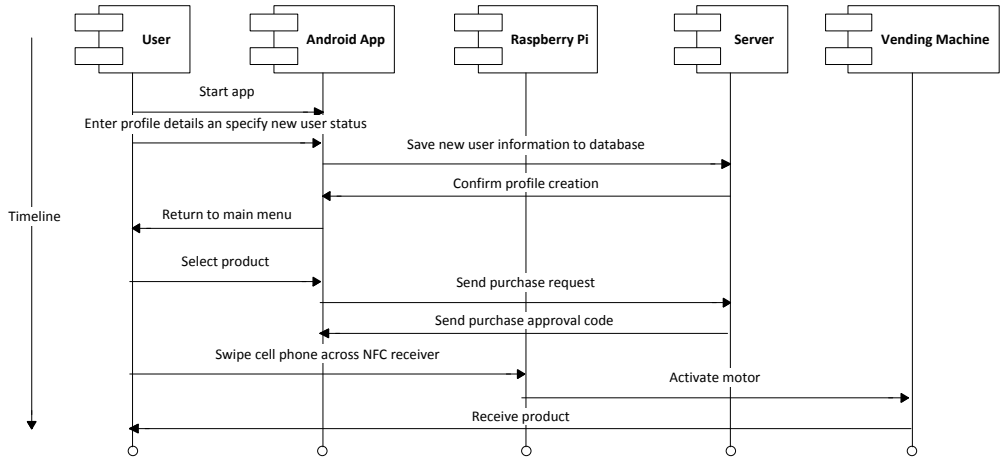

Figure 6.3: The third user scenario.

Chapter 7

Conclusion

This chapter concludes this project design report. It gives a brief discussion on the complete system and measures it against the project objectives set out in Section 1.4 of this report. Lastly, it discusses possible improvements that can be made to the system.

7.1 System Performance

The project objectives set out in Section 1.4 of this report are repeated here for convenience. They are:

- The system must make provision for both NFC and Quick Response Code-based (QR Code) payments.
- The system must make use of a web server based in the cloud.
- An Android application must be made that will allow transactions to be completed using NFC.
- A demonstration model vending machine must be designed and constructed.
- All the data transfers between the user's cellphone and the server must be encrypted.
- Extra layers of security must be added.

In the final system, both QR Code and NFC technology have been implemented. It was found that the NFC option has a faster transaction time between product selection and when the product is dispensed. However, the NFC option is currently limited to cellphones that have NFC antennas and are based on the Android Operating System.

It was found that the QR Code payment option typically has a slower transaction time. This is mainly due to some bottlenecks that the system currently has. These bottlenecks include the Raspberry Pi's moderate processing power, which increases the encryption time and the time it takes the Pi to process and decode a live video stream from the web cam. Extra work can go into optimising the system to make it run faster.

Furthermore, a server has been made that runs in the cloud. This server interacts with a customer's cellphone through the Android NFC app that was made or the cellphone's internet browser. All data transactions between the server and the customer's cellphone are encrypted with an asymmetric encryption scheme with extra layers of security added.

Lastly, a working demonstration vending machine was designed and made. It houses all of the vending machine components and allows a customer to buy one of two products using his cellphone.

7.2 Future Work

The vending machine system that has been designed in this project has been designed to work as planned. With that being said, there is potential to expand upon the work that has been completed thus far.

This section discusses some potential improvements and additions that can be made to this system

7.2.1 Commercialisation

The vending machine currently uses faux money which has no real-world value. Therefore, the system in its current state is not ready to be deployed across Stellenbosch University's campus.

To commercialise the system will require that customers be able to load money from their bank accounts, credit cards or student accounts onto their vending machine user accounts.

This will not be a simple task as this will require integration with the financial institutions of South Africa.

7.2.2 Polish

This project focused on the practical and functional aspects of the complete system and almost no attention was given to the aesthetics of the complete system. Therefore, some extra work can go into making the web server and Android NFC app more pleasant to use.

7.2.3 Integration

One important aspect that must be focused on to increase the odds of making this project more commercially successful, is its integration with current vending machines, i.e. add a cashless option to current vending machines that are restricted to cash.

This integration may become complicated, as each vending machine manufacturer uses different components, such as different size motors and control units.

7.2.4 More Payment Options

Some work can go into expanding the number of payment options available to the user.

This is a feasible option, seeing as the Raspberry Pi has 28 General Purpose Input Output Pins, of which only 4 are currently being used. These 24 other pins can be used to control other payment options.

Some options that may be considered are Bluetooth, Short Message System (SMS), Instant Messaging (such as WhatsApp and BlackBerry Messenger) and Unstructured Supplementary Service Data (USSD).