DESIGN AND IMPLEMENTATION OF TEXT TO BRAILLE CONVERTER

Major Project Report Submitted in partial fulfilment

Of the requirement for the undergraduate degree of

Bachelor of Technology

In

Electronics and Communication Engineering

By

Guggilam Lohitha Lakshmi – 221710402015
Uppula Soundharya Lahari – 221710402062
Turlapati Bharadwaj – 221710402061
K Vishnu Kalyan Chakravarthi – 221710402022

Under the Guidance of

Dr. P. Trinatha Rao
Professor
Dept. of EECE

Department of Electrical, Electronics and Communication Engineering
GITAM School of Technology
GITAM Deemed to be University
Hyderabad Campus -502329
April 2021

DECLARATION

We submit this Mini Project work entitled **"DESIGN AND IMPLEMENTATION OF TEXT TO BRAILLE CONVERTER"** to GITAM School of Technology, GITAM Deemed to be University, Hyderabad campus in partial fulfilment of the requirements for the award of the degree of "Bachelor of Technology" in **"Electronics and Communication Engineering".**

We declare that it was carried out independently by me under the guidance of Dr. P. Trinatha Rao, Professor, GITAM School of Technology, GITAM (Deemed to be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD
Date: 30-04-2021

Guggilam Lohitha Lakshmi – 221710402015
Uppula Soundharya Lahari – 221710402062
Turlapati Bharadwaj – 221710402061
K Vishnu Kalyan Chakravarthi – 221710402022

GITAM School of Technology
GITAM Deemed to be University
Hyderabad Campus-502329, India
Date: 30-04-2021

CERTIFICATE

This is to certify that the Major Project Report entitled "**DESIGN AND IMPLEMENTATION OF TEXT TO BRAILLE CONVERTER**" is submitted by **Guggilam Lohitha Lakshmi (221710402015), Uppula Soundharya Lahari (221710402062), Turlapati Bharadwaj (221710402061) and K Vishnu Kalyan Chakravarthi (221710402022)** in partial fulfilment of the requirement for the award of Bachelor of Technology in **Electronics and Communication Engineering** at GITAM (Deemed to be University), Hyderabad.

It is faithful record work carried out by her at the Electrical, **Electronics & Communication Engineering Department**, GITAM School of Technology, GITAM Deemed to be University, Hyderabad Campus under my guidance and supervision.

Dr. P. Trinatha Rao 30/05/2021
Professor
Department of EECE

Dr. K. Manjunathachari
Professor and HOD
Department of EECE

ACKNOWLEDGMENT

We wish to take this opportunity to express my sincere gratitude to all those who helped, encouraged, motivated and have extended their cooperation in various ways during my training program.

We would like to thank Respected Prof. N. Siva Prasad, Pro Vice-Chancellor, GITAM (Deemed to be University), Hyderabad and Prof. N. Seetharamaiah, Principal, GITAM School of Technology, GITAM (Deemed to be University), Hyderabad.

We would like to thank respected Dr. K. Manjunathachari, Head of the Department of Electrical, Electronics and Communication Engineering, GITAM (Deemed to be University), Hyderabad, for giving us such an excellent opportunity to expand our knowledge for our branch and giving us guidelines to present the major project report. It helped us a lot to realize what we study.

I would like to thank respected faculty guide Dr. P. Trinatha Rao, Professor, Dept. of EECE, GITAM (Deemed to be University), Hyderabad, who helped me make this major project report a successful accomplishment.

Guggilam Lohitha Lakshmi – 221710402015

Uppula Soundharya Lahari – 221710402062

Turlapati Bharadwaj – 221710402061

K Vishnu Kalyan Chakravarthi – 221710402022

ABSTRACT

The aim is to design a Text to Braille Converter using Arduino. In the present design, Braille for the English language is considered. The Braille system is classified into static and dynamic. A fixed system to represent English 26 alphabets requires static braille dots of size 3x2 matrix. The characters in the form of dots are punched on papers or metal sheets. Static Braille system needs many documents to be carried to read documents. This burden is overcome by using a dynamic braille system.

Braille is a tactile system to represent text. It is not a language but is a code. A text in Braille consists of several braille cell where each cell represents an alphabet or symbol. A single cell consists of 6 individual dots. A combination of raising high and raising low of these pins compose an alphabet. There is a standard braille code for each alphabet of most of the languages. Generally, texts are embossed on paper, and visually impaired persons move finger over the dots to read text.

Braille technology is growing by many folds with the given technical advancements in various fields of science. With Braille code, many languages such as English, French, and Spanish may be written and read. Many people use this code in their native language all over the world, thereby providing literacy to one and all. The hypothesis is to keep the user's fingers rest over the cell and actuate cell pins to make it feel like moving a finger over cells.

TABLE OF CONTENTS

CHAPTER 1 ..... **1**

INTRODUCTION **1**

1.1 INTRODUCTION 1

1.2 HISTORY 2

1.3 BASIC CONCEPTS 4

1.4 ASCII 5

1.5 ASCII TO BRAILLE CONVERSION 6

1.6 COMPLEXITIES OF BRAILLE 8

CHAPTER 2 ..... **10**

LITERATURE SURVEY **10**

CHAPTER 3 ..... **13**

HARDWARE AND SOFTWARE **13**

3.1 HARDWARE 13

3.1.1 ARDUINO *13*

3.1.2 ARDUINO UNO TECHNICAL SPECIFICATIONS *14*

3.1.3 PIN CONFIGURATIONS *15*

3.1.4 ATMEGA328 ..... *16*

3.1.5 PUSH PULL SOLENOIDS *17*

3.1.6 IRFZ44n MOSFET *18*

3.1.7 RESISTORS *18*

3.1.8 DIODE *19*

3.1.9 HC 05 BLUETOOTH MODULE *19*

3.2 SOFTWARE 20

3.2.1 ARDUINO IDE *20*
3.2.2 WRITING SKETCHES *21*
3.2.3 LIBRARIES *21*
3.2.4 SERIAL MONITOR *22*
3.2.5 ANDROID STUDIO *23*
3.2.6 JAVA *24*
3.2.7 SOFTWARE DEVELOPMENT KIT (SDK) *24*

CHAPTER 4 ..... **25**

TESTING OF COMPONENTS **25**

4.1 ARDUINO UNO 25
4.2 MINI PUSH-PULL SOLENOIDS 26
4.3 HC-05 BLUETOOTH MODULE 27
4.4 INTEGRATED TESTING 28

CHAPTER 5 ..... **29**

DESIGN OF THE CIRCUIT & APP **29**

5.1 CIRCUIT DESIGN 29
5.1.1 SOLENOID CIRCUIT *30*
5.1.2 HC-05 BLUETOOTH CIRCUIT *31*
5.2 APP DESIGN 31

CHAPTER 6 ..... **33**

IMPLEMENTATION AND APPLICATION **33**

6.1 IMPLEMENTATION 33
6.2 PROJECT RESULTS 35
6.3 APPLICATIONS 37

6.3.1 ELECTRONIC BRAILLE DISPLAYS.....37

6.3.2 BRAILLE WATCHES 38

6.3.3 BRAILLE PHONES.....39

CONCLUSION 40

FUTURE SCOPE.....41

REFERENCES 42

APPENDIX.....43

LIST OF FIGURES

FIGURE 1.1 GRADE 1 AND GRADE 2 3
FIGURE 1.2 ASCII TABLE 5
FIGURE 1.3 BRAILLE FROM VARIOUS SOURCES 6
FIGURE 1.4 MATRIX REPRSENTATION 7
FIGURE 1.5 BRAILLE OF A 7
FIGURE 1.6 PRINT GRADE 2 COMPUTER BRAILLE 8
FIGURE 3.1 ARDUINO UNO 13
FIGURE 3.2 ATMEGA 328 PINOUT 16
FIGURE 3.3 SOLENOIDS 17
FIGURE 3.4 IRFZ44N 18
FIGURE 3.5 1N4007 DIODE 19
FIGURE 3.6 HC 05 BLUETOOTH MODULE 19
FIGURE 3.7 ARDUINO IDE 20
FIGURE 3.8 SERIAL MONITOR 22
FIGURE 3.9 ANDROID STUDIO 23
FIGURE 3.10 SDK 24
FIGURE 4.1 ARDUINO TESTING 25
FIGURE 4.2 SOLENOID TESTING 26
FIGURE 4.3 HC 05 TESTING 27
FIGURE 4.4 INTEGRATED TESTING 28
FIGURE 4.5 SOLENOID INTEGRATED TESTING 28
FIGURE 5.1 CIRCUIT DESIGN 29
FIGURE 5.2 SOLENOID CIRCUIT 30
FIGURE 5.3 BLUETOOTH CONNECTION 31
FIGURE 5.4 APP DESIGN 32

FIGURE 6.1 BASIC ASCII TO BRAILLE CHART 33
FIGURE 6.2 CIRCUIT IMPLEMENTATION 34
FIGURE 6.3 TEST RESULT 35
FIGURE 6.4 APP BLUETOOTH CONNECTION 35
FIGURE 6.5 ENTER TEXT IN APP 36
FIGURE 6.6 BRAILLE RESULTS 36
FIGURE 6.7 ELECTRONIC BRAILLE DISPLAY 37
FIGURE 6.8 BRAILLE WATCH 38
FIGURE 6.9 BRAILLE SMARTPHONE 39

ABBREVATIONS

1. AT : Assistive Technology
2. ASCII : American Standard Code for Information Interchange
3. AVR : Advanced Virtual RISC
4. CPU : Control Processing Unit
5. EEPROM: Electrically Erasable Programmable Read Only Memory
6. GND : Ground
7. GNU : GNU's Not UNIX.
8. GPI : General Public License
9. IDE : Integrated Development Environment
10. IEEE : Institute of Electrical and Electronics Engineers
11. JDK : Java Development Kit
12. JVM : Java Virtual Machine
13. LGPL : Lesser General Public License
14. MCU : Microcontroller
15. MEMS : Micro Electro Mechanical Systems
16. MLF : Micro Lead Frame
17. MOSFET: Metal Oxide Semiconductor Field Effect Transistor
18. NCBI : National Center for Biotechnology Information
19. NLP : Natural Language Processing
20. OS : Operating System
21. QFN : Quad Flat No-leads
22. PAN : Personal Area Network
23. PMW : Pulse Width Modulation
24. RISC : Reduced Instruction Set Computer
25. SDK : Software Development Kit

26.SRAM : Static Random Access Memory

27.TQFP : Thin Quad Flat Package

28.TTL : Transistor - Transistor Logic

29.UI : User Interface

30.USART : Universal Synchronous Asynchronous Receiver Transmitter

31.USB : Universal Serial Bus

32.WHO : World Health Organization

33.WORA : Write Once Run Anywhere

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Communication is a process of exchanging information. Writing is one of the basic skills used in communication to express their thoughts and feelings. Languages have different writing systems, orthography and syntax. Sighted people read visually, while blind people read through touch. In the early 18th century, the philosopher Bishop Berkeley assumed that the basis of perception was touch, with the hand being used as a unitary sense organ like the eye (Katz, 1925). Charles Barbier developed the first dot for tactile perception. Raised dots were designed to be read by touch in the dark. Later, Louis Braille learned and modified the system to be more suitable for sight loss people. His version was more logical and economical. Braille's version is now used widely for reading and writing by people with visual impairments. Braille characters originally consisted of six dots, but the encoding has since been extended to eight dots. The Royal Institution officially adopted the Braille code for Blind Youth in 1854. The Missouri School for the Blind in the United States was the first institution to adopt the code. According to the latest statistical data on the magnitude of blindness and visual impairment from the World Health Organization (WHO), more than 161 million people worldwide were visually impaired, of whom 124 million people had low vision, and 37 million were blind. Many of them rely on Braille as a tool of learning and communication. Although Braille dots still do not resemble print letters, Braille has been adapted to almost every language globally and remains the major medium of literacy for blind people everywhere. Since Louis Braille published his first embossed Braille book in 1829, millions of books have been published in Braille for people with visual impairment. Recently, Braille is suffering from decreasing popularity due to the use of alternative technologies, like speech synthesis. However, as a form of literacy, Braille is still playing a significant role in educating people with visual impairments. On the other hand, reading straight from the text can avoid potential errors or problems like indecipherable meanings or misspelling caused by speech synthesis. Therefore, Braille should still be a critical part of blind education and culture.

Automatic Braille Translation is a popular topic in the AT of visual impairment and has been wildly discussed and analyzed since the 1960s. Currently, there are some commercially available programs and other computer-assisted applications which specialize in Braille translation. Some of these use personal computers to achieve translation and other functions, such as Duxbury, the most popular multi-language Braille translation software. In this case, the translation speed tends to be strongly related to particular computers utilized in the process. This kind of software is mainly designed for users with normal eyesight, so it has low accessibility for two visually impaired users and is used by transcribers translating existing print texts. They have to read the computer screen using some AT tools, such as screen reading software or Braille displays.

1.2 HISTORY

The history of Braille can be traced back to the 1800s. The following text illustrates the origins of Braille, and it was published on the website of the Duxbury Systems Company. It was a French army captain, Charles Barbier de la Serre, who invented the basic technique of using raised dots for tactile writing and reading. His original objective was to allow soldiers to compose and read messages at night without illumination. Barbier later adapted the system and presented it to the Institution for Blind Youth, hoping it would be officially adopted there. He called the system Sonography because it represented words according to sound rather than spelling. However, Barbier's system was too complex for soldiers to learn. Based on Barbier’s system, in 1821, the young Frenchman, Louis Braille, developed the system known as Braille, widely used by blind people to read and write. Each Braille character or "cell" comprises six dot positions arranged in a rectangle containing two columns of 3 dots each. A dot may be raised at any of the six positions, so, counting the space in which no dot is raised, there are 64 such combinations in total (that is, 2 to the power of 6). There is no intentional relation between the arrangement of dots in a cell and the shape of the corresponding ink-print character.

The original Braille is a quite straightforward dots system which includes characters for each letter of the alphabet, punctuation marks, and numerals. This corresponds to Grade 1 Braille, where there is nearly a one-to-one correspondence between letters and Braille cells. Although it is clearly easy to transcribe Braille by simply substituting the equivalent Braille character for its printed equivalent using Grade 1 Braille, such a character-by character transcription is used only by beginner and the process is significantly time-consuming. On the other hand, the size of the Braille cell is such that only about 25 lines of about 40 cells each, that is 1000 characters, can fit on a page of the usual size, which is the size of A4 page. This contrasts with the 3500 or so characters that will fit on a standard, smaller, typed page. Moreover, Braille paper must be much heavier to hold the dots, and the dots themselves considerably increase 6 the effective thickness of a page. The result is that paper Braille is very bulky. In English-language Braille, for which 189 contractions have been developed, almost all Braille is written in Grade 2. Therefore, many Braille cells have multiple meanings.

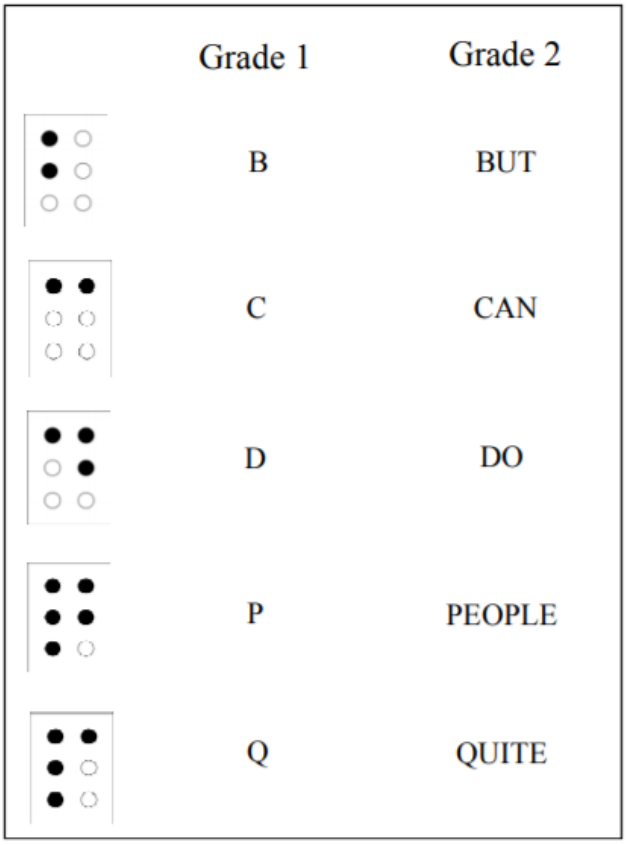

Figure 1.1 Grade 1 and Grade 2

Grade 2 Braille can contain more information, and therefore it can be read and produced much faster than Grade 1 Braille. Grade 2 Braille, although an effective way for the blind to learn and communicate, had not been accepted and utilised widely. One reason is Grade 2 Braille has complex rules about how to use contractions, and also because translation between Braille and text was very time-consuming and expensive. But with the development of digital technology and innovations in Braille education, Grade 2 Braille has been accepted as one of most important ways for the blind to learn and communicate.

1.3 BASIC CONCEPTS

When translation between text and Braille is considered, there are two kinds of alphabets involved. One is the alphabet of Latin letters, Arabic numerals, punctuation marks, and some special symbols. The other includes 64 Braille codes.

However, instead of using dot representations, there is a subset of the American Standard Code for Information Interchange (ASCII) character set which uses 64 of the printable ASCII characters to represent all possible six-dot combinations of Braille. This subset is called the Braille ASCII set, or computer Braille set.

Braille ASCII uses the 64 ASCII characters between 32 and 95 inclusive. All capital letters in ASCII correspond to their equivalent values in Braille. Unlike standard print, all letters in Braille are lower-case by default, unless otherwise specified by preceding them with a capitalisation symbol. Therefore, the translation process can be regarded as a symbol transformation from one kind of ASCII alphabet set into the other.

The two alphabets can be defined as follows:

- A finite, nonempty set ∑1 of Latin letters, Arabic numerals, punctuation marks, and special marks is called
  Alphabet 1: ∑1= {A, B, C, … , X, Y, Z, a, b, c, … , x, y, z, 0, 1, 2 , … , 9, ., ,, !, #, $, space …}
- A finite, nonempty set ∑2 of 64 Braille ASCII characters is called Alphabet 2: ∑2 = {space, !, ", #, $, %, ^, … , _ }

1.4 ASCII

ASCII stands for American Standard Code for Information Interchange. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort. ASCII was developed a long time ago and now the non-printing characters are rarely used for their original purpose. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
0	00	000	0000000	NUL (null character)	32	20	040	0100000	space	64	40	100	1000000	@	96	60	140	1100000	`
1	01	001	0000001	SOH (start of header)	33	21	041	0100001	!	65	41	101	1000001	A	97	61	141	1100001	a
2	02	002	0000010	STX (start of text)	34	22	042	0100010	"	66	42	102	1000010	B	98	62	142	1100010	b
3	03	003	0000011	ETX (end of text)	35	23	043	0100011	#	67	43	103	1000011	C	99	63	143	1100011	c
4	04	004	0000100	EOT (end of transmission)	36	24	044	0100100	$	68	44	104	1000100	D	100	64	144	1100100	d
5	05	005	0000101	ENQ (enquiry)	37	25	045	0100101	%	69	45	105	1000101	E	101	65	145	1100101	e
6	06	006	0000110	ACK (acknowledge)	38	26	046	0100110	&	70	46	106	1000110	F	102	66	146	1100110	f
7	07	007	0000111	BEL (bell (ring))	39	27	047	0100111	'	71	47	107	1000111	G	103	67	147	1100111	g
8	08	010	0001000	BS (backspace)	40	28	050	0101000	(	72	48	110	1001000	H	104	68	150	1101000	h
9	09	011	0001001	HT (horizontal tab)	41	29	051	0101001	)	73	49	111	1001001	I	105	69	151	1101001	i
10	0A	012	0001010	LF (line feed)	42	2A	052	0101010	*	74	4A	112	1001010	J	106	6A	152	1101010	j
11	0B	013	0001011	VT (vertical tab)	43	2B	053	0101011	+	75	4B	113	1001011	K	107	6B	153	1101011	k
12	0C	014	0001100	FF (form feed)	44	2C	054	0101100	,	76	4C	114	1001100	L	108	6C	154	1101100	l
13	0D	015	0001101	CR (carriage return)	45	2D	055	0101101	-	77	4D	115	1001101	M	109	6D	155	1101101	m
14	0E	016	0001110	SO (shift out)	46	2E	056	0101110	.	78	4E	116	1001110	N	110	6E	156	1101110	n
15	0F	017	0001111	SI (shift in)	47	2F	057	0101111	/	79	4F	117	1001111	O	111	6F	157	1101111	o
16	10	020	0010000	DLE (data link escape)	48	30	060	0110000	0	80	50	120	1010000	P	112	70	160	1110000	p
17	11	021	0010001	DC1 (device control 1)	49	31	061	0110001	1	81	51	121	1010001	Q	113	71	161	1110001	q
18	12	022	0010010	DC2 (device control 2)	50	32	062	0110010	2	82	52	122	1010010	R	114	72	162	1110010	r
19	13	023	0010011	DC3 (device control 3)	51	33	063	0110011	3	83	53	123	1010011	S	115	73	163	1110011	s
20	14	024	0010100	DC4 (device control 4)	52	34	064	0110100	4	84	54	124	1010100	T	116	74	164	1110100	t
21	15	025	0010101	NAK (negative acknowledge)	53	35	065	0110101	5	85	55	125	1010101	U	117	75	165	1110101	u
22	16	026	0010110	SYN (synchronize)	54	36	066	0110110	6	86	56	126	1010110	V	118	76	166	1110110	v
23	17	027	0010111	ETB (end transmission block)	55	37	067	0110111	7	87	57	127	1010111	W	119	77	167	1110111	w
24	18	030	0011000	CAN (cancel)	56	38	070	0111000	8	88	58	130	1011000	X	120	78	170	1111000	x
25	19	031	0011001	EM (end of medium)	57	39	071	0111001	9	89	59	131	1011001	Y	121	79	171	1111001	y
26	1A	032	0011010	SUB (substitute)	58	3A	072	0111010	:	90	5A	132	1011010	Z	122	7A	172	1111010	z
27	1B	033	0011011	ESC (escape)	59	3B	073	0111011	;	91	5B	133	1011011	[	123	7B	173	1111011	{
28	1C	034	0011100	FS (file separator)	60	3C	074	0111100	<	92	5C	134	1011100	\	124	7C	174	1111100	|
29	1D	035	0011101	GS (group separator)	61	3D	075	0111101	=	93	5D	135	1011101	]	125	7D	175	1111101	}
30	1E	036	0011110	RS (record separator)	62	3E	076	0111110	>	94	5E	136	1011110	^	126	7E	176	1111110	~
31	1F	037	0011111	US (unit separator)	63	3F	077	0111111	?	95	5F	137	1011111	_	127	7F	177	1111111	DEL

Figure 1.2 ASCII Table

1.5 ASCII TO BRAILLE CONVERSION

ASCII Braille is a set of numerical characters codes to representing six-dot braille cells electronically. ASCII Braille formally known as The North American ASCII code.

Braille ASCII uses the 64 ASCII characters between 32 and 95 inclusive. All capital letters in ASCII correspond to their equivalent values in uncontracted English Braille. Note however that, unlike standard print, there is only one Braille symbol for each letter of the alphabet. Therefore, in Braille, all letters are lower-case by default, unless preceded by a capitalization sign (⠠ dot 6).

ASCII Hex	ASCII Glyph	Braille Dots	Braille Glyph	Unicode Braille Glyph	Braille Meaning	ASCII Hex	ASCII Glyph	Braille Dots	Braille Glyph	Unicode Braille Glyph	Braille Meaning
20	(space)	000000			(space)	40	@	010000			(accent prefix)
21	!	011011			the	41	A	100000			a
22	"	000100			(contraction)	42	B	101000			b
23	#	010111			(number prefix)	43	C	110000			c
24	$	111001			ed	44	D	110100			d
25	%	110001			sh	45	E	100100			e
26	&	111011			and	46	F	111000			f
27	'	000010			'	47	G	111100			g
28	(	101111			of	48	H	101100			h
29	)	011111			with	49	I	011000			i
2A	*	100001			ch	4A	J	011100			j
2B	+	010011			ing	4B	K	100010			k
2C	,	000001			(uppercase prefix)	4C	L	101010			l
2D	-	000011			-	4D	M	110010			m
2E	.	010001			(italic prefix)	4E	N	110110			n
2F	/	010010			st	4F	O	100110			o
30	0	000111			"	50	P	111010			p
31	1	001000			,	51	Q	111110			q
32	2	001010			;	52	R	101110			r
33	3	001100			:	53	S	011010			s
34	4	001101			.	54	T	011110			t
35	5	001001			en	55	U	100011			u
36	6	001110			!	56	V	101011			v
37	7	001111			(or)	57	W	011101			w
38	8	001011			" or ?	58	X	110011			x
39	9	000110			in	59	Y	110111			y
3A	:	100101			wh	5A	Z	100111			z
3B	;	000101			(letter prefix)	5B	[	011001			ow
3C	<	101001			gh	5C	\	101101			ou
3D	=	111111			for	5D	]	111101			er
3E	>	010110			ar	5E	^	010100			(currency prefix)
3F	?	110101			th	5F	_	010101			(contraction)

Figure 1.3 Braille from various sources

In Braille, each word is represented by 2x3 matrix and defined by a system that converts the 2x3 array into an array of bits. The Braille cell is represented by numbers 0 to 5 as below.

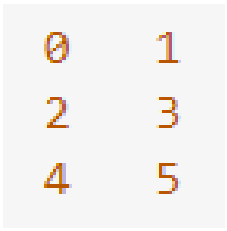

Figure 1.4 Matrix representation

We have to compress the representation into a single byte of data and each digit above can be represented in a single bit, numbered 0-5.

By giving every character its own bit pattern, you can then convert any possible string into a set of bytes for conversion by your library.

Thus A is represented by raising the top left pin, which is the same as bit 0 which can be represented by the byte B000001 or 0x01:

Figure 1.5 Braille of A

1.6 COMPLEXITIES OF BRAILLE

In generally in braille system one character can have many meanings, depending on where it is placed within a word and to which grade it was translated into.

For example, the character with dots two and five raised has three meanings.

For grade two, it could mean ‘cc’ if it is in the middle, ‘con’ if it is at the front of a word or could mean the number three in the ASCII Braille. Another example would be the character with the dots two, three and five raised is the letter ‘f’. This character could mean ‘ff’ when used in the middle of a word, at the beginning of a word, it would mean ‘to’ and at the end of a word, the same character represents an exclamation point.

The contraction rules take into account the linguistic structure of the word thus, contractions are not to be used when their use would alter the usual Braille form of a base word to which a prefix or suffix has been added. And some portions of the transcription rules are not fully codified and rely on the judgement of transcribe. Thus, when the contraction rules permit the same word in more than one way, preference is given to "the contraction that more nearly approximates correct pronunciation."

Print		knowledgeable
Grade 2 Translation	Braille Characters	
	ASCII Characters	KL$GEA#

Figure 1.6 Print, Grade 2 and Computer Braille

Another example of this complexity is shown where ‘wh’ can be replaced by one character, except when the word is actually two words combined. So the ‘wh’ in a while can be replaced by a character, but the ‘wh’ in rawhide cannot be replaced by one character.

Trying to find out whether or not a word is actually two words combined together would require a large dictionary and thus would also an increase the CPU usage with very little benefit in the actual translation since this occurrence would occur infrequently at best. Thus, for computerized Braille translation, either both rules would either be left out of the translation tables or allow the contraction into one character. Also, contractions could not be used across syllable boundaries, would fall into the above category.

This shows how parts of the word are contracted to different characters. For example the part of the word ‘know’ is translated to the cell one-three which is represented in ASCII by the character ‘k’, ‘ed’ is contracted to the cell one-two four-six which in ASCII is represented by the ASCII character “$” with the letters ‘ble’ being contracted into the ASCII character ‘#’ which is the Braille cell of three four-five-six.

There is also the extra problem with grade two Braille is that in translating into this grade gives problems with loss of translation. For example, translating | and / into Braille gives the same character, thus losing the capability to translate back into English.

There are also problems as different cultures or people using different languages would use different Braille translation rules which can cause confusion. So having a system where Braille can be translated into one language and then into Braille system of a different nature so that the same text can be translated from one Braille system into another would be useful.

CHAPTER 2

LITERATURE SURVEY

In the current world, visual impairment is a common problem. There have been inventions and innovations in many technological fields to help the visually impaired. Invention of Braille code made dramatic change in the life of visually impaired people. They can read proficiently using the traditional braille dots. Normal people were curious to know how the visually impaired are reading via touch. Many educators were attempted to teach Braille for the new blind readers.

Over the last decade, there has been a dramatic reduction in the widespread use of Braille (NCBI, 2006). Of the approximately 14,000 users of the services of the National Council for the Blind in Ireland (NCBI), about 2.8% are registered users of Braille (NCBI, 2006). This figure is similar to that reported for the UK (Keil & Clunies-Ross, 2002). In the US, each year approximately 75,000 people lose all or part of their sight. Nevertheless, almost 90% of blind US children are not being taught Braille (the National Federation of the Blind Jernigan Institute, 2009).

The main factors contributing to low braille literacy in people with sight loss are advances in technology to assist the blind to read texts, such as, screen readers, audio books, text-to-speech software; improvements in medical technology that help reduce the incidence of early blindness; the shortage of skilled braille teachers.

According to World Health Organization (WHO), about 89% of world's visually impaired live in developing countries and majority of the people are living with the lowest pay as the national salary i.e. the average salary if the person is so low that.

Most of the devices in the market are too costly and not so comfortable to use, or Braille scanners, yet a low-cost Braille system is not available in the market for visually impaired persons in the developed countries where braille reading and writing can be taught without a Braille teacher.

Many researchers were looking for low cost, portable effective real-time displays for visually impaired people. Various actuators such as solenoids, piezoelectric materials, and electro-active polymers have been used in developing the Braille displays. However refreshable Braille displays had very promising features to fit into the requirements of low cost, portability and real time application. The overview of some of the research work related to the conversion of various types of scripts to Braille is done by P. Blenkhorn.

P. Blenkhorn ET. Al. have worked on the problem of converting Word-Processed Documents into formatted Braille documents. The problem has been addressed in context to the users of the word processor who want to produce Braille documents. The translator will be integrated with the word processor. It makes easy to use as translation is possible with the help of menu items in MSWord. They have used DLL written in C, to translate Text to Braille. Braille Out system produces reliable contracted Braille from a wide variety of documents. The layout is good and speed is acceptable.

P. Blenkorn has addressed the problem of converting Braille characters into print. The problem has been addressed in the context of Standard English Braille Characters. To address this problem the researcher has presented Finite State System and matching algorithm. Author has also focused on the contracted English Braille. He has also focused on the letter sign conversion. The author has also used decision table concept to clearly state all the rules used in the algorithm. The system is tested on the set of standard English Braille words and on the Braille, files produced by Torch Trust for Blind.

The basic idea behind this is to develop a software project for text to braille converter. This mainly focuses on translating English to any Braille by NLP - Natural language processing. It is the software platform where the text is converted through Arduino. The output can be obtained as text.

A simple and portable two module electronic refreshable braille cell unit is implemented. Six pierced dots are used to represent 26 alphabets and symbols. It uses a method of two tactile refreshable Braille cells consisting of six solenoid pins each, for conversion of alphanumeric character into Braille character.

A cognitive system was developed to reduce the difficulty in reading for visually impaired is presented. The performance of the system is compared and analysed with the existing system. As well the work can be extended to device with wireless technology.

Parameters like form factor, power consumption and tactile effectiveness are vital aspects that have to be considered in designing a braille cell to meet the requirements. This operates the cell pins in a way that when the users rest their fingers over the cell it can make feel like moving fingers on cell. This also shows the development and design Braille book system that is low cost and portable and has one Braille cell.

ASCII Braille is a set of numerical characters codes to representing six-dot braille cells electronically. ASCII Braille formally known as The North American ASCII code.

Braille ASCII is a subset of ASCII characters set and consists of 64 of the printable ASCII characters to represent the combination in six –dots Braille. ASCII Braille is also known as Grade zero Braille. This system would take characters from the ASCII tables of values and map these to the Braille system. Therefore, this system has a separate set of Braille cells reserved for capital letters, lower case letters and numbers.

The solenoids of 10volts. A solenoid converts electrical energy into mechanical work. The IRFZ44N is known for its high drain current and fast switching speed. This is a N-channel MOSFET. Resistors will provide 220 Ohms of resistance wherever it is placed and will handle 1/4 watts. The IN4007 is a rectifier diode, designed specifically for circuits that need to convert alternating current to direct current.

A Bluetooth is connected to the app. HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not.

All these devices are proposed to overcome the sufferings of hearing and visually impaired people. Hence with these ideas we can further develop algorithm for system performance and various other parameters such as time conversion, data transfer rate, long distance communication and can also provide additional features if required. This again proves that communication is necessary for survival irrespective of impairments. All these proposed models have considered all possible type of communication outputs such as text, Braille code output and audio output irrespective of distance.

CHAPTER 3

HARDWARE AND SOFTWARE

3.1 HARDWARE

3.1.1 ARDUINO

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its products are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone.

Arduino boards are available commercially in preassembled form or as do it yourself (DIY) kits. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (shields) and other circuits.

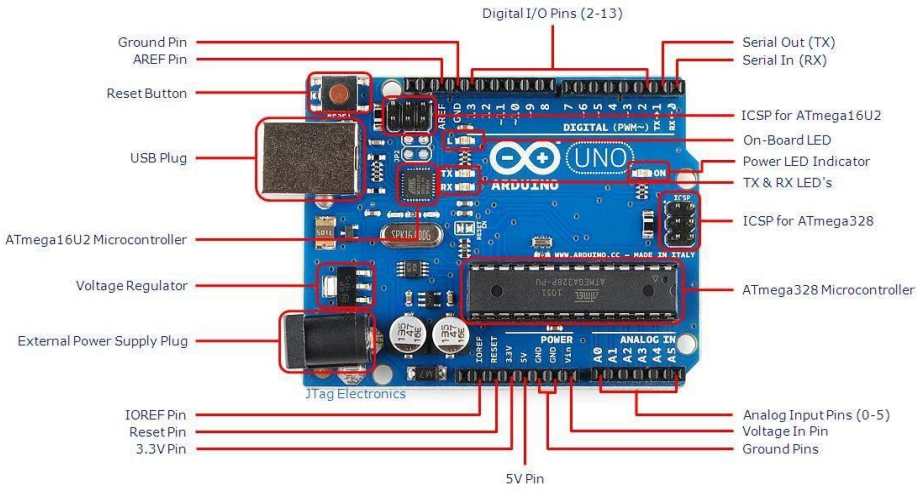

Figure 3.1 Arduino Uno

The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using C and C++ programming languages. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2005 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

3.1.2 ARDUINO UNO TECHNICAL SPECIFICATIONS

1. Microcontroller: Microchip ATmega328P
2. Operating Voltage: 5 Volts
3. Input Voltage: 7 to 20 Volts
4. Digital I/O Pins: 14 (of which 6 provide PWM output)
5. Analog Input Pins: 6
6. DC Current per I/O Pin: 20 mA
7. DC Current for 3.3V Pin: 50 mA
8. Flash Memory: 32 KB of which 0.5 KB used by bootloader
9. SRAM: 2 KB
10. EEPROM: 1 KB
11. Clock Speed: 16 MHz
12. Length: 68.6 mm
13. Width: 53.4 mm
14. Weight: 25 g

3.1.3 PIN CONFIGURATIONS

1. LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it's off.
2. VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
3. 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator and can damage the board.
4. 3V3: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
5. GND: Ground pins.
6. IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.
7. Reset: Typically used to add a reset button to shields which block the one on the board.

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate 13 at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labelled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. In addition, some pins have specialized functions.

1. Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.
2. External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
3. PWM (pulse-width modulation): 3, 5, 6, 9, 10, and 11. Can provide 8- bit PWM output with the analogWrite() function.
4. SPI (Serial Peripheral Interface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
5. TWI (two-wire interface) / I²C: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
6. AREF (analog reference): Reference voltage for the analog inputs.

3.1.4 ATMEGA328

The ATmega328 is a single-chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core.

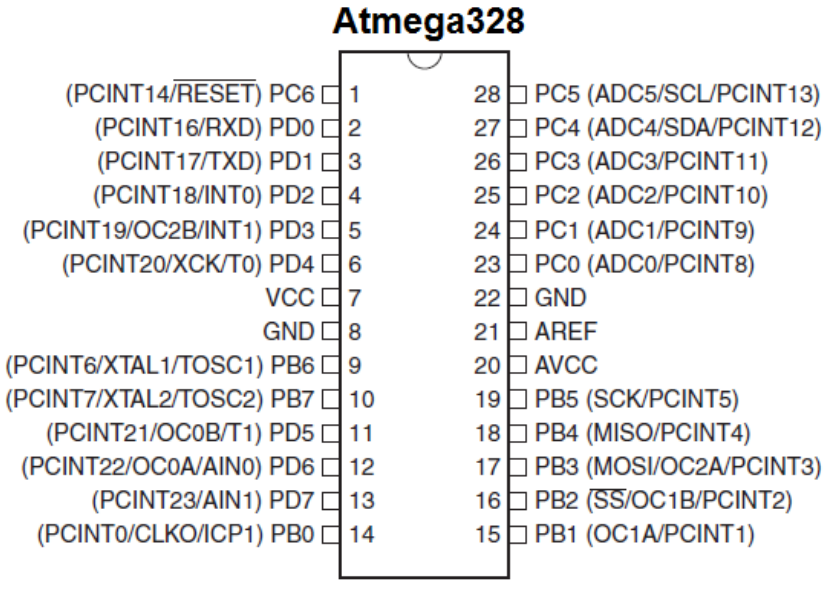

Figure 3.2 Atmega 328 Pinout

The Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8- channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1MIPS per MHz.

3.1.5 PUSH PULL SOLENOIDS

A solenoid is a device comprised of a coil of wire, the housing and a moveable plunger (armature). When an electrical current is introduced, a magnetic field forms around the coil which draws the plunger in. Like all magnets, the magnetic field of an activated solenoid has positive and negative poles that will attract or repel material sensitive to magnets. In a solenoid, the electromagnetic field causes the piston to either move backward or forward, which is how motion is created by a solenoid coil.

Figure 3.3 Solenoids

More simply, a solenoid converts electrical energy into mechanical work. Solenoids are a great way to induce linear motion for pushing, pulling or controlling switches and levers. This smaller solenoid is designed to work directly with 5V which makes it a great match for embedded projects. It has a throw of about 4.5mm and 2M2 mounting holes on the body.

3.1.6 IRFZ44n MOSFET

The IRFZ44N is an N-channel MOSFET with a high drain current of 49A and low R_{ds} value of 17.5 mΩ. It also has a low threshold voltage of 4V at which the MOSFET will start conducting. Hence it is commonly used with microcontrollers to drive with 5V. However a driver circuit is needed if the MOSFET has to be switched in completely.

Figure 3.4 IRFZ44n

The IRFZ44N is known for its high drain current and fast switching speed. Adding to that it also has a low R_{ds} value which will help in increasing the efficiency of switching circuits. The MOSFET will start turning on with a small gate voltage of 4V, but the drain current will be maximum only when a gate voltage of 10V is applied. If the MOSFET has to be driven directly from a microcontroller like Arduino then try the logic level version IRLZ44n MOSFET.

3.1.7 RESISTORS

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Resistors act to reduce current flow, and, at the same time, act to lower voltage levels within circuits. In electronic circuits, resistors are used to limit current flow, to adjust signal levels, bias active elements, and terminate transmission lines among other uses.

This resistor will provide 220 Ohms of resistance wherever it is placed and will handle 1/4 watts. Use these low value resistors for voltage dividers and where you need to keep the current flow as high as possible.

3.1.8 DIODE

1N4007 is a rectifier diode, designed specifically for circuits that need to convert alternating current to direct current. It can pass currents of up to 1 A, and have peak inverse voltage (PIV) rating of 1,000 V.

Figure 3.5 1N4007 Diode

A 1N4007 is a widely used general purpose diode. It is normally build to use as rectifier in the power supplies section of electronic appliances for converting AC voltage to DC with other filter capacitors.

3.1.9 HC 05 BLUETOOTH MODULE

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC.

Figure 3.6 HC 05 Bluetooth Module

HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds. This module works on 3.3 V.

3.2 SOFTWARE

3.2.1 ARDUINO IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, mac OS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards. The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring.

The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

Figure 3.7 Arduino IDE

The Arduino IDE employs the program argued to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

3.2.2 WRITING SKETCHES

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

3.2.3 LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch.

3.2.4 SERIAL MONITOR

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

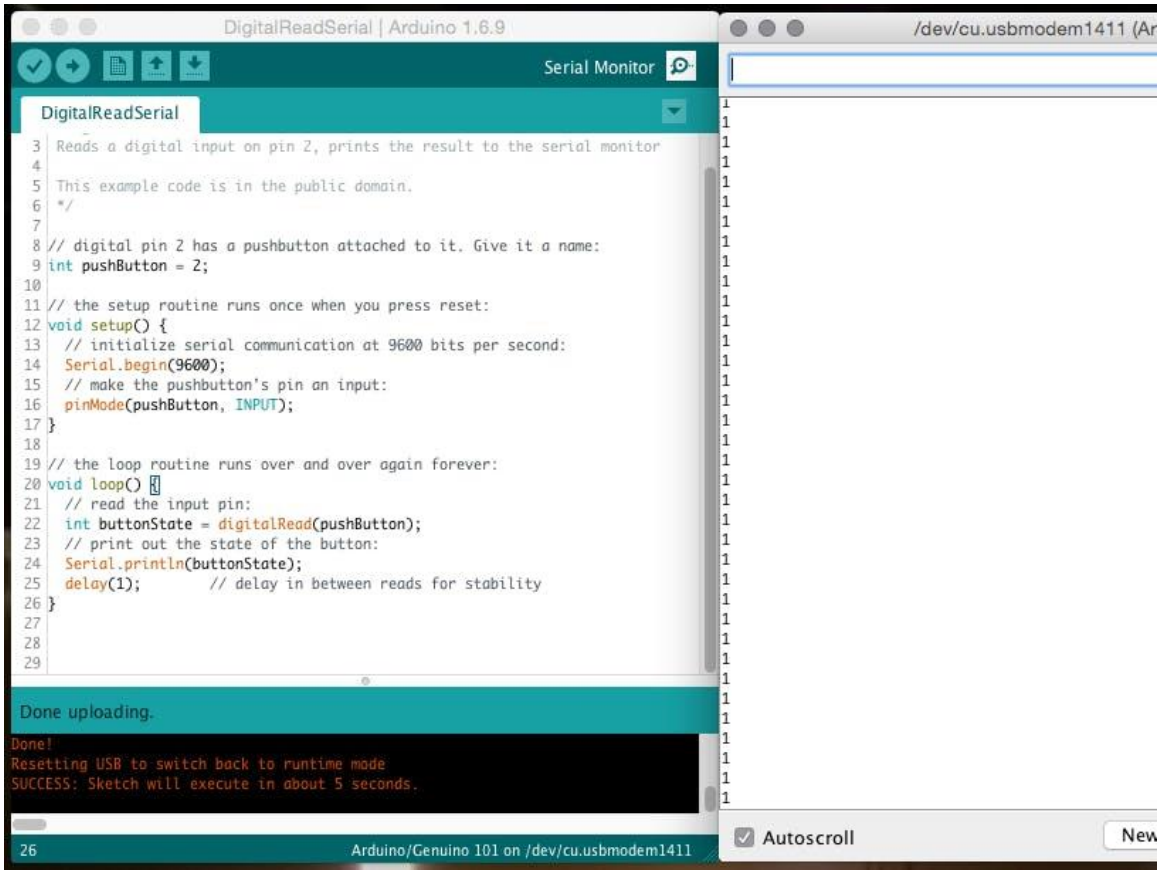

Figure 3.8 Serial Monitor

3.2.5 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

Android Studio supports all the same programming languages of IntelliJ e.g. Java, C++, and more with extensions, such as Go and Android Studio 3.0 or later supports Kotlin and all Java 7 language features and a subset of Java 8 language features that vary by platform version. External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12. At least some new language features up to Java 12 are usable in Android.

Android Studio provides extensive tools to help you test your Android apps with JUnit 4 and functional UI test frameworks. With Espresso Test Recorder, you can generate UI test code by recording your interactions with the app on a device or emulator. You can run your tests on a device, an emulator, a continuous integration environment, or in Firebase Test Lab. Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

Figure 3.9 Android Studio

3.2.6 JAVA

Java is a class-based, object-oriented programming language that is designed to have a few implementation dependencies as possible. It is a general-purpose programming language intended to let the application developers write once, run anywhere (WORA), meaning that compiled java code can run on all platforms that support java without the need of recompilation. Java applications are typically compiled to bytecode that run on any java virtual machine (JVM) regardless of underlying computer architecture. The syntax for java is similar to C and C++, but has fewer low-level facilitates than either of them. The java runtime provides dynamic capabilities that are typically not available in traditional compiled languages.

3.2.7 SOFTWARE DEVELOPMENT KIT (SDK)

A software development kit (SDK) is a collection of software development tools in one installable package. They facilitate the creation of applications by having a compiler, debugger and perhaps a software framework. They are normally specific to a hardware platform and operating system combination. To create applications with advanced functionalities such as advertisements, push notifications, etc; most application software developers use specific software development kits.

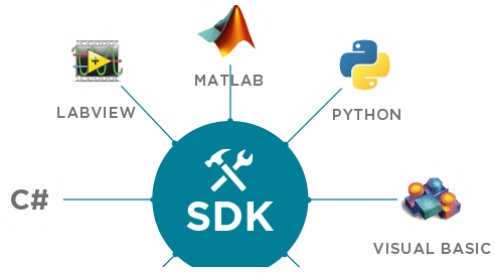

Figure 3.2 SDK

Some SDKs are required for developing a platform-specific app. For example, the development of an Android app on the Java platform requires a Java Development Kit. For iOS applications (apps) the iOS SDK is required. For Universal Windows Platform the .NET Framework SDK might be used. There are also SDKs that add additional features and can be installed in apps to provide analytics, data about application activity, and monetization options.

CHAPTER 4

TESTING OF COMPONENTS

4.1 ARDUINO UNO

The Arduino Uno is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which six can be used as PWM outputs and six can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button.

Figure 4.1 Arduino Testing

To verify the working of Arduino, use any basic program for testing. Connect the Arduino board to the computer and upload the code through Arduino IDE. Verify the working by sending inputs through the serial monitor and acknowledging the output. To verify that you are receiving correct data we can test it by setting each channel to ground and power and read the output in the Serial Monitor. To do this we need a wire to connect between the input connectors and power connectors on the Arduino Uno board. Then follow the series of steps.

- Connect one end of the wire to A0 port
- Connect the other end to GND port
- Analog0 in the Serial Monitor should now read 0.0 volts
- Remove the wire from GND and connect it to 5V
- Analog0 should now read approximately 5.0 volts
- Remove the wire from 5V and connect it to 3.3V
- Analog0 should now read approximately 3.3 volts
- Repeat the same procedure with A1, D2 and D3

4.2 MINI PUSH-PULL SOLENOIDS

Solenoids are made of a coil of copper wire with an armature in the middle. Solenoids function much like electromagnets, producing a magnetic field when a current is applied to them, but they lack the magnetic cores that allow for adjustment of that magnetic field's power. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull (from one end) or push (from the other). This solenoid in particular is very small, with a 20mm long body and a captive armature with a return spring. This means that when activated with 5V DC, the solenoid moves and then the voltage is removed it springs back to the original position, which is quite handy.

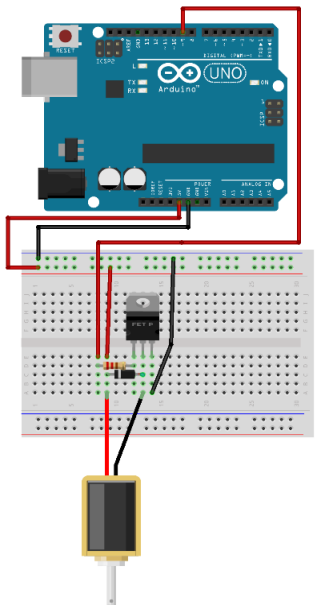

Figure 4.2 Solenoid Testing

To drive a solenoid we will need a power transistor and a protection diode. We will need a good power supply to drive a solenoid, as a lot of current will rush into the solenoid to charge up the electro-magnet, about 1 Amp, so we should be careful of trying to activate from a computer's USB. Detecting a faulty solenoid is easily done with an electrical multi-meter: once the connections to the power source have been tested and deemed functional, test the continuity and resistance of the solenoid. If the multi-meter fails to beep during the continuity test or fails to provide a reading during the resistance test, the solenoid should be replaced.

4.3 HC-05 BLUETOOTH MODULE

HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration. Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth. It is simple to communicate, we just have to type in the Bluetooth terminal application of smartphone. Characters will get sent wirelessly to Bluetooth module HC-05. HC-05 will automatically transmit it serially to the PC, which will appear on terminal. Same way we can send data from PC to smartphone.

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin.

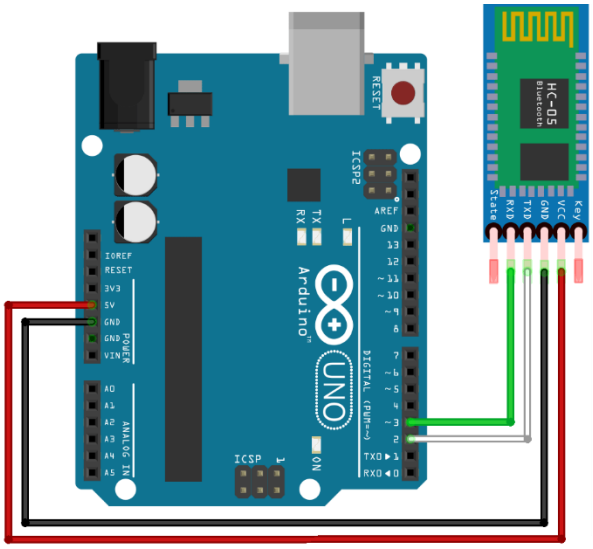

Figure 4.3 HC05 Testing

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU. During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode.

4.4 INTEGRATED TESTING

It is a level of testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

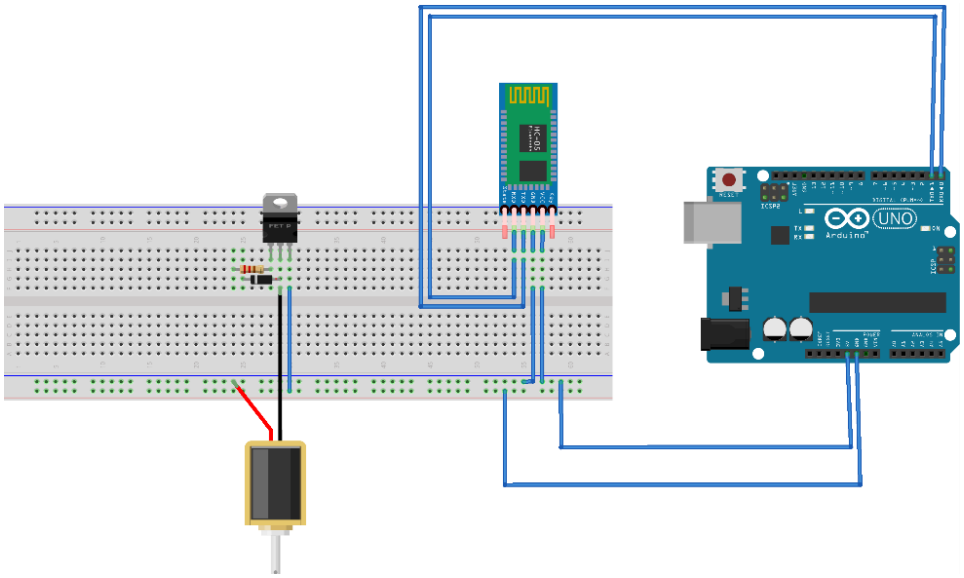

Figure 4.4 Integrated Testing

We have connected the solenoid, Bluetooth module (HC-05) and Arduino. Here the MOSFET and diode acts as a relay circuit to the solenoid. The Bluetooth module is connected to the transmitter and receiver pins of Arduino Uno. We send the input text and check for the working of the solenoids in corresponding with Bluetooth. We have then repeated the process for all the six solenoids.

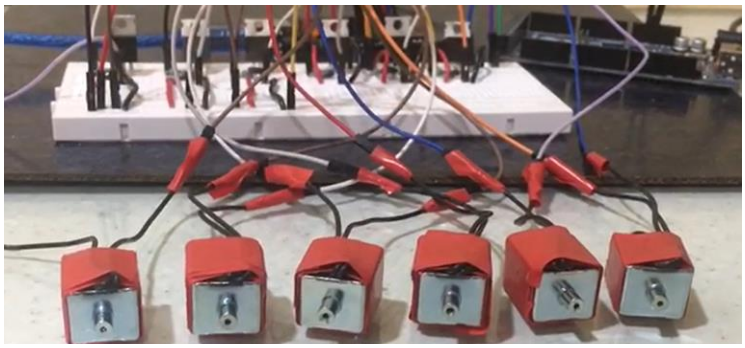

Figure 4.5 Solenoids Integrated Testing

CHAPTER 5

DESIGN OF THE CIRCUIT & APP

5.1 CIRCUIT DESIGN

We are going to use Arduino Uno for the conversion process. The code is based on the conceptual pattern of Braille language. Here, we will be using seven pins and one ground pin. The pin numbers 2, 3, 4, 5, 6, 7 are used as the output pins and each pin will be connected to the solenoids. For every different pattern, we will be using a different format of the coding.

The Solenoids are represented in a 2x3 matrix and the pins are connected to the cell as follows:

0 – Pin 2 1 – Pin 3

2 – Pin 4 3 – Pin 5

4 – Pin 6 5 – Pin 7

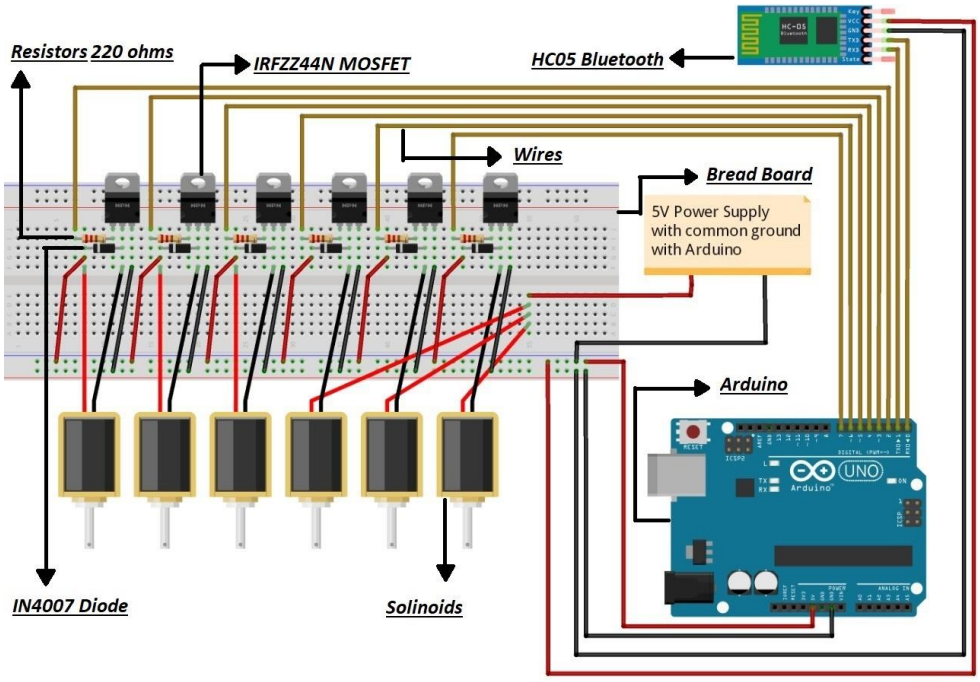

Figure 5.1 Circuit Design

5.1.1 SOLENOID CIRCUIT

The solenoid is an electromagnet. The metal rod in the center is called an armature. It will be drawn into the solenoid when power is applied to the coil. When power is cut, the spring will return the armature to its starting position. The solenoid draws 1.1A which is much more than the GPIO pins on the Feather can source or sink. Therefore, a logic level N-channel MOSFET (IRFZ44n) and current recirculation diode will be used to drive the solenoid. The MOSFET can act like a switch. When voltage is applied to the gate pin it allows current to flow between the drain and the source pins.

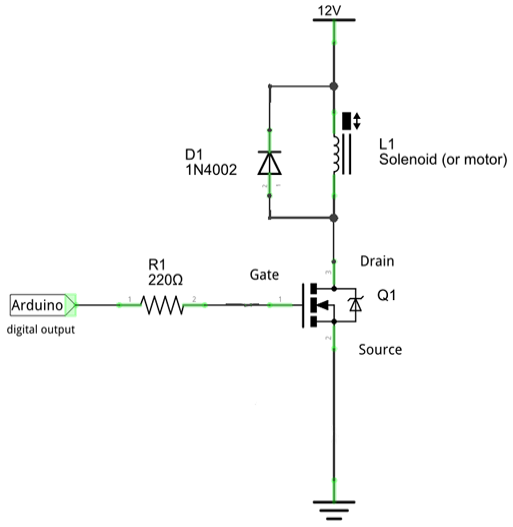

Figure 5.2 Solenoid Circuit

We Connect 5v Power and Ground from your Arduino to your power and ground rails on the breadboard. Then, connect the solenoids to separate lines on the breadboard, one to the 5v power from step 2, the other needs to connect to the collector (middle) of the transistor. We connect the 1N4002 Diode between the two solenoid cables, this will prevent current discharging back through the circuit when the solenoid coil discharges. We will insert power transistor on three separate lines of the breadboard, with the flat side facing toward the outside. Ensure the collector's leg is connected to the solenoid and diode line. Then we connect a 220-ohm Resistor from the base leg of the transistor to a separate line and emitter leg to the ground rail. We connect the other side of the resistor from step 6 to digital pin 9, that's our control pin.

5.1.2 HC-05 BLUETOOTH CIRCUIT

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC.

We connect VCC with VCC of Arduino. The connect GND with any GND of Arduino. We will connect Rx pin with Tx of Arduino and Tx pin with Rx of Arduino.

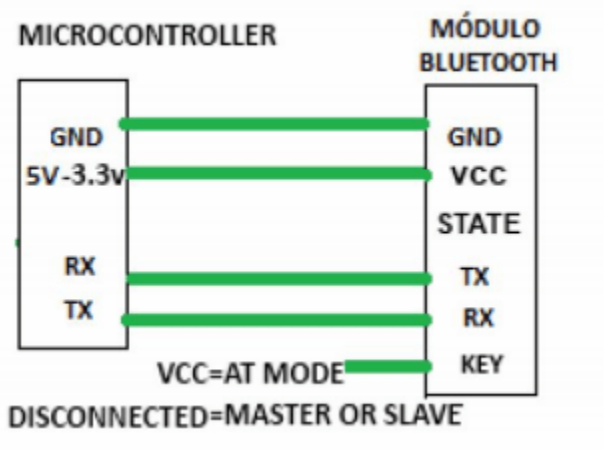

Figure 5.3 Bluetooth Connection

5.2 APP DESIGN

Bluetooth is a wide spread wireless technology paradigm low power consumption characteristics for exchanging data over short distances via short wavelength radio transmissions from immobile or mobile Bluetooth powered devices to create a Personal Area Network (PAN) with high security level.

The Android operating system is an open source and free platform; it is not bound to one hardware provider or manufacturer. This openness of Android is allowing a quick gain of the market share; but this would be covered later in market analysis. Android also can run on many devices with various screen resolutions and sizes. For an Android device to be certified as compatible; it have to follow certain hardware rules including but not limited to: a compass, a Global Positioning System (GPS) trait, a camera and a Bluetooth transceiver. The coding of the application would be worked upon to reach the needed functionalities needed to be implemented in the software.

Choosing the appropriate programming language is not difficult. Java is the best suited programming language for developing Android applications. Java is a well spread language with a lot of support and help documentation and resources. All the needed tools are free and easily installed. These tools are recommended by Google and are free. They include:

Java Development Kit (JDK): it is the foundation of the Android software development kit (SDK).

Android software development kit (SDK): Provides access to the Android's libraries and allowing developing for Android platforms.

Android Development Tools (ADT): Do all background work for developer, such as creating the files and structure required for an Android application.

Android Studio (IDE): provides tools or environment for writing Android programs and joins Java, the Android software development kit and the Android Development Tools (ADT) together and also building the app.

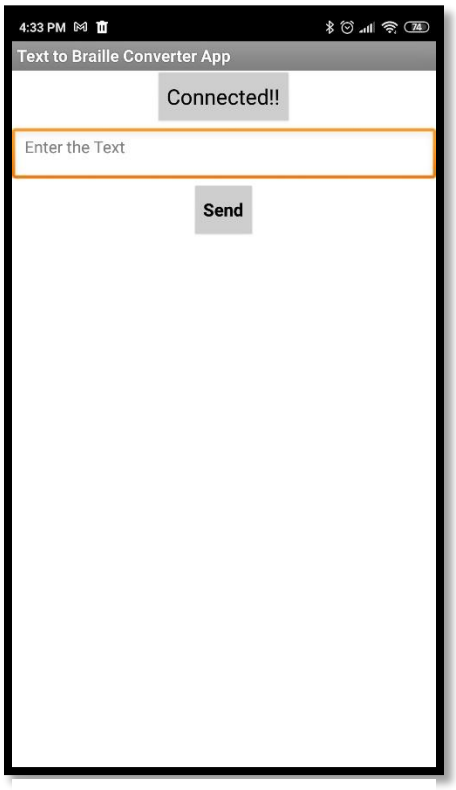

Figure 5.4 App Design

CHAPTER 6

IMPLEMENTATION AND APPLICATION

6.1 IMPLEMENTATION

The Arduino has been coded so as to accept the characters serially and generate the corresponding braille code. The output is digitally toggled using the Digital I/O pins of the Arduino. The input to the system is given in the form of commands (letters or words) through a computer or the android app. These commands are given to the micro-controller for processing. The microcontroller then decodes the given command into the corresponding Braille code. The output from the microcontroller is obtained through the digital pins which are connected to 6 Solenoids.

Each and every ASCII is assigned a value i.e. 100, then later the 64 values to be represented in Braille are assigned values. These assigned values are the 6 bit binary representation for the pins. They are then send as the output HIGH or LOW according to the ASCII value taken as input from the app.

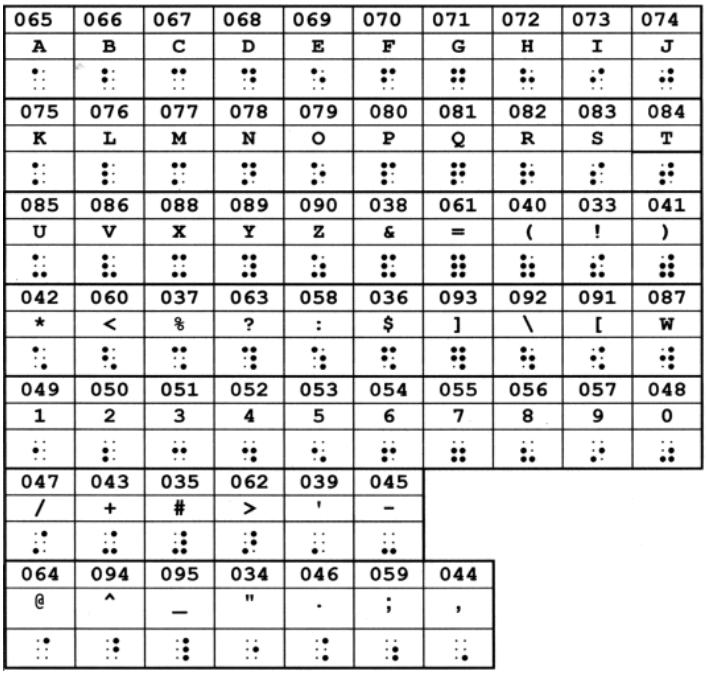

Figure 6.1 Basic ASCII to braille chart

The entire system is dependent of the tactile senses of the user. The user just has to place his/her hand on the interface that is set up in order to interpret the output. The system performance was tested for a set of English Alphabets (A-Z) and Arabic Numbers (0–9) and the observations were analysed. If the input ASCII was not any of the 64 values then all the pins of solenoids will be given a low value.

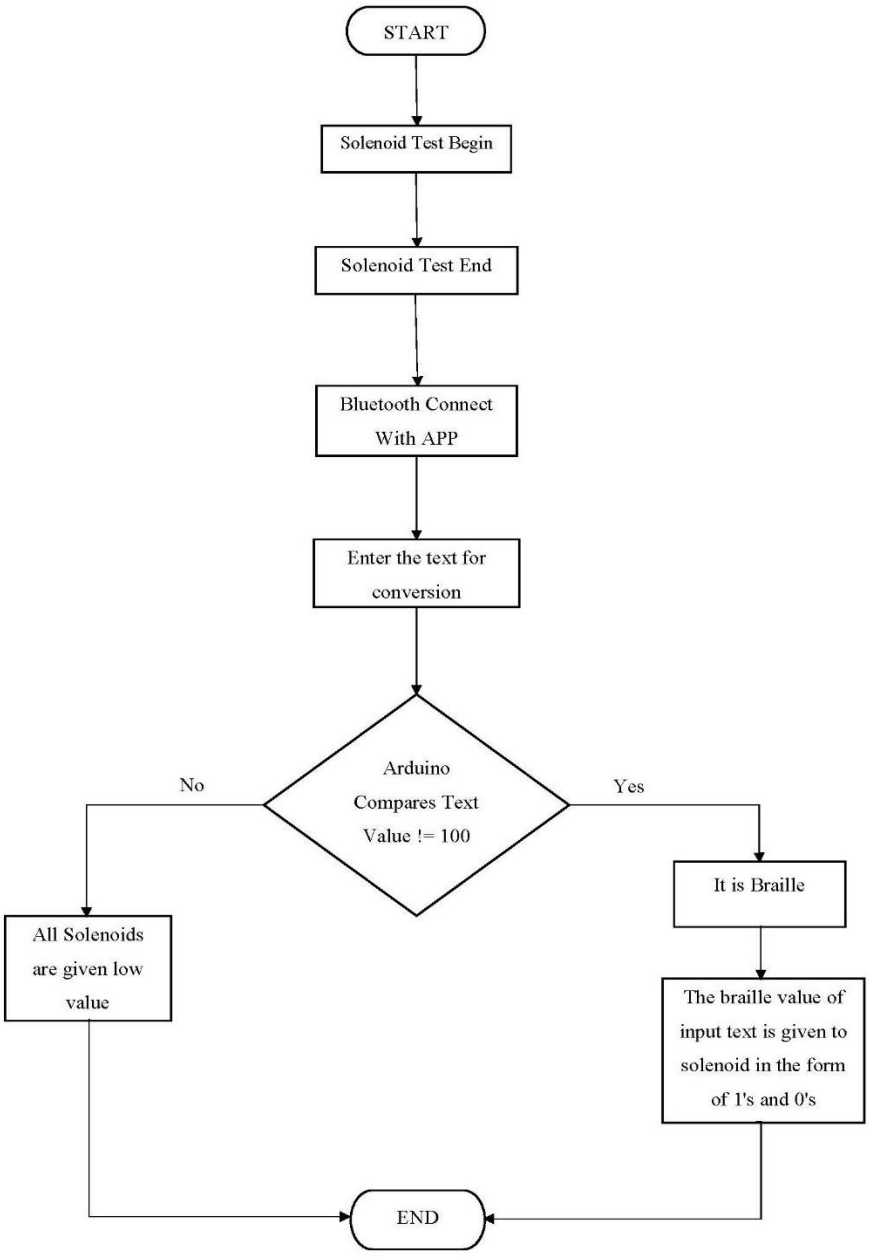

Figure 6.2 Circuit Implementation

6.2 PROJECT RESULTS

We have connected all the push pull Solenoids and the HC 05 Bluetooth module to the Arduino Uno. We have to turn on the Bluetooth in our android device and connect it to the HC 05 Bluetooth module. The device starts with a test to check whether all solenoids are working or not. Then we enter text in the Bluetooth app which sends the data to Bluetooth module in turn determining the Braille of the enter text.

1. The starting of module: Test Solenoids

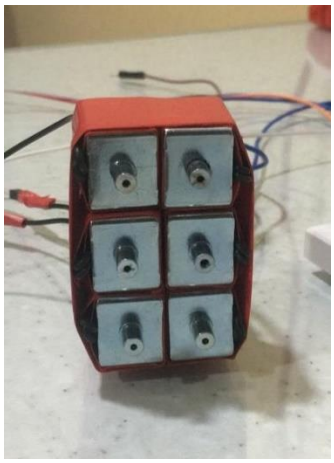

Figure 6.3 Test Result

2. Connect to Bluetooth App using HC-05

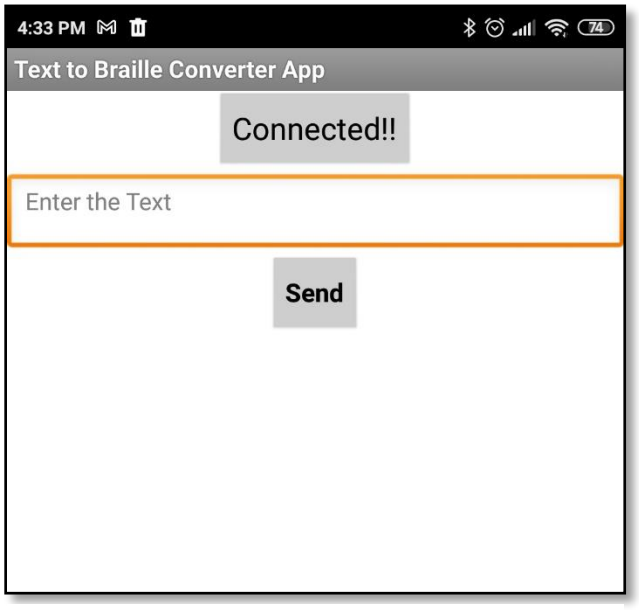

Figure 6.4 App Bluetooth Connection

3. Enter Text in the app and send

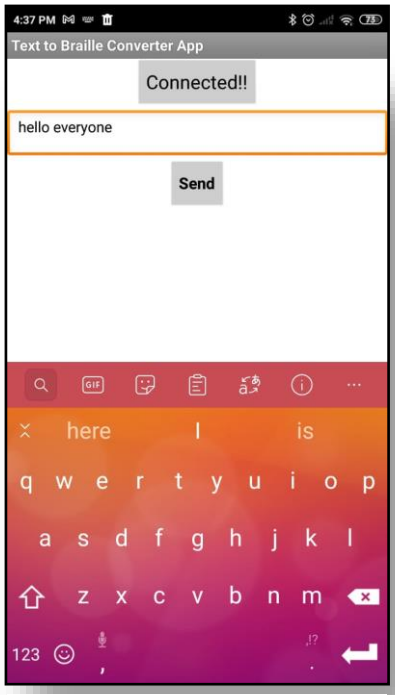

Figure 6.5 Enter Text in App

4. The module will show the Braille of sent text

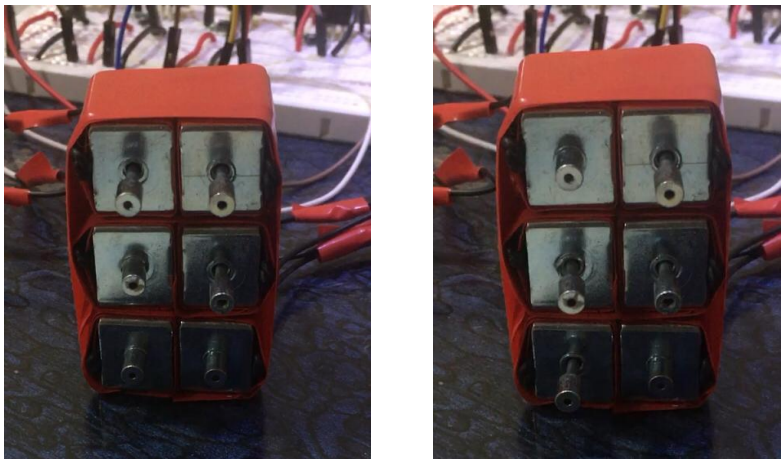

Figure 6.6 Braille Results

6.3 APPLICATIONS

Braille isn't used to transcribe and write books and publications alone. It is also used on signage in public spaces, such as lift key pads, door signs and on restaurant menus, and for labelling everyday items like medications. It is also used as an accessible format for various documents, such as bank statements. This helps blind and visually impaired people to identify objects quickly, work out where something is when out and about, keep safe and maintain their independence. Normal human beings read braille code with their eyes whereas the visually impaired person with their fingers.

6.3.1 ELECTRONIC BRAILLE DISPLAYS

Electronic refreshable braille displays create the dots in braille cells by raising and lowering plastic or metal pins to correspond to the dots in the letters or numbers being represented. Little has changed in this technology, almost since the technology first appeared, and this design contributes to the cost. Typically, braille displays offer anywhere from 10-80 cells in a linear display, and the cost goes up dramatically as the number of cells increases. Although the cost of personal computers has declined dramatically over the years, the decline in cost for a device with an electronic braille display has not.

Figure 6.7 Electronic Braille Display

As a result, until recently, even a small braille display that will fit in a pocket and connect by Bluetooth to a computer, tablet, or smartphone costs nearly $1,000, and a note taker with a small braille display that includes the other features you might expect in a tablet, like a word processor, Wi-Fi, or email, can start at nearly $3,000! Electronic Braille has been disproportionately expensive from the outset, and the Holy Grail of braille displays has been reducing the cost of the technology behind the display without reducing the durability of the display or the feel of the dots.

6.3.2 BRAILLE WATCHES

A braille watch is a portable timepiece used by the blind or visually impaired to tell time. This application can be used in braille watch to make digital. It is used by touching the dial and noticing the embossments.

Figure 6.8 Braille Watch

Both analog and digital versions are available. The analog versions have a protective glass or crystal cover that is flipped open when time needs to be read and the clock-hands are constructed to not be susceptible to movement at the mere touch of the finger that a blind person uses to observe their positions. In the digital form, the dots (like braille script) keep changing position as time changes. In this case, one must understand the Braille alphabet to read the watch.

6.3.3 BRAILLE PHONES

The Braille Phone is a unique innovation. It envisions a unique tactile screen that is made up of grid of pixels which can rise up from surface of the screen. In a typical screen these pixels change colors, which a blind user cannot comprehend. The "height" movement, however, makes every bit of information "touchable". The screen, as an array of pixels, can display shapes, images, characters, Braille and animations. For a non-sighted this display can be metaphorised as a sculpture in a world of paintings (touchscreen phones). Tactile also happens to be more "private" medium as compared to visual and audio interfaces which can easily be snooped upon.

Figure 6.9 Braille Smartphone

As the ubiquitous smartphones of the day, Braille Phone provides access to all features. Instead of using visual display, it relies on tactile screen as an interface. Along with the screen, the UX is seamlessly designed for easy usage with touch-only UI. Non-sighted users have access to all features of smartphone such as phone calling, messaging, organiser, maps, music, email etc.

CONCLUSION

The basic idea behind the design and implementation of the described system is to help visually impaired people achieve the basic objective of reading by using digitized Braille cell. The proposed method implements a braille cell whose aim is to help the visually impaired read. When a series of these modules are kept side by side, a sentence of the braille characters can be created. This can be compared to a braille book in size. The module can represent a book's worth of information in less weight.

The text that is read is then actuated in braille form. The research proved that Braille could be read from the braille cell simply by placing the fingers due to the patterns actuating rather than sliding the fingers across the already formed braille patterns. The digital braille system developed allows the visually impaired people to access any electronic text easily and read on the move. This innovative prototype demonstrates that it is possible to create a digital braille module with reasonable price and improve the visually impaired daily life. The digital braille system helps visually impaired people gain computer literacy. The benefits gained would transform reading for the blind and encourage digital consumption among the young. The product provides sufficient flexibility to the developer to reconfigure it to a device capable of collecting the text data independently without any human intervention. This way e-books can be directly connected to the system and the provision to read any book will applicable. This can prove to be of great help to visually impaired people.

Standards for Braille translation are much higher than for print. According to the context, this level of accuracy is necessary because Braille uses the same ASCII code for different purposes. Hence, even slight errors can cause extreme difficulties in interpretation. The results obtained with the translator show that the system can implement text-to-Braille translation with high accuracy.

FUTURE SCOPE

Developments in technology mean it is now cheaper to read through a computer using screen readers or audio files than Braille. However, this doesn't replace the need for Braille, in the same way as using computers hasn't replaced the need for us to learn to write by hand. It is sometimes said that speech is for speed, and Braille is for accuracy. Braille can provide layout information more efficiently than audio. It's also easier to spot things like spelling mistakes when reading Braille than it is to hear mispronunciations amongst a lot of speech.

Technological developments have made Braille far more usable for blind people. Apart from making it easier to convert, it also makes Braille far more portable. A whole braille book can now be stored on a small disk or memory stick, rather than taking up reams of paper and shelves of storage space. It's clear that technology will continue to make huge breakthroughs in enabling blind and partially sighted people to communicate in new ways in the future. But for almost two centuries little has equalled the practicality and simplicity of Braille.

Many of the new technologies now available to blind people may in fact increase the use of Braille as a way to interact with people and computers and will mean better access to information than ever before. The arrival of portable wireless devices that scan text and translate it to soft (refreshable) Braille will mean greater access to written information in a range of new environments. Likewise the emergence of portable electronic books may make it easier to access information in Braille and other accessible formats.

No one knows what the future holds and undoubtedly some technological advances will replace Braille. But in the same way the invention of the printing press reduced but didn't replace the use of pen, so Braille is likely to remain useful for a huge range of everyday uses.

REFERENCES

[1] Noushad S, Zafar Iqbal M, “Single cell Bangla braille book reader for visually impaired people”, *IEEE*, Nov.2018.

[2] Abhinav K, Kishor B,“Low cost e-book reading device for blind people”, *IEEE,* pp 516–520, Jun.2015.

[3] Alexander R, O’Modhrain S, Brent Gillespie R, Matthew W, Rodger M, “Refreshing refreshable braille displays”, *IEEE,* pp. 287–297, Oct.2015.

[4] Green SR, Gregory BJ, Gupta NK, “Dynamic braille display utilizing phase-change microactuators”, *IEEE*, pp. 307–310. Apr.2006.

[5] Blenkhorn, P. "A System for Converting Print into Braille", IEEE Transactions on Rehabilitation Engineering, vol. 5, No. 2, 1997, pp. 121-129.

[6] Slaby W. "Computerized Braille Translation", Journal of Microcomputer Application, vol. 13, issue n2, pp. 107-113, 1990.

[7] Durre I. and Tuttle D. "A Universal Computer Braille Code for Literary and Scientific Texts", International Technology Conference, December 1991.

[8] Jonathen, A. "Recent Improvement in Braille Transcription", in Proceedings of the ACM annual Conference, vol. 1, 1972, Boston, pp. 208-218.

[9] Jorgen V. "Computerized Braille production", ACM SIGCAPH Computers and the Physically Handicapped, issue 15, pp. 35-40, 1975.

[10] Xiaosong W, Seong-Hyok K, Haihong Z, Chang-Hyeon J, Mark GA, “A refreshable braille cell based on pneumatic microbubble actuators”, *J Microelectromech Syst,* pp. 908–916, Jan.2012.

[11] Matsumoto Y, Arouette X, Ninomiya T, Okayama Y, Miki N, “Vibrational braille code display with MEMS-based hydraulic displacement amplification mechanism” *IEEE,* pp. 19–22, Mar.2010.

APPENDIX

Arduino Code:

```
#define mylimit 256

byte asciitobraille[255];


byte inputbit;

byte mask = 1;




void setup()

{

  int i;

  for(i = 0; i < mylimit; i = i + 1)

  {

    asciitobraille[i] = 100;

  }


  asciitobraille[32] = 0;  // blank is 000000

  asciitobraille[33] = 14;  // exlamation mark is 001110

  asciitobraille[34] = 7;  // double quote is 000111
```

```
asciitobraille[34] = 2;  // single quote is 000010

asciitobraille[40] = 15;  // left parenthesis is 001111

asciitobraille[41] = 15;  // right parenthesis is 001111

asciitobraille[44] = 8;  // comma is 001000

asciitobraille[46] = 13;  // period is 001101

asciitobraille[48] = 28; // 0 is 011100

asciitobraille[49] = 32; // 1 is 100000

asciitobraille[50] = 40; // 2 is 101000

asciitobraille[51] = 48; // 3 is 110000

asciitobraille[52] = 52; // 4 is 110100

asciitobraille[53] = 36; // 5 is 100100

asciitobraille[54] = 56; // 6 is 111000

asciitobraille[55] = 60; // 7 is 111100

asciitobraille[56] = 44; // 8 is 101100

asciitobraille[57] = 24; // 9 is 011000

asciitobraille[58] = 12; // colon is 001100

asciitobraille[59] = 10; // semicolon is 001010

asciitobraille[63] = 11; // question mark is 001011

asciitobraille[65] = 32; // A is 100000

asciitobraille[66] = 40; // B is 101000
```

```
asciitobraille[67] = 48; // C is 110000

asciitobraille[68] = 52; // D is 110100

asciitobraille[69] = 36; // E is 100100

asciitobraille[70] = 56; // F is 111000

asciitobraille[71] = 60; // G is 111100

asciitobraille[72] = 44; // H is 101100

asciitobraille[73] = 24; // I is 011000

asciitobraille[74] = 28; // J is 011100

asciitobraille[75] = 34; // K is 100010

asciitobraille[76] = 42; // L is 101010

asciitobraille[77] = 50; // M is 110010

asciitobraille[78] = 54; // N is 110110

asciitobraille[79] = 38; // O is 100110

asciitobraille[80] = 58; // P is 111010

asciitobraille[81] = 62; // Q is 111110

asciitobraille[82] = 46; // R is 101110

asciitobraille[83] = 26; // S is 011010

asciitobraille[84] = 30; // T is 011110

asciitobraille[85] = 35; // U is 100011

asciitobraille[86] = 43; // V is 101011
```

```
asciitobraille[87] = 29; // W is 011101

asciitobraille[88] = 51; // X is 110011

asciitobraille[89] = 55; // Y is 110111

asciitobraille[90] = 39; // Z is 100111

asciitobraille[97] = 32; // a is 100000

asciitobraille[98] = 40; // b is 101000

asciitobraille[99] = 48; // c is 110000

asciitobraille[100] = 52; // d is 110100

asciitobraille[101] = 36; // e is 100100

asciitobraille[102] = 56; // f is 111000

asciitobraille[103] = 60; // g is 111100

asciitobraille[104] = 44; // h is 101100

asciitobraille[105] = 24; // i is 011000

asciitobraille[106] = 28; // j is 011100

asciitobraille[107] = 34; // k is 100010

asciitobraille[108] = 42; // l is 101010

asciitobraille[109] = 50; // m is 110010

asciitobraille[110] = 54; // n is 110110

asciitobraille[111] = 38; // o is 100110

asciitobraille[112] = 58; // p is 111010
```

```
asciitobraille[113] = 62; // q is 111110

asciitobraille[114] = 46; // r is 101110

asciitobraille[115] = 26; // s is 011010

asciitobraille[116] = 30; // t is 011110

asciitobraille[117] = 35; // u is 100011

asciitobraille[118] = 43; // v is 101011

asciitobraille[119] = 29; // w is 011101

asciitobraille[120] = 51; // x is 110011

asciitobraille[121] = 55; // y is 110111

asciitobraille[122] = 39; // z is 100111

pinMode(2, OUTPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

pinMode(6, OUTPUT);

pinMode(7, OUTPUT);

Serial.begin(9600);

Serial.println("Text to Braille Converter");

Serial.println("Solenoid test");

Serial.println("Begin");
```

```
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  delay(2000);
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  Serial.println("Successfull!! - End Test");
  Serial.println("Enter the text:");
}



void loop()
{
```

```
if (Serial.available() > 0)
{
  inputbit = Serial.read();
  if(asciitobraille[inputbit] == 100)
  {
    Serial.println("Not translatable");
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(7,LOW);
  }
  else
  {
        int outputbit = 2;
        for(mask = 000001; mask < 64; mask <<= 1)
        {
           if(asciitobraille[inputbit] & mask)
```

```
            {
              digitalWrite(outputbit,HIGH);
            }
            else
            {
              digitalWrite(outputbit,LOW);
            }
          outputbit = outputbit + 1;
          }
      Serial.print("The letter: ");
      Serial.print(char(inputbit));
      Serial.println(" is converted to braille");
    }
    delay(2000);
  }
}
```

App Screen Code:

```
package org.appinventor;

import components.runtime.HandlesEventDispatching;

import components.runtime.EventDispatcher;

import components.runtime.Form;

import components.runtime.Component;

import components.runtime.ListPicker;

import components.runtime.TextBox;

import components.runtime.Button;

import components.runtime.BluetoothClient;

import components.runtime.util.YailList;

class Screen1 extends Form implements HandlesEventDispatching {

  private ListPicker ListPicker1;

  private TextBox TextBox1;

  private Button Button1;

  private BluetoothClient BluetoothClient1;

  protected void $define() {

    this.AboutScreen("Text to Braille Converter App");

    this.AlignHorizontal(3);

    this.AppName("GITAM_Project");
```

```
    this.Icon("Gandhi_Institute_of_Technology_and_Management_logo.jpg");
    this.Title("Text to Braille Converter App");
    ListPicker1 = new ListPicker(this);
    ListPicker1.BackgroundColor(0xFFCCCCCC);
    ListPicker1.FontSize(18);
    ListPicker1.Text("Connect Bluetooth");
    TextBox1 = new TextBox(this);
    TextBox1.Width(LENGTH_FILL_PARENT);
    TextBox1.Hint("Enter the Text");
    Button1 = new Button(this);
    Button1.FontBold(true);
    Button1.FontSize(15);
    Button1.Text("Send");
    BluetoothClient1 = new BluetoothClient(this);
    EventDispatcher.registerEventForDelegation(this, "BeforePickingEvent",
"BeforePicking" );
    EventDispatcher.registerEventForDelegation(this, "AfterPickingEvent", "AfterPicking"
);
    EventDispatcher.registerEventForDelegation(this, "ClickEvent", "Click" );
  }
```

```
 public boolean dispatchEvent(Component component, String componentName, String eventName, Object[] params){
  if( component.equals(ListPicker1) && eventName.equals("BeforePicking") ){
   ListPicker1BeforePicking();
   return true;
  }
  if( component.equals(ListPicker1) && eventName.equals("AfterPicking") ){
   ListPicker1AfterPicking();
   return true;
  }
  if( component.equals(Button1) && eventName.equals("Click") ){
   Button1Click();
   return true;
  }
  return false;
 }
 public void ListPicker1BeforePicking(){
  ListPicker1.Elements(YailList.makeList(BluetoothClient1.AddressesAndNames()));
 }
 public void ListPicker1AfterPicking(){
```

```
    ListPicker1.Selection(BluetoothClient1.Connect(ListPicker1.Selection()));
    ListPicker1.Text("Connected!!");
  }
  public void Button1Click(){
    BluetoothClient1.SendText(TextBox1.Text());
    TextBox1.Text("");
  }
}
```