Digital Notepad for the Blind

Aditi Kulkarni, Varsha Thomas, Puneet Madan
Department of Electrical and Electronics Engineering
BITS Pilani, Goa -403726
f2012161, f2012010, f2012107 @goa.bits-pilani.ac.in

Abstract:

We have attempted to design a digital notepad and e-book reader for the visually impaired. The main aim of this project was to design an affordable and portable system that will be convenient to take notes and replace the traditional method of using a slate and stylus. The project has progressed through all the stages in the embedded design cycle. The goal was formulated based on a literature review of available devices in the market. The experimental results have been recorded and stored for the first prototype.

Keywords*: visually impaired, blind, Braille, notepad, affordable, digital, technology*

Introduction:

Braille was introduced to Britain in 1861. In 1876, a French-based system with a few hundred English contractions and abbreviations was adopted as the predominant script in Great Britain. There are three levels of complexity in English Braille. Grade 1 is a (nearly) one-to-one transcription of printed English, and is restricted to basic literacy. Grade 2, which is nearly universal in print beyond basic literacy materials, abandons one-to-one transcription in many places (such as the letter ⠡ ⟨ch⟩) and adds hundreds of abbreviations and contractions. Both grades have been standardized. Our basic prototype has been built using the Grade 1 system. We plan to take it to Grade 2 in the future. The dots in Braille are in groups of six called cells. They are arranged in two rows of three columns each.
The slate and stylus method used by blind people requires them to make the impressions on the paper in the reverse order so that it can be read from left to right. It is a tedious and time consuming method.

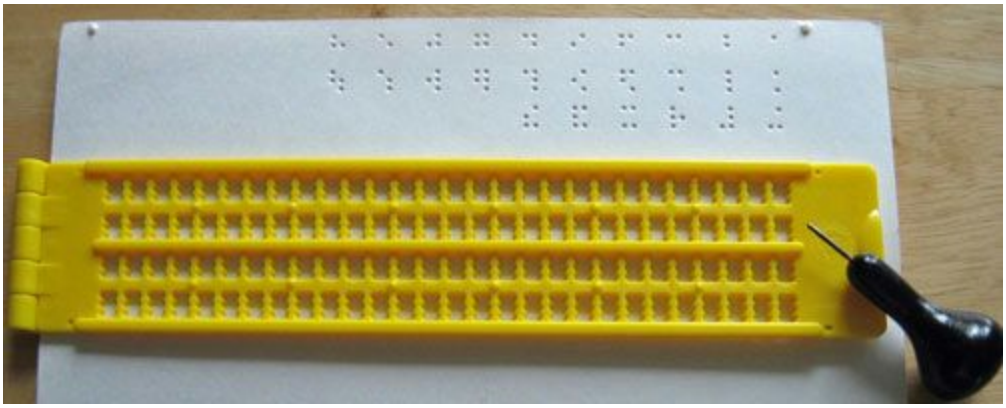

Figure 1: Slate and stylus for writing Braille

Technology has removed many barriers to education and employment for visually impaired individuals. Students with visual impairments can complete homework, do research, take tests, and read books along with their sighted classmates, thanks to advances in technology. Adults with visual impairments can continue to work and pursue a tremendous range of careers because of the use of computers and other devices. Some of the common features that we have replaced are speech recognition software, touchscreen with audio feedback and Qwerty keyboards. Speech recognition software is not a very reliable method of taking input. The performance varies a lot with a person's accent, gender, age. A good quality touch screen raises the cost of a device by a huge margin. Qwerty keyboards are a viable option as most blind people are familiar with them. However, if the size of a Qwerty keyboard is maintained large enough for user's comfort, it could hinder the portability of the device.
In this paper, we have explained the requirements that we had identified after a literature review. We had also designed a questionnaire to get feedback from potential users as part of the elaboration phase. We have then described the specifications of the device. The design flow and methodology are explained next complete with a flowchart and algorithm. Finally, the experimental results after the first prototype have been recorded.

Requirements:

The requirements of our device have been identified based on our analysis of a few devices and their shortcomings. We have also kept in mind our aim of minimizing the cost of the device. Hence we have decided to implement the basic and most essential features of a digital notepad while making it as user-friendly as possible.
Our idea was to conform as much as possible with the traditional Braille script only in a less tedious way. We have digitized the six Braille cells using buttons. The system gets input from these buttons. The first requirement of our device is to convert the Braille input to English characters. The concept is a simple dictionary code to map Braille patterns to characters. The next requirement is providing an audio output for each character typed in by the user. The implementation of these essential functionalities could be hindered by real-time constraints

could lead to a few restrictions on the user. The time of conversion from Braille to text and then text to audio must not exceed the typing speed if the user. Optimization of the data retrieval rates can increase the processing speed. A quad-core processor and 1GB RAM should not put major limitations on the typing speed. The issue of button bounce needs to be handled. This requires a slight delay between consequent pressing of a character(set of buttons). Besides the six basic buttons we need buttons for space, end of character, end of file, delete and so on. These function have been implemented using two buttons which are end-of-character and end-of-file. Other functions will be implemented using a combination of these buttons.
We also require sufficient memory to store the files created by the user and also the e-books for the audio reader. Optimal usage of memory can be done using compression and decompression of files. We require a bi-state switch that will allow users to shift from reading to writing mode easily and vice versa. In the reading mode, audio output and the scrolling per file must be synchronized.
An important requirement in a device like ours is handling unanticipated shutdown. The files must be saved before this happens. Other requirements that could be included are mentioned as part of our future work.

Specifications:

The functional and non-functional specifications of our first prototype have been described here. Some of the functional features can further be improved. Also the non-functional features can be enhanced to further reduce the cost. Specifications of the first prototype and any modifications have been mentioned here.
We have designed the notepad using the Raspberry Pi B+ module. It has a 1GB RAM and a 900 MHz quad-core processor. The Raspberry Pi is a robust microcontroller that comes equipped with a host of other features such as HDMI output, 4 USB slots, 40 GPIO pins, 15-pin MIPI camera serial interface, display interface and a graphics processor unit(GPU). While some of these features may be used to add more functionalities to the notepad, most of them are not required for our device like the HDMI, GPU, camera interface. The core functionality is all that we need and it is available in some low priced microcontrollers as well. For instance, the Raspberry Pi Zero priced at around a mere rupees 300($5) has a 1GHz core processor(as compared to 900 GHz of the B+ model) and 512MB RAM (as compared to the earlier 1GB). The 512MB RAM is shared with the GPU. Since we will not require the GPU, these features should suffice to give us a reasonable performance.

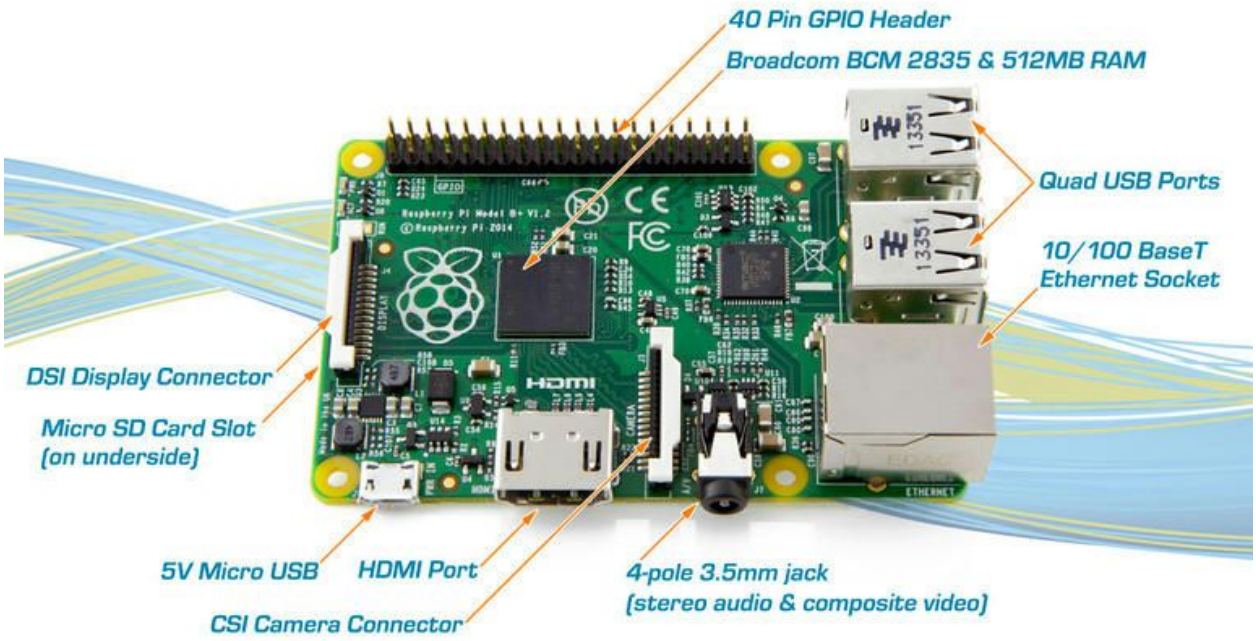

Figure 2: The Raspberry Pi 2 B+

We have used tactile push-buttons to act as Braille cells. The 6x6x4.5mm buttons have a 4 pin DIP. The small size will enable compact packaging. The audio output is got from a earphone that gets plugged into the audio jack of the Raspberry Pi B+. Any earphone with a 3.5 mm jack(used most commonly with mobile phone devices) will work well.
However, the Pi Zero module does not have a 3.5 mm jack port. It has a USB port for power and data. Adapters are available for 3.5 mm jack to USB conversion that cost around a hundred rupees. An 8GB SD card is used for storing written files and books.
Functional specifications of the device are the timing delay, size of the memory buffer, multi-threading, data compression and decompression. The timing delay introduced between any two character presses is 50 milliseconds. This resolves the bounce issue and prevents multiple output generation from a single button press. The size of the memory buffer used is 20 characters after which data gets pushed to the file and the buffer is emptied. When the end-of-file button is pressed, a high priority interrupt is invoked that compresses and saves the data file into a specified folder.

Design:

The Digital Notepad can be divided into 2 main modes: Read and Write Mode. The complete system is a Multithreading system to handle audio and button input simultaneously.
For the write mode, the first Thread is used to take in Braille input from the 6 Button (6 Dots) input. The user can press the 6 Dots in whichever combination of arrangement and then when

the user has finalized on the input , The “End-Of-Character” button is pressed. At this time, An event is created and the input values of these 6 button are read.

Advantages

1) The user can type in the combination in whichever combination he/she prefers.
   For Example: If the user wants to type in ‘d’, then he/she can type in any of the following order ->
   I. (Dot1)-(Dot2)-(Dot4)
   II. (Dot1)-(Dot4)-(Dot2)
   III. (Dot2)-(Dot1)-(Dot4)
   IV. (Dot2)-(Dot4)-(Dot1)
   V. (Dot4)-(Dot1)-(Dot2)
   VI. (Dot4)-(Dot2)-(Dot1)

2) Since, “End-Of-Character” is being listened to as an Event, the time and resource utilization is reduced as opposed to Polling for the same input. Another advantage is the rate of Power Consumption. The microcontroller is put to sleep by stopping all the clocks and reducing power consumption to a few microamps.
3) The user can press a button by mistake, and still rectify it by just picking up the button as long as the user does not press the End-Of-Character button.

Once the End-Of-Character button is pressed, the 6-bit input stream is converted to the corresponding integer, which ranges from 0 to $(2^6 - 1 = 63)$. Then this integer is used as index in the dictionary which is an array structurally to give the corresponding character. This character is added to a buffer of size 20. At a same time, another thread handles the TTS(Text-to-Speech) conversion which is headless i.e. The software inhibits within the embedded system and Hence, requires no server-client communication. This converts the character to the corresponding audio sound.

Advantages

1) The dictionary is lightweight as it is a 64B static array. The algorithm takes O(1) to convert a 6-Bit input sequence to the corresponding character.
2) This algorithm uses a buffer write which is very less expensive than a file write.
3) This avoids the need for server-client communication and the delays corresponding to the same for a per character basis.

Once, the buffer becomes full, The data is compressed and written into a g-zip file. Then the buffer is emptied and continues to take in more characters. This continues till the End-of-File button is pressed. This is a high priority Event, which invokes the third thread, which kills the former 2 threads safely, and ends the program after closing the file.

Advantages

1) FIle Compression leads to lesser disk utilization, faster read and write file operations and is independent of the byte order.

The Read mode is the interplay of 3 Main thread: File Decompression, Audio play and the TTS conversion. Once, the user enters into read mode, The audio plays the names of the available

files one after the other. The user then chooses the file and the audio for the corresponding file content begins. In this, The File decompression Thread read from the file and writes into a buffer. Then buffer is read to a particular delimiter like '.',':' etc. and this data is sent to the Mary TTS server handled by the TTS conversion thread. The audio response to this request is played by the audio play thread.

Advantages

1) While the audio for the previous data is being played, The next batch of text data is sent to the Mary TTS server. In this play, the resources are utilized more efficiently, and reduces the visibility of delay in text to audio conversion. Thus, playing the audio conversion smoothly. In other words, The Read Ahead algorithm is implemented to improve time and resource efficiency.
2) The server client communication is utilized here as text to speech conversion is a very heavy weight operation which if ran on an embedded system remotely would lead to high power consumption and performance inefficiencies.

Experimental Results:

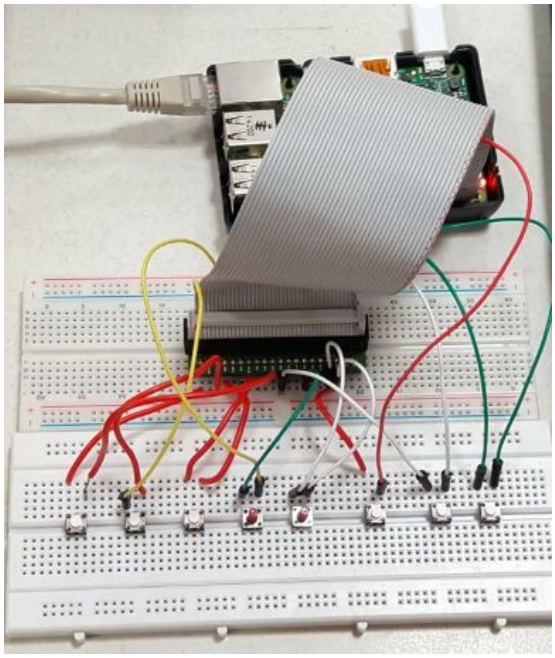

Figure 3: Experimental set-up for testing

We have tested the basic prototype at every stage of the design. The first stage of implementation was detecting button press using a Raspberry Pi. This was fairly easy because of the in-built GPIO libraries that Python provides. The only issue with button press was bouncing wherein the button acted as a spring and gave multiple outputs for a single button press. The issue was resolved by adding a time delay after each button press. The small oscillations of the button after the actual press went undetected because of the time delay. The

delay needs to minimized to have no affect on the writing speed as the next button can be pressed only after the delay time is over. We reduced the time from 2 milliseconds in steps of 0.5 and the smallest delay at which the system gives a good output was found to be 0.05 milliseconds.
The next step was detecting the pressing of multiple buttons at a time. We had to decide between sequential and simultaneous pressing of buttons for each character. Sequential pressing of buttons would require the users to press the button in a certain order. Or the system would have to consider all the possible sequences in which the patterns can be pressed. The conversion time would increase. Braille typewriters currently in use also employ simultaneous pressing of buttons per character.
The next step in the design was mapping the Braille input to English characters. Grade 1 Braille was used to create the dictionary for the same. For the audio conversion the initial design was to carry out the text to audio conversion only after the entire word had been typed in. While this reduced the overall conversion time from button input to audio output,errors made while typing if any would be detected only after the whole word had been typed in. And the text to speech conversion time taken does not create a major difference in the overall conversion rate. So an end of character audio feedback is used instead of the end of word audio feedback.
Once all the data has been entered the end of file button is used to save and quit the file.
The conversion times were calculated as follows.

Number of cells in character	Characters	Delay added
1	a	$0.05 + \Delta_{BrailleToText} + \Delta_{TextToSpeech}$
2	b, c, e, i, k	$0.1+ \Delta_{BrailleToText} + \Delta_{TextToSpeech}$
3	d, f, h, j, l, m, o, s, u	$0.15+ \Delta_{BrailleToText} + \Delta_{TextToSpeech}$
4	g, n, p, r, t, v, x, z, w	$0.2+ \Delta_{BrailleToText} + \Delta_{TextToSpeech}$
5	q, y	$0.25+ \Delta_{BrailleToText} + \Delta_{TextToSpeech}$
6	-	-

Table 1: Conversion time required based on number of cells per character

Here $\Delta_{BrailleToText}$ is the time the dictionary takes for mapping the Braille sequence to English characters, $\Delta_{TextToSpeech}$ is the time taken by the audio conversion software. The dictionary data structure takes a constant time irrespective of its size and it is minimal as compared to the delay added.

Conclusion and Future Work:

This paper describes a very simple and affordable digital notepad for visually impaired users. The basic requirements were identified, potential risks were taken into account and solutions were formulated. Periodic feedback taken at every stage of development was used to make decisions and improvements. An iterative development model is being used. All the stages in the design process will be further developed in every iteration. Some of these ideas have been mentioned here.
Firstly, the cost of the device can be significantly reduced if we use a simpler and less costly microcontroller like the Pi Zero.

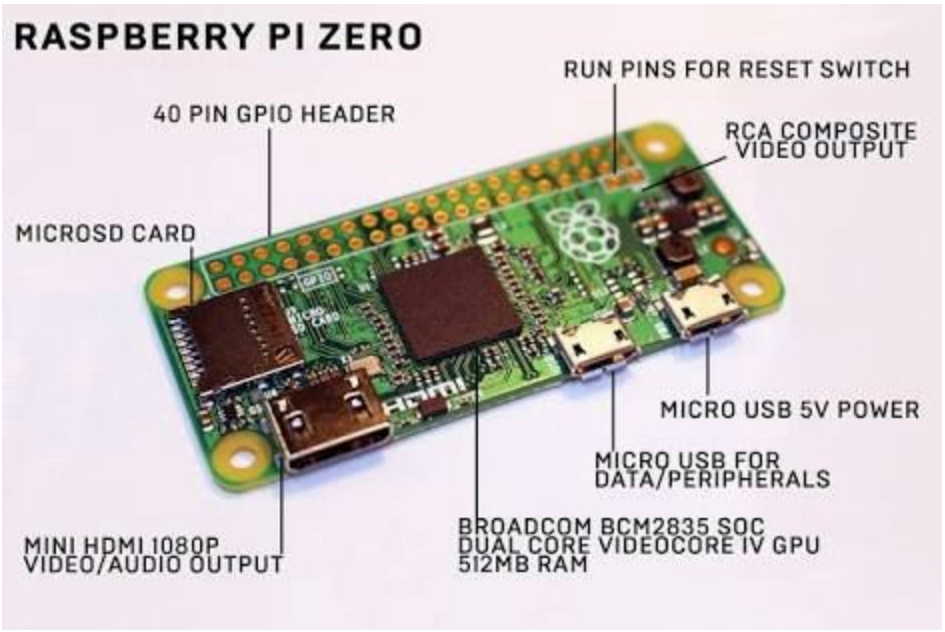

Figure 4: Raspberry Pi Zero

It has all the basic functionalities that we require. Secondly, we plan on upgrading our Braille dictionary to Grade 2 which has connotations for syllables and shortcut keys for frequently used words. Thirdly, we plan on adding a speaker module to the device so that users can also listen to the output without earphones. Lastly, an option between text to audio conversion at end of word and at the end of character can be included. This will enable users to choose when they want the feedback as per their convenience.

References:

http://www.afb.org/info/living-with-vision-loss/using-technology/assistive-technology/123
https://en.wikipedia.org/wiki/English_Braille
https://en.wikipedia.org/wiki/File:A_aesthetic_braille_typewriter_video.ogv

http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/RaspberryPi_GPIO/RaspberryPi_GPIO.html
https://help.ubuntu.com/community/TextToSpeech
https://www.raspberrypi.org/blog/raspberry-pi-zero/