

PRICE AND STOCK CHECKING ROBOT IN RETAIL STORE

BY

TOH MIKI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

Supervisor's name

Date: _____

Date: _____

PRICE AND STOCK CHECKING ROBOT IN RETAIL STORE

By

TOH MIKI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

DECLARATION OF ORIGINALITY

I declare that this report entitled “**PRICE AND STOCK CHECKING ROBOT IN RETAIL STORE**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Mr Teoh Shen Khang who have given me this bright opportunity to engage in an robotics project. It is my first step to establish a career in robotics field. A million thanks to you.

To a very special person in my life, Johndy Chuah, for his patience, unconditional support and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

Robotic products are getting more and more popular in today's world. Robot is created to replaced manpower as robot is said to be more accurate and almost error free when performing task. The price and stock checking robot is an automated robotic product created to serve the retail store.

Problems that happen daily in retail store includes misplaced price tag, the price stated on price tag is not up-to-date, there are still stocks in the garage but the shelves are empty. These mistakes cause losses in profit to be earn by the business owners. Furthermore, employee could have utilized these boring auditing hours in more meaningful and important tasks.

The proposed system will provide a much better way of completing these auditing jobs. It will be more accurate than human and could greatly ease the employees work. In a long term basis, it will create more profit to the business.

Prototyping methodology will be adapted. The functionality of this robot will be scaled down to several small tasks and added slowly to each prototype. The robot will mainly do stock and price check. After the check, the report will be sent to the GUI.

TABLE OF CONTENTS

TITLE PAGE	i
TABLE OF CONTENTS	vii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Project Scope	3
1.3 Project Objective	3
1.4 Impact Significance and Contributions	4
1.5 Background Information	5
1.6 Proposed Approach	6
1.7 Highlight of what had been achieved	7
1.8 Report Organization	8
CHAPTER 2 LITERATURE REVIEW	9
2.1 Background	9
2.2 Barcode	9
2.3 Existing Bar-code Scanning Solution	10
2.4 Optical Character Recognition (OCR)	11
2.4.1 Tesseract OCR	11
2.5 Robot Operating System	13
2.5.1 Component of ROS	13
2.5.2 ROS Ecosystem	14
2.5.3 Advantages of ROS	14
2.6 Review and Comparison on Existing Product	16
2.6.1 Bossa Nova's Shelf-Scanning Robot	16
2.6.2 Simbe Robotics Tally	17
2.6.3 Comparison of Existing and Proposed System	19
2.7 Chapter Summary	20

CHAPTER 3	SYSTEM DESIGN	21
3.1	System Flowchart	21
3.2	The Use Case Diagram	24
3.3	The ROS Block Diagram	25
3.4	Block Diagram	26
3.4.1	Barcode Scan	26
3.4.2	OCR Block Diagram	27
3.5	ERD Diagram of Database	28
3.6	The Design of GUI	29
CHAPTER 4	METHODOLOGY AND TOOLS	30
4.1	Methodology	30
4.2	System Design Specifications	31
4.2.1	Hardware Tools	31
4.2.2	Software Tools	35
4.3	System Architecture Design	40
4.4	Project Timeline	41
CHAPTER 5	IMPLEMENTATION, TESTING AND RESULTS	42
5.1	Testing Component	42
5.2	Robot Control	44
5.2.1	Step I Robot Walking	44
5.2.2	Image Capture after Price Tag Detection	45
5.3	Step II- Barcode Scanning (Product Code Retrieval)	46
5.4	Step III – Extract the Price from the Price Tag (OCR)	47
5.5	The mostFrequent() Function	50
5.6	Step IV – Database and Price Comparison	50
5.6.1	Database	50
5.6.2	The Price Comparison	51
5.7	The Admin GUI	53

CHAPTER 6 CONCLUSION

6.1 Summary of the Project	58
6.2 Summary of the Product	59
6.3 Future Work	59

REFERENCES	60
-------------------	-----------

LIST OF FIGURES

Figure Number	Title	Page
Figure 1-1-1	Example of Price Tag in Retail Store	3
Figure 1-5-1	Traditional Bar-code Scanner	5
Figure 2.2-1	EAN-13 Barcode	9
Figure 2-3-1	Example of 1D and 2D Barcode	10
Figure 2-4-1	The System Architecture of the Tesseract OCR	12
Figure 2-5-1	The Label on the Maps Shows the Current User of ROS	15
Figure 2-6-1	Bossa Nova's Shelf-scanning Robot	16
Figure 2-6-2	Simbe Robotics Tally	17
Figure 3-1-1	System Flow Chart	21
Figure 3-1-2	Example of Barcode for Product Code Retrieval	22
Figure 3-2-1	Use Case Diagram for Price and Stock Checking Robot	24
Figure 3-3-1	The ROS Block Diagram	25
Figure 3-4-1	The Block Diagram of Barcode Scanning	26
Figure 3-4-2	The Block Diagram of OCR	27
Figure 3-5-1	ERD Diagram for Price and Stock Checking Robot	28
Figure 3-6-1	The Design Idea of Main GUI	29
Figure 4-1-1	Prototype Model	30
Figure 4-2-1	Example of Turtlebot 3. "Burger".	32
Figure 4.2-2	Webcam Use for Robot.	33
Figure 4-2-3	Raspberry Pi	34
Figure 4-2-4	Image Retrieval using OpenCV	35
Figure 4-2-5	Importing Python Built-in Function	35
Figure 4-2-6	ROS Multi-communication	36
Figure 4-2-7	Python and MySQL Code	37
Figure 4-2-8	Python Integrate with MySQL Code	37
Figure 4-2-9	GUI for Admin Login using Tkinter	38

Figure 4-2-10	Sample Python Code using Tkinter	39
Figure 4-3-1	System Architecture Design for Proposed System	40
Figure 4-4-1	Gantt Chart showing timeline for FYP2	41
Figure 5-1-1	The Rack Used for Testing	42
Figure 5-1-2	The Barcode for Testing	42
Figure 5-1-3	The Product Data in Database	43
Figure 5-2-1	Step 1 Robot Control	44
Figure 5-2-2	Terminal Output	44
Figure 5.2-3	The Images Captured	45
Figure 5-3-1	The Barcode Scanning Code	46
Figure 5-3-2	The Output after Scanning Process	46
Figure 5-4-1	OCR Part 1	47
Figure 5-4-2	Result after Eliminating Unwanted Background	48
Figure 5-4-3	Output after Dilation	48
Figure 5-4-4	The Comparison of With and Without Image Processing	49
Figure 5-4-5	Output of OCR	49
Figure 5-5-1	mostFrequent() Function	50
Figure 5-6-1	Creation of MySQL Database	50
Figure 5-6-2	The Details of the Tables	51
Figure 5-6-3	The SQL Query	51
Figure 5-6-4	The Price Comparison	52
Figure 5-7-1	The Checking Report in the GUI	53
Figure 5-7-2	Login Interface	55
Figure 5-7-3	The GUI for Add Product	55
Figure 5-7-4	Edit Product Interface	56
Figure 5-7-5	View Product Interface	56
Figure 5-7-6	Delete Product Interface	57

LIST OF TABLES

Table Number	Title	Page
Table 2-6-1	Comparison between Reviewed and Proposed System	19
Table 5-1-1	The Product Code and Price of the Testing Price Tag	43
Table 5-1-2	Expected Result	43
Table 5-7-1	Expected Output of Black Box Testing	53
Table 5-7-2	Actual Output of Black Box Testing	54

LIST OF ABBREVIATIONS

<i>OCR</i>	Optical Character Recognition
<i>1D</i>	One Dimension
<i>2D</i>	Two Dimension
<i>CEO</i>	Chief Executive Officer
<i>DBMS</i>	Database Management System
<i>EAN</i>	European Article Number
<i>ETC</i>	Et Cetera
<i>GUI</i>	Graphical User Interface
<i>OCR</i>	Optical Character Recognition
<i>RM</i>	Ringgit Malaysia
<i>OpenCV</i>	Open Computer Vision
<i>PiCams</i>	Raspberry Pi Camera
<i>QR Code</i>	Quick Response Code
<i>RFID</i>	Radio Frequency Identification
<i>ROS</i>	Robot Operating System
<i>SQL</i>	Structured Query Language
<i>UPC</i>	Universal Product Code
<i>Webcam</i>	Web Camera

Chapter 1

Introduction

1.1 Problem Statement and Motivation

In the past few years, price of goods in the market are updated very frequently due to economic fluctuation. The problem arises due to the price tag together with the bar-code on the rack is usually being forgotten to be changed. Hence, the wrong information will be displayed to the customer and mislead them. According to Marketing (2018), the issue above leads to some problems such as customer refuses to pay for the item due to the price is vary from the price displayed on the rack and the customer might feel offended on this issue. This problem will cause the company to lose their trust towards customers.

According to Bae, Han, Cha and Lee (2016 , pp. 1-2), Price checking procedure is actually very troublesome. The employee in charge will need to check the price stated on the price tag and compare it with the latest price in the pricing database one by one using the portable reader. This kind of job will need to be carried out in a daily basis. As this job is done manually, thus, a lot of manpower will be required as the employees will need to go around whole yard. Considering the time taken needed to finish the job, if that employee needs half an hour to do checking for stationary department and there are still electrical department, baking materials department and others. If there are 10 departments in total, the employee will need to spend 5 hours everyday to do the checking job which is consider as wastage of manpower towards the business.

Furthermore, imagine that there are 1000 kinds of goods in the retail store, it is very hard to know or find out which product is already sold off. In a more general description, it is very troublesome if the employee need to walk around the store front to check for the stock to make sure that there are always products to be displayed on the rack. The employee in charge will need to do stock check after a certain time period to prevent this problem from happening.

In summary, business owners require a price and stock checking robot for retail stock to deal with the cases mentioned above. This robot will handle day to day price checking procedure to make sure that the price shown on the shelves are always accurate. The robot will also be given the responsibility to inform the employee once the item on the rack is empty.

After the robot finished the checking procedure, a report will be generated and sent to the GUI to inform the employees about the faulty price tag and stock available. There will be an interface for the employee to look on.

The retail store which owned massive retail stocks and large store front such as supermarket like Tesco, Giant, Econsave, and etc will encounter the issues mentioned above. The larger the storefront, the longer the time taken for the employee to check for the correctness of the price to keep it up-to-date. Therefore, this product will be suitable for them (William, 2018).

Next, retail store who wishes to expand their business should use this robot too. This is because the implementation of robots in working environment aids to the long term benefits. A robot can replace several manpower, if the business owner hires three workers to do price checking per day, the salary and bonus for those employees can be cut down.

The benefits might not be seen in short term but it sure will bring long term benefits to the business. Besides that, the accuracy and effectiveness will increase gradually as human being will make more mistakes than robots.

The correctness of the price bar-code displayed on the rack is very important as it is the first thing that tells the customer about the price of the item. If the price is overstated, the customers will have negative feelings about the store and would not buy that item from the store, thus reduce the sales of the business. According to IHL Report in 2015, it stated that global retail store losses 448Billion Dollar annually due to empty shelves and out-of-stock (Engagedynamicaaction, 2019).



Figure 1-1-1 Example of Price Tag in Retail Store

1.2 Project Scope

This project will develop a price and stock checking robot that will be able to help the business owner to manage their inventory better. There are two parts in this proposed system:

Firstly, the robot will be able to do price checking. For this part, the robot will walk along the rack which has price tag attached on it to perform price check by doing comparison with the price in the database. If the displayed price is varying from the database, the system will notify the admin by sending them to the graphic user interface.

Next, the robot will retrieve the stock available from the database, if the rack is empty, it will notify the owner by sending report.

1.3 Project Objective

1. To develop a walking robot to perform price check
 - The system aims to produce a robot that is able to walk along the rack.
 - The robot will retrieve information from the image capture using web camera.
 - Comparison between price detected and the price in database will be done.

2. To check out the stock status
 - The robot will retrieve the stock availability from the database once the barcode is scanned
 - If the stock status is empty, the system will send alert to the employee's user interface to fill up the empty stock.

3. To create an automated price and stock checking procedure
 - This system will be able to reduce the amount of manpower required to perform daily price check.
 - This system will send information to the employee's interface based on the detection results. Therefore, that employee does not need to walk around the storefront and do checking from time to time.

1.4 Impact Significance and Contribution

This project is proposed to help business owners of retail store and hypermarket to manage their display stock with ease. This price and stock checking robot will check the availability of the stock on the rack by retrieving the stock availability from the database and the correctness of the price stated on the price tag displayed. By helping the business owners to make sure the stock in the rack will be always available which would eventually improve on their sales and reduces the business losses due to empty shelves.

Besides that, this robot can make sure that the customers always get the actual correct price instead of the expired one to prevent them from getting confused. Thus results in lesser argument between the customers and cashier which could smoothen the payment process. The satisfaction of the customer towards the business might increase too.

This project will help in reducing the cost and improve on the profit of the company. It reduces the daily task job as well as reduces human error. Besides, it can also help in increasing the labour cost and employees' productivity. Manpower can be optimized by letting them to handler more important and complicated work.

1.5 Background Information

In traditional way of retail business while bar-code has not been invented, the sticker price tag is paste on every product. To amend the price, they have to replace all of the old sticker price tag with a new tag of the new price. This causes the work to be troublesome and consuming a lots of time. With the invention of bar-code, price checking procedure is becoming easier. The use of sticker price tag reduced and is replaced with bar-code price tag placed on the rack.

However, in order to check correctness the price of the product, the employee will have to scan the bar-code on the price tag using the bar-code scanner (Figure 1.2) to retrieve the price from the database and compare it to the price printed on the price tag one by one. This is time consuming and troublesome process.



Figure 1-5-1 Traditional Bar-code Scanner

Besides that, stock checking is also another troublesome procedure. The employees have to walk around the storefront frequently to check for the display stock availability. When the rack is empty, he or she will have to report to the store and fill up the rack. This might not efficient since the stock might be empty for a long period. Besides, there are a variety of the stock and some with the same packages with

different flavour, it causes unavoidable human error. In order to make a business successful and profitable, employee performance optimization is very important. According to Simbe Robotics (2019), their CEO claims that global retail store losses about 472 Billion Dollar per year due the the lack of performance optimization.

Therefore, a price and stock checking robot is used to make the activity stated above automated. This robot is specially needed in supermarket with huge storefront. The two main functionality of this kind of robot is to perform price tag validation and check for the stock availability by retrieving the stock availability from the database. Other than these, it is able to send report to the employee in daily basis about the faulty price tag and the empty display stock on the rack. The location will be shown ease employee's job.

1.6 Proposed Approach

There are two main parts In order to achieve price checking procedure smoothly which are scanning the barcode to get product code and getting the display price from the price tag using Optical Character Recognition (OCR).

The purpose of getting the product code from the barcode is to retrieve the latest price from the database in order to do price comparison. However, pyzbar, the barcode scanning approach used in this system might not always get the result accurately. Hence, this system is designed to capture several images of the price tag for barcode scanning to make comparison between scanning output in order to get the accurate barcode.

After the product code is obtained, the system will retrieve the latest price of the product from the MySQL database. In order to do price comparison, the system needs to retrieve the price stated on the price tag using Tesseract OCR. The approaches of getting accurate OCR result is same as the barcode scanning step above which is take several images and make comparison to get accurate result. However, there are a lot of redundant information in the price tag that will be read by the Tesseract OCR which includes the barcode, the currency symbol and noise from the image. Therefore, this redundant information must be eliminated to get the price only. For example, if

the result the OCR is “RM 12.90/pcs”, “RM” and “/pcs” is not necessary and will be eliminated, “12.90” will be the output for price comparison.

1.7 Highlight of what had been achieved

- Able to walk along the rack and stop to detect bar-code
- Able to capture photo using webcam and OpenCV
- Able to retrieve product code by scanning the barcode on the captured price tag
- Able to perform OCR on the captured image to get only the price on the price tag by eliminating all redundant information
- Able to retrieve the stored data on MySQL database and do price comparison
- Able to display the stock status and price comparison report on the graphic user interface using Python Tkinter
- Able to allow the user to add, delete, view and update the stock/ inventory through interacting with the graphic user interface and store all the information to the MySQL database
- Robot will able to continue to walk and repeat the scanning process until it reaches the end point
- Able to regenerate new price tag with latest price

1.8 Report Organization

This report is divided into 6 chapters. Chapter 1 will be the introduction to this project. Problem statement, motivation, project scope and objectives will be discussed in detail in this chapter. Besides, contribution and background information will also be defined.

In Chapter2, literature review and some comparison and review on similar existing system has been discussed. Strength and weakness of these systems are highlighted.

In Chapter 3, the system design is explaining on necessary information needed to build the project. This includes system flowchart, use case diagram, ROS Block Diagram, Block Diagram, ERD Diagram of Database and Design of GUI.

In Chapter 4 will be the methodology and tools which explaining the methodology, system design specifications, system architecture design, and project timeline.

In Chapter 5 is the testing implementation and result will be shown.

Lastly, in chapter 6, is the conclusion of the project which describes the summary of the project and product and future development and improvement.

Chapter 2

Literature Review

2.1 Background

In this project, the robot should be able to get the product code from the barcode using web camera and retrieve the price on the price tag by using optical character recognition. Besides, the robot will communicate with the database using MySQL and python to retrieve the latest price. ROS will act as the meta-operating system for the robot in order to perform low level device control. Lastly, this literature review also reviews existing price and stock checking robot to compare the functionality between the reviewed system and this system.

2.2 Barcode

A barcode is a machine-readable internal representation of information. Barcodes are usually printed on items, such as price tag, tickets, packaging of goods and etc (Osborne & Russell 2018).

- **EAN-13**

An EAN-13 barcode is a 13 digit (12 data and 1 check) bar coding standard. The digits on the bar code are split into four main groups such that the first and second digit, the first group of 5 as the manufacturing company's code and the next group of 5 which is the product code, and lastly, the last digit which is used as a check digit, form four distinct set of numbers (Upasani et al 2016).



Figure 2-2-1 EAN-13 Barcode

2.3 Existing Bar-code Scanning Solution

Zbar (Zbar 2019)

Zbar, an open source software that design for C++, Python, Ruby bindings and Perl. It can be implemented on Microsoft windows and linux as a command-line application. It can read the barcode from different sources. For example, image file, raw intensity sensors, and video streams. It can support various popular symbologies which is the type of barcodes. The type of the barcode includes UPC-E, EAN-8, EAN13/UPC-A, Code 39, Code 128. The image capture from the OpenCV will be passed to the dedicated barcode decoding library. It will decode the barcode as well as the QR Code.

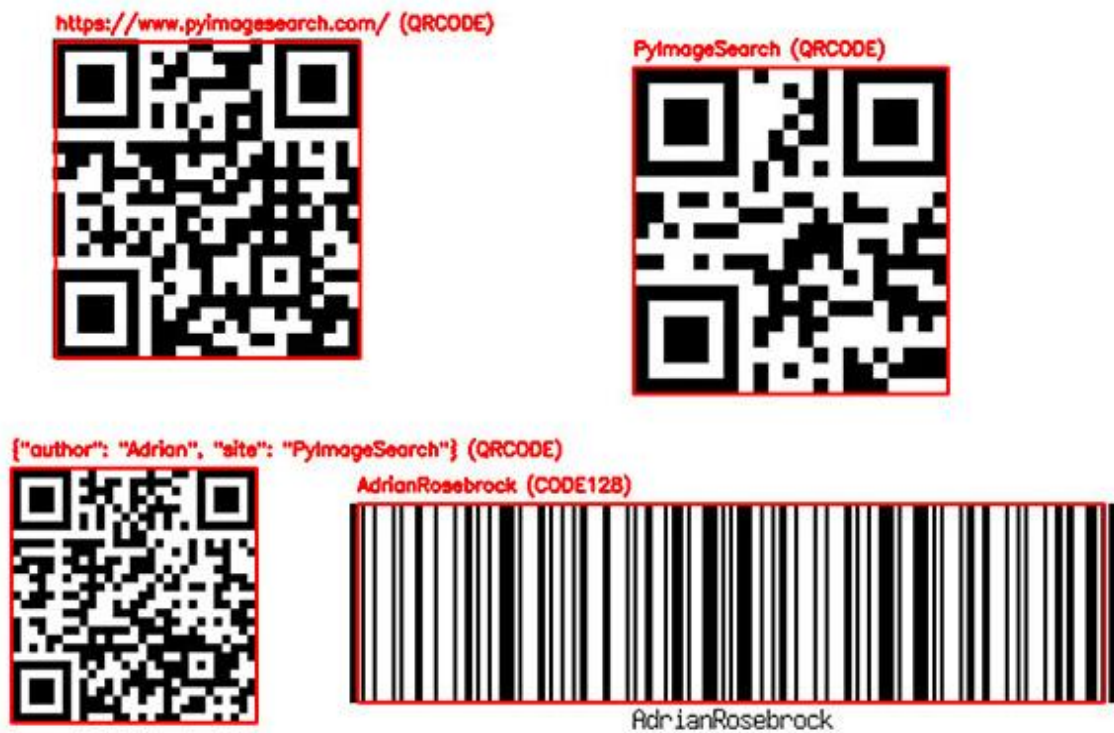


Figure 2-3-1 Example of 1D and 2D Barcode

The Strength of Zbar (Herbst, 2016)

- Cross Platform

Zbar can be used on multiplatform or platform independent. It means that it can work across multiple operating environments or multiple types of platform.

- High Speed

Zbar can read the data in the barcode within 2 to 3 seconds right after the barcode is detected.

- Small Code Size

Zbar library is ready and open source. It eases the programmer to program it easily and save their time.

The Weakness of Zbar

- Low Accuracy

Zbar is too quick to read the barcodes and it results the wrong data in barcode to be decoded.

2.4 Optical Character Recognition (OCR)

2.4.1 Tesseract OCR

According to Smith (2007, pp. 629-633), it is an open source engine which is developed at HP between 1984 to 1994. It is highly portable and focus more on providing less rejection than accuracy (Patel, Patel, & Patel ,2012). Tesseract accepts input image that is in binary form and it can handle both the inverse-White on Black text and the traditional Black on White text. (Mithe, Indalkar & Divekar, 2013, pp. 72-75) Tesseract OCR performs step by step as shown in the block diagram below (Figure 2.3).

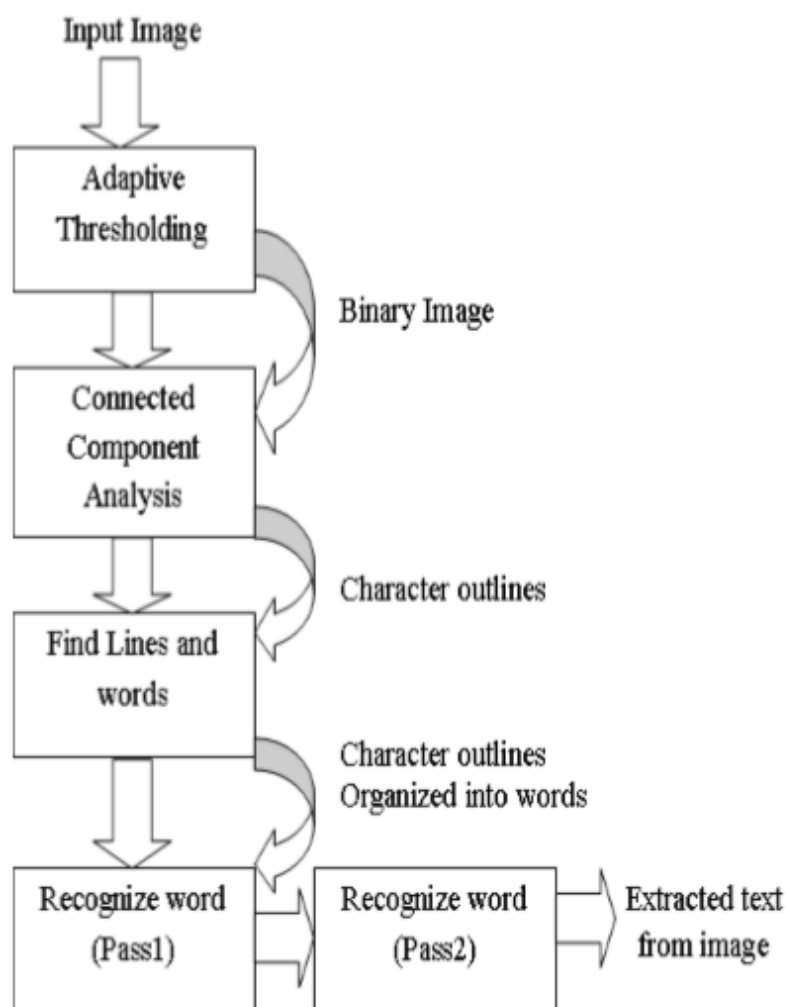


Figure 2-4-1 The System Architecture of the Tesseract OCR

How Tesseract works

Adaptive Thresholding is the step after the image is inputted. It will change the input picture into binary form. After that, the next step will be the connected component analysis. This step is used to take out character outlines. This method transforms the OCR of image into gray scale. In the next step, outlines will be converted into Blobs which will be arranged into text lines. Text is then separated into words using fuzzy and definite spaces. There will a Character recognition of text will begin as two-pass process. Firstly, there will be a set of training data which those words will be transfer into the adaptive classifier in the first pass. Attempt will be made to recognize words from the text. Meanwhile, in the second pass, text will be recognized more clearly by

BCS (Hons) Computer Science

the adaptive classifier. In the final stage, issue will be resolved and text from image will be filtered out (Patel, Patel, & Patel ,2012).

Advantages and Limitations of Tesseract OCR

Tesseract OCR is open source and it is accessible in the form of Dynamic Link Library. Furthermore, it can make available in graphics mode easily. Tesseract is more accurate in extracting text and it is a faster way in OCR processing. (Patel, Patel, & Patel ,2012). The limitation is the accuracy of detecting words is dependent on the quality of input document. For example, if a poor quality camera is use to capture the input image, the detection results will not be nice. As improvement table boundaries detection techniques is used to obtain a better results. Text post-processing techniques are used to detect the noise in the document and correct those bad-recognized words. (Mithe, Indalkar & Divek et al, 2013, pp. 72-75)

2.5 Robot Operating System (ROS)

ROS is a framework used in development of robotics software. It contains a huge number of small programs that runs simultaneously and passes messages to communicate each other's rapidly. ROS is a meta-operating system which runs on the existing operating system. ROS encourage code reuse in robotics research instead of orienting towards middleware, framework and so-called robot software platform (Quigley, Gerkey and Smart, 2015).

2.5.1 Component of ROS (Pyo et. al, 2017)

- 1.Client Library – A client library is used to support various type of programming language.
2. Robotics Application – Service application that works upon the Robotics Application Framework.

3. Robotics Application Framework – Create Robotics Applications.
4. Communication Layer – Reception and data transmission.
5. Hardware Interface Layer – Perform hardware control.
6. Software Development Tools – Programming tool use to create, maintain or support other programs.
7. Simulation – Tools that can control robot in virtual space.

2.5.2 ROS Ecosystem (Pyo et. al, 2017)

The term ‘Ecosystem’ refers to structure which connects the end users, hardware manufacturer, app developers and also the operating system developing companies in ROS. Ecosystem is actually not a new concept in market. Robotics is also forming its own ecosystem. In the beginning, a lot of hardware technologies were overflowing. However, there was no operating system to join them all together. Now, some software platforms are invented and a ROS ecosystem is built later on. Most of the robots in market support ROS now.

2.5.3 Advantages of ROS (ROS, 2018)

ROS provides free and open source libraries and tools to aid the robot application creation. From ROS components, it is predictable that ROS is now and will become the standard in Robotics for years. Over past few years, the use of ROS is growing tremendously, for now there is a huge community of users worldwide. ROS is said to be language-neutral as it supports a multi-language programming. ROS is a peer-to-peer architecture which enables the connected computers to perform parallel tasks in a network. However, it is not easy for beginners to learn as proper and good documentation is not prepared.



Figure 2-5-1 The Label on the Maps Shows the Current User of ROS

2.6 Review and Comparison on Existing Product

2.6.1 Bossa Nova's Shelf-scanning Robot



Figure 2-6-1 Bossa Nova's Shelf-scanning Robot

Bossa Nova is a self-scanning robot created in United States. It is becoming more and more popular in grocery store across the country. For example, Walmart, one of the biggest hypermarket in United States, uses them in theirs stores. (Kolodny, 2018)

There are three main functions of the Bossa Nova robot which the first is it could scan the shelves in the stores to find out which product is in stock and needs to be refilled. Besides, the second functionality is the misplaced items will be determined and notify the employees. Lastly, it can also help to check the incorrect prices and missing product labels on the price tag. (Bossanova, 2019)

The Strength of Bossa Nova's Shelf-scanning Robot

The robot is designed with obstacles avoidance. So when it walks around the storefront it can avoid bump into people and collide with the shelves. It can work two times more accurate and three times faster than human, human error made by the employee can be reduced greatly. Multiple barcodes can be scanned in a very short

time. It is very efficient and employee price and stock check routine can be done by it more effectively.

The Weakness of Bossa Nova's Shelf-scanning Robot

As the robot body is quite big in size, it is claimed that it may not be the suitable to work in store with smaller storefront. It is hard for the robot the walk along each rack. It is even harder for them to capture information and avoid obstacles at a smaller format store.

2.6.2 Simbe Robotics Tally



Figure 2-6-2 Simbe Robotics Tally

Tally is a inventory checking robot for grocery store created by Simbe Robotics in United States of America. It is programmed to check stock status on the shelves three

times in a day. Tally is designed to be thoughtful about its movement. The design of Tally is bottom-heavy means that it will not easily knocked down when collision happens (Simbe, 2019)

Simbe says that their products will focus mainly on inventory monitoring. The main function of Tally is it will notify the owner once the stock gets empty or the inventory is getting low. Secondly, it will send the location to the user once it detects misplaced product on the shelf. Thirdly, it will also make sure that the price is always correct. Last but not least, Tally's RFID capability allows it to provide accurate inventory audit and it could remember all the location of each product in the storefront (Francis, 2018).

The Strength of Simbe Robotics Tally

The body of Tally robot is quite small so it is usable in both small and big retail stores. Tally will provide daily report and accumulative analysis to the retail stores which encompassed pricing audit, low inventory, and out of stock. With a suite of sensors, Tally can operate securely during normal business hour alongside employees and shoppers and does not need any infrastructure changes to the store (Businesswire, 2019).

The Weakness of Simbe Robotics Tally

Simbe Tally target market is from small to large store. The height of a Tally robot is insufficient to reach a shelf that is too tall. Thus, it might not be suitable for store with very tall rack.

2.6.3 Comparison of Existing and Proposed System

Features	Bossa Nova	Tally	Proposed Robot
Perform price validation	Yes	Yes	Yes
Check stock availability	Yes	Yes	Yes
Generate report to notify the employee	Yes	Yes	Yes
Regenerate of price tag	No	No	Yes
GUI connects to printer	No	No	Yes

Table 2-6-1 Comparison between Reviewed and Proposed System

2.7 Chapter Summary

In this chapter the existing bar-code scanning algorithm Zbar and Tesseract OCR, the optical character recognition approach, has been discuss. There are also some comparison between two existing price and stock checking robot for retail store. Strength and limitations of these existing products are determined and compared. In this proposed robot, the limitations of the applications will be improved. As from the two existing system above, they are unable to helps in generate new bar-code. Their job will end after the send the checking report. In this proposed robot, after sending report to GUI, employee can press on the regenerate price tag button and the system will help to print out the price tag alongside with the location of the price tag to be pasted.

Chapter 3

System Design

3.1 System Flowchart

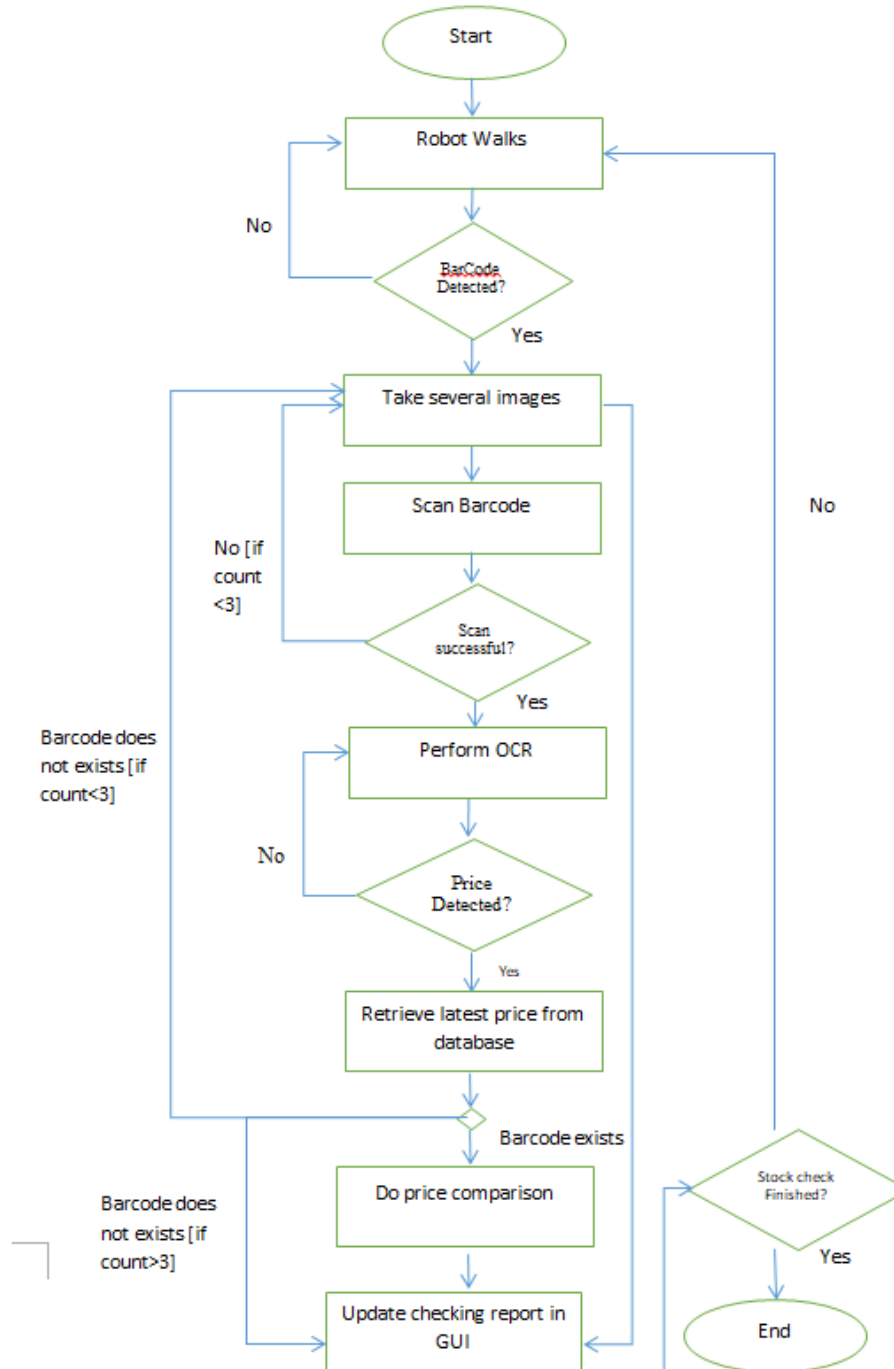


Figure 3-1-1 System Flow Chart

The program flow of the price and stock checking process are shown above. It can be categorized into five main activities.

i) Robot Walk

After the robot is activated, it will walk along the rack and try to detect the existence of bar-code. If no bar-code is detected along the rack, the robot will continue to walk. If there is bar-code detected, the robot will then stop and take several images of the price tag. These images are taken for the purpose of product code and price retrieval which will be carried on later.

ii) Product code retrieval (Barcode Scanning Step)

Next, the robot will send the images captured to the system. The system will proceed to the barcode scanning step. If the bar-code is successfully scanned, the robot will retrieve the product code information from the price tag. For example, if the price tag is as below, the product code retrieved will be the digits below the barcode “9555654654112”.

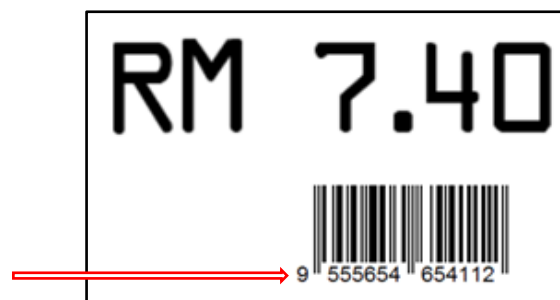


Figure 3-1-2 Example of Barcode for Product Code Retrieval

iii) The price retrieval (OCR step)

OCR will be applied to the images captured. The system will get the price stated on the price tag during this process. From the above image, the system will retrieve the price “7.40”.

iv) Data retrieval from database and price comparison

After step ii and iii, the system will access the database and retrieve the latest price of the product using the product code retrieved in step ii. If the product code does not exist, the system will ask the robot to do image capture again. However, this step will only be repeated for three times. After the count of three, the barcode will be classified as a bad bar-code which means the barcode contains error. In the other hand, if the product code does exists, the latest price will be retrieved and price comparison will be done.

v) GUI display

The comparison result will be sent to the GUI for the user to see. After sending the report, it will continue to walk and start detecting new barcode (Step i).

3.2 The Use Case Diagram

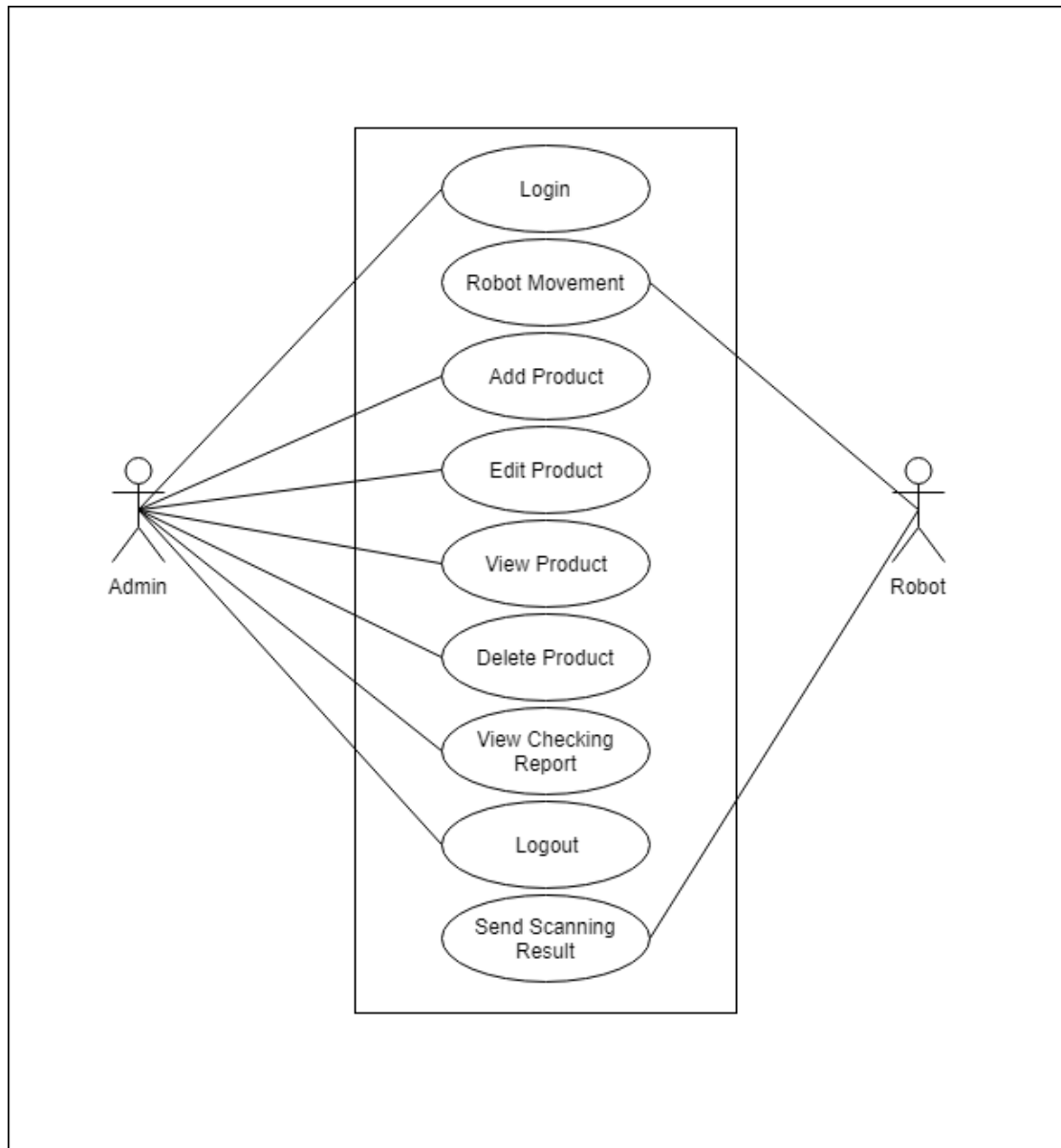


Figure 3-2-1 Use Case Diagram for Price and Stock Checking Robot

The above diagram shows use case for the system. There are two main actors in this system which is the admin or the business owner, and the robot. The admin can interact with the system GUI by performing login, add product, edit product, view product, delete product, view checking report, and log out.

Meanwhile, the robot will get the moving instruction from the system and move until it detects the barcode. The robot will be able to send scanning results which are

several captured images of the price tag to the system. The system will then process the backend job such as retrieving product code, OCR and performing price comparison.

The checking report that includes the correctness of the price tag and the stock count in the garage will be sent to end user.

3.3 The ROS Block Diagram (Step i)

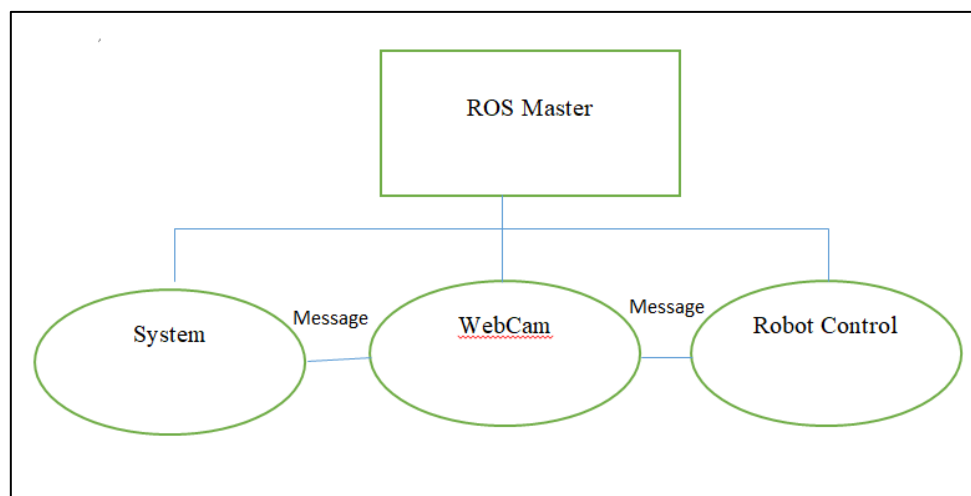


Figure 3-3-1 The ROS Block Diagram

The ROS master will have four nodes, mainly the webcam the backend system and the robot control node. The webcam will be in charging with image capturing and act as the real time camera. After barcode is detected, message will be sent to both the system and the robot control node to ask the robot to stop. The backend system node will send message to webcam and robot control node after the processing job is done. The robot will continue these process until the scanning job is completed.

3.4 Block Diagram

In this part, the details working in barcode scanning for product code retrieval (Step ii) and OCR for displayed price retrieval (Step iii) will be discussed in details.

3.4.1 Barcode Scan (Step ii)

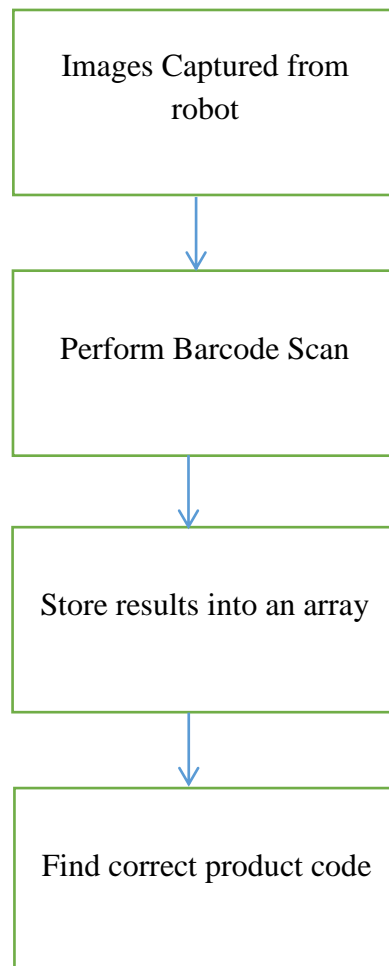


Figure 3-4-1 The Block Diagram of Barcode Scanning

After the price tag is captured by the webcam use OpenCV, the system will start to perform barcode scanning on the images captured. The system is asked to take four photos of the price tag. This step is to ensure the correctness of the price retrieved. The system will start to perform barcode scanning of those photos one by one and save all the scanning results to a string array. After comparing all the output of scanning in the array, the output with highest possibilities will be chosen as the correct product code.

3.4.2 OCR Block Diagram (Step iii)

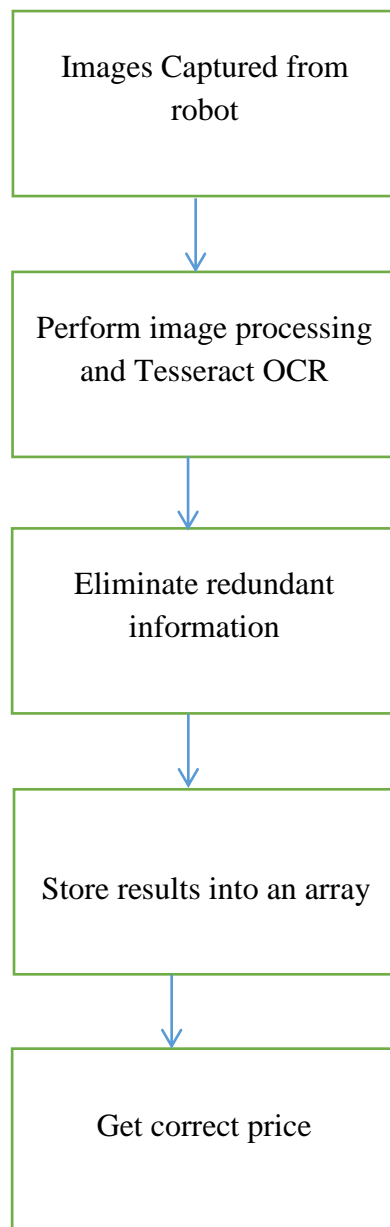


Figure 3-4-2 The Block Diagram of OCR

After the price tag is captured by the webcam use OpenCV, the system will start to perform OCR on the images captured. Before reading the image, the system will eliminate the unneeded region of the photo which will be discussed future in chapter 4. Then, Tesseract OCR will transform the images to words one by one. The system will then eliminate those unwanted information and keep the price to the array. After comparing all the output of OCR in the array, the output with highest possibilities will be chosen as the final price detected.

3.5 ERD Diagram of database (Step iv)

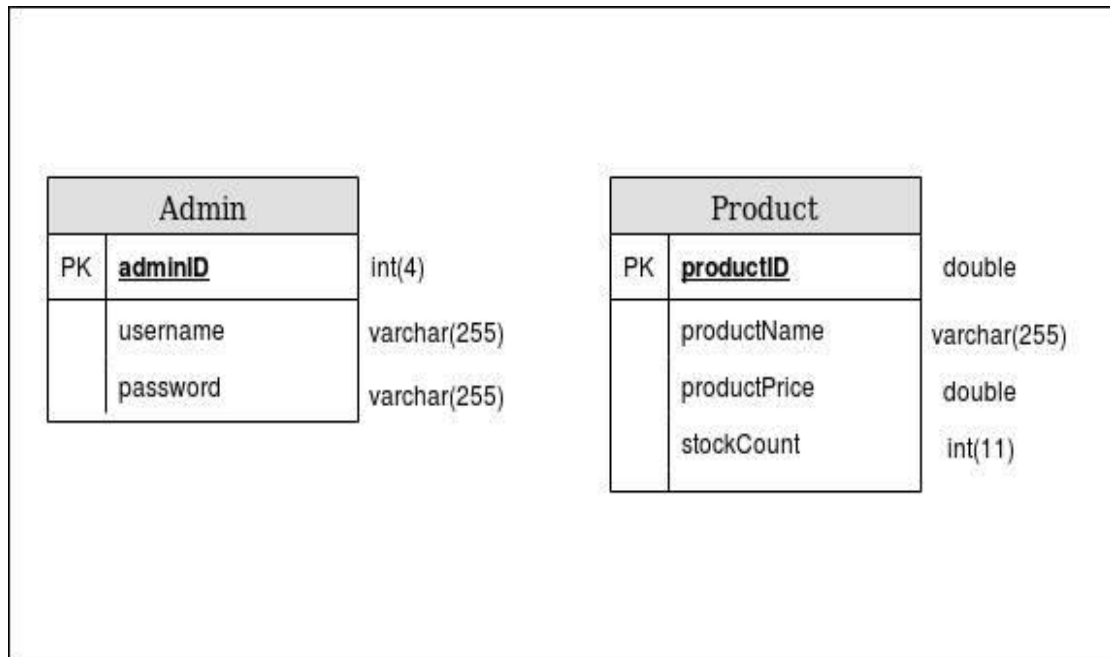


Figure 3-5-1 ERD Diagram for Price and Stock Checking Robot

The ERD diagram of the Price and Stock Checking Robot database system depicts in above named “shopping mall”. There are two main tables which is the admin and the product table. Admin table is used to store details such as adminID, username and also password. Meanwhile, the product info will be stored in product table. The productID will be the 13 letter barcode. Double type is used because INT type only support digits up to 2,147,483,647 digits and barcode is usually made up of 13 digits.

3.6 The Design for GUI (Step v)

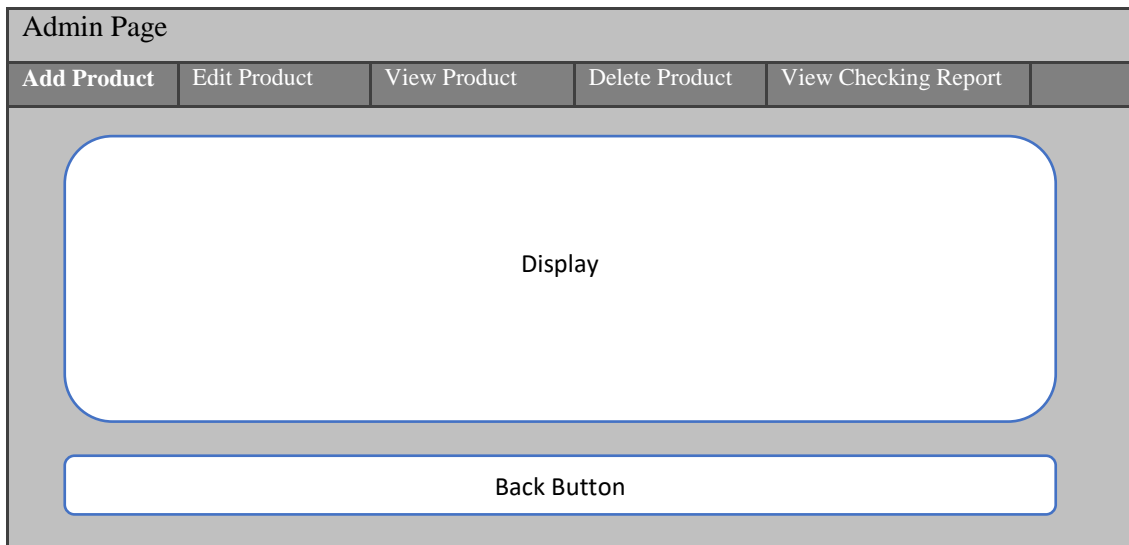


Figure 3-6-1 The Design Idea of Main GUI

This will be the main interface of the GUI. There will be five buttons in the control panel for the user to interact with. Admin can do product maintenances here by adding, updating, deleting and etc. The back button will bring the admin back to the login page (log out button).

Chapter 4

Methodology and Tools

4.1 Methodology

In this project, the Prototyping Model is used. This model is one of the systems development methods. An early estimation of the final system will be built, tested, and then reworked until a satisfied version is done in prototype model. After that the complete and finalize system will be established.

This model is chosen as problems and bugs of the robot can be easily detected. So, developer can fix this problem in the next prototype. As robot is a very complicated work, by using this model, complicated functionality can be scaled into numerous simple tasks which brings higher success rate.

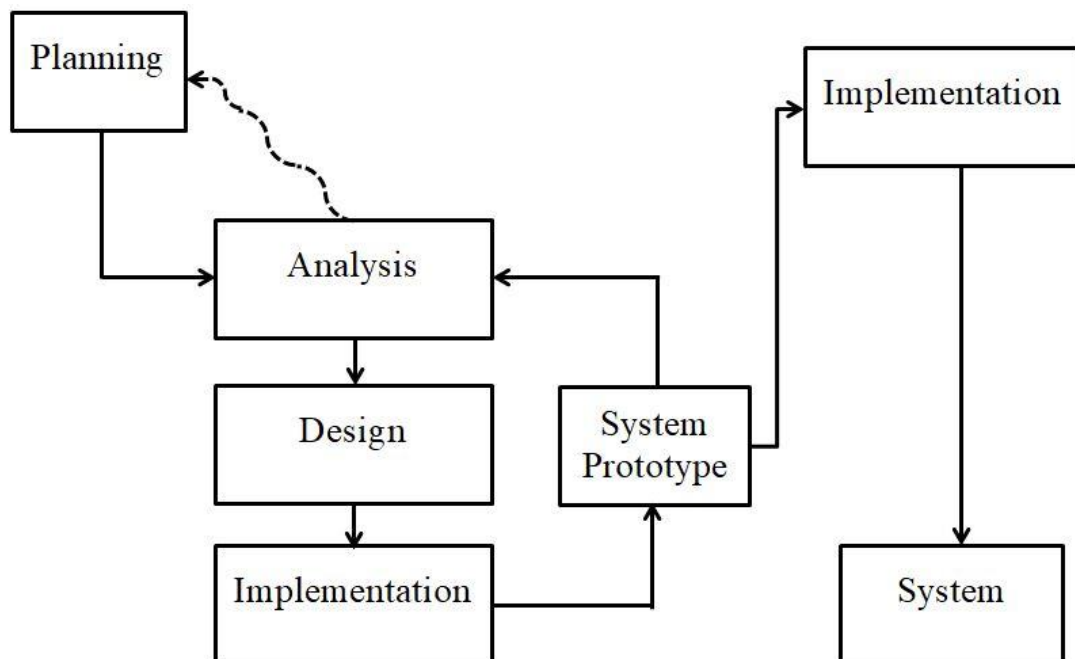


Figure 4-1-1 Prototype Model

First, in the planning phase, the problem arise are found and form the initial requirements. The project management plan will then be documented. Problem statement, project scope, motivation and etc. should be included in this document to determine the job scope that needed to perform in order to develop a good system.

The requirement is gathered from the various retail store business owners to decide what functionality should be equipped in the proposed system. The gathered requirements will then be analyzed to determine the necessity and priority level of those requirements towards implementing this robot.


Thirdly, in the design phase, the functionality and characteristic of the proposed robot will be designed based on user requirements. In this stage, suitable programming language, software and hardware to use are listed out.

After initial analysis and design the initial prototype will be implemented. User comments towards the prototype will be recorded and documented. Improvements will be made based on the latest requirement. Analysis, design and implementation step will be iterating to create as many prototypes as needed until a satisfied prototype is created.


The final prototype also referred as the final version of the robot will be implemented after several prototypes.

4.2 System Design Specifications

Hardware Involved



Software Involved



4.2.1 Hardware tools

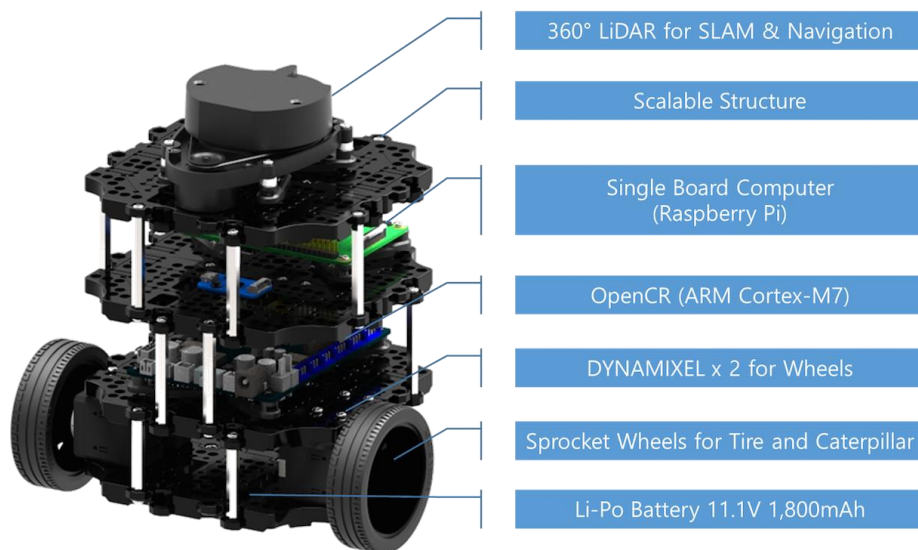


Figure 4-2-1 Example of Turtlebot 3. “Burger”.

Turtlebot3

The TurtleBot 3 was introduced at a conference called ROSCon in 2016. It is designed for university graduate students and developers who wanted to use ROS, Robot Operating System, which will be discussed in details at below. During spring 2017, two models of Turtlebot3 have been invented which are called “Burger” and “Waffle” (Figure 3.2). It was designed to be software paradigm and modular with an open hardware (Thai, 2017, pp. 629-633). In this program, turtlebot3 Burger will be used.



Figure 4.2-2 Webcam Use for Robot.

Web Camera

Web Camera is also known as Webcam. It is a digital camera which connects on a computer. By the use of internet, it can send live pictures to any location. WEB camera is usually attached on the top part of service mobile robot. (Lin, Jia, Abe & Takase, 2004, pp. 851-856). In this project, webcam will be used as a tool to scan barcode and take images.



Figure 4-2-3 Raspberry Pi

Raspberry Pi

The Raspberry Pi is credit-card sized mini computer that can plug into a computer monitor. It uses only keyboard and mouse. It could perform like a desktop computer, from playing high-definition video and browsing internet, to word-processing, and playing games. (Raspberry Pi, n.d.) Pi is turtlebot3 as the “brain” of the system. SD Card with Raspian install in it will be plugged in to the turtlebot3 and make connection with the remote PC.

4.2.2 Software Tools

```
image = cv2.imread("/home/miki/fyp/result/opencv_frame_{}.png".format(count+1))
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

ret, thresh = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY_INV)
blur = cv2.medianBlur(thresh, 1)
kernel = numpy.ones((10, 20), numpy.uint8)
img_dilation = cv2.dilate(blur, kernel, iterations=1)
```

Figure 4-2-4 Image Retrieval using OpenCV

OpenCV

OpenCV is also called Open Computer Vision which is a library that implements many commonly used algorithms in computer vision. Meanwhile, Computer vision is more focuses on pull out structured information from images. It can also apply on videos which are sequences of images. OpenCV is used to perform tasks such as recognizing predefined shapes and objects, human face tracking and motion detection in a video. Additionally, it also provides the infrastructure to deal with images and videos. (Bradski & Kaehler, 2008). The above code shows OpenCV functionality on changing the image color from colorful to gray scale and make the image looks more contrasts.

```
import mysql.connector
import tkinter
import sys
import numpy as np
from __main__ import *
from Tkinter import *
```

Figure 4-2-5 Importing Python Built-in Function

Python

Python is a high-level programming language that handles many tasks that the programmers will need to handle in low-level languages such as C. One of the benefits of Python is they provide a lot of open-source modules which can be downloaded and used freely (Kelly, 2016). Python provides many built in library

which includes, numpy, tkinter and also tkMessageBox which is suitable for the use of this proposed system (Figure 3.6). Python can also be used in OpenCV as in Figure 3.5. It supports various type of function such as object detection, computational photography and etc. Some research said that Python will be used as the major programming language in the future (Kelly, 2016.). Lastly, the installation of python is actually easy and user friendly. Users only need to use the “pip install” command, which support installation of mostly everything. (<https://www.python.org/>)

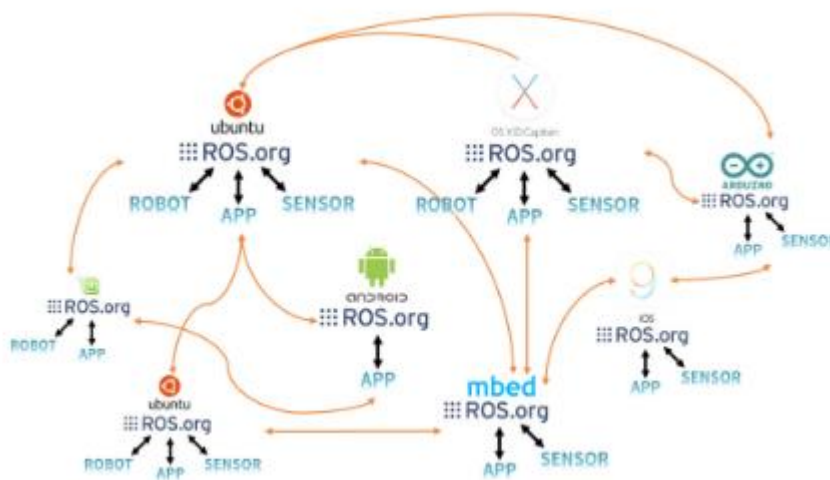


Figure 4-2-6 ROS Multi-communication

Robot Operating System (ROS)

ROS system is design to meet and achieve a specific set of challenges that the researcher encountered while developing robots. It is an open-source, meta-operating system for robots (Quigley et. al, 2009, p.5).

ROS data communication is supported by multiple operating systems, programs and software which makes it suitable for the use of robot development where many kinds of hardware component are combined (Figure 3.8).

It also provides organized operating system services such as device drivers, hardware abstraction. The commonly used features are, motion planning, package management and etc (Pyo, Cho, Jung & Lim, 2017).

```
Database changed
mysql> show tables;
+-----+
| Tables_in_shoppingmall |
+-----+
| admin                    |
| product                  |
+-----+
2 rows in set (0.00 sec)
```

Figure 4-2-7 Python and MySQL Code

```
def Login(event=None):
    usr=USERNAME.get()
    pswd=PASSWORD.get()
    print(usr)
    print(pswd)
    if usr == "" or pswd == "":
        lbl_text.config(text="Please complete the required field!", fg="red")
    else:
        myselect = ("SELECT * FROM admin WHERE username = %s AND password = %s")
        param = (usr, pswd)
        mycursor.execute(myselect, param)
        myresult = mycursor.fetchone()

        if myresult is not None:
            HomeWindow()
            USERNAME.set("")
            PASSWORD.set("")
            lbl_text.config(text="")
        else:
            lbl_text.config(text="Invalid username or password", fg="red")
            USERNAME.set("")
            PASSWORD.set("")
```

Figure 4-2-8 Python Integrate with MySQL Code

MySQL

My Structured Query Language (SQL) is one of the languages used to work with DBMS (Gouhar, 2017), the Database Management System. In this project, MySQL will be used. A database named “shoppingmall” is created and store two tables, which is the “admin” and “product” tables (Figure 3.8). MySQL table is integrated with the python code to create the inventory system for the admin to access (Figure 3.9).

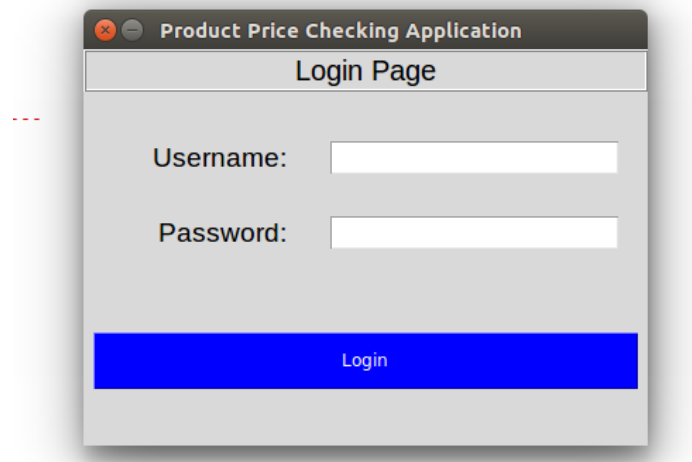


Figure 4-2-9 GUI for Admin Login using Tkinter

Graphic User Interface

A GUI act as a middleman that allows the interaction and communication between user and electronic devices. Instead of using a text-based user interfaces, GUI supports the use of visual indicators such as icons and menus to carry out the command. With the use of GUI, user does not need to interact to the electronic devices through programming language (Computer Home, 1998). A GUI in this system allows the user to check report sent by the robot.

```
from tkinter import*  
#connect to database  
database=sqlite3.connect("C:/Users/mikit/Desktop/my_db.sqlite")  
window =Tk()  
window.title("Try to run")  
#mainloop calls the endless  
#the window will wait for a  
window.mainloop()
```

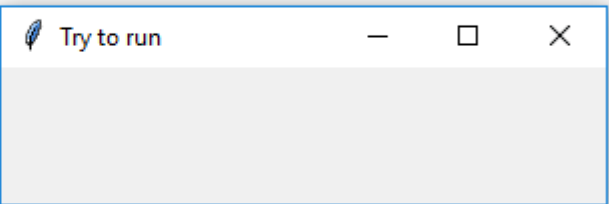
The image shows a code editor window with Python code for a Tkinter application. The code imports Tkinter, connects to a SQLite database, creates a Tk window titled "Try to run", and starts the mainloop. A small window titled "Try to run" is shown in the foreground, partially overlapping the code editor. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Figure 4-2-10 Sample Python Code using Tkinter

Tkinter

Tkinter is the built in library of Python. Tkinter is the combination of Python and Tk GUI toolkit. Tkinter can create complicated GUI which looks like native desktop environment when application is running. In this report, tkinter is used to create the GUI for the inventory system to connect the system to the end user.

4.3 System Architecture Design

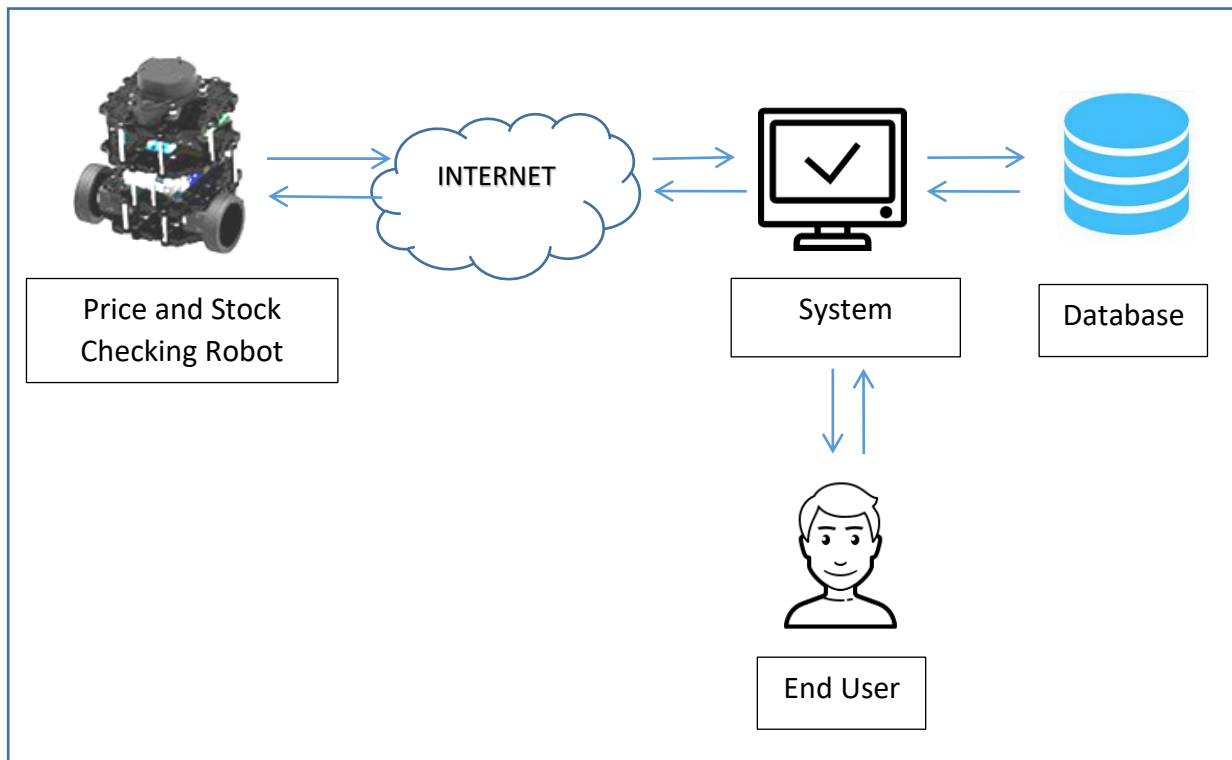


Figure 4-3-1 System Architecture Design for Proposed System

Figure 4.12 shows the architecture design for the price and stock checking robot.

For the environment setup:

1. Robot- Install Ubuntu and ROS for communication of raspberry pi. Setup the robot and make sure remote PC can communicate with the robot through the connection of internet.
2. System- Install MySQL, Python, Zbar and Tesseract OCR library to Ubuntu for creation of the back end system which includes database, GUI and the back end system.

4.4 Project Timeline

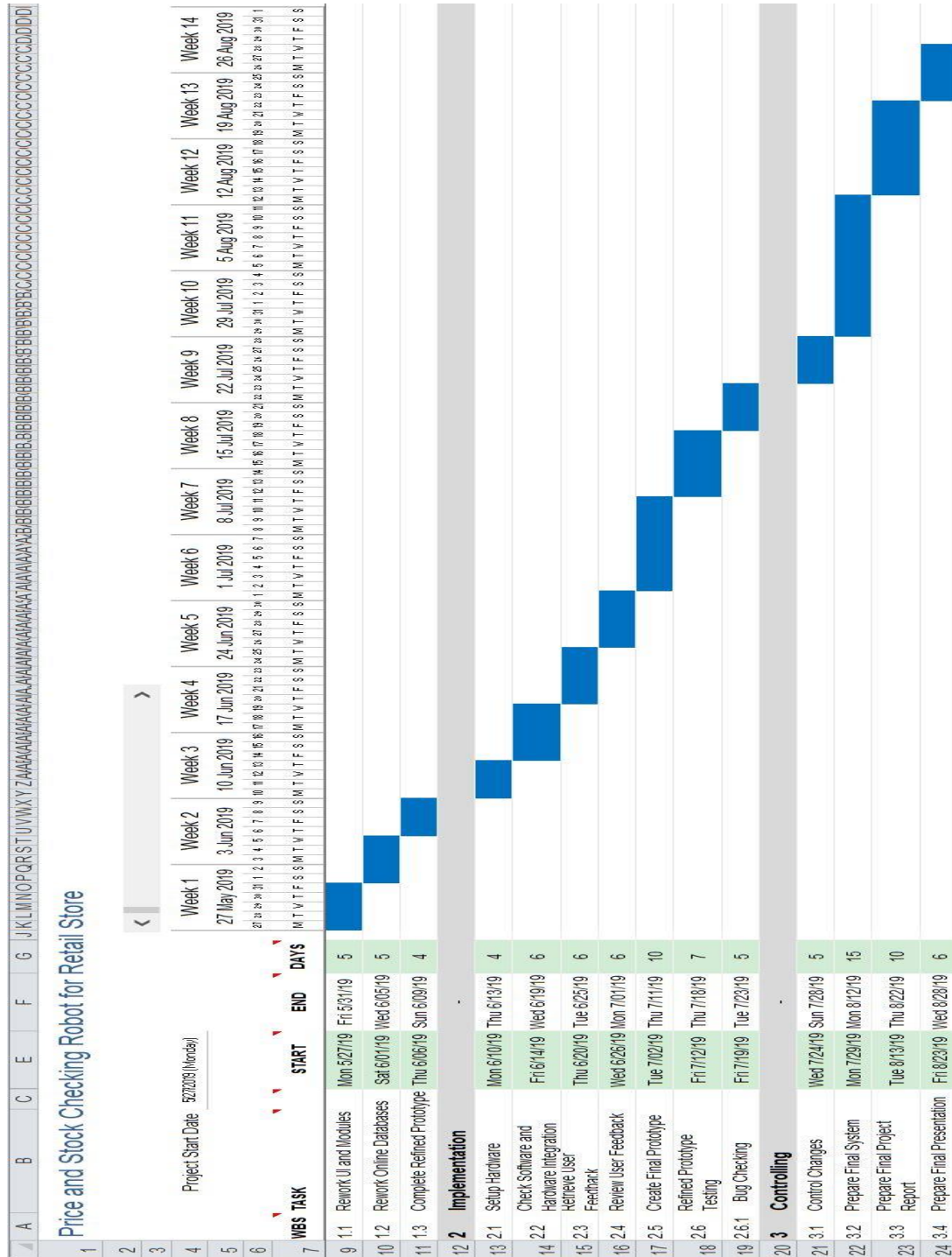


Figure 4-4-1 Gantt Chart showing timeline for FYP2

Chapter 5

Implementation, Testing and Results

5.1 Testing Component

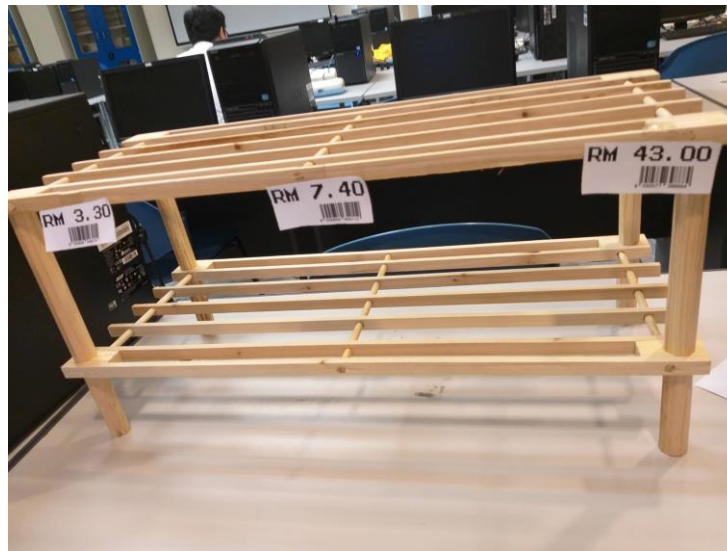


Figure 5-1-1 The Rack Used for Testing

The figure above shows the rack with price tag on it. It acts as the scale down size of the shopping mall rack.



Figure 5-1-2 The Barcode for Testing

Figure 5.2 shows the three price tags used for testing of the system. The product codes and prices of these price tags are shown in table below.

No	Product Code	Price
1	9550577398884	43.00
2	9555654654112	7.40
3	9555658546215	3.30

Table 5-1-1 The Product Code and Price of the Testing Price Tag

```
mysql> select * from product;
+-----+-----+-----+-----+
| productID | productName | productPrice | stockCount |
+-----+-----+-----+-----+
| 9555654654112 | Hatomugi Face Cleanser | 7.7 | 11 |
| 9555658546215 | Mimi Snacks | 3.3 | 12 |
| 9876543210123 | dynamo | 13.4 | 11 |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Figure 5-1-3 The Product Data in Database

The figure above shows the product table in the system database. Therefore, according to the database, the end results after the price comparison process of this testing should be as follow:

No	Product Code	Status	Comparison Results	Stock Count
1	9550577398884	Does not exist in database.	-	-
2	9555654654112	Exists in database.	Wrong price	11
3	9555658546215	Exists in database.	Correct Price	12

Table 5-1-2 Expected result

5.2 Robot Control

5.2.1 Step I: Robot Walking



Figure 5-2-1 Step 1 Robot Control

The robot will start at an initial point. The robot will then walk along the rack in a straight line. The robot will then stop at the places which contain price tag. The white tag on the floor shows the spot that the robot will stop and perform scanning.

```
miki@miki-X456UR: ~/fyp
Robot walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Stop
detection of barcode!
Screenshot Saved!
Robot Stop
detection of barcode!
Screenshot Saved!
Robot Stop
detection of barcode!
Screenshot Saved!
Robot Stop
detection of barcode!
Screenshot Saved!
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
Robot Walks---no detection of barcode....
```

Figure 5-2-2 Terminal Output

The diagram above shows the output in the terminal. When the robot is walking, the message “Robot walks --- no detection of barcode...” as in the orange circle will be

shown. This means that the robot does not detect any barcode. Then when the price tag is detected, the terminal will show message “Robot Stop”, “detection of barcode” and “Screenshot saved” message. In this state, the robot will stop, and capture image of the price tag. After the capture, the robot will continue to travel. As in the diagram, the “Robot walks message is shown again, proves that it continues to walk and detect the existence of price tags.

5.2.2 Image Capture after Price tag detection

After the detection of barcode, the robot will stop and take some images of the price tag. These images will be saved into a file named “Results”.

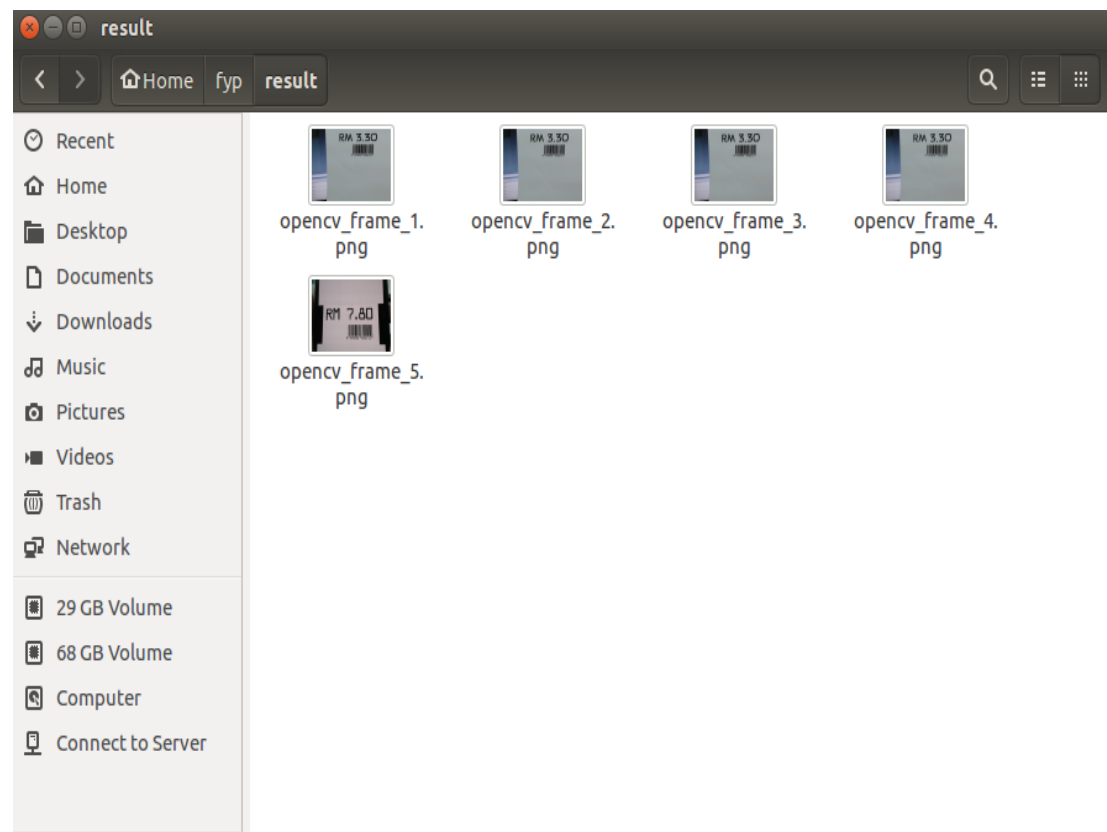


Figure 5.2-3 The Images Captured

These images will be used later on in step ii – barcode scanning and step iii – OCR.

5.3 Step II – Barcode Scanning (Product Code Retrieval)

In this step Zbar library will be used to extract the product code information from the barcode of the captured images.

```
def get_productCodeAndPrice():
    temp=1
    count=0
    tempArray=[]
    while(count<4):
        image_p="/home/miki/fyp/result/opencv_frame_{}.png".format(temp)
        im = cv2.imread(image_p)
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        productCode=pyzbar.decode(gray)
        for code in productCode:
            productData=code.data.decode("utf-8")
            if (len(productData)==13):
                tempArray.append(productData)
                print("Product code from image"+ str(count)+" = "+ str(tempArray[count]))
                temp+=1
                count+=1
        product_code_info.append(mostFrequent(tempArray))
    print("The actual product code after comparison =" + str(product_code_info[0]))
```

Figure 5-3-1 The Barcode Scanning Code

The code in the red bounding box shows the captured images are scanned one by one to obtain the product code. If the length of decoded results are 13 characters (The length of standard barcode), the scanned results will be append to the array named productData which is a temporary array to store results. The real code will be saved to the product_code_info array after the filtration in the mostFrequent() function.

```
Product code from image0 = 9555658546215
Product code from image1 = 9555658546215
Product code from image2 = 9555658546215
Product code from image3 = 9555652546815
The actual product code after comparison =9555658546215
```

Figure 5-3-2 The Output after Scanning Process

Above image shows the decoding of barcode in the captured images by using the Zbar library. The reading of the forth captured image is different from another three images. The systems manage to choose the product code with the highest occurrence, which is “9555658546215”. This shows that this system could get a more accurate scanning result for future use.

5.4 Step III – Extract the price from the price tag (OCR)

```
def getPrice_OCR():
    count=0
    store=[]
    new=""
    priceGood=[]
    while(count<4):
        image = cv2.imread("/home/miki/fyp/result/opencv_frame_{}.png".format(count+1))
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        white_bg = 255*numpy.ones_like(image)
        ret, thresh = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY_INV)
        blur = cv2.medianBlur(thresh, 1)
        kernel = numpy.ones((10, 20), numpy.uint8)
        img_dilation = cv2.dilate(blur, kernel, iterations=1)
        im2, ctrs, hier = cv2.findContours(img_dilation.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        sorted_ctrs = sorted(ctrs, key=lambda ctr: cv2.boundingRect(ctr)[0])
        for i, ctr in enumerate(sorted_ctrs):
            # Get bounding box
            x, y, w, h = cv2.boundingRect(ctr)
            roi = image[y:y+h, x:x+w]
            if (h > 50 and w > 50) and h < 200:

                cv2.rectangle(image, (x, y), (x + w, y + h), (255, 255, 255), 1)
                #cv2.imshow('{}\n'.format(i), roi)

            #--- paste ROIs on image with white background
            white_bg[y:y+h, x:x+w] = roi
```

Figure 5-4-1 OCR Part 1

The code above shows the image processing processes before going to the OCR process. The system will firstly change the image into gray scale, perform blur and dilation using the openCV library. Then the system will crop out the region of interest. This step is inevitable as tesseract OCR library can only scan images that is clear and contrast enough. After selecting the contours, the contour image will be sorted according to their coordinate and pasted onto a new image with white background. This step makes sure that the unwanted background and noises are eliminated in order to get a more accurate OCR results.

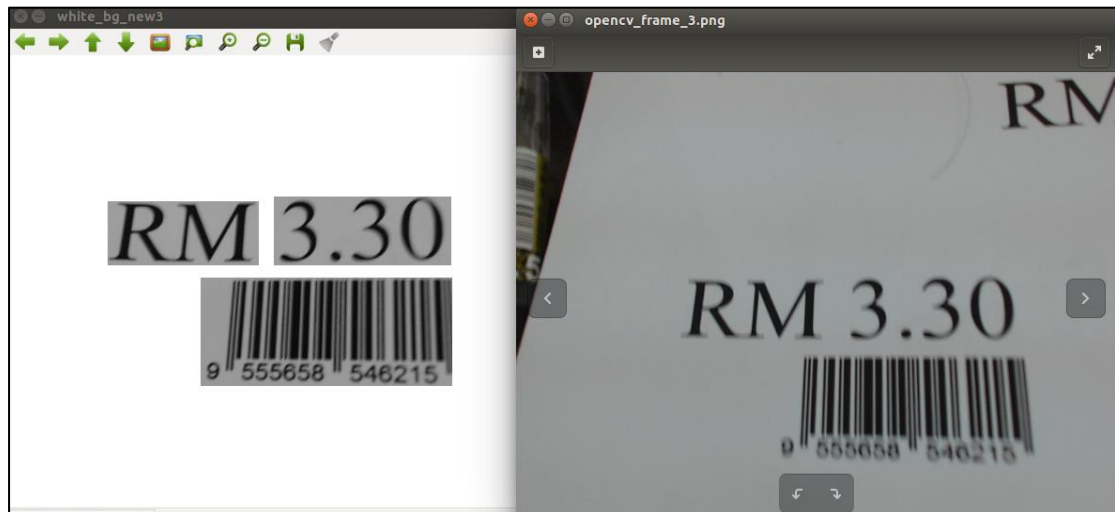


Figure 5-4-2 Result after Eliminating Unwanted Background

The image on the left side shows the output after the unwanted background and noises are eliminated. The image on the right side is the original image that captured by the robot. After this eliminating process, the precision of reading the words using OCR increases.

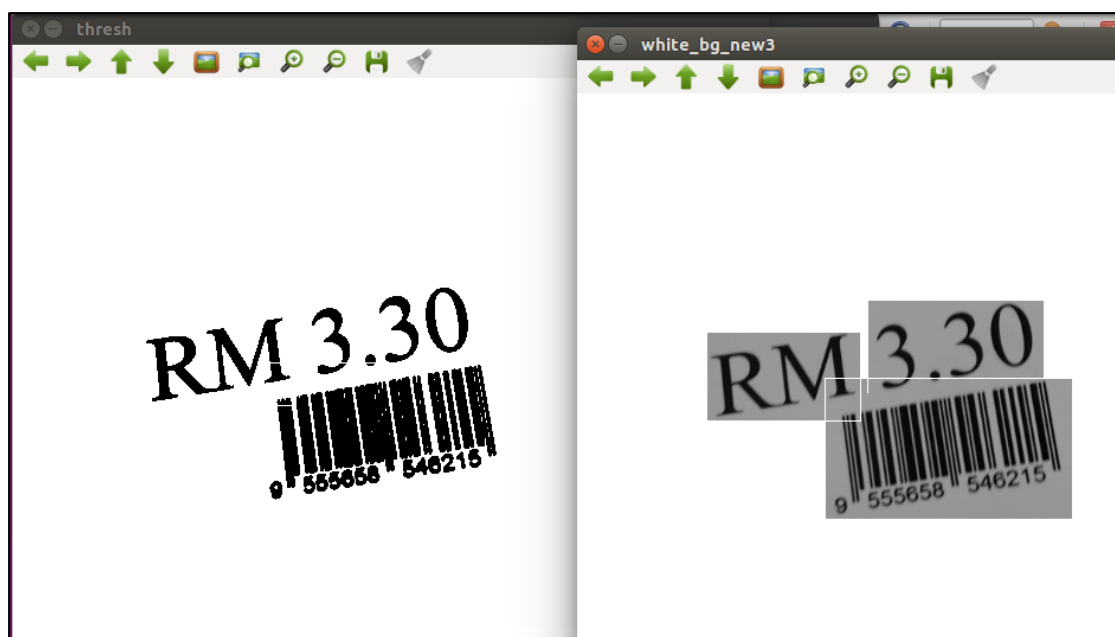


Figure 5-4-3 Output after Dilation

The last step before sending the image to the OCR is dilation. The image is not in black words and white background. Although the same steps (the extraction of image, translate to black words white background and etc) will be gone through in OCR built in process, this step is necessary as it does help to get a more sharp and accurate results.

```
The results of OCR without image processing = [u'SR', u'L\n\nRM 3.30', u'3.30',  
'']  
The results of OCR with image processing = [u'3.30', u'L\n\nRM 3.30', u'3.30',  
u'3.30']
```

Figure 5-4-4 The Comparison of With and Without Image Processing

The upper part of the image shows the result of OCR without the application of above image processing step. Meanwhile, the bottom is the OCR with image processing step. It shows that with image processing before OCR can get a more accurate output. OCR without image processing takes larger memory and longer time to decode.

From the testing result above,

- Accuracy without image processing = 25% (only the third output is correct)
- Accuracy with image processing = 75% (The second output is still wrong)

```
The results of OCR = [u'R\n\n3()\nJUsRm', u'5\n\n%\n\nMll', u'RM 3.30\nJURHI'  
u'RM 3.30\nJani']  
After eliminating redundant data, price = [u'R\n', u'5\n', u'3.30', u'3.30']  
The final Price the system get is = RM 3.30  
/nThe final product code is 9555658546215 The price get is RM3.30
```

Figure 5-4-5 Output of OCR

The results of OCR – the OCR output before elimination of unwanted data

After eliminating unwanted data, the data set will be passed to the mostFrequent() function to get the most accurate price results. However, the readings of OCR gets only 50% of accuracy as web camera is not able to capture a very high resolution of images.

Finally, the system are manage to get the output correctly, which is 955565846215 and RM3.30 that will be used to do price comparison and stock count extraction later.

5.5 The mostFrequent() function

```
def mostFrequent(List):  
    return max(set(List), key=List.count)
```

Figure 5-5-1 mostFrequent() Function

The main functionality of this function is to compare and filter out the scanned results and the results of OCR. It will count the output with most existence and return it. This step is necessary so that the system can always get the correct output.

5.6 Step IV – Database and price comparison

5.6.1 Database

```
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_shoppingmall |  
+-----+  
| admin                   |  
| product                 |  
+-----+  
2 rows in set (0.00 sec)
```

Figure 5-6-1 Creation of MySQL Database

A database named shoppingmall is created using MySQL for this system. As this is a small inventory system, two tables are built, namely admin table and the product table.


```
mysql> desc admin;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| adminID    | int(4)        | NO   | PRI | NULL    | auto_increment |
| username   | varchar(255)  | NO   |     | NULL    |               |
| password   | varchar(255)  | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> desc product;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| productID     | double       | NO   | PRI | NULL    |               |
| productName   | varchar(255) | NO   |     | NULL    |               |
| productPrice  | double       | NO   |     | NULL    |               |
| stockCount    | int(11)      | YES  |     | 0       |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Figure 5-6-2 The Details of the Tables

The details and data type of the tables are shown above.

5.6.2 The price Comparison

```
class Page5(Page):
    def __init__(self, *args, **kwargs):
        Page.__init__(self, *args, **kwargs)
        label = Label(self, text="----Checking Report-----")
        label.pack()
        lbl=Label(self,text="Product List: ", bg="red",font="white")
        lbl.pack(padx=5, pady=10)
        codeDetected=sys.argv[1]
        priceDetected=float(sys.argv[2])
        query1=("select productID from product where productID="+codeDetected)
        mycursor.execute(query1)
        myResult=mycursor.fetchone()
        query=("SELECT productPrice FROM product WHERE productID =" +codeDetected)
        lbv=Listbox(self,width=100,height=10)
        #print(np.asarray(myResult)[0])
        if(myResult!=None):
            mycursor.execute(query)
            myResult2=mycursor.fetchone()
            query3=("SELECT stockCount from product WHERE productID="+codeDetected)
            mycursor.execute(query3)
            myStock=mycursor.fetchone()
            print(np.asarray(myResult2)[0])
```

Figure 5-6-3 The SQL Query

The code in the red box shows the system is extracting the information from the database.

```
if(np.asarray(myResult2)[0]!=priceDetected):  
    lb11=Label(self,text="Product "+codeDetected+ " Are False!")  
    lb11.pack()  
    lb2=Label(self,text="The price stated is RM "+str(priceDetected)+" it should be RM "+  
    str(np.asarray(myResult2)[0])+" Stock Left = "+ str(np.asarray(myStock)[0]))  
    lb2.pack()  
  
if(np.asarray(myResult2)[0]==priceDetected):  
    lb=Label(self,text="Price tags are correct!")  
    lb.pack()  
    lbx=Label(self,text="Product "+codeDetected+" Price RM "+str(priceDetected)+" Stock Left = "+ str(np.asarray(myStock)[0]))  
    lbx.pack()  
if(myResult == None):  
    lbp=Label(self,text="Barcode "+codeDetected+" contains error! Barcode does not exists in database! Please Check!")  
    lbp.pack()
```

Figure 5-6-4 The Price Comparison

Firstly in query1, the system will check the existence of the product code, if the code is not in the database, error message 3 above will be prompt.

If the code exists, “query” and “query3” in figure 5.15 will be executed. The product price and the stock count will be retrieved from the database. The system then entered the price comparison if statement in figure 5.16.

If the price detected is different from the price in database, error message 1 will be prompt. If the price is same as the price in the database, then message 2 will be shown.

5.7 The Admin GUI

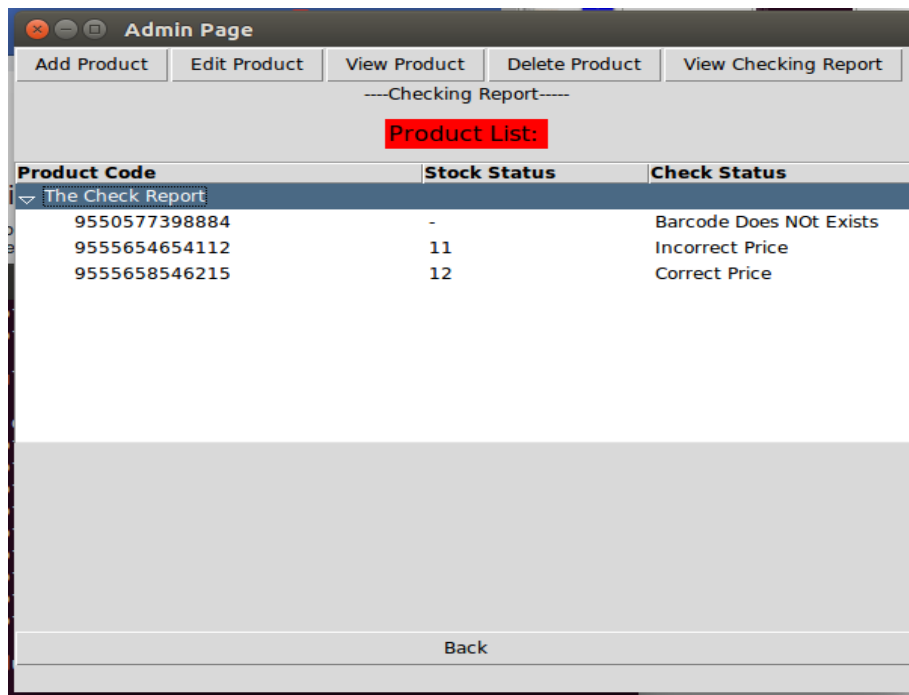


Figure 5-7-1 The Checking Report in the GUI

The image above shows the final results of the price comparison. The checking report is displayed in the admin GUI.

By using the Black Box testing

The expected output:

No	Product Code	Status	Comparison Results	Stock Count
1	9550577398884	Does not exist in database.	-	-
2	9555654654112	Exists in database.	Wrong price	11
3	9555658546215	Exists in database.	Correct Price	12

Table 5-7-1 Expected Output of Black Box Testing

The actual output:

No	Product Code	Output	Stock Count	Pass/Fail
1	9550577398884	Message 3	-	Pass
2	9555654654112	Message 2	11	Pass
3	9555658546215	Message 1	12	Pass

Table 5-7-2 Actual Output of Black Box Testing

The expected output and the actual output show the same output.

Login interface

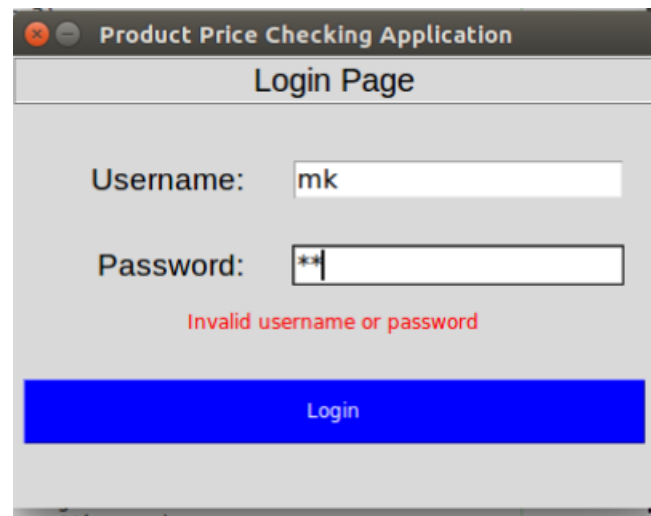


Figure 5-7-2 Login Interface

Add product interface

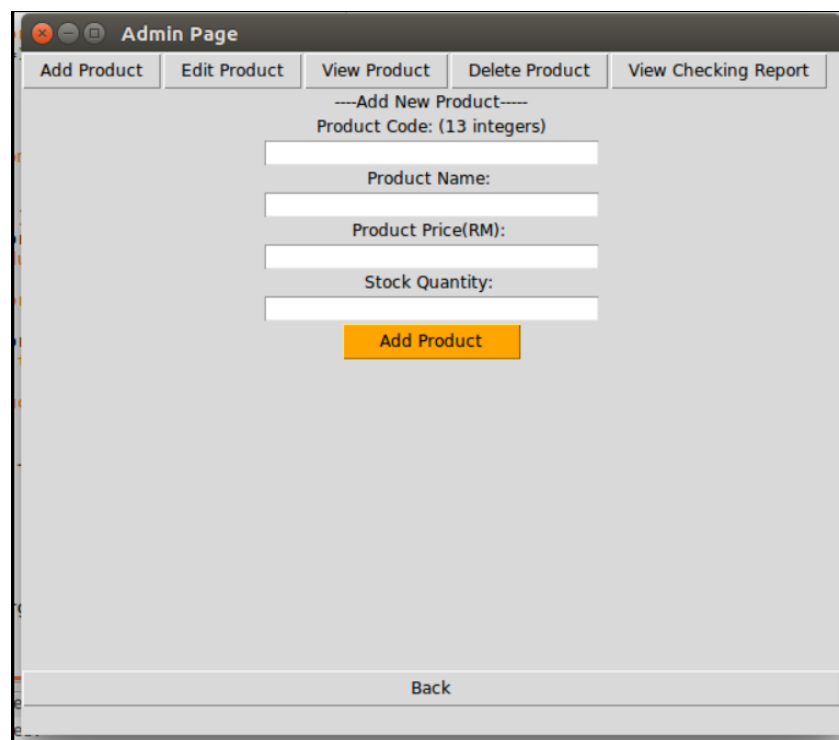


Figure 5-7-3 The GUI for Add Product

Edit Product Interface

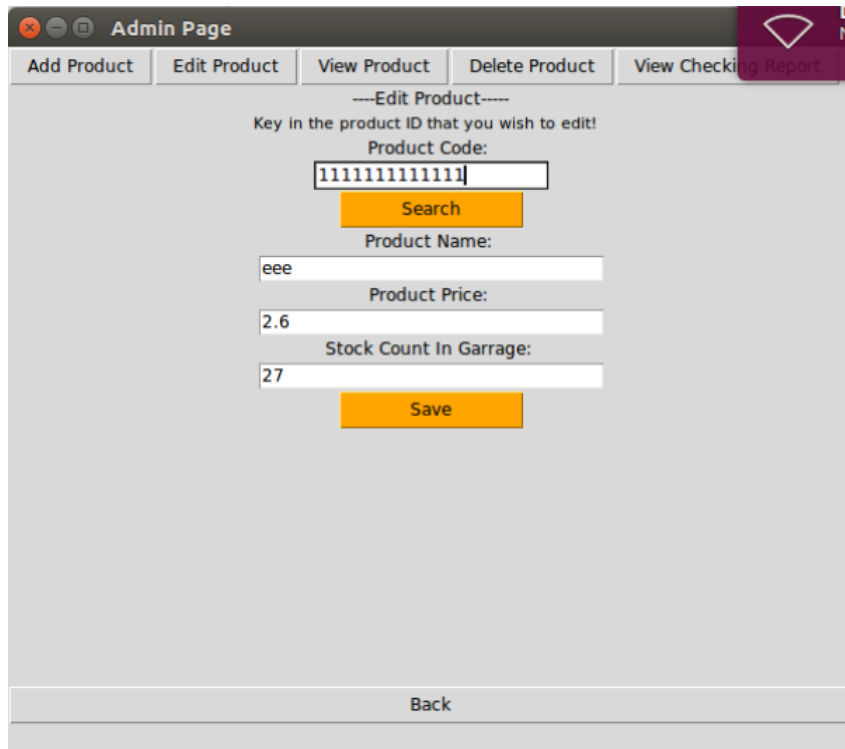


Figure 5-7-4 Edit Product Interface

View Product Interface

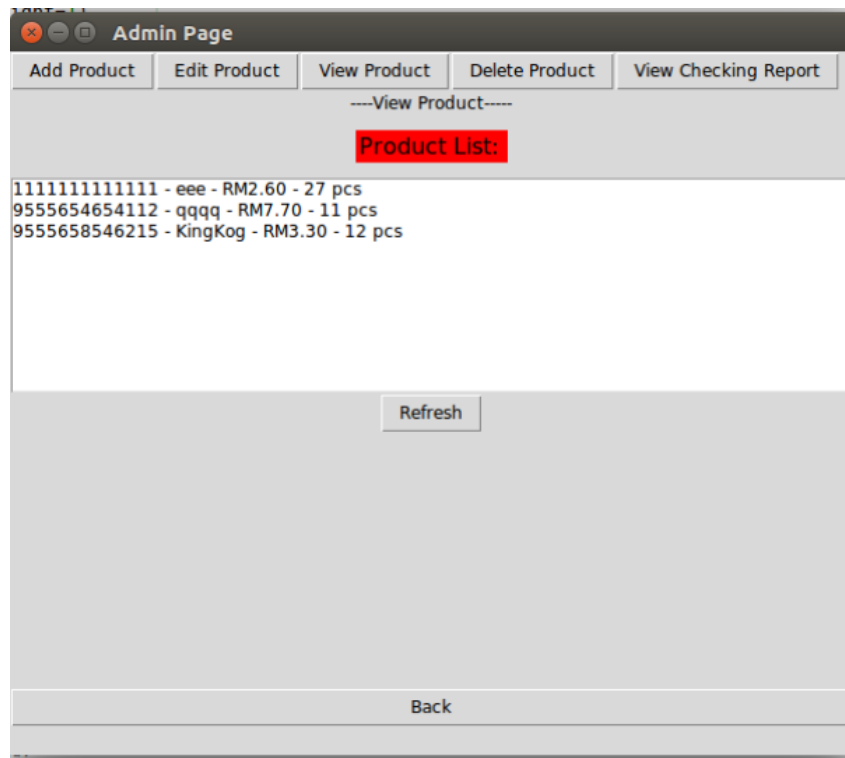


Figure 5-7-5 View Product Interface

Delete Product Interface

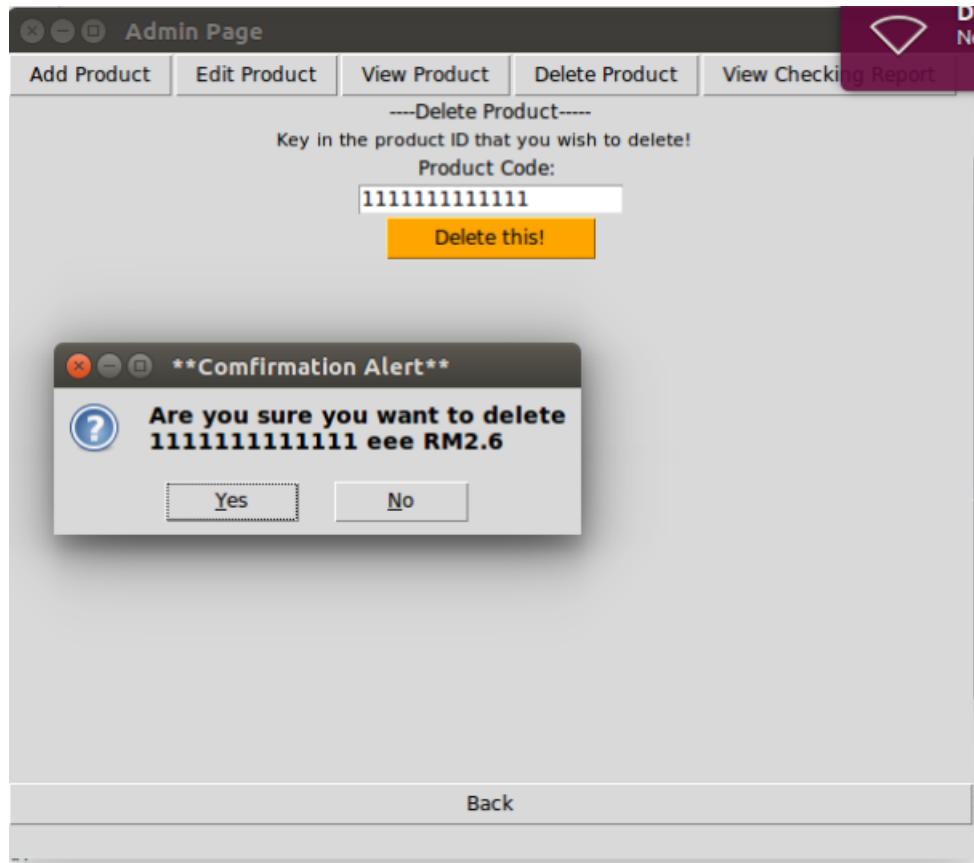


Figure 5-7-6 Delete Product Interface

CHAPTER 6

CONCLUSION

6.1 Summary of the project

Retail store is getting more and more important to human. People buy groceries, needs and wants from there. As the demand is getting higher, the size of retail store is getting bigger too. Based on observation, in most of the time, the price tag on the shelves is not up-to-date or being misplaced. Furthermore, the empty shelves problem is getting more and more frequent. This causes inconvenient to the customer. Customer satisfaction is very important to maintain a business. The major aim of having a business is to make profit from it. However, retail store owners often loss tons of money due to the empty shelf and waste a lot of manpower resources.

The arise of these problems became the motivations of implementing a stock and price checking robots for retail store. The implementation of this robot can replace the manpower needed to perform the daily price and stock checking procedure which is very time-consuming.

Other than just solving the problems stated above, the robot can also ease the employee's job by sending the checking report to the GUI. As a conclusion, the employee will only need to take the new price tag and placed it to the location provided by the system. The time-consuming and boring daily audit is now becoming simple, easy and fun.

6.2 Summary of the product

In this project, the robot is able to walk and stop for price tag capturing. It will capture the price tag and save them as image file. Barcode Scanning will be performed to detect product code and OCR will be used to retrieve the price on the price tag. Comparisons will be done for each result to make sure the final detection results are always accurate. A MySQL database is created to store information about the products such as code, price and details. The system can now successfully retrieve price from the database and also validate the existent of the bar-code. A simple GUI is created to allow end user to view on the report sent by the system.

6.3 Future Work

There are several hardware limitations of this project which include the web camera. The resolution of the web camera is too low, therefore, sometimes the detection will still get wrong price from the price tag even comparisons are done. This is because OCR requires very clear images. Secondly, as turtlebot3 is a small size robot, therefore, it can scan only up to two rows of rack. However, considering this as a scale down size of robotics system, it still performs well and smooth. In addition, this project will be better with the implementation of obstacle detection. The obstacle with a certain distance can be detected by using a ultrasonic sensor. The robot will stop and avoiding itself from colliding with the obstacle. After improving all the hardware limitations above, the robot might be able to perform a more outstanding job.

REFERENCES

- Bae, S.M., Han, K.H., Cha, C.N. and Lee, H.Y., 2016, December. Development of inventory checking system based on uav and rfid in open storage yard. In *2016 International Conference on Information Science and Security (ICISS)* (pp. 1-2). IEEE.
- Bossanova , 2019. Product Inventory Data for Retail | Bossa Nova Robotics. Available from: <<https://www.bossanova.com/>> [Accessed 1 Apr. 2019].
- Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc."
- Business Wire, 2019. Simbe Robotics and Advantage Solutions Partner to Bring In-Store Visibility to Leading Consumer Goods Manufacturers. Available from: <<https://www.businesswire.com/news/>> [Accessed 1 Apr. 2019]
- Computer Home (1998), *GUI*. [online]. Available from:
<<https://www.computerhope.com/jargon/g/gui.htm>> (Accessed 14 August 2018).
- Eikvil, L. (1993). Optical character recognition. *citeseer.ist.psu.edu/142042.html*.
- Engagedynamicaction, 2019. REPORT: Retailers and the Ghost Economy: \$1.75 Trillion Reasons to Be Afraid. Available from:<http://engage.dynamicaction.com/ws-2015-05-ihl-retailers-ghost-economy-ar_lp.html> [Accessed 2 Apr. 2019].
- Francis, S 2018. ‘Simbe Robotics installs inventory robot Tally in Decathlon store’, *Robotics And Automation News* 11 December. Available from:
<<http://roboticsandautomationnews.com/>> [Accessed 1 Apr. 2019]
- Gouhar, A., (2017). Database Management System. *International Journal of Engineering Science*, 11766.
- Herbst, B. (2016) Android Barcode Scanning Library Landscape [online]. Available from <<https://medium.com/@bherbst/android-barcode-scanning-library-landscape-109292b81b65>> (Accessed 14 August 2019).

- Kelly, S., (2016). What Is Python?. In Python, PyGame and Raspberry Pi Game Development (pp. 3-5). Apress, Berkeley, CA.
- Kolodny, L 2018, 'Bossa Nova just raised another \$29 million for its grocery store robots used by Walmart', *CNBC* 21 June. Available from <<https://www.cnbc.com/>>. [Accessed 1 Apr. 2019].
- Lin, W., Jia, S., Abe, T., & Takase, K. (2004, December). Localization of mobile robot based on ID tag and WEB camera. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on* (Vol. 2, pp. 851-856). IEEE.
- Marketing (2018) Retail Pricing Problem and Customer Satisfaction [online]. Available from <<https://www.fellowrobots.com/retail-pricing-problems-and-customer-satisfaction/>> (Accessed 14 August 2019).
- Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1), 72-75.
- Osborne, J. and Russell, D., Samsung Electronics Co., Ltd., 2018. *Modulation of display imagery for barcode simulation*. U.S. Patent Application 15/528,826.
- Patel, C., Patel, A., & Patel, D. (2012). Optical character recognition by open source OCR tool tesseract: A case study. *International Journal of Computer Applications*, 55(10).
- PHP 2019. What Can PHP Do?. Available from:
<<https://www.php.net/manual/en/intro-whatcando.php>>
- Pyo, Y., Cho, H., Jung, R., & Lim, T. (2017). *ROS Robot Programming*. GeumCheon-gu, Seoul: ROBOTIS,Ltd
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).
- Quigley, M., Gerkey, B. and Smart, W.D., 2015. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc."

ROS (2018) *Is ROS for Me? [online]*. Available from:

<http://www.ros.org/is-ros-for-me/> (Accessed 14 August 2018).

Scikit-Learn (2018) *Machine Learning in Python [online]*. Available from

<http://scikit-learn.org/stable/> (Accessed 14 August 2018).

Simbe, 2019. Simbe | Say hello to Tally.. Available from:

<<https://www.simberobotics.com/platform/tally/>> [Accessed 1 Apr. 2019].

Smith, R. (2007, September). An overview of the Tesseract OCR engine. In

Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (Vol. 2, pp. 629-633). IEEE.

Thai, C. N. (2017). ROBOTIS' Robot Systems. In *Exploring Robotics with ROBOTIS Systems* (pp. 5-21). Springer, Cham.

Upasani, S.S., Khandate, A.N., Nikhare, A.U., Mange, R.A. and Tornekar, R.V.,

Robust Algorithm for Developing Barcode Recognition System using Web-cam.

Wilkes, T. C., McGonigle, A. J., Pering, T. D., Taggart, A. J., White, B. S., Bryant, R.

G., & Willmott, J. R. (2016). Ultraviolet imaging with low cost smartphone sensors: Development and application of a raspberry Pi-based UV camera. *Sensors*, 16(10), 1649.

Zbar 2019, *Zbar Bar Code Reader*. Available from: < <http://zbar.sourceforge.net/>>.