Palestine Polytechnic University

College of IT and Computer Engineering

Department of Computer Engineering and Sciences

Graduation Project

Braille to Text/Voice Converter

Project Team

Bayan Halawani

Wisam Younes

Samer Isieed

Project Supervisor

Dr. Radwan Tahboub

Hebron-Palestine

May 2013

جامعة بوليتكنك فلسطين

الخليل – فلسطين

كلية تكنولوجيا المعلومات وهندسة الحاسوب

دائرة هندسة وعلوم الحاسوب

اسم المشروع

Braille to Text/Voice Converter

أسماء الطلبة

وسام يونس يونس	بيان "محمد خالد" الحلواني	سامر نعمان اسعيد

بناء على نظام كلية تكنولوجيا المعلومات وهندسة الحاسوب وإشراف ومتابعة المشرف المباشر على المشروع، وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة هندسة الحاسوب وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

....

ABSTRACT

Communication between people is a very important thing, sighted people use script provided by languages such as English or Arabic to write on papers. However, in case of people who are blind, they use a different type of script known as Braille named after its founder “Louis Braille” which is the system of reading and writing used by people who are blind where they feel raised dots on a Braille page with tips of their fingers; each language can be represented using Braille script, which differs from one language to another.

The Braille to Text/Voice Converter (BT/VC) is a system that designed to help sighted people to be able to understand Braille script without any knowledge in Braille language using mobile phones on Android operating system by converting a Braille image taken by the mobile camera into other scripts.

The aim of this project is to develop a system that is able to translate a Braille script into multilingual script and represents the converted script as text or voice to the user using Android application, the system was developed to translate Braille papers into English language.

DEDICATION

To our parents

To our families and friends

To our beloved Palestine

To our teachers

To all whom support us

To blind Associations

Moreover, for you we dedicate this work

OUR REGARDS

PROJECT TEAM

ACKNOWLEDGMENTS

We wouldn't forget our families and our parents for helping us all the time; they encourage us to be the best, so we give them big thanks for their support, care and love.

We would like to thank our teachers in Palestine Polytechnic University for their major efforts in the years of studying, and we would give a special thank for our supervisor **Dr.Radwan Tahboub** for his effort and useful advices to complete this project, also a special thank for the teachers **Eng.Khalid Daghamen** and **Dr.Hashem Tamimi** for their useful advices.

We would like to give great thanks for the **deanship of graduate research** and **Dr.Sameer Khader** for their financial support**,** which is represented in buying a **S**amsung Galaxy SIII that has been used in the implementation and testing of this project, in addition to covering a big part of cost for the courses that needed to build the application.

We would like to thank the manager of the Blind Association in Hebron **Ms.Reem Tahboub**, all teachers and students in the association for their cooperation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BT/VC	Braille to Text/Voice Converter
OBR	Optical Braille Recognitions
PC	Personal Computer
PIC	Peripheral Interface Controller
MC	Microcontroller
OS	Operating System
LCD	Liquid Crystal Display
SDK	Software Development Kit
Mm	Millimeter
RGB	Red Green Blue
ADT	Android Development Tool
GUI	Graphical User Interface

Chapter 1 Introduction

1.1 Project Overview

Interacting with people is a very important thing in our life. People use languages as a way of communication to deal with others, one of these communication methods is writing, sighted people use the language scripts to write on papers. However, blind or visually impaired people use another type of scripting language, which is Braille, founded by a blind man called "Louise Braille". Braille language rely on the sense of touching (fingertip), it is difficult for sighted people to interact with blind people using such language.

This project proposes to build a system to improve the way of communication between sighted people and blind people as much as possible, the idea is to build a system that takes Braille scripts and convert it to a text /voice to users.

With this system, we might be able to reduce the gap between sighted people and blind or visually impaired people, by having a system that reads Braille scripts without the need to have any knowledge in Braille.

1.2 Project Goal

To design and implement a system that will be able to convert the Braille language scripts into multilingual scripts, using effective algorithms and techniques.

1.3 Project Objectives

BT/VC System is supposed to achieve the following objectives:

1. Make it easier for the teachers to instruct blind students.
2. Help the parents to keep track of their blind child’s study, without the need to learn Braille language.
3. Support multilingual script.
4. Reduce the gap between sighted and blind or visually impaired people.

Furthermore, there are some additional objectives including design an effective database, build an easy use GUI, design a testing mechanism, make the system flexible, usable and maintainable as much as possible.

1.4 Project Importance

The importance of the project came from the ability to reduce the gap between blind or visually impaired people and sighted people, blind or visually impaired people use Braille language that is a perfect way to read and write by those people, but most of sighted people cannot understand Braille language.

The previous problem raised in education system, while the students are blind or visually impaired, some of the teachers are so, the rest are sighted, the blind teachers do their class preparation by using Braille script, but the manager who is sighted has to take a time from blind teachers to read their class preparation, because the manager does not know Braille language.

In addition, the parents of a blind or visually impaired child who start learning Braille language have to keep track of their child progress in Braille without the need to have any knowledge about Braille language.

In addition, the need to design a portable system that will help sighted people to be able to understand the script written in Braille, also one of the points that encourage us is the lack of researches for BT/VC.

Note that there is still no system available that uses mobile environment and support multiple languages.

1.5 Work Methodology

During research, the team found the following three design options for converting Braille into text/voice:

- **Computer-Based System**

  The system will be developed using MS Visual Studio environment, in this case a scanner will be used to scan the Braille paper, apply some image processing algorithms on the scanned image using C# programming language, convert each script in the image to its equivalent script in the desired language, then output the text to the screen, and finally export the converted text as voice to the user.

  Software Tools: Windows 7 OS, MS Visual Studio 2010, MS Office 2007, EmguCV, Dia Software, Smart Draw, Scanner driver.
  Hardware Tools: Scanner, Computer Core2Due, Connectors.

- **Sensor-Based System**

  The system will be developed using Braille sensor, this sensor will be attached to a speakers and an LCD display, each time the sensor passed on a Braille script it will convert it and export the converted script to the LCD screen, and export it as a voice using the speaker attached with the system.

  Software Tools: Windows 7 OS, MPLAP Software, Smart Draw, Dia Software.
  Hardware Tools: Braille Sensor, PIC Microcontroller, Connectors.

  As the team study these two existing design options, they found that the first option lacks portability, the second one lacks efficiency and feasibility, so a new design option is possible using smart phones, which is becoming increasingly developed with a huge amount of memory, processor speed and high-resolution cameras.

- **Mobile-Based System**

  The system will be developed using Android environment, it will be able to capture the Braille paper using mobile camera, apply some image processing algorithms to convert the Braille script into multilingual script and display the result as text or voice to the user.

  Software Tools: Windows 7 OS, MS Visual Studio 2010, SDK for android, EmguCV, Smart Draw.
  Hardware Tools: Android Mobile (Galaxy S3), Computer Core2Due, Connectors.

1.6 Cost Estimation

The estimated cost for three alternative design options will be described in this section:

1.6.1 Computer-Based System

Table 1.1 shows the estimated cost for this option:

Table 1-1: Estimated Cost for Computer-based System

Software	No.	Commercial Cost
MS Windows 7	1	50$
MS Office 2007	1	50$
Dia Software	1	Free Software
Visual Studio 2010	1	170$
Devices Drivers	1	20$
Hardware		
Scanner	1	100$
Connectors	1	30$
PC core2due	1	190$
Total		610$

1.6.2 Sensor-Based System

Table 1.2 shows the estimated cost for this option:

Table 1-2: Estimated Cost for Sensor-based System

Software	No.	Commercial Cost
MS Windows 7	1	50$
MS Office 2007	1	50$
Dia Software	1	Free Software
Visual Studio 2010	1	1٧0$
MPLAB	1	200$
Hardware		
Braille Sensor	1	300$
Connectors	1	80$
PIC MC	1	30$
PC Core2Due	1	190$
Total		10٧0$

1.6.3 Mobile-Based System

Table 1.3 shows the estimated cost for this option:

Table1-3: Estimated Cost for Mobile-based System

Software	No.	Commercial Cost
MS Windows 7	1	50$
MS Office 2007	1	50$
Dia Software	1	Free Software
Visual Studio 2010	1	100$
SDK Android	1	Free Software
Smart Draw	1	50$
Hardware		
Galaxy(S3) Mobile	1	500$
Connectors	1	10$
PC Core2Due	1	190$
Total		**900$**

1.6.4 Human Resources

The project team consists of three students, work 32 weeks for two semesters, 3 hours a day with 5$ per hour, assuming 4 working days a week.

If we consider that the computer engineer takes 1000$ per month, assuming that he/she works 25 days per month, with 8 hours per day, we can calculate the hourly payment for him/her is 5$.

Each student get: (32 weeks * 4 days * 3 hours *5$) = 1920$.

So the total human resources cost for three students: 1920$*3 = 5760$.

1.7 Project Risks Management

1.7.1 **Risks**

Some risks may occur during our project, we consider the following types of risks:

- **People Risks**
  - A member may get ill or die.
  - A member has some problems with some tasks.
  - A member leaves the team.
  - A member has some personal problems and does not involve in teamwork.

- **Hardware Risks**
  - Mobile corrupted or failure.
  - Computer failure.
  - Connector failure.

- **Late Delivery**
  - Team members do not deliver their work on time.

- **Software Risks:**
  - The software is not compatible with mobile version.
  - Hard to deal with software environment.
  - The software is not compatible with operating system.
  - The software may be not responding.
  - Losses the project documentation or data.

1.7.2 Risk Avoidance

In order to manage these risks, we want to deal with each risk as following:

- **People Risks**

  If any problem happened during the work process, it will have a big effect on the project delivery time, in order to solve this problem, we want to consider it and have some strategies to avoid this risk including:
  - Divide the work between two members instead of three.
  - Take some training courses related to the project.
  - Try to involve members to increase the interaction between them.
  - Try to solve the problems, and give members the motivations to continue with this project.

- **Hardware Risks**
    - Take care of the equipment.
    - Maintain the failure if we can.
    - If maintain is not available, then we have to buy a new hardware.

- **Late Delivery**
    - Increase member effort.
    - Keep tack with the supervisor.

- **Software Risks:**
    - Use other software, and insure that it is compatible.
    - Read the software guide carefully and ask for help if needed.
    - Concern that the program is original and having a license.
    - Make a daily backup of project files and data.
    - Restart the software if it is not responding.

1.8 Related Projects

The following related projects and papers were studied:

- **A system of converting Braille into print**

This paper provides a detailed description of a method for converting Braille, as it is stored as "character" into a computer, into print. The system has been designed to be configurable for a wide range of languages and character sets, and uses a predominantly table driven method to achieve this. The algorithm is explained in the context of the conversion of Standard English Braille into print and the tables for this transformation are given [1].

Our project is more developed than this project; it contains one of our project parts, which is converting to English only. In our project, we will add multilingual supports; in addition, our system will be portable and we will add the voice system that is not available in this system.

- **Arabic Braille Recognition and Transcription into Text and Voice**

This paper presents a system for a design and implementation of Optical Arabic Braille Recognition (OBR) with voice and text conversion. The implemented algorithm based on a comparison of Braille dot position extraction in each with the database generated for each Braille cell. Many digital image processing have been performed on the Braille filling and finally image filtering before dot extraction. The work for each Braille cell used as a base for word reconstruction with the corresponding voice and text conversion database. The implemented algorithm achieves expected result through letter and words recognition and transcription accuracy over 99% and average processing time around 32.6 sec per page. Using MATLAB environment [2].

In our project we intend to design a system that supports multilingual, our system supposed to be portable, we will use one of the most widely available developed systems, which is the mobile system with Android operating system.

- **A Portable device for translation of Braille to literary text**

  This paper presents the development of a portable device for the translation of embossed Braille to text. The device optically scans a Braille page and outputs the equivalent text output in real time, thus acting as a written communication gateway between sighted and binds or vision impaired people [3].

  In our project, we will use smart mobile with high-resolution camera instead of using microcontroller, PC, scanner and other equipment that was used in this project, in term of cost our project will be less than this project.

- **Optical recognition of Braille writing using standard equipment**

  The goal of this research is to develop a system that converts, within acceptable constraints, Braille image to computer readable form (text). By having an image on the computer-using house scanner and the output will display as a text file to the user on a personal computer (PC) [4].

  In term of portability, our project will use the mobile instead of the PC, in addition to the camera, which is build-in in the smart mobile instead of using a scanner.
  In term of cost, our project will be less in cost than this project because we just need a smart mobile with high-resolution camera, which is widely used these days.

1.9 Time Plan

1.9.1 First Semester

In this semester, the project will go through the following tasks:

Task [A]: Select the idea

In this task, the team determined the main idea of the project.

Task [B]: Project preparation

In this task, the team visited the blind association in Hebron, and studied some related projects.

Task [C]: General project analysis

In this task, the team studied the alternatives systems related to project.

Task [D]: Determining requirements

In this task, the team determined all requirements (software and hardware) that are needed in all possible alternatives.

Task [E]: Study project principle

In this task, the team studied the principle tools for each alternative.

Task [F]: Study related theory and collecting information

In this task, the team studied related theory of Braille language, in addition to image processing algorithms that are needed in the project.

Task [G]: System conceptual design and analysis

In this task, the team designed block diagram for each design option and analyze each one individually.

Task [H]: Project documentation

In this task, the team has to document all the collected data and information about the project.

The time plan for the first semester is shown in Table 4.1.

Table 1-4 First semester time plan.

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	▨															
B	▨	▨														
C			▨	▨	▨											
D						▨	▨	▨								
E								▨	▨	▨						
F			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
G										▨	▨	▨	▨	▨		
H			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

1.9.2 Second Semester

In the second semester, the project will go through the following tasks:

Task [I]: Project implementation

In this task, the team will go through the implementation for the BT/VC.

Task [J]: Project testing and maintaining

In this task, the team tends to test the project working and maintain the problem that appears in the system.

Task [K]: Project documentation

In this task, the team has to continue through documentation.

The tentative time plan for the second semester is shown in Table 1.5.

Table 1-5: Second semester time plan

Week / Task	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
I	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
J												▨	▨	▨	▨	▨
K			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Chapter 2 Background and Literature Review

2.1 Overview

This chapter will present a theoretical background for the subjects that are related to this project including the principle of Braille language and the rules of this language, some image processing algorithms that will be used in this project and overview of Android operating system.

2.2 Braille Script

2.2.1 Braille History

Braille is the system of reading and writing used by people who are blind where they feel raised dots on a Braille page with tips of their fingers.

The Braille code invented by “Louis Braille” Figure 2.1 in the early of nineteenth century, It is coming from the military system which used raised dots to send the message at the night. The Braille code system has become widely used by several communities because of its simplicity, comfortably and usability in reading and writing for blind people. Braille was applied or translated into several languages including Arabic and English languages [5- 7].

The Braille script construct of cells, each cell contains 6 raised dots that arranged in three rows and two columns, which are numbered 1 through 3 from top to bottom on the left, and 4 through 6 from top to bottom on the right according to Figure 2.2.

Note that both Arabic and English Braille scripts read from left to right.

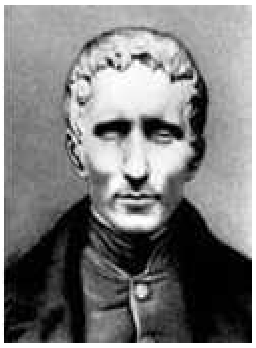


Figure 2.1: Louis Braille

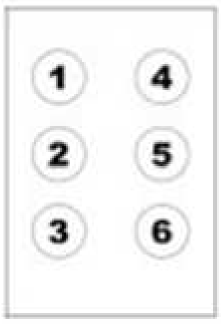


Figure 2.2: Braille Cell

These six dots can be raised or flat according to the character, so these dots are used in combination to give 63 (2^6-1) different characters within a single cell. In addition, this clearly means that there cannot be one to one correspondences between Braille cell and word.

2.2.2 Braille Dimensions

The dimensions of a Braille cell have been set according to the tactile resolution of the fingertips of person. There are horizontal and vertical fixed distances between dots in a cell, and between cells. Each dot has approximately 0.02 inches or (0.5 mm) in height, the horizontal and vertical spacing between dot centers within a Braille cell is approximately 0.1 inches or (2.5 mm), the blank space between dots on adjacent cells is approximately 0.15 inches or (3.5 mm) horizontally and 0.2 inches or (5.0 mm) vertically as shown in Figure 2.3. A standard Braille page is 11 inches by 11.5 inches and typically has a maximum of 40 to 43 Braille cells per line and 25 lines [8].

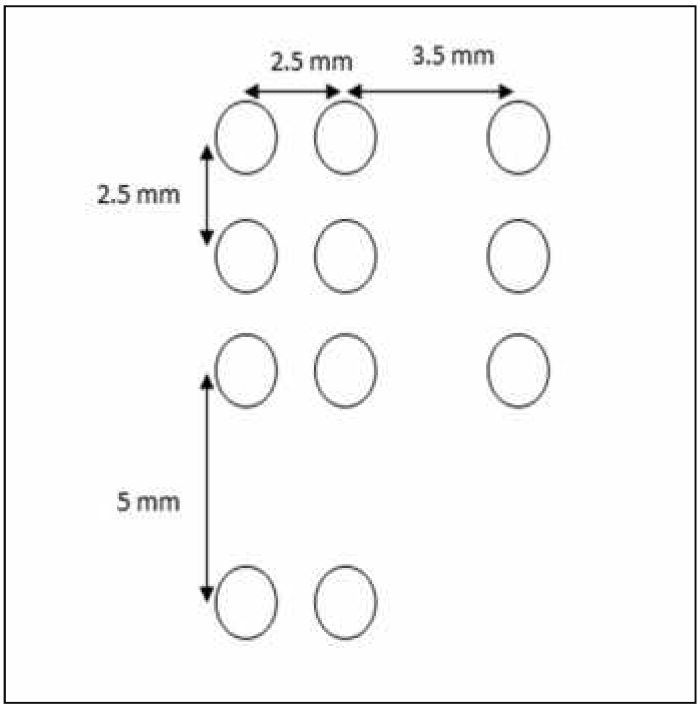


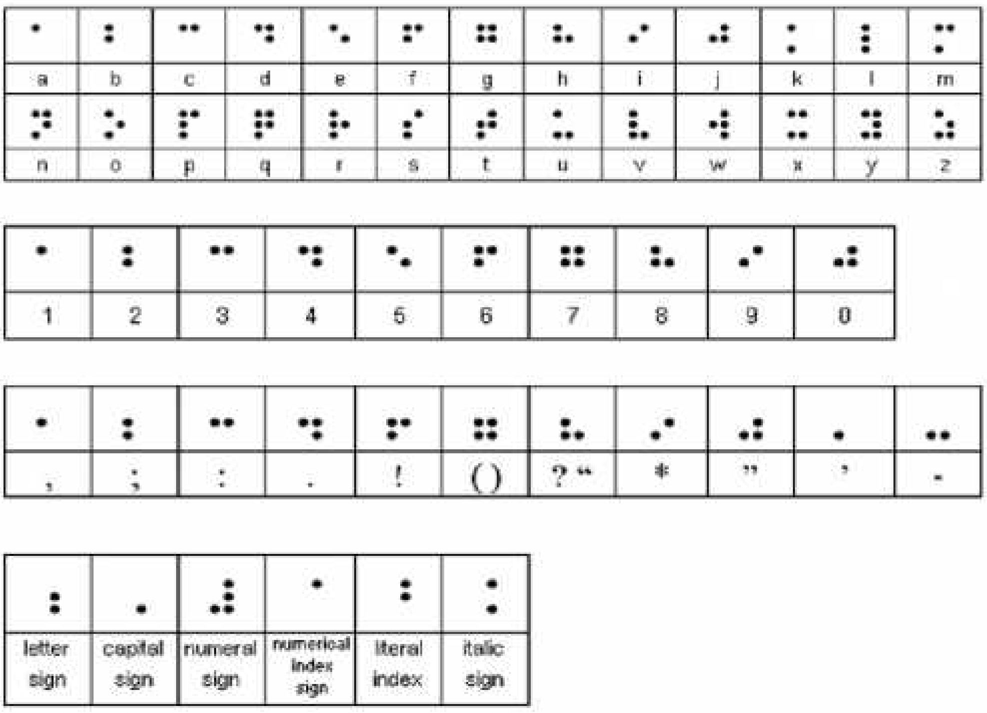
Figure 2٣.: Braille Standard Dot Distance

2.2.3 The Braille Alphabet

- **Grade 1**

Grade 1 is the simplest one, in grade 1, each possible arrangement of dots within a cell represents only one letter, number, punctuation sign, or special Braille composition sign it is a one to one conversion as shown in Figure 2.4.

Individual cells cannot represent words or abbreviations in this grade of Braille. Because of this grade's inability to shorten words, books and other documents produced in grade 1, Braille script is larger than normally printed text. Therefore, those who are new to learn Braille script typically use grade 1 Braille [9].



a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z

1	2	3	4	5	6	7	8	9	0

,	;	:	.	!	( )	? “	*	”	’	-

letter sign	capital sign	numeral sign	numerical index sign	literal index	italic sign

Figure 2.4: Grade 1 Braille

By itself, a Braille letter assumed to be in lower case. To show an uppercase letter, the capital sign putted in front of the Braille letter as show in Figure 2.5.

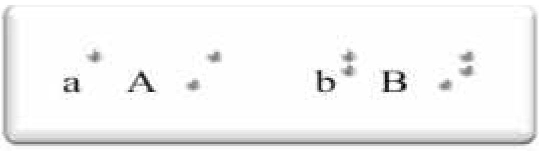


Figure 2.5: Braille capital letter sign

The number sign used in the same way by putting it in the front of cell as shown in Figure 2.6.

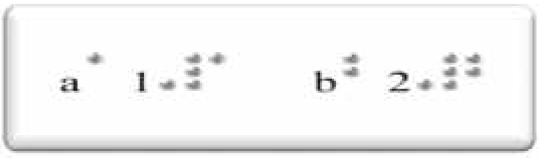


Figure 2.6: Braille number sign

For more information about other Braille, grades see Appendix A.

2.3 Image Processing[10]

Image processing is a very important part in this project; it will be used in order to get the Braille cells out from the Braille paper picture taken by the cell phone camera.

The following section presenting the needed image processing concepts.

2.3.1 Image Processing Overview

Image processing is one of the computer science fields. Image processing operation goes into several steps; Figure 2.7 represents the image processing elements, including image acquisition, which is the first and most important step in any pattern recognition system. The process of acquiring images digitally can be achieved by using a number of different equipment such as scanners or digital cameras or cell phones cameras. The next step is to apply some image processing algorithms; Image preprocessing is an essential operation during this step which eliminate some errors that may occur while acquisition; errors include noise, deformation, bad illumination or blurring. Image preprocessing can be used for image enhancement by reducing noise, sharpening images, or rotating a skewed page. Image processing element has to make some interaction with some storage device to give a new image and may display that image on an output device.

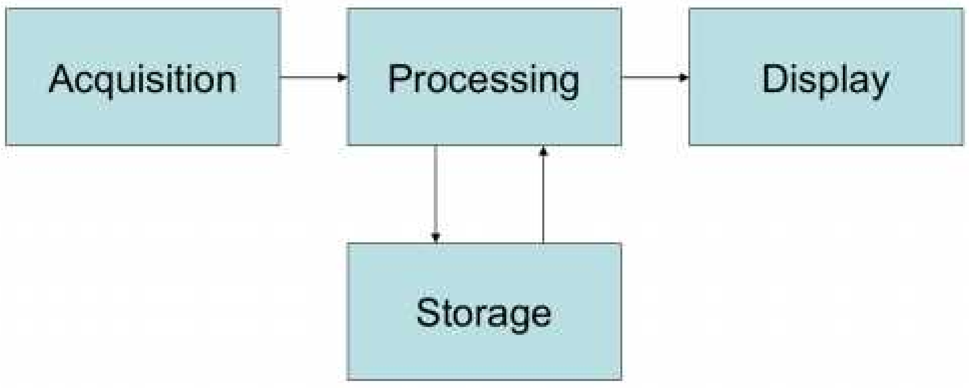


Figure 2.7: Image processing elements

2.3.2 RGB Color Model

It is the same way like the human eye works, and its widely used in computer monitors, televisions, and any other seen media.

It consists of three primary colors (red, green, blue) where the rest of colors can be formed by mixing these colors as shown in Figure 2.8.

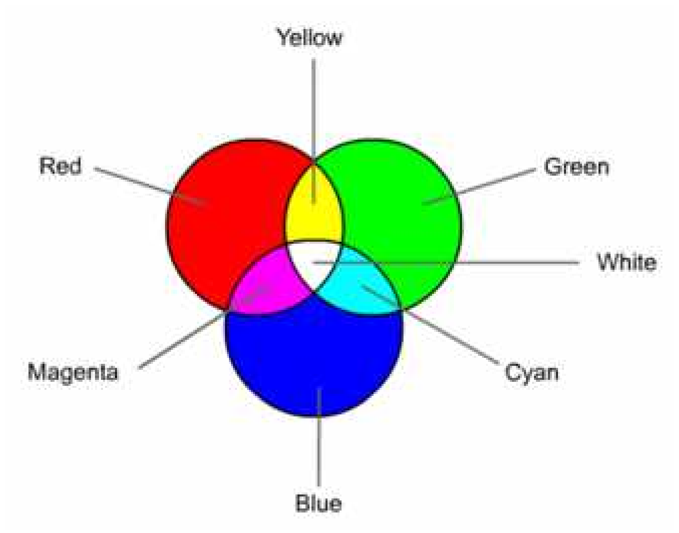


Figure 2.8: RGB color model

In addition, RGB model can be represented as three-dimensional space; the axes are red, green, and blue as shown in Figure 2.9.

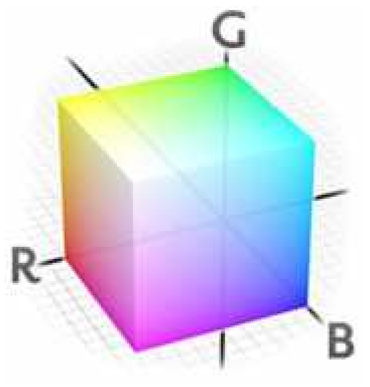


Figure 2.9: 3D - RGB representation

2.3.3 Image Processing Algorithms

Not every image that the user captures is considered perfect, so some image processing algorithms used in order to enhance the image, to get the image ready to be processed. Sometimes we do not need the whole image, so some image processing algorithms used to focus on the wanted parts; this section describes some algorithms that are related to this project.

2.3.3.1 Noise Filtering

The taken images naturally has noise that will make the image unclear and will cause some errors in the subsequent used algorithms, there are some noise filtering algorithms including mean filter, median filter, Gauss filter. These algorithms differs in something called the kernel, where the median filter has no kernel, where the new image has to be computed by running the convolution between each pixel in the original image and the filtering kernel.

In this project, noise-filtering algorithms will be used to decrease the noise effect on the taken image. First, the convolution process will briefly described.

- **Median Filter**

Median filter has no kernel, take eight neighbors where the wanted pixel is the center and sort them ascending take the value that is in the middle of the sorted numbers and exchange it with the center.

123	124	125
129	127	9
126	123	131

Suppose we have the following neighbors of the pixel located at the center which has the value of 127 so when we take the neighbors for it, and sort them ascending, we get the following: 9 – 123 – 123 – 124 – $\textcircled{125}$ – 126 – 127 – 129 – 131 the center value will be changed to 125.

2.3.3.2 Grayscale Image

The RGB image can be represented in one-dimensional array, because it will make the processing easier, faster and will reduce the amount of information in an image.

To covert RGB into Grayscale image by summing the Red, Green and Blue components of the image, then find the average of the summation, then it will generate the following equation:

$$Gray = \frac{Red + Green + Blue}{3}$$

However, the human eye is more sensitive to green component, the following equation used instead: $Gray = 0.3 * Red + 0.59 * Green + 0.11 * Blue$

2.3.3.3 Black and White Image

This is done by choosing a value (threshold), that will be between 0-255 this is done after converting the image to grayscale image, so the values that is less than that value represents a color that had to be changed to white color, represented by the value of 0, the values that is greater than that value represents a color that has to be converted to black, represented by the value of 1.

2.3.3.4 Thresholding Technique[11]

Thresolding is a technique that is used to set all pixels whose is intensity value are above some threshold (T) to a certain color (e.g. black), and all remaining pixels whose intensity under the threshold (T) to another color (e.g. white).

- **Global thresholding**

In the global thresolding technique, the histogram of the input image should appear in two peaks, corresponding respectively to the signals from the background and the object. Global thresholding consists of setting an intensity value (threshold) such that all pixels having intensity value below the threshold belong to one color, the remaining belong to another degree of color . Global thresholding is as good as the degree of intensity separation between the two peaks in the image. It is an unsophisticated segmentation choice.

- **Adaptive thresolding**

It also called (local thresholding) changes the threshold dynamically over the image. This technique is more sophisticated technique than the previous one its can accommodate changing in lighting condition in the image.

Adaptive thresholding typically takes a grayscale or color images as input and in the simplest implementation, the output image will be a binary image representing the segmentation as shown in Figure 2.10. For each pixel in the image, the threshold has to be calculated, if the pixel value is below the threshold it is set as background value, otherwise it will set as the foreground value.

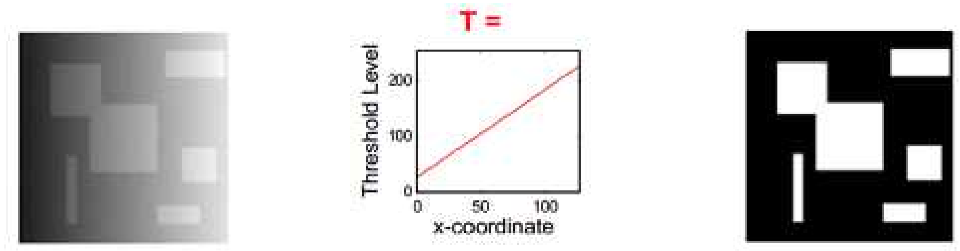


Figure 2.10: Adaptive thresholding

2.3.3.5 Mathmatical Morphology[12]

One of the operations used to analyze the images based on the algebra of non-linear operators operating on object shape, used in images that need Shape simplification, Shape enhancing, Skeletonizing, Thinning, thickening, image segmentation and noise filtering.

The basic operations in mathematical morphology are:

- **Dilation**

Dilation expands the connected sets of 1s of a binary image; it can be used for expanding shape, filling holes and gaps as shown in Figure 2.11.

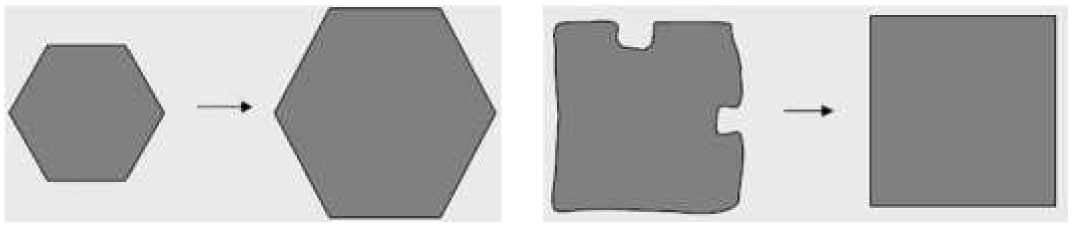


Figure 2.11: Dilation

- **Erosion**

Erosion shrinks the connected sets of 1s of a binary image, it can be used shrinking shapes, removing bridges and branches and small protrusion as shown in Figure 2.12.

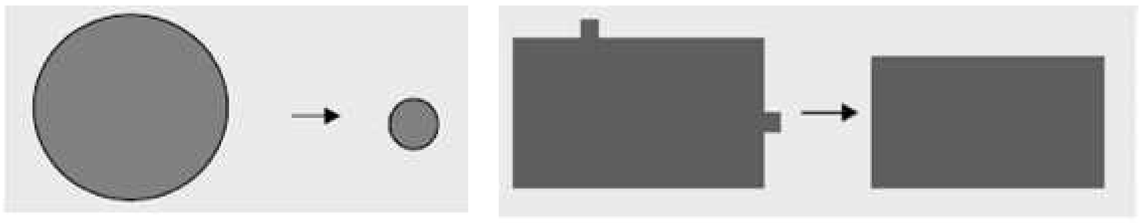


Figure 2.12: Erosion

2.4 Android Operating System

2.4.1 Abstract

Mobile is the most commonly used device in the world, previously cell phones used just for calls between people, so the cell phones at that time had a simple design and features according to its mission, later on the cell phones design has some additional features including Cameras, Bluetooth, etc…

Today, the smart phones appeared with touch screens and a very large number of features, stylish and versatile components, which opens the field to build a huge number of applications, which help to make the life easier.

To build applications for the cell phones we need to know the existing operating system in that cell phone, because every family of cell phones has its own operating system, including windows phone from Microsoft Company, IOS from Apple, Android from Google, and others.

2.4.2 Overview

An open source mobile operating system, released by Google in 2005 to be fit with touch screens smart phones and Tablet devices, the growth increased rapidly especially in 2011 and 2012 as shown in Figure 2.13 **[13]**, which makes it one of the best existing operating systems in the world, Android has goes through several versions since it released, started with Android 1.0 and ended with Android 4.2 each of them has a sweet name like Cupcake, Donut, and the last one is Jelly Bean.

In 2008 Google announced the android market called ”Google Play” which is a site that contains all applications prepared in Android operating system, its permit the users to download any application they want, some of these applications are free and others are paid using visa cards always.[13]

Figure 2.13: Android Growth Rate

2.4.3 Required Tools[14]

The following are the package of tools needed for Android programming environment; each of them installed individually and need some steps to be ready to work together:

Figure 2.14: programming tools

- **Microsoft Visual Studio**

In programming an Android operating system application, you need to have an integrated development environment (IDE) which is Microsoft Visual Studio with C# environment.

- **Android SDK**

The Android SDK contains a debugger, libraries, emulator, sample codes, and tutorials, which can help you in programming and running applications, you can download it from Android developer [8].

- **Mono for Android**

The mono for android is a development tool used to build mobile applications with Android in .Net environment.

2.4.4 Android Contents and Services

Android has some contents and services that can be used in any application you build and there are Activities, Contents Providers, and Intents.

2.4.5 EmguCV[15]

EmguCV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc. The wrapper can be compiled in Mono and run on Windows, Linux, Mac OS X, iPhone, iPad and Android devices.

- **Cross Platform**

EmguCV is written entirely in C#. The benefit is that it can be compiled in Mono for Android and therefore is able to run on any platform Mono supports, including Linux, Mac OS X, iOS and Android.

EmguCV can be used from several different languages, including C#, VB.NET, C++ and IronPython.

EmguCV has many advantages; one of those advantages is that the garbage collection is done automatically.

- **Architecture Overview**

EmguCV has two layers of wrapper as shown in Figure 2.15.

- The basic layer (layer 1) contains function, structure and enumeration mappings which directly reflect those in OpenCV.
- The second layer (layer 2) contains classes that mix in advantages from the .NET world.[10]

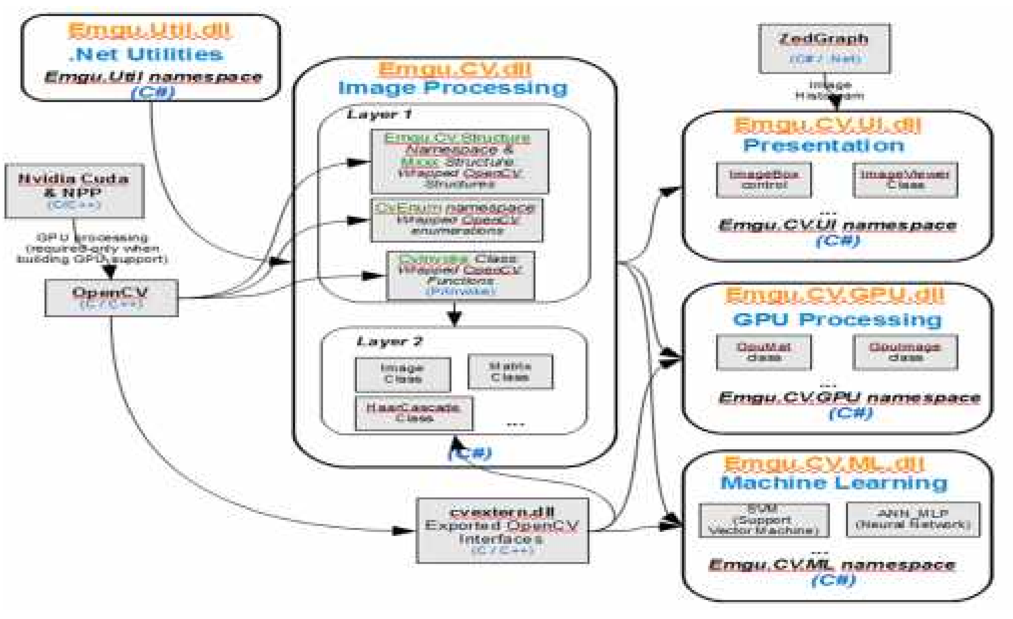


Figure 2.15: Emgu CV Architecture

2.4.6 Android Features

There are number of features in Android that helps to develop applications; some of those features are needed in this project including storage, multimedia [10].

2.4.7 Project Structure In Android

- **Activity**

The activity is the source codes for the project and it always a C# class that contains the code needed for every activity.

- **Resources**

A folder that contains group of other folders such as “drawable” used to save any media file, “layout” contains the xml files, which are the design of every activity, and others.

- **References:**

The references allow the developers to add some library as needed in developing such as EmguCV library for image processing operations.

- **Properties**

The properties file includes all properties for an application such as minimum android target, build properties, and application manifest.

Chapter 3 **System Design and Analysis**

3.1 Overview

This chapter will focus on the detailed system objectives, the possible design options for the system with block diagram for each option, the conceptual block diagram, use cases and activity diagram.

3.2 Detailed System Objectives

The main objective is to convert Braille scripts into other scripts; this section will discuss the system objective in more detailed.

1. Support an application to convert Braille script into multilingual script, to reduce the gap between sighted and blind people, and help the parents to keep track of their blind children study by using this application, without the need to learn Braille language.

2. Support an easy use interface, BT/VC system will allow the user to take Braille image using mobile camera and convert it to selected language in appropriate time.

3. To overcome drawbacks of some related projects. BT/VC system differs among others that it can support multilingual script, and providing flexibility by using voice system to read the given script as an easy way of learning, and provide portable system by using smart phones.

3.3 System Analysis

The system will support converting Braille script into multilingual script, the conversion process passes through several stages, including image acquisition, image preprocessing including image enhancement, converting the image into Grayscale level, by using some image processing techniques to separate the Braille dots from the whole image, the raised dots can be represented in white color, segmenting the image by dividing it into a number of segments, each segment represents a cell which has six dots, in grade 1 a cell represents a character.

A group of cells forms a word, then by analyzing every cell individually to get the character, and convert the cell to its equivalent character using a predefined Hash Table for multilingual script, assembling the converted script and store it in an array of characters then into a file, so that the user will be able to get the converted script as text or voice.

3.3.1 Functional Requirments

1. The system should be able to read Braille image as input.
2. The system should be able to convert Braille image into it equivalent Arabic or English script.
3. The system should support exporting the converted script into text or voice.
4. The system should support multilingual script.
5. The system should be portable.

3.3.2 Non-Functional Requirments

1. The system should be flexible.
2. The system should be efficient.
3. The system should be easy to use.
4. The system should be maintainable.

3.4 Design Options

This section will focus on the options that are available to accomplish this project, there are three options that can handle and use to accomplish this system.

3.4.1 Computer-Based System Diagram

The user scans a Braille paper using a scanner that is connected to a personal computer, after that the Braille paper will stored in PC as image, then in PC side some processing will be done on image using processor, to convert the Braille script and output the result as a text in the screen or as voice using PC speaker as shown in Figure 3.1.

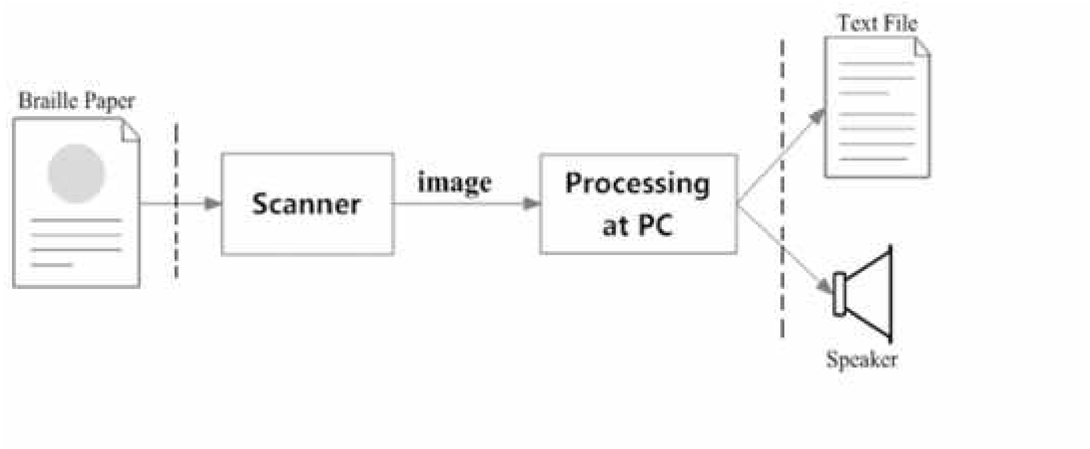


Figure 3.1: Computer-Based Block Diagram

3.4.2 Sensor-Based System Diagram

The user rolls a sensor over the Braille script, which will converts the Braille cells into signals, then the sensor sends these signals to the build-in microcontroller to analyze them, and find the corresponding alphabet character and then output this result as voice using speaker as shown in Figure 3.2.

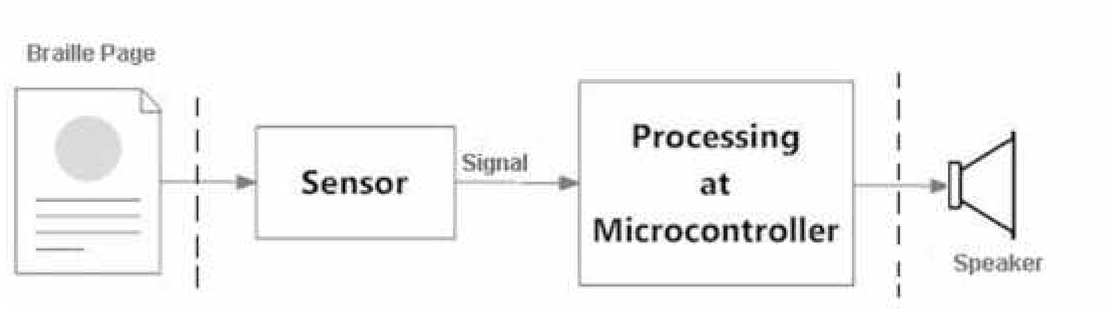


Figure 3.2: Sensor-Based Block Diagram

3.4.3 Mobile-Based System

In this option the user will use smart phones which is widely used today to convert Braille script into sighted people script, by using some features of mobile phones, like the camera which is used to capture the Braille page as an image, also the user will use the voice system of the mobile to output the converted text as sound, in addition to some features including storage, processor as shown in Figure 3.3.

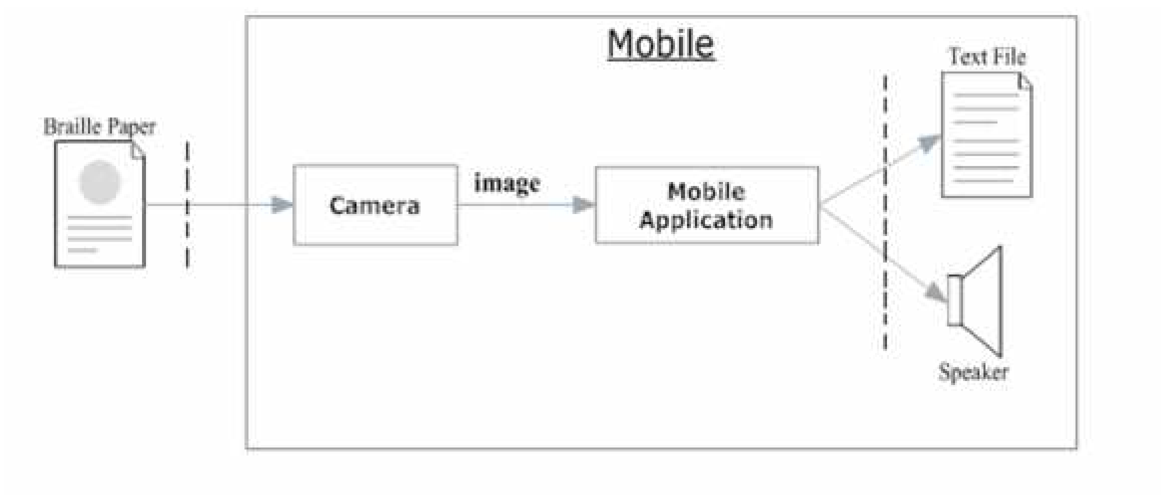


Figure 3.3: Mobile-Based Block Diagram

3.4.4 Design Options Comparison

During the analysis, the team compares among the three available design options in term of portability, flexibility, existence and availability as follow:

Table 3-1: Design Options Comparison

Option / Term	Computer based system	Sensor based system	Mobile based system
Portability	✗	✓	✓
Flexibility	✓	✗	✓
Existence	✓	✓	✗
Availability	✓	✗	✓

According to the comparison, computer based system lacks portability, Sensor based system lacks both flexibility and availability, Mobile based system achieves all terms and not implemented yet, so the candidate one to use is the Mobile based system.

Chapter 4 **Detailed System Design**

4.1 Overview

This chapter will represents the conceptual block diagram of the system, system use cases, the conversion process activity diagram, system sequence diagram and system's graphical user interface.

4.2 Conceptual Block Diagram

Figure 4.1 shows the conceptual block diagram for the BT/VC system, followed by a description for each stage in the system.

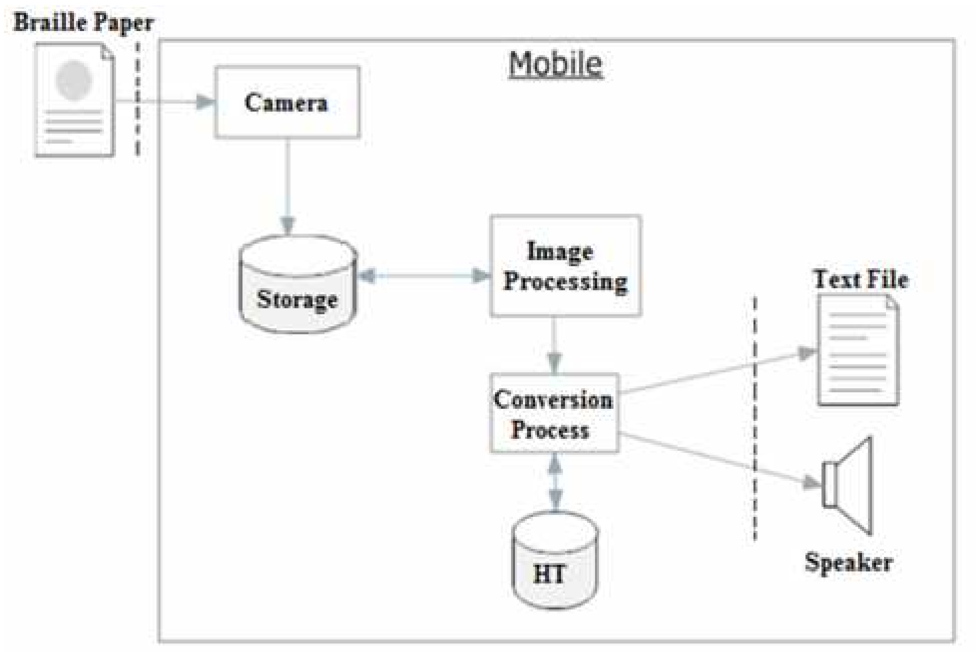


Figure 4.1: System Conceptual Block Diagram

4.2.1 System Stages and Components

- **Braille paper**

Braille paper is the system input, that the user wants to convert it to text or voice, this paper has standard dimensions and properties, such as a fixed number of lines, words and spaces between cells and words as discussed in section 2.1.

- **Camera**

Camera is the device that is used to capture a Braille page; it should have a resolution of at least 3 Mega pixels, to be able to get a clear picture in order to get more accuracy in translation process.

- **Storage**

The storage of the mobile is used to save the pictures taken by camera, in addition to get pictures that are already saved in the storage media to process it.

- **Image processing**

In this stage of the system, the image needs to have preprocessing operations to enhance them as rotation, contrast and noise reduction. In addition, image processing used to start converting image into gray then filters it and finally converts it to black and white to separate raised dots in image to be black or white, which are discussed in section 2.3.

- **Conversion process**

The most important stage in this system, which includes all operations needed to analyze the image and convert each cell to its equivalent binary code then using hash table to convert it from binary to English letter.

- **Hash table**

Hash table has been developed to convert the binary code that was gotten from the conversion process into ASCII Code that represents a character.

- **Text file / speaker**

The output stage of the system, text file contains the converted text for the Braille script, and the speaker used to out the text as voice.

4.3 Use Case Diagram

Use case diagram is a technique for capturing functional requirements of a system from an end user's point of view and it clear and unambiguous description of how the end-user and the system are to interact with each other and to provide a basis for validation testing.

The use cases for the system to be created were generalized for the things, which the user of the system should be able to do, without making any assumptions about how this design is to be implemented.

The use case diagram below Figure 4.2 describes the types of scenarios that the user can participate within the system.

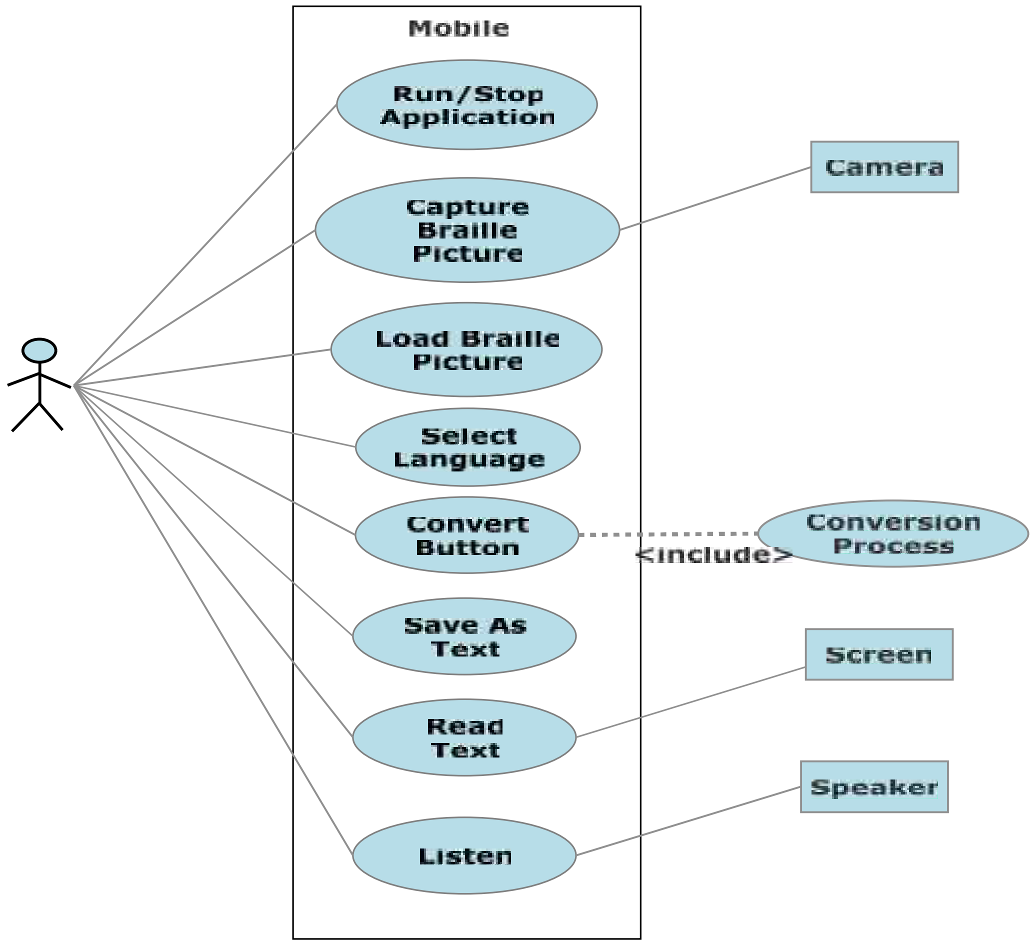


Figure 4.2: Use Cases Diagram

The user would be able to run the BT/VC application, capture Braille picture using mobile camera or load existing Braille image stored in the mobile, select the conversion language, click covert button, save the output as a text file, read output text or listen to the output using mobile speaker and close the application.

4.4 Use Cases Scenarios

- **Run/Stop Application**

Use Case name: Run/Stop Application.
Goal in Context: To run or close the application.
Pre-Conditions: the mobile is turned on.
Primary Actor: user.
Main Success Scenario Steps:

1. Open the application.
2. Exit the application.

Exception:

1. System crash.
2. Mobile crash.

Priority: essential.
When available: always.
Frequency of use: whenever the user want to convert Braille into text.
Channel to actor: Via mobile.
Secondary actors: none.

- **Capture Braille image**

Use Case name: capture Braille image.
Goal in Context: to get the Braille image.
Pre-Conditions: the application is running, mobile camera working.
Primary Actor: user.
Main Success Scenario Steps:

1. Press capture picture button.
2. Capture Braille image in suitable way.
3. Close the camera.

Exception:

1. Camera crash.
2. Not Braille image.
3. Application crash.

Priority: not essential if the Braille image already exists.
When available: always.
Frequency of use: whenever the user want to capture a Braille image.
Channel to actor: Via mobile application.
Secondary actors: Camera.
Channel to secondary actor: via mobile.

- **Load Braille image**

Use Case name: load Braille image.
Goal in Context: to get existing Braille image.
Pre-Conditions: the application is running, image available in mobile storage.
Primary Actor: user.
Main Success Scenario Steps:

1. Press load picture button.
2. Load existing Braille image from storage.

Exception:

1. Image is not available in storage.
2. Not Braille image.
3. Application crash.
4. Storage crash.

Priority: not essential if the user wants to capture a new Braille image.
When Available: when the Braille image is available in the storage.
Frequency Of use: whenever the user want to load a Braille image.
Channel to actor: Via mobile application.
Secondary actors: none.

- **Select conversion language**

Use Case name: select conversion language.
Goal in Context: to select the output language.
Pre-Conditions: the application is running, image is selected.
Primary Actor: user.
Main Success Scenario Steps:

1. User chooses the conversion language.
2. The output text direction will change.

Exception:

Application crash.

Priority: essential.
When available: always.
Frequency of use: whenever the user want to convert an image.
Channel to actor: Via mobile application.
Secondary actors: none.

- **Conversion**

Use Case name: conversion command.
Goal in Context: to convert the Braille image into text.
Pre-Conditions: the application is running, image is already getting.
Primary Actor: user.
Main Success Scenario Steps:

1. Press convert button.
2. Call conversion algorithm.
3. Convert Braille image into text.

Exception:

1. Image is not available.
2. Application crash.

Priority: essential.
When available: always.
Frequency of use: whenever the user want to convert a Braille image.
Channel to actor: Via mobile application.
Secondary actors: none.

- **Save as text**

Use Case name: save as text.
Goal in Context: to save the converted Braille image as a text file.
Pre-Conditions: the application is running, image is converted to text.
Primary Actor: user.
Main Success Scenario Steps:

1. Presses save as text button.
2. Get suitable text direction.
3. New text file created.
4. Set name to text file.
5. Transfer the converted text into the created file.

Exception:

1. Cannot find the suitable direction.
2. File name is already exists.
3. File extension not correct.
4. Application crash.

Priority: not essential if the user would not like to save the text.
When available: always.
Frequency of use: whenever the user want to save the converted text.
Channel to actor: Via mobile application.
Secondary actors: none.

- **Read Text**

Use Case name: read converted text.
Goal in Context: To let the user reads the converted text.
Pre-Conditions: the mobile is turned on, application running, image is converted to text.
Primary Actor: user.
Main Success Scenario Steps:

1. Click read text button.
2. Wait until the text is displayed.
3. The user would be able to read the text.

Exception:

1. Cannot access for the converted text.
2. Cannot find the suitable direction.
3. System crash.

Priority: not essential if the user would not like to read the text.
When available: always.
Frequency of use: whenever the user want to read converted text.
Channel to actor: Via mobile application.
Secondary actors: Screen.
Channel to secondary actor: Via mobile application.

- **Listen**

Use Case name: listen converted text.
Goal in Context: To let the user hears the converted text.
Pre-Conditions: the mobile is turned on, image is converted to text.
Primary Actor: user.
Main Success Scenario Steps:

1. Click listen text button.
2. User would be able to listen to the converted text.

Exception:

1. Cannot access for the converted text.
2. Cannot find the text language.
3. Voice library crash.
4. Speaker crash.
5. System crash.

Priority: not essential if the user would not like to hear the text.
When available: always.
Frequency of use: whenever the user wants to listen to converted text.
Channel to actor: Via mobile application.
Secondary actors: speakers.
Channel to secondary actor: via mobile.

4.5 Conversion Process Activity Diagram

The conversion process goes throw several stages in order to convert the Braille image into text or voice.

First, the application checks whether the image is Braille image or not, if it is not the conversion process will end.

If the image is Braille image, the application will get the language that the user chooses, and according to this language the application will decide whether to change the direction of writing or not, then application will get the first cell as a start point, check whether the end of frames(cells) or not, if it is not then apply some image processing algorithms on the cell to get its corresponding ASCII code, this ASCII character will be saved on an array of characters, this array will be saved to a file then, when end of frame is reached then the conversion is finished as shown in Figure 4.3.

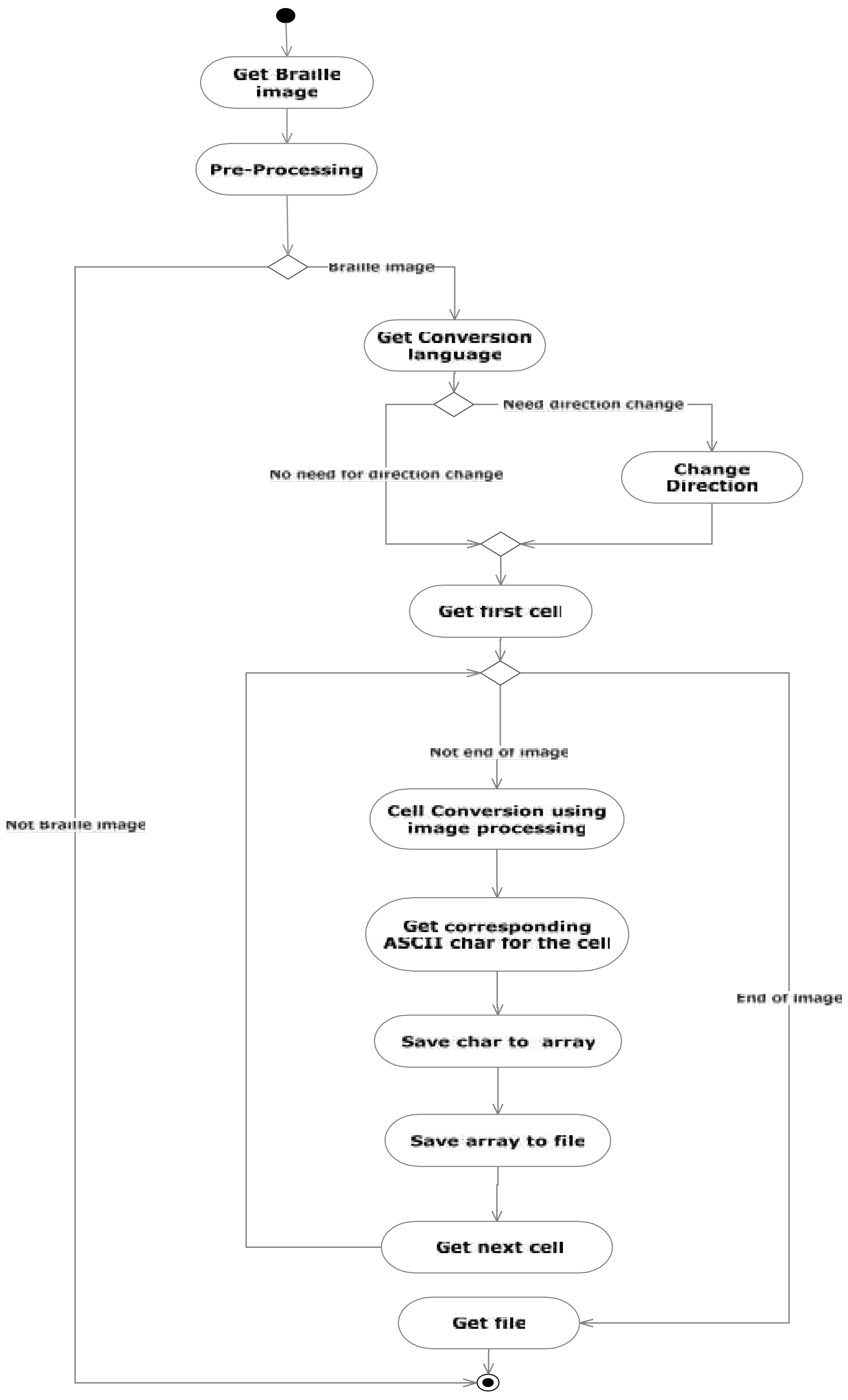


Figure 4.3: Conversion process activity diagram

4.6 Sequence Diagram

The following Figure 4.4 represents the main system sequence diagram, as shown there are three main objects in the system: System user, Mobile Device and Mobile Application.

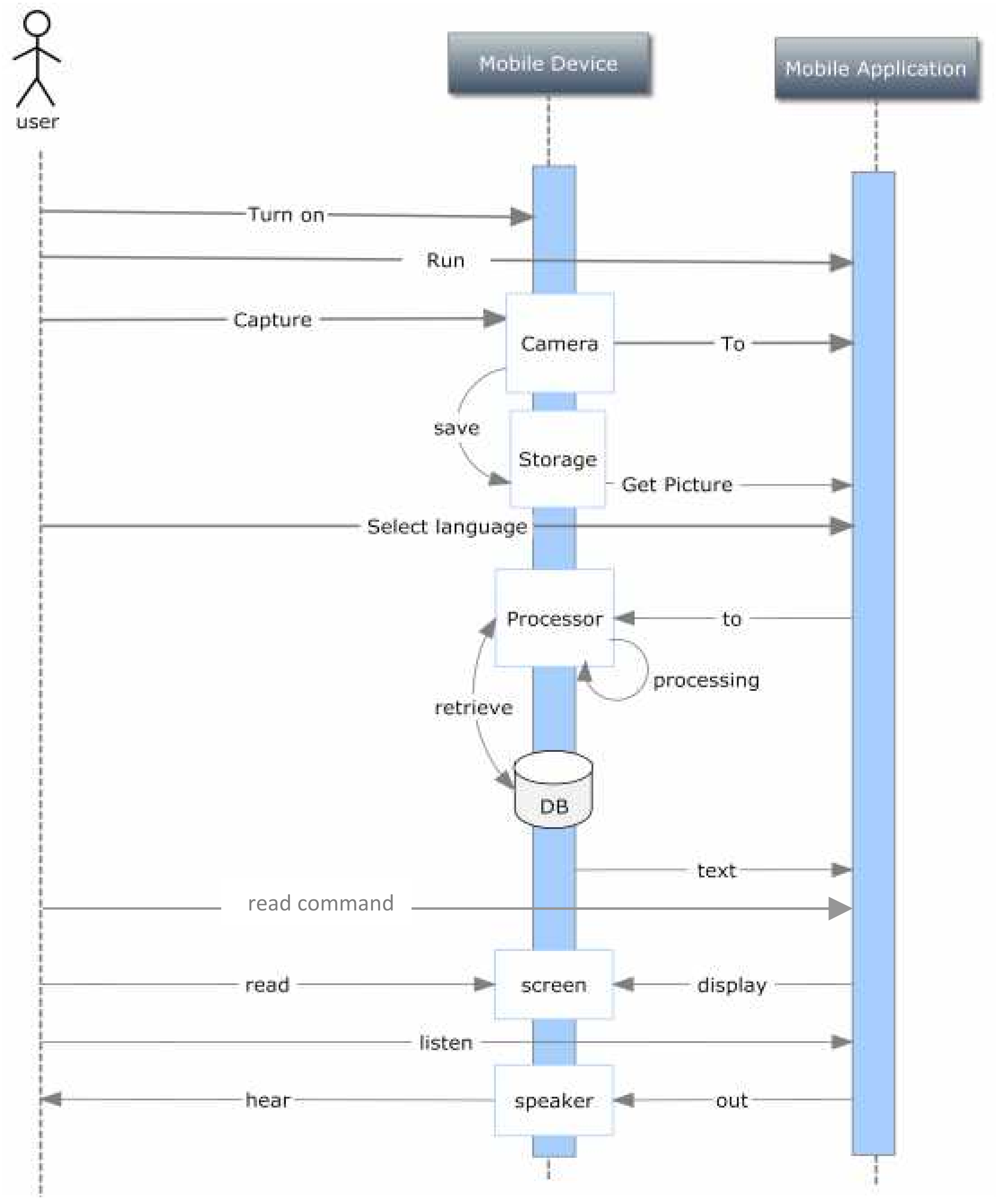


Figure 4.4: Sequence Diagram

First of all the user turns on the mobile device if it is not turned on, then the user will runs the application on the mobile device, after that the user has to choose whether to capture a photo for the Braille paper using the mobile camera that has to be saved in the mobile storage, or to choose the Braille paper image from the pictures library that is already saved in the storage, the application then gets the selected image from the mobile storage waiting for the user command to select the conversion language, the application recognizes the selected language and do some processing including image processing on the selected image and character conversion on the mobile processor that are connected to the conversion database that we will create, the program then waits for a command to export the converted text, the user has two choices , whether to read or to hear the converted text, if the user chooses to read the text the application has to display it using the mobile display, if the user selects to hear it then the program has to get the converted text out to the device speaker.

4.7 Detailed System Implementation

The system divided into three main stages, and they are dependent. In other words, each stage depends on the previous one. The first stage is to get the Braille paper as image weather by capturing a new Braille paper or by loading an existing image. The second stage is the conversion process; in this stage, the Braille image will be scanned using BT/VC algorithm to convert each cell individually. In the third stage, the result will be displayed to the user weather as text or voice. The following points will explain each stage individually:

- **Getting Braille paper as image**

  In this stage, the system will allow the user to capture the image using mobile camera or to load an existing Braille image from the storage, the following image Figure 4.5 has been gotten during this stage.

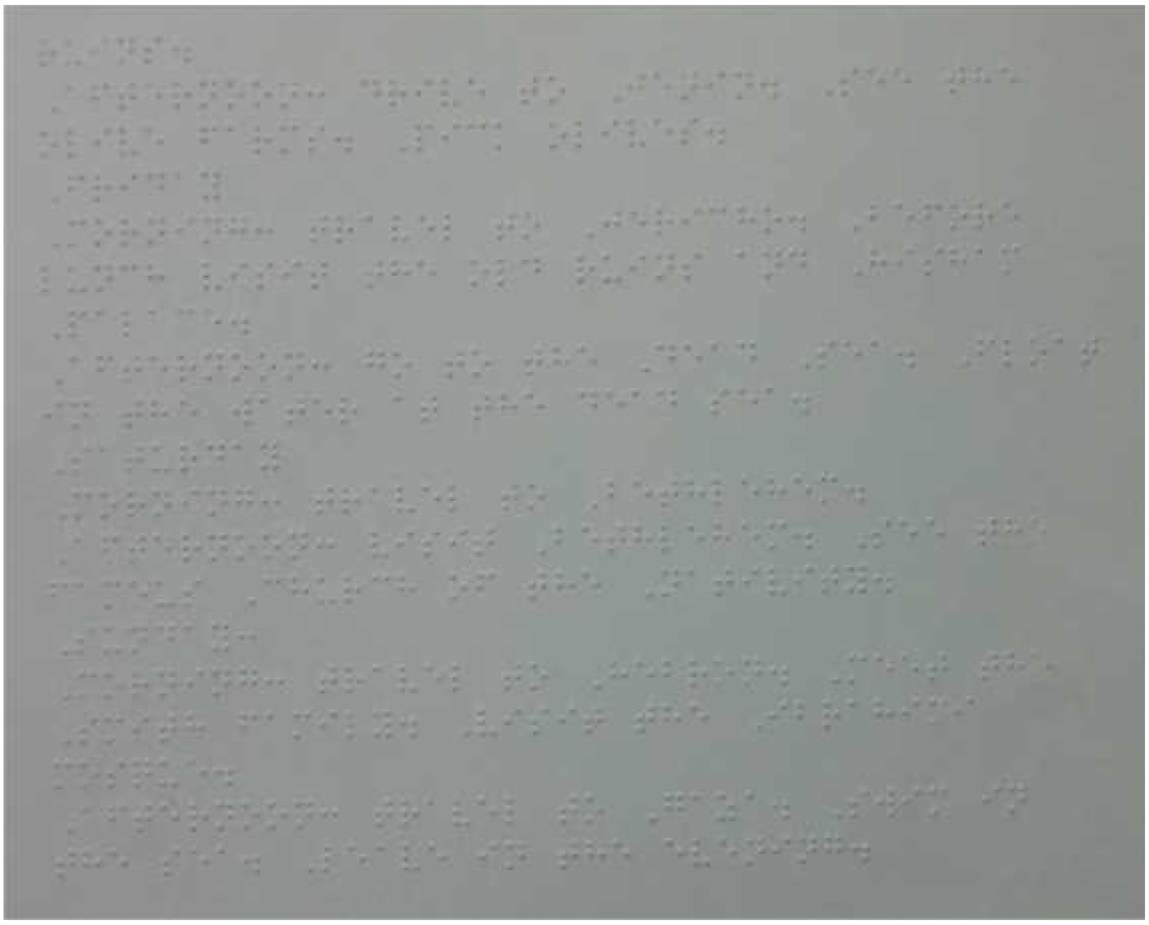


Figure 4.5: Captured Image

- **Conversion process stage**

  This stage includes all the operations that have been applied on the Braille image:

1. The first operation is to separate the dots from the background, this operation is done using local (adaptive) thresolding. The following Figure 4.6 shows the image after the adaptive thresholding has been applied:

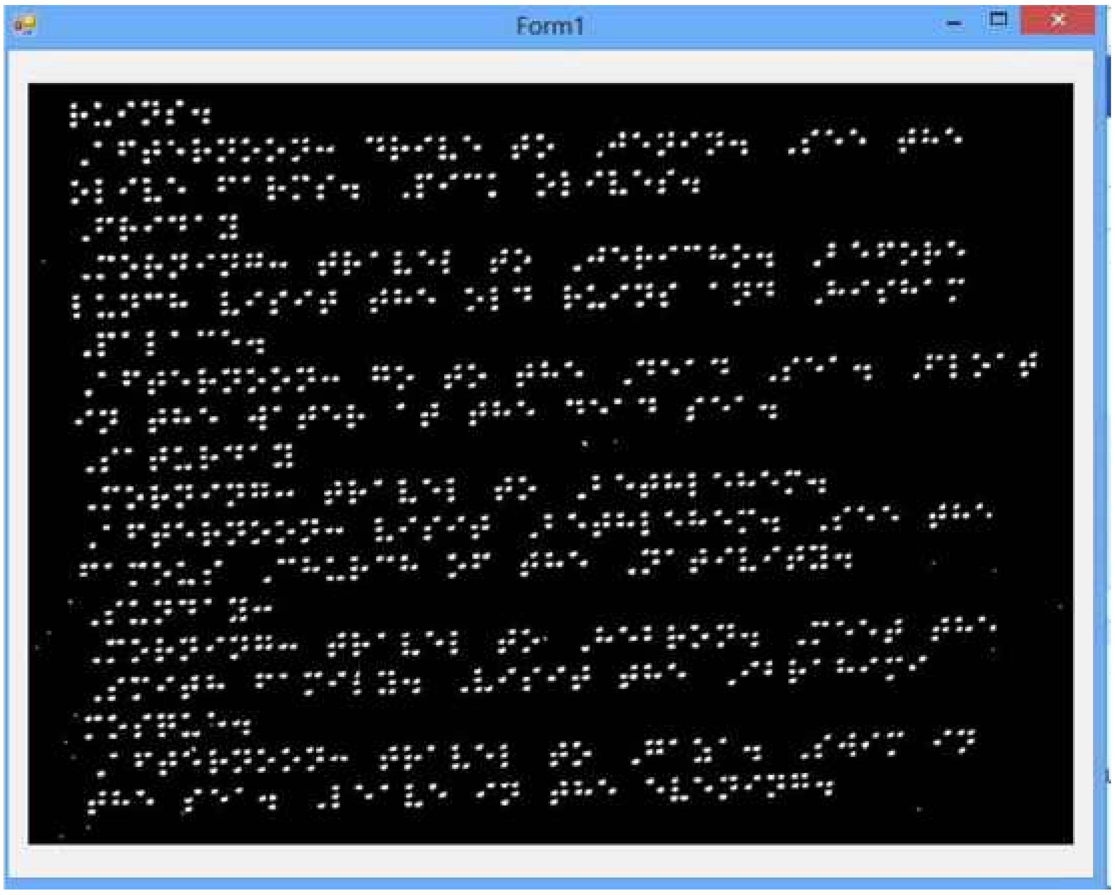


Figure 4.6: Image after adaptive thresolding.

2. The second operation is to ensure that the captured image is a Braille image, and this will done by calculate the sum of pixels values in a 100 by 100 window from the image, if the sum of these values are above a certain threshold, the image is considered to be non- Braille image, else it will considered to be a Braille image.

3. The third operation is to remove the noise and enhance the dots, the image was filtered and the erosion and dilation image processing techniques have been applied to the image as shown in Figure 4.7 in order to enhance dots and make the image ready for the next stage.

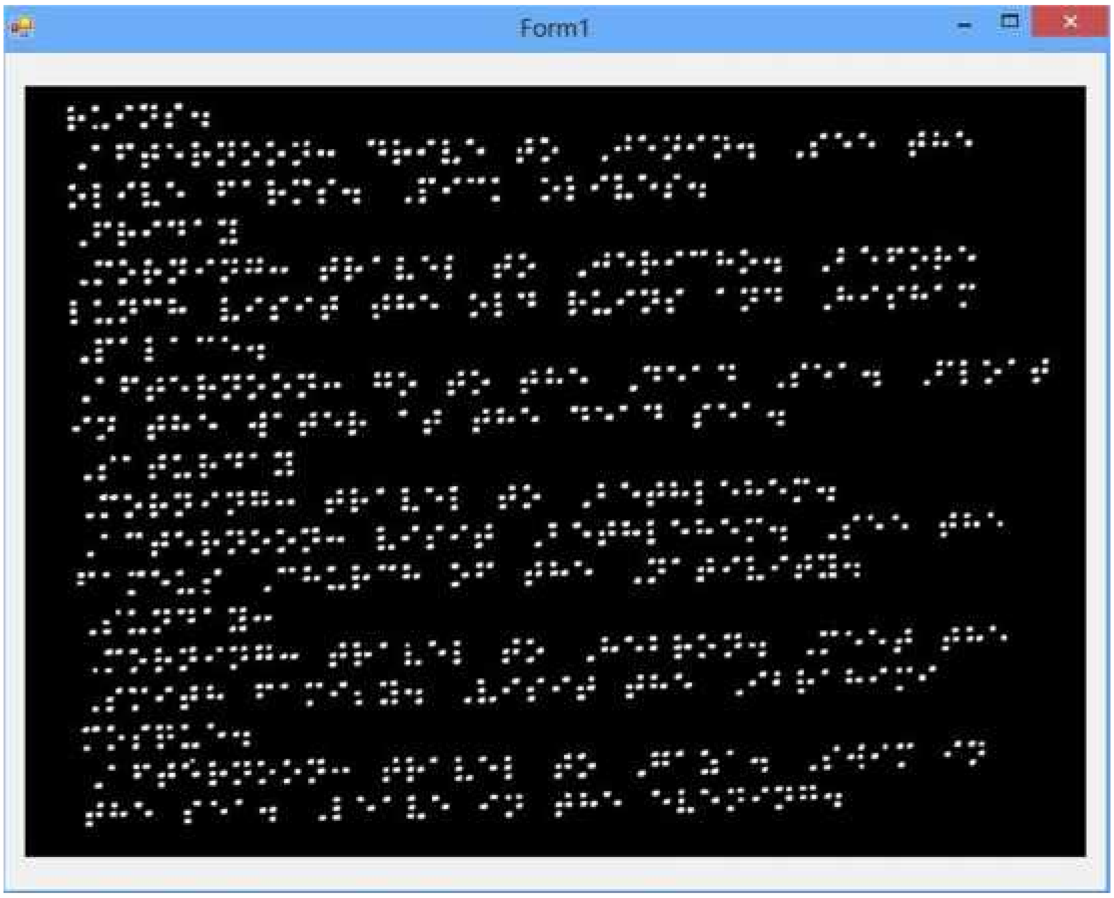


Figure 4.7 : image after noise removal

4. The fourth operation is to correct the gradient (slop) of the image in order to get a slat image as much as possible as shown in Figure 4.8.

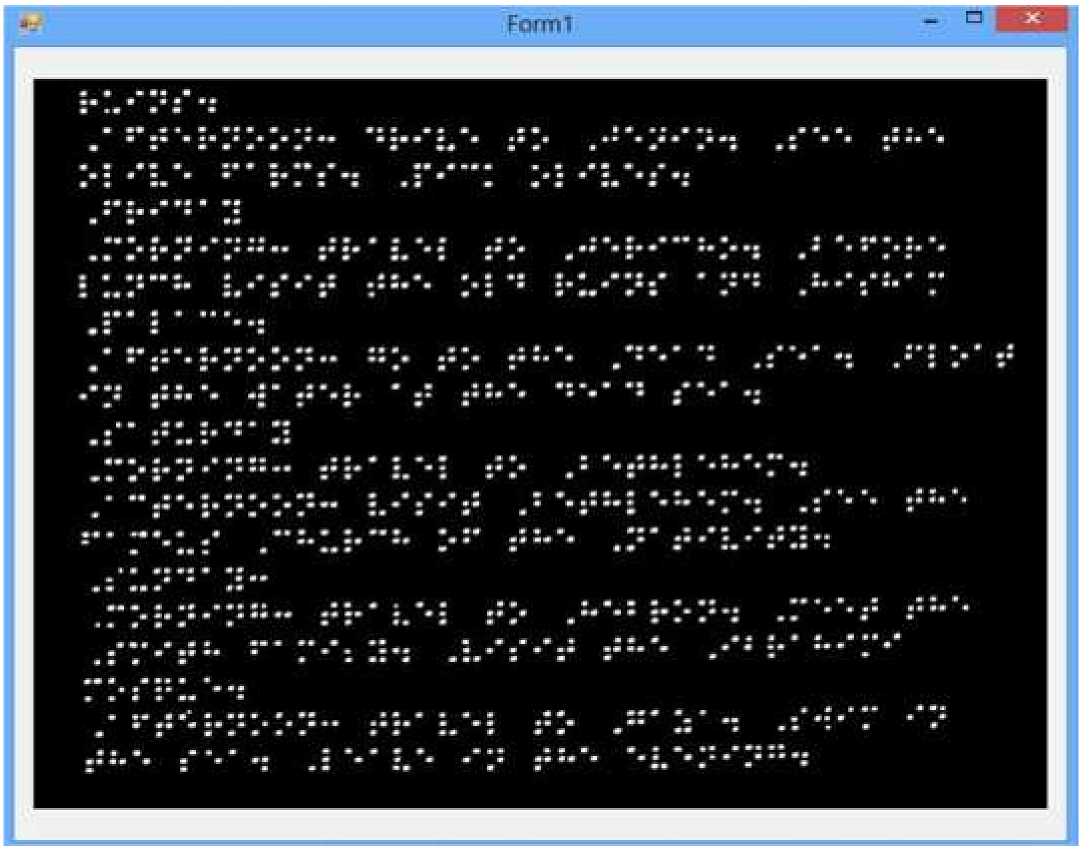


Figure 4.8: Image after gradient correction

5. In this stage, which is the final stage, the Braille cells and dots have been recognized as shown in Figure 4.9 and each cell will has its own binary code, which is represent the equivalent character, and this binary code will send to the hash table and an ASCII code will be retrieved from a hash table.

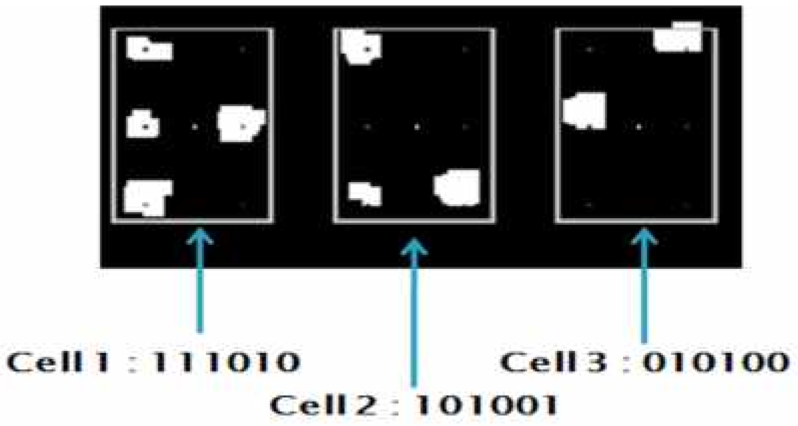


Figure 4.9 : Cell recognition

4.8 UML Diagram

The system contains five main classes as shown in Figure 4.10:

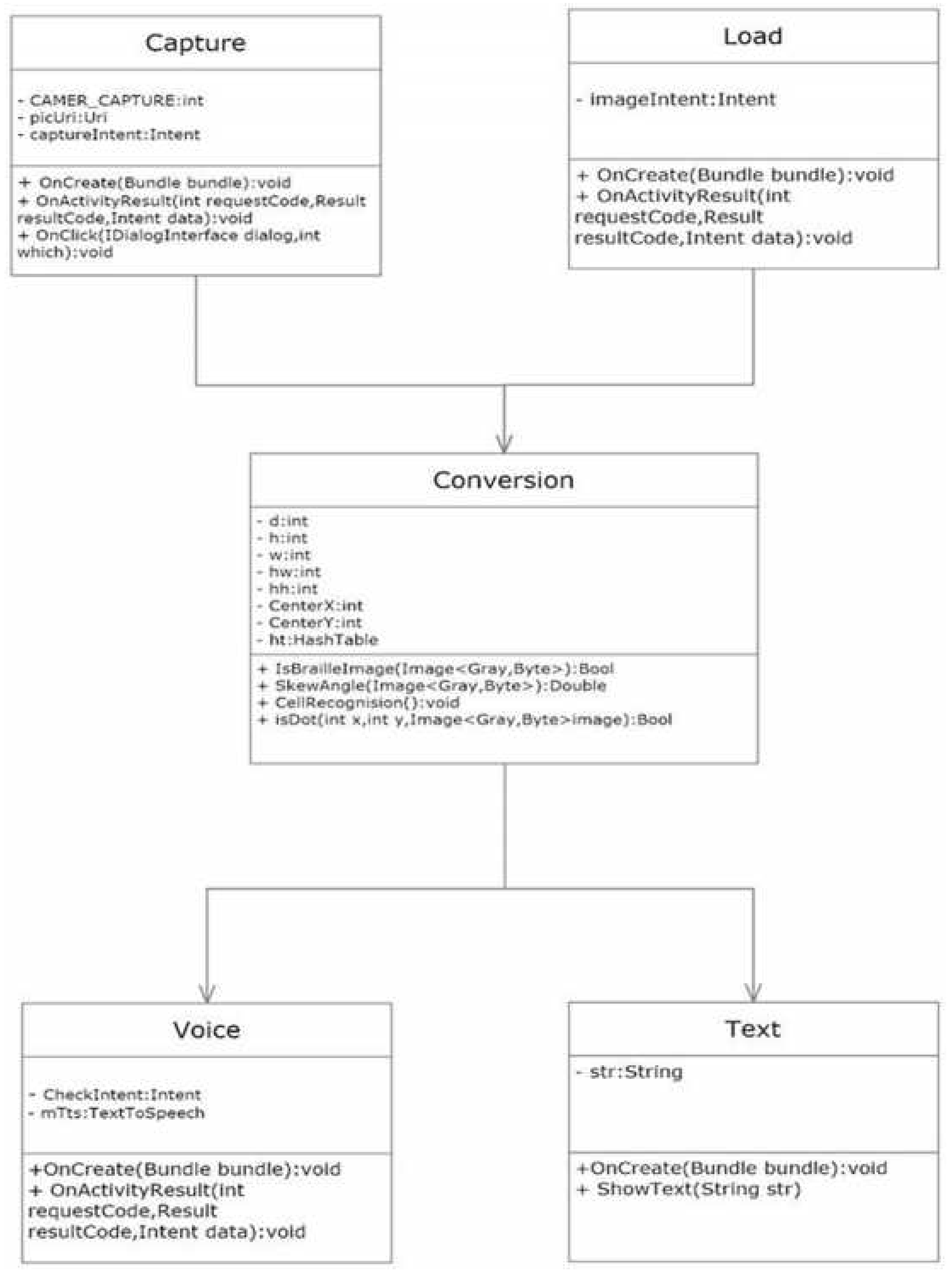

Capture
- CAMER_CAPTURE:int
- picUri:Uri
- captureIntent:Intent
+ OnCreate(Bundle bundle):void
+ OnActivityResult(int requestCode,Result resultCode,Intent data):void
+ OnClick(IDialogInterface dialog,int which):void
Load
- imageIntent:Intent
+ OnCreate(Bundle bundle):void
+ OnActivityResult(int requestCode,Result resultCode,Intent data):void
Conversion
- d:int
- h:int
- w:int
- hw:int
- hh:int
- CenterX:int
- CenterY:int
- ht:HashTable
+ IsBrailleImage(Image<Gray,Byte>):Bool
+ SkewAngle(Image<Gray,Byte>):Double
+ CellRecognision():void
+ isDot(int x,int y,Image<Gray,Byte>image):Bool
Voice
- CheckIntent:Intent
- mTts:TextToSpeech
+OnCreate(Bundle bundle):void
+ OnActivityResult(int requestCode,Result resultCode,Intent data):void
Text
- str:String
+OnCreate(Bundle bundle):void
+ ShowText(String str)

Figure 4.10: System UML Diagram

4.8.1 Capture Class

This class will be used to allow the user to capture a Braille paper and save it as image in the storage device, then to show it in an image view.

- **Variables**

1. **Camera_Capture:** an integer variable represents the request code for the intent to capture image.
2. **picUri:** variable with type of URI used to represent the path of the captured image.
3. **captureIntent:** The intent that allow the user to capture image.

- **Operations**

1. **OnCreate:** The first method called when the activity is started.
2. **OnActivityResult:** The method that called after the command startActivity, and here this method is saving the captured image, or cropping the captured image according to the request code.
3. **OnClick:** A button event that show a welcome message.

4.8.2 Load Class

This class will be used to allow the user to load a picture from mobile and to show it in an image view.

- **Variable**

imageIntent: intent that allow the user to loan an image from mobile.

- **Operations**

1. **OnCreate:** first method called when the activity is started.
2. **OnActivityResult:** The method that called after the command startActivity, and here this method is saving the loaded image.

4.8.3 Conversion Class

This class include the conversion process from Braille to text and export the result to the final stage, the main goal from this class is to segment each cell in Braille paper, then divide each cell into six regions, in order to check if there is a dot in this region or not, finally the binary code will be gotten and will send to a hash table and get an ASCII value for each binary code.

- **Variables**

1. **d:** This variable is used as a distance for accessing the center of the dots region.
2. **h:** This variable is referred to the height of the cell, and its three time greater than the width of the cell.
3. **w:** This variable is referred to the width of the cell.
4. **hw:** Represent half of the value of the width of cell.
5. **hh:** Represent half of the value of the height of cell.
6. **CenterX:** Represent the center of the cell in the x coordinate.
7. **CenterY:** Represent the center of the cell in the y coordinate.
8. **document**: is a string variable represent the text that is converted, and this increased when each cell is converted.

- **Operations**

1. **IsBrailleImage:** This method is used to confirm that the loaded image is a Braille image.
2. **SkewAngle:** This method is used to find the angle of the gradient image.
3. **CellRecognision:** This method is used to recognize the cells in the Braille paper.
4. **isDot:** A method is used to check the region is has a dot or not.

4.8.4 Text Class

- **Variable**

str: String variable to get the converted text and display it.

- **Operations**

1. **OnCreate:** first method called when the activity is started.
2. **ShowText :** A method to display the text to the text area for the user.

4.8.5 Voice Class

This class is used in the last stage of the application, used to export the result as voice to the user.

- **Variables**

1. **checkIntent:** A variable of type intent used to make text to speech action.
2. **mTts:** variable of type TextToSpeech used to speak the text that converted.

- **Operations**

1. **OnCreate:** first method called when the activity is started.

2. **OnActivityResult:** The method that called after the command startActivity, and here used to convert the text to speech.

4.9 Conversion Class Detailed Implementation

The first step is to determine the center of the cell that represented in Figure 4.11 as red dot.

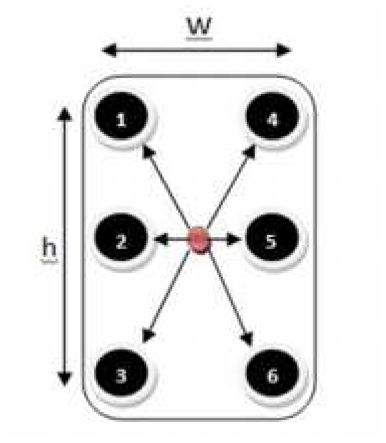


Figure 4.11: Braille Cell

Assume that x and y coordinates setting at the left top corner of the cell which is (0, 0) coordinates, so the following equations are used to obtain the center of the rectangle.

*CenterX =x+ 0.5*w.*

*CenterY =y+ 0.5*h.*

The following equation will help to reach the dot region in such cell

*h = 1.5 * w.*

*hw=0.5*w-d.*

*hh=0.5*h-d*

Note that d variable represents a small distance, and it has been subtracted from the half width and height in order to reach the center of dot region.

Now the following coordinates are suggested to used in order to reach the region of each dot individually from the center point CenterX , CenterY

Dot1: (centerX-hw,centerY-hh)

Dot2: (centerX-hw,centerY)

Dot3: (centerX-hw,centerY+hh)

Dot4: (centerX+hw,centerY-hh)

Dot5: (centerX+hw,centerY)

Dot6: (centerX+hw,centerY+hh)

Now, after we reached a pixel (assume it is was in the center of the dot) which is in the region of the dot we want to look on the neighborhood of that pixel, if there is at least on pixel weather it was in the center or on its neighborhoods have the value above some threshold T, so it is considered to be a raised dot, if not its will consider to be a flat dot.

- **Presudocode BT/VC cell and dot recognition algorithm**

```
Main Function()
Define document as String
Assume i,j = staying at top left corner of the cell
h = 1.5 * w
 hw=0.5*w -d
 hh=0.5*h-d.
For i =0 to image width
   For j=0 to image height
     CenterX =i+ hw
     CenterY =j+ hh
  document = document + hashtable(CellRecognision(centerX,centerY,image))
i= i + h
End for
j=j+(w+h)
End for
End main function
Fuction CellRecognision(centerX,centerY,image))
Define C as String
       If     IsDot(centerX-hw,centerY-hh,image)
C =”1”
Else C = “0“
       If    isDot(centerX-hw,centerY,image)
C+=”1”
Else C+=”0”
       If    isDot(centerX-hw,centerY+hh,image)
C+=”1”
Else C+=”0”
       If    isDot(centerX+hw,centerY-hh, image)
C+=”1”
Else C+=”0”
       If    isDot(centerX+hw,centerY, image)
C+=”1”
Else C+=”0”
       If    isDot(centerX+hw,centerY+hh)
       C+=”1”
       Else C+=”0”
       End function
Function bool isDot(dot_C_X,dot_C_Y,image Dotsim)
       Define T as Int  //Threshold
       For i=x TO dot_C_X +value
       if    Dotsim.Data[i, dot_C_Y] > T
       IsDot = true
       Break
       i=i+1
       End for

       For i=x TO  dot_C_X -value
       if  Dotsim.Data[i, dot_C_Y] > T
       IsDot = true
       Break
       i=i+1
       End for
       For j=xj TO  dot_C_Y +value

       if Dotsim.Data [dot_C_X, j] > T
       IsDot = true
       Break
       j=j+1
       End for
       For i=x  TO  dot_C_X -value
       if    Dotsim.Data[dot_C_X, j>T
       IsDot = true
       Break
       i=i+1
       End for
       Return IsDot
       End bool function
```

4.10 Graphical User Interface

Graphical User Interface is a very important criteria for the application quality, because it is the main component that the end users deal with, regardless of how the system designed.

The following sections discuss the graphical user interface for the BT/VC system:

- **Start Window**

The first window that the system runs, the user will get the picture that he wants to convert using the application, and there are two ways to get the picture: the first one by using the mobile camera and the second one is to load an existing picture as shown in Figure 4.12.

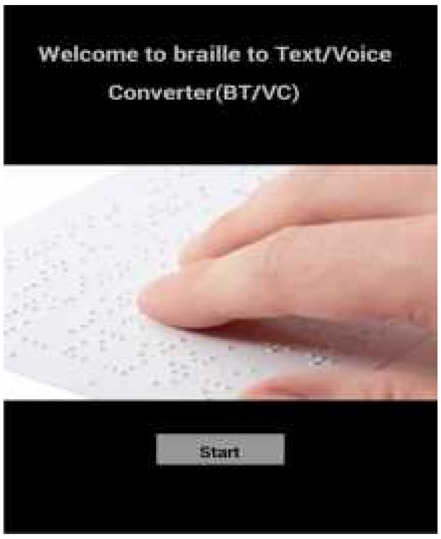


Figure 4.12: Start Window

- **Conversion Window**

In this stage, the user chooses whether to capture an image or load an existing one, and then the selected picture will be displayed to user, after that the user will press a convert button waiting for the result window as shown in Figure 4.13.

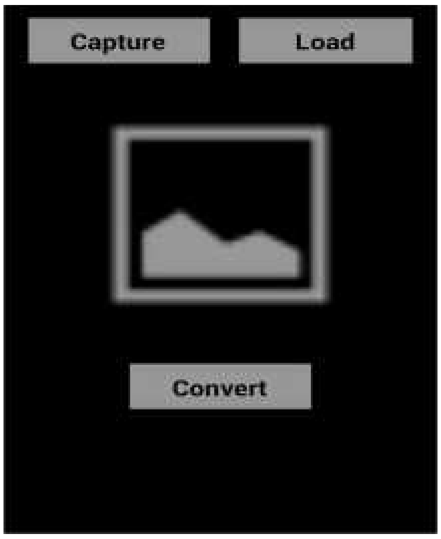


Figure 4.13: Conversion Window

- **Result Window**

When the conversion process completed, this window will appear to ask the user to choose the result way, weather to read the result as a text file or to listen it using media player as shown in Figure 4.14.

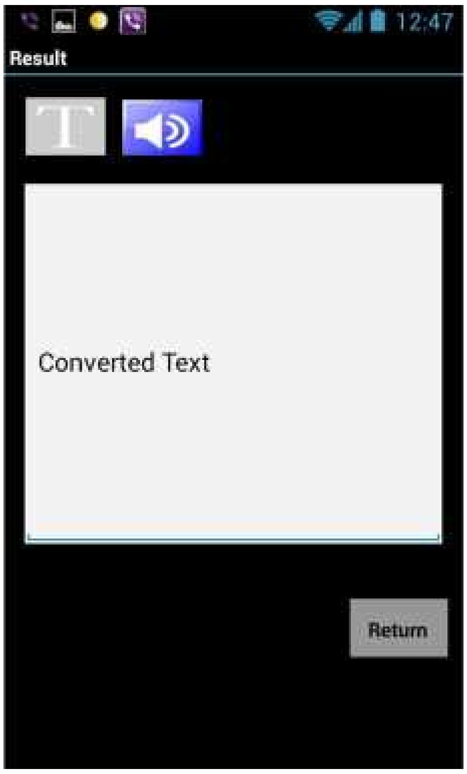


Figure 4.14: Result window

Chapter 5 Testing and Results

5.1 Overview

During the implementation stage, the team tests each single part of the system individually and the results have been recorded. This chapter will focus on the testing stage of the project systematically, and finally a summary for the testing scenarios.

5.2 Installing and Preparing the System

This section describes how to install the application on mobile and preparing it to be ready to use.

After the completion of implementing the mobile application, using Microsoft Visual Studio with C# language and mono for android plug-in with EmguCV image processing library, it should be deployed on the mobile to be ready to use, and there are two methods for deploying:

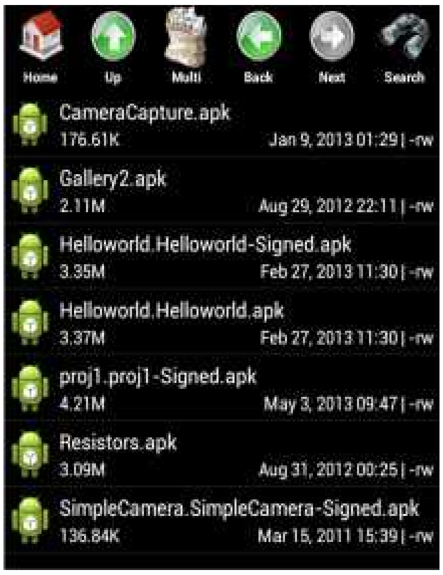
5.2.1 Using USB cable or Bluetooth

The first is step is connecting the USB cable to the mobile, and activating the USB option to be checked, then copying the .apk file from computer to any folder in mobile SD card, the apk file can be found in the following path in computer:

<Project-Directory>\bin\Release\<**project_name>-Signed.apk**

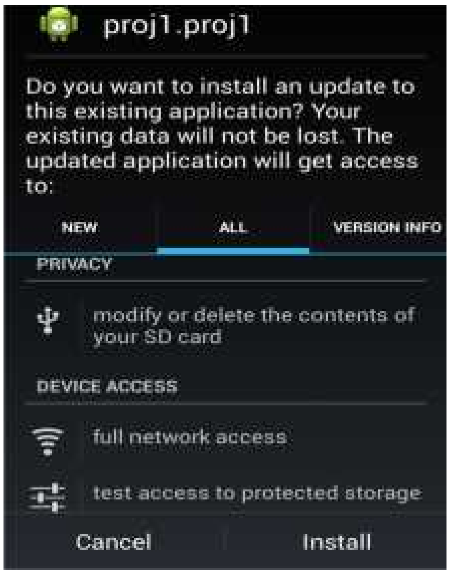
After copying the previous file to mobile, disconnect the USB cable, and go to the file manager to get the location of copied apk file as shown on Figure 5.1, open the apk file and press on "install" button to start installing as shown in Figure 5.2.

Figure 5.3 shows installation progress; finally open the installed application and start to use it as shown in Figure 5.4.



CameraCapture.apk
176.61K Jan 9, 2013 01:29 | -rw
Gallery2.apk
2.11M Aug 29, 2012 22:11 | -rw
Helloworld.Helloworld-Signed.apk
3.35M Feb 27, 2013 11:30 | -rw
Helloworld.Helloworld.apk
3.37M Feb 27, 2013 11:30 | -rw
proj1.proj1-Signed.apk
4.21M May 3, 2013 09:47 | -rw
Resistors.apk
3.09M Aug 31, 2012 00:25 | -rw
SimpleCamera.SimpleCamera-Signed.apk
136.84K Mar 15, 2011 15:39 | -rw

Figure 5.1: File manager


proj1.proj1

Do you want to install an update to this existing application? Your existing data will not be lost. The updated application will get access to:

PRIVACY

modify or delete the contents of your SD card

DEVICE ACCESS

full network access

test access to protected storage

Figure 5.2: Install confirmation

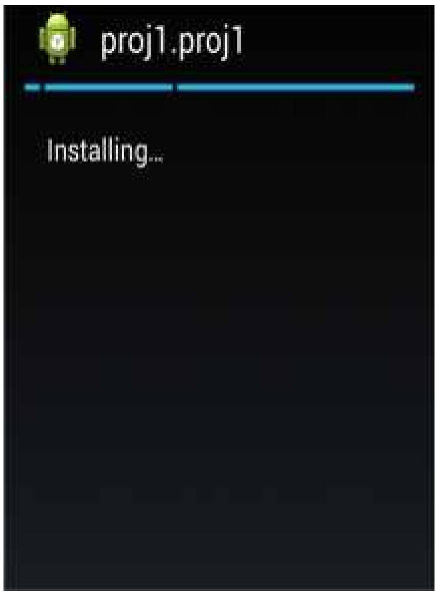


Figure 5.3: Installing application

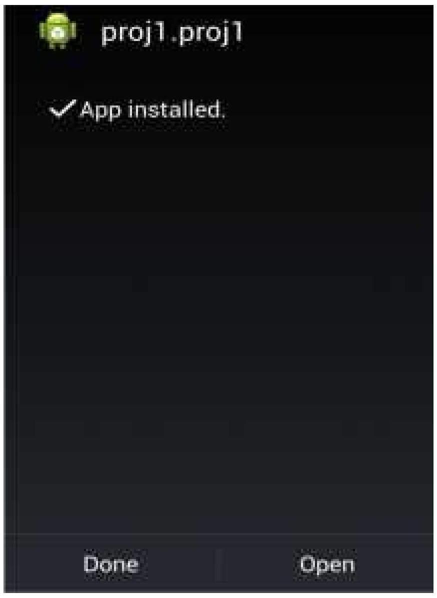


Figure 5.4: Done Installing application

5.2.2 Using Visual Studio

The second method to deploy an application is using Visual Studio environment, first of all connect the USB cable to the computer, be sure the debugging mode is checked on mobile, then go to the project in Visual Studio, right click choose “deploy” as shown in Figure 5.5. The window in Figure 5.6 shows the devices connected to computer, choose you device and start deploying, when deploying is finished the application will appear in the applications at mobile, open it and start use.

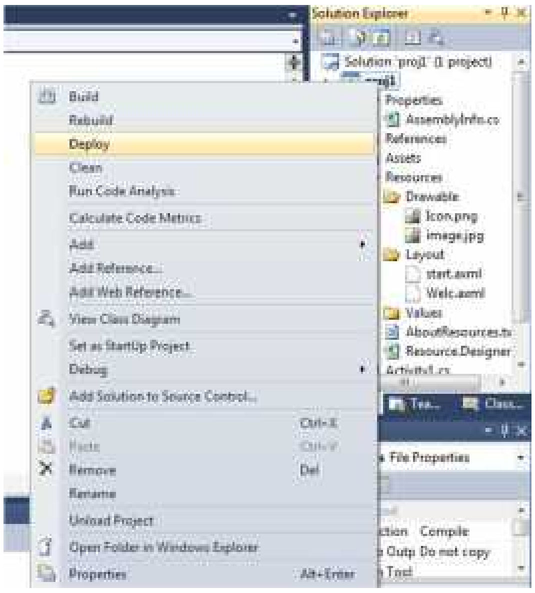


Figure 5.5: Deploy application to device

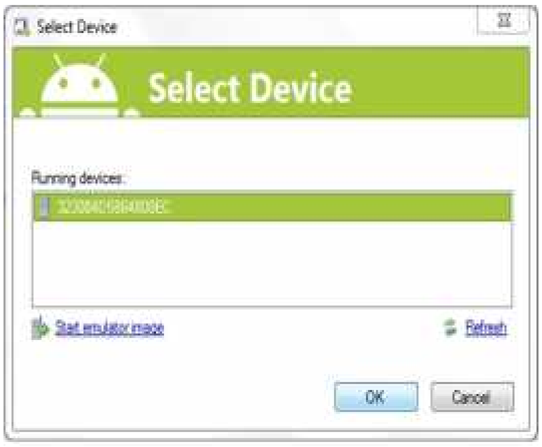


Figure 5.6: select target device

5.3 Project Testing

This section will discuss the most possible testing scenarios that may occur during system usage, three samples have been tested for each of the following states:

i. Testing with ideal image

First, the ideal image has been tested in order to ensure the efficiency of the BT/VC algorithm; theoretically, it should give 100% correct results, the following Table 5.1 shows the results that have been recorded during the testing stage.

Table 5-1: Ideal images results

Sample	Result percentage(%)
S1	100
S2	100
S3	99
Average	99.6

ii. Testing with captured image (ordinary)

The captured image has been tested without applying the suggested skew correction algorithm, and the following Table 5.2 shows the results that have been recorded during the testing stage.

Table 5-2: Ordinary images results

Sample	Result percentage(%)
S1	60
S2	55
S3	63
Average	59.3

iii. Testing enhanced captured image

The captured image has been tested with applying the suggested skew correction algorithm, and the following Table 5.3 shows the results that have been recorded during the testing stage.

Table 5-3: Enhancement images results

Sample	Result percentage(%)
S1	70
S2	60
S3	68
Average	66

iv. Testing the scanned image

The image has been scanned using a scanner and the following results have been recorded as shown in Table 5.4.

Table 5-4: Scanned image

Sample	Result percentage(%)
S1	83
S2	75
S3	77
Average	78.3

v. Testing image with sparse data

The sparse image contains small number of cells, which means that the original image was cropped so the image has one or two lines, the following results have been recorded as shown in Table 5.5 .

Table 5-5: Sparse data images

Sample	Result percentage(%)
S1	95
S2	90
S3	97
Average	94

5.4 Summary

The following Table 5.6 shows a summary for all situations that have been tested and its percentage results.

Table 5-6: Average results for all situation

Sample / State	Ideal Image	Ordinary	Skew Algorithm	Scanned	Sparse Data
Average (%)	99.6	59.3	66	78.3	94

According to the three Braille samples that have been tested in different situations using BT/VC algorithm, the ideal(artificial) image gives 99.6% correct results, and 0.4 incorrect results has been happened due to existing of some noise in the third sample. The ordinary captured sample gives 59.3% due to having a skewed angles and some noise that affects the results.

Then after applying a suggested algorithm to enhance the image and make it slat image instead of skewed image as much as possible, 66% correct results and 34% incorrect results has been happened due to have some skewed angles and noise. After that, a scanned image has been entered to the BT/VC algorithm, 78% correct results and 22% incorrect results due to having some noise and some rotation in angle.

Finally a sparse samples (image contain few lines and cells) tested by BT/VC and 94% correct results has been gained and 6% incorrect results due to have some noise.

Chapter 6 Challenges and Conclusion

6.1 Overview

This chapter describes the project achievements, the challenges and the problems that faced the team during the implementation stage with their suggested solutions, also the project conclusion that describes all things that the team concludes during the project, finally this chapter will talk about the future work.

6.2 Achievements

Most of the project goals and objectives have been achieved; this section summarizes the main objectives that achieved in this project.

- **Design algorithm to convert Braille script into another script.**
- **Use effective techniques and tools to build a mobile application.**
- **Built a *Mobile application* to convert Braille script into another script.**
- **Built a *Desktop application* to convert Braille script into another script.**
- **Design a good graphical user interface.**

6.3 Challenges

This section will describe the main challenges that faced the team during the implementation stage, and their suggested procedure to solve each challenge individually.

6.3.1 The programming environment

Choosing the suitable environment was the first challenge that the team has faced during the implementation stage, firstly the team choose Java environment to build the system, but the performance in Java was not fast comparing with C# (Mono for Android), and as you know the mobile has limited power and memory resources. Therefore, the team decided to choose the C# with Mono for Android plug-in to build the application.

6.3.2 Separating the dots from the background (thresholding)

The team spent a lot of time searching for the best thresholding technique to separate the dots from background, at the first, the team use simple technique depending on the grayscale image histogram, but it was too specific(overfitting) and give different result from one image to another. Second, the team tried to use edge detection algorithms it was a little bit better than the previous one, but it need a specific capturing conditions to give good results.

The solution was to use adaptive thresholding discussed in section 2.3.3.4, which changes the threshold dynamically over the image, and it gives a good result for 90% images.

6.3.3 Braille paper shape

The Braille paper is 11 by 11.5 inch, which is approximately square, and we all know the modern mobiles screen is rectangle, so when we capture a Braille page, we got some extra information outside the Braille paper where it is not needed and causes a noise that will affect the conversion process which lead to get incorrect translation.

The solution was to perform cropping process on the captured image that will crop just the Braille paper from the whole image, and the user is responsible to do that.

6.3.4 Out of memory exception

Dealing with images in C# environment with Mono for Android plug-in gives errors loading large images, because the mobile power and memory recourses are limited, loading a 5-mega pixel resolution image or greater gives an out of memory exception.

The solution was to capture the image in 3-mega pixel resolution that will decrease the image size, so that we can load the image without errors, in this case the high resolution is a problem and not a solution dealing with mobiles and image processing.

6.3.5 Elimination conditions

Elimination conditions is one of the primary problems in dealing with images generally, especially when dealing with sensitive images such as Braille paper images, if the image is in a bad elimination conditions it is hard to separate the dots out of background using thresholding algorithm.

The solution is a user responsibility to capture the image in good elimination conditions and to use the adaptive thresholding that can overcome this problem. otherwise, it will give an error during separating dots from the background that will leads to have an errors in the whole conversion process that depends on thresholding technique.

6.3.6 Skewed Image

The images that captured from a camera cannot be prefect slat, especially when dealing with a very sensitive image such as Braille images, some of dots may be misaligned with the others as shown in Figure 6.1 and this will leads to have incorrect results. So the rotation angle should be found to rotate the image according to this angle in order to get a slat image as much as possible.

A suggested solution for this problem is to find the sum of rows on a Braille cell, after that the image is rotated with a small angle (e.g. 0.01 degree), then the sum of rows will be calculated for the rotated image, this process will be repeated many times in order to get the distribution of pixels as shown in Figure 6.2.

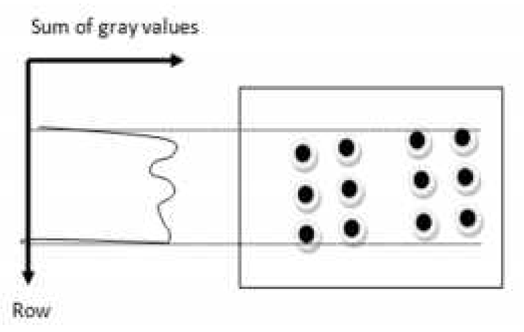


Figure 6.1: Skewed image

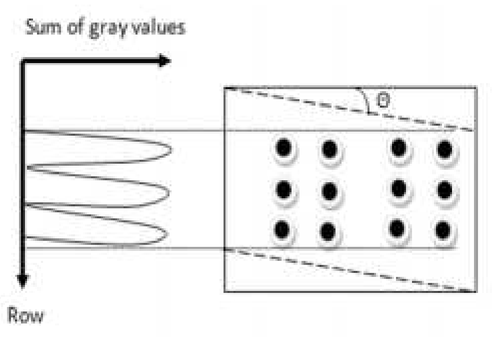


Figure 6.2: Un-Skewed image

6.4 Conclusion and Recommendations

During and after the implementation has been finished, we concludes many things from that experience; this section will discuss these collusions.

The existence of such a system to convert a Braille paper into another language will help those sighted people they are dealing with blind people in their daily life to be able to read a Braille paper without have any knowledge in Braille, because it is not easy to learn a Braille language.

We learned that working on something is totally different from talking about it, which means that we should never talk before try, and everything should be studied and prepared well before advance into a practical stage.

Working with a team is very important thing, and give a power and meaning to the project, also during work as a team, too many solution will be gotten and this help us a lot when we faced some challenges, each one has his/her own idea and the suggested solution was made from these ideas.

Dealing with images in term of image processing issue it is not an easy task, and it was harder than we thought. Which mean that the ability to have 100% results is not possible in this project.

Braille image is a sensitive image, which means it should be captured under a suitable situation in order to get good results.

It is possible to program an application for android using C# instead of JAVA and we decide to use C# because it is faster than JAVA, also it have a powerful library for image processing called EmguCV that is independent platform library.

According to the resources limitation of the mobile, a high-resolution image is a problem and not a solution, so it is enough to capture the image with 3 M-pixel and it will be clear enough to apply the BT/VC algorithm and get good results.

Adaptive thresholding technique that has been used to separate the Braille dots from the background is an effective technique and it gives a very good result for more than 90% from the images.

Morphology techniques can help to enhance the image from a noise as discuss in chapter 2.

Braille Text/Voice Converter is a good algorithm, because there is no need to go through all pixel in the image, which means it is fast algorithm, and we were unable to compare our project with the other due to non-existence of such application that convert a Braille script into another script using the mobile.

The captured image always has a skew angle (or the image has a rotated angle in 3rd axis), which means it will give incorrect results during the testing, and the suggested algorithm has enhanced (not 100%) the image from being skewed as discuss in section 6.3.6. BT/VC gives a good result for un-skewed images such as more than 70% from the whole text can be recognized.

6.5 Future Work

Future work will focus on multilingual scripts, improve the un-skewing algorithm, improve BT/VC algorithm, and have more collaborative user interface.

Appendices

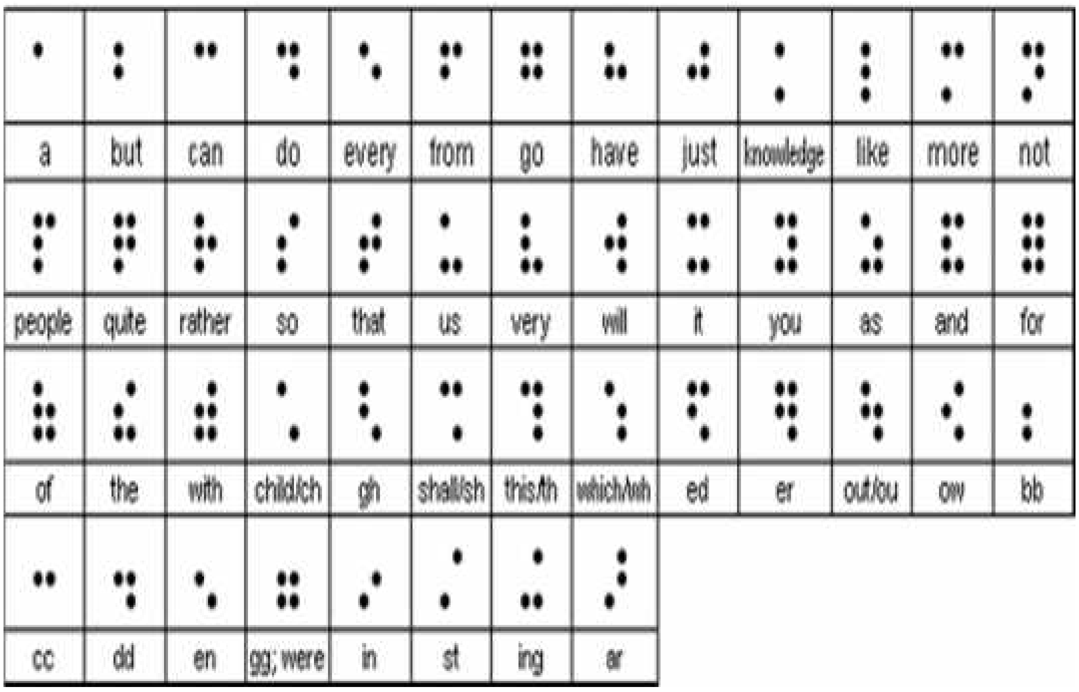
APPENDIX A

Braille Grades 2 and 3

Grade 2

In early 2000s, grade 2 Braille was introduced as a space-saving alternative to grade 1. In Braille grade 2, a cell can represent a shortened form of a word as shown in Figure A.1, making this the most popular Braille grade.

There are part-word contractions, which often stand in for common suffixes or prefixes, and whole-word contractions, in which a single cell represents an entire commonly used word. Most or all of the vowels in a word in order to shorten it. A complex system of styles, rules, and usage has been developed for this grade of Braille [9].

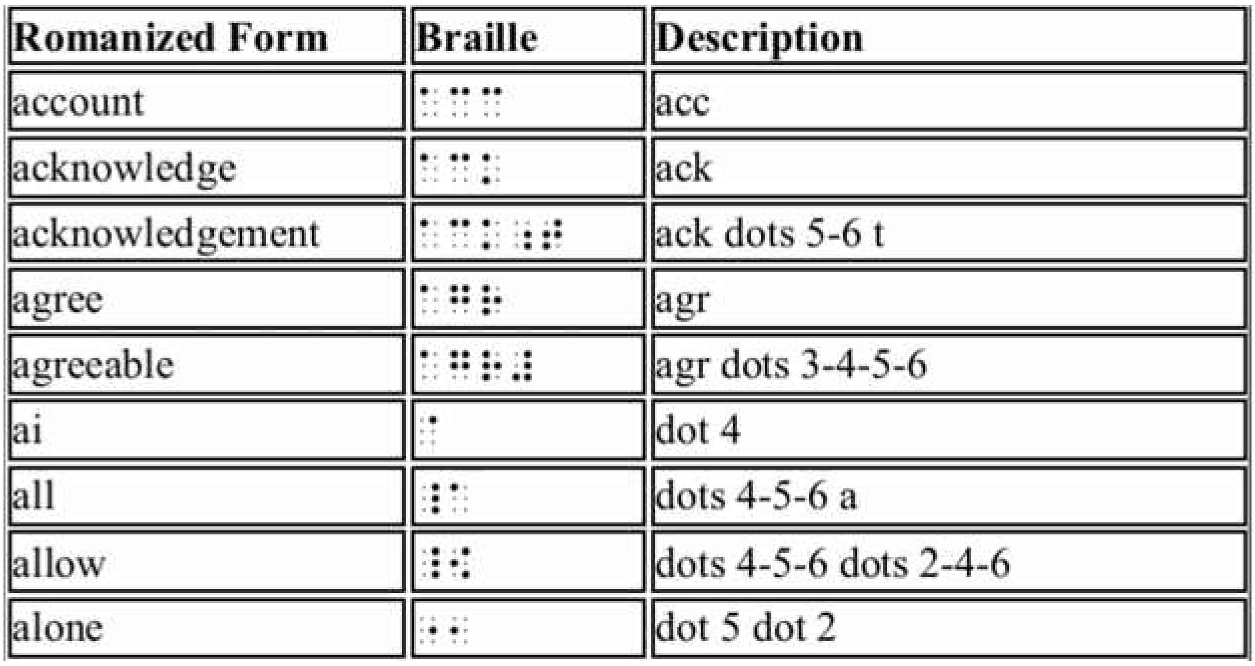


⠁	⠃	⠉	⠙	⠑	⠋	⠛	⠓	⠚	⠅	⠇	⠍	⠝
a	but	can	do	every	from	go	have	just	knowledge	like	more	not
⠏	⠟	⠗	⠎	⠞	⠥	⠧	⠺	⠭	⠽	⠵	⠯	⠿
people	quite	rather	so	that	us	very	will	it	you	as	and	for
⠷	⠮	⠾	⠡	⠣	⠩	⠹	⠱	⠫	⠻	⠳	⠪	⠆
of	the	with	child/ch	gh	shall/sh	this/th	which/wh	ed	er	out/ou	ow	bb
⠒	⠲	⠢	⠶	⠔	⠌	⠬	⠜					
cc	dd	en	gg; were	in	st	ing	ar					

Figure A.1: Grade 2 words and abbreviations

Grade 3

Grade 3 Braille is a form of Braille shorthand as shown in Figure A.2 which represents a sample of grade 3 Braille, It is similar to grade 2 Braille that it uses contractions and abbreviations to save space; however, it goes far beyond grade 2 in the share number of contractions - over 300 of them - and also uses omitted vowels and decreased spacing between words and paragraphs to save space [9].



Romanized Form	Braille	Description
account	⠁⠉⠉	acc
acknowledge	⠁⠉⠅	ack
acknowledgement	⠁⠉⠅ ⠰⠞	ack dots 5-6 t
agree	⠁⠛⠗	agr
agreeable	⠁⠛⠗⠼	agr dots 3-4-5-6
ai	⠈	dot 4
all	⠸⠁	dots 4-5-6 a
allow	⠸⠪	dots 4-5-6 dots 2-4-6
alone	⠐⠂	dot 5 dot 2

Figure A.2: Grade 3 words and shorthand

APPENDIX B

MONO for android

How to start the first project in android

Create project

In Visual Studio select “File >> New >> Project …”, choose from the left side the “Visual C#” then choose “mono for android” then choose “Android Application Project”, add its name then the visual studio will create all the necessary android project files and configurations as shown in Figure B.1.

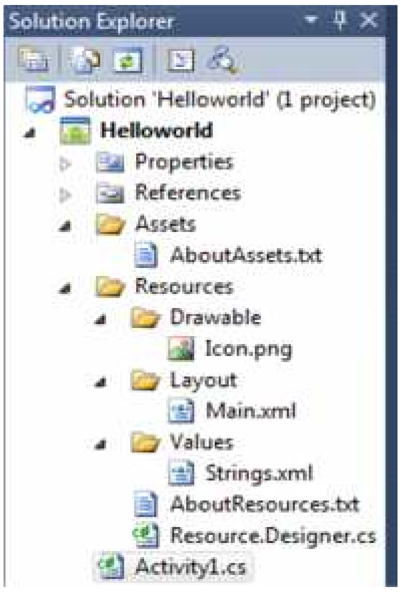


Figure B.1: Hello World folders

Writing a code

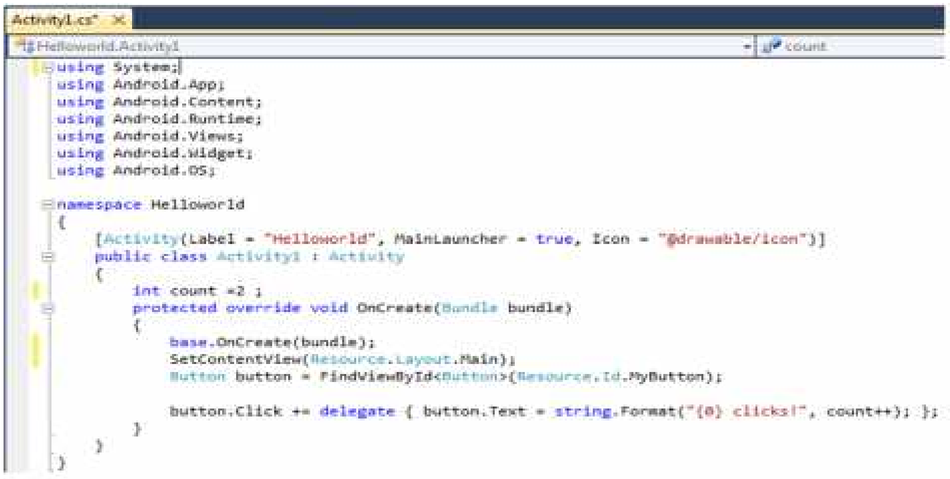
As shown in Figure B.2, the android code has the following parts:

Libraries used

Include the libraries that are used in a project and add other libraries can be imported on demand.

Class header and body:

Each class is an activity and this class has a header which include the activity name extends the Activity, also has a body which includes the methods and the code of the class.



```
using System;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;

namespace Helloworld
{
    [Activity(Label = "Helloworld", MainLauncher = true, Icon = "@drawable/icon")]
    public class Activity1 : Activity
    {
        int count =2 ;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Main);
            Button button = FindViewById<Button>(Resource.Id.MyButton);

            button.Click += delegate { button.Text = string.Format("{0} clicks!", count++); };
        }
    }
}
```

Figure B.2: Hello World Code

Deploy a project

Any android application can be run using two different methods; the first one using a real android device and the other one using emulator, which is a virtual android device.

Figure B.3 shows the run using a virtual device for the previous example “Hello World”.

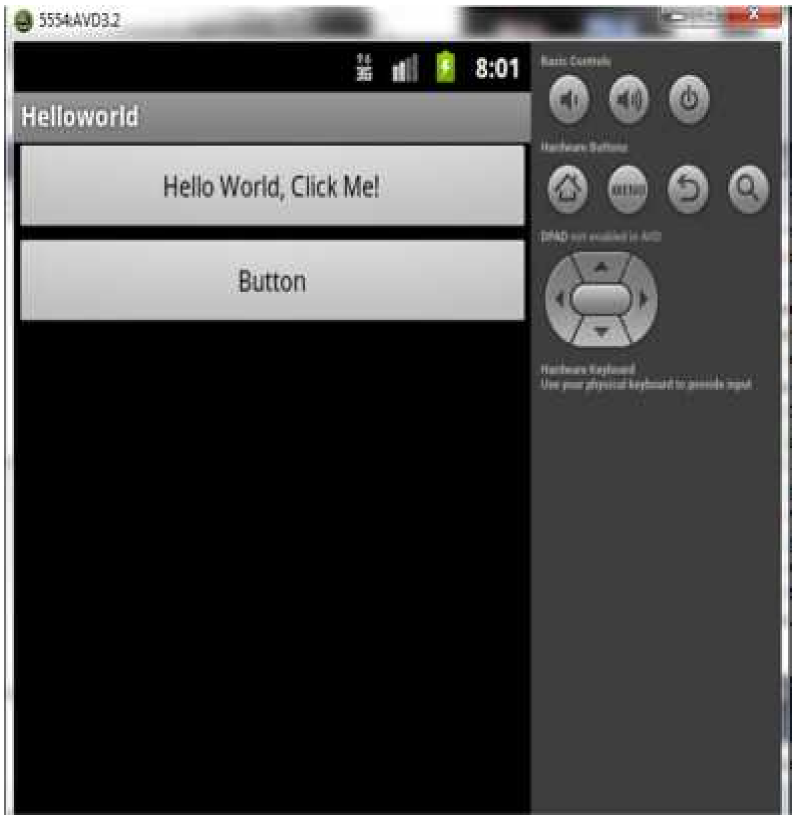


Figure B.3: Deploying application using emulator

APPENDIX C

PROJECT CODE

Activity 1:

```
using System;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;

namespace proj1
{
    [Activity(Label = "Proj1", MainLauncher = true, Icon = "@drawable/icon")]
    public class Activity1 : Activity
    {


        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.Welc);


            Button st = FindViewById<Button>(Resource.Id.button1);

            st.Click += delegate
            {
                StartActivity(typeof(Activity2));

            };

        }
    }
}
```

Activity 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Provider;
using Android.Content.PM;
using Android.Graphics;
using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Android.Graphics.Drawables;
using System.Drawing;
using System.Collections;

namespace proj1
{
    [Activity(Label = "Capture and Load")]
    public class Activity2 : Activity
    {

        Hashtable ht = new Hashtable();
        Java.IO.File _file;
        static int CAMERA_CAPTURE = 1;
        static int PIC_LOAD = 2;
        Bitmap im = null;
        public String s = "";
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // Create your application here
            pop_hash();

            SetContentView(Resource.Layout.start);


            Button Capture = FindViewById<Button>(Resource.Id.Capture);
            Button Load = FindViewById<Button>(Resource.Id.Load);
            Button Convert = FindViewById<Button>(Resource.Id.Convert);

            Convert.Enabled = false;
            Capture.Click += delegate
            {


                var intent = new Intent(MediaStore.ActionImageCapture);

                var availableActivities =
this.PackageManager.QueryIntentActivities(intent,
                    PackageInfoFlags.MatchDefaultOnly);

                if (availableActivities != null && availableActivities.Count >
0)
                {
                    var dir = new Java.IO.File(

Android.OS.Environment.GetExternalStoragePublicDirectory(
                        Android.OS.Environment.DirectoryPictures), "BTVC");

                    if (!dir.Exists())
                    {
                        dir.Mkdirs();
                    }

                    _file = new Java.IO.File(dir,
String.Format("myPhoto{0}.jpg", Guid.NewGuid()));
                    intent.PutExtra(MediaStore.ExtraOutput,
Android.Net.Uri.FromFile(_file));
                    StartActivityForResult(intent, CAMERA_CAPTURE);
                }
            };


            Load.Click += delegate
            {

                var imageIntent = new Intent();
                imageIntent.SetType("image/*");
                imageIntent.SetAction(Intent.ActionGetContent);
                //mod = 3;
                StartActivityForResult(Intent.CreateChooser(imageIntent, 
"Select photo"), PIC_LOAD);
                Convert.Enabled = true;

            };


            Convert.Click += delegate
            {
                String r=conv(im);
                Intent i = new Intent(BaseContext, typeof(Activity3));
                i.PutExtra("Text", r);
                StartActivity(i);

            };


        }


        protected override void OnActivityResult(int requestCode, Result 
resultCode, Android.Content.Intent data)
        {

            if (requestCode == CAMERA_CAPTURE)
            {

                base.OnActivityResult(requestCode, resultCode, data);
                var imageView = FindViewById<ImageView>(Resource.Id.picture);

                // make it available in the gallery
                var mediaScanIntent = new 
Intent(Intent.ActionMediaScannerScanFile);
                var contentUri = Android.Net.Uri.FromFile(_file);
                mediaScanIntent.SetData(contentUri);
                this.SendBroadcast(mediaScanIntent);
                var image = 
(Bitmap)MediaStore.Images.Media.GetBitmap(ContentResolver, contentUri);
                TextView tv = FindViewById<TextView>(Resource.Id.textView1);
                imageView.SetImageBitmap(image);
                tv.Text = image.Height.ToString() + "," + 
image.Width.ToString();


            }

            else if (requestCode == PIC_LOAD)
            {
                base.OnActivityResult(requestCode, resultCode, data);

                TextView tv = FindViewById<TextView>(Resource.Id.textView1);
                var imageView = FindViewById<ImageView>(Resource.Id.picture);
                imageView.SetImageURI(data.Data);
                String path = getRealPathFromURI(data.Data);
                Bitmap bm = BitmapFactory.DecodeFile(path);
                im = bm;
                tv.Text = bm.Height.ToString() + "," + bm.Width.ToString();
            }


        }
        public String getRealPathFromURI(Android.Net.Uri contentUri)
        {

            String[] projection = new String[] {
            Android.Provider.MediaStore.MediaColumnsConsts.Data };
            ContentResolver cr = this.ContentResolver;
            Android.Database.ICursor cursor = cr.Query(contentUri, projection,
                 null, null, null);
            if (cursor != null && cursor.Count > 0)
            {

   cursor.MoveToFirst();
                int index =

cursor.GetColumnIndex(Android.Provider.MediaStore.MediaColumnsConsts.Data);
                return cursor.GetString(index);
            }
            return null;
        }
        protected void pop_hash()
        {
            ht.Add("000001", 32);//capital sign
            ht.Add("001111", 32);//number sign
            ht.Add("000000", 32);//space
            ht.Add("100000", 97);//a
            ht.Add("110000", 98);//b
            ht.Add("100100", 99);//c
            ht.Add("100110", 100);//d
            ht.Add("100010", 101);//e
            ht.Add("110100", 102);//f
            ht.Add("110110", 103);//g
            ht.Add("110010", 104);//h
            ht.Add("010100", 105);//i
            ht.Add("010110", 106);//j
            ht.Add("101000", 107);//k
            ht.Add("111000", 108);//l
            ht.Add("101100", 109);//m
            ht.Add("101110", 110);//n
            ht.Add("101010", 111);//o
            ht.Add("111100", 112);//p
            ht.Add("111110", 113);//q
            ht.Add("111010", 114);//r
            ht.Add("011100", 115);//s
            ht.Add("011110", 116);//t
            ht.Add("101001", 117);//u
            ht.Add("111001", 118);//v
            ht.Add("010111", 119);//w
            ht.Add("101101", 120);//x
            ht.Add("101111", 121);//y
            ht.Add("101011", 122);//z
            ht.Add("000100", 122);//z
            ht.Add("010000", 44);//,
            ht.Add("011000", 59);//;
            ht.Add("001000", 39);//'
            ht.Add("010010", 58);//:
            ht.Add("001001", 45);//-
            ht.Add("010011", 46);//.
            ht.Add("011010", 33);//!
            ht.Add("011001", 63);//?
            ht.Add("001011", 34);//"
            // ht.Add("110110", 103);//
            /////////////////////////////////

        }

        public String conv(Bitmap bm)
        {
            Image<Gray, Byte> im = new Image<Gray, Byte>(bm);

            im = im.ThresholdAdaptive(new Gray(255), 
Emgu.CV.CvEnum.ADAPTIVE_THRESHOLD_TYPE.CV_ADAPTIVE_THRESH_GAUSSIAN_C, 
Emgu.CV.CvEnum.THRESH.CV_THRESH_BINARY_INV, 171, new Gray(5));
            im = im.Dilate(1);
            im = im.Erode(3);
            im = im.Dilate(2);

            String cell = "";
            for (int i = 75; i < im.Width; )
            {
                if (i + 55 > im.Width) { break; }
                else
                {
                    Rectangle r = new Rectangle(i, 75, 40, 60);

                    int cy = i + r.Width / 2;//x
                    int cx = 75 + r.Height / 2;//y
                    cell += Convert.ToChar(Convert.ToInt16(ht[cellReco(cx, cy, 
im)]));
                    i += 55;

                }
            }
            return cell;

        }



        String cellReco(int centerx, int centery, Image<Gray, Byte> newim)
        {
            String cell_eq = "";

            newim.Data[centerx, centery, 0] = 255;

            if (Dot_In_square_recognition(centerx - 24, centery - 12, newim))
                cell_eq = "1";
            else
                cell_eq = "0";
            if (Dot_In_square_recognition(centerx, centery - 12, newim))
                cell_eq += "1";
            else
                cell_eq += "0";
            if (Dot_In_square_recognition(centerx + 24, centery - 12, newim))
                cell_eq += "1";
            else
                cell_eq += "0";
            if (Dot_In_square_recognition(centerx - 24, centery + 12, newim))
                cell_eq += "1";
            else
                cell_eq += "0";
            if (Dot_In_square_recognition(centerx, centery + 12, newim))
                cell_eq += "1";
            else
                cell_eq += "0";
            if (Dot_In_square_recognition(centerx + 24, centery + 12, newim))
                cell_eq += "1";
            else
                cell_eq += "0";

            return cell_eq;
        }

        Boolean Dot_In_square_recognition(int x, int y, Image<Gray, Byte>
Dotsim)
        {
            Boolean IsDot = false;

            for (int i = x; i < x + 1; i++)
            {
                if (Dotsim.Data[i, y, 0] > 200)
                    IsDot = true;
            }
            for (int i = x; i > x - 1; i--)
            {
                if (Dotsim.Data[i, y, 0] > 200)
                    IsDot = true;
            }
            for (int j = y; j < y + 1; j++)
            {
                if (Dotsim.Data[x, j, 0] > 200)
                    IsDot = true;
            }
            for (int j = y; j > y - 1; j--)
            {
                if (Dotsim.Data[x, j, 0] > 200)
                    IsDot = true;
            }
            return IsDot;
        }

    }


}
```

Activity 3:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;

using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Speech.Tts;

namespace proj1
{
    [Activity(Label = "Result")]
    public class Activity3 : Activity, TextToSpeech.IOnInitListener
    {
        TextToSpeech mTts;
        Java.IO.File _file;
        protected override void OnCreate(Bundle bundle)
        {

            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Result);

            EditText et = FindViewById<EditText>(Resource.Id.editText1);
            String s = Intent.GetStringExtra("Text");
            et.Text = s;
            Intent checkIntent = new Intent();
            checkIntent.SetAction(TextToSpeech.Engine.ActionCheckTtsData);
            StartActivityForResult(checkIntent, 100);
            // Create your application here
            Button spk = FindViewById<Button>(Resource.Id.Speak);
            Button save = FindViewById<Button>(Resource.Id.save);
            Button ret = FindViewById<Button>(Resource.Id.ret);

            spk.Click += delegate
            {
                mTts.Speak(et.Text, QueueMode.Flush, new Dictionary<string,
string>());
            };

            ret.Click += delegate
            {
                StartActivity(typeof(Activity1));
            };

            save.Click += delegate
            {
                var dir = new Java.IO.File(

Android.OS.Environment.GetExternalStoragePublicDirectory(
                        Android.OS.Environment.DirectoryPictures), "BTVC");

                        var jfs = new

Java.IO.File(Android.OS.Environment.ExternalStorageDirectory.ToString(),
                    "test.txt");
                Toast.MakeText(BaseContext, "Saved to test.txt",
ToastLength.Long).Show();
                var jfw = new Java.IO.FileWriter(jfs);
                jfw.Write(s);
                jfw.Close();
            };
        }

        void TextToSpeech.IOnInitListener.OnInit(OperationResult status)
        {
            Console.WriteLine("Hello");
        }

        protected override void OnActivityResult(int requestCode, Result
resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);
            if (requestCode == 100)
            {
                mTts = new TextToSpeech(this, this);
            }
        }
    }
}
```

REFERENCES

[1] Paul Blenkhorn, “A system for converting Braille into print”, IEEE Transaction on rehabilitation,Vol.3, No.2, June, 1995.

[2] Saad D. Al-Shamma and Sami Fathi, “Arabic Braille Recognition and Transcription into Text and Voice”, cairo international biomedical engineering conference, No. 5, cairo , Egypt , December , 2010.

[3] Lain Murray and Andrew Pasquale, “A portable device for the translation of Braille to literary text”, university of Curtin for technology, Western Australia, 2007.

[4] Jan Mennens, Luc van Tichelen, Guido Francois and Jan J. Engelen,” Optical recognition of Braille writing using standard equipment”, IEEE Transaction on rehabilitation,Vol.2, No.4, December, 1994.

[5] Roy, Noëlle, Louis Braille 1809–1852, a French genius, Valentin Haüy Association website, retrieved 2011-02-05.

[6] How Braille began, http://www.brailler.com/braillehx.htm

[7] History of Braille, http://www.brailleworks.com/Resources/HistoryofBraille.aspx

[8] Hermida, X. F., et al, “A Braille O.C.R. for Blind People”, Proceedings of ICSPAT-96. Boston (U.S.A.). October, 1996.

[9] What is Braille, http://www.acb.org/tennessee/braille.html

[10] R. Gonzalez and R. Woods, Digital Image Processing, Addison-Wesley Publishing Company, 1992. & Susstrunk, Buckley and Swen. "Standard RGB Color Spaces”. 2005.

[11] http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm

[12] R. Gonzalez and R. Woods, Digital Image Processing, Addison-Wesley Publishing Company, 1992.

[13] ,[14] Android Developers, http://developer.android.com/

[15] EmguCV, http://www.emgu.com/wiki/index.php/Main_Page.