UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Development of Cashless Vending Machine

by

JC Lock
16016548

Mechatronic Project 488

Final Report

Department of Mechanical and Mechatronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602.

Supervisor: Prof. G-J van Rooyen

September 2013

Declaration

I, the undersigned, hereby declare that the work contained within this report is my own, original work.

Signature: .....
JC Lock

Date: .....

MECHATRONIC PROJECT 488: SUMMARY

Student: JC Lock
Co-worker: N/A

Title of Project
Development of a Cashless Vending Machine.
Objectives
Program and make a model vending machine which accepts payments made via a cellphone.
Which aspects of the project are new/unique?
The easy use of simple cellphone technologies most students currently have built into in their cellphones, such as NFC and QR Codes.
What are the findings?
A working test model was built which accepts faux money paid with either QR Codes or NFC. All the necessary security measures, i.e. encryption and challenge codes, were added as well as a central web server that tracks user transactions.
What value do the results have?
The results show that the machine is working reliably. Some improvements can be made, but the overall machine is working as expected.
If more than one student is involved, what is each one's contribution?
Only one student was involved in this project.
Which aspects of the project will carry on after completion?
This project has been successfully completed and will therefore not continue.
What are the expected advantages of continuation?
N/A
What arrangements have been made to expedite continuation?
N/A

Student	Date	Lecturer

Abstract

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

NFC	Near Field Communication
QR Code	Quick Response Code
SU	Stellenbosch University
RFID	Radio Frequency Identification
USSD	Unstructured Supplementary Service Data
ZXing	Zebra Crossing
OS	Operating System
NDEF	NFC Data Exchange Format
SNEP	Simple NDEF Exchange Protocol
LLCP	Logical Link Control Protocol
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
EC2	Elastic Compute Cloud
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
i^2c	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver Transmitter
SPI	Serial Peripheral Interface
GPIO	General Purpose Input Output
BJT	Bipolar Junction Transistor

Variables

I	Current	[A]
P	Power	[W]
V	Voltage	[V]
R	Resistance	[Ω]

Variable Subscripts

b	Base
p	Raspberry Pi
r	Relay

Constants

β	Transistor Current Amplification

Chapter 1

Introduction

This is the introduction chapter to the project. In this chapter the problem statement is given along with existing solutions to the problem. The goals and objectives of this project are given. Lastly, an overview of the rest of this report is given.

1.1 Problem Statement

The vending machines currently used at Stellenbosch University (SU) make exclusive use of cash transactions. These vending machine systems are the de facto standard throughout the world and have been for the largest part of the last two centuries, but they do have one drawback: they require a hard cash transaction to take place and 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 that has been identified is for a vending machine that accepts cashless transactions.

1.2 Existing Solutions

Currently there are plenty of cashless payment options 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.

1.2.1 Credit and Debit Cards

An well-known and convenient alternative to cashless solutions are the plastic cards most modern adults carry around. This is especially true in first world countries

with mature banking systems.

The great advantage these cards holds over the alternatives are that they are relatively easy to get from a bank and that they have become very reliable.

A possible disadvantage is that such systems may become costly and complicated to implement because of each banks different security and transfer protocols.

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 swipe and it it not required that a password be entered.

However, this leads to some security concerns because if the card gets stolen, the thief can use the money on the card for his/her own gain. Thankfully though, these cards most commonly work with pre-loaded money, so provided that there wasn't too much money loaded onto the card, the theft victim will not be too badly off.

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 transfer money from one account to another.

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, bread, even the monthly rent. Its currently regarded as the most advanced mobile payment platform in the world [Jack and Suri (2011)].

An advantage of implementing such a system is that it is proven to work and is usable by almost any cellphone.

The disadvantage is that it requires third party vendors to provide systems and services, which will 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 likely candidate to become the standard method of mobile payment. Especially in Europe and North America, there have been significant advances in making

this payment method a more attractive solution, with Google making the largest contribution with the addition of NFC protocols to its Android platform.

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 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), Valentina Pasquali and Denise Bedell (2013)], however, mobile and card transactions are expected to surpass cash as the leading payment method by 2015 [Harry Wollop (2010)].

The main goal of this project is to develop a vending machine that will make use of this increase in cashless payment by adding the option to pay for a product with a customer's cellphone.

1.4 System Objectives

The system objectives are:

- Must have at least two methods of mobile payment.
- The solution must be built while keeping commercialisation in mind.
-

1.5 Report Structure

In this report, some background information on all the technology, concepts and programs used in this project is given. Then the overall system design is discussed, which is followed by a discussion on the detailed design of the whole system. Then the system tests are discussed and analysed, which is then followed by a discussion on the complete system and a project conclusion.

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), web server and encryption algorithms.

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 [**?**]. The technology has since evolved and matured and is today widely used in the media industry for storing some data, such as a web address or phone number. See Figure 2.1 for an example of a QR code.

Figure 2.1: Example of a QR Code

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

scan a QR Code requires a camera that can produce a digital image of appropriate quality. This image is then processed by a QR Code library, e.g. the ZXing library (see section 2.1.1), 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 web cam and the ZXing library.

2.1.1 Zebra Crossing Library

The Zebra Crossing Library (ZXing for short) is a well-known QR Code coding and decoding library. It is commonly built into smart phone applications to decode QR Codes embedded inside a static image or a video stream, but 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. A desktop version, called ZBar, is also available. 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 has evolved to be available to and be part of millions of people's lives.

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 to close to one another, typically around 4 centimeters.

This technology is commonly found in modern cellphones. However, recently the technology has been ported to other platforms such as a 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 a open-source library for Linux systems [Libnfc Team (2013*b*)]. It allows a desktop computer to communicate with a NFC device based on the 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.

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 [Stephen Tiedemann (2013)].

2.2.2 Android

Google's Android operating system is currently the most widely used cellphone operating system being used word wide with, an estimated 80% market share [Darrel Etherington (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 aggressively promotes the use of NFC as an alternative payment option in modern retail outlets with applications, such as Google Wallet [Balaban, Dan (2012)].

2.2.3 NFC Communication Protocols

2.2.4 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 frequency of 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

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 handle Hypertext Transfer Protocol (HTTP) GET and POST requests.
- Integration with SQL databases, e.g. MySQL, SQLite3, etc.
- Hypertext Markup Language (HTML) template design feature.
- Makes provision for the execution of Python scripts.
- It is fully scalable to commercial level servers.
- Django has an offline server debugging function available.

This framework is expandable to commercial size servers that is 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 Django authors 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 handing functionality. These applications may 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). It allows a user to rent a virtual computer from AWS from which to run their own applications. These applications are most commonly web-based, in other words the virtual machines run a web server that is accessible by anyone around the world.

2.3.3 Apache

Apache is a popular web server application available on most platforms. For simplicity, the details of Apache will not be discussed. However, a very notable feature of Apache is that it is designed 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 currently the most popular web server application in use today, with an estimated 53.4% of the world's servers running on apache [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.

2.4 Encryption

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

Encryption is most commonly used where sensitive information is being transmitted, e.g. banking codes, personal e-mails, etc.

There are two main encryption schemes, namely symmetric and asymmetric encryption.

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. A famous example of symmetric encryption is the Enigma cipher machine used by Nazi Germany during the Second World War.

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, such as known and chosen plaintext attacks [Biham (1994)].

2.4.2 Asymmetric Encryption

More recently, asymmetric encryption, also known as public-key encryption, has become the new standard for 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 on the receiver's side.

This private and public key pair 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). The public key half of the pair is used to encrypt data and may be publicly distributed to anyone who wants one (in practise this is done differently 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. 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 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, this 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 a 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 peoples mail, that 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 special keys to everyone who wants one. These special 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 hasn't 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 her 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 is a large prime modulus'. [Jeffrey S. Leon (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 (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' [Jeffrey S. Leon (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 public domain and is free to use by anyone. It is maintained by the PyCrypto Team [PyCrypto Team (2013)].

2.4.4 Base 64

Base 64 is an encoding scheme which takes binary data and encodes it to ASCII characters. This is most often used where the output of encrypted text is unreadable binary data and ASCII characters are preferred because they are easier to read and transmit.

It is also important to note that the the output of the base 64 encoding is approximately 33% longer than the input string.

Here is an example of a base 64 encoded string:

Original text:
Hi, I'm a base 64 encoded string!
Base 64 encoded output:
SGksIEknbSBhIGJhc2UgNjQgZW5jb2RlZCBzdHJpbmch

Chapter 3

System Design

3.1 System Overview

The vending machine system consists of three main parts, namely the QR Code component, the NFC component and the actual vending machine.

Figure 3.1 and 3.2 gives a diagrammatical layout of the complete system. It shows the interactions between the different sub-components of the complete system.

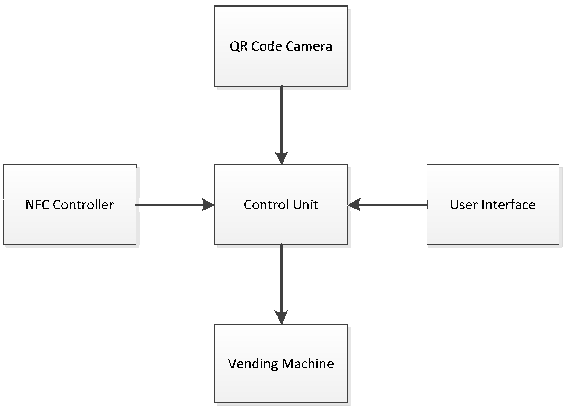

Figure 3.1: System overview from the control unit's perspective

It can be seen that the complete system is divided up into five main parts, namely

1. Control unit.
2. NFC controller.

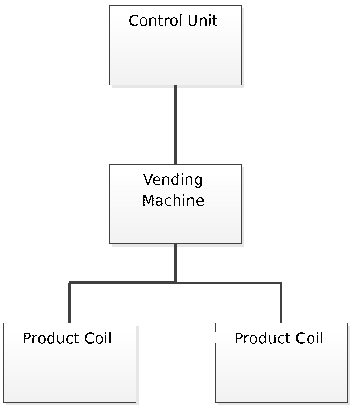

Figure 3.2: System overview from the vending unit's perspective

3. QR Code camera.
4. Vending machine unit.
5. Vending machine coils.

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

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.

3.2.1 Arduino Uno

The Arduino Uno is popular open-source microcontroller (see Figure 3.3).

It is based on the 8-bit Atmel ATmega328 ARM microprocessor. Its official specifications are [**?**]:

Operating Voltage:	5 V
Processor:	Atmel ATmega328 (clocked at 16 MHz)
GPIO Pins:	14 (6 of which are PWM enabled)
Memory:	32 kB Flash, 2kB SRAM, 1kB EEPROM

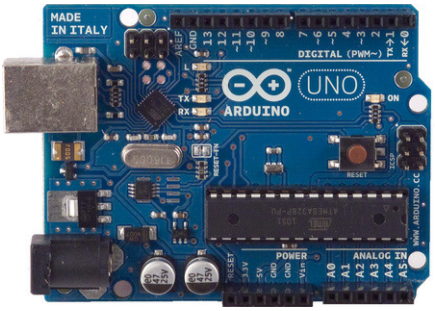

Figure 3.3: Picture of an Arduino Uno microcontroller [man (2013*b*)]

Communication: i^2c, UART, SPI.
Price: R310.00

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, but remarkably simple, version of C and comes with its own Integrated Development Environment (IDE). This, along with its relatively low cost and adequate specifications, makes the Arduino Uno an attractive option for this project.

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 for for a control unit for the vending machine.

The Pi is made 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 [eLinux (2013)]:

Communication: SPI, UART, USB, i^2c, Ethernet
Memory: 512 MB RAM
Processor: ARM 6 clocked at 700 MHz
Video: HDMI video output
GPIO: 28 pins
Price: R400.00

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 instead of the Arduino Uno for the following reasons:

- The Pi has more processing power (700MHz vs 16MHz).
- The Pi's ability to easily interface with desktop computer peripheral hardware, such as a keyboard, mouse and webcam.
- The excellent support structure in place and the information on existing and ongoing projects that are available on the internet.
- The Pi can output its video feed to a computer screen, which makes it possible to add a simple user interface (GUI) for uder interaction.

Stated simply, the Raspberry Pi is a compact, traditional desktop computer which makes it very easy to work with, and its low price makes it an excellent choice for use as the central controller for the vending machine.

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 ahead 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.

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 RFID cards, such as student or staff cards, as 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 using 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.
- There was one lying in the supply cupboard.

There is currently a camera add-on available for the Raspberry Pi, but this is relatively expensive and (approximately $30 [Adafruit (2013)] versus the Pi's $35), 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 traditional coil mechanism is used. Such systems are the most familiar and simple methods of dispensing goods. See Figure 3.4 for an example.

Figure 3.4: Example of vending machine coil system [**?**]

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 12V DC mootors from Faulhaber [man (2013*a*)]. Although these motors are rated for 12V, 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 12V 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 12V. The Raspberry Pi can deliver a maximum of 5V. Therefore it was decided that the relay will be permanently connected to a 12V DC supply, but will be switched by a 2N2222 transistor, which is controlled directly from the Pi's GPIO pins (see section 4.6 for a detailed discussion).

This allows the Pi to directly control the motors and due to the circuits construction, the Pi is protected from the relatively high voltages and currents involved in the working of the motor nd 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). Its made of 1.6mm mild steel plate and was made by Fabrinox, Paarl. See Appendix A for detailed manufacturing drawings.

Chapter 4

Detail Design

4.1 Relay Switch Circuit

As explained in section 3.5.3, a relay switch (the Mantech NT72C 12V DC relay [man (2013*c*)]), in conjunction with a 2N2222 Bipolar Junction Transistor (BTJ) [**?**], is used by the Raspberry Pi to switch the motor on and off. See Figure 4.1 for the circuit diagram.

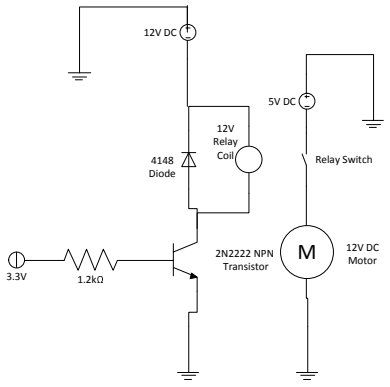

Figure 4.1: 12V relay transistor switch.

The relevant parameters for these components are [man (2013*c*), **?**]:

$$P_r = 0.36W$$

$$V_r = 12V$$

$$\beta \approx 10$$

$$V_p = 3.3V$$

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.

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 for some background information). 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 for some background on EC2 and Section 2.3.3 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 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 4.5 for more details).

The website apps are discussed in this section.

display_qrcode

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

As seen from Figure **??**, 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/her cell phone'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 for some background information on base 64 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 vending machine's public key (see Section 2.4.2 for more detail on public and private keys) and, following the security code scheme described in Section 4.3, 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/her 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 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 base 64. After this is completed, the app embeds this data into a QR Code, which is returned to the customer's cell phone screen.

load_money

This app allows the customer to load money onto his/her 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/her account.

nfc

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

As seen from Figure **??**, 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/her account. If this check fails, the customer is informed of his/her 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 a confirmation code, according to the security code scheme spesified in Section 4.3, 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 webpages.

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
```

4.2.4 nfc

4.2.5 qr code

4.3 Security Scheme

4.4 vending program

4.4.1 nfc

Because the Arduino version of this shield is locally available, and the cost issues related to importing a NFC chip that is made for the Raspberry Pi, the Arduino version was bought for R780.00. Its Transistor-Transistor Logic (TTL) serial interface was configured in such a way so that it can serially communicate with the Raspberry Pi's UART interface. It can be powered by the 5V output pin from the Raspberry Pi.

4.4.2 qr code

4.5 Android app

4.6 Motor and coil

Chapter 5

System Tests

5.1 Transistor Switch

5.1.1 Current and Voltage Limits

5.1.2

Chapter 6

Conclusion

Appendix A

Vending Machine Drawing

List of References

(2013 September*a*). 12V DC Motor Spec Sheet. http://www.webstudies.sun.ac.za/webct/urw/lc4130011.tp0/cobaltMainFrame.dowebct.

(2013 September*b*). *Arduino Uno*. http://arduino.cc/en/Main/arduinoBoardUno.

(2013 September*c*). *NT72 Spec Sheet*. http://www.mantech.co.za/datasheets/products/NT72C_NHG.pdf.

Adafruit (2013 September). Raspberry Pi Camera Board. http://www.adafruit.com/products/1367.

Adafruit Industries (2012 November). Adafruit NFC Controller Shield. http://www.adafruit.com/products/789.

Balaban, Dan (2012 August). Google unveils cload-based revamp of Wallet but keeps NFC technology. *NFC Times*. http://nfctimes.com/news/google-unveils-cloud-based-revamp-wallet-keeps-nfc-technology.

Biham, E. (1994). New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, vol. 7, no. 4, pp. 229–246.

Chandler, Nathan (2012 March). What's the difference between RFID and NFC. *How Stuff Works*. http://electronics.howstuffworks.com/difference-between-rfid-and-nfc.htm.

Darrel Etherington (2013 August). Android Nears 80% Market Share In Global Smartphone Shipments, As iOS And BlackBerry Share Slides, Per IDC. *Tech Crunch*. http://techcrunch.com/2013/08/07/android-nears-80-market-share-in-global-smartphone-shipments-as-ios-and-blackberry-shar

Django Software Foundation (2013 September*a*). List of Django sites. http://www.djangosites.org/.

Django Software Foundation (2013 September*b*). Why does Django exist? https://docs.djangoproject.com/en/1.5/faq/general/#why-does-this-project-exist.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, vol. 31, no. 4, pp. 469–472.

eLinux (2013 September). Raspberry Pi Specifications. `http://elinux.org/RPi_Hardware`.

elinux.org (2013 September). Low Level Peripherals. `http://elinux.org/RPi_Low-level_peripherals#Referring_to_pins_on_the_Expansion_header`.

Google Play (2013 September). Barcode Scanner. `http://www.distimo.com/iq/app/google-play-store/zxing-team/barcode-scanner?country=us&category=723&device=`.

Harry Wollop (2010 April). Cash to be used in fewer than half transactions by 2015. *The Telegraph*. `http://www.telegraph.co.uk/finance/personalfinance/consumertips/banking/7585891/Cash-to-be-used-in-fewer-than-half-transactions-by-2015.html`.

Humphrey, D.B. (2004). Replacement of cash by cards in us consumer payments. *Journal of Economics and Business*, vol. 56, no. 3, pp. 211–225.

Jack, W. and Suri, T. (2011). Mobile money: the economics of m-pesa. Tech. Rep., National Bureau of Economic Research.

Jeffrey S. Leon (2008 March). The ElGamal Public Key Encryption Algorithm. `http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/el-gamal.pdf`.

Libnfc Team (2013 September*a*). Compatible Hardware. `http://nfc-tools.org/index.php?title=Devices_compatibility_matrix`.

Libnfc Team (2013 September*b*). Libnfc. `http://nfc-tools.org/index.php?title=Main_Page`.

Netcraft (2013 June). June 2013 Web Server Survey. `http://news.netcraft.com/archives/2013/06/06/june-2013-web-server-survey-3.html`.

NFC Forum (2013 September). NFC Forum. `http://www.nfc-forum.org/specs/spec_list/`.

ov511 (2013 September). ov511 Known Cameras. `http://alpha.dyndns.org/ov511/cameras.html`.

Polet, Nick (2013 April). Deploying Django on AMazon EC2 server. *Nick Polet's Blog*. `http://nickpolet.com/blog/1/`.

PyCrypto Team (2013 September). PyCrypto. `https://launchpad.net/pycrypto`.

Python Package Index (2013 September). Python Package Index. `https://pypi.python.org/pypi`.

Soon, T.J. (2008). Qr code. *Synthesis Journal*, pp. 59–78.

Stephen Tiedemann (2013 September). Nfcpy. `https://launchpad.net/nfcpy`.

Valentina Pasquali and Denise Bedell (2013 January). Payments Volumes Worldwide. `http://www.gfmag.com/component/content/article/119-economic-data/12528-payments-volumes-worldwide-new.html`.

Walton, C.A. (1983 05). Portable radio frequency emitting identifier. Available at: `http://www.google.com/patents/US4384288`

Weinstein, L.S. (2009). Tflâs contactless ticketing: Oyster and beyond. *Transport for London, London, UK (September 2009)*.