Technological Renovation of the Wireless Braille Cellular Accessory for the Deaf-blind by Adding Speech Synthesis Through Bluetooth

by

César Augusto Pulido Plazas, BSEE

Thesis

Presented to the Faculty of the Graduate School

of the Pontificia Universidad Javeriana

in Partial Fulfilment

of the Requirements

for the Degree of

Magister en Ingeniería Electrónica

Pontificia Universidad Javeriana

June 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Considering reports on the inequality of opportunities for individuals with visual impairments, it has been shown that this community would benefit from support mechanisms for their daily work [15], purportedly widening their opportunities. In addition, in Colombia, 80% of the people with sight impairments lives in poverty [2]. Thus, this work of applied research seeks to bridge the inequality gap between the sighted and the sight impaired, facilitating the use of the cellphone as a communication device and an information gateway.

With a Bluetooth Braille reader, the deafblind can access electronic information wirelessly. Some of these readers can access text message, e-mail, and Internet applications. There are mobile apps that enable this functionality by sending through Bluetooth, the contents of the cellphone screen to the Braille cells of the reader. Then the user uses buttons on the reader to navigate and interact with the apps, including sending text to the apps. For Android phones, there is one such app called BrailleBack [9]; in iOS, there is VoiceOver [3]; and in Symbian and Windows Mobile, Mobile Speak [5].

The purpose of this project was to research, design and implement a Braille Bluetooth reader for cellphones with a row of Braille cells, a Perkins keyboard, and a cellphone app that does Text-To-Speech (TTS) synthesis of what is written using the Braille reader. The proposed voice feedback is an alternative to confirm what is written and could increase the speed of writing. Furthermore, this assistance tool has the potential to serve as an aid for learning the Braille writing system.

To implement voice feedback, one could use a TTS System on Chip (SoC), which has limited languages and can cost around US$60 [16] . An alternative would be to implement the TTS engine in the Braille reader's microcontroller, in which case a high computing capacity processor would be needed, increasing costs and energy consumption on the Braille reader.

The contribution of this project was the implementation of the TTS using an application for smartphones, that can generate the TTS of what is written with the Perkins keyboard of the Braille reader. This proposed solution is cheaper since it does not require a high computing capacity processor nor a TTS System on Chip. It is not limited to the quantity of languages that can be translated to text, since the smartphone supports a myriad of languages for TTS. Finally, it is simpler since there is no additional hardware modules such as in the case of using a TTS SOC, and it requires less programming than implementing or porting a TTS engine into the microcontroller.

1.1 Theoretical Framework

1.1.1 Speech Synthesis

The speech synthesis process consists of artificially reproducing human speech by a computer or machine that generates the corresponding vocal utterances. Generally, these expressions are generated in the instant at which they are required without being previously stored.

Applications of this technology are manifold. For example, it is used in environments where there are no suitable means for displaying characters or a certain type of message, such as a text message. Also, in situations where the user's eyes are occupied with other tasks or when there are severe vision problems, such as with blind people.

A speech synthesis is composed by the following modules [10].

Pronunciation Rules

Depending on the phrase or the meaning of the word in a given environment, the pronunciation and the intonation of each of the letters can change, so that the same phoneme can be generated with distinct combinations of sounds. To determine the correct pronunciation and intonation of the words, the following strategies are used:

Solutions based in dictionaries with morphological components, storing the quantity of phonemes necessary of a lexicon. The different forms of intonation are gained by rules of flexion, derivation, and composition.

Solutions based in pronunciation rules that are in turn based on the phonological information of the dictionaries. The database for this solution is usually smaller than the one required for the first method, but it tends to be less exact.

Prosody Generation

The degree of naturalness with which text is pronounced depends on prosodic features. Among these, there is intonation modulation, amplitude modulation and the

Table 1.1 Bluetooth LE's physical layer specifications

BLE Characteristic	**Value**
Frequency Band	2.4 GHz
Maximum Throughput	1 Mb/s
Communication Distance	> 100 m
Transmission Power	-20 a $+10$ dBm
Reception Sensibility	< -80 dBm
Bandwidth	2 MHz
Transmission Mode	Full Duplex

duration of the phrase which includes pauses coherent with the phrasing and the context. This features prevent ambiguity of meaning in phrases and the words in the text are better stressed. Knowing the syntactic structure of the phrase, one can calculate the prosody of the majority of the phrases.

Signal Processing

After calculating the rules for the text, this data is transferred to the signal generator. At this point is where the voice is synthesized. Sounds are generated by sections and then are joined together to form the phrases.

1.1.2 Bluetooth 4.0 LE Technology

Bluetooth is a short range wireless network technology, used to communicate devices. The main feature of Bluetooth 4.0 LE (also Bluetooth Smart), is that it defines a network protocol that can work on low cost radio circuits with distances over 100 meters, using a reduced quantity of energy as compared with traditional Bluetooth.

Bluetooth LE's physical layer specifications are summarized in table 1.1 [4].

1.1.3 Vision loss

To better understand the causes and consequences of loss vision, the International Council of Ophthalmology (ICO) distinguishes between four different main aspects that are classified as organ aspects and person aspects. [6]. Among the aspects of the organ, there are the structural changes of the organ that affect the health of the eye, from which defects are considered as diseases, disorders or injuries. The other aspect of the organ is related to the functional changes of the organ whose defects are considered as impairments since they affect the visual functions. Among the aspects of the person, one describes the skills and abilities of the individual that affect the functional vision and its defects are considered as disabilities. The last aspect refers to the social and economic consequences that affect the quality of life of the person with regards to their vision and its defects are described as handicaps and lack of social participation.

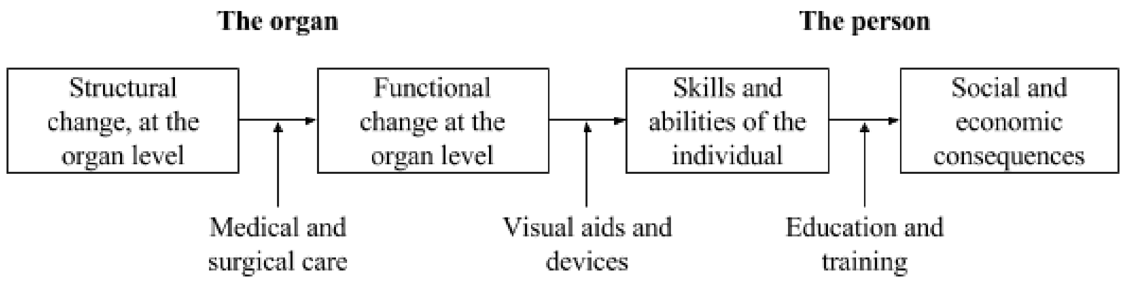

The organ
The person
Structural change, at the organ level
Functional change at the organ level
Skills and abilities of the individual
Social and economic consequences
Medical and surgical care
Visual aids and devices
Education and training

Figure 1.1 Vision Loss Interventions [6, table 2]

The ICO says that the aspects are correlated, some more strongly than others. In addition, they propose the strongest correlations to be depicted as links as in Fig. 1.1. Furthermore, they indicate that aspects on the left of the diagram are causes of the aspects to the right. More importantly, these links are not fixed and can be influenced by various interventions that permit rehabilitation. The ICO identifies that, "[f]or any given visual impairment, visual aids and devices can reduce the ability loss and improve the ability to perform various activities."[6, pp. 5]:

1.1.4 Braille System

Braille is a reading and writing system of embossed text used by the blind. It consists of an array of raised dots that represent letters, numbers, punctuation, and in some cases whole words. The basic symbol is called a Braille cell, which consists of 6 or 8 dots that are organized on 2 columns of 3 or 4 dots. Each dot is numbered as seen in Fig. 1.2. 6-dot Braille cells only use numbering from 1 to 6.

Historically, the traditional Braille cell has consisted of 6 dots which limits the system to representing 63 characters. Certain ASCII characters cannot be represented, and all numbers, upper-case letters, and lower-case letters must be prefixed with a special character since they share the same dot codes.

The 8-dot Braille cell system was necessary to represent the 95 printable symbols of the ASCII system so as to better read the contents of a computer screen, and thus it became popularized by the electronic Braille reader [7].

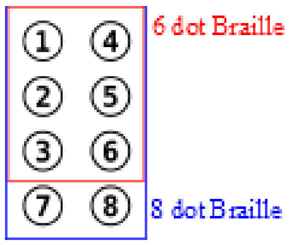


Figure 1.2 Braille cell dot numbering

However, text continues to be printed with 6-dot cells, given that the 8-dot system takes 33% more space on a line than with the 6-dot system.

To write in Braille, one can use a Perkins keyboard, which can contain 6 or 8 keys and a space bar. Each key represents a dot on the Braille cell and one or more keys must be pressed at the same time to generate a character. Depicted in Fig. 1.3, is the form and dot numbering of the Perkins keyboard.

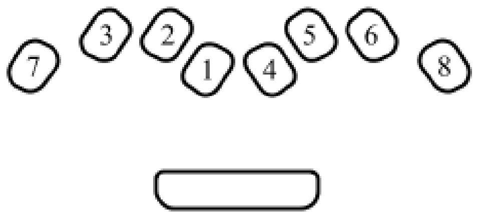


Figure 1.3 Perkins keyboard form and numbering

1.2 Project Objectives

General Objective:

Design and implement a Bluetooth Braille reader with TTS interaction using Android and mobile devices.

Specific Objectives:

1. Research and select the most suitable alternative for designing and implementing a Braille reader with a Perkins keyboard, a row of Braille cells, and Bluetooth connectivity.

2. Evaluate and implement an RTOS for the execution and administration of tasks and shared resources on the Braille reader.

3. Design and implement the hardware module and the software for the Perkins keyboard, Braille cells, and mass storage.

4. Implement an Android app for TTS in Spanish, corresponding to the written text on the Perkins keyboard.
5. Integrate through Bluetooth the Braille reader with the Android TTS app.
6. Validate and verify the operation of the entire system.

Chapter 2

System Design

The system's requirements respond to the necessities of the users of the Braille reader, which are summarized in the following list:

- Portable Braille readers that allow them to read text, such as books, on the go.
- Braille readers that aim to be comfortable and inconspicuous.
- Braille readers that open the door to the use of smartphones with tactile screens.
- Users that are deafblind, need tools that aid in their communication through TTS and Braille.
- Braille readers that present typed text as synthesized voice for ease of writing and for learning Braille.
- Braille readers that support the ASCII system's set of characters for use of the reader with computers.

With those necessities in mind, this project seeks to create a system consisting of a Braille reader with a Perkins keyboard and an Android app to allow the user of the Braille reader to communicate with the Android device over Bluetooth for TTS conversion. An important consideration of the system's design was the restrictions on the system's energy and memory footprint as well as the size of the Braille reader. These restrictions affect the usability of the device by the target community, given that the app's and the Braille reader's energy consumption affect the amount of time a user will be able to operate the Braille reader, running on battery, while in communication with the app. More over, the size and weight of the Braille reader affects its portability, so it was important to constraint the size and weight of the

Table 2.1 System Requirements

The device must include TTS while typing with a Braille keyboard.
The device must pair wirelessly with a cellphone.
The device must be able to send a receive text characters at least at the average typing speed of a person.
The device must fit its components in a box that protects them from handling, and protects the users from high voltages.
The device must be optimized for size and component placement.
The device must operate with rechargeable battery.
The device must use 8-point Braille cells and a 8 key Perkins keyboard for ASCII support.

printed circuit, the components, and the enclosing box. The complete system's requirements are summarized in Table 2.1.

The wireless Braille reader's hardware consists of a row of 12 Braille cells, each with a navigation button; a Perkins keyboard; 12 control buttons; a Bluetooth transceiver module; a flat battery that supports at least 7 hours of autonomous operation; a battery charger; a Flash memory for local storage; and a microcontroller running the firmware of the Braille reader. The firmware of the Braille reader consists of a group of tasks that run the communication processes for the Bluetooth module, the Flash memory, button input, Braille output and the user interface. An RTOS manages all the tasks.

In addition, an Android app uses the TTS engine from the Android API to translate text sent through Bluetooth from the Braille reader into voice. This method has the advantage of placing most of the computational load of the TTS process on the Android device. The Braille reader must only send the text through Bluetooth to the Android device, thus keeping a lower computational load for this process.

In addition, given that the use of smartphones in Colombia is among the fastest growing in the world, with Colombia ranked at 16 with a 23.2% growth for the year 2014 [8], the solution presented here is attractive. Since an ubiquity of smartphones indicates that many users may have a smartphone already, for many users all they need to do is install one more app on their smartphone.

Fig. 2.1 shows the main system diagram, showing the interconnection between the Braille reader and the Android device.

The general operation of the system is as follows: An Android application is loaded on an Android device with BLE support, which presents to the user a text

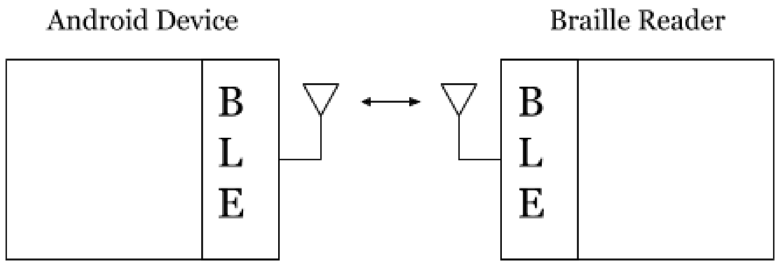


Figure 2.1 System block diagram including the wireless Braille reader and the Android device

file that can be read or written to. Once the Android device is paired with the Braille reader through Bluetooth, the Android app will send to the reader the text that must be shown on the Braille cells. Also, the Braille reader will send to the app, the text that is typed with the Perkins keyboard so that the app shows this text on the screen and optionally synthesize it to text.

Fig. 2.2 shows the block diagram of the Braille reader. In this implementation a Bluetooth module connects through UART to the microcontroller, sending and receiving the text packets between the Braille reader and the Android app. In addition, The Braille cells require to be controlled through 2 half SPI ports. Also, the memory module connects through the microcontroller through an SPI port to receive reading writing and other configuration commands and to send the data that needs to be read to the microcontroller. Additionally, the battery charger is configured through an I2C port from the microcontroller and it sends alarms to the microcontroller indicating problems such as battery failures or unexpected current surges. A USB connector indicates to the Battery charger the amount of current that it may source from it and allows the programming of the microcontroller. Finally a set of buttons make up the Perkins keyboard, which are each connected individually to a pin in the microcontroller to signal the pressing and depressing of the buttons.

The individual blocks of Fig. 2.2 are described as follows.

2.1 Microcontroller Selection

The Microcontroller that was chosen, required an UART port to communicate with the Bluetooth chip, as SPI port to communicate with the Flash memory, an I2C port to communicate with the battery charger chip, and two half SPI-like MOSI ports to control the Braille cells. Finally, GPIO ports to interconnect with the buttons. Of course, the ports needed to implement the different communication protocols individually and they needed to work simultaneously on the microcontroller.

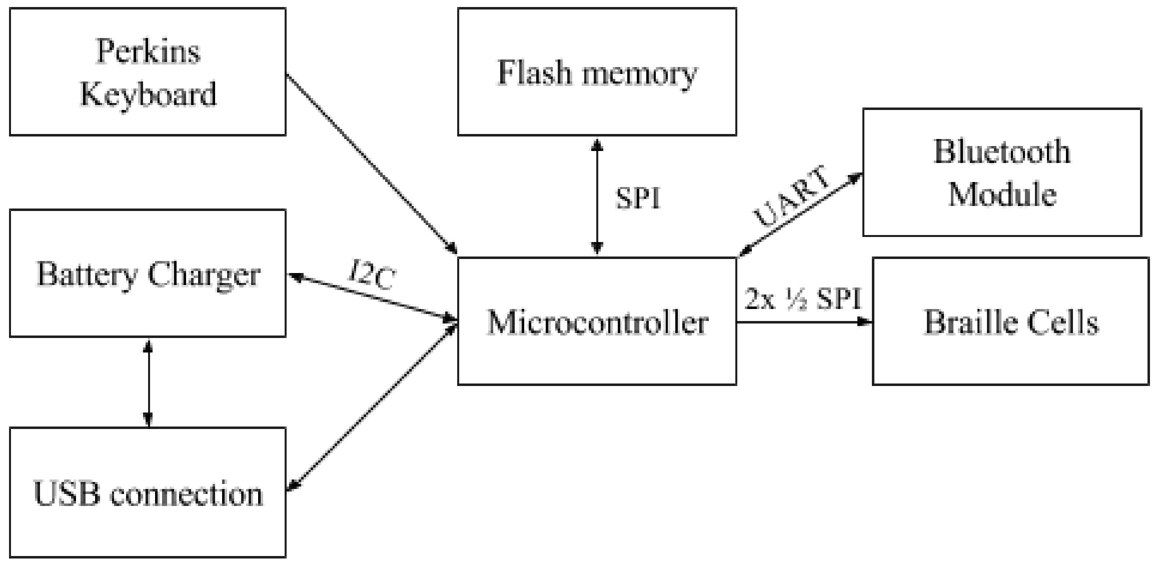


Figure 2.2 Block Diagram of the Braille Reader

The Texas Instruments MSP430F5529 [21] features an ultralow-power RISC core, and was selected to minimize size and power consumption. Among other things it was required that microcontroller work in the natural ambient temperatures of different places, this one working in a range of -40 to 85 degrees Celsius. Also, it was expect that the footprint of program memory to stay under 100 kB and the data memory under 25 kB, and the MSP430 sports 128 kB of Flash memory. Finally it was necessary that the microcontroller have a quiescent mode of operation available to optimize power consumption.

At the time of this writing, its main competitor was the STM32L151VC from STMicroelectronics [17], consuming 16 μA in sleep mode and 2.1 mA during execution at 8 MHz; in addition, the microcontroller spans an area of 256 mm^2. It also works in the temperature range of -40 to 85 degrees Celsius and it has 256 kB of Flash and 32 kB of RAM memory.

The MSP430F5529 is small microcontroller spanning an area of 196 mm^2, and advertised at 7 μA in sleep mode and 2.32 mA during execution at 8 MHz. The sleep mode low power consumption is achieved with a power management module that regulates the systems internal voltage and a clock tree that disables peripherals and modules that are not used. Also, there are excellent development tools, an abundance of design examples and it is a low cost option, making it a good choice for the design.

2.2 Peripherals

The Braille reader's Braille line displays text stored in the Flash memory, typed with the Perkins keyboard or received from the Android device. A set of navigation buttons over the Braille line allow the selection of menus that appear on the Braille line.

Besides the navigation buttons, and the Perkins keyboard there is a set of 6 control buttons that allow for reading and editing text, and enabling and cycling through Text-To-Speech modes. 6 navigation buttons for additional menu navigation and text browsing.

Finally, a Texas Instruments CC2541 BLE core based on the Intel 8051 processor relays the data to be sent from the Braille reader and the data received from the Android device, and manages all layers of the Bluetooth communication with the Android device.

Chapter 3

Hardware Implementation

3.1 Power Subsystem Design

The power subsystem consists of a battery charger, and three DC/DC voltage converters that output 2.7 V, 4.5 V, and 200 V. Figure 3.1 shows how the battery charger, the bq24295 from Texas Instruments, also controls the power source to the system, be it from the battery, the USB connection, or both, depending on the availability of each and the system's load.

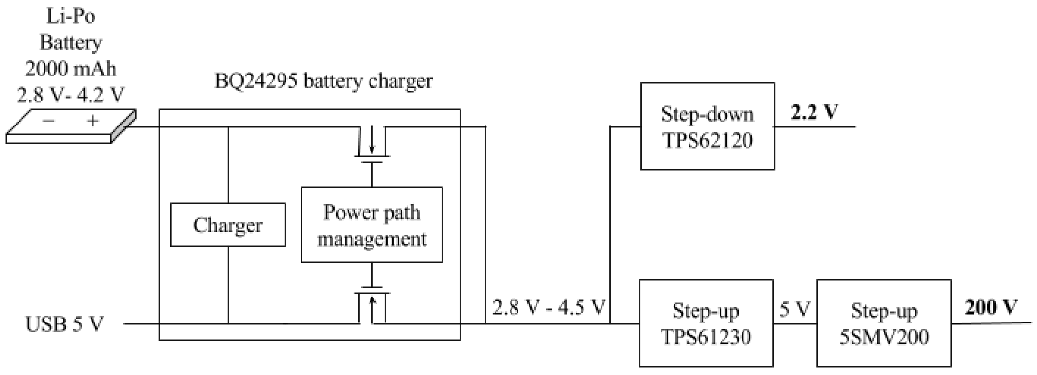


Figure 3.1 Power subsystem design

To find the required battery mAh, it was necessary to make an initial estimate of the system's average current consumption using calculated efficiencies for each of the voltage converters, and the current consumption of each module.

The BQ24295 has a low quiescent current of 4 mA, which allows for a 97% power efficiency on the output, which is unregulated [23].

As follows, the efficiencies of the voltages converters was calculated by simulating each converter using PSpice models of the converters with Texas Instrument's

WEBENCH Design Center.

3.1.1 2.2 V Step-down Converter

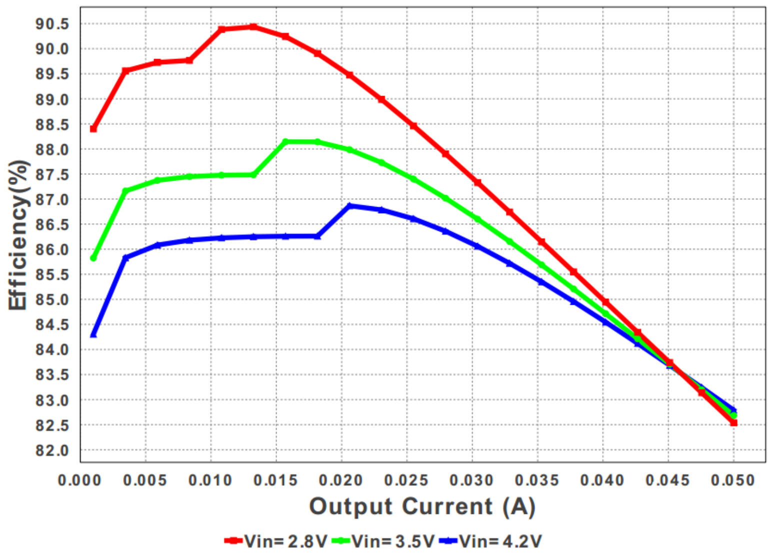


Figure 3.2 Efficiency vs. output current curves for the 2.2 V step down converter

To get 2.2 V, the TPS62120 step-down regulator was selected and its efficiency vs. output current profile was calculated as shown in Figure 3.2 [18]. Thus, given the current consumption ranges for the modules at 2.2 V in table 3.1, the efficiency of the converter was calculated to be between 82% and 90.5%.

Table 3.1 Current consumption for modules running at 2.2 V

Module	**Current consumption**
Bluetooth Module CC2541	0.4 μA – 22 mA[19]
Microcontroller MSP430F5529	3.9 μA – 2.6 mA[21]
Flash Memory AT45DB641E	0.4 μA – 18 mA[1]
Buttons	1 μA – 2 mA
TOTAL	5.7 μA – 48.7 mA

3.1.2 200 V Step-up Converter

The 200 V is necessary to power the 12 Braille cells. The estimated current consumption at 200 V of the device is summarized in table 3.2. Fig. 3.3 shows the efficiency of the 200 V step-up converter [14]. At 1.14 mA, the efficiency is about 35%.

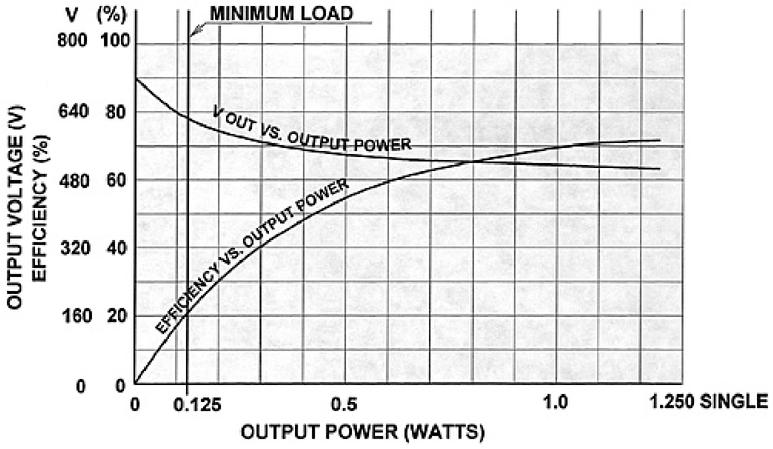


Figure 3.3 Efficiency vs. power curves for the 200 V step up converter[14]

Table 3.2 Current consumption for modules running at 200 V

Module	Current consumption
Braille cells	115.2 μA[11]
Braille cell controllers	600 μA[12]
Polarization resistor	425 μA
TOTAL	1.14 mA

3.1.3 4.5 V Step-up Converter

The 4.5 V converter is a TPS61230 [22] step-up converter by Texas Instruments whose efficiency was estimated to be between 91% and 99% for a current consumption of 150 mA given the efficiency vs. current curves in Fig. 3.4

3.1.4 Battery Required Capacity

An initial estimate of battery consumption was performed assuming a power consumption profile for a 3 second interval summarized in Table 3.3. Here, the current of the modules is normalized to the battery's nominal voltage of 3.7 V, and includes the additional current consumption due to the power system's inefficiencies. In addition, only the maximum current consumption for each module is considered—the quiescent current profile being negligible.

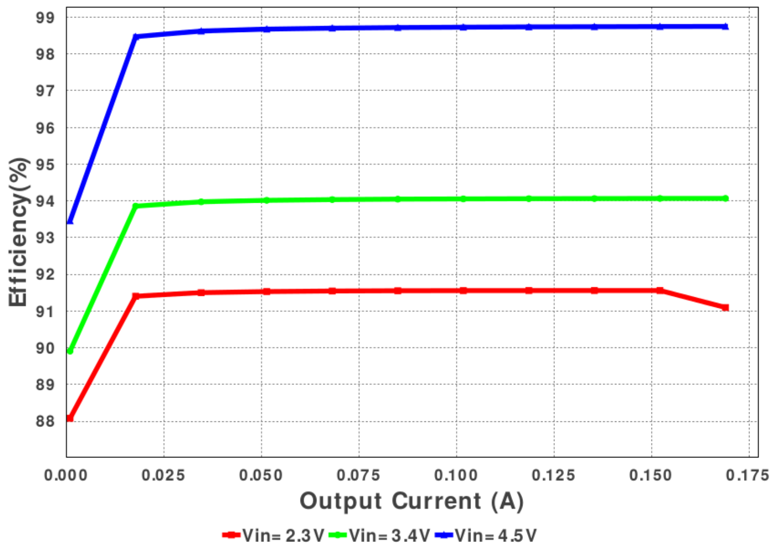


Figure 3.4 Efficiency vs. output current curves for the 4.5 V step down converter

Table 3.3 Assumed current consumption profile for system modules during a 3 second interval

Module	Current consumption	Time
Bluetooth	15 mA	1 ms
Flash memory	12 mA	1 ms
Buttons pressed	0.7 mA	1 s
Microcontroller	1.8 mA	5 ms
Braille cells	20 mA	3 s
Braille cell controllers	102 mA	3 s
Polarization resistor	72 mA	3 s

For a 7 hour autonomous operation with the battery fully charged, equation 3.1 gives an estimate of the battery charge capacity needed.

$$Q_{bat} = 7h \cdot \frac{I_{bt}t_{bt} + I_{flash}t_{flash} + I_{but}t_{but} + I_{msp430}t_{msp430} + I_{cells}t_{cells} + I_q t_q}{3 \quad \text{seconds}} \tag{3.1}$$

The current consumption is clearly dominated by the Braille cells. According to this equation, we need a 1360 mAh battery to maintain operation continuously. A 2000 mAh battery was chosen to maintain a safety margin since a precise estimate of power consumption was not available at the time of purchase. The MLP674361 battery by McNair is a 2000 mAh Li-ion batter with a small size footprint (63.5 mm x 44.2 mm x 7 mm).

3.1.5 Runtime Current Consumption and Verification

A sample acquisition current was measured on the Braille reader as seen on Fig. 3.5 for a period of 3s during execution. During this period, the Bluetooth device on the Braille reader advertised, then connected, exchanged its characteristics with the Android application, and began sending data packets at 4 packets per second. In addition, a 500 byte file was read from the memory. The Braille cells alternated the output on each dot at a rate of 4 times per second during the first half of the period, and then remained still. As expected, the current consumption is dominated by the Braille cells. The AC RMS current of the Braille reader is about 56 mA. Assuming that this AC RMS current consumption for the profile is the same for the total period for which the battery operates autonomously, with a 2000 mAh battery, the Braille reader can run autonomously for 36 hours with a full charge.

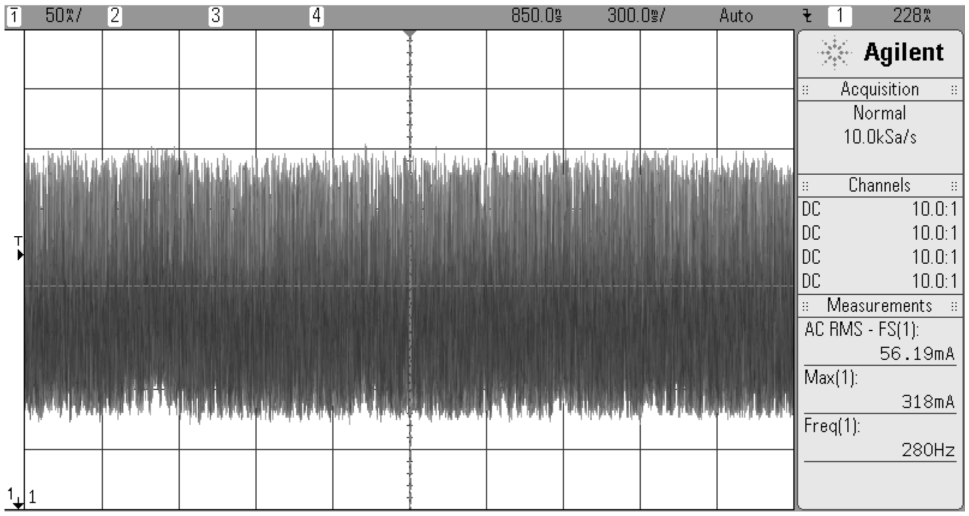


Figure 3.5 Current profile of the Braille reader for 3 seconds

Chapter 4

Braille Reader Firmware

4.1 Bluetooth Firmware

The Bluetooth module on the Braille reader is a single-device configuration device. It has a slave LL profile, meaning that the device advertises and receives requests to connect, which it may or not accept, and implements an ATT server profile exposing it's data for reading and/or writing.

In the following section, the GATT profile for the application is described.

4.1.1 Bluetooth GATT Profile

The GATT Profile defines the sub-procedures for using the ATT layer, that is the rules for exposing data known as "attributes". All of the pieces of data that are being used by the profile are called Characteristics and are described in Table 4.1.

Table 4.1 GATT profile

Handle	**Type**	**Value**	**Permission**
0x0001	≪ Primary Service≫	≪ GAP ≫	Read
0x0002	≪ Send Message	≪ Notify, Read≫	Read
	≪ Characteristic Declaration≫		
0x0003	≪ Value≫	≪ 6 byte array≫	Read
0x0004	≪ User Description≫	"TX Frame"	Read
0x0005	≪ Configuration≫	≪ Notify Config.≫	Read, Write
0x0006	≪ Receive Message	≪ Write≫	Read
	≪ Characteristic Declaration≫		
0x0007	≪ Value≫	≪ 20 byte array≫	Write
0x0008	≪ User Description≫	"RX Frame"	Read

Here there are 2 sets of important characteristics: the Send Message Characteristics and the Receive Message Characteristics.

Send Message Characteristic

The Bluetooth device receives packets from the MSP430 microcontroller through the UART interface which contains information about the commands and text to be sent to the Android app. For this purpose, the set of Send Message Characteristics define the form of the packets containing this information and the method of sending this information. In particular, the characteristic can be configured by the app to send notifications whenever a new package arrives for the app to fetch it. In this way, no new packets will be missed.

The packets that are sent are formatted as in Table 4.2.

Table 4.2 Send Message Frame Format

TYPE	SIZE	DATA	ID

The TYPE byte indicates if the packet that is being sent contains a text or a command. As of now, DATA is only up to 2 bytes long, so that SIZE, which indicates the bytes in data, is either 1 or 2. Since consecutive packets sent through Bluetooth BLE can arrive in different orders, an ID number composed of 2 bytes is used to organize the data on arrival.

Receive Message Characteristic

The Bluetooth device also sends packets to the MSP430 microcontroller containing the text to be shown on the Braille cells. Here, the Send Message Characteristic defines a write-only characteristic that receives the packets from the app that contain the text to be sent to the Braille cells.

The packet has the format shown in Table 4.3.

Table 4.3 Receive Message Frame Format

SIZE	DATA	ID

In this case, DATA can be up to 17 bytes long, containing the data to be shown on the Braille reader. The Bluetooth module forwards the packets to the MSP430 for further processing.

4.1.2 Preprocessor Definitions

Table 4.4 contains the preprocessor definitions that are included in the Bluetooth projects build.

Table 4.4 Preprocessor definitions on the Bluetooth module

Defined Symbol	**Description**
HALNODEBUG	When not set, it fills the code with asserts to aid during compilation, as well as code that halts during debugging on different errors. After debugging, the symbol should be defined to reduce size.
POWER_SAVING	Enables a task that takes the Bluetooth chip into low power mode for a short period of time to save energy.
HAL_UART_ISR	When set, it enables the interrupt handling routine of the UART.
HAL_UART_DMA	When set, it enables DMA for the handling the UART read and writes.

4.1.3 OSAL

A control loop runs tasks in a round-robin fashion with priority for each task, called the OSAL (Operating System Abstraction Layer). Currently the tasks that are included in the OSAL are the LL, the Hal, the HCI, the L2CAP, the GAP, the GATT, and the Application task. A loop verifies whether each task has an event to process and in this case, the necessary task's call backs are executed as shown in Fig 4.1.

4.1.4 Bluetooth Braille Reader Application: BleBraille2.c, BleBraille2.h

The Bluetooth Braille reader application configures the GAP, the GATT service, the HCI service. It initializes the services as soon as the OSAL starts the application and handles events from each of them. In particular, the GATT service send events every time the send message characteristic configuration characteristic is changed to activate or deactivate notifications, and if the notifications are activated, the GATT service sends an event whenever a send message characteristic value is read by the Android app. The HCI service is configured to enable the UART RX and TX with 8 bytes of software buffer each, with a hardware receive interrupt synchronization that calls a callback function. The callback function detects if we are running on a receive and send interrupt, and the algorithm runs a finite state machine that parses

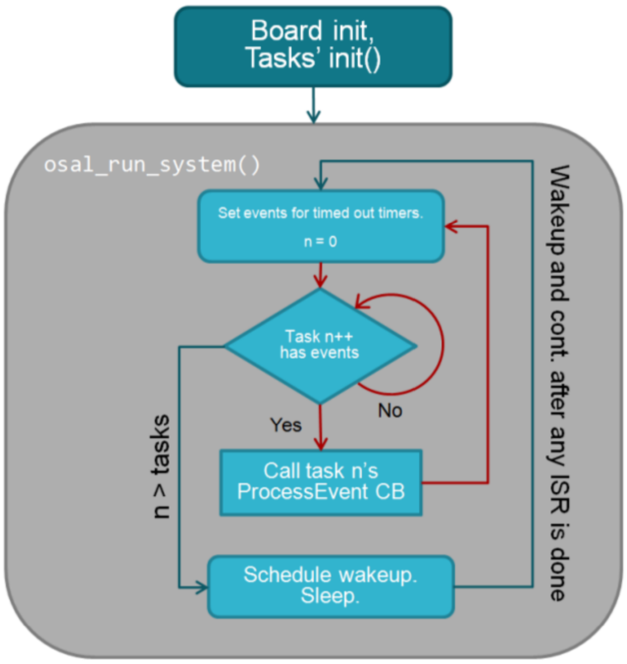


Figure 4.1 Operating System Abstraction Layer Task Loop[20]

the packets received. Whenever there is data to be sent the send buffer is filled, and whenever the interrupt occurs, the buffer is emptied.

4.1.5 Throughput Tests

When typing with the Perkins keyboard, proficient typists should be able to send characters to the Android application and not loose any data that is sent. For this reason it was important to measure the maximum throughput of the Bluetooth communication before data begins to be lost. To verify this, an Android tablet with a Broadcom BCM4330 Bluetooth Transceiver was used for the test, and was placed at a distance of 60 cm from the Braille reader. The android application that communicates with the Braille reader was loaded on the tablet with a modification to record the data received and to measure the time between packet arrivals. Also, the Braille device was set to send 500 packets, each containing a character representing something typed on the Perkins keyboard, at a constant rate. Finally, the Android application received the data and the rate of successful message reception was measured. Table 4.5 shows the packet send data rate as well as the percentage of successful packet reception.

As can be seen, the maximum communication throughput without errors for 500 packets can be approximated at 240 packets per minute. This corresponds to

Table 4.5 Packet Reception Success Rate

Packet Send Data Rate (Packets per minute)	Percentage of Successful Packet Reception
300	81%
267	93.2%
250	98.8%
240	100%
200	100%

a maximum typing speed of 240 characters per minute. For comparison purposes, research done to determine the typing speed of people shows that the average is 39 words per minute, and that the fastest typists can go up to 110 words per minute [13]. Given that in average, a word is composed of 5 characters, to type 240 characters per minute in a Perkins keyboard is comparable to typing at 48 words per minute, or above the average person who types on a regular keyboard.

4.2 Microcontroller Firmware

The MSP430 firmware runs on top of an RTOS, the TI-RTOS, which serves as an abstraction layer to the software written, and also administers shared resources.

4.2.1 TI-RTOS configuration: app.cfg

All of the RTOS services (semaphore, mailboxes, events, clocks, tasks, calendar, etc.) are defined and configured in the app.cfg file and on compilation by going through the app.cfg file. The advantage of having them all ready on compilation is that the code becomes less bloated with configurations, and instead it all remains in a single file.

4.2.2 Start-up: main.c

The main configures peripherals: the Braille, UART, button, battery charger and memory drivers, as well the system calendar. After configuration, it initializes the RTOS.

4.2.3 UART interface: uartBtMsg.c,uartBtMsg.h

The UART interface handles the sending and receiving of packets to and from the Bluetooth module. It send packets containing the text written with the Perkins keyboard or commands to the android app.

A mailbox pends on data that can be received from other tasks. It then pends on a binary semaphore that is posted every time the UART interrupts to indicate that is ready to send data. Once that happens, the packet is sent to the Bluetooth module through the UART peripheral

When packets are received from the Bluetooth module, an interrupt occurs which puts the data on a mailbox without pending. The microcontroller finite state machine handles the reception the reading of the mailbox.

Figure 4.2 depicts the flow chart of the logic described for the UART interface.

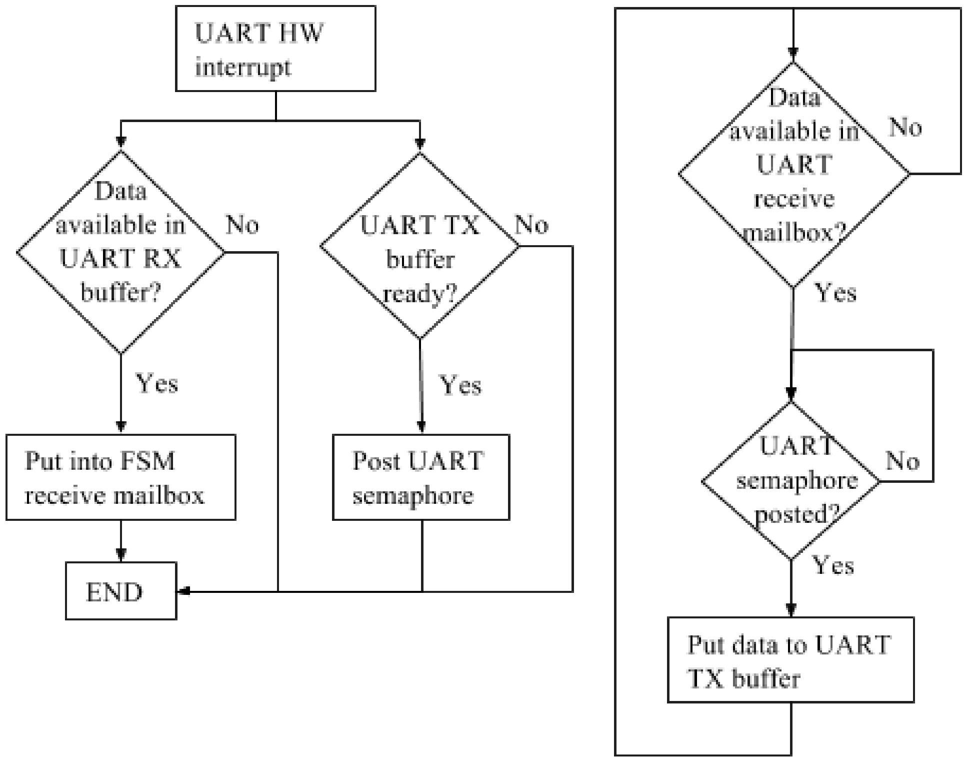

UART HW interrupt
Data available in UART RX buffer?
No
Yes
Put into FSM receive mailbox
END
UART TX buffer ready?
No
Yes
Post UART semaphore
Data available in UART receive mailbox?
No
Yes
UART semaphore posted?
No
Yes
Put data to UART TX buffer

Figure 4.2 UART Interface Flow Diagram

The UART process defines the transmission and reception frames in Table 4.6 and Table 4.7

Table 4.6 Transmit packet format

SOF	TYPE	SIZE	DATA	EOF

The start-of-frame (SOF) and end-of-frame (EOF) bytes are used as a synchronization window on the Bluetooth and microcontroller modules.

Table 4.7 Receive packet format

SOF	SIZE	DATA	EOF

4.2.4 Braille cells task: braillecells.c, braillecells.h

The Braille cells are configured to work with 2 half SPI ports, controlling 6 cells, one of them which was implemented as a virtual half SPI port.

One clock task implements the virtual SPI port, sending one byte at a time whenever it is initiated. Once it finishes, it disables itself and posts a semaphore indicating that is ready for sending a new byte.

A hardware interrupt indicates whenever the second SPI port is ready to receive data by posting on another semaphore.

Finally, the writer task is pending on a mailbox from packets that other tasks send, it turns on the Braille cells for reception and begins to transmit the data to the cells, synchronizing the sending with the semaphores mentioned above. After sending a packet, it disables the Braille cells for reception.

4.2.5 Buttons task: buttons.c, buttons.h

The buttons task, periodically reads the state of all buttons on the Braille reader. It detects when the keys are pressed and then depressed, to then send to various mailboxes the buttons that were pressed. Button bouncing filtering is done in hardware with a simple RC filter.

The mailboxes are read on the finite state machine of the device to handle them.

4.2.6 Memory task: mem.c, mem.h

The memory task implements the firmware to handle writes and reads to the memory by implementing a simple file system. The file system consists of a block containing a directory table and two sets of linked lists of blocks: one set is a linked list of empty blocks and the other are linked lists of blocks belonging to individual files.

The directory table can have up to 32 entries where 31 are files and 1 is the first empty block in the linked list of empty blocks. This limitation of entries is based on the size of each directory entry. Each directory entry consists of a file name of 12, null ended, 8 bit characters, and an index to the first block in the file, and an index to the last block in the file, each index made up of 16 bits.

Each block is 512 bits in size, where the first 2 bytes indicate the number of bytes used int the block, the next 508 bytes are the content of the block, and the last 16 bits are an index to the next block in the file. With this set-up there is no external fragmentation, only internal fragmentation.

A pair of mailboxes handle the writing and reading directly to the memory. A mutex prevents tasks that read and write to the memory, interrupt and work while other tasks are reading or writing to the memory in order to prevent fragmentation and of data.

4.2.7 Finite State Machine: ctrlFSM.c, ctrlFSM.h

The finite state machine works with 5 different states. The first one is the off state, which enables only one tasks that polls the state of the turn on/off button. While it is not polling, it allows the sleep task to run. When the button is pressed for a second, the system starts the buttons tasks and the Braille cells to start sending data to them.

The second state initiates a 1 second clock that detects changes on the calendar. It then begins to waits on a set of events. The first one indicates that in the calendar occurred in order to update the data sent to the Braille cells. A second one indicates that a set of buttons was pressed in order pan the calendar on the Braille cells and so it reads their state from the buttons mailbox. In the case that space bar was pressed, the finite state machine goes into the third state.

The third state displays a menu that is defined in linked structs each containing the data to display on the Braille cells and the action to perform in case the menu is selected. The menu is selected using the navigation buttons. Depending on the menu selected, the finite state machine can go back to the second state to display the clock, but it can also go to the fourth state to connect with the Android app, as well as going to the fifth state to read, write and create files on the Braille reader.

The fourth state asks the Bluetooth module to begin advertising and begins to listen and send on the UART module. The android application begins with a text file that can be begun to be edited and read from the Braille reader. As the user writes and pans the cursor, the Android application begins to send the text to display on the Braille cells. By pressing the 6 control keys at the same time, the user can exit the connection with the Bluetooth module to get back to the finite state machine.

The fifth state loads the list of files that can be opened on the Braille cells and the user may select the one they want to open. By pressing the select + shift button, the user may create a new file. In this case the user must enter the name of the file and by pressing select + shift, the user may begin to write and read on this new file.

Figure 4.3 shows the aforementioned FSM as a mealy FSM. For simplicity, the state changes to the OFF state are shown with red arrows. They have the same inputs, outputs and point to the same state. In addition, the green state changes to the MENU state, have the same inputs and outputs.

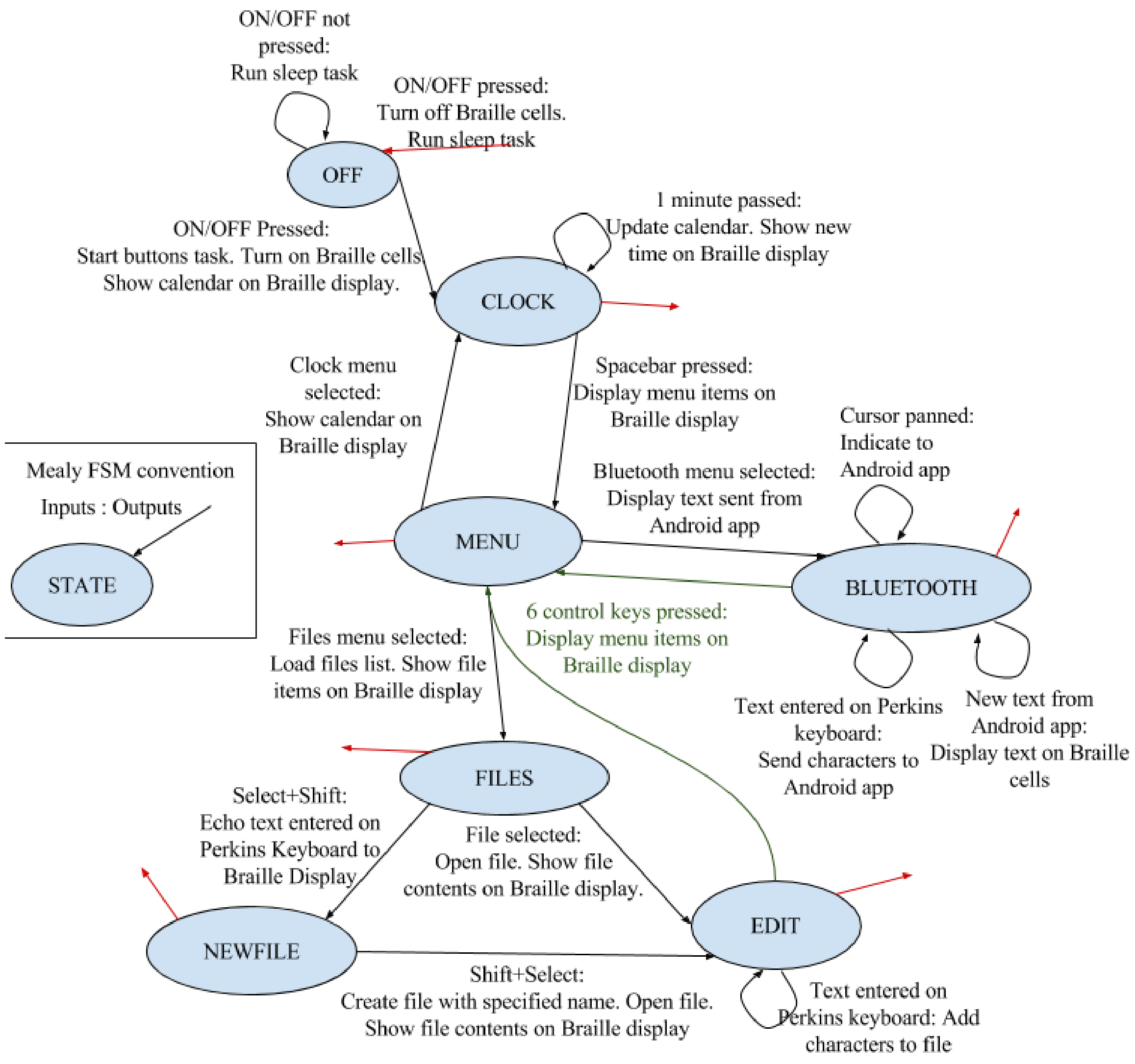

ON/OFF not pressed: Run sleep task
ON/OFF pressed: Turn off Braille cells. Run sleep task
OFF
ON/OFF Pressed: Start buttons task. Turn on Braille cells Show calendar on Braille display.
1 minute passed: Update calendar. Show new time on Braille display
CLOCK
Clock menu selected: Show calendar on Braille display
Spacebar pressed: Display menu items on Braille display
Cursor panned: Indicate to Android app
Mealy FSM convention
Inputs : Outputs
STATE
MENU
Bluetooth menu selected: Display text sent from Android app
BLUETOOTH
6 control keys pressed: Display menu items on Braille display
Files menu selected: Load files list. Show file items on Braille display
Text entered on Perkins keyboard: Send characters to Android app
New text from Android app: Display text on Braille cells
FILES
Select+Shift: Echo text entered on Perkins Keyboard to Braille Display
File selected: Open file. Show file contents on Braille display.
NEWFILE
EDIT
Shift+Select: Create file with specified name. Open file. Show file contents on Braille display
Text entered on Perkins keyboard: Add characters to file

Figure 4.3 Main Control Mealy Finite State Machine

4.3 Memory footprint

Mentioning the memory footprint is important in order to understand the limitations that can be defined in future projects that would use the firmware that was developed. In particular, in an update of the microcontrollers used in this project it would not always be desirable to increase in the memory capacity but instead, other facts such as lower power consumption, size, or ease of use. Table 4.8 summarizes the memory consumption on the main microcontroller and the Bluetooth chip.

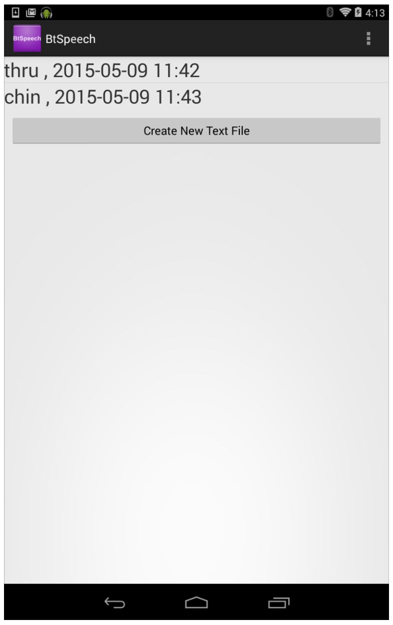
Table 4.8 Braille Reader Memory Footprint

Memory Type	**Size (kB)**
MSP430 code	82
MSP430 stack and data	23
CC2541 code	110
CC2541 stack	45

Chapter 5

Android Application

When the application is started, the user is presented with a text entry box, ready to enter text to the file. The user may go back to the main screen which is shown in Fig. 5.1. In this screen a list of files that have been created is shown. The user may open on of those files to continue reading or editing, or he may press the 'create new file' button to create a new file, as shown in fig 5.2.



thru , 2015-05-09 11:42
chin , 2015-05-09 11:43

Figure 5.1 Screenshot of main activity running

The text file editor and reader is shown in fig 5.3. The user may select from the options to change the language in which the text is read and also select the modes in which the text read:'Speak all' cause the application to read the whole text in

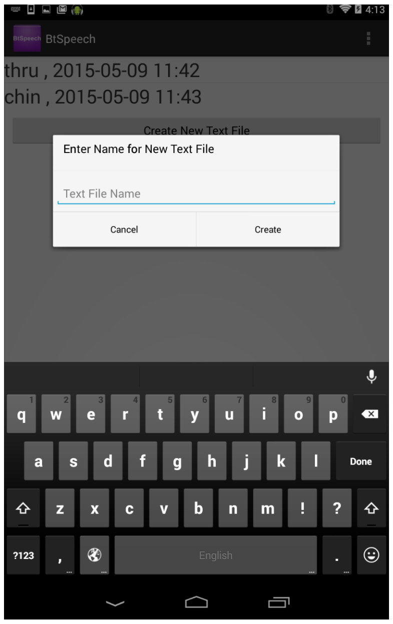


Figure 5.2 Screenshot of how to create a new file

the selected language. With 'Speak each letter' each letter that is typed, is read out load; this option can be set in combination with the 'Speak each word' option to read each word that is typed individually. Finally the 'Dont Speak' option disable the text to speech functionality.

When the user is ready to leave, they may press the back button on the application and a dialog will appear asking them if they want to save the changes to application. On returning to the main screen, they will see their newly created file on the list of files.

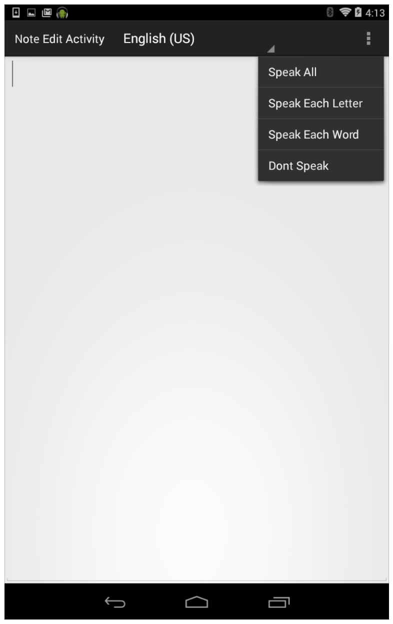


Figure 5.3 Screenshot showing the text editor

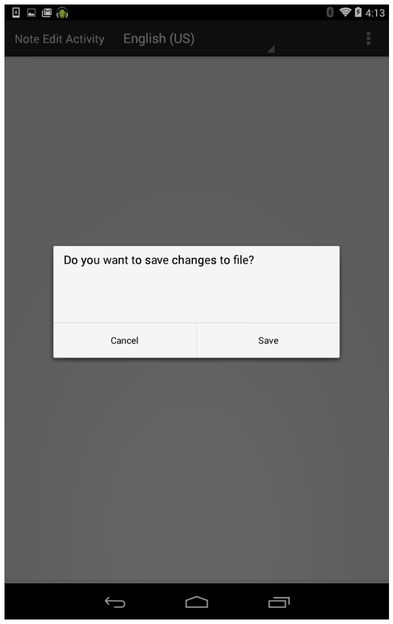


Figure 5.4 Screenshot showing how the user may save the changes to file before exiting

Chapter 6

Validation with blind users

The device was given to blind users at the Instituto Nacional Para Ciegos (INCI) to be tested so as to receive feedback on the functionality of the system. The group consisted of 3 blind adults that had used electronic Braille cells, and one of them which had used a Perkins keyboard in the past.

During the process of verification, each user was set up with the Android application running on text input mode and they were able to enable or disable the TTS feedback, save their text input and open a different file for text input or reading.

The group blind users of the INCI voiced the following recommendations:

- Since the text to speech on the Android device was being performed for every word that was written. The users asked for a letter-by-letter text to speech feedback of what they were writing. In addition they wanted to have both features at the same time. These features were included in the final design of the system.

- Also they asked that it would be good if they could configure the speed at which the text to speech voice is speaking, so this was also included in the final prototype.

- The users also wished that they could use the Braille reader to control other apps besides the one the device was made for. However there is still more work to do before this is possible.

- The user that had used the Perkins keyboard in the past had a hard time pressing on buttons of the Perkins keyboard on the device since he mentioned that the buttons were too stuff. The buttons could not be changed at the moment since the pad configuration on the printed circuit board for the Perkins buttons was incompatible with any buttons that were available at the time.

- They also noticed that there was a mechanical problem with the text being shown on the Braille cells that were on the side of the device, whereby their pressing too hard of the cells with their fingers pushed them down, so an attempt was made to make the cells more mechanically robust to downward pressure.

The test were not able to be made until an enclosing box for the components was 3D printed, due to dangerous high voltages that the device works with. The box was tediously designed to be able to fit all the components, to remain small and to be robust to pressure and breakage. Fig. 6.1 shows the Braille reader inside the box as it was used to be tested by the blind users.

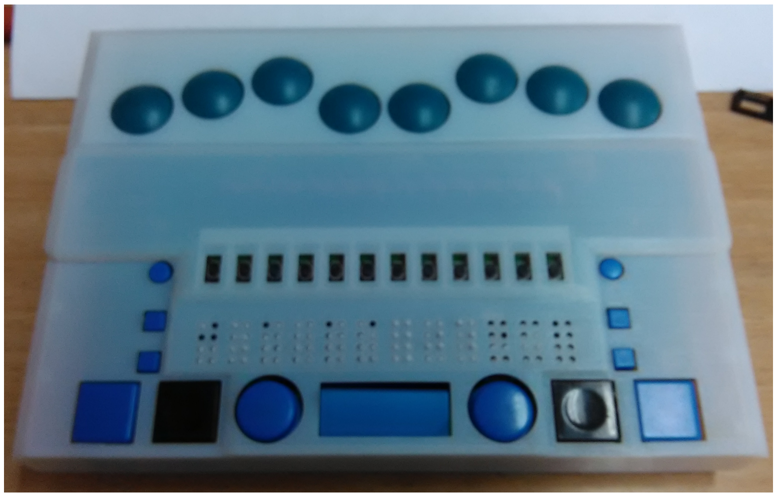


Figure 6.1 Final Braille reader prototype inside it's 3D printed box

Chapter 7

Conclusions

In this thesis, it was presented the information concerning the design and implementation of a Bluetooth Braille reader with TTS interaction using Android. The method of implementing TTS through Bluetooth using an Android device is cheaper, simpler to implement and has more features than previous implementation of TTS for Braille readers.

In addition, the most suitable alternatives in terms of component selection, hardware design, and software design were taken into account so as to ensure low memory, energy usage and a small size. At 10.5 x 14.5 x 2.75 cm, the device had a suitable size for the blind users who found it to be small enough for portable use. In addition, the device surpassed the desired autonomous operation with a single charge, lasting more than a day with a full charge. Also, the device was tested by blind users and their recommendations were considered and featured in the final design.

The communication throughput of the device with the Android app through Bluetooth limits the typing speed of an user to 240 characters per minute, or a comparable speed of 48 words per minute which remains suitable for an average user. Since the Braille writing system was devised to permit contractions while typing, it is likely that the comparable speed in words per minute is higher than 48, and so more research to learn the average typing speed of Perkins users is needed. In any case, this speed limitation might pose a problem with typists who expect to be able to type much faster than 240 characters per minute.

Future work for this project can focus on the recommendations by the blind users, specifically on work on the firmware that allows the Braille reader to control the Android system. Moreover, Furthermore, given that the Braille cells were shown to dominate the power consumption of the device, more work can be put into lowering this current consumption, perhaps by controlling the amount of time that the 200 V DC-DC converter remains online. Finally, to decrease PCB size, the programming header for the microcontroller should be removed and be programmed by USB only.

Appendix A

Glossary of Acronyms

- API: Application Programming Interface
- ASCII: American Standard Code for Information Interchange
- ATT: Low Energy Attribute Protocol
- BLE: Bluetooth Low Energy
- DMA: Direct Memory Address
- GAP: Generic Access Profile
- GATT: Generic Attribute Profile
- GPIO: General Purpose Input Output
- HAL: Hardware Abstraction Layer
- HCI: Host Controller Interface
- INCI: Instituto Nacional Para Ciegos
- RTOS: Real-Time Operating System
- RISC: Reduced Instruction Set Computer
- L2CAP: Logical Link Control and Adaptation Protocol
- LL: Link Layer)
- MOSI: Master Out, Slave In
- OSAL: Operating System Abstraction Layer

- SoC: System on Chip
- SPI: Serial Peripheral Interface
- TTS: Text-To-Speech
- UART: Universal Asynchronous Receiver/Transmitter

Appendix B

List of DVD Contents

The following list describes the contents of the folders that are included in the DVD that acompanies this project, which are the appendixes.

- ADT Workspace: Source code for the Android application
- BleBraille2 and Profiles: Source code for the Bluetooth module firmware
- DesignSpark: Source code for the schematic and PCB design of the Braille reader
- CCSV6: Source code for the microcontroller firmware

Bibliography

[1] *64-Mbit DataFlash (with Extra 2-Mbits), 1.7V Minimum SPI Serial Flash Memory*. Adesto Technologies Corporation, Inc., January 2015. URL http://www.adestotech.com/wp-content/uploads/DS-45DB641E-027.pdf.

[2] Marisol Angarita and Sara Rubio. Realidad y contexto situacional de la pobación con limitación visual en colombia. una aproximación desde la justicia y el desarrollo humano. Technical report, INCI, Bogota, Colombia, 2011. URL http://www.inci.gov.co/observatorio-social/informes-estadisticos/otros-estudios-e-investigaciones?download=53:censo-discapacidad-visual.

[3] Apple. Voiceover, 2015. URL https://www.apple.com/accessibility/ios/voiceover/.

[4] *Physical Layer Specification*. Bluetooth SIG, June 2010.

[5] Code Factory. Mobile speak app, 2014. URL http://codefactoryglobal.com/app-store/mobile-speak/.

[6] August Colenbrander. Visual standards: Aspects and ranges of vision loss with emphasis on population surveys. In 29^{th} *International Congress of Ophthalmology*. International Council of Ophthalmology, Sydney, Australia, April 2002. URL http://www.icoph.org/downloads/visualstandardsreport.pdf.

[7] Judy Dixon. Eight-dot braille. Technical report, Braille Authority of North America, September 2007. URL http://www.brailleauthority.org/eightdot/eightdot.pdf.

[8] eMarketer. Smartphone user growth in south africa among fastest worldwide. December 2014. URL http://www.emarketer.com/Article/Smartphone-User-Growth-South-Africa-Among-Fastest-Worldwide/1011752.

[9] Google. Brailleback readme, October 2013. URL https://code.google.com/p/eyes-free/source/browse/trunk/braille/brailleback/README.

[10] Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. Prentice Hall, 2008.

[11] *Modul P20*. Metec AG. URL http://www.ti.com/product/tps61230.

[12] *16-Channel Serial to Parallel Converter with High Voltage Backplane Driver and Push-Pull Outputs*. Microchip, September 2008. URL http://ww1.microchip.com/downloads/en/DeviceDoc/hv509.pdf.

[13] Teresia R. Ostrach. Typing speed: How fast is average: 4,000 typing scores statistically analyzed and interpreted. Technical report, Five Star Staffing, Orlando, Fl, 1997. URL http://orbitouch.com/wp-content/uploads/2012/03/Average-OrbiTouch-Typing-Speed.pdf.

[14] *SERIES AV/SMV SURFACE MOUNT and PLUG-IN*. PICO Electronics, Inc. URL http://www.picoelectronics.com/dcdclow/pe64.htm.

[15] Helen Simson, Deborah Gold, and Biljana Zuvela. An unequal playing field: Report on the needs of people who are blind or visually impaired living in canada. Technical report, CNIB, Toronto, ON, 2005. URL http://www.cnib.ca/en/research/projects/Pages/needs-study.aspx.

[16] Sparkfun. Emic 2 text-to-speech module, July 2012. URL https://www.sparkfun.com/products/11711.

[17] *Ultra-low-power 32-bit MCU ARM-based Cortex-M3*. STMicroelectronics, March 2015. URL http://www.st.com/web/catalog/mmc/FM141/SC1544/SS1374/LN1041/PF255518.

[18] *15V, 75mA High Efficient Buck Converter*. Texas Instruments, July 2010. URL http://www.ti.com/product/tps62120.

[19] *2.4-GHz Bluetooth^{TM} low energy and Proprietary System-on-Chip*. Texas Instruments, June 2013. URL http://www.ti.com/product/cc2541.

[20] *Texas Instruments CC2540/41 Bluetooth^{TM} Low Energy Software Developer's Guide v1.3.2*. Texas Instruments, 2013. URL http://www.ti.com/product/cc2541.

[21] *MSP430F5529 Mixed Signal Microcontroller*. Texas Instruments, May 2013. URL http://www.ti.com/product/msp430f5529.

[22] *TPS6123x High Efficiency Synchronous Step Up Converters with 5-A Switches*. Texas Instruments, October 2014. URL http://www.ti.com/product/tps61230.

[23] *BQ24295: I2C controlled 3 A single cell USB charger with adjustable voltage and 1.5 A OTG*. Texas Instruments, January 2015. URL http://www.ti.com/product/bq24295.