



University of **HUDDERSFIELD**

University of Huddersfield Repository

Arif, Shahab

Electronic Braille Document Reader

Original Citation

Arif, Shahab (2013) Electronic Braille Document Reader. Masters thesis, University of Huddersfield.

This version is available at <https://eprints.hud.ac.uk/id/eprint/18089/>

The University Repository is a digital collection of the research output of the University, available on Open Access. Copyright and Moral Rights for the items on this site are retained by the individual author and/or other copyright owners. Users may access full items free of charge; copies of full text items generally can be reproduced, displayed or performed and given to third parties in any format or medium for personal research or study, educational or not-for-profit purposes without prior permission or charge, provided:

- The authors, title and full bibliographic details is credited in any copy;
- A hyperlink and/or URL is included for the original metadata page; and
- The content is not changed in any way.

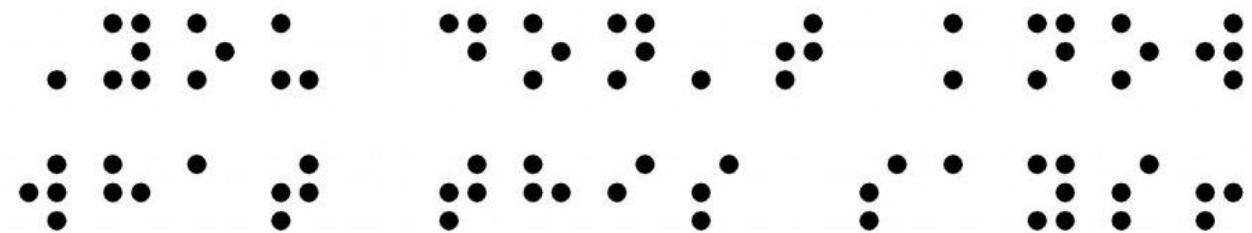
For more information, including our policy and submission procedure, please contact the Repository Team at: E.mailbox@hud.ac.uk.

<http://eprints.hud.ac.uk/>

Electronic Braille Document Reader

Shahab Arif

Supervisor: Dr. Violeta Holmes



Abstract

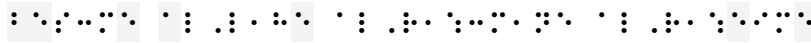
An investigation was conducted into developing a portable Braille device which would allow visually impaired individuals to read electronic documents by actuating Braille text on a finger. Braille books tend to be bulky in size due to the minimum size requirements for each Braille cell. E-books can be read in Braille using refreshable Braille displays connected to a computer. However, the refreshable Braille displays are expensive, bulky and are not portable. These factors restrict blind and visually impaired individuals from accessing much of the literature which isn't available in Braille.

The proposed device overcomes the problem of carrying bulky Braille books by allowing multiple e-books to be saved in a portable memory device. By convert text from Latin characters into Braille patterns, it will give the blind access to books which were never published in Braille. The single Braille cell design reduces the bulk of the device allowing it to be portable and reducing the cost. An additional benefit of the device is that it can be integrated into a glove and worn thus giving the user freedom to carry on with other tasks while reading.

A prototype was developed to prove Braille could be read by actuating Braille characters on a finger. The device read text from an SD card, translated it into Braille characters and actuated the Braille pattern. Blind volunteers proficient in Braille reading were able to decipher the Braille text actuated on the finger after some practice.

The investigation confirmed the feasibility of the Electronic Braille document reader built around a microcontroller system translating text into Braille. It also proved the theory that Braille could be read from a single Braille cell by the patterns actuating on the finger instead of the finger sliding across an already formed Braille pattern. A portable Electronic Braille Document Reader promises provide substantial benefits to blind and visually impaired individuals, and overcome the limitations of Braille books.

Acknowledgements



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

In the name of Allah, Most Gracious, Most Merciful.

All praise be to Allah the most gracious most merciful for all the blessings bestowed upon me and providing me with the knowledge, ability, patience and opportunity to complete my M.Sc. I am extremely grateful for the luxury the almighty has blessed me with and pray he gives me the strength and ability to use all I have in wealth and knowledge for good and attain his pleasure.

I would like to thank my supervisor Dr. Violeta Holmes for all the support and assistance over the years. Without her enthusiasm and encouragement I may not have achieved all that I have. She is the perfect example of what a supervisor should be like and it has been a pleasure working with such a nice person.

I would also like to thank my mum Samina and dad Mohammed Arif for everything they have done for me over the years. Their love, support, encouragement and teachings have enabled me to get this far and become the man I am. They deserve all the credit for the work I have done until now and any work I undertake in the future. Thank you for putting up with all the tantrums, late nights and everything else. My siblings also deserve gratitude as they have always been there for support and guidance. My sisters deserve a special thank you for supporting me financially through the hard times and being there at times of need.

I would like to thank my friend Safwan Dingmar for all the support and help he's provided over the years. He has always been there to bounce ideas off when required and provide constructive criticism when I've gone off the wrong path. He's been there as a friend and a brother through the years at university and God willing will be for many years to come. A big thank you to Haseeb Kayani and the rest of my wonderful friends who have provided me with abundance happiness, joy and laughter.

Lastly I would like to thank the Islamic society of university of Huddersfield for catering for my spiritual needs and allowing me to gain religious knowledge along with my M.Sc. The ISOC committee has been a great support and organizing events with them through the year provided a perfect way to take my mind off my studies and come back with a fresh mind.

I pray Allah blesses each and every one who helped, supported or contributed with all the happiness and success in this life and the hereafter.

Table of Contents

List of Figures.....	vi
List of Tables.....	viii
1.0 Introduction	1
1.1 Aims:	1
1.2 Objectives:.....	1
1.3 Scope	2
1.4 Background.....	3
1.4.1 Braille	3
1.4.2 Electronic Braille.....	3
1.5 Consumer survey	4
1.6 Dissertation Outline.....	5
2.0 Literature Review	6
2.1 Braille	6
2.2 Types of Braille	7
2.2.1 Six Dot Braille.....	7
2.2.2 Eight Dot Braille.....	7
2.2.3 Grade 1	8
2.2.4 Grade 2	8
2.3 Braille in Different Languages	9
2.4 Braille Technology	10
2.4.1 Refreshable Braille Displays	10
2.4.2 Braille Embossers/Note takers.....	10
2.4.3 Speech Synthesizers	10
2.4.4 Dictaphones	10
2.5 Overview of technologies suitable for EBDR actuators.....	11
2.5.1 Bi-stable Linear Moving Magnet Actuators	11
2.5.2 Linear Piezo motors.....	12
2.5.3 Piezoelectric Ceramic Bimorph Actuators	13
2.5.4 Flex Motors	14
2.5.5 Shape Memory Alloy (SMA) / Nitinol Wire.....	15
2.5.6 FET Polymeric Actuators	16
2.5.7 Bucky-Gel Actuator.....	17
3.0 System Design.....	18
3.1 Software Design	18
3.1.1 Braille Translation	20

3.1.2 SD Interface	22
3.2 Hardware Design	25
3.2.1 Actuators.....	25
3.2.2 Drive Electronics	30
4.0 System Test	32
4.1 Text Processing	32
4.2 Actuators.....	33
4.3 EBDR	33
5.0 Results	34
5.1 Text Processing	34
5.2 Actuators.....	35
5.3 EBDR	36
6.0 Discussion.....	37
6.1 Text Processing	37
6.2 Actuators.....	38
6.3 EBDR	39
7.0 Conclusion.....	40
8.0 Further Work	41
9.0 References	42
10 Appendices	44

List of Figures

Figure 1 Braille Character Z.....	7
Figure 2 8-dot Braille Character Z	7
Figure 3 Quote in Grade 1 Braille [8].....	8
Figure 4 Quote in Grade 2 Braille [8].....	8
Figure 5 Example of Arabic Braille	9
Figure 6 Example of Chinese Braille	9
Figure 8 Diagram explaining the Internals and Functions of a Squiggle Motor [21].....	12
Figure 7 Squiggle Motor	12
Figure 9 Bimorph Actuator [13].....	13
Figure 10 Flexy Motor.....	14
Figure 11 2 mm Micro motor	14
Figure 12 Shape Metal Alloy - Nitinol Wire	15
Figure 13 Right Angle Pull/ Simple Lever Examples [27].....	15
Figure 14 Cross-Sectional illustration of Actuator showing FET, Polymeric Actuator with a semisphere on the tip.....	16
Figure 15 Sheet type Braille Display.....	16
Figure 16 Operation of Bucky-gel Actuator [30]	17
Figure 17 Braille Display using Bucky-gel [29].....	17
Figure 18 EBDR flow diagram.....	18
Figure 19 Picture of Arduino Mega 2560.....	18
Figure 20 Flow Code Diagram	19
Figure 21 Braille Pattern	19
Figure 22 Braille patterns	20
Figure 23 List of characters	20
Figure 24 Input text	20
Figure 25 Setup outputs.....	21
Figure 26 Main loop	21
Figure 27 Adafruit microSD shield [32].....	22
Figure 28 SD Declarations	22
Figure 29 SD Setup	23
Figure 30 Initialize SD Card Function	23
Figure 31 Read File from SD Card Function	24
Figure 32 Nitinol in Series Configuration	25
Figure 33 Lever Configuration	26
Figure 34 V Configuration	26
Figure 35 (A) Relaxed Nitinol (B) Activated Nitinol.....	26
Figure 36 Wide Angle Configuration.....	27
Figure 37 Amount of actuation in WAV configuration.....	27
Figure 38 Braille Pin Construction.....	27
Figure 39 Nitinol resting in screw slot	28
Figure 40 Fully assembled actuator.....	28
Figure 41 Actuator relaxed vs activated	28
Figure 42 Top actuator plate.....	29
Figure 43 Bottom actuator plate	29
Figure 44 Actuators assembled together	29
Figure 45 Transistor Connections.....	30
Figure 46 Drive Circuit with Indication LED's	31

Figure 47 Electronic Braille Document Reader Prototype	31
Figure 48 LED Braille module	32
Figure 49 PEACE be upon you! 321 in Braille	32
Figure 50 Fully Assembled EBDR.....	33
Figure 51 Serial Monitor output – Successful initialization.....	34
Figure 52 Braille Output on LED Braille cell	34
Figure 53 Serial Monitor output - Failed initialization.....	34
Figure 54 Actuator Test.....	35
Figure 55 Actuated Braille Text.	36
Figure 56 Actuation difference.....	36
Figure 57 Braille on a Conveyor	41

List of Tables

Table 1 Characteristics of Bistable Linear Magnetic Actuators..... 11

Table 2 Linear Piezo Specifications 12

Table 3 Piezo Ceramic Specifications 13

Table 4 Flex Motors Specifications..... 14

Table 5 Nitinol Specifications 15

Table 6 FET Polymeric Actuator Specifications 16

Table 7 Bucky-gel Actuator Specifications 17

Table 8 Braille Code to Binary Format 19

1.0 Introduction

Blind and visually impaired individuals use a matrix of dots called Braille to read. Currently only a limited number of books get translated in to Braille which requires a human to read and type the entire book in Braille. The books which are translated are bulky due to the minimum size requirement of a Braille cell. The challenge the blind people face is having to source the desired books in Braille and then not being able to carry more than a few because of the large size.

Refreshable Braille displays exist which have a line of Braille cells and can display text from a computer when connected to one. These allow the blind to be able to use computers and access the vast catalogue of literature online. The drawback of these is the phenomenal costs and size of one.

There is a clear need for a device which can overcome these obstacles for the blind. This project looks into the feasibility of designing a device which would allow a blind user to access any digital text document by converting it to Braille and displaying it in a manner suitable for them to read.

1.1 Aims:

An Electronic device capable of reading text from external memory, and translating it from Latin characters to a Braille pattern which can be felt by a user.

1.2 Objectives:

- Design a text processing module capable of:
 - Reading text from external memory
 - Translating into Braille pattern
 - Determine the type of Braille to use
 - Determine the grade of Braille translation
 - Actuating the pattern on a Braille cell
- Design a refreshable Braille cell capable of actuating the Braille pattern
 - Determine which actuators are capable
- Build a prototype to test the feasibility of proposed device

1.3 Scope

The Electronic Braille Document Reader (EBDR) will allow the user to read any text in digital form. Users can download any text into the devices memory; whether it's newspapers or E-books. The EBDR has an advantage over conventional Braille books as it allows the user to read the book using a finger through a glove, while still being able to carry on with other tasks such as walking, housework, etc. Following are the tasks taken to accomplish the aims.

1. Research into which Braille system is best suited for the device.
2. Investigate what is the best way to read Braille.
3. Research into Patents – investigate if there is a need for a patent. Apply for a patent if needed.
4. Talk to a blind person to get design ideas.
5. Research the best way to translate text to Braille.
6. Examine different microcontrollers and figure out the one best suited for the job.
7. Explore ways to move the pins and choose the best.
8. Program microcontroller to read text from external memory and translate to Braille pattern.
9. Design a refreshable Braille cell to allow user to feel the Braille pattern.
10. Program microcontroller to move the pins to represent the text in Braille.
11. Finish prototype.
12. Test prototype with the blind and gather feedback.
13. Improve the prototype according to the feedback.
14. Test prototype with improvements.
15. Finalise the design.

1.4 Background

1.4.1 Braille

In 1821 a blind Frenchman Louis Braille invented the Braille writing system to help blind people read and write. Braille consists of a cell of six raised dots arranged in two columns of three dots, which the user can read by feeling using a finger. The Braille system is used worldwide by blind people and has also been translated in other languages. [1]

Over time Braille has been adapted to be used with different subject matter e.g. mathematics and music. As six-dot Braille was adapted to be used in other mediums, it became evident the 64 combinations (with some combinations too similar to be used) were the restricting factor of Braille. Eventually the six-dot Braille was extended to eight dots with a 4 high and 2 wide cell. The eight-dot Braille gives a total of $256(2^8)$ different combinations. All the 256 combinations of the eight-dot Braille are encoded in Unicode, while the six-dot Braille is usually stored in ASCII Braille. The advantage of the eight-dot Braille over the six-dot in written text is that a single cell can differentiate between case, numbers and symbols. [2][3]

1.4.2 Electronic Braille

Many everyday items have been significantly improved with the emergence of electronics. But these have been insignificant for the blind. Technology has made a contribution to the blind in the form of refreshable Braille display, which connect to a computer and allow a blind person to read what's displayed on the screen. But these haven't leaped forward much since their invention. The main downfall of the refreshable Braille Display is the cost of each unit and the bulky size. Currently the National Institute of Standards and Technology in the US is working on a rotating-wheel Braille display which allows the user to continually read while keeping the finger stationary. The rotating wheel design should cut the cost and the size down but it will still cost as much as a computer. [4]

Braille books are very bulky and are also hard to read without a stable place to rest the book. Only a fraction of the books published in text are also published in Braille. For the sighted the problem of lugging around many books was overcome by the invention of the E-book readers. The same can be done for the blind with the Electronic Braille Document Reader. A blind user can potentially carry around hundreds of books in a glove. An additional benefit of the Electronic Braille Document Reader is that it can provide the blind access to the thousands books which are never published in Braille, by translating regular e-books into Braille.

1.5 Consumer survey

A meeting was arranged at the Society for the Blind of Dewsbury Batley and District with a blind individual called Janet who was skilled in reading Braille to discuss the systems feasibility of allowing identification of Braille patterns using one Braille cell actuated on a finger. The opportunity was also utilized to discuss other aspects which could potentially affect the final outcome.

Currently available Braille technology was discussed to ascertain whether the Electronic Braille Document Reader would benefit the blind in anyway. The outcome suggested that the EBDR would be adding to the market in a unique way by giving the user freedom, which none of the current products offered. The Braille user, Janet explained the current electronic Braille systems were constrained to only displaying computers screens and at a cost of as much as a high end personal computer, it's not a worthwhile investment for many.

Another point Janet made was that Braille books tend to be big and always require a flat stable surface to rest while reading by sliding fingers across the page to feel the character patterns. The EBDR benefits blind by allowing many books to be saved in a relatively compact device which does not require any input from the user to work. The ease of use also encourages the use of Braille reading with the young. There are currently many voice synthesizers which can read text out and dictaphones which can be used for recording notes. But they have drawbacks of their own, which were explained by the former British home secretary David Blunkett in an article written for the BBC on the 200th anniversary of the invention of Braille. Mr Blunkett said that Braille gave him the ability to make his notes in the House of Commons in privacy. Braille also allowed him to be able to read notes while still being able to focus on the speeches. [5]

The meeting with Janet also gave a better understanding of how Braille is read. Each Braille dot has to be a minimum size as too small would be very hard to feel by an average user. Braille can be written in two forms, Grade I (which is a one to one representation of each letter in Braille) and Grade II (which is written in short form). Currently 100% accurate text translation to grade II Braille can only be done by a human and not a machine due to the different meaning of patterns depending on the context. Janet explained all users of Braille can read grade I Braille, so the Electronic Braille Reader would not need to use grade II Braille but would be a beneficial addition.

Finally while concluding the meeting, when asked if the device would be a useful addition to the devices available to the blind, Janet answered yes it would be as it would make reading books easy and give access to more books.

The consumer survey confirmed unequivocally that there is a need for a device such as EBDR and encouraged us to continue our research. The results of which are detailed in this dissertation.

1.6 Dissertation Outline

The dissertation is set out in eight main chapters followed by references and appendices.

Chapter 1 Introduction identifies the aims and objectives of the research while also discussing the background and the motives for the project.

Chapter 2 Literature Review presents an analysis of the Braille system and other technologies available to the blind. Also presents a review of the technologies available for the development of the EBDR prototype.

Chapter 3 System Design details the design process involved in the design of the EBDR prototype.

Chapter 4 System Test describes the steps taken to test the prototype.

Chapter 5 Results Presents the results of the tests carried out on the prototype.

Chapter 6 Discussion discusses the outcome of the test results. Also describes the problems encountered and solutions used to overcome the problems.

Chapter 7 Conclusion presents the outcomes of the research.

Chapter 8 Further Work highlights areas of further investigation and outlines recommendations on development of the prototype.

Chapter 9 References provides references to literature used in the research.

Chapter 10 Appendix includes information that supports the research work but is not included in the research. The full code for the EBDR can be found in the appendix.

2.0 Literature Review

2.1 Braille

Tactile forms of reading for the blind have existed since ancient times. As blind people touch and feel objects around them to picture their surroundings their finger tactile receptors get exceptionally well developed. This allows them to be able to feel details sighted people barely notice. Over time blind people have developed ways in which they are able to read using the sense of touch to varied success.[6]

Sighted people have also contributed to the effort by developing tactile systems which allow the blind to read text. Much of the effort was in the form of raised letter in large sizes so the tactile shape of the letter could be felt. These were very slow and difficult methods as they required the reader to trace around the individual character.

Charles Barbier de la Serre who served in the French army as a captain came up with “night writing” a method of writing using dots and dashes which could be read by touch, in response to Napoleon’s demand for a code which would allow for silent communication in the darkness of the battlefield. Unfortunately for Barbier the code proved to be too complicated for the soldiers to decipher with touch alone and was rejected. Barbier later demonstrated his writing system at the Royal Institute for Blind children in Paris where a young Louis Braille studied.[7]

Louis who lost his eyesight due to an accident at the age of 3 was a bright student experimenting with different ways of reading. Louis recognized the advantages the night writing code had over the current reading system available but he also saw the drawbacks and limitations. With Barbier’s code it was impossible for the reader to move swiftly from one character to another due to its large size and it was also limited to 36 basic sounds to make a word instead of the alphabet. Louis decided to develop the system further to provide a viable solution. By the age of 15 in 1824 Louis had improved the system enough to represent the entire alphabet and numbers in a six dot cell the size of a fingertip.[1]

The reading system proved very popular with the fellow students who along with Louis taught the rest how to read and also started translating books in to the new writing system. This new reading and writing system was not so popular with the sighted teachers at the school who would have to learn it and believed that it made the blind too independent. The staff at the school tried to ban the reading system which was referred to by everyone as Braille, but it proved to be too popular and resilient. They finally gave up and embraced it in 1843 by which time Louise Braille had become the first Blind professor at the school. Because Louis Braille’s reading system was easy to read and write it built a place in the hearts of its blind users. Due to its popularity, France adopted Braille as its official communications system for the blind people in 1854, two years after Louis Braille died of tuberculosis. The rest of the world slowly followed and today Braille is used all over the world and has been adapted to most languages.[1][7]

2.2 Types of Braille

2.2.1 Six Dot Braille

There are many variation of Braille in existence which could be used for the Electronic Braille Document Reader (EBDR). The most commonly used Braille system is the original six-dot Braille, which consists of a cell of six raised dots arranged in two columns of three dots. The dot positions are numbered top to bottom 1 to 3 on the first column (left), and 4 to 6 on the second column (right). Any Braille character can be described using the positions, e.g. The letter 'S': can be described as 2-3-4. The six-dot Braille has a total of 63 combinations, but some of the combinations feel too similar to be used e.g. ⠠ and ⠡ so are omitted. The punctuations are represented by their own set of patterns. But numbers use the same patterns as the alphabets 'a' to 'j'. They are recognized by the context they are in and the symbol placed before it e.g. before a number a Braille pattern 3-4-5-6 ⠠ is placed.[2][8]

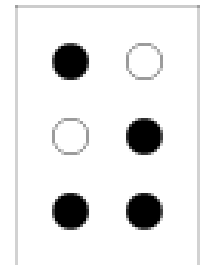


Figure 1 Braille Character Z

2.2.2 Eight Dot Braille

As the restriction of the six-dot Braille became evident it was extended to eight-dot Braille which gave 256 combinations. The eight-dot Braille has two extra dots at the bottom of the cell; each eight-dot cell consists of two columns of four dots. The two extra dots positions are numbered 6(left) and 7(right). The extra combinations allow all special characters to have a unique pattern. The main advantage of the eight-dot Braille is that all details of the character can be represented in a single cell e.g. case, number or punctuation[3]. The eight dot Braille is also popular in technical areas such as mathematics and sciences. It has also gained popularity in refreshable Braille displays as the extra two dots can represent extra information such as cursor position and various text attributes.[9]

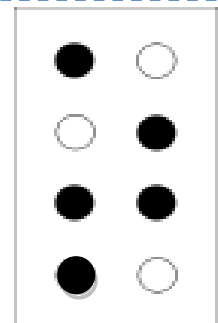


Figure 2 8-dot Braille Character Z

2.2.3 Grade 1

Grade 1 Braille, which is sometimes also called uncontracted Braille, is the exact substitution of each letters to its corresponding Braille patterns of the alphabet. It is usually used for teaching beginners and labelling because it takes more space and slow to read. Figure 3 shows a quote written in grade 1 Braille.

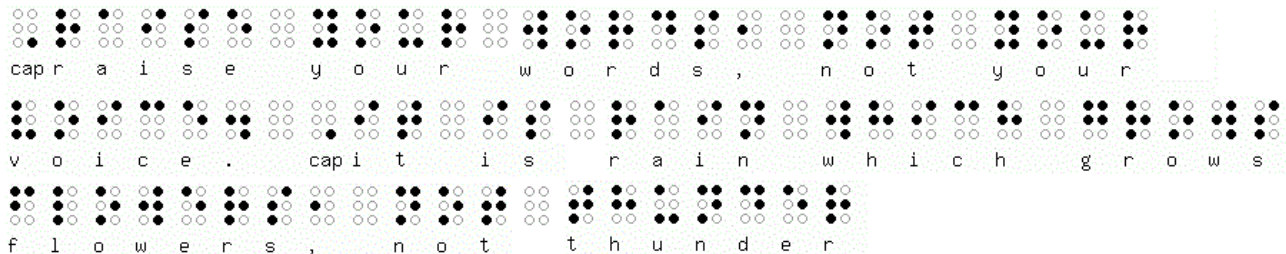


Figure 3 Quote in Grade 1 Braille [8]

2.2.4 Grade 2

Grade 2 Braille, also known as contracted Braille is when words are written in shorthand. The grade 2 Braille uses the same Braille characters as the grade 1 but with some extra combinations for commonly used words (e.g. and ⠠⠠⠠, for ⠠⠠⠠ and the ⠠⠠⠠) and common sounds (e.g. ch ⠠⠠⠠, ed ⠠⠠⠠ and ow ⠠⠠⠠). Another way the grade 2 Braille differs from grade 1 is when writing; many words can be shortened to just a few characters e.g. Braille can be written as Brl ⠠⠠⠠. Some things in grade 2 Braille can mean different things depending on the context therefore this type of Braille is used by experienced Braille users. Most publications use the grade 2 Braille because it's quicker to read and write and also takes up less space. The picture in Figure 4 shows a quote written in grade 2 Braille.[10][11] The quote was translated using online Braille converters, which can do the job but aren't 100% accurate as only humans are able to understand the context and are able to apply the rules accordingly.

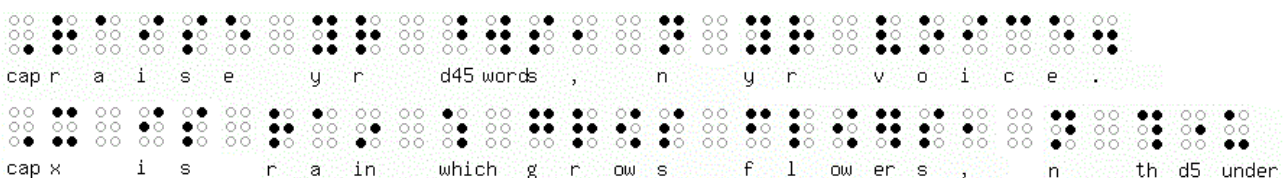


Figure 4 Quote in Grade 2 Braille [8]

2.3 Braille in Different Languages

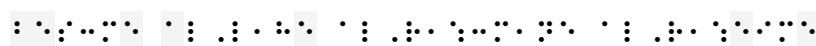
The popularity of Braille code to represent text for blind and visually impaired has encouraged others to use the Braille code to represent scripts used by other languages like Arabic and Chinese. Many languages which use the Latin alphabet like Turkish and Spanish found a relatively simple way of adapting the Braille code and including a few more patterns for letters like ş and ğ in Turkish which doesn't exist in the English or French alphabet.

Other languages with completely different scripts like Arabic, Chinese and Hindi had a harder time trying to come up with a standard. At the time of India's independence there were eleven Braille scripts in use in different parts of the country for the many different languages spoken in India. The government urged UNESCO to help come up with a unified Braille code for the country. They understood the different languages in India, Pakistan and Sri Lanka were based on phonetics with different scripts representing the sounds produced by different constants and vowels.[12][13]

A standard system was devised called the Bharati Braille. It assigns cells to characters based on the phonetic sounds. Similar sounds to the English characters are given the same Braille pattern. This also allows Indian languages to be transliterated to English and then encoded into Grade 1 Braille. The Bharati Braille uses the same patterns for all the languages, which causes a problem in multilingual texts that it's hard to spot the language change. The difference can only be spotted from the context. But the advantage it has is that multilingual user's don't have to learn new codes for all the languages. The success of the Bharati Braille caused Bangladesh, Nepal, and Sri Lanka to also adopt it as a standard for Blind communication.[13][14].

The Arabic Braille is a bit more complicated though letter assignments generally correspond to English, Greek and Russian Braille. Arabic has its rules governing the pronunciation of words and are expressed using symbols surrounding letters. Even with the complexity, a standard Braille format in Arabic exists and the Holy Book Qur'an has also been made available in Braille for blind Muslims[15]. To understand the Arabic Braille one must first have thorough knowledge of the Arabic language so further information on Arabic Braille can be found online[16][14]. Chinese and similar scripts like Japanese are also complicated due to the way they are written. Chinese for instance does not have an alphabet as such but uses characters which are based on drawings of the real object. There are over 100,000 characters in the Chinese writing system [17]. So Chinese Braille is written using the sound of the language rather than the characters. Each syllable uses three Braille cells; one for the initial, one for the final and one for the tone [18]. The UNESCO report on world Braille usage explains the adaptation and standardization of Braille in the world [14]. Following are some examples of Braille in other languages.


Arabic Braille



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ (bismi-llāhi r-raḥmāni r-raḥīm)
In the name of Allah, Most Gracious, Most Merciful.

Figure 5 Example of Arabic Braille

Chinese Braille



國語點字記號 (guóyǔdiǎnzìjìhào)
Mandarin Braille notation

Figure 6 Example of Chinese Braille

2.4 Braille Technology

Through history as technology progressed people adapted it to aid the disabled. The new age brought electronics to the world in the shape of computers, phones, TV's and much more. Much of the technology was developed with just the sighted people in mind. As the technologies evolved and became more popular people began thinking of ways to extend this technological advances for the blind. Following are some examples of how technological advances; specifically in electronics has helped the blind.

2.4.1 Refreshable Braille Displays

In the 90s personal computers became very popular making their way into many households. As much of the computers use is based on visual interaction, they weren't accessible to the blind. Refreshable Braille Displays were developed to give the blind access to computers. A refreshable Braille display substitutes a computer screen with a device with usually two lines of refreshable Braille cells which actuate to form the text outputted from the computer. Piezo actuators are used to actuate the Braille dots in most displays available today. They also include some navigation keys and many have a Braille keyboard included.

These displays are a great benefit to the blind and visually impaired as they provide a way to access the vast amount of literature available through the World Wide Web. By using Braille displays blind and visually impaired individuals are able to work in a society where few businesses operate without computers. Unfortunately these displays are very big, bulky and expensive. The piezo actuators used to actuate the dots tend to be big which also prevent the displays from having more than two lines to display. Currently each cell on the displays roughly costs \$100; this means that an average sized Braille display with 30-40 cells costs a few thousand dollars. The cost of these devices is a major limiting factor and prevents people from buying them. There is currently a lot of on-going research into improving these displays and bringing the cost down.[19][20]

2.4.2 Braille Embossers/Note takers

Braille embossers are like typewriters for Braille. A user feeds embossing paper in one end and uses the six keys; one for each dot of the cell and one button to move over to the next cell to emboss Braille patterns onto the paper.

Note takers allow the user to take notes in electronic format. Note takers have a Braille keyboard and a line of refreshable Braille cells usually of about 8 – 20 cells, which allow for the text to be read back.

2.4.3 Speech Synthesizers

Speech Synthesizers read text out loud. These are used very commonly by blind to read as listening to an audio is much faster than reading Braille. Speech synthesizers are commonly used in conjunction with refreshable Braille displays by the blind when operating a computer. Many books come in audio book formats which have been recorded by a human reading the book. Speech synthesizer's use artificially produced human speech to read the text, usually in monotone and lack the human touch. The monotone and artificial nature of the voice makes it hard to understand at times and often not very pleasing to the ear, though advances in speech synthesizers are eliminating those issues.

2.4.4 Dictaphones

A Dictaphone is a device which can record speech for later playback. They are used by many professionals to take notes where writing down on paper or typing isn't suitable at the time. They are also used by blind people to take notes as an alternative to Braille note taking which can be cumbersome at times. Audio files can be stored for listening to later or to type up when suitable. As useful as they are to the blind and visually impaired community, they can't replace Braille completely. When listening to playback it isn't as easy to jump from one desired point to another as it is on Braille embossed paper. Another drawback being it's not always suitable to talk into a Dictaphone when in a meeting.

2.5 Overview of technologies suitable for EBDR actuators

Instead of the user scanning the Braille dots on a page, the Electronic Braille Reader autonomously changes the dot pattern on the user's finger. To achieve this, a finger piece with Braille keys which the user can read from by feeling is required. There are many different technologies available which could in theory be adequate for actuating the Braille patterns. Following are a number of possible solutions for the EBDR actuators.

2.5.1 Bi-stable Linear Moving Magnet Actuators

The moving magnet actuator is simply a permanent magnet in-between two electromagnets. When an electromagnet is switched on it either attracts or repels the permanent magnet depending of the polarity. Though currently the actuators are only available in bigger packages than needed, it is conceptually possible to manufacture moving magnet actuators in smaller sizes which may be more suitable for Braille keys. [21]

The downside of using magnetics for actuators which need to be packed tightly with neighbouring actuator for the other dots is the magnetic interference between the actuators. The magnetic field from the switching on of the electromagnet in one of the actuator may induce actuation in the adjacent actuator.

Operating voltage	± 2 to 6V
Dimensions	Diameter 5 x Height 6 mm
Availability	NO

Table 1 Characteristics of Bistable Linear Magnetic Actuators

2.5.2 Linear Piezo motors

The piezoelectric motors can be manufactured to be very small, way smaller than conventional electromagnetic motors. These motors are also much more efficient and precisely controllable. The piezo motors consist of piezoelectric actuators attached to a nut which has a threaded screw in it. The motors work by causing the piezo elements to vibrate ultrasonically to create resonant orbital vibrations in a nut. The linear motion is in-turn created by the minute vibrations which drive a threaded screw forwards and backwards. The motor runs by providing the piezoelectric actuators with a set of phase-shifted waveforms that match the resonant frequency of the motor. Following is a diagram explaining the internals and the workings of the squiggle piezoelectric motor. [22]



Figure 7 Squiggle Motor

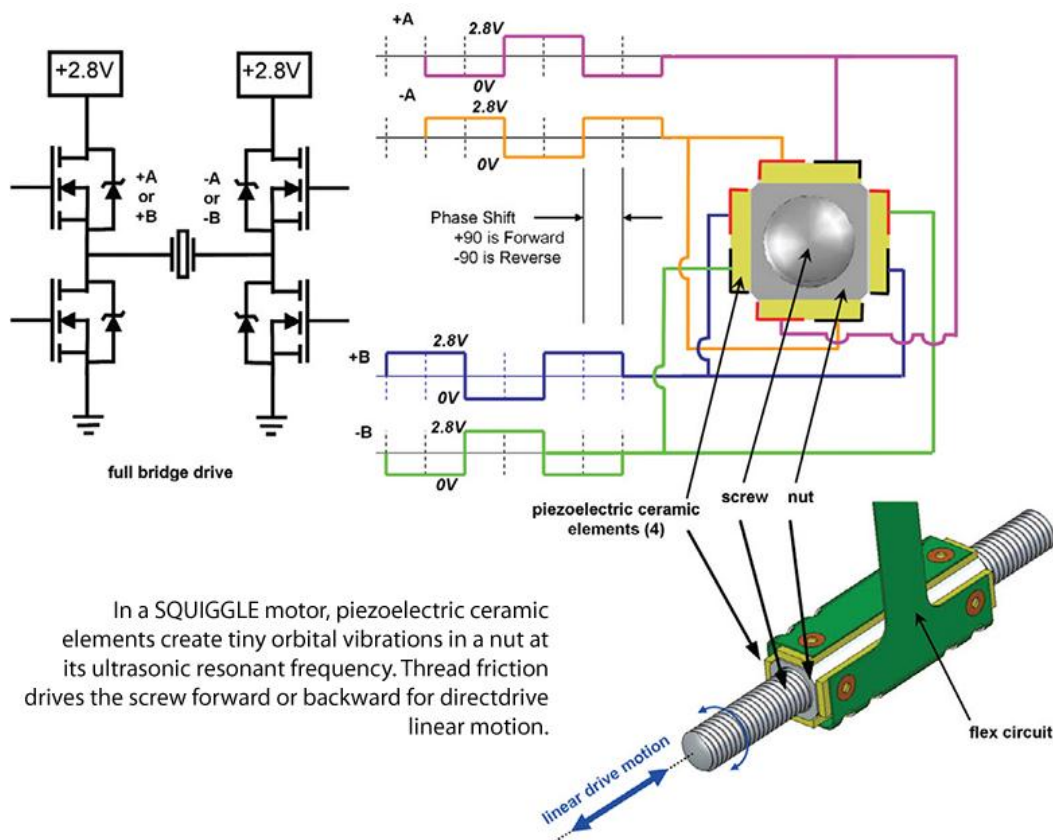


Figure 8 Diagram explaining the Internals and Functions of a Squiggle Motor [21]

Operating voltage	3.3v
Dimensions	1.8 x 1.8 x 6 mm
Availability	YES

Table 2 Linear Piezo Specifications

2.5.3 Piezoelectric Ceramic Bimorph Actuators

The piezoelectric effect is when a force is applied to a certain ceramic; a voltage proportional to the force applied is produced. Conversely when a voltage is applied to the ceramic, the structure changes shape. This piezoelectric effect isn't only limited to ceramics either, many other materials also exhibit the same property which was discovered by Pierre and Jacques Curie in the 1880's.

The piezoelectric actuator can be used for the Electronic Braille Reader by applying a voltage to it, causing it to bend upwards and push the pins. Almost all currently available refreshable Braille displays in the market use bimorph piezoelectric actuators. A drawback of the type of actuators is that the bigger the deflection needed, the bigger the actuator would need to be. [12][13]



Figure 9 Bimorph Actuator [13]

Another drawback of the Piezo ceramic is that it requires 200v DC to be operated. The 200v can seem dangerous but because of the low current it is relatively safe. The high voltage can be acquired by stepping up the voltage using a DC-DC step up converter.

Operating voltage	200 V DC
Dimensions	1.9 x 47 x 0.76
Availability	YES

Table 3 Piezo Ceramic Specifications

2.5.4 Flex Motors

Flex motors have been developed by the nanotechnology group at Cranfield University. They use the principle of ultrasonically vibrating a piezoelectric disk to induce rotary motion. The ultrasonic vibration of the piezoelectric disk causes the attached amplifier structure to vibrate vertically. The ratchetting of elastic legs convert the vibrations into a rotary motion.[23][24]

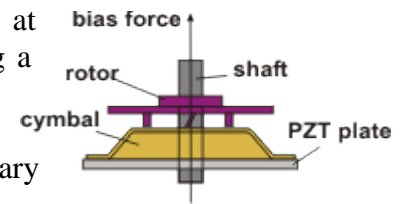


Figure 10 Flexy Motor

The flex motors can be configured to give linear output, which is required for the Braille keys. The 2mm piezoelectric micromotor from flexmotor is small enough to be an individual dot for the Braille cell. But another way the flex motors could be used in the Electronic Braille Reader is by vibrations. When the motor runs, by placing a finger on it vibrations would be felt. Further work would be required to ascertain whether a Braille user could distinguish which dots are vibrating.[23][24]



Figure 11 2 mm Micro motor

Operating voltage	2 Vpk / 500 kHz
Dimensions	2 mm thickness 6 mm
Availability	NO

Table 4 Flex Motors Specifications

The flex motors offer great potential for the Braille actuator. They offer the required actuation in a compact package with low power requirements. As they work on the piezoelectric principle they do not offer any interference with neighbouring actuators. Unfortunately they could not be acquired for experimenting. The research project which developed the motors finished, but they haven't been put into production as explained by Professor Robert Dorey, head of Microsystems & Nanotechnology Centre at Cranfield University

"I'm sorry to say that at this stage there are no definitive plans to put them into production. Unfortunately the project that developed them has now finished so we're not even making them in the lab at this stage."

The complete email can be found in the appendix section B.

2.5.5 Shape Memory Alloy (SMA) / Nitinol Wire

Shape Memory Alloys, sometimes also referred to as smart materials exhibit characteristics which allow them to return to a predetermined shape when heated. A piece of SMA can be made to remember a shape by bending it to a shape and then heating it to a high temperature. At room temperature the piece can be bent into many shapes but will return to the remembered shape when heated again. Heat from passing current through the material can induce the change.[25]

The most common SMA is Nitinol which is an alloy of nickel and titanium. It can be manufactured in many shapes, sizes and forms. Nitinol wire is a popular form of SMA which contracts when current is passed through but requires an opposing force to pull it back when cooled. In different configurations it can be used as actuators. The most basic configuration in Figure 12 gives about 3% movement of the wire.[25][26]

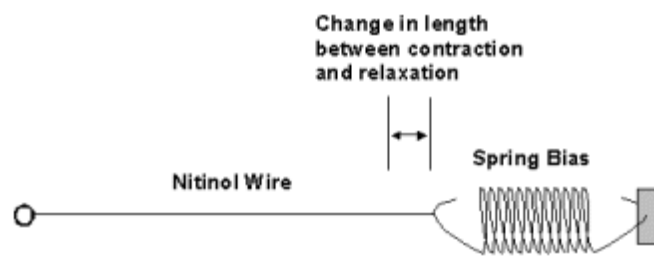


Figure 12 Shape Metal Alloy - Nitinol Wire

To increase the percentage of movement, a more complex configuration can be used such as shown in Figure 13 below. The movement in the wire can be made to push a Braille pin.

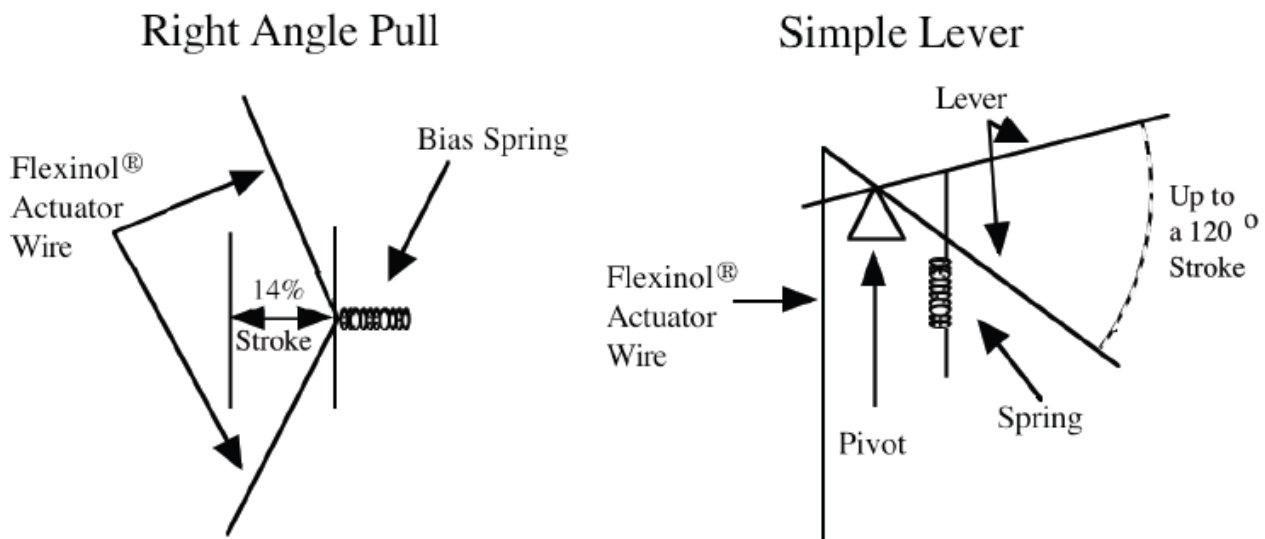


Figure 13 Right Angle Pull/ Simple Lever Examples [27]

Operating voltage	9v
Dimensions	Variable
Availability	YES

Table 5 Nitinol Specifications

2.5.6 FET Polymeric Actuators

The FET-Polymeric actuator is made by integrating organic Field Effect Transistor (FET) and Electro Active Polymer (EAP) actuator. The actuator is made by layering thin-film actuators made of ionic polymer metal composite (IPMC), which exhibit large displacement and high response rates with a layer of organic field effect transistors printed on a plastic sheet. A rectangular plastic actuator is mechanically processed from a perfluorinated polymer electrolyte membrane and at the tip of the rectangular actuator a semisphere is attached. This actuator layer covers the FET. The FET is used to turn on the actuator which bends as a voltage is applied and the semisphere rises to form a Braille dot. The actuators are capable of providing a displacement of 0.4mm while operating at a voltage of $\pm 2.5\text{v}$. The following Figure 14 shows a cross-sectional illustration of the actuator.

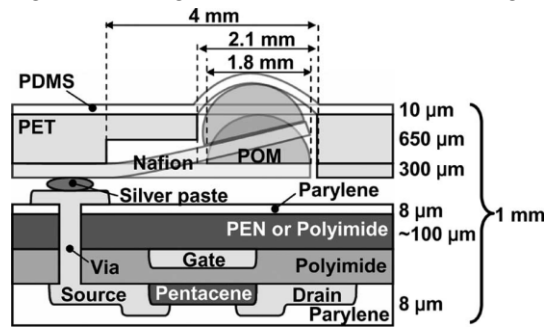


Figure 14 Cross-Sectional illustration of Actuator showing FET, Polymeric Actuator with a semisphere on the tip

As all of the layers are made of flexible materials the actuators are lightweight, mechanically flexible and shock resistant making them perfect for a mobile device. The simplicity of the actuators also allows for them to be manufactured next to each other in Braille configuration. Researchers working on the actuators have manufactured a sheet type Braille display prototype with an array of 24 Braille letters consisting of 4 lines of 6 letters. The Following Figure 15 shows the (a) sheet type Braille display prototype and (b) the different layers of the display

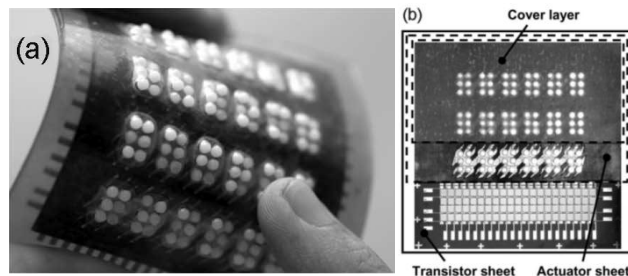


Figure 15 Sheet type Braille Display

Though the prototype worked as intended it still requires further development. Currently the time taken for the actuators to reset is in the magnitude of several minutes, which is too slow. The force and displacement of the Braille dots also varies from dot to dot, this inconsistency makes it difficult to read. The variation is due to inaccuracies in the manufacturing process and can be refined out in large scale production. Another limiting factor currently is the IPMC actuators operate in wet conditions whereas the FETs require dry atmospheric conditions and degrade easily in moisture. This currently limits the life span of the actuators.[28][29]

Operating voltage	± 2.5 Volts
Dimensions	4 x 4 x 1 mm
Availability	NO

Table 6 FET Polymeric Actuator Specifications

2.5.7 Bucky-Gel Actuator

A Bucky gel actuator is a bimorph actuator consisting of two bucky gel electrodes sandwiching a gel layer of polymer-supported internal ionic liquid electrolyte. The bucky gel is a gelatinous room temperature ionic liquid containing single walled carbon nanotubes (CNTs) which allow for quick and long lived operation in air at low applied voltages.

As the following Figure 16 shows, when a square wave with a voltage of ± 3.5 volts at 5mHz is applied between the two bucky gel electrodes, it induces the transfer of ions to the electric layer resulting in the bending motion of the bimorph actuator.[30][31]

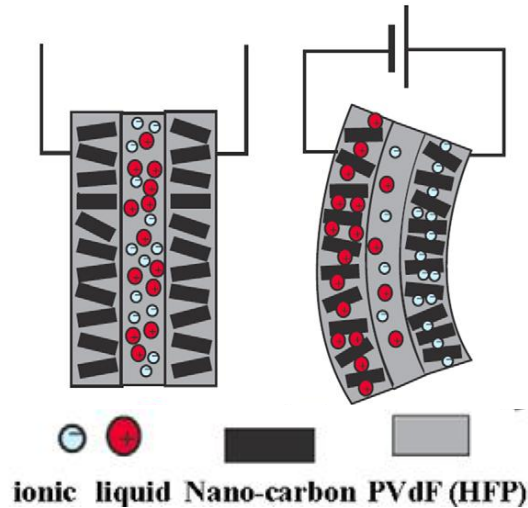


Figure 16 Operation of Bucky-gel Actuator [30]

The bucky gel actuator offers high stability and quick response with minimal voltage requirement and in dry conditions. These qualities make it perfectly suited to actuating Braille dots. The researchers working on the actuator have implemented them in Braille form. The researchers developed an ultra-light and thin Braille display consisting of 6 Braille cells shown in Figure 17. Currently the actuators lack absolute consistency due to manufacturing intolerances which can be refined in large scale production. [30]

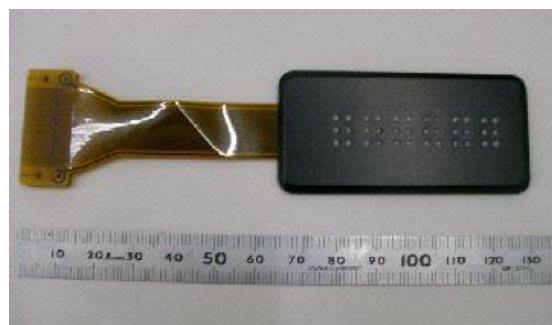


Figure 17 Braille Display using Bucky-gel [29]

Operating voltage	± 3.5 Volts
Dimensions	1 mm
Availability	NO

Table 7 Bucky-gel Actuator Specifications

The literature survey into the Braille standards and existing technologies has highlighted the lack of affordable, portable and usable electronic Braille readers. We propose a novel system design for EBRD which will address some of the problems in the existing technologies to overcome their shortcomings.

3.0 System Design

The Electronic Braille Document Reader (EBDR) has two main parts which need to be evaluated. The software and hardware section: the software part reads the text and converts it to the Braille format and the hardware part which actuates the Braille pattern. The following Figure 18 shows the flow diagram of the EBDR.

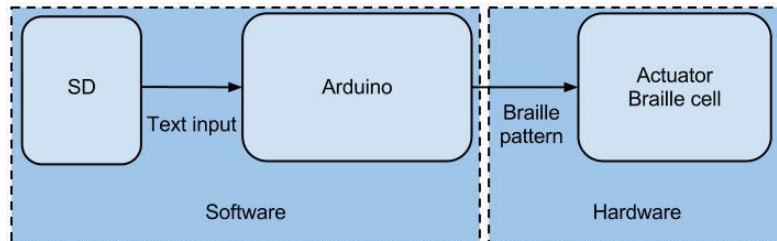


Figure 18 EBDR flow diagram

The two parts can be developed and tested independently. The following sections look at the design of each part.

3.1 Software Design

The EBDR requires text written using Latin character in ASCII format to be read from the memory card and translated into Braille format. The Braille pattern then needs to be actuated on a refreshable Braille cell. The processing of the text requires a programmable microcontroller.

There are many microcontrollers available in the market which can handle the job. An Arduino which is an open-source electronic prototyping platform was chosen because of its flexibility and ease of use. The Arduino board can be expanded using shields e.g. SD shield for the use of SD cards and uses the Arduino Programming Language which is based on C/C++ but is simpler to code. An Arduino Mega 2560, which uses the Atmel's ATmega2560 microcontroller, was chosen because it provides the required I/O and ability to extend with extra shields if required.



Figure 19 Picture of Arduino Mega 2560

To actuate the Braille pattern first the text has to be read from the SD card and then each character has to be cross-referenced with its equivalent Braille pattern from a table. The resulting pattern is then sent to the output and the process repeated over again for the next characters. The following Figure 20 shows the flow diagram of the code.

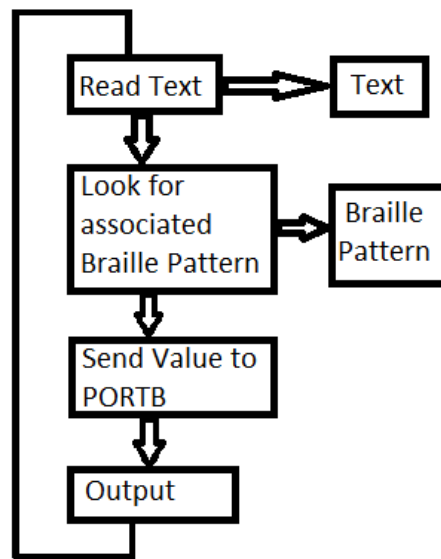


Figure 20 Flow Code Diagram

The code to achieve the task was broken down in two steps. The first step being able to read text from within the code and convert to Braille and the next step being able to read the text from an external SD card. To be able to convert the text to Braille a table with the Braille pattern for the entire alphabet, numbers (0-9) and symbols was created in a form which could be sent to the output. To do this each Braille pattern was turned into binary code with '1' being dot on and '0' being dot off. Eight dot Braille was chosen because of its ability to encode all the character information in a single cell. The following Table 8 shows an example of a few characters. The Figure 21 shows how the Braille cell is configured and the encoding of the character detail.

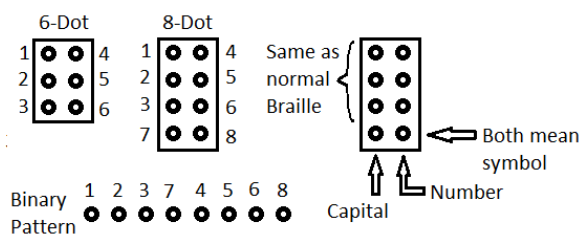


Figure 21 Braille Pattern

B	⠠	11010000
G	⠠	11001100
5	⠠	10000101

Table 8 Braille Code to Binary Format

The Braille code in binary format can then be sent to the output pins on the Arduino which set the required pins high and low. The following pages explain the development of the Arduino code for the EBD.

3.1.1 Braille Translation

The following Figure 22 shows the Braille pattern encoded in binary form in the Arduino code.

```
boolean pattern[72][8]={

    // Lowercase
    /*a*/ {1,0,0,0,0,0,0,0}, /*b*/ {1,1,0,0,0,0,0,0}, /*c*/ {1,0,0,0,1,0,0,0}, /*d*/ {1,0,0,0,1,1,0,0},
    /*e*/ {1,0,0,0,0,1,0,0}, /*f*/ {1,1,0,0,1,0,0,0}, /*g*/ {1,1,0,0,1,1,0,0}, /*h*/ {1,1,0,0,0,1,0,0},
    /*i*/ {0,1,0,0,1,0,0,0}, /*j*/ {0,1,0,0,1,1,0,0}, /*k*/ {1,0,1,0,0,0,0,0}, /*l*/ {1,1,1,0,0,0,0,0},
    /*m*/ {1,0,1,0,1,0,0,0}, /*n*/ {1,0,1,0,1,1,0,0}, /*o*/ {1,0,1,0,0,1,0,0}, /*p*/ {1,1,1,0,1,0,0,0},
    /*q*/ {1,1,1,0,1,1,0,0}, /*r*/ {1,1,1,0,0,1,0,0}, /*s*/ {0,1,1,0,1,0,0,0}, /*t*/ {0,1,1,0,1,1,0,0},
    /*u*/ {1,0,1,0,0,0,1,0}, /*v*/ {1,1,1,0,0,0,1,0}, /*w*/ {0,1,0,0,1,1,1,0}, /*x*/ {1,0,1,0,1,0,1,0},
    /*y*/ {1,0,1,0,1,1,1,0}, /*z*/ {1,0,1,0,0,1,1,0},

    // Uppercase
    /*A*/ {1,0,0,0,1,0,0,0}, /*B*/ {1,1,0,1,0,0,0,0}, /*C*/ {1,0,0,1,1,0,0,0}, /*D*/ {1,0,0,1,1,1,0,0},
    /*E*/ {1,0,0,1,0,1,0,0}, /*F*/ {1,1,0,1,1,0,0,0}, /*G*/ {1,1,0,1,1,1,0,0}, /*H*/ {1,1,0,1,0,1,0,0},
    /*I*/ {0,1,0,1,1,0,0,0}, /*J*/ {0,1,0,1,1,1,0,0}, /*K*/ {1,0,1,1,0,0,0,0}, /*L*/ {1,1,1,1,0,0,0,0},
    /*M*/ {1,0,1,1,1,0,0,0}, /*N*/ {1,0,1,1,1,1,0,0}, /*O*/ {1,0,1,1,0,1,0,0}, /*P*/ {1,1,1,1,1,0,0,0},
    /*Q*/ {1,1,1,1,1,1,0,0}, /*R*/ {1,1,1,1,0,1,0,0}, /*S*/ {0,1,1,1,1,0,0,0}, /*T*/ {0,1,1,1,1,1,0,0},
    /*U*/ {1,0,1,1,0,0,1,0}, /*V*/ {1,1,1,1,0,0,1,0}, /*W*/ {0,1,0,1,1,1,1,0}, /*X*/ {1,0,1,1,1,0,1,0},
    /*Y*/ {1,0,1,1,1,1,1,0}, /*Z*/ {1,0,1,1,0,1,1,0},

    // Numbers
    /*1*/ {1,0,0,0,0,0,0,1}, /*2*/ {1,1,0,0,0,0,0,1}, /*3*/ {1,0,0,0,1,0,0,1}, /*4*/ {1,0,0,0,1,1,0,1},
    /*5*/ {1,0,0,0,0,1,0,1}, /*6*/ {1,1,0,0,1,0,0,1}, /*7*/ {1,1,0,0,1,1,0,1}, /*8*/ {1,1,0,0,0,1,0,1},
    /*9*/ {0,1,0,0,1,0,0,1}, /*0*/ {0,1,0,0,1,1,0,1},

    // Symbols
    /*./*/ {0,0,1,1,0,0,0,1}, /*,/*/ {0,0,1,0,0,0,0,1}, /*:/*/ {0,1,1,1,0,0,0,1},
    /*!*/ {0,1,1,1,0,1,0,1}, /*?*/ {0,1,1,1,0,0,1,1}, /*'*/ {0,0,1,1,0,1,1,1}, /*(*/ {0,1,1,1,0,1,1,1},
    /*)*/ {0,1,1,1,0,1,1,1}, /*-*/ {0,0,1,1,0,0,1,1}, /*space*/ {0,0,0,0,0,0,0,0}
};
```

Figure 22 Braille patterns

Boolean data type is used to declare the Braille pattern. The pattern array is named “pattern” and has 72 different patterns each consisting of 8 bits.

Each character pattern is saved in form of 0’s and 1’s which can be sent to the output.

```
char alphabet[72]={'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',
'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',
',','.',',','!','?',',','(',')','(',')','(',')',
};
```

Figure 23 List of characters

A list of all 72 characters was declared as char as shown in the above Figure 23. The char identifier is followed by the name “alphabet” which is used to call the list later in the code.

```
String input = "TeSt Data 1823"; //the text to be converted

int letter = 0; //reference to the current letter (the current row in the pattern array)
```

Figure 24 Input text

Figure 24 shows the text to be translated to Braille was declared as a string and named input. Another variable was declared as letter to find the position in the pattern array. At this point all the declarations have been done. In an Arduino code the next step is to setup all the inputs and outputs.

```

void setup() {

  for(int p=2; p<=9; p++){ //Loop to go through pin 2 till 9
    pinMode(p, OUTPUT); //declare pins 2 to 9 be output
  }
}

```

Figure 25 Setup outputs

The setup as shown in Figure 25 is kept simple with only the outputs used declared. In order to set a pin to output mode a function from the Arduino's library can be used. The pin Mode function has two parameters which can be set, the pin and its mode; input/output. In the code the pin Mode function was called and the parameters set to output but the pin number was set as a variable p. The function was called within a for loop which was set to increment p from 2 till 9. This made the function keep looping until pins 2 till 9 were set to output.

```

void loop() {

  for(int dot=2; dot<=9; dot++) { //Loop to move through pin 2 to 9
    int x=0;
    while(alphabet[x] != input.charAt(letter)) { x++; } //Go through input and compare with alphabet to look for same characters
    digitalWrite(dot, pattern[x][dot-2]); //Use position of the character in alphabet to determine the position of Braille pattern
  }

  if(letter==input.length()-1) { //If position is at end of input text then reset position
    letter=0;
  } else {
    letter++; //Otherwise keep incrementing position
  }

  delay(2000); //delay of 2 seconds
}

```

Figure 26 Main loop

The actual work happens in the main loop which is initiated with a void loop() and can be seen in Figure 26. Firstly a for loop is used to select a dot, within that loop a while loop is used to find the pattern for the text. The line `while(alphabet[x] != input.charAt(letter)) { x++; }` says if alphabet is not equal to input then increment x which changes the position in alphabet list. The while loop continues until the statement is true at which point the program continues to the next statement. The statement `digitalWrite(dot, pattern[x][dot-2]);` uses the write function from the Arduino library. The function is followed by its parameters which in the case of digitalWrite is pin number and value to be sent to it. In the statement used the parameters are defined as the value of dot for the pin and the state in the pattern array at the position x, the [dot-2] adjusts the position of x to account the pins starting at 2 and not 0. The for loop ends with the brace after that statement.

The for loop is followed by an if else statement `if(letter==input.length()-1) { letter=0;} else{letter++;}` which says if the value of letter is equal to length of input string then reset letter to 0 otherwise increment the value of letter. The value of letter used in the while loop to determine the position in input.

The main loop ends with a delay function `delay(2000);` which gives a 2 seconds delay before going back to the start and continuing the entire process for the next character.

3.1.2 SD Interface

The SD card interfacing for the EBDP requires an extra SD shield to be attached to the Arduino Mega2560. The adafruit SD breakout board was chosen for this because of its simplicity in design. It provides a way to interface a microSD card with the Arduino without the unnecessary extras. The following Figure 27 shows the microSD shield which was used with the pinout connection to the Arduino Mega2560.

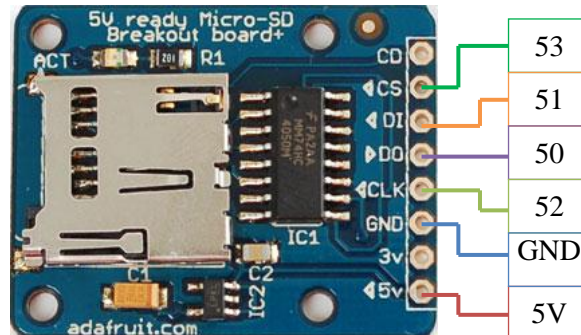


Figure 27 Adafruit microSD shield [32]

The adafruit shield requires its own library which is available from the adafruit website [32]. Once downloaded and linked to the Arduino software it can be called up when required by using `#include <SD.h>` as was done in the code.

```
#include <SD.h> //SD Library

const int intChipSelectpin = 53; // Chipselect for Arduino 2560(mega) pin 53

// File on the SD Card
File myFile; // the file object
const String strFileName = "test.txt"; // Name of the file on SD card
String strLineRead; // the string read from the strFileName file

const boolean SerialReporting = true; //Turn serial reporting on/off true/false
```

Figure 28 SD Declarations

For the SD card to work a few things first need to be declared as shown in Figure 28. For SD card shields a chip select pin has to be declared which varies depending on the board used. For adafruit interfacing with mega2560 the pin connects to pin 53 on the Arduino so has to be declares by `const int intChipSelectpin = 53;` statement.

Before the start of the code a few global variables first needed to be declared. A variable myfile is declared as a file for when a file is read from the SD its name can be stored using the variable. The name of the file from the SD card “test.txt” is saved as a constant string called strFileName. Another string strLineRead is also declared as it will be used in a function to read from the file.

To help with debugging the Arduino code a serial monitor can be utilized to get feedback from the board. As it’s not always required it was declared at the start, with an option to either turn the serial reporting function on or off.


```

void setup() {
    if (SerialReporting) {
        Serial.begin(9600); //set up Serial library at 9600 bps
        Serial.println("Just began..."); //print out

        // Success of operations
        boolean Success;

        Success = InitializeSDCard(); // Initialize SD Card

        if (Success) { // If all OK, then read card
            Success = ReadFileFromSDCard(); // Read the text from file strFileName into the strLineRead string
        }

        if (SerialReporting && !Success) Serial.println("Card Reading stopped with errors!"); // If it fails, report stop and don't continue
                                                // with the loop() function

        input = strLineRead; // update input string with data from SD
    }
}

```

Figure 29 SD Setup

The setup as shown in Figure 29 required some extra code for the SD card reading to work. Firstly the serial reporting was setup to assist with debugging if required later. It was achieved by the command `Serial.begin(9600);` which initiated serial communication at 9600 bits per second; a standard for Arduino serial communication. It was then followed by `Serial.print("Just began...");` which prints "just began..." on the serial monitor when the program starts.

A variable Success was declared as a Boolean function to help confirm successful initialization of the SD card. The variable success is used to store the value of the initializeSDCard function which is setup later in the code.

An if statement was used to determine the progression of the program by saying if the previously declared Boolean success is true then call the ReadFileFromSDCard function. If it is not then the next if statement prints "Card Reading stopped with errors!" to the serial monitor.

The original input string is also updated with the strLineRead. This allows for the code in the main loop to remain unchanged. Two of the functions called in the setup loop are declared at the end of the code.

```

// SD CARD FUNCTIONS
boolean InitializeSDCard(void) {
    // Initializes the SD Card and reports any problems
    // or success through the Serial port
    //=====

    if (SerialReporting) Serial.println("Initializing SD card..."); //If enabled, report via Serial Port

    // must be left as an output or the SD library functions will not work
    pinMode(53, OUTPUT); // pin 53 on mega, must be left as an output or the SD library functions will not work

    if (!SD.begin(intChipSelectpin)) {
        if (SerialReporting) Serial.println("Initialization failed!"); //If enabled, report Error via Serial Port
        return false; // report failure
    }

    if (SerialReporting) Serial.println("Initialization done."); //If enabled, report Success via Serial Port:
    return true; // report success
}

```

Figure 30 Initialize SD Card Function

The initialize SD card function shown in the Figure 30 is declared as Boolean as the result from it is required as true or false state. At the start of the function the serial print command is used to report to the serial monitor that SD card is initializing.

For the SD library to work the SS pin on the board has to be declared as an output. The SS pin on the Arduino Mega2560 is pin 53, which is already declared in the SD library. All that is required is to set it to output which was done using the statement *pinMode(SS, OUTPUT);*.

The main part of InitializeSDCard function is to check whether the microSD card is inserted or not. An if statement was used to say if the command SD.begin does not detect SD card through intChipSelectpin which is pin53 then print “Initialization failed!” to serial monitor and return false.

If it does not detect that then the code moves on to the following if statement which prints “Initialization done.” to serial monitor if serial reporting has been turned on. The code then moves to *return true* to report success.

```
boolean ReadFileFromSDCard(void) {
    // Reads File from file strFileName
    //=====
    strLineRead = ""; // Clear the string that receives the text in the file
    char chrRead;
    // char array used, to be able to pass the file name to the SD.open() function from strFile
    char charFNameBuf[50];
    strFileName.toCharArray(charFNameBuf, 50);

    myFile = SD.open(charFNameBuf); // Open the file for reading;

    if (myFile) {
        if(SerialReporting) Serial.println("File " + strFileName + " opened successfully");// report Success via Serial Port

        // read from the file until there's nothing else in it:
        while (myFile.available()) {
            chrRead = myFile.read();// Read the character as a number equivalent to its ASCII code
            strLineRead.concat(chrRead); // Concatenate the character read into the final string
        }
        myFile.close(); // close the file:

        if(SerialReporting) { // print string read to serial port
            Serial.println("String read:");
            Serial.println(strLineRead);
            Serial.println();
        }
        return true;
    }
    else {
        //If enabled, report Error via Serial Port
        if(SerialReporting) Serial.println("Error opening file " + strFileName );//report Error via Serial Port
        return false;
    }
}
```

Figure 31 Read File from SD Card Function

The read file from SD card function shown in Figure 31 actually reads the data from the SD card. This function is also declared using Boolean as true or false states are required when called upon in the setup loop. Firstly the strFileRead is cleared ready to receive the text in the file. A few variables were also declared like the *charFNameBuf[50]* which is used as a buffer while passing the file. The file was opened with the command *SD.open(charFNameBuf)* and stored in the variable myFile.

Then an if statement was used to confirm if the file was opened, if so then the file name followed by “opened successfully” was printed to serial monitor. Within the if statement a while loop was used to read the file until there’s nothing left. While myFile was available it was read using the command *read()* and saved in to the variable *chrRead*. The data was then concatenated in to a final string *strLineRead*. After reading the file was closed using the statement *myFile.close();*.

The *serial.println* command is then used to first print “string read:” followed by the data read from the test.txt file. The finish the successful reading of the SD card true state is returned.

At the end of the function an else statement was added to complete the if statement. So if myFile was available it is read else “Error opening file” followed by files name is reported to the serial moitor and a false state is returned. With the end of this function the EBDR code is complete.

3.2 Hardware Design

The EBDR requires text in Braille format to be actuated on a finger. The Braille pattern received from the Arduino is in the form of eight pins at a state high or low, each pin representing a dot. Each actuator was connected to the outputs of the Arduino.

3.2.1 Actuators

Many of the actuators reviewed earlier are suitable to actuate the Braille dots. Unfortunately most of the actuators ideal for the device are still currently in development and require refinements until they are available. Other actuators are too large to be arranged in a Braille formation or are too expensive to get hold of to experiment.

To test the feasibility of the EBDR a Braille module is required which can actuate the pattern to be felt by a finger. For this test the overall compact form factor of the Braille module isn't required except that the Braille dots fit within the standard cell dimensions.

As the Nitinol wire is inexpensive and is readily available, it was chosen to build the actuator. The Nitinol wire is available in different sizes of length and thickness. The differences in thickness affect the speed and strength of the actuation. The thicker the wire the more force it can exert but at a slower rate. However, the thinner the wire the faster the actuation, but with a loss of force. Nitinol wire with a diameter of 0.5mm was chosen to design the actuator due to its fast switching ability.

To build the actuator the wire was cut to length. One end was attached to cardboard and on the other end a spring was attached to apply a reverse force to pull the wire back to its original length when cooled. Due to the Nitinol being an alloy of nickel and titanium solder does not stick to it, therefore crimps are usually used to make an electrical connection and fasten it to a structure. In this case the ends were held by squeezing them in between a M2 nut and bolt. An electrical connection was made by attaching crocodile clips to the ends of the bolts. Although crimps are more secure they are hard to source for the size needed and they don't allow for adjustments in length once crimped as nuts and bolts do.

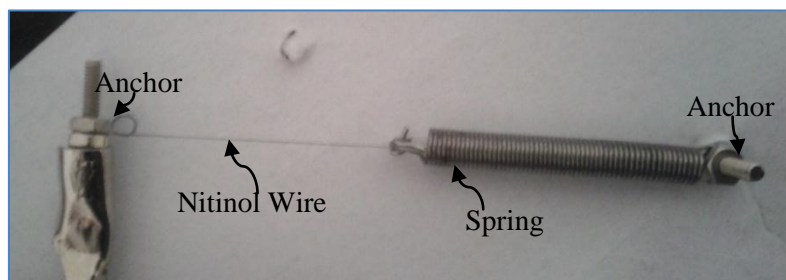


Figure 32 Nitinol in Series Configuration

In a simple series configuration with the Nitinol wire attached to a spring on one end as shown in Figure 32, the Nitinol moves only very slightly, barely noticeable. In order to amplify the actuation a few more complex configurations were experimented with. One of those used a lever to amplify the movement of the Nitinol. The lever configuration has the wire attached to one end of the lever close to the fulcrum and a spring on the other end to provide the opposing force. When the Nitinol is activated, it pulls the lever down; the small movement on one side of the fulcrum is amplified by the lever on the other side. The amplification depends on the length of the lever and where the Nitinol is attached to it. Figure 33 shows the lever configuration.

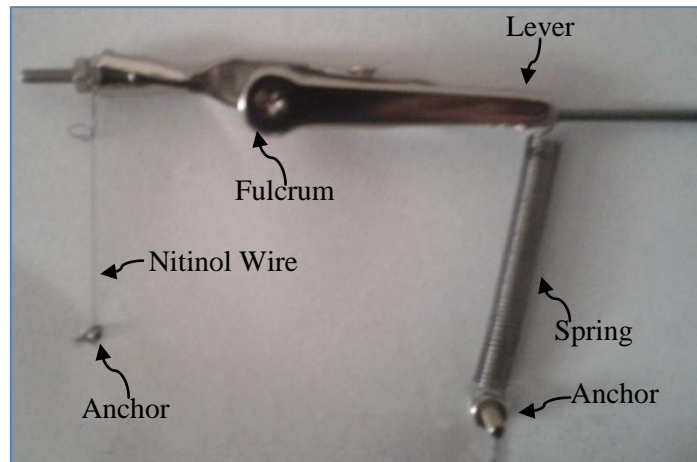


Figure 33 Lever Configuration

The lever configuration does amplify the movement of the Nitinol considerably but due to the pulling forces on the Nitinol it breaks easily and often. The complexity of having many parts also makes it difficult to control the amount of movement, and replicating seven more with similar amounts of movement is impossible with so many variables.

Another configuration experimented with was the V configuration where the Nitinol is anchored at both ends with bolts and a spring pulls the wire from the middle forming a V with the Nitinol. When the Nitinol is activated it shrinks and the point of the V moves up, when it cools the spring pulls the point back to the original position. The following picture in Figure 34 shows the V configuration:

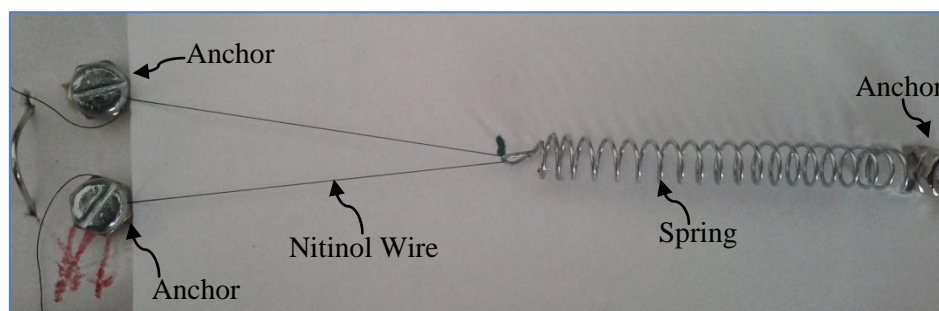


Figure 34 V Configuration

The V configuration shown in Figure 35 amplifies the movement slightly, enough to be seen. The figure below shows the Nitinol in a relaxed state next to an activated one to compare the amount of movement.

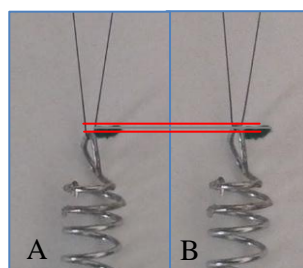


Figure 35 (A) Relaxed Nitinol (B) Activated Nitinol

The configuration provides movement but not enough as required for the Braille actuator. The length, tension of spring and angle of V were adjusted to see if they affected the amount of actuation. The actuator worked best when the tension from the spring was high so that the Nitinol was tight. The length of the Nitinol made a difference but only when it was changed considerably. The major improvement came by changing the angle

of the V. This was done by increasing the distance between the two ends of the Nitinol wire. Best performance was achieved when the V was opened up in to a horizontal line with the spring pulling down from the middle. The following Figure 36 shows the wide angle V (WAV) configuration actuator.

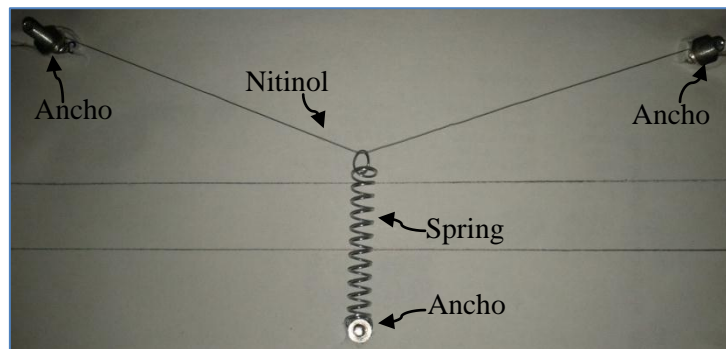


Figure 36 Wide Angle Configuration

The WAV configuration amplified the actuation to the amount required for the Braille actuator. The Figure 37 below shows the comparison of the relaxed and activated Nitinol.

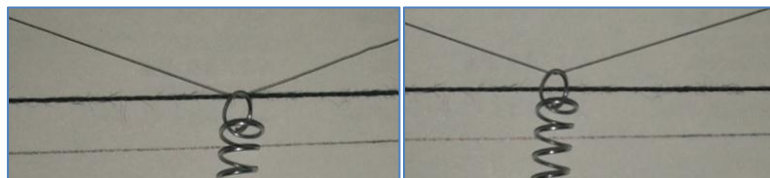


Figure 37 Amount of actuation in WAV configuration

The WAV configuration as shown in Figure 36 was chosen to build the actuators to raise the Braille dots. In that configuration the Nitinol pulls the spring up and when relaxed the spring pulls the Nitinol back down, but for the EBDR a pin needs to be pushed up a few millimetres through a hole on a flat surface. It also needs to actuate eight pins arranged in two columns of four dots so each actuator needs to occupy limited space. The actuator was modified to incorporate a pin and minimize its size.

The Braille pins were made using parts which were available at hand. Figure 38 shows the parts used to build the Braille pin alongside a fully constructed pin. The springs were taken out from pens as well as the ink tubes. The thin rods were taken out of lever arch folders and screws and washers were bought from a local hardware store. The M1 screw was longer than required so was sawed in half and then screwed in to the bottom of the plastic ink tube which had a slightly smaller diameter hole, thus providing a tight fit. The plastic ink tube was cut to 1.5cm and the rod was pushed in through the other end, the rod had the same diameter as the inner diameter of the ink tube so was a snug fit.



Figure 38 Braille Pin Construction

To reduce the size of the actuator the spring was moved up inside the opened up V. This way the spring pushes the wire instead of pulling, as the springs in pens are compression springs it works better this way. The pin is then slid inside the spring and the slot in the screw rests on the Nitinol as can be seen in Figure 39. Two bolts at the top in line with the ends of the Nitinol hold the spring in place. When the Nitinol is activated it shrinks and pushes the pin up and compresses the spring which in turn pushes the Nitinol back in shape when it cools. Figure 40 shows the actuator fully assembled:

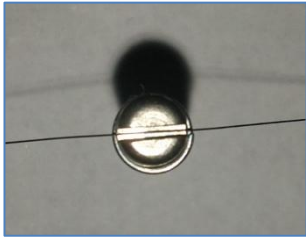


Figure 39 Nitinol resting in screw slot

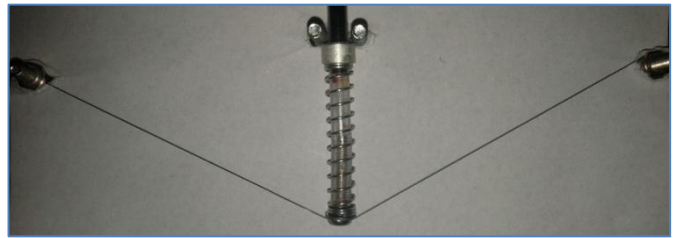


Figure 40 Fully assembled actuator

The actuator works consistently and provides the required amount of actuation. Figure 41 shows the comparison of the relaxed and activated actuator side by side. The Nitinol wire reacts fast when the required power is applied to it and can actuate in less than 0.5 seconds. To reset the actuator, the power is switched off at which point the wire starts to cool and the spring pushes it back to shape. The cooling period is considerably longer and can be in excess of 1 second. To reduce the reset time a stronger opposing force can be applied but that then requires more power to actuate and overcome the force. It also increases the likelihood of the Nitinol wire breaking due to excess stress. Another way is to help the wire dissipate the heat quicker. A simple way to do this is to use a fan to create airflow around the Nitinol wire. A small fan scavenged from a computer was used to cool the Nitinol which reduced the reset time to less than a second.

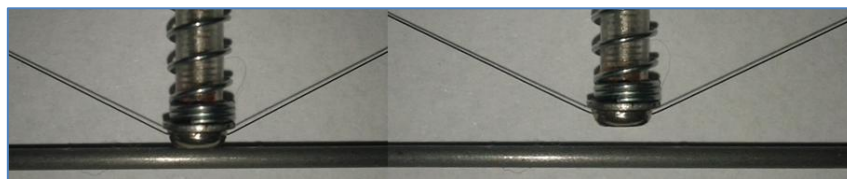


Figure 41 Actuator relaxed vs activated

As the above actuator design functioned as required, it was chosen for the prototype. For the prototype the actuator needed to be replicated seven times and assembled in a way so all the pins lined up to form a Braille cell.

The prototype with all actuators arranged in a Braille cell was constructed with plates of Plexiglas as the base. In the chosen design the actuator spans horizontally so the four pins in the Braille cells can easily be stacked on top of each other. But the two columns are forced apart due to the size of the actuator which spans about 9cm from one end of the Nitinol to the other. To overcome this the actuators were divided into two columns and each column built on a different level, with the pins of the bottom column being longer so they extend through to the same height as the tip of the top level as can be seen in Figure 44. The actuators on the top level were offset at 45 degrees so that the pins coming through from the bottom level don't make contact with the Nitinol wire. Figure 42 and Figure 43 show the arrangement used for the eight actuators to form a Braille cell. The top plate sits on top of the bottom plate with a 4cm gap in between for the actuators. The holes (red on the diagram) on the top plate for the left column actuator help line up both plates. Further details on the dimensions can be found in the appendix section C.

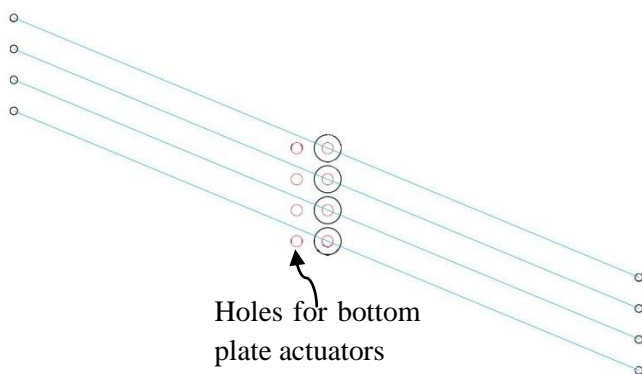


Figure 42 Top actuator plate

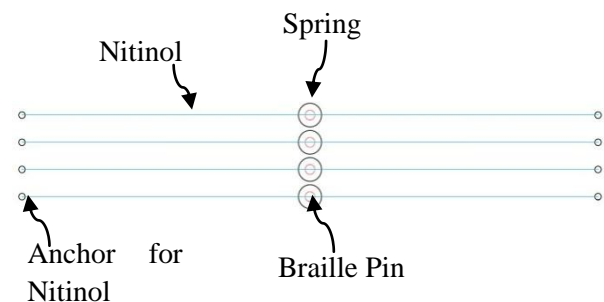


Figure 43 Bottom actuator plate

Holes were drilled in the Plexiglas plates in the configuration shown in the above Figure 42 and Figure 43. Large holes with a diameter of 3.6mm were drilled half way through the plate for the springs to slot in and a smaller 1.5mm hole inside it going all the way through for the pins. Both plates had a column of these holes for each column of the Braille cell. The top plate had another column of the 1.5mm holes for the bottom actuators to come through to form the entire Braille cell. Holes with a diameter of 1mm were drilled for the Nitinol wire which was secured in the hole using screws.

An electrical connection with the Nitinol was made by attaching a wire to the screw holding the Nitinol in place. One side of the actuators was connected together and grounded. Power was applied to the other side to activate each actuator. Figure 44 shows the actuators on the top and bottom plate.

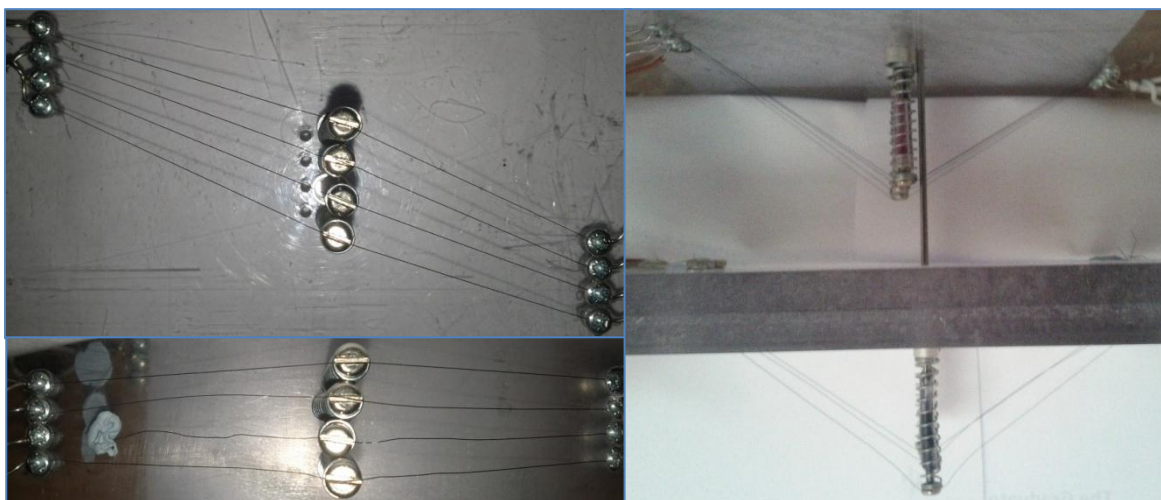


Figure 44 Actuators assembled together

3.2.2 Drive Electronics

The Nitinol requires considerable amount of current to actuate the required amount. Approximately 500mA is required to actuate each actuator at the required speed. To prevent the pins on the Arduino from being damaged transistors were used to power the actuators. The transistors were used as a switch to direct power from the battery to the Nitinol wire. NPN Darlington pair transistors TIP122 were used as they are able to handle the large current flow and are intended for switching applications.

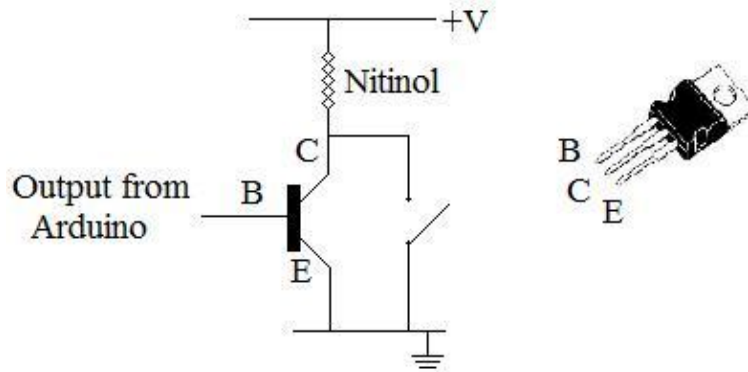


Figure 45 Transistor Connections

The above diagram shown in Figure 45 depicts how the transistor was connected to the Nitinol. The Darlington pair includes two transistors together so the amplified current from one transistor is further amplified by the second transistor. A Darlington pair behaves like a single transistor. To turn on the transistor 0.7V is required at the base emitter junction of each transistor. So the Darlington requires 1.4V to turn on. The Nitinol was connected to power from one end and the other end leading to the collector of the transistor. The emitter of the transistor was connected to the ground terminal. When logic 0 is provided to the base of the TIP122 the Nitinol is left floating as the impedance between collector and emitter is very high. When logic 1 is provided to the base, its collector and emitter get shorted and as a result the Nitinol gets grounded allowing the current to flow. An override switch is connected in parallel with the TIP122 transistor to manually switch the Nitinol.

Figure 46 shows the drive circuit integrated with the connection module linking everything together. Push to make switches were used to manually turn on the actuator and LED's were also connected in parallel with the Nitinol actuators to indicate on/off states when the Braille cell is not visible.

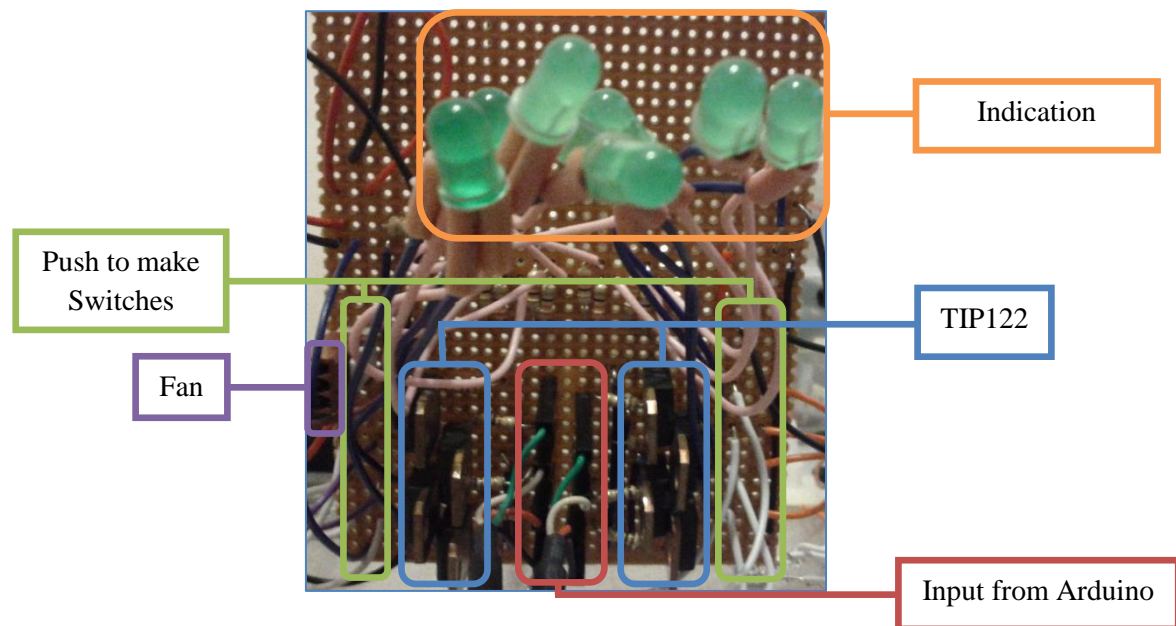


Figure 46 Drive Circuit with Indication LED's

The different parts were assembled together on the Plexiglas. Wires with pins and connectors were used to interface the Arduino with the actuators as they could be easily connected and disconnected. Holes were drilled in the top plate to mount the eight manual override Push to make switches and indication LED's in Braille configuration. A fan was mounted on the side of the plates so that it would blow air over all eight actuators. The image in Figure 47 below shows the top view of the fully assembled EBDR.

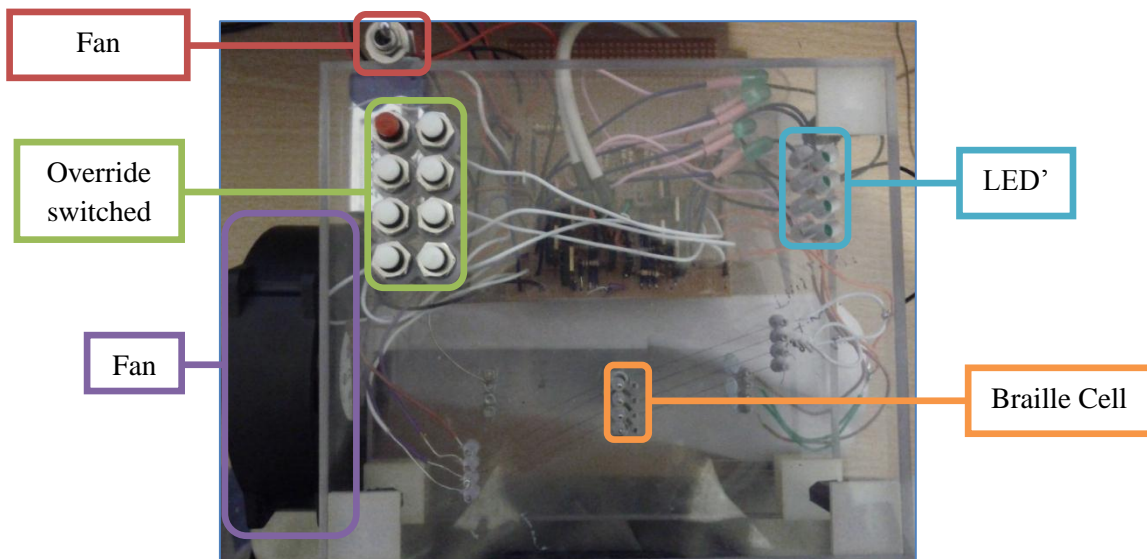


Figure 47 Electronic Braille Document Reader Prototype

4.0 System Test

The EBDR consists of two main parts, the text processing module which is part of the software section reads text from SD card and translates into Braille pattern, and the actuator module part of the hardware section which actuates the Braille pattern. It was designed and built in two modules so it could be tested separately and as a single integrated system. The following sections describe the test procedure for the separate modules and the EBDR as a complete product.

4.1 Text Processing

To test the code and determine if the Arduino was able to translate the text into Braille pattern, an LED module was built. The LED's were arranged in a Braille cell and each LED connected to a pin on the Arduino instead of the actuators. The LED's displayed the corresponding Braille pattern, which would otherwise be actuated by the actuators. Figure 48 shows the LED module.

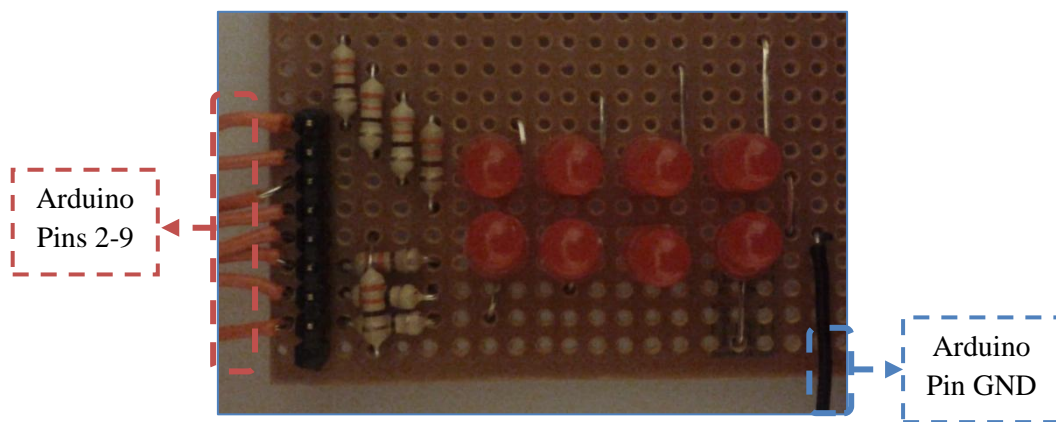


Figure 48 LED Braille module

A test file in .txt format was saved in the SD card with the text “PEACE be upon you! 321” to test if the Arduino would translate the text correctly. The microSD card was then inserted in to the microSD card adapter on the SD shield.

The test text was also manually translated into Braille form to be able to compare the results from the LED Braille display. Figure 49 shows the text “PEACE be upon you! 321” translated in to Braille.



Figure 49 PEACE be upon you! 321 in Braille

The serial monitor in the Arduino software was used to get feedback from the Arduino. When writing the code serial reporting was implemented to report back the progress. The serial monitor was implemented to inform if the SD card would initialize, if the file could be opened and then to display the text from the SD card.

4.2 Actuators

The actuators when built on the Plexiglas plates were built with push to make switches in series with them to manually turn each one of them on. LED's were also connected in parallel to the actuators to indicate if the actuator was on or not when the pins were covered with a finger.

To test the actuators, power was applied to the circuit and each actuator was turned on one by one. The cooling fan was also turned on to help the actuator reset quicker.

4.3 EBDR

The individual pieces of EBDR were tested independently to confirm the correct working order of each module. After the correct working order of the modules the Arduino was then connected directly to the actuator module. Figure 50 shows the fully assembled Electronic Braille Document Reader.

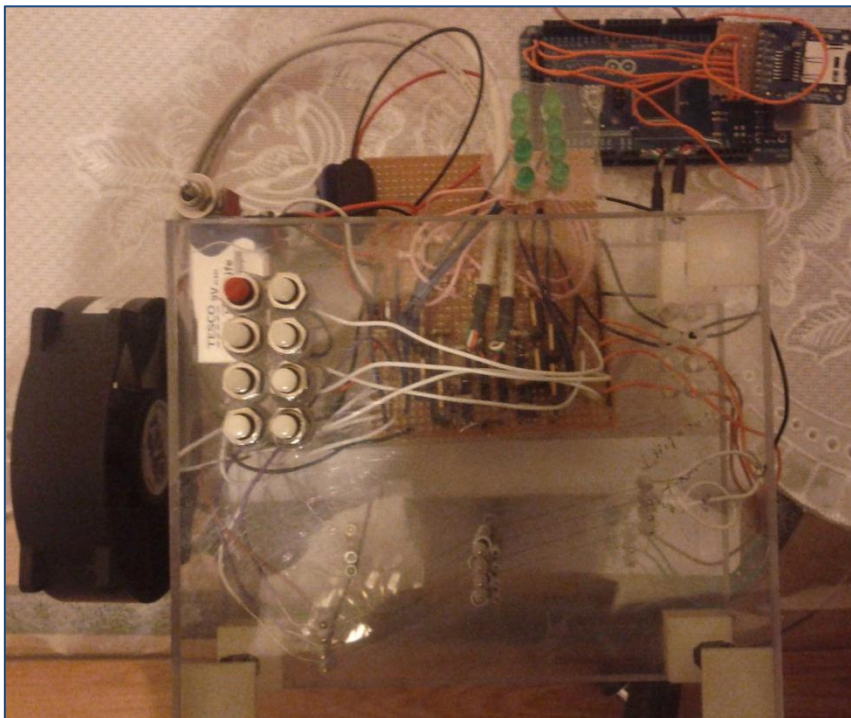


Figure 50 Fully Assembled EBDR

The same text file was used to test the whole assembly as the one used to test the Arduino module individually. Power was supplied to the actuator module and the fan turned on then the Arduino was also turned on to provide the Braille pattern to the actuators.

An appointment was arranged with a visually impaired and a blind individual to test the EBDR by reading the Braille by touch. They were made to read out the pattern they could feel. The text in the SD card was displayed on the Braille along with some random characters actuated manually using the switches. Additional questions were also asked on the usability and the benefit of such device.

5.0 Results

The individual modules of the EBDR, and the fully assembled EBDR prototype were tested using multiple inputs and the output results were recorded for further comparison and analysis. The following sections present the results of the tests conducted on the individual modules as well as the assembled prototype.

5.1 Text Processing

The serial monitor showed the progress of the code. It indicated the SD card initialized correctly and displayed the text from the test.txt file on the serial monitor. Figure 51 shows the output on the serial monitor.

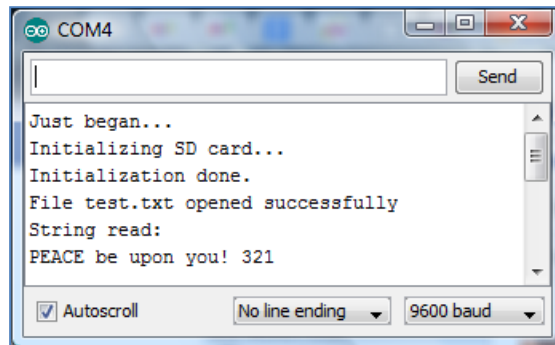


Figure 51 Serial Monitor initialization

output – Successful

Straight after the report on the serial monitor the LED started displaying the Braille patterns. Pictures of each pattern was taken and compared with the expected output. The Braille patterns matched perfectly. The following image in Figure 52 shows all the Braille patterns displayed in sequence.

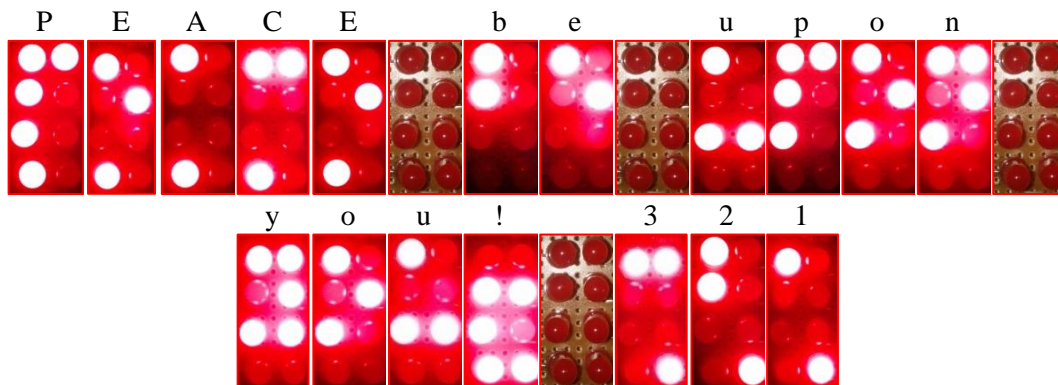


Figure 52 Braille Output on LED Braille cell

The SD card was then removed from the device and the test run again. The serial monitor indicated that the SD card failed to initialize. Figure 53 shows the output on the serial monitor.

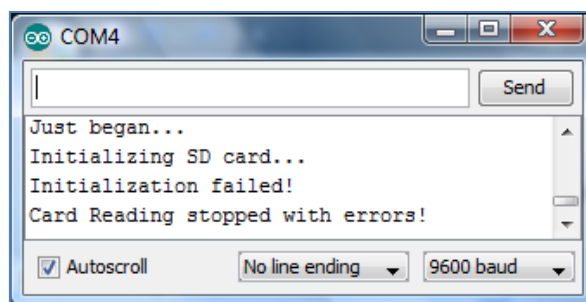


Figure 53 Serial Monitor output - Failed initialization

5.2 Actuators

Each actuator was actuated manually using the push to make switches. The actuators actuated as required. The indicator LED's also displayed the actuated dot. They were then actuated in different Braille configurations. Figure 54 below shows the actuators and the LED's displaying the Braille patterns.

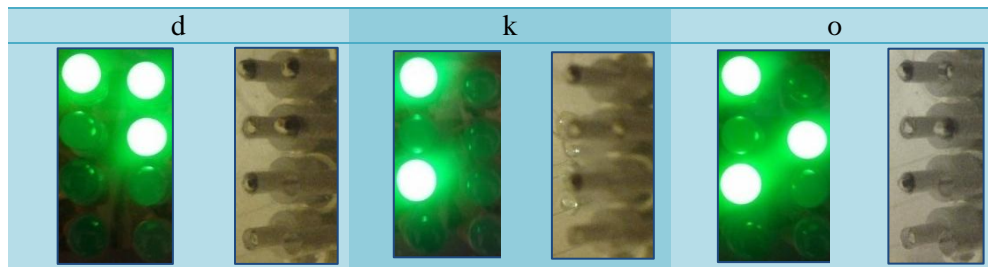


Figure 54 Actuator Test

The actuators when actuated individually functioned well, but when actuated in group had problems. When actuating two dot combinations the actuators didn't have much problem but additional dots added a delay between them. The dots actuated one after another instead of all together. This was because the actuators required extra power at the start than was available to all at once.

The actuators also varied slightly in the amount of actuation so the heights of all the dots were slightly different.

The cooling fan made a considerable difference to the working of the actuators. It decreased the reset times of the actuators and also ensured the actuators didn't get too hot as the Nitinol would sometimes start to smoke when actuated for long periods without the fan.

5.3 EBDR

At first the Braille patterns seemed to be actuating randomly. With detailed inspection of the system it was discovered that the Arduino was scrolling through the patterns too fast for all the actuators to finish actuating. To rectify the problem the scroll speed was slowed down by setting the delay in the Arduino code to 3 seconds. This gave ample time for all the actuators to finishing actuating and the user to read the pattern. Figure 55 below shows the actuators actuating the text from the SD card in Braille pattern.

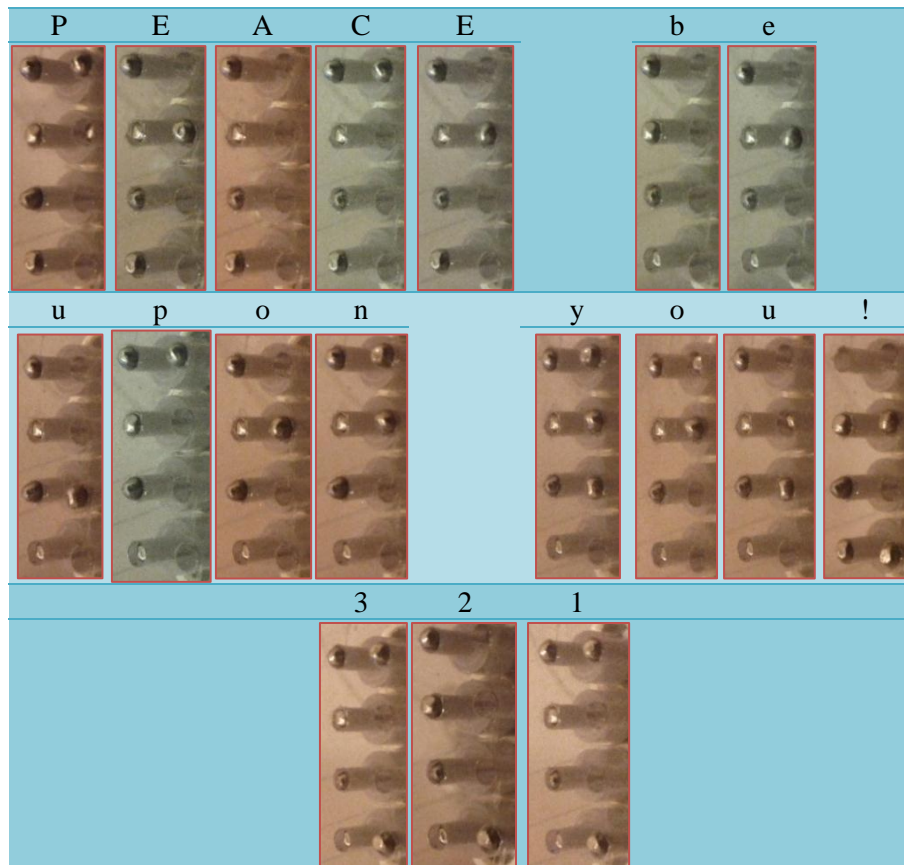


Figure 55 Actuated Braille Text.

The volunteers chosen to test the EBDR read the text by touch alone. They were not able to read at first but after a few minutes getting themselves used to it they were able to decipher the text but at a very slow rate. After about 15 to 20 minutes of practice the speed increased though still a long way from their usual reading speeds.

Random patterns were actuated using the manual switches and the volunteers asked to say the active dots positions to ensure they were not making sense of the text simply by context. Both the volunteers managed to get 50% patterns right.

The Nitinol wire on some of the actuators was tight so the tip of the pin was already protruding a little at the resting position, while on other actuators the wire was loose so the pins were hanging below the resting position and would only protrude a small amount when actuated. This caused a great deal of confusion for the volunteers when determining which dots were active.

The actuators when actuated protruded at different lengths some more than others. The actuators with extra height were thought to be better in lab tests but when tested on volunteers they complained that it was too much and would irritate the finger tip. The Figure 56 shows the difference between two dots.



Figure 56 Actuation difference

6.0 Discussion

The aim of the project to prove the feasibility of a device which would read text from an external memory and actuate it in Braille form to function as an e-book reader for the blind was successfully met. The EBDR proved the feasibility of such a device. The following sections discuss the design, testing and results of the project.

6.1 Text Processing

The decision to choose Arduino for the text processing proved to be the right decision as it made the programming a lot simpler. The Arduino development environment has a vast library of functions and large open source community online. The development of the translation code was simple and without many issues. Interfacing with SD card was also simple in Arduino using SD library readily available online compared to other microcontroller environments.

The Arduino successfully read text which was in Latin characters from an SD card and converted it to a Braille pattern. Using the 8 dot Braille it was able to display the entire alphabet, numbers 0-9 and symbols in a single Braille cell. This negated the need for extra cells used in 6 dot Braille to depict the character information.

The device lacked the usability of an e-book but that would later be implemented. Currently the device could only read one text file saved on the SD card with a predetermined name. There is no way of skipping through the words either. These abilities are easily implemented mostly through software which would require extensive development before the device is ready for public use. The current code proved the core conceptual idea of an e-book reader for the blind

6.2 Actuators

The actuators for the Braille dots are the most important aspect of the whole device. Being the most important they were also the most problematic part of the project. There are many marvellous actuators being developed with the potential to advance electronic Braille devices. The problem faced was none of the actuators suitable for such device were available for experimenting. Most were either still in development or were awaiting investment for mass production.

So until something better was made available alternative methods were required to actuate the Braille dots. Some alternative actuators were researched but they were too expensive to acquire or too large to arrange in a Braille pattern like solenoids. The Nitinol wire was seen as a feasible material to build cheap actuator as it is freely available online at low cost.

The development of the actuators took a long time; needing a lot of experimenting with many different configurations to achieve the required actuation. The building of eight Nitinol actuators in a Braille cell was another complex task requiring many modifications to the design. Prototyping equipment at the university was also a limiting factor as access to technologies like 3D printers would have decreased the time spent making the mounts for the actuators. The technicians in the mechanical laboratory were asked to drill the many holes in the Plexiglas plates which required precise alignments. As helpful as they were it was frustrating having to wait several days for simple alterations because of their busy schedule.

The design of the actuators had the wire held in place by feeding it through a hole, then a bolt being screwed in it from where it went in. When the actuators were used repeatedly, the wire started to stretch and would require tightening. To tighten the wire the screw was removed and the wire pulled through the other side of the hole until it was tight and the screw tightened back into the hole. The thin wire was very hard to keep hold of and pull as it would slip often while adjusting. This was a major problem as the Nitinol wire required a lot of adjustment and often. Although the actuators weren't the best and had many problems they fulfilled the requirements of the EBDR by actuating the Braille pattern.

The biggest problem encountered with the Nitinol wire actuators when testing was inconsistency. The actuators actuated with different lengths to one another which caused confusion while reading. Due to the amount of actuation depending on the length of the wire, the tension of the spring and the angle of the V, it was near impossible to match identically for all eight actuators.

The Nitinol actuators were assembled ensuring the wire was tight and pushing against the spring. But when power was applied to them the first time to activate the actuators, the wire stretched slightly and with some actuators the wire became loose and required tightening. The loosening of the wire was a problem encountered again after some use. When the actuators were left actuated for a long time the wire would get too hot and the opposing tension on the springs would stretch it. The cooling fan minimized this problem, but after constant testing the wire would eventually stretch enough that it was too loose against the spring, so wouldn't actuate the required amount or not at all in some cases. When the wire stretched it reduced in thickness and the constant stretching of the wire caused it to be too thin, which eventually would result in the wire breaking when actuated. As the stretching of the wire and the need for tightening of the wire wasn't taken into account in the designing of the actuator it was a complicated and tedious process, which meant the tension on the wire wasn't identical or constant. This was another factor in causing the wire to break. After some use, the actuators also lost performance and required readjustments or replacement Nitinol. Therefore the numerous limiting factors of the Nitinol wire restrict the use of Nitinol for Braille actuators.

6.3 EBDR

The two module design of the EBDR made the designing, building and testing simple. The modules were tested individually and once both were ready they were brought together as a single unit. The two modules were interfaced using wires with pins which would plug into the connectors on the Arduino thus were easy to attach and remove.

The Arduino read and processed the text as required without problems and outputted the required Braille pattern which was passed over to the actuators. The Nitinol wire actuators actuated the Braille pins as required. But they had flaws which prevented the EBDR to perform as expected. The actuators were bigger and used more power than would be desired for the final device but were suitable for prototype testing.

Identifying blind individuals who would be able to test the device was surprisingly hard. The blind students at the university did not want to participate at all, even before being told about the details. But two individuals, one of whom used to be blind for 8 years but now is partially sighted agreed to take part and were enthusiastically looking forward to testing the device when told the details.

The volunteers had difficulty reading at start and were sliding the finger on the Braille cell to read. They had to be told to make sure not to move the finger to read as that would void the results and they happily obliged and resisted moving the finger. A strip of blue tack was stuck on both sides of the Braille cell to help the user keep the finger in place and avoid sliding around. After some practice familiarising themselves with the new system of reading without sliding the finger on Braille cells, they were able to make sense of the patterns being actuated.

The differences in the actuators made the reading by the volunteers harder as they had to take into account the actuator which wasn't resetting low enough and would seem like an active dot. Another issue the volunteers complained about was the height of some of the actuators was too much and was irritating. The actuators were the main problem encountered through the project.

Setting the problems with the actuators aside the ability to read with the Braille pattern actuating on the finger tip was tested. Tests proved that reading Braille on one finger without moving is possible but because of the major difference between this and how usual Braille is read the users would require some practice before they could read proficiently at a reasonable pace. Once a user can train themselves to expect the individual dots touching the finger vertically instead of the finger sliding over them horizontally, they would be able to read properly. As most users would already be Braille users with finger tips sensitive to small detail it wouldn't take long for one to adapt to the new system. Both volunteers enjoyed testing the EBDR and said the need to learn to adapt to a new different reading method are outweighed by the advantages gained by the EBDR. They expressed an urgent need for such device to facilitate reading in Braille.

7.0 Conclusion

A prototype of an Electronic Braille Document Reader was developed capable of reading text in Latin form from an SD card and actuating the text in Braille form which could be read by a blind person by placing the finger on the Braille cell.

The EBDR proved the theory that Braille could be read from a single Braille cell by the patterns actuating on the finger instead of the finger sliding across an already formed Braille pattern. This allows the EBDR to give the blind the ability to take away any electronic text and read on the move. The SD card can save many books and documents. The actuators can actuate the text in Braille form for users to read. due to the actuators being the biggest cost, the single cell design of the EBDR also means that the overall cost of the device would be a considerably less than the likes of refreshable Braille displays which require many cells to display line of text. It could give the blind access to unlimited literature at a reasonable price of a couple of hundred pounds. The benefits gained would transform reading for the blind and encourage among the young.

Further development would be required to implement greater usability functions mainly in the software and suitable actuators sourced before the device would be ready for common use. But the future seems bright for Braille from all the research conducted for the project. Its evident the need for such devices will drive development and a device to read books in Braille would be available in the near future. Whether it be the device proposed in this report or another novel design is a question to which the answer only the future holds.

The project has been a great learning experience with a great deal of knowledge acquired. It has given a great insight into the research and development process used to develop everyday gadgets, and has provided great appreciation for the engineers working to design and perfect them.

8.0 Further Work

The work done thus far has been enough to prove the concept. To develop it to a usable device further work is required.

The code needs further development to incorporate extra functionality, such as being able to scroll through a list of books and select one to open and read. A book marking feature; to save the position in a book would also be required. Other features like being able to skip words, lines or paragraph and adjusting the speed of reading would also be crucial to the usability of the device. Further research and testing would need to be carried out with blind users, to determine what would be the best and most intuitive ways of navigating through the devices menus.

The actuators require a lot of work as the actuators used in the prototype, are not suitable for Braille use. Better actuators either need to be developed or sourced from the current researcher working on the suitable ones.

The EBDR as a whole needs more work for it to be ergonomically comfortable for users to use for long term reading.

The whole design of the EBDR could also be changed. A design change to a number of Braille cells in a row assembled on a conveyor belt system could prove better. Each cell would actuate to form a Braille character and the conveyor would slide it beneath the users' finger. This would avoid the need for the user to learn a new way of reading Braille as that may put people off from taking up such devices. Another benefit this system would have over the other system is the user could use different fingers to read. While testing the EBDR with the blind volunteers one of them mentioned that after sometime spent reading even the paper Braille the finger gets tired and they move to a different finger. The conveyor design would make it easy for the user to change the reading finger. Figure 57 below shows a concept drawing of the conveyor design.

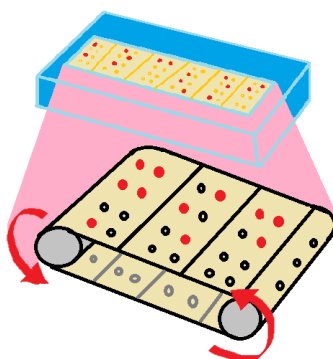


Figure 57 Braille on a Conveyor

The conveyor design would negate some of the downsides of the single Braille design while still providing the same benefits. It would still give the blind access to limitless literature and the freedom to carry it around with them. The additional cells and apparatus for the conveyor would increase the cost, but the additional cells would also mean the user can easily pause the text and slide the finger back and forth to re-read a word if required; making it easy to go back to the start of the word. This new development in the design would require extensive testing and comparisons made to the previous model to see which would be best suited to the needs of the users.

9.0 References

- [1] R. Freedman, *Out of Darkness: The Story of Louis Braille*. Sandpiper, 1999.
- [2] “Louis Braille: 6 dot Braille Alphabet, Numbers, Punctuation & Symbols.” [Online]. Available: <http://6dotbraille.com/>. [Accessed: 07-Sep-2012].
- [3] “Unified 8 dot Braille Code: Alphabet, Numbers, Punctuation & Symbols.” [Online]. Available: <http://8dotbraille.com/>. [Accessed: 07-Sep-2012].
- [4] J. Roberts, O. Slattery, and D. Kardos, “49.2: Rotating-Wheel Braille Display for Continuous Refreshable Braille,” in *SID Symposium Digest of Technical Papers*, 2012, vol. 31, pp. 1130–1133.
- [5] “Why Braille is brilliant,” *BBC*, 02-Jan-2009.
- [6] W. Michael, V. Gnanakumaran, and D. Goldreich, “Tactile Spatial Acuity Enhancement in Blindness: Evidence for Experience-Dependent Mechanisms,” *The Journal of Neuroscience*, p. 10, May 2011.
- [7] M. Herron, “Blind visionary - [engineering heritage],” *Engineering Technology*, vol. 4, no. 8, pp. 84–85, May 2009.
- [8] “What is Braille?” [Online]. Available: <http://www.brailleenterprises.com/whatis.htm>. [Accessed: 07-Sep-2012].
- [9] “Eight-dot-braille.” [Online]. Available: <http://www.brailleauthority.org/eightdot/eightdot.html>. [Accessed: 23-Aug-2012].
- [10] “Grade Two Braille Contractions.” [Online]. Available: <http://www.99main.com/~charlief/brl/brl2.htm>. [Accessed: 07-Sep-2012].
- [11] “Contracted (grade 2) braille explained - RNIB.” [Online]. Available: http://www.rnib.org.uk/livingwithsightloss/readingwriting/braille/braille/codes/Pages/contracted_braille.aspx. [Accessed: 07-Sep-2012].
- [12] “Bharati Braille - Acharya.” [Online]. Available: http://acharya.iitm.ac.in/disabilities/bh_brl_discuss.php. [Accessed: 27-Aug-2012].
- [13] H. Joshi, “Braille - In India.” [Online]. Available: <http://www.bpaindia.org/VIB%20Chapter-VI.pdf>.
- [14] S. C. Mackenzie, “World Braille Usage,” *UNESCO*.
- [15] *The Quran in Arabic Braille [Al-Fatihah]*. 2011.
- [16] M. Bijani, “Reading The Braille Quran.” [Online]. Available: <http://www.bpcprograms.com/munawar/quran/#analysis>. [Accessed: 27-Aug-2012].
- [17] “Chinese Alphabet - Is there such a thing?” [Online]. Available: <http://www.orientaloutpost.com/chinese-letters.php>. [Accessed: 27-Aug-2012].
- [18] “Braille for Chinese.” [Online]. Available: http://www.omniglot.com/writing/braille_chinese.htm.
- [19] “New Focus 40 Blue Wireless Braille Display for the Visually Impaired by Freedom Scientific.” [Online]. Available: <http://www.freedomscientific.com/products/fs/focus-40-blue-new-product-page.asp>. [Accessed: 29-Aug-2012].
- [20] RNIB, “2011_06_use_of_braille_displays.doc.” [Online]. Available: www.rnib.org.uk/aboutus/.../2011_06_use_of_braille_displays.doc. [Accessed: 28-Aug-2012].
- [21] Cedrat, “Bi-stable Linear Moving Magnet BLMM.” [Online]. Available: http://www.cedrat.com/fileadmin/user_upload/cedrat_groupe/Technologies/Actuators/Magnetic%20actuators%20%26%20motors/fiche_BLMM/Bi-stableLinear_Moving_Magnet_BLMM_01.pdf.
- [22] New Scale Technologies, “SQUIGGLE micro motors for OEMs - SQL-1.8,” http://www.newscaletech.com/doc_downloads/Design-World-Mechatronics-Article-10-2010.pdf. [Online]. Available: <http://www.newscaletech.com/motorsforoem.html>. [Accessed: 31-Aug-2012].
- [23] J. T. Leinvu, S. A. Wilson, and R. N. Whatmore, “Flextensional ultrasonic motor using the contour mode of a square piezoelectric plate,” *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 51, no. 8, pp. 929–936, 2004.
- [24] “Ultrasonic Motor & Micromotor-Flexmotor.” [Online]. Available: http://www.flexmotor.com/page2_mm.htm. [Accessed: 04-Sep-2012].
- [25] “core.form-ula Shape Memory Alloy.” [Online]. Available: <http://www.core.form-ula.com/page/23/?s=rem>. [Accessed: 06-Sep-2012].
- [26] J. Iovine, “TALKING ELECTRONICS Nitinol.” [Online]. Available: <http://talkingelectronics.com/FreeProjects/Nitinol/Nitinol-2.html>. [Accessed: 06-Sep-2012].
- [27] DYNALLOY, “TechnicalCharacteristics Flexinol.pdf.”

- [28] Y. Kato, T. Sekitani, M. Takamiya, M. Doi, K. Asaka, T. Sakurai, and T. Someya, "Sheet-Type Braille Displays by Integrating Organic Field-Effect Transistors and Polymeric Actuators," *IEEE Transactions on Electron Devices*, vol. 54, no. 2, pp. 202–209, Feb. 2007.
- [29] Kato, S. Iba, T. Sekitani, Y. Noguchi, K. Hizu, K. Takenoshita, Y. Takamatsu, S. Nakano, K. Fukuda, K. Nakamura, T. Yamaue, K. Asaka, H. Kawaguchi, M. Takamiya, T. Sakurai, and T. Someya, "A flexible, lightweight braille sheet display with plastic actuators driven by an organic field-effect transistor active matrix," *IEEE International Electron Devices Meeting 2005 IEDM Technical Digest*, vol. 00, no. c, pp. 97–100, 2005.
- [30] T. Sugino and K. Asaka, "Fully plastic actuators based on ionic-liquid-based bucky-gels and their applications to Braille display," in *SICE Annual Conference (SICE), 2011 Proceedings of*, 2011, pp. 1696–1697.
- [31] K. Mukai, K. Asaka, K. Kiyohara, T. Sugino, I. Takeuchi, T. Fukushima, and T. Aida, "High performance fully plastic actuator based on ionic-liquid-based bucky gel," *Electrochimica Acta*, vol. 53, no. 17, pp. 5555–5562, Jul. 2008.
- [32] "Micro SD card Tutorial - using SD cards with an Arduino!" [Online]. Available: <http://www.ladyada.net/products/microsd/>. [Accessed: 12-Aug-2012].

10 Appendices

A. Arduino code

```
#include <SD.h> // SD Library

const int intChipSelectpin = 53; // Chipselect for Arduino 2560(mega) pin 53

// File on the SD Card
File myFile;           // the file object
const String strFileName = "test.txt"; // Name of the file on the SD card
String strLineRead;     // the string read from the strFileName file

const boolean SerialReporting = true; // turn serial reporting on/off true/false

boolean pattern[72][8]={
    // Lowercase
    /*a*/ {1,0,0,0,0,0,0,0}, /*b*/ {1,1,0,0,0,0,0,0}, /*c*/ {1,0,0,0,1,0,0,0}, /*d*/ {1,0,0,0,1,1,0,0},
    /*e*/ {1,0,0,0,0,1,0,0}, /*f*/ {1,1,0,0,1,0,0,0}, /*g*/ {1,1,0,0,1,1,0,0}, /*h*/ {1,1,0,0,0,1,0,0},
    /*i*/ {0,1,0,0,1,0,0,0}, /*j*/ {0,1,0,0,1,1,0,0}, /*k*/ {1,0,1,0,0,0,0,0}, /*l*/ {1,1,1,0,0,0,0,0},
    /*m*/ {1,0,1,0,1,0,0,0}, /*n*/ {1,0,1,0,1,1,0,0}, /*o*/ {1,0,1,0,0,1,0,0}, /*p*/ {1,1,1,0,1,0,0,0},
    /*q*/ {1,1,1,0,1,1,0,0}, /*r*/ {1,1,1,0,0,1,0,0}, /*s*/ {0,1,1,0,1,0,0,0}, /*t*/ {0,1,1,0,1,1,0,0},
    /*u*/ {1,0,1,0,0,0,1,0}, /*v*/ {1,1,1,0,0,0,1,0}, /*w*/ {0,1,0,0,1,1,1,0}, /*x*/ {1,0,1,0,1,0,1,0},
    /*y*/ {1,0,1,0,1,1,1,0}, /*z*/ {1,0,1,0,0,1,1,0},

    // Uppercase
    /*A*/ {1,0,0,1,0,0,0,0}, /*B*/ {1,1,0,1,0,0,0,0}, /*C*/ {1,0,0,1,1,0,0,0}, /*D*/ {1,0,0,1,1,1,0,0},
    /*E*/ {1,0,0,1,0,1,0,0}, /*F*/ {1,1,0,1,1,0,0,0}, /*G*/ {1,1,0,1,1,1,0,0}, /*H*/ {1,1,0,1,0,1,0,0},
    /*I*/ {0,1,0,1,1,0,0,0}, /*J*/ {0,1,0,1,1,1,0,0}, /*K*/ {1,0,1,1,0,0,0,0}, /*L*/ {1,1,1,1,0,0,0,0},
    /*M*/ {1,0,1,1,1,0,0,0}, /*N*/ {1,0,1,1,1,1,0,0}, /*O*/ {1,0,1,1,0,1,0,0}, /*P*/ {1,1,1,1,1,0,0,0},
    /*Q*/ {1,1,1,1,1,1,0,0}, /*R*/ {1,1,1,1,0,1,0,0}, /*S*/ {0,1,1,1,1,0,0,0}, /*T*/ {0,1,1,1,1,1,0,0},
    /*U*/ {1,0,1,1,0,0,1,0}, /*V*/ {1,1,1,1,0,0,1,0}, /*W*/ {0,1,0,1,1,1,1,0}, /*X*/ {1,0,1,1,1,0,1,0},
    /*Y*/ {1,0,1,1,1,1,1,0}, /*Z*/ {1,0,1,1,0,1,1,0},

    // Numbers
    /*1*/ {1,0,0,0,0,0,0,1}, /*2*/ {1,1,0,0,0,0,0,1}, /*3*/ {1,0,0,0,1,0,0,1}, /*4*/ {1,0,0,0,1,1,0,1},
    /*5*/ {1,0,0,0,0,1,0,1}, /*6*/ {1,1,0,0,1,0,0,1}, /*7*/ {1,1,0,0,1,1,0,1}, /*8*/ {1,1,0,0,0,1,0,1},
    /*9*/ {0,1,0,0,1,0,0,1}, /*0*/ {0,1,0,0,1,1,0,1},

    // Symbols
    /**/ {0,0,1,1,0,0,0,1}, /*,*/ {0,0,1,1,0,0,0,1}, /*./*/ {0,0,1,1,0,0,0,1}, /*:*/ {0,1,1,1,0,0,0,1},
    /*!*/ {0,1,1,1,0,1,0,1}, /*?*/ {0,1,1,1,0,0,1,1}, /*""*/ {0,0,1,1,0,1,1,1}, /*(* */ {0,1,1,1,0,1,1,1},
    /*)*/ {0,1,1,1,0,1,1,1}, /*-*/ {0,0,1,1,0,0,1,1}, /*space*/ {0,0,0,0,0,0,0,0}
};

char alphabet[72]={ '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',
    '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',
    '!',',',';',':','?','"','(',')','-'',' '
};

String input = "TeSt Data 1823"; //the text to be converted

int letter = 0; //reference to the current letter (the current row in the pattern array)
```

```

void setup() {

    for(int p=2; p<=9; p++){ //Loop to go through pin 2 till 9
        pinMode(p, OUTPUT); //declare pins 2 to 9 be output
    }

    if (SerialReporting) { // If Serial port monitoring is enabled
        Serial.begin(9600); //set up Serial library at 9600 bps
        Serial.println("Just began..."); //print out

        boolean Success; // Success of operations

        Success = InitializeSDCard(); // Initialize SD Card

        if (Success) { // If successful, then read card
            Success = ReadFileFromSDCard(); // Read the text from file strFileName into
the strLineRead string
        }

        if(SerialReporting && !Success) Serial.println("Card Reading stopped with errors!"); // If it fails,
report stop and don't continue

//with the loop() function
        input = strLineRead; //update input string with data from SD
    } }

void loop() {

    for(int dot=2; dot<=9; dot++) { //Loop to move through pin 2 to 9
        int x=0;
        while(alphabet[x] != input.charAt(letter)) { x++; } //Go through input and compare with
alphabet to look for same characters
        digitalWrite(dot, pattern[x][dot-2]); //Use position of the character in alphabet to determine
the position of Braille pattern
    }

    if(letter==input.length()-1) { //if position is at end of input text the reset position
        letter=0;
    } else {
        letter++; //Otherwise keep incrementing position
    }

    delay(4000); //delay of 2 seconds
}

// SD CARD FUNCTIONS
boolean InitializeSDCard(void) {
    // Initializes the SD Card and reports any problems
    // or success through the Serial port
    //=====

    if(SerialReporting) Serial.println("Initializing SD card..."); //If enabled, report via Serial Port

    // must be left as an output or the SD library functions will not work
    pinMode(SS, OUTPUT);

```

```

    if (!SD.begin(intChipSelectpin)) {
        if(SerialReporting) Serial.println("Initialization failed!"); //If enabled, report Error via Serial
Port
        return false; // report failure
    }

    if(SerialReporting) Serial.println("Initialization done."); //If enabled, report Success via Serial Port:
    return true; // report success
}
boolean ReadFileFromSDCard(void) {
    // Reads File from file strFileName
    //=====

    strLineRead = "";          // Clear the string that receives the text in the file
    char chrRead;

    char charFNameBuf[50];      // char array used to be able to pass the file name to the SD.open()
function from strFile
    strFileName.toCharArray(charFNameBuf, 50);

    myFile = SD.open(charFNameBuf);    // Open the file for reading:

    if (myFile) {
        if(SerialReporting) Serial.println("File " + strFileName + " opened successfully");
        //report Success via Serial Port
        while (myFile.available()) { // read from the file until there's nothing else in it
            chrRead = myFile.read(); // Read the character as a number equivalent to its ASCII
code
            strLineRead.concat(chrRead); // Concatenate the character read into the final string
        }
        myFile.close(); // close the file:

        //print string read to serial port
        if(SerialReporting) {
            Serial.println("String read:");
            Serial.println(strLineRead);
            Serial.println();
        }
        return true;
    }
    else {
        //report Error via Serial Port
        if(SerialReporting) Serial.println("Error opening file " + strFileName );
        return false;
    }
}

```

B. Email Conversation with Flexi motors

RE: Uni of Huddersfield - piezoelectric micromotor

[Shahab Arif U0757852](#)

Sent: 28 October 2011 16:46

To: [Dorey, Rob \[r.a.dorey@cranfield.ac.uk\]](mailto:r.a.dorey@cranfield.ac.uk)

Dear Robert

I am disappointed to see such a good product shelved, it would have been very useful. I hope it gets taken into production one day soon.

Thank you for your help.

Regards

Shahab Arif

From: Dorey, Rob [r.a.dorey@cranfield.ac.uk]

Sent: 27 October 2011 12:41

To: Shahab Arif U0757852

Subject: RE: Uni of Huddersfield - piezoelectric micromotor

Dear Shahab Arif,

I'm sorry to say that at this stage there are no definitive plans to put them into production. Unfortunately the project that developed them has now finished so we're not even making them in the lab at this stage.

All the best

Robert

Professor Robert Dorey CSci CEng FIMMM FHEA
Chair in Nanomaterials
Head of Microsystems & Nanotechnology Centre

Microsystems & Nanotechnology Centre | Materials Department
Building 70 | Cranfield University | Cranfield | Bedfordshire | UK | MK43 0AL
T: +44 (0)1234 750111 ext 2726 | F: +44 (0)1234 751346
www.cranfield.ac.uk

From: Shahab Arif U0757852 [<mailto:U0757852@hud.ac.uk>]

Sent: 26 October 2011 12:53

To: Dorey, Rob

Subject: RE: Uni of Huddersfield - piezoelectric micromotor

Dear Robert

Thank you very much. Have you got any idea when they will be in production? Or if it's possible for me to run some tests on them to prove the feasibility? As

the specs for MM-2mm piezoelectric micromotors best suit my requirements and I'm finding it hard to find anything else suitable.

Regards
Shahab Arif

From: Dorey, Rob [<mailto:r.a.dorey@cranfield.ac.uk>]
Sent: 25 October 2011 16:41
To: Shahab Arif U0757852
Cc: Giaracuni, Enza
Subject: RE: Uni of Huddersfield - piezoelectric micromotor

Dear Shahab,

Dr Wilson is no longer with Cranfield so your message has been passed on to me.

Unfortunately the micromotors are still not in production so I am unable to send you any samples

I wish you all the best with your Masters
All the best

Professor Robert Dorey CSci CEng FIMMM FHEA
Chair in Nanomaterials
Head of Microsystems & Nanotechnology Centre

Microsystems & Nanotechnology Centre | Materials Department
Building 70 | Cranfield University | Cranfield | Bedfordshire | UK | MK43 0AL
T: +44 (0)1234 750111 ext 2726 | F: +44 (0)1234 751346
www.cranfield.ac.uk<<http://www.cranfield.ac.uk>>

From: Giaracuni, Enza
Sent: 24 October 2011 12:56
To: Dorey, Rob
Subject: FW: Uni of Huddersfield - piezoelectric micromotor
From: Shahab Arif U0757852 [<mailto:U0757852@hud.ac.uk>]<[mailto:\[mailto:U0757852@hud.ac.uk\]](mailto:[mailto:U0757852@hud.ac.uk])>
Sent: 24 October 2011 12:40
To: Giaracuni, Enza
Subject: Uni of Huddersfield - piezoelectric micromotor

Dear Enza Giaracuni

My name is Shahab Arif and I am doing a Masters by research in Electronic Engineering at the University of Huddersfield. I contacted Dr. Steve Wilson last year in regards to the MM-2mm piezoelectric micromotor for my project but they were not in production at the time. I have taken on the project as my masters by research project this year and require the motors. Are the micromotors in production now? And is it possible for me to get a few samples to test?

Regards
Shahab Arif

C. EBDR Dimensions

