

Speech Synchronised Braille Printer

Ashwin P.S.¹ Karthik.B² Karthik.D³ Senthamizh Selvi.R⁴

Department of Electronics and Communication Engineering
Easwari Engineering College, Ramapuram, Chennai.^{1,2,3,4}
psashwin95@gmail.com¹, karthik.b.1195@gmail.com²
karthikdrj@gmail.com³, senthamizhselvir74@gmail.com⁴

Abstract

This project, *Speech Synchronised Braille Printer*, is a proposal which will help people with visual impairment. People with visual impairment use Braille as their mode of written communication, which is purely based on touch. People without any impairment can easily take written notes and memos to help them get by in their day to day aspects and academic as well as life goals. To enable the Visually Impaired Community to have the same ability, we propose a Voice to Braille Printing mechanism through a semi-portable device. The device is built on Raspberry Pi and is internet-enabled and converts voice to braille and prints the braille characters. Moreover, Braille printers cost a lot – between 50,000 INR and 1, 30,000 INR which makes purchasing them by an individual impractical. This project proposes cost-effective alternatives for expensive Braille printers, while remaining real-time and efficient. Using the proposed device, Visually Impaired People can easily print notes in Braille just by speaking or playing a recording. This device shall serve as an affordable tool to take lecture notes for visually impaired graduates and also help the people with both visual and hearing impairments in real time communication.

Keywords - Voice to Braille, Internet and Web Technologies, Raspberry Pi, Braille Printer

1. Introduction

Presently in India, the population of people with visual impairments stands at five million persons[1] and that of people with both visual and hearing impairments is estimated to be nearly four lakhs.[2] People with visual impairment and people with both visual and hearing impairments face difficulty in communication and pursuing their goals. They communicate through Braille – a tactile writing system. It is written with embossed paper. Braille characters are rectangular blocks called cells that contain bumps called raised dots. The number of dots and their arrangement inside the cell distinguish one character from another. Braille Material needs to be printed in printers called Braille Embossers. These printers are direct impact printers which render text as tactile braille cells. The cost of

these printers start roughly from One lakh rupees, as can be seen from Figure 1 below, and the individual consumer cannot afford such a printer. There is not any viable method for short-hand notes for students with these impairments.



Figure 1. Price of Portable Braille Embosser

In this paper we propose to record the voice of a speaker using microphones and print the translation of the speech in braille. We use Microsoft Bing Speech API to convert Speech to Text. We then implement Text to Braille Translation using Liblouis Open Source Translation. [3] We print the converted Braille onto paper using a Modified Dot Matrix Printer to emulate Braille embossers. We use the following modules to implement our idea.

- Raspberry Pi 3 Model B Version 1.2 with Microphone and Internet connectivity.
- Speech To Text Module
- Text To Braille Module
- Modified Dot Matrix Printer Module

This system uses microphones to capture voice input. The Speech to Text and Text to Braille conversion is done in the Raspberry Pi through Software Implementation and finally Braille Printing is done through modified Epson LX300+II Printer.

This paper helps people with visual impairments to affordably print Braille notes and encourages them to pursue academic goals, without considering their impairment as a hindrance.

2. Methodology

The main aim of this paper is for the recording of the voice by microphone and print the translation in braille. This requires microphone module connected to the Raspberry Pi for recording voice and modified

dot matrix printer to print the braille. The methodology to implement voice to braille is modularized into four modules:

1. Speech to text conversion
2. Text to braille conversion
3. Modification of dot matrix printer
4. Push button circuitry

The modular implementation can be understood from Figure 2 below:

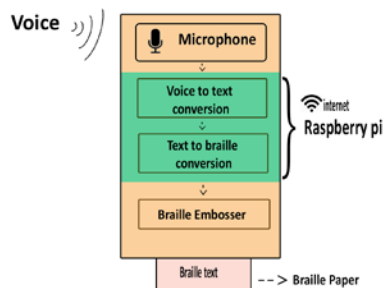


Figure 2. Block Diagram

2.1. Speech to text conversion

This module captures voice from a person through microphone and converts it into text. This module involves the following steps:

- Python script Runs in background waiting for “Start Record” Button Press from User
- On Button Press, the python script invokes bash commands from Raspberry pi to record voice through the microphone.
- Record till the Stop Record button press from the user.
- Save the recorded voice as “english.wav”. The Python script invokes another python script to transcribe the voice to text.
- This Python script calls Bing Speech API.
- The converted text is saved to “Speech.txt” file.

The purpose of using Bing Speech API is because it uses highly efficient Context Dependent Natural Language Processing[4] powered by online engines, which are used in many Microsoft Speech-Text applications including Cortana.

The Steps can be understood using the flow chart representation in Figure 3 that follows:

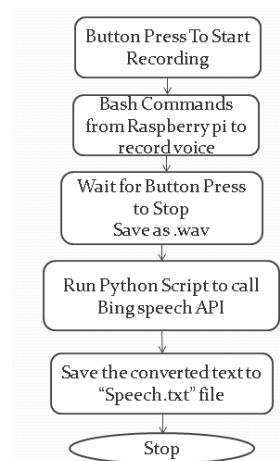


Figure 3. Speech to Text Conversion Flow Chart

2.2. Text to braille conversion

This module converts the text from Speech to Text Module to Braille Text and implements necessary changes in final output to make it printer ready. The module does it through the following steps:

- Python Script waits for Stop Recording Button Press from user.
- On Button Press, bash commands are invoked that invoke the Liblouis library.
- The Liblouis library checks the look-up tables [5] and converts the text to braille.
- The converted braille string is saved to “Translated.brf” file.
- Modifications need to be done to the Translated file to make it print ready.
- A C binary “RevLine” is invoked which reads Translated.brf and Reverses each line and word in place and saves it as Print.txt

The steps above can be understood from the flowchart in Figure 4 that follows:

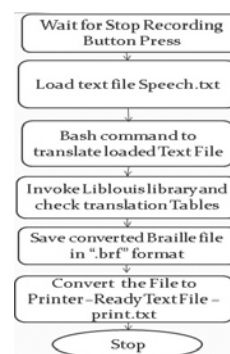


Figure 4. Text to Braille Module Flow Chart

2.3. Modification of dot matrix printer

There are two methods available for building a braille embosser:

- Building a Braille embosser from the Scratch
- Modify a Dot matrix printer to a braille embosser by
 - Removing the ink ribbon
 - Modifying the platen of printer

In this paper we chose to go with modification of Dot matrix printer. [6] In the printer the ribbon is removed to enable direct contact of the printer head pins with the paper. Moreover, to avoid direct contact of print head pins with the platen, the platen is covered in a Cushioned Foam Sleeve. These modifications enable the printer to emulate a Braille Embosser.

This module prints the braille text file. It follows the following steps.

- Python Script waits for Print Button Press from User.
- On Button Press, Bash command is invoked that sends print command to the printer.
- The contents of file are printed.

2.4. Push button circuitry

In the final product, the Raspberry Pi will have to run headless i.e. without any display. Therefore, an interface has to be present for the Visually Impaired User to give commands. Push Buttons are used as an interface in this implementation. Five push buttons are used. Each push button is associated with a command. The button-command association can be seen from Table I as follows.

TABLE I. Push Button – Command Association

Push Button	Command
Button 1	Shutdown Raspberry Pi
Button 2	Restart Raspberry Pi from Idle State
Button 3	Start Recording
Button 4	Stop Recording
Button 5	Print

Push Buttons are connected to General Purpose Input Output (GPIO) Pins on Raspberry Pi in order to perform the listed commands in the table. 6 Pins in Raspberry Pi – Pins Number 5,15,16,17, 39 and 40 are used. Pin 39 is a Ground Pin while others are GPIO pins. The connections between Raspberry pi and the Push buttons can be seen from Figure 5 that follows:

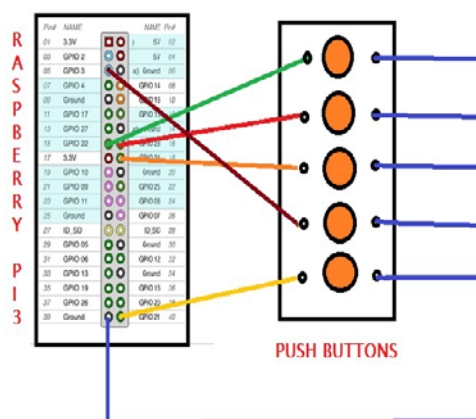


Figure 5. Push-Button – Pi Connections

3. Working

The software needed to implement this system comprises of four scripts. The four scripts are seen in Figure 6 below:

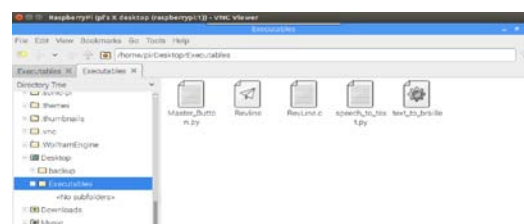


Figure 6. Scripts needed for Execution

The entire system is controlled through buttons interface. On Button Press of specific push buttons, defined actions take place. The Button Press actions are defined in the script Master_Button.py, whose contents are shown in Figure 7 below.

```
#!/usr/bin/env python
import os
import subprocess
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(23,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(22,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(23,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(24,GPIO.IN,pull_up_down=GPIO.PUD_UP)

command1 = 'record -g -f cd -t wav /home/pi/Desktop/Executables/english.wav'
command2 = '/home/pi/Desktop/Executables/print_to_braille'
command3 = 'cat /home/pi/Desktop/Executables/print.txt|pr'

while True:
    inputValue1 = GPIO.input(22)
    inputValue2 = GPIO.input(23)
    inputValue3 = GPIO.input(24)
    inputValue4 = GPIO.input(23)

    if(inputValue1 == False):
        global audproc
        audproc = subprocess.Popen(command1,shell=True,stdout=subprocess.PIPE)
        global pid
        pid = audproc.pid
        sleep(0.4)
        continue

    if(inputValue2 == False):
        os.kill(pid-1, signal.SIGINT)
        transproc = subprocess.Popen(command2,shell=True)
        returnValue1 = subprocess.Popen.wait(transproc)
        continue

    if(inputValue3 == False):
        transproc = subprocess.Popen(command3,shell=True)
        returnValue2 = subprocess.Popen.wait(transproc)
        sleep(0.4)
        continue

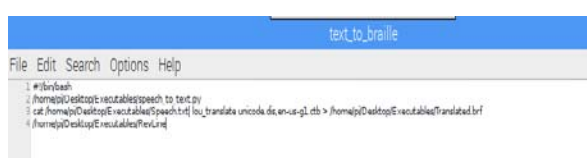
    if(inputValue4 == False):
        os.system("sudo shutdown -h now")
        sleep(0.4)
        break
```

Figure 7. Master_Button Script

Master_Button defines actions for each button as follows:

- On press of Start Record Button, the microphone starts recording audio.
- On press of Stop Record Button, the audio recording is stopped and text_braille script is invoked, which internally invokes speech_to_text.py, lou_translate and RevLine scripts.
- On press of Print Button, uniprint command is used to print the print.txt file.
- On press of Shutdown Button, shutdown command is invoked.

Text_braille is a BASH script. Code is written in the script as seen in Figure 8 and saved:



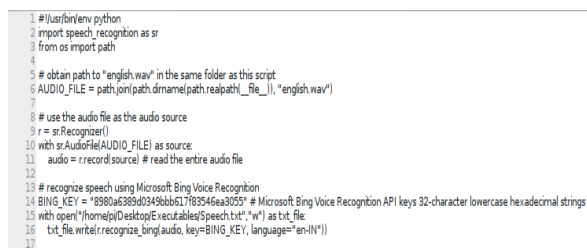
```

1 #!/bin/bash
2 #home/pi/Desktop/Executables/speech_to_text.py
3 cat /home/pi/Desktop/Executables/speech.txt | lou_translate unicode ds-en-us-g1.ctb > /home/pi/Desktop/Executables/Translated.brf
4 #home/pi/Desktop/Executables/RevLine

```

Figure 8. Text_to_Braille Script

Speech_to_text.py is a python script which converts the output of recording “english.wav” into text output “Speech.txt”. The code for speech_to_text.py is written as seen in Figure 9 and saved:



```

1 #!/usr/bin/env python
2 import speech_recognition as sr
3 from os import path
4
5 # obtain path to "english.wav" in the same folder as this script
6 AUDIO_FILE = path.join(path.dirname(path.realpath(__file__)), "english.wav")
7
8 # use the audio file as the audio source
9 r = sr.Recognizer()
10 with sr.AudioFile(AUDIO_FILE) as source:
11     audio = r.record(source) # read the entire audio file
12
13 # recognize speech using Microsoft Bing Voice Recognition
14 BING_KEY = "9580a6389d034960bb21783546ea91051" # Microsoft Bing Voice Recognition API keys 32-character lowercase hexadecimal strings
15 with open("/home/pi/Desktop/Executables/speech.txt", "w") as txt_file:
16     txt_file.write(r.recognize_bing(audio, key=BING_KEY, language="en-IN"))
17

```

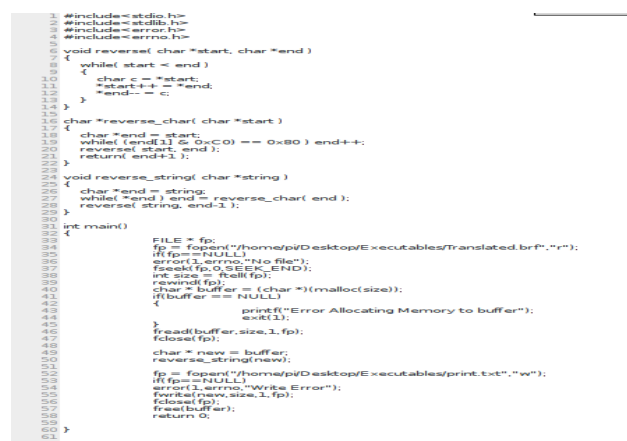
Figure 9. Speech_to_text.py Script

Lou_translate is a BASH command that invokes the liblouis library. “Speech.txt” is given as input and Lou_translate translates it using Translation tables into “Translated.brf”.

During printing, mirroring occurs.[7] The printer prints characters on one side and the embossed patterns have to be felt by reversing the paper. Thus every pattern, word and sentence will be printed in the reverse. To counter this, braille patterns are reversed in the look-up tables itself. (Dot pattern 1 appears as Dot pattern 4, Dot pattern 2 as Dot pattern 5, and Dot pattern 3 as Dot pattern 6 and vice versa).

Then finally RevLine command is invoked, the purpose of which is to reverse each line and word in place, to counter mirroring. The RevLine is a C binary file. RevLine.c is the source file for RevLine

and code for it is written as seen in Figure 10 which follows.



```

#include<stdio.h>
#include<stdlib.h>
#include<error.h>
#include<errno.h>

void reverse( char *start, char *end )
{
    while( start < end )
    {
        char c = *start;
        *start++ = *end;
        *end-- = c;
    }
}

char *reverse_char( char *start )
{
    char *end = start;
    while( *end != '\0' ) end++;
    reverse( start, end-1 );
    return end-1;
}

void reverse_string( char *string )
{
    char *end = string;
    while( *end ) end++;
    reverse( string, end-1 );
}

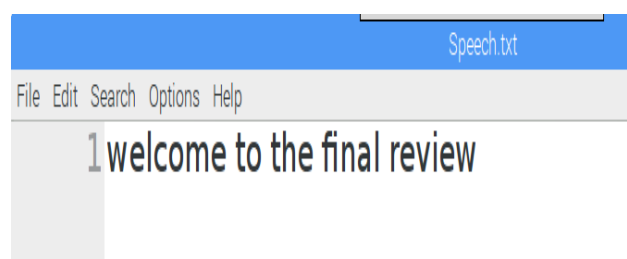
int main()
{
    FILE * fp;
    fp = fopen("/home/pi/Desktop/Executables/translated.brf", "r");
    if( fp == NULL )
        error(1, errno, "No file");
    fseek( fp, 0, SEEK_END );
    int size = ftell( fp );
    rewind( fp );
    char * buffer = (char *) (malloc( size ));
    if( buffer == NULL )
        printf( "Error Allocating Memory to buffer" );
        exit(1);
    fread( buffer, size, 1, fp );
    fclose( fp );
    char * new = buffer;
    reverse_string( new );
    fp = fopen("/home/pi/Desktop/Executables/print.txt", "w");
    if( fp == NULL )
        error(1, errno, "Write Error");
    fwrite( new, size, 1, fp );
    fclose( fp );
    free( buffer );
    return 0;
}

```

Figure 10. RevLine.c Code

4. Results and Discussion

After saving all these scripts, Master_Button.py script is made to run at the Boot-up of Raspberry Pi by adding it to /etc/rc.local. When the script is executed at boot, the buttons are pressed and the output is obtained. The output of speech_to_text.py script, Speech.txt file is shown in Figure 11:



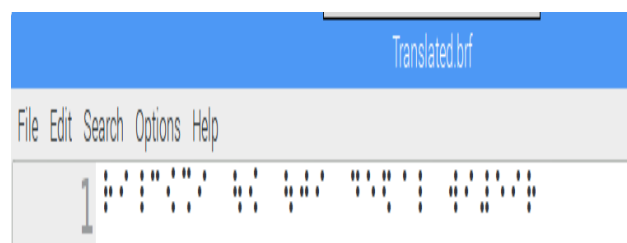
```

1 welcome to the final review

```

Figure 11. Speech to Text Conversion Output

Lou_translate output “Translated.brf” is shown in Figure 12:



```

Translated.brf
1 welcome to the final review

```

Figure 12. Text to Braille Module Output

RevLine output “print.txt” is shown in Figure 13.



Figure 13. Printer Ready File Output

Finally braille output is obtained as shown in Figure 14:

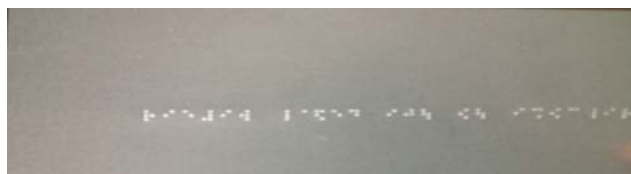


Figure 14. Braille Printer Output

Thus we have obtained the braille output from the given speech input. The final braille output which we obtained is the expected braille output. The text to braille conversion software used here is very much accurate. Butter Paper is used for printing braille here, since embossing is not felt on other kinds of papers and the rendering of braille on Butter Paper is found to be similar to the same on Braille Paper by Embossers.

5. Conclusion and Future work

This paper has been developed mainly to reduce the cost of Braille printing and also to make the braille printer affordable and available to the individuals. This paper projects outputs same as that of the existing high cost braille printers and also helps the visually impaired people to recognize the Braille output easily. This paper involves easy modification from the already available dot-matrix printer. Speech recognition functionality is an added bonus to this paper. This functionality helps the visually impaired people to print braille for their personal use, without using a computer or a mediator, by speaking the sentences to be printed with the help of microphone. The sentences they speak are then translated to braille by the processor and thus printed. Multiple copies can also be printed by a simple button press operation of the Print Button.

Improvements can be made such that vibrational feedback can be provided for the visually impaired to know which button they have just pressed. The microphone module is highly sensitive to distance from speaker. By using industry-class MEMS microphone, this disadvantage in the product can be overcome. Further development in this product can see a whole new dimension opened for people with

visual impairments to compete and grow with the rest of the society.

6. References

- [1] Census India 2011, "Disabled Population By Type Of Disability, Educational Level And Sex - 2011 (India & States/UTs)", Retrieved from www.censusindia.gov.in/2011census/Disability_Data/DDW-0000C-29.xlsx
- [2] Rehabilitation Council of India, "Deafblindness", Retrieved from www.rehabcouncil.nic.in/writereaddata/deafblind.pdf, pp. 14 - 15.
- [3] C.Egli, "Liblouis – A Universal Solution For Braille Transcription Services", Daisy Technical Conference, October 2009.
- [4] G. Dahl, D. Yu, L. Deng and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks For Large Vocabulary Speech Recognition", *IEEE Transactions On Audio, Speech, And Language Processing*, Vol. 20, No. 1, January 2012, pp. 30 – 42.
- [5] K. Bawdekar, A. Kumar and R.K.Das, "Text To Braille Converter", *International Journal of Electronics and Communication Engineering and Technology (IJECET)*, Volume 7, Issue 4, July 2016, pp. 54–61.
- [6] G. Singh, P. Kumar, "Improved Braille Printer", Indian Patent: 3331/DEL/2016, November 2013.
- [7] S. Padmavathi et al., "Economic Printing of Braille Documents", *International Journal of Emerging Technologies in Computational and Applied Sciences*, 8(2), March 2014 - May 2014, pp. 170-173.