International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 1
ISSN: 2088-5334

A Tesseract-based Optical Character Recognition for a Text-to-Braille Code Conversion

Robert G. de Luna

Department of Electronics Engineering, De La Salle Lipa, Lipa City, 4217, Philippines
E-mail: robert.deluna@dlsl.edu.ph

***Abstract*— This study provided a platform that converts printed text documents into corresponding braille code that will trigger the palpable output of the braille cells. The system is composed of two main parts: the image scanner and the microcontroller-based braille platform. The image scanner captures the printed text document and performs a series of pre-processing algorithms where the processed image will be subjected to character recognition using Tesseract. It is open-source software for character recognition capable of recognizing text characters in different fonts and sizes. SimpleCV, also an open-source software for computer vision and a simpler version of an OpenCV, was utilized in pre-processing of images where binarization, filtering, edge detection, and character segmentation are performed. This will allow the microcontroller-based braille platform to interpret the printed characters from the generated braille code in ASCII format that will trigger the palpable output of the braille cells. The developed system was subjected to functionality and accuracy testing to assess its performance. Accuracy was based on the capability of the system to produce the right braille outputs that match the scanned line of text which are in Arial font. The testing was conducted utilizing the Arial font size of 12, 14, 16, 18, 20, 22, and 24. Results show that the system is capable of recognizing text with greater than 85 % accuracy starting at font size 18 with an average accuracy of 88.09 % and increases accordingly as the font size increases.**

***Keywords*— optical character recognition; braille; image processing; tesseract; text-to-braille.**

I. INTRODUCTION

An estimated 332,150 people in the country are bilaterally blind, while the current number of persons with bilateral low vision has already reached 2,179,733 last 2017 [1]. Effective literacy can be hindered by blindness, especially for children as they are beginning their path to learning. According to Resources for the Blind, Inc., there are more than half a million blind people and many more who are visually impaired to a lesser degree in the Philippines. Statistics also reveal that about 100 children lose their sight every week [2].

Blind children should be given the privilege to learn to read and write through specific methods like the braille. It is a tactile reading system specially developed for the blind or partially sighted persons [3]. This invention by Frenchman Louise Braille during the 19th century has made a turning point for blind literacy. Due to this innovation, it has now been adopted as the standard form of written communication for blind people.

There are international research works already been made to address this problem for the blinds. A study created a system consists of the following three modules: a portable, low-cost refreshable Body-Braille, an easy Braille writer, and a remote communication system through SMS [4]. In the other paper, the implementation of Braille to Text/Speech Converter on FPGA Spartan 3 kit was produced [5]. The input is given through braille keypad, which consists of different combinations of cells. Another study presented a comprehensive, unified braille Unicode system providing detailed mapping of 8-dot braille Unicode pattern to represent the transcribing codes as well as the math, science, and computer symbols/characters [6].

In 2017, a study produced a prototype of an affordable Braille display for the blind people to read when input is given through a computer [7]. An automatic system was used to convert computer written text to the Braille language [8]. Another study presented a novel design of a low-cost, low-power, portable and user-friendly Braille system [9]. The advantage of the system is that the designed system serves as both Braille writing and reading system, so visually impaired people can enhance their Braille writing and reading skills without the assistance of a Braille teacher. The designed system takes the input through the Braille keyboard and produces the Braille output in Braille display; the corresponding English characters are also displayed on the LCD and also in the laptop if it is connected.

There are also some studies that produced braille system with specific language application. One study proposed a method for converting Braille codes to Malayalam voice message implemented using MATLAB which can be read out too through the computer [10]. In this paper, Braille code is extracted from the input image and it is mapped to the Malayalam database and spoken out. Also, in 2017, a study presented a hardware design and implementation of a portable Bengali Braille embosser [11]. A simple microcontroller is used as a controlling unit of the embosser, an algorithm in the conversion of the text in a braille script implemented in a raspberry pi was proposed [12]. Specific for Sri Lanka, a study developed a braille converter with a text-to-speech translator to empower visually impaired people [13]. In 2017, Team Tactile, a group of students from the Massachusetts Institute of Technology, produced from one competition a tactile device that is inexpensive and portable that translates text in real-time [14].

In this paper, the researcher created an embedded system that scans printed text documents and converts the text images into a Braille code. A stand-alone program to implement image pre-processing algorithms was created using SimpleCV that runs in a mini-computer called the Raspberry Pi with Tesseract as an engine for character recognition. The Braille code is then generated in the microcontroller-based braille platform as an actuator to provide a palpable output that corresponds to recognized text for the blind to read. Thus, this application will be beneficial for the visually impaired person as it can scan and translates text-content printed documents into Braille format for them to read and understand.

II. Materials and Methods

A. Materials

The hardware components of the system are composed of two parts namely, the image scanner and the braille platform. Tables I and II show the lists of hardware equipment and software used in the creation of the system.

TABLE I
List of Hardware Components

Quantity	Equipment	Specifications
1	Webcam (Logitech)	8 Megapixel
1	Microcomputer (Raspberry Pi 2B)	Quadcore CPU 1 GB RAM
1	Micro SD Memory Card (Apacer)	16 GB Class 10
1	Microcontroller (Arduino Mega)	16 MHz Oscillator 54 digital IOs 16 analog inputs
1	Braille Cell (SC11 KGS)	16 cell modules 200 V DC
1	Numeric Keypad (A4Tech)	USB Type

The Raspberry Pi was chosen as the acting computer due to its capability to handle image processing as well as to execute stand-alone programs in open-source software [15]. These conditions were found to match the main process and the needed portability feature of the Text-to-Braille system. In this study, the image pre-processing was considered as the basic step for having better outputs in character recognition. In addition, a lighting component should be implemented in the scanner in order to compensate for the effect of ambient light as well as to maintain and ensure the desired outputs of the image processing.

TABLE II
List of Software Components

Software	Function	Manufacturer
Matlab	Image Processing	MathWorks Inc. Massachusetts, USA
Tesseract	Character Recognition	Google California, USA
SimpleCV	Computer Vision	Sight Machine Company California, USA

Logitech C525 was the camera used. It was found to be more desirable than the Raspberry Pi camera module due to its higher resolution of 8 megapixels with an autofocus feature [16]. It is evaluated to be functional because it is flexible and easy to operate. The images are immediately saved within the system. However, the camera is not able to perform multiple capturing. It has to be reset after each use because of the absence of an automatic function to close its port after data transmission.

A python script was used in image pre-processing because it contains all the pre-processing algorithms in SimpleCV. It is an open-source framework for building computer vision applications. With it, we get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, colour spaces, buffer management, eigenvalues, or matrix versus bitmap storage [17]. Although the initial plan was to create the pre-processing algorithm in OpenCV, it was found out that before the programming environment can be accessed, some libraries must first be built with OpenCV. These libraries should be compatible with the compiler to prevent errors in the compilation of codes. In this study, SimpleCV was used which is a simpler version of the OpenCV. It can be programmed in the same environment with the latter but has only limited libraries. Matlab with its image processing toolbox is also a good alternative but the raspberry pi is incapable of executing the program and can only be used as a controller for peripheral devices [18]. Tesseract engine was used in the system for character recognition. Tesseract was originally developed at Hewlett- Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some C++izing in 1998. In 2005, Tesseract was open-sourced by HP and since 2006 it is developed by Google. Tesseract is an application software that can be easily integrated or embedded in the Raspberry Pi. Tesseract is selected due to its capability of recognizing characters with different fonts and sizes [19].

The braille used in the study is the SC11 16-cell module manufactured by the KGS Corporation in Japan. An Arduino Mega as the controller was also used in the platform. It features 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button [20].

B. Methodology

1) Description of the System: The system is capable of allowing the visually impaired to access digital information in Braille; through which pins rise and retract to represent Braille dots based on the scanned words of a particular printed text. The system is composed of two necessary parts, the scanner, and the braille platform. In the scanner part, there will be an operator in charge of placing a single sheet of printed text on the capturing surface. The desired line of text needed to be captured will be set through the help of a fixed paper cover on the scanning platform. The camera is fixed at the height of 6 inches from the capturing platform and thus, the printed sheet will be the one to be adjusted in order for it to fit within the coverage area. By pressing the capture button, the camera will begin to take the image of the specific area of text depending on what is desired to be read. When the user already pressed the capture button, there will be a voice prompt saying "Capturing, please wait" through the speaker. The captured image will then be saved in JPEG (.jpg) format and is ready for image processing.

After acquiring an image, it will be automatically sent to the Raspberry Pi board which contains the stand-alone program coded in SimpleCV where the image will undergo image pre-processing before being fed into the Tesseract engine for character recognition. When the pre-processing is completed, there will be another voice prompt, "Done capturing," signaling that the Tesseract already recognized the character from the pre-processed image. The recognized characters in the form of ASCII equivalent codes are sent serially to the microcontroller for controlling the braille platform which is informed by the "Sending to braille. Please wait" voice output. The braille platform will be the tangible output for the blind which is composed of braille cells similar to a tactile display.

The controls will be dictated by the press of a button on the numeric keypad integrated on the braille platform. It must first be toggled off to load the program and then toggled on to proceed with the actual control of the system. It has a green LED indicator that when lit, signifies that it is already toggled on. However, this feature is only for the operator and not for the blind. The sequence of the output of the braille cells will be determined by the 'Back' and 'Next' buttons assigned to the modified numeric keypad. These buttons provide navigational options for the reader and should be followed by 'Send' and 'Enter' to execute. For every output in the braille, there will always be a voice prompt of "Please read the file."

The system as a whole was subjected to accuracy testing. This depends on producing the right braille outputs that match the scanned line of text in Arial font. During the testing, the proponent found out that the system gets more accurate for increasing size in the font. The testing was conducted utilizing Arial fonts 12, 14, 16, 18, 20, 22, and 24 and based on the results obtained; the system functions poorly with font size 12 as it has 0% average accuracy. The system was then found to reach the target accuracy of 85% starting at font size 18 and above. There is higher accuracy for a larger font size since the character features are more detailed on those larger fonts.

2) Process and Procedures: Figure 1 shows the block diagram of the system. After capturing an image of the printed text document, the system will employ different image pre-processing techniques that will extract the needed text out from the captured image. At the Raspberry Pi, it will undergo the processes of acquisition, binarization, noise filtering, edge detection, character segmentation, and text recognition as part of the image pre-processing stage (Pradeep, KM, & Jacob, 2014).

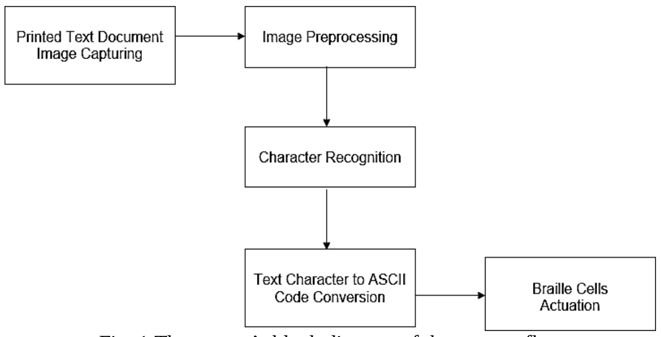

Printed Text Document Image Capturing
Image Preprocessing
Character Recognition
Text Character to ASCII Code Conversion
Braille Cells Actuation

Fig. 1 The system's block diagram of the process flow

Image acquisition serves as the first step in order to obtain the image for pre-processing. Image binarization afterward converts the colored levels of the image through (Red-Green-Blue) RGB to grayscale conversion. The thresholding process will convert the gray image into a black and white image in order to have a definite differentiation between the letter pixels and background pixels.

The binary image is then utilized for noise filtering in order to remove unwanted pixels in the image. The next stage is the edge detection where it is used to find boundaries between lines of objects and determines discontinuities in the image. The resulting image will have bounding boxes on the area covered by the text. This area will contain the individual lines of text which will be extracted. For character segmentation, it is used to analyze and provide a clear definition between letter spaces and word spaces. Each character within a word will be segmented and blank spaces in between words will be the basis for separating the words.

The segmented characters will then undergo character recognition through the Tesseract engine. The recognized character will then be subjected to the microcontroller for the conversion into a corresponding braille code in the form of ASCII. This code will be used by the controller to actuate the cells of the braille platform.

3) Testing the *Functionality and Accuracy of the System:* The functionality test requires the whole system to meet certain conditions to be operationally reliable. It will serve as a measure of whether the study met the set of parameters that were considered. The braille must actuate in such a way that the pins rise and retract depending on the input state. The microcontroller controls the pins of the refreshable braille to rise and retract. For the functionality, the pins of each cell were tested by applying high and low voltages in the form of different number systems programmed in the microcontroller. The system must convert ASCII codes to the Braille Code as an output. The corresponding ASCII code must have its corresponding Braille code output displayed by the refreshable braille cell controlled by the microcontroller.

For the test, the accuracy of the system, the actual outputs are compared to the expected outputs. The OCR must recognize different fonts. All 26 letters of the alphabet for both uppercase and lowercase, numbers 0-9, and basic punctuation marks were tested under a series of trials. Braille output should correctly represent the expected equivalent ASCII character input. ASCII code equivalent to a particular letter or number will be the input to the braille platform and the Braille output should represent the expected equivalent character.

The overall system accuracy represents an acceptable accuracy rate of 85 % above in every trial. Sample simulation was performed, which involves an actual document with different font and font size to which the braille output of the whole system will be compared. Each word output of the refreshable braille will be compared to its corresponding word printed on the document. The number of correct output words will determine the accuracy of the whole system. The font size that will give 85 % and above accuracy will be established and will be suggested to be used with the system.

III. Results and Discussion

A microcontroller-based braille platform has been successfully developed that outputs the corresponding braille character of the scanned text in a tactile display through a serial connection with the raspberry pi. Through hardware and software integration, a system was created which response to every successful serial transfer of recognized characters in ASCII format by accepting them, matches them to the equivalent braille codes stored in the microcontroller, and finally actuates the equivalent braille outputs. The braille display was first found out to have no standard preprogrammed codes for the braille characters. Thus, a binary pattern was established for actuating each dot of the braille cell with the corresponding English character in a grade1, 8-dot braille format. The binary code for the high and low pattern was then represented in hexadecimal for data compression. Consequently, the hexadecimal equivalent will be assigned to every serial ASCII input received. This process was used for the ASCII-to-Braille conversion in the microcontroller.

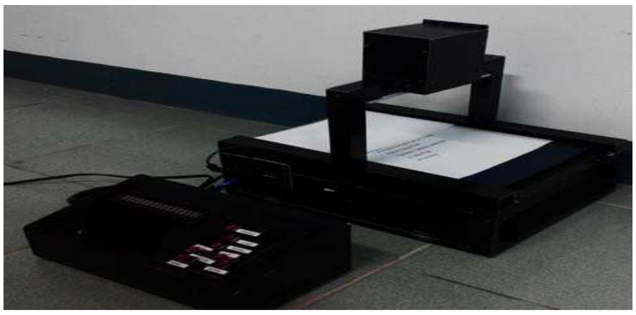
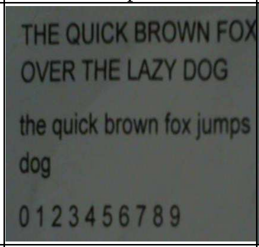
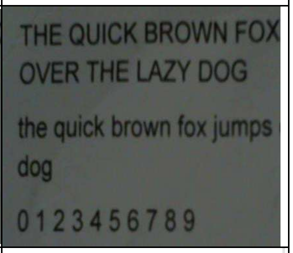
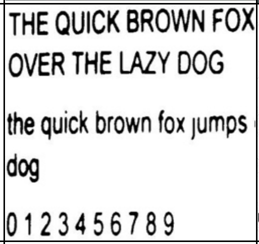
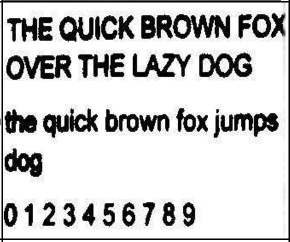
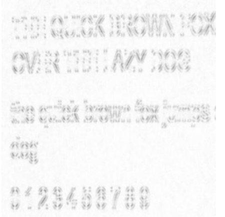
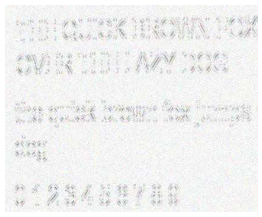
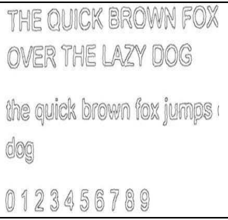
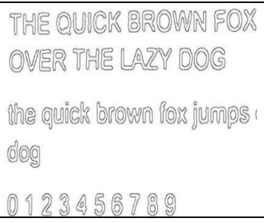
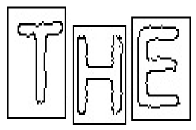
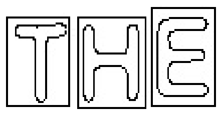


Fig. 2 The system set-up used in the study showing the braille platform and the image capturing device.

Figure 2 shows the set-up of the system. For the acquisition and pre-processing of the image of the printed text document, a programming environment suitable and compatible with the Raspberry Pi was established. In this case, SimpleCV in Python language was utilized in coding the image pre-processing steps of image acquisition, binarization, noise filtering, edge detection, and character recognition.

An 8 megapixels webcam with autofocus feature was used in the study which provided clear capture of the sample document. Once captured, the image will be automatically loaded by the program and will undergo a series of transformations. The image results were previewed through the remote desktop connection of the Raspberry Pi with a Laptop. In testing the functionality of the image preprocessing algorithms, five different sample images each containing portions of text captured by the camera, were used as in Table III.

TABLE III
Sample Results of Image Processing Using SimpleCV and Matlab

Algorithm	SimpleCV	Matlab
Image Acquisition		
Binarization		
Filtering		
Edge Detection		
Character Segmentation (first word)		

The tested images are all uniform in Arial font with a font size of varying font sizes like 12, 14, 16, 18, and 20. These images were called individually in the program and the results were displayed after the execution in order to track the changes for every algorithm applied. In addition, a comparison of outputs was made between SimpleCV and Matlab. This is done for verification if both software produces the same or close outputs with each other. Shown in Table III is the sample results of the image preprocessing technique.

Based on the results, it can be observed that both SimpleCV and Matlab software produced the same outputs with minimal variations. In the outputs of binarization, it can be observed that SimpleCV selects a threshold value closer to white. Thus, it has a cleaner output. For the other algorithms, there were almost no considerable variations with their respective outputs.

The braille platform is part of the system hardware that is responsible for providing the palpable reading output for the blind. For the development of the hardware part, the proponent first made use of the schematic diagram as well as the datasheet of the SC11 braille cells manufactured by KGS Corporation (Japan) in order to accomplish the appropriate wirings with the Arduino microcontroller. The braille platform was also integrated with a numeric keypad which was modified to have specific buttons that facilitate the main controls for the whole system such as the program execution, image capturing and braille display navigation. The braille platform is then subjected to both functionality and accuracy tests. Shown in Figure 3 is the actual set-up of the braille platform used for the testing.

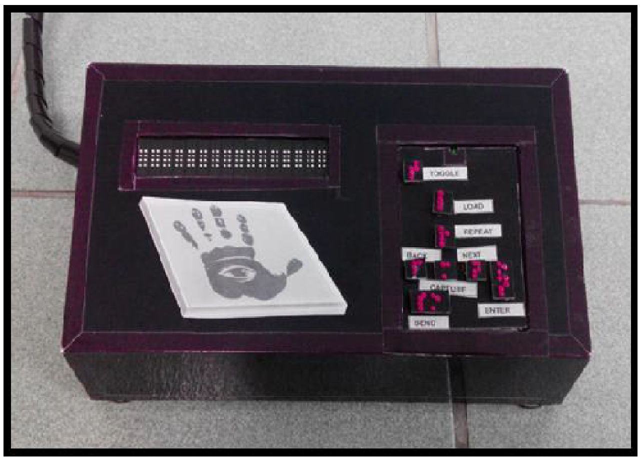
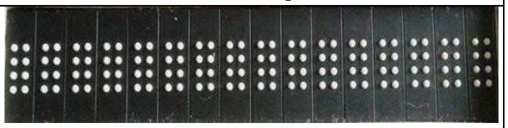


Fig. 3 Microcontroller-based braille platform

The braille platform is first evaluated in terms of its functionality in order to ensure a working output. For the testing proper, the braille cells must rise and retract depending on the input state of high and low. This is done through the Arduino serial monitor. Table IV shows the actual response of the braille cells for the two input states. It can be observed from the table that assigning a value of '1' for each bit actuates all braille cells to rise. On the other hand, assigning '0' for each bit causes the entire braille cells to retract. Based on the results, each braille dot for every cell is considered functional.

The functionality test for the braille characters is carried out to determine the response of the braille cells to serial signals which include data representation in binary, decimal, hexadecimal and ASCII. During the initial phases of testing, it was found out that there were no standard pre-programmed commands for actuating every braille character in the tactile display. Thus, the proponent first derived the specific order of each braille dot starting from the most significant bit up to the least significant bit which results in an 8-bit binary pattern. This binary pattern reflects the high and low input states to the braille cells and the pattern varies for every character. The proponent was able to accomplish this through the help of the datasheet of the SC11 braille cells. The binary values can then be compressed into decimal and hexadecimal signals to be easily carried out at the Arduino serial monitor for actuating the corresponding braille character. On the other hand, the ASCII equivalent of each character is being sent by the Raspberry Pi and will be handed to the Arduino microcontroller. Using a specialized code for ASCII-to-Braille script conversion, the response of the braille cells for every ASCII signal can be determined.

TABLE IV
FUNCTIONALITY TEST OF THE REFRESHABLE BRAILLE CELLS RESPONSE TO THE BINARY, DECIMAL, AND HEXADECIMAL CODE

State	Braille Output
All pins are high	 BIN: 11111111 DEC: 255 HEX: FF
All pins low	BIN: 00000000 DEC: 0 HEX: 0

For multiple character braille cells, the 8-bit pattern is simply being sent serially. Thus, in the case of the 16-cell braille display used in the system, the user must send a total of 128 bits (16 cell x 8 bits/cell) to the Arduino microcontroller. Figure 4 presents how an 8-bit binary pattern is derived from a sample 8-dot braille character.

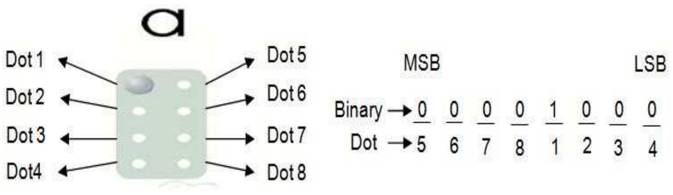
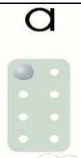
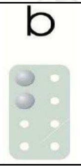
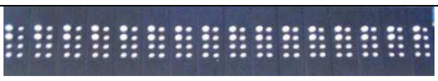
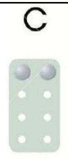
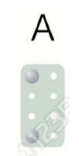
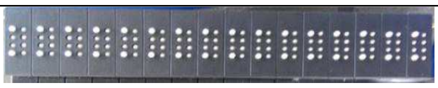
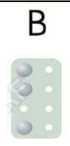
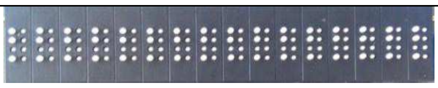
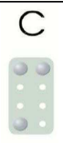
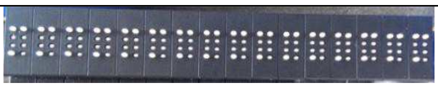


Fig. 4. An 8-bit binary pattern of a sample braille character "a" showing the MSB, LSB, binary code, and the dot configuration

One crucial finding in this test is that the serial port of the Arduino Mega automatically resets right after each data transmission. The disadvantage of this is that when the next sets of data arrive exactly at the same time during which the Arduino is reset, the incoming data will not register into the Arduino and will simply be discarded. Consequently, the system will not respond and will cause an error in the program. To resolve this issue, the proponent put a three-second delay at the start and end of each transmission process to ensure that the Arduino is not reset and already initializes when the next set of data is received. Tables V and VI show some sample results of braille actuation based on what character to display. It is divided into two parts due to some differences in the response of the braille cells on the ASCII signals of some characters.

TABLE V
RESULT OF THE ACCURACY TEST FOR ALPHANUMERIC CHARACTERS AND OTHER SPECIAL CHARACTERS

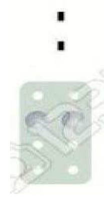
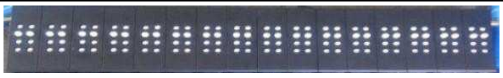
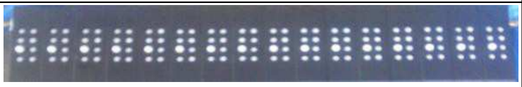
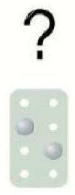
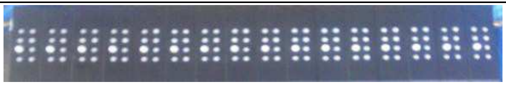
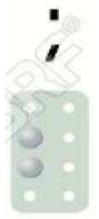
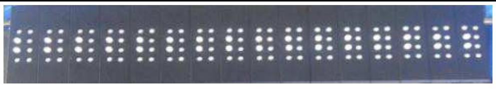
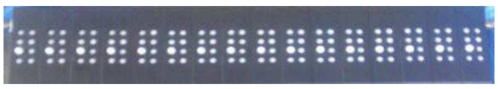
Braille Character	Braille Output
	BIN: 00001000 DEC: 08 HEX: 0 ASCII: 61
	 BIN: 00001100 DEC: 12 HEX: 0C ASCII: 62
	BIN: 10001000 DEC: 136 HEX: 88 ASCII: 63
	 BIN: 00001001 DEC: 09 HEX: 09 ASCII: 41
	 BIN: 00001101 DEC: 13 HEX: 0D ASCII: 42
	 BIN: 10001001 DEC: 137 HEX: 89 ASCII: 43

Based on the results, it can be seen that the braille cells accurately respond to either binary, decimal, hexadecimal, and ASCII representation of the following characters. Thus, it can represent the alphanumeric characters and special characters (! ( ), - * .) at 100% accuracy based on the code, it will receive.

Shown in Table VI are the special characters which accurately respond to binary, decimal, and hexadecimal signals but not to ASCII. Based on the results, it can be observed that when the Raspberry Pi sends characters such as a colon (' : '), question mark (' ? '), and semicolon (' ; '), the braille platform displays the representation of a period (. '') instead. This is due to the limitation of the Arduino platform to represent the following characters in a data type called 'char.'

In the ASCII-to-Braille script conversion code, all signals received are stored in a data type called 'char'. Since those characters cannot be stored under 'char' data type, there is no way for these characters to be given their equivalent binary, decimal, or hexadecimal representation. The best that the system can do is to assign these characters to another character which is recognizable as 'char.' Since period ('.') is the most common character that can be seen in a document, the proponent prefers to assign these three characters with a 'period' equivalent, which therefore gives the results presented.

TABLE VI
RESULTS OF THE ACCURACY TEST FOR SPECIAL BRAILLE CHARACTERS WITH ERROR IN ASCII

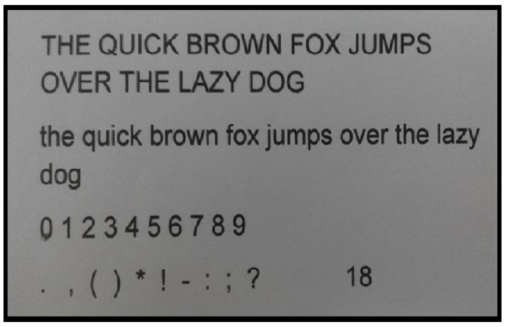
Braille Character	Braille Output
	 BIN: 01000100 DEC: 68 HEX: 44
	 ASCII: 3A
	BIN: 00100100 DEC: 36 HEX: 24
	 ASCII: 3F
	 BIN: 00000110 DEC: 06 HEX: 06
	 ASCII: 3B

Aside from the braille cells, the numeric keypad is also tested in terms of the functionality of the modified keys, which serve a specific purpose. The Numlock button is used to toggle the numeric keypad on/off. The Load or '8' button is used to execute the stand-alone program. The enter button is the key to be pressed to execute other commands. The Send or '0' button signals the serial transfer to the microcontroller. Back and Next is for display navigation while the Capture button serves as the camera trigger.

After each of the important components of the system has been tested and evaluated, the functionality and accuracy of the overall project will be evaluated as a whole. To be able to do so, the system is tested under different sets of conditions and parameters designed to represent the conditions that it will come across during actual operations. The results of each test are tabulated and analyzed to be able to assess if the objectives have been met. The following legend was used:

Legend: ☑ Correct Match ☐ Incorrect Match

Figure 5 shows the sample test image used to determine the accuracy of the system in determining the standard Arial font.


THE QUICK BROWN FOX JUMPS
OVER THE LAZY DOG

the quick brown fox jumps over the lazy
dog

0 1 2 3 4 5 6 7 8 9

. , ( ) * ! - : ; ?   18

Fig. 5 Test image results for different testing conditions

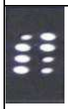
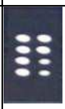
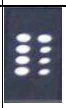
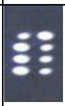
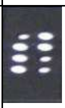
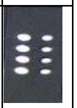
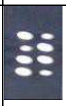
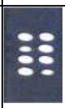
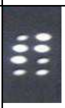
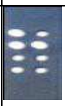
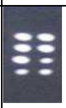
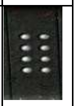
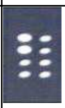
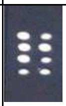
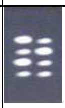
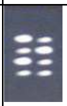
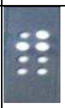
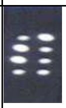
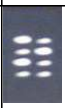
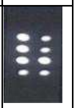
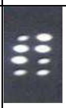
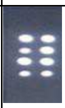
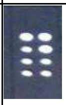
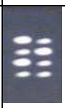
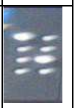
The contents of the image are selected to contain all the characters which are expected to be recognized by the system. The overall accuracy of the system for each trial is determined using the formula:

% Accuracy = (Number of Times Character is Correctly Recognized / Total Number of Characters) x 100

where the total number of characters under each trial is fixed at 108 characters.

In the table of results, the corresponding box for a character is checked when the character is being recognized correctly while it is left as it is when not correctly recognized. This is to facilitate better visualization as to the accuracy of the system for each trial. The system is tested under the standard Arial font containing all the programmed recognizable characters using different font sizes ranging from 12 to 24 with an increment of 2 for every increase in font size with each testing for every font size done 30 times starting from the actual placing of the document at the image scanner. This is done to determine the font size to which the system will function accurately, in this case, to determine the font size to which the system will start to have an accuracy of 85% and above. Table VII show sample tabulation of the result.

TABLE VII
OUTPUT SAMPLE OF THE PLOTTING FROM BRAILLE TESTED USING ARIAL 14

Trials	T	H	E		Q	U	I	C	K		B	R	O	W	N		F	O
																		
1	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	
Trials	X		J	U	M	P	S		O	V	E	R		T	H	E		L
																		
1	✓			✓				✓		✓	✓		✓		✓	✓	✓	
Trials	A	Z	Y		D	O	G		t	h	e		q	u	i	c	k	
																		
1	✓	✓		✓		✓		✓		✓	✓	✓			✓			✓
Trials	b	r	o	w	n		f	o	x		j	u	m	p	s		o	v
																		
1	✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓
Trials	e	r		t	h	e		l	a	z	y		d	o	g		0	1
																		

The target accuracy of 85 % is used because no OCR program has ever been able to read 100% correctly. As an industry-standard, 85% is regarded as good performance. Actually, there is no perfect optical character recognition engine. Modern OCR systems handled only about 85% accurate data. During the testing, the proponent found out that the system gets more accurate for increasing size in the font. The testing was conducted utilizing Arial fonts 12, 14, 16, 18, 20, 22, and 24 and based on the results obtained, the system functions poorly with font size 12 as it has 0% average accuracy. The system was then found to reach the target accuracy of 85% starting at font size 18 and above. There is higher accuracy for a larger font size since the character features are more detailed on those larger fonts.

The system was initially tested starting with font size 12. However, there were no outputs from the braille due to its incapability of recognizing characters at Arial font size 12. Shown in Table VIII is the summary of the accuracy results.

TABLE VIII
SUMMARY OF ACCURACY RESULTS IN PERCENTAGES VIS-A-VIS FONT SIZES.

Trial	**Font Size**						
	12	**14**	**16**	**18**	**20**	**22**	**24**
1	0	63.89	86.11	88.89	90.74	97.22	94.44
2	0	40.74	61.11	88.89	95.37	97.22	97.22
3	0	45.37	68.52	90.74	92.59	96.30	97.22
4	0	37.04	69.44	85.19	89.81	97.22	96.30
5	0	48.15	72.22	85.19	87.96	94.44	92.60
6	0	33.33	48.15	87.04	91.67	95.37	97.22
7	0	49.07	59.26	87.04	96.30	96.30	97.22
8	0	42.59	64.81	91.67	88.89	97.22	94.44
9	0	30.56	49.07	88.89	86.11	96.30	95.37
10	0	53.70	81.48	82.41	96.30	96.30	93.52
11	0	30.56	56.48	87.96	91.67	95.37	96.30
12	0	61.11	51.85	85.19	93.52	96.30	96.30
13	0	40.74	50.00	92.59	88.89	96.30	97.22
14	0	43.52	51.85	86.11	95.37	95.37	94.44
15	0	49.07	77.78	82.40	90.74	94.44	96.30
16	0	39.81	46.11	81.48	92.59	95.37	96.30
17	0	34.26	59.26	89.81	90.74	96.30	96.30
18	0	44.44	57.41	87.04	93.52	97.22	96.30
19	0	47.22	54.63	85.19	90.74	95.37	97.22
20	0	47.22	74.07	89.81	92.59	95.37	93.52
21	0	35.19	34.26	85.19	88.89	94.44	96.30
22	0	37.96	57.41	88.89	90.74	93.52	97.22
23	0	35.19	55.56	89.81	93.52	95.37	97.22
24	0	35.19	52.78	88.89	88.89	95.37	96.30
25	0	51.85	76.85	91.67	85.19	95.37	93.52
26	0	34.26	54.63	86.11	91.67	95.37	97.22
27	0	48.15	63.89	92.59	93.52	96.30	96.30
28	0	47.22	67.59	89.81	91.67	96.30	95.37
29	0	50.93	69.44	92.59	94.44	95.37	97.22
30	0	66.11	78.70	93.52	94.44	97.22	97.22
AVE	**0**	**44.15**	**61.69**	**88.09**	**91.64**	**95.86**	**96.00**

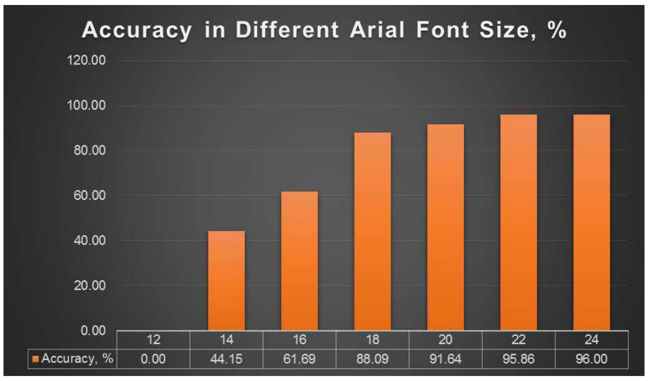


Fig. 6 Accuracy of the system in different Arial font sizes.

Table VIII and Fig. 6 shows that the system has greater capability in recognizing Arial characters being tested starting at a font size of 18 where it registered 88.09 %. As expected, a further increase in the font size of the samples will provide a significant increase in the system's accuracy of detection.

IV. CONCLUSIONS

The general objective of this study is to develop a braille system that provides the blind alternative access to reading materials through optical character recognition that converts printed text into their equivalent braille script. This study found that a microcontroller-based braille platform has been successfully developed generating outputs that correspond to the braille character of the scanned text in a tactile display through a serial connection with the Raspberry Pi. The whole system displays 0% average accuracy when tested with Arial font of font size 12 but exhibits higher than the target 85% accuracy when used with Arial font of font size 18 and above. Tesseract engine an effective tool for character recognition of different fonts and sizes. It is open-source application software that can be easily integrated or embedded in the Raspberry Pi.

ACKNOWLEDGMENT

The author would like to acknowledge the Office of Research and Publication (ORP) of De La Salle Lipa (DLSL) for the funding assistance in this research. The author also recognized the help of the members from the Electronics Engineering Department under the College of Information technology and Engineering.

REFERENCES

[1] M. Jaymalin, "Over 2 million Pinoys blind, sight-impaired," The Philippine Star, 6 August 2017. [Online]. Available: https://www.philstar.com/headlines/2017/08/06/1726085/over-2-million-pinoys-blind-sight-impaired. [Accessed 2018].

[2] S. S. d. Guzman, "The 'blind' side," The Philippine Star, 4 July 2011. [Online]. Available: https://www.philstar.com/opinion/2011/07/04/702093/blind-side. [Accessed 2018].

[3] C. Simpson, "Rules of Unified English Braille," 2013. [Online]. Available: http://iceb.org/ueb.html. [Accessed 2018].

[4] R. Sarkar, S. Das and D. Rudrapal, "A low cost microelectromechanical Braille for blind people to communicate with blind or deaf-blind people through SMS subsystem," in 3rd IEEE International Advance Computing Conference (IACC)., Ghaziabad, 2013.

[5] P. Rajarapollu, S. Kodolikar, D. Laghate and A. Khavale, "FPGA Based Braille to Text & Speech for Blind Persons," International Journal of Scientific & Engineering Research, vol. 4, no. 4, pp. 348-353, 2013.

[6] A. K. Garg, "The unified braille Unicode system: Presenting an ideal unified system around 8-dot Braille Unicode for the braille users world-over," in 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, 2016.

[7] M. E. Adnan, N. M. Dastagir, J. Jabin, A. M. Chowdhury and M. R. Islam, "A cost effective electronic braille for visually impaired individuals," in 2017 IEEE Region 10 Humanitarian Technology Conference, 2017, Dhaka.

[8] A. Moise, G. Bucur and C. Popescu, "Automatic System for Text to Braille Conversion," in 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Targoviste, 2017.

[9] S. Sultana, A. Rahman, F. H. Chowdhury and H. U. Zaman, "A novel Braille pad with dual text-to-Braille and Braille-to-text capabilities with an integrated LCD display," in 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, 2017.

[10] B. K. Rajan and V. Anjitha, "Braille code conversion to voice in malayalam," in 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2017.

[11] M. A. Rahman, K. Dhar and M. A. Ullah, "Hardware design and implementation of Bengali Braille embosser," in 2017 4th International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, 2017.

[12] A. Raghunandan and A. MR, "The Methods Used in Text to Braille Conversion and Vice Versa," International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 3, 2017.

[13] P. De Silva and N. Wedasinghe, "Braille Converter and Text-To-Speech Translator for Visually Impaired People in Sri Lanka," American Journal of Mobile Systems, Applications and Services, vol. 3, no. 1, pp. 1-9, 2017.

[14] M. W. Haas, "This Device Translates Text to Braille in Real Time," Smithsonian, 2017. [Online]. Available: https://www.smithsonianmag.com/innovation/device-translates-text-braille-real-time-180963171/.

[15] J. Marot and S. Bourennane, "Raspberry Pi for image processing education," in 25th European Signal Processing Conference (EUSIPCO), Greece, 2017.

[16] "Logitech," 2018. [Online]. Available: https://www.logitech.com/en-us/video/webcams. [Accessed 2018].

[17] "Computer Vision platform using Python," SimpleCV, 2018. [Online]. Available: http://www.simplecv.org.

[18] "Getting Started with MATLAB Support Package for Raspberry Pi Hardware," Mathworks, 2018. [Online]. Available: https://au.mathworks.com/help/supportpkg/raspberrypiio/examples/getting-started-with-matlab-support-package-for-raspberry-pi-hardware.html.

[19] "Tesseract OCR," Tesseract, 2018. [Online]. Available: https://github.com/tesseract-ocr/tesseract.

[20] "Mega 2560 R3 Board based on Arduino," MakerLab Electronics, 2018. [Online]. Available: https://www.makerlab-electronics.com/product/arduino-mega-2560-r3/.