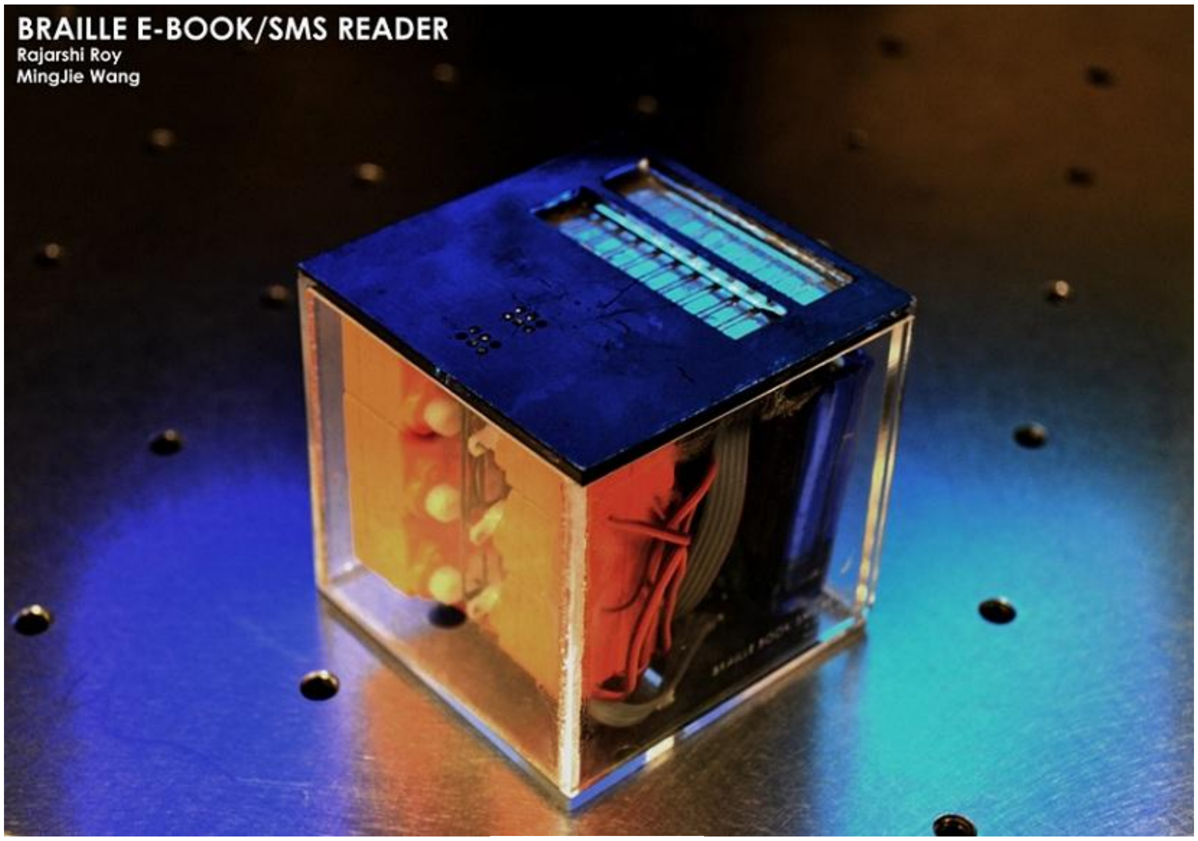
PORTABLE REFRESHABLE BRAILLE DISPLAY



By

Mingjie Wang

Rajarshi Roy

Final Report for ECE 445, Senior Design, Spring 2012

TA: Jane Tu

02 May 2012

Project No. 10

Abstract

This report documents the design, construction and verification of the titular device, which is able to read eBooks that are stored in microSD cards, read SMSes from mobile phones over Bluetooth, and display them as Braille. A parts list and corresponding costs are also provided so as to provide an estimate on the manufacturing capital required.

Contents

1. Introduction

Current commercially available electronic Braille readers (aka refreshable Braille displays (RBDs)) are based on piezoelectronics, making them very expensive – prices range from $1,000 to $15,000 [1], [2]. In addition, most are designed to be computer peripherals and thus are not portable. Using sub-micro servos instead, we have brought down the cost by an order of magnitude while keeping the size small. Coupled with rechargeable batteries, users can easily carry our device while on the go.

The device possesses a microSD socket for accessing stored ebooks, as well as a capacitive touch interface for browsing them. The device can also interface with mobile phones via Bluetooth and a custom application, which will capture SMS for display in Braille. The device currently consists of two Braille cells, and is designed such that more cells can be added with only minor modifications to the entire design. All these serve to make the device more accessible to the visually impaired.

Figure 1 below shows the various modules of our device. Our target is to have Braille cells whose dimensions conform to an established, widely used specification. The refresh speed of the cells must be at least as fast as the mean reading speed of 7.5 characters/s [4]. Since the device is to be portable, we aim to create a rechargeable device with at least one hour battery life.

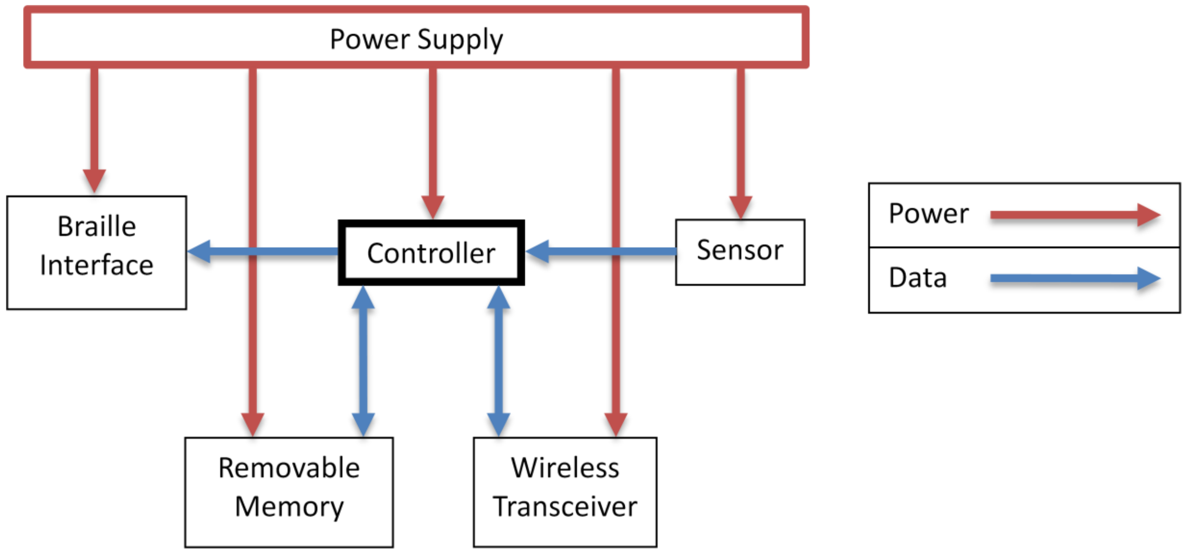


Figure 1.1

1.1. Requirements

Braille Interface (BI)

BI.1: Braille cells to meet Braille size specifications as laid down by the National Library Service for the Blind and Physically Handicapped (NLS) of the Library of Congress [3]:

0.057" (1.45mm) – dot base diameter
0.092" (2.34mm) – intracell dot spacing
0.245" (6.23mm) – intercell dot spacing (same line)
0.019" (0.48mm) – dot height

a) Dimensions and placement of holes for Braille pins on top plate match specifications to within 5%.
    i. CAD design of top plate matches the dimensions exactly.
b) Servo rotation produces vertical displacement of 0.019"±5%.

BI.2: At most 0.133s for Braille cell to refresh. Time includes

a) microprocessor latency – moment from parsing of ASCII characters to sending of PWM signals.
b) servo latency – moment from receiving PWM signal to moving pins to the desired height.

Controller (CO)

CO.1: Encodes alphanumeric characters into control signals for servos

CO.2: Receives data from external sources.

a) Removable memory – Serial Peripheral Interface Bus (SPI).
b) Wireless transceiver – Serial Port Profile (SPP) or Host Controller Interface (HCI).
c) Sensor – Inter-Integrated Circuit (I^2C) interface.

CO.3: Performs gesture recognition on sensor data.

CO.4: Voltage gauging and auto shutoff.

Sensor (SE)

SE.1: Each pad of capacitive touch sensor detects user touch.

Removable Memory (RM)

RM.1: Read from memory with no errors.

RM.2: Memory does not become corrupted with time and multiple accesses.

Wireless Transceiver (WT)

WT.1: Receives from Bluetooth devices that are at least 1m away with no errors.

a) Minimal radio interference.
    i. No electrical components, surface/internal traces, GND planes or exposed biases are located below the transceiver's antenna and shielding connections.

Power Supply (PS)

PS.1: Provides 3.3V at 735mA throughout 1 hour of continuous operation.

PS.2: Rechargeable. PS.1 is still met after recharge.

2. Design

2.1. Braille Interface

It consists of a plate on top of two stacks of six servos. Each servo is responsible for lowering and raising a pin, which will emerge through perforations on the top plate to form a Braille dot. The perforations serve as guides for the pins, restricting their motion along the vertical axis and together they form two six-dot Braille cells which users will touch. The dimensions of the perforations will meet BI.1. Sub-micro servos are chosen for their small size and weight and the ability to maintain their positions when no power is supplied. This reduces power consumption as compared to mechanisms where power is required to maintain the state of the Braille dots at all times. Figure 2.1 shows the mechanical principle of operation. Calculations (see following subsection) indicate that the required dot height of 0.019" and refresh rate of at least 7.5 characters/s can be easily achieved with the setup; this is supported by actual measurements (see verification section).

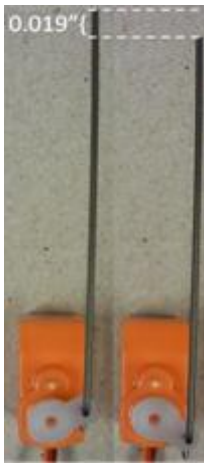


Figure 2.1

The servos are rated at 3V-4.8V, and it was experimentally determined that the lowest operating power supply voltage is 2.7V, after which the servo's movement becomes jittery. Thus we need to ensure that supply voltage powering the servos must never drop below 2.7V.

In order for the device to be able to accommodate an arbitrary number of Braille cells, the dimensions of each servo must be smaller than that of one Braille cell. Figure 2.2 shows that this is indeed possible.

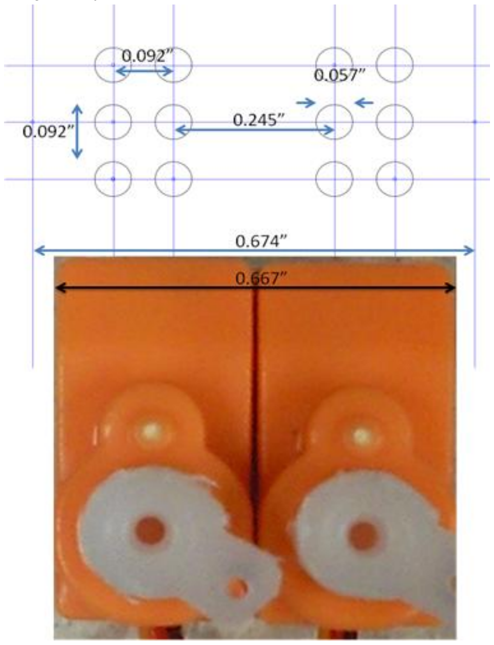


Figure 2.2

To create the top plate as well as other pieces of the chassis, a 2D drawing of the pieces is created in Inkscape. It is then fed to a laser cutter, which cuts the pieces out of an acrylic sheet. Superglue is used to join the pieces together. The servos are also superglued together to create stacks, which are then hotglued to the top plate. Hotglue is used in this case so that the stacks can be removed (via heating with hot air tool) should debugging/testing be required. See Figure A.1 for a cut-away view of the constructed setup.

2.1.1. Required and Achievable Braille Cell Refresh Rate:

Braille dot height as per specification is $0.019" \approx 0.0004826\ m$. If Braille dot is level with face plate when at rest, then this is the height delta, d that the servo needs to produce. Since servo produces rotational motion, this translates to finding the angle delta, θ. Due to geometry of the mechanism, the arm will be moving about the horizontal so as to minimize θ while maximizing d. Figure 2.3 illustrates the quantities involved.

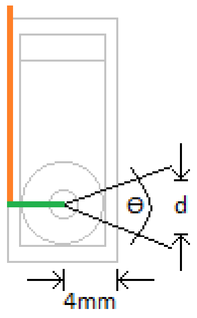


Figure 2.3

To minimize space, length of servo arm (green) will be reduced to half of servo body width. Further decrease in arm length will result in reduced pin (orange) movement due to geometry of mechanism.

Using cosine rule:

$$\theta = \cos^{-1}\frac{2\times 0.004^2 - 0.0004826^2}{2\times 0.004\times 0.004} \approx 6.92° \tag{2.1}$$

The speed of the motor is advertized to be $0.08s/60°$. Therefore the time taken to achieve θ:

$$\frac{0.08}{60}\times 6.92 \approx 0.00922\ s \tag{2.2}$$

Time allowance for latency:

$$0.133 - 0.00922 \approx 0.124\ s \tag{2.3}$$

Latency could be due to microcontroller processing time. Microcontroller has a clock speed of 8MHz, and thus such latency is in the order of microseconds – too small to impact response time. Assuming time taken to achieve θ, including latency, is 0.01 s, then potential refresh rate is 1/100 = 100 characters/s

2.2. Removable Memory

The purpose of a removable memory in this design is to store e-books. A micro-SD card will be used as removable memory. This component was chosen as compared to other memory alternatives such as the USB memory stick due to certain reasons. Firstly, SD card can be interfaced at 3.3V (as compared to 5V for USB), which is the voltage level of all the IC components in our design. Secondly, the micro-SD card supports the SPI protocol [5] which is supported by the ATMEGA328 of the controller on pins PB2 (CS), PB3 (MOSI), PB4 (MISO), PB5 (SCK) [6]. Finally, the micro-SD card can store a large amount of memory (2GB in our case) and is interface-able with most computers. The SD card chosen for this project is the SanDisk 2GB Transflash MicroSD Memory Stick Card.

2.3. Wireless Transceiver

The purpose of a wireless receiver in this design is to receive SMS data from a mobile device. The Bluetooth communication protocol was chosen since mobile devices can communicate wirelessly to close range peripherals via Bluetooth. The Arduino Bluetooth Module [7] was chosen as the Bluetooth adapter for the design. Figure 2.5 depicts the module.

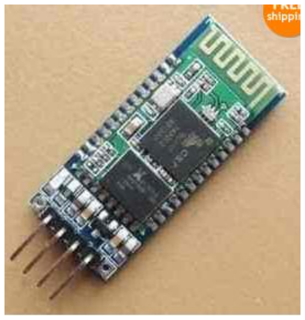


Figure 2.4
Arduino Bluetooth Module

The operating range of the device is 3.3V to 5V and thus can be interfaced to our design without the need for voltage level shifting. This device uses the Serial protocol which is supported by the ATMEGA328 on pins PD0 (RX) and PD1 (TX) [8].

2.4. Sensor

The purpose of the capacitive touch sensor is to allow the user to scroll forward and backward through text (e-book or SMS). Also, the sensor should let user change books and finally toggle the SMS mode off after reading an SMS.

The user interface consists of a left slider for scroll through text/SMS and a right slider for scrolling through book. The two sliders must be pressed together for greater than two seconds to toggle the SMS mode off.

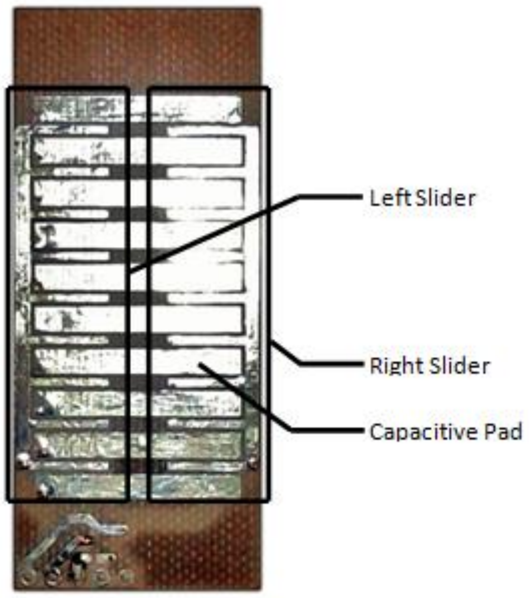


Figure 2.5
Capacitive touch surface with 12 sensing pads

The MPR121 Proximity Capacitive Touch Sensor Controller was used for the capacitive sensing from the capacitive pads. The operating voltage of this IC is from 1.71V to 3.6V [9]. Thus, this IC was suitable for our 3.3V system. Also, this IC provides an I2C interface which is supported by the ATMEGA328's PC5 (SCL) and PC4 (SDA) pins. Finally, the MPR121 IC has 12 channels of capacitive sensing for connection to 12 capacitive sensor pads. The capacitive touch surface is implemented on a PCB, with each copper pad serving as a touch surface. As seen in Figures 2.5 & 2.6, the sensor pads have been organized to 10 pads (#2-#11) for vertical slider resolution and 2 (#1, #12) pads interleaving the vertical pads for left or right slider press determination.

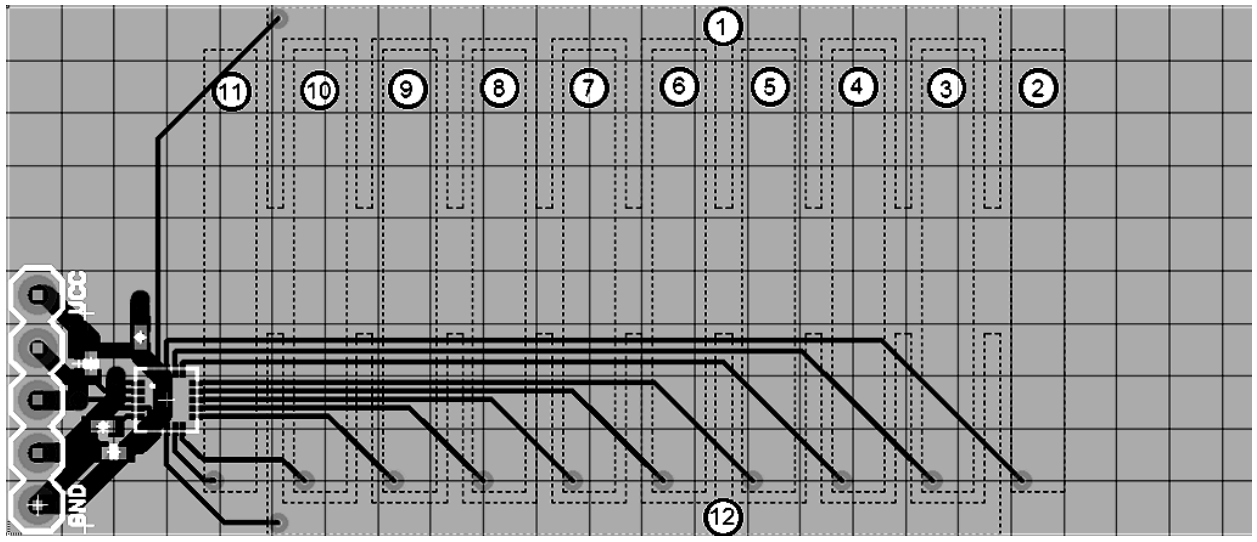


Figure 2.6
PCB layout of capacitive sense board

As seen in Figure 2.7, pins ELE0 to ELE11 of MPR121 connects to the capacitive pads. The ADDR pin is grounded to fix its I2C address to 0x50. A 0.1uF decoupling capacitor is added between VCC and ground. Also, 10K pull-up resistors are added to the I2C pins SCL and SDA since these pins are open collector pins. The output port that connects to the "sense" controller board contains power supply to the MPR121 IC: VCC = 3.3V and GND. The port contains the I2C bus pins SCL and SDA. The port also contains the IRQ pin which serves as an interrupt enable for the controller board whenever the state of a capacitive pads changes.

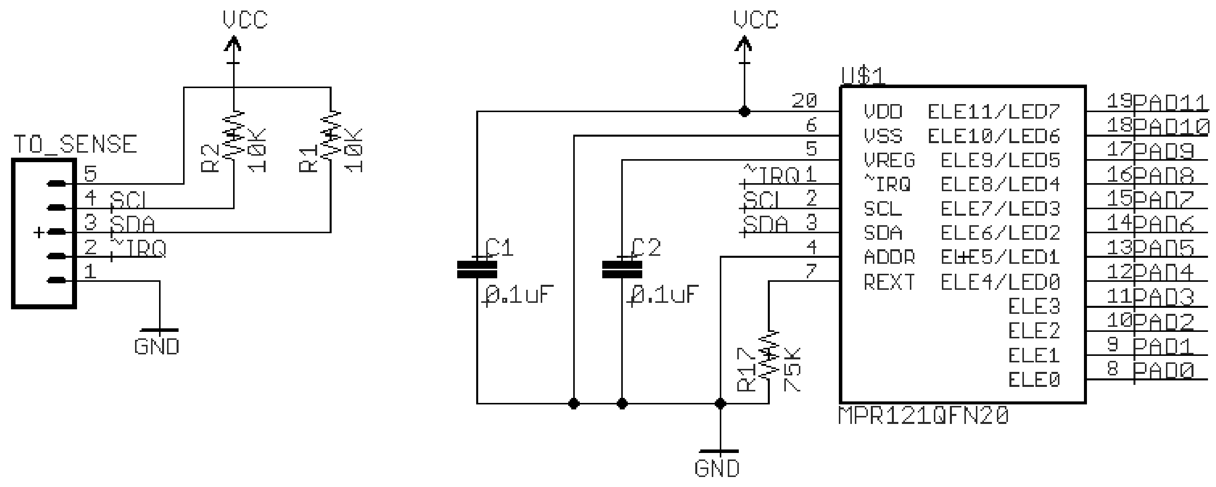


Figure 2.7
Circuit schematic of capacitive sense board

2.5. Controller

The controller is implemented as two separate PCBs. One, which we denote as the sense board, is responsible for interfacing with the wireless transceiver, removable memory, and sensor. The other, denoted as the control board, is responsible for interfacing with the power gauging IC and the 12 servos. ATMEGA238 is chosen to be the microcontroller as it is compatible with the Arduino Software Libraries [10]. Due to the limited number of control channels of the IC, it is not possible to use one ATMEGA238 to perform all the aforementioned functions. Therefore each board has its own microcontroller to direct all operations. To display Braille text, the sense board would fetch ASCII text from the removable memory or the wireless transceiver and send them to the control board via serial interface. The control board will then convert the ASCII characters into servo control signals to actuate the Braille pins. Thus is apparent that the design allows additional Braille cells to be added easily – one control board will be added for every two new cells. New boards will be connected to the same serial port and be given a unique ID, allowing the sense board to address each control board individually.

2.5.1 Sense Board

The purpose of the sense board is to direct the control board to display alphanumeric characters on the Braille interface based on data read from the micro-SD card or received from the Bluetooth receiver. It is also responsible for performing gesture recognition based on data from the from the capacitive touch board.

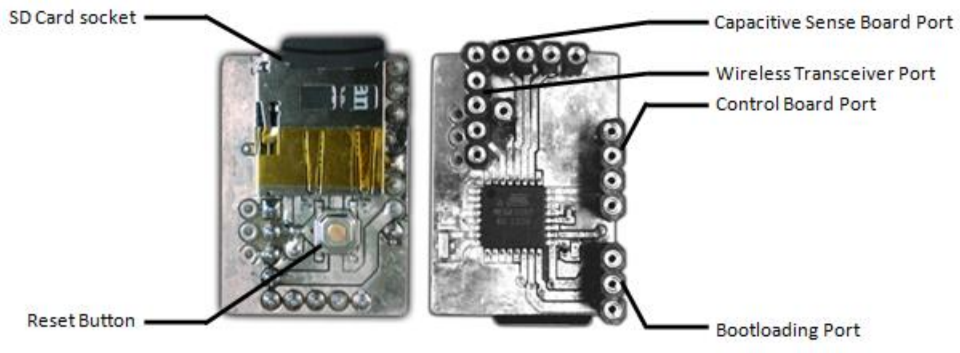


Figure 2.8
Sense board PCB

As mentioned in the previous sections, the SD-card socket, capacitive sense board port and the wireless transceiver port connect to the SPI pins, I2C pins and Serial pins of the ATMEGA328 respectively. The sense board communicates with the control board via the pins PC0 and PC1 of the ATMEGA328. This communication cannot be conducted over the Serial pins since the Serial pins of the ATMEGA328 are used to communicate with the wireless transceiver. To communicate over these two digital pins, the SoftwareSerial Library is used [11]. To boot-load the ATMEGA328 microcontroller, the bootloading port connecting to pins PB3-PB5, PC6 of the IC is used. A reset button is connected to pin PC6 for resetting the microcontroller during programming and debugging. A green LED is connected to pin PB5 for use during debugging as an indicator of functionality.

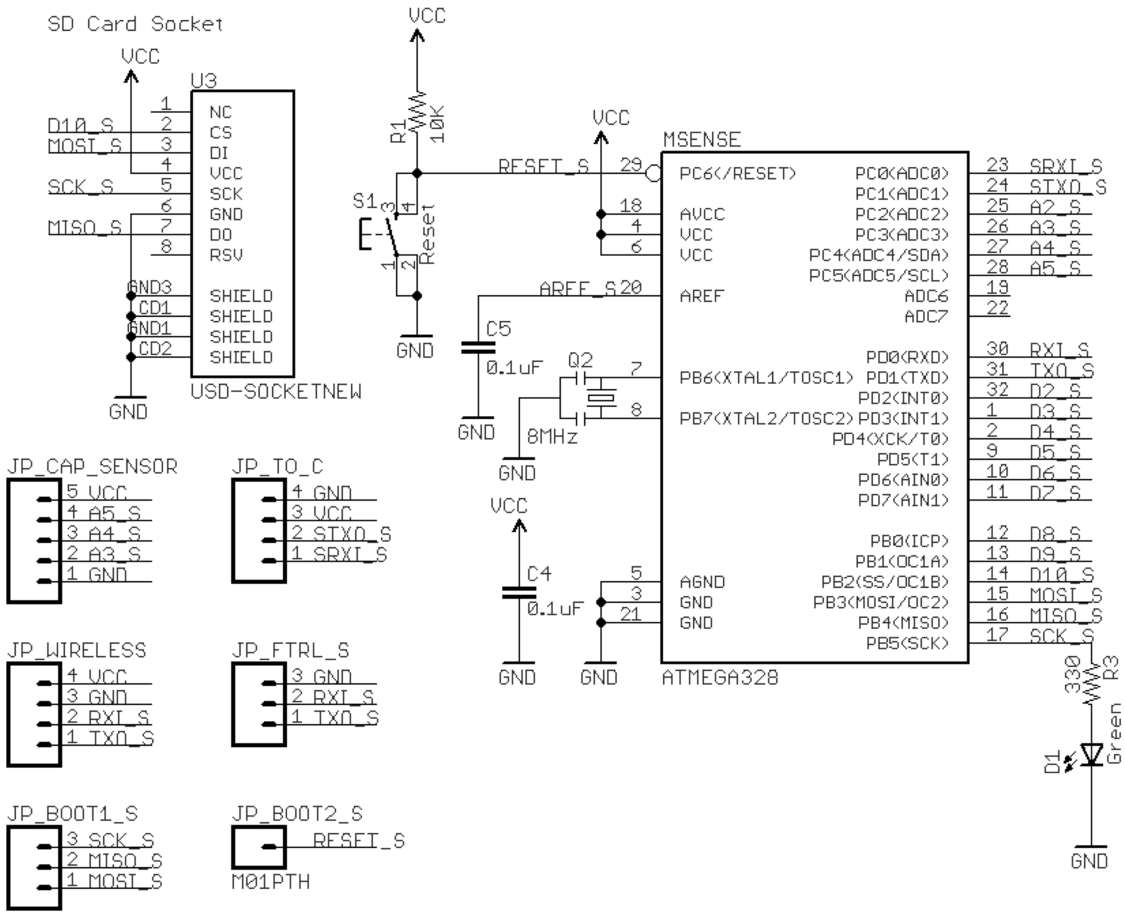


Figure 2.9
Circuit schematic of sense board

2.6. Power Supply

The power module consists of two 3.3V 1A voltage regulators, one 3.3V 500mA voltage regulator, a fuel gauging IC and a two-cell 7.4V 800mAh 15C rechargeable lithium-polymer battery (LiPo). The device is designed such that all modules are powered at 3.3V. This is because communication with the microSD card utilizes the SPI interface, which operates only at 3.3V while all other modules have a wider operating voltage range inclusive of 3.3V. If a different supply voltage is used, we would have to up and down-convert the voltage of the data signals on the SPI interface. Each one of the two 1A voltage regulators is responsible for powering six servos, while the 500mA voltage regulator is responsible for powering all the other modules. Multiple regulators are used so as to better insulate the rest of the circuit from the high and irregular power consumption of the servos – it was found experimentally that six servos can draw up to 2.1A at 3.3V (See Figure A.2). Note that the maximum current output of the 1A regulator is 2.5A [8], which is higher than the largest current drawn and thus is appropriate for our application. While a single high current output voltage regulator can be used, it makes for a less robust system, since sudden high current drain by the servos can cause fluctuations in the regulator's output voltage and adversely impact the performance of all modules sharing the same power line. Each 1A regulator has an enable input that is connected to the microcontroller on the control board, allowing us to cut off power to the servos to conserve power. The fuel gauging IC enables voltage and battery life monitoring. LiPo is chosen for its low weight, slim form factor, and high discharge capability. The first two factors contribute favorably towards the portability of our device, while the third factor caters to the servos' high power needs. The entire power module less LiPo is located on the control board. This is because the board is responsible for sending PWN signals to the servos, for controlling their power supply (via the voltage regulators' enable input) and for communication with the fuel gauging IC. Thus it would be convenient from the PCB design standpoint to place all corresponding ICs on the same board. A consequence of this design is that the control board would need to have a power port so as to supply power the sense board.

2.7. PCB

2.7.1. Size

The size of the PCB is constrained by the size of our device. The figure below depicts a 3D model that gives an estimate on the size of our device. Only the servos, Braille top plate and LiPo battery are included since these are the components that take up the most space. It's clear from the diagram that our PCB should be no larger than 5cm X 4.1cm.

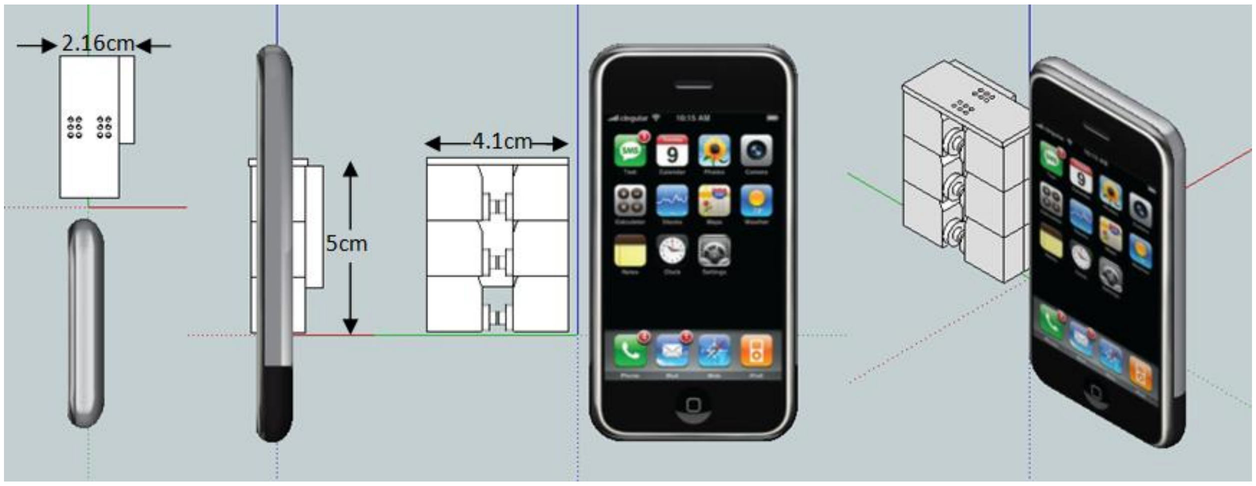


Figure 2.10

2.7.2. Parts Placement

To minimize radio interference, the Bluetooth module should be positioned such that no electrical components, surface/internal traces, ground planes or exposed biases are located below the its antenna [12]. This means that the module can only be placed at the periphery of the board.

Decoupling capacitors (capacitors connected in parallel to an IC on the VCC line) must be positioned as close as possible, electrically, to the IC [13]. This enables the capacitor to best achieve its purpose of reducing current and voltage spikes that is experienced by the IC.

Ports should be positioned at the edge of the board so as to allow easy access for connectors.

Parts should be positioned such that the number of crossing airwires (EAGLE's representation of unrouted wires, denoted by yellow lines) is kept to a minimum. An example of this is shown in Figure 2.11. When airwires cross, it means that the trace connecting the pins must either pass through a via, or that it must make a detour around the IC. Both are undesirable since the former leads to more soldering work (because the Electronic Service Shop does not produce plated vias), while the latter can obstruct other traces and thereby lead to even more crossing airwires. However, note that some crossings are unavoidable due to geometry of the connections.

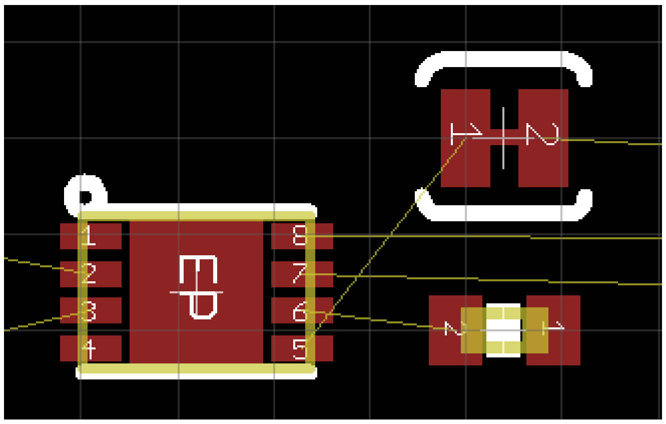


Figure 2.11

Since the Electronic Service Shop does not produce plated vias, we have to solder via wires in order to connect the two sides of the PCB, resulting in bumps that can prevent IC pins from contacting the board. Consequently no vias can be placed below ICs.

2.7.3. Trace Thickness

Traces have internal resistance, which varies inversely with their width. This means that it is always desirable to have wide traces, since it reduces power loss and waste heat generation. Furthermore, thin traces are more susceptible to burning out when large currents pass through them (detailed information on trace width and current capacity can be found on the Electronic Service Shop webpage [14]). However, wide traces are not always possible due to several factors, as exemplified by Figure 2.12.

The pins of the ICs are of fixed width and this constrains the width of the traces connecting them. For example, pin 8 is thinner than pin 27, thus the widest trace connecting them can only be as wide as pin 8. Traces also have to compete with other components such as vias (green rings), resistors (yellow rectangles with white center), and other traces for PCB real estate.

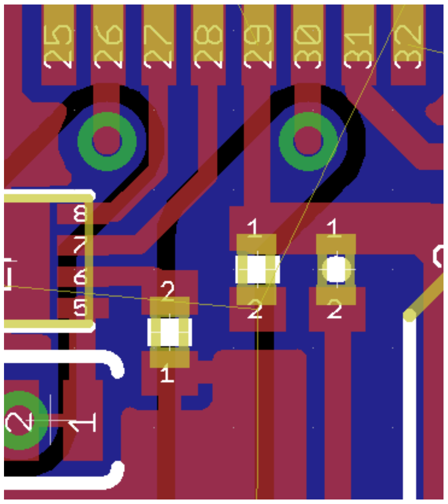


Figure 2.12

3. Design Verification

3.1. Overview

A number of the procedures were simplified and others combined so as to allow for easier testing. This often meant that the precision/accuracy of the test results are reduced. However, we will show that they are still within acceptable error margins. Other procedures were entirely revamped due to lack of proper equipment.

3.2. Braille Interface

BI.1.a - Accuracy requirement of 0.001" was changed to 5% margin due to impracticality of ensuring such a high precision. The laser beam is of finite diameter and thus would burn away more material than desired. The degree of burning varies with material and thickness and no documentation exists to give estimates on the degree of burning. Therefore the only way to ensure a precision of 0.001" is to repeatedly laser cut test samples and adjust the CAD file based on the result.

BI.1.b - We were not able to verify the height of the Braille dots due to unfeasible measurement procedure. We had intended to use a precision protractor to verify that programming the microcontroller to move the servo by 1° corresponds to 1° of actual rotation. However, we realized that the geometry of the servos makes proper measurement very difficult – the servo arm is too wide and too short to produce a reliable reading on the protractor. The precision required (5% of 0.019" = 0.0095") and the geometry of the setup means that a special rig must be built to make the measurements – a highly impractical endeavor given the technical skills we have, the amount of available resources and the amount of work involved for other modules. Ultimately we resorted to using a plastic rule and visual inspection to ensure that the heights of all the pins are uniform and are approximately 0.019".

Table A.1 shows that all measurable dimensions do indeed meet the specification to within 5%. We do not expect the reduced accuracy to adversely impact reading experience, since we are able to discern the displayed Braille characters via touch even without any Braille training.

BI.2 - Verification procedure for the refresh speed was changed to using a video camera as it is much easier to conduct, though with lower accuracy. Observe that since the Braille pattern for each alphabet is distinct, we do not have to worry about aliasing issues. Initial tests resulted in extremely high servo speeds; consequently delays were introduced in the automatic scrolling routine due to concerns with overheating servos. Results show that the device took 1.602s to display the entire alphabet when a delay of 20ms was introduced (See Figure A.3 for frames and timestamp). This corresponds to a latency of 61.6ms/character (16.2 characters/s), which is less than half the maximum allowed latency of 133ms (7.5 characters/s). Thus the refresh rate requirement is met. Although 30 fps is too slow to capture the exact moments when the pins rise and fall, this is not an issue due to the high refresh rate – even if we were to include extra frames before 'A' and after 'Z' have been displayed, the device still takes 1.635s (62.9ms/character).

3.3. Removable Memory

RM.1 - Since the functionality of the device does not require writing to the removable memory, the requirement RM.1 was modified to only require reads from memory to be errorless. The original verification required the read from the micro-SD card to be performed using a SD card reader connected to a laptop. However, now that the sense board's functionality (CO.2) depends on this test, the verification has been modified to test the reading of the microSD card via the device itself. The verification procedure in table B.1 tests that a file from the sd card can be read entirely without errors. Figure A.4 shows that all characters in the file (a, b, c, d, e, f, … x, y, z) were read correctly. The maximum latency between two reads was 9ms and the average latency between two reads was 7.88ms, thus meeting all the expected results of this test. The latency was tested to make sure it doesn't exceed the maximum refresh latency of 133ms.

RM.2 - The requirement that the memory does not become corrupted with time and multiple accesses is verified by manufacturer. A one-year warranty was received with the purchase of the SanDisk micro-SD card attests to the reliability of the removable memory.

3.4. Wireless Transceiver

WT.1 - The verification procedure involves transmitting a 160 character long SMS message from a mobile device at increments of 0.5 meters away from the Braille e-book reader (see table B.1 for details). An android phone was used to carry out this procedure. Figure A.5 shows that the message is received correctly for close proximity, 0.5m and 1m, thus passing requirement WT.1 which requires receiving from Bluetooth devices that are at least 1m away with no errors. It can be observed from the results that messages were correctly received for up to a distance of 4.5m. The length of the message tested is 160 since the size limit of an SMS is 160 characters.

3.5. Sensor

SE.1 - The verification procedure involves the tester to touch each of the capacitive pads and record the output to indicate that the touch has indeed been recognized. Figure A.6 that all the twelve bits corresponding to the twelve capacitive pads were '1' at some point of time, indicating that touch was recognized when the user had placed his finger over the corresponding pad. Thus, the requirement that each pad of the capacitive touch sensor detects user touch has been met.

3.6. Controller

CO.1 - The procedure for this requirement has been formalized to take pictures of each actuated Braille characters rather than simple observation. Figure A.7 shows that each actuated Braille characters match their corresponding Braille encoding correctly for all 26/26 alphabets of the English language. Thus, the requirement that the device can encode alphanumeric characters into control signals for servos has been met. The Braille encodings for numbers 1 to 0 match the Braille encodings for alphabets A to J and thus need not be tested separately.

CO.2 - This requirement requires the device to receive data from external sources. Since the device has the met the requirements of correctly receiving data from the micro-SD card (RM.1), wireless transceiver (WT.1) and capacitive sensor (SE.1), the device meets the CO.2 requirement.

CO.3 - Performs gesture recognition on sensor data.
The procedures for this requirement have also been formalized to take advantage of the firmware and serial monitor used for debugging. It directs the user to perform a series of gestures and the output is printed to the serial monitor. The output data as seen in Figure A.8 matches the expected results exactly and thus meets this requirement.

CO.4 - Verification test was combined with that for PS.1. The new test is better than the original one that uses a DC power supply, since replacing the bench supply with LiPo gives more realistic operating conditions. The left cell does indeed turn off when the debugging console indicates that the state of charge (SOC) has dropped below 20%.

3.7. Power Supply

The original verification procedure involves data acquisition instruments recording the voltage and current drawn for the entire duration of the test. However, all myDAQs were checked out and none of the PCI DAQ cards that are installed in the lab computers are functional. While the oscilloscope is able to record data, it can only save up to 1000 data points. Continuous data logging can be achieved via interfacing with the Agilent VEE program; however the device driver is not properly installed and the computers were unable to detect any connected oscilloscope.

PS.1 - Consequently we had to resort to using the onboard power gauging IC to log power data. The IC can measure up to 10V with an accuracy of ±30mV and a resolution of 2.5mV [10]. For SOC, it has a resolution of 1/256% [10]. Figure A.9 shows the LiPo's output voltage and it's SOC during 70min of operation. The power saving mode, which disables the left cell when SOC has dropped below 20%, is disabled in this test. Both voltage and SOC drops with time and at around 4000s (~67min), the Braille cells cease all movement. This is evident from the plot, which shows that the voltage and SOC rising and leveling off for the remainder of the test. However, note that the SOC is not 0% and thus the battery is still capable of supplying power to the other modules. This is evident from the fact that we are a still able to obtain voltage and SOC data from the debugging console even after the Braille cells have stopped moving. The servos maintained sharp movement throughout their duration of operation and thus we conclude that the device does indeed have at least one hour of battery life.

PS.2 - Figure A.9 shows that the device is able to maintain its original duration of operation after the battery has been recharged.

Note that due to technical limitations, we can only rely on visual inspection to determine performance (sharp vs. jerky motion) of servos as opposed quantitative measures such as supply voltage and current. This is because we are unable to monitor the aforementioned quantities for all 12 servos simultaneously. Monitoring each servo separately is inadequate since it is possible for the monitored servo to have consistent performance while other servos experience momentary jittering.

4. Costs

4.1. Parts

Table 1

Part	Quantity	Manufacturer	Retail Cost ($)	Bulk Purchase Cost (Quantify)($)	Actual Cost ($)
HK-282 Ultra-Micro Servo	12	HobbyKing	3.990	NA	3.990
800mAh 7.4V 15C LiPo	1	Sky Lipo	4.670	NA	4.670
Arduino Wireless Bluetooth Module	1	Gem.M. Technology	8.620	NA	8.620
2GB microSD card	1	SanDisk Corporation	2.000	NA	2.000
ATMEGA328P-AU	2	Atmel	3.050	(100) 1.70280	3.050
MAX17044G+T	1	Maxim Integrated Products	-*	(10,000) 1.30038	free
MPR121QR2	1	Freescale Semiconductor	1.200	(1,000) 0.62000	1.200
MIC5216-3.3YM5 TR		Micrel Inc	1.980	(3,000) 1.01100	1.980
MIC39101-3.3YM	2	Micrel Inc	1.630	(2,500) 0.83200	1.630
PTS525SM10SMTR LFS (Tactile Switch)	2	C&K Components	0.610	(3,000) 0.26100	0.610
101-00660-68-6 (MicroSD socket)	1	Amphenol Commercial Products	1.970	(1,000) 0.98600	1.970
APG1608CGKC/T (Green LED)	2	Kingbright Corp	0.230	(2,000) 0.07800	0.230
CSTCE8M00G55-R0 (8MHz resonator)	2	Murata Electronics North America	0.660	(3,000) 0.23400	0.660
RC0402JR-0775KL (75kΩ resistor)	1	Yageo	0.012	(10,000) 0.00130	0.012
RC0402FR-0710KL (10kΩ resistor)	4	Yageo	0.014	(10,000) 0.00149	0.014
RC0402FR-071KL (1kΩ resistor)	2	Yageo	0.014	(10,000) 0.00149	0.014
RC0402FR-07330RL (330Ω resistor)	2	Yageo	0.014	(10,000) 0.00149	0.014
478-1751-1-ND (10µF capacitor)	2	AVX Corporation	0.770	(2,000) 0.17600	0.770
F930J475KAA (4.7µF capacitor)	1	Nichicon	0.440	(2,000) 0.07600	0.440
GRM155R61A105KE15D (1µF capacitor)	4	Murata Electronics North America	0.095	(10,000) 0.01323	0.095
C1005X5R0J104K (0.1µF capacitor)	6	TDK Corporation	0.026	(10,000) 0.00462	0.026
C1005X7R1E103J (10F capacitor)	1	TDK Corporation	0.026	(10,000) 0.00462	0.026
PCB	1	-	10.000	NA	free
Acrylic casing	1	-	3.000	NA	3.000
Total			**94.366**	**83.81846**	**84.366**

*Unable to find supplier that does not sell in bulk. Obtained free sample from manufacturer instead.

4.2. Labor

Table 2

Member	$/hour	# of weeks	Hours/week	Total # of hours	Subtotal	Multiplier(x2.5)
Mingjie Wang	$20	12	20	240	$4,800	$12,000
Rajarshi Roy	$20	12	20	240	$4,800	$12,000
					Total:	$24,000

5. Conclusion

5.1. Accomplishments

We created a 2.6" × 2.6" × 2.6" device that reads eBooks stored in microSD cards, read SMSs from mobile phones over Bluetooth, displays them as Braille, and has 1 hour of battery life. We also created a working capacitive touch surface from scratch.

5.2. Uncertainties

The dimensional accuracy of the Braille cells is not as high as originally intended. However, due to the small scale involved, it is questionable that any improvements would result in improvement in user experience.

Without working DAQ devices, we were only able to partially characterize the electrical performance of the device and its various modules. While the device does indeed have a battery life of 1 hour, a more comprehensive characterization will may expose any undesirable performance characteristics. The additional information will subsequently allow us to fine-tune the design and produce better performance. For example, by logging voltage and current drawn at higher time resolutions for every servo, we may find that there is a steep drop in output voltage for the servos' voltage regulator when more than 4 servos are moving simultaneously. We can then introduce delays in between commands so as to stagger the servos' movement to resolve the problem.

5.3. Ethical considerations

Table 3

IEEE Code of Ethics	Considerations
1. To accept responsibility in making decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.	Explosions can be a problem for battery-powered mobile devices (as evidenced by exploding cell phones). We will design and test the device to ensure that the LiPo battery maintains stable performance.
2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist.	We want to score well in this course and to create an awesome device; but we do have other commitments that compete for our time, attention and resources. We will try our best to balance and strive for these goals in a scrupulous manner.
3. To be honest and realistic in stating claims or estimates based on available data.	We will advertise accurate battery life and reliability of performance.
4. To reject bribery in all its forms.	Not applicable.
5. To improve the understanding of technology, its appropriate application, and potential consequences.	Not applicable.
6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.	We admit that we are not experienced and skilled in creating such devices. But we are willing to learn, and we need it to graduate.
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.	We will do that throughout the course.

8. To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin.	Not applicable.
9. To avoid injuring others, their property, reputation, or employment by false or malicious action.	Not applicable.
10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.	We will do that throughout the course.

5.4. Future work

The current device can only be powered by a 7.4V LiPo. This means that to produce the output voltage of 3.3V, the voltage regulators have to down-convert by as much as 4.1V, which translates to high power loss at the regulators themselves. Thus we can design the device to take a 3.7V LiPo instead and see if it results in a longer battery life.

To charge the current device, the user would have to plug it into a standalone LiPo charger. To reduce total cost of ownership and improve ease of use, we can incorporate a charging IC into the device so that users can connect the device directly to the power adapter for charging.

Since there is still an appreciable amount of unused space in the current device, we can design a new chassis with smaller form factor to improved portability.

Acknowledgements

We would like to thank Mark, Scott, Dan and Wally of the Electronics Service Shop for providing us with advice and assistance throughout the semester. We want to especially thank Mark for milling all our tiny PCBs, one of which was produced on very short notice. We are also grateful to the people at Champaign-Urbana Community Fab-Lab for letting us use the laser cutter to create our acrylic chassis. We also want to thank our TA, Jane, for her patience and guidance throughout the semester. Lastly, we would like to thank our fellow course mates fostering a vibrant and cooperative environment in 246EL.

References

[1] National Library Service for the Blind and Physically Handicapped (NLS) of the Library of Congress, (NLS Reference Circulars) Braille Literacy: Resources for Instruction, Writing Equipment, and Supplies. [Online]. Available: http://www.loc.gov/nls/reference/circulars/brailleliteracy.html

[2] Handy Tech North America. Handy Tech Braille Note Takers and Displays. [Online]. Available: http://www.handytech.us/ht.html

[3] *Braille Books and Pamphlets*, NLS Specification # 800, 2005

[4] Legge, Gordon E., Cindee M. Madison, and Stephen J. Mansfield. "Measuring Braille Reading Speed With The MNREAD Test." *Visual Impairment Research* 1.3 (1999): 131-145. *Academic Search Premier*. Web. 20 Feb. 2012

[5] *Secure Digital Input/Output (SDIO) Card Specification*, SD Association, 2001

[6] Arduino. Reference: SD Library. [Online]. Available: http://arduino.cc/en/Reference/SD. Accessed March 2012

[7] Ebay. Arduino Bluetooth Module Wireless Transceiver PCB RS232. [Online]. Available: http://www.ebay.com/itm/Arduino-Bluetooth-Module-Wireless-Transceiver-PCB-RS232-/180712962730. Accessed April 2012

[8] Arduino. Reference: Serial. [Online]. Available: http://arduino.cc/en/Reference/Serial. Accessed March 2012

[9] *Proximity Capacitive Touch Sensor Controller*, Freescale Semiconductor, Tempe AR, 2011

[10] Arduino. Reference: Libraries. [Online]. Available: http://arduino.cc/en/Reference/Libraries. Accessed March 2012

[11] Arduino. Reference: SoftwareSerial Library. [Online]. Available: http://arduino.cc/en/Reference/SoftwareSerial. Accessed March 2012

[12] *RN-41 – Class 1 Bluetooth® Module*, Roving Networks, Los Gatos, 2011

[13] Russell McMahon. Reply to Question on StackExchange - Electrical Engineering: Combining capacitors for multiple chips. [Online]. Available: http://electronics.stackexchange.com/questions/27902/combining-capacitors-for-multiple-chips

[14] Mark Smart. Design Requirements For Our Milled PCBs. ECE Electronics Service Shop. [Online]. Available: http://www.ece.illinois.edu/eshop/pcbdesign/DesignReq.htm. Accessed April 2012

[15] *MIC39100/39101/39102: 1A Low-Voltage Low-Dropout Regulator*, Micrel Inc., San Jose, CA, 2005

[16] *MIC5216: 500mA-Peak Output LDO Regulator*, Micrel Inc., San Jose, CA, 2007

[17] SparkFun Electronics. LiPower Shield. [Online]. Available: http://www.sparkfun.com/products/10711. Accessed March 2012

[18] *MAX17043/MAX17044: Compact, Low-Cost 1S/2S Fuel Gauges with Low-Battery Alert*, Maxim Integrated Products, Sunnyvale, CA, 2010

[19] *IEEE Code of Ethics*, IEEE Policies, Section 7 - Professional Activities (Part A - IEEE Policies), 2006

Appendix A – Miscellaneous Tables and Figures

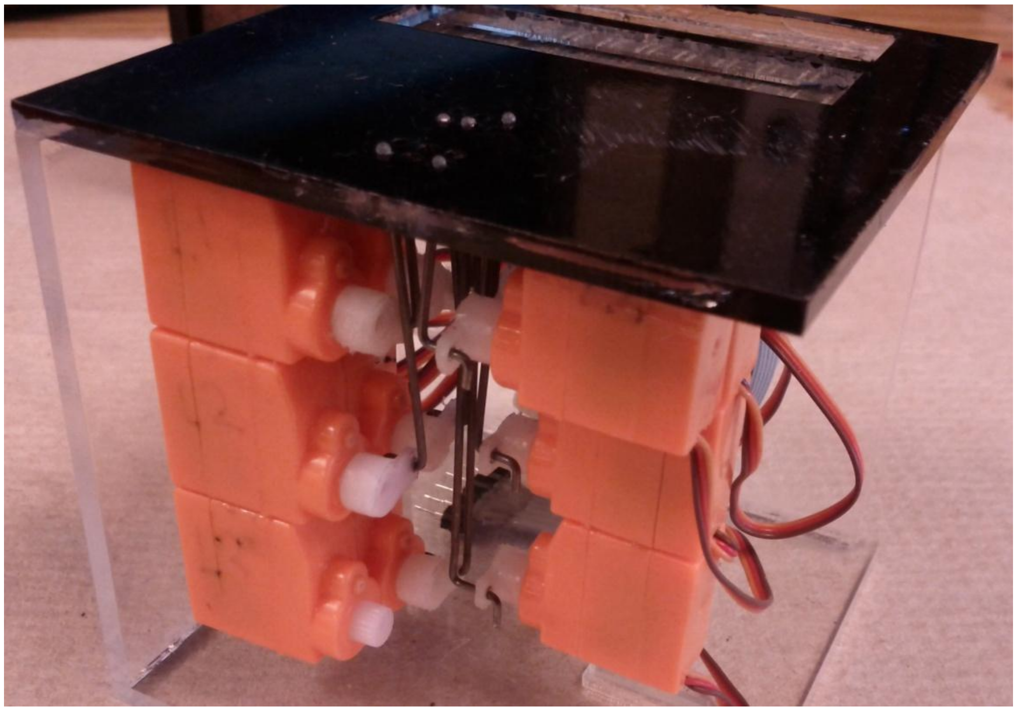


Figure A.1
Cut-away view of servo stacks, Braille pins and top plate

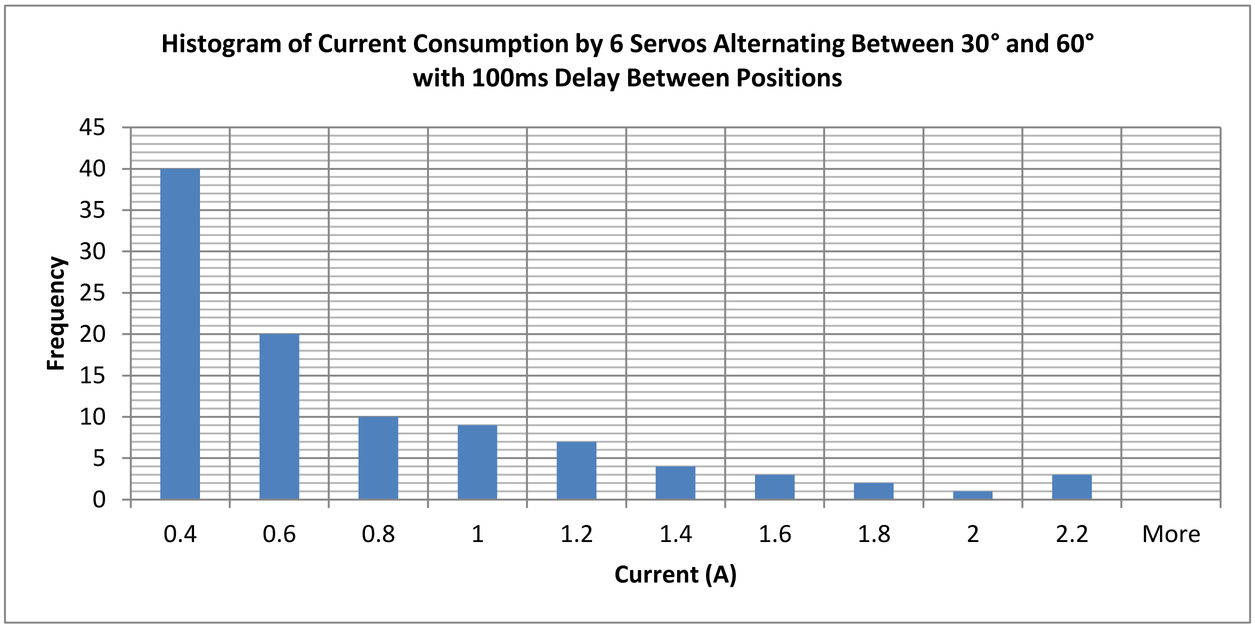


Figure A.2
100ms delay is introduced in the microcontroller program to slow down rate of servo movement. This simulates actual usage.

Table A.1
Measurements of laser-cut Braille cell

	Dot base diameter	Intracell dot spacing	Intercell dot spacing
Desired (inches)	0.057	0.092	0.245
Measured (inches)	0.059	0.090	0.244
Deviation (%)	+3.51	-2.17	-0.41%

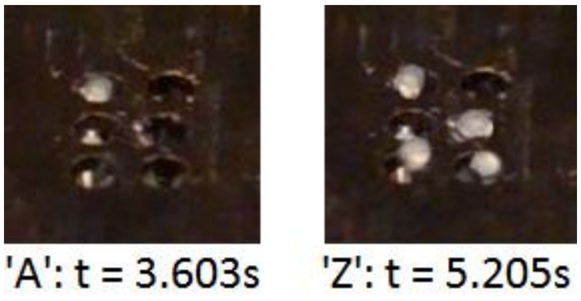
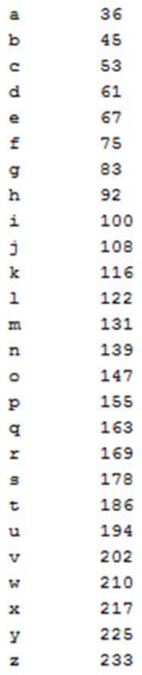
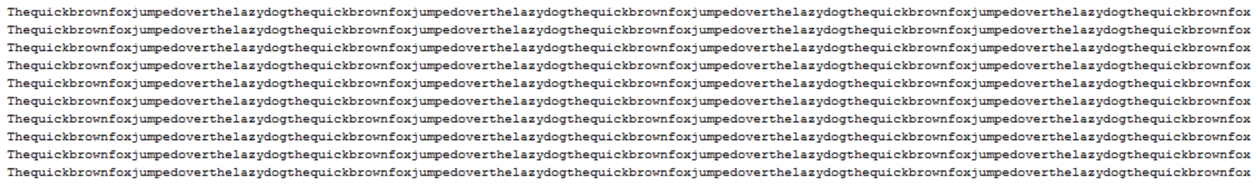


Figure A.3
Frames captures of video and corresponding timestamps



```
a        36
b        45
c        53
d        61
e        67
f        75
g        83
h        92
i        100
j        108
k        116
l        122
m        131
n        139
o        147
p        155
q        163
r        169
s        178
t        186
u        194
v        202
w        210
x        217
y        225
z        233
```

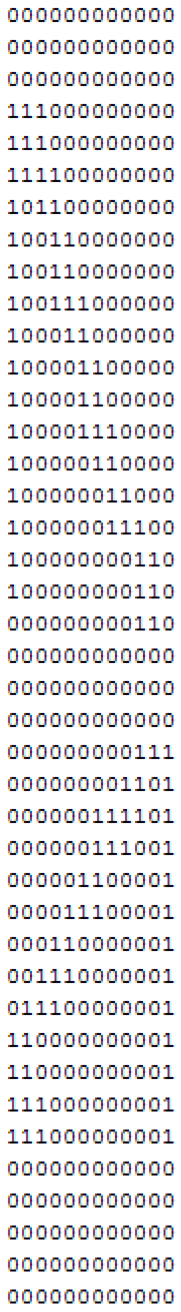
Figure A.4
Serial monitor output showing data from SD card file (left) and time-stamp of retrieval (right)



```
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
Thequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfoxjumpedoverthelazydogthequickbrownfox
```

Figure A.5

Serial monitor output showing data received from the wireless transceiver. Note that data were received 10 times: (close proximity, 0.5m, 1m, 1.5m, 2m, 2.5, 3m, 3.5m, 4m, 4.5m)



```
000000000000
000000000000
000000000000
111000000000
111000000000
111100000000
101100000000
100110000000
100110000000
100111000000
100011000000
100001100000
100001100000
100001110000
100000110000
100000011000
100000011100
100000000110
100000000110
000000000110
000000000000
000000000000
000000000000
000000000111
000000001101
000000111101
000000111001
000001100001
000011100001
000110000001
001110000001
011100000001
110000000001
110000000001
111000000001
111000000001
000000000000
000000000000
000000000000
000000000000
000000000000
```

Figure A.6

Serial monitor output showing data received from the capacitive sense board. A ‘1’ indicates a touched pad.

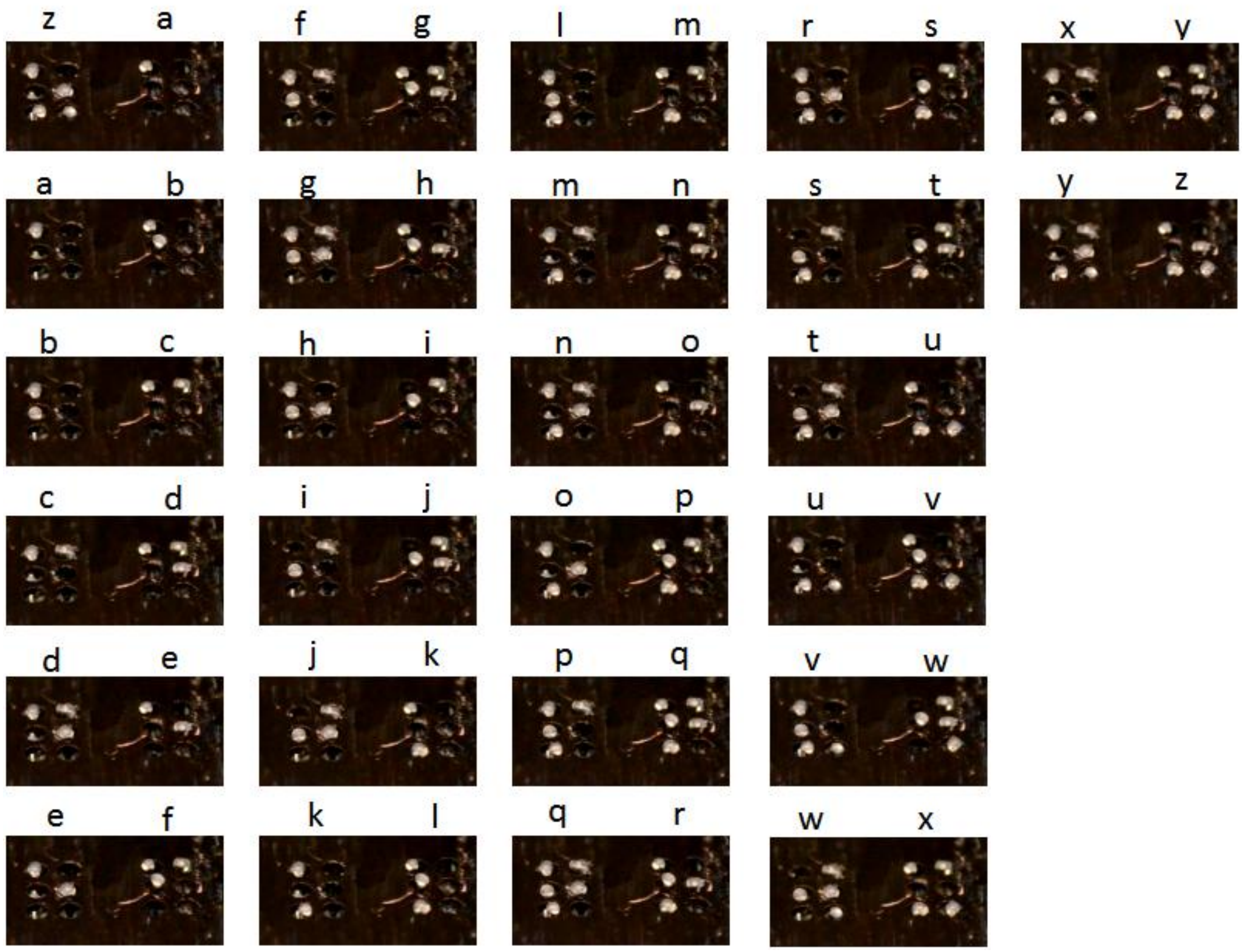
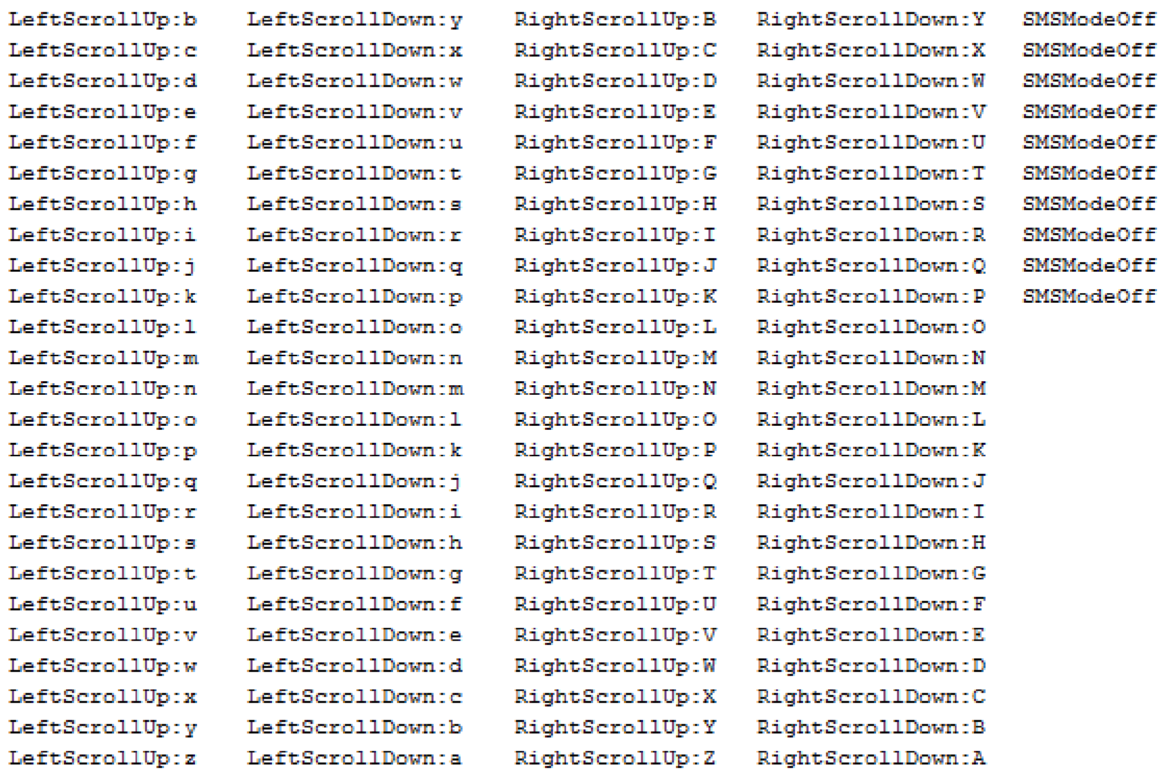


Figure A.7
Both Braille cells can correctly display the English alphabets from ‘A’ to ‘Z’



```
LeftScrollUp:b   LeftScrollDown:y   RightScrollUp:B   RightScrollDown:Y   SMSModeOff
LeftScrollUp:c   LeftScrollDown:x   RightScrollUp:C   RightScrollDown:X   SMSModeOff
LeftScrollUp:d   LeftScrollDown:w   RightScrollUp:D   RightScrollDown:W   SMSModeOff
LeftScrollUp:e   LeftScrollDown:v   RightScrollUp:E   RightScrollDown:V   SMSModeOff
LeftScrollUp:f   LeftScrollDown:u   RightScrollUp:F   RightScrollDown:U   SMSModeOff
LeftScrollUp:g   LeftScrollDown:t   RightScrollUp:G   RightScrollDown:T   SMSModeOff
LeftScrollUp:h   LeftScrollDown:s   RightScrollUp:H   RightScrollDown:S   SMSModeOff
LeftScrollUp:i   LeftScrollDown:r   RightScrollUp:I   RightScrollDown:R   SMSModeOff
LeftScrollUp:j   LeftScrollDown:q   RightScrollUp:J   RightScrollDown:Q   SMSModeOff
LeftScrollUp:k   LeftScrollDown:p   RightScrollUp:K   RightScrollDown:P   SMSModeOff
LeftScrollUp:l   LeftScrollDown:o   RightScrollUp:L   RightScrollDown:O
LeftScrollUp:m   LeftScrollDown:n   RightScrollUp:M   RightScrollDown:N
LeftScrollUp:n   LeftScrollDown:m   RightScrollUp:N   RightScrollDown:M
LeftScrollUp:o   LeftScrollDown:l   RightScrollUp:O   RightScrollDown:L
LeftScrollUp:p   LeftScrollDown:k   RightScrollUp:P   RightScrollDown:K
LeftScrollUp:q   LeftScrollDown:j   RightScrollUp:Q   RightScrollDown:J
LeftScrollUp:r   LeftScrollDown:i   RightScrollUp:R   RightScrollDown:I
LeftScrollUp:s   LeftScrollDown:h   RightScrollUp:S   RightScrollDown:H
LeftScrollUp:t   LeftScrollDown:g   RightScrollUp:T   RightScrollDown:G
LeftScrollUp:u   LeftScrollDown:f   RightScrollUp:U   RightScrollDown:F
LeftScrollUp:v   LeftScrollDown:e   RightScrollUp:V   RightScrollDown:E
LeftScrollUp:w   LeftScrollDown:d   RightScrollUp:W   RightScrollDown:D
LeftScrollUp:x   LeftScrollDown:c   RightScrollUp:X   RightScrollDown:C
LeftScrollUp:y   LeftScrollDown:b   RightScrollUp:Y   RightScrollDown:B
LeftScrollUp:z   LeftScrollDown:a   RightScrollUp:Z   RightScrollDown:A
```

Figure A.8

Serial monitor output of gesture recognition firmware. All gestures have been successfully recognized. Note that the original screen-capture which was a single column of data has been divided into columns ease of viewing

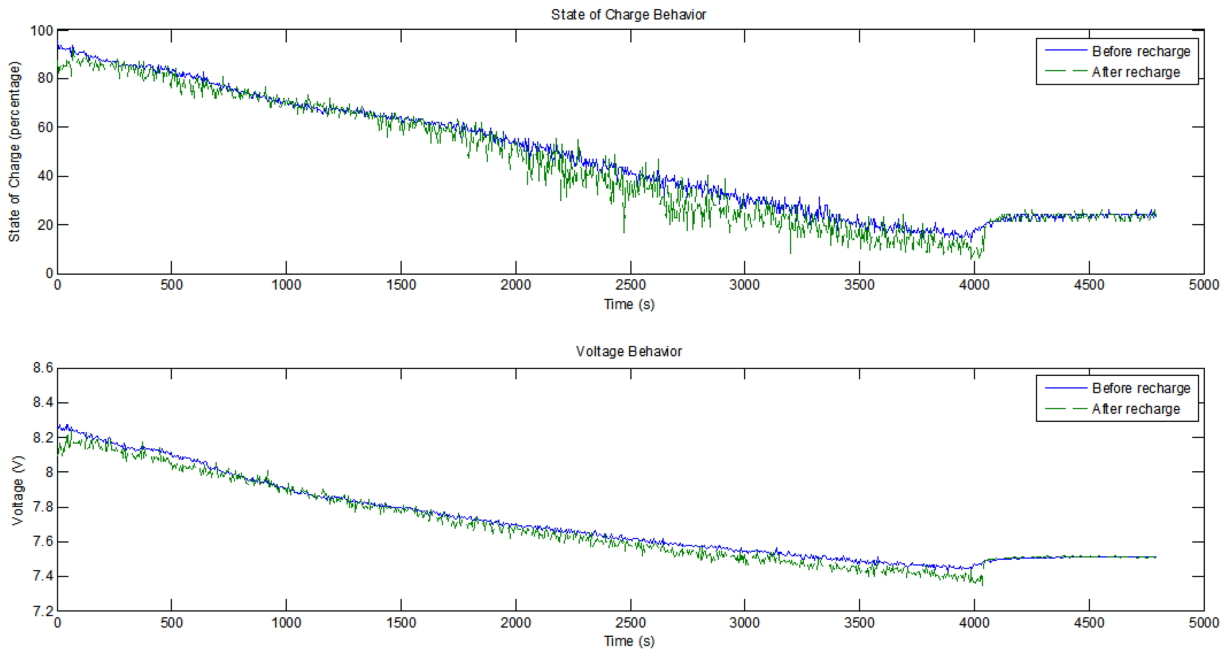


Figure A.4

Plots of battery's output voltage and SOC during simulated regular operating conditions.

Appendix B – Verifications Table

Table B.1

Req.	Verification	Expected Results	Pass
BI.1	a. Use digital calipers to measure dimensions. i) Select objects in the CAD program (Inkscape) and check the dimensions palette. Use built-in rules to measure separation of elements. b. Use control board to command the servo. Measure length of servo arm (via digital calipers) and angle of arm rotation (via precision protractor).	a. Dimensions should match specifications to within 5%. i) Values math the required dimensions exactly. b. Vertical displacement should be calculated via measurements to be 0.019”±5%.	a. Yes b. No
BI.2	Program microcontroller to automatically scroll through alphabets A to Z continuously and record action of Braille cells via video camera at 30 fps. Use video editing software to identify the frames and corresponding timestamps at which A and Z are displayed. Divide the time difference by 26 to find the refresh rate.	Period must be at most 0.133s. a. Longest time taken should be <1000 us. b. Sum of servo latency and microprocessor latency should be at most 133ms.	Yes
CO.1	1) Load firmware to sense board automatically scroll through English alphabets a to z and send them for display to control board every four seconds 2) Capture an image of every new character displayed on the Braille interface	26/26 alphabets displayed match their corresponding Braille pattern	Yes
CO.2	Perform tests for SE.1, RM.1, and WT.1	SE.1, RM.1. and WT.1 must all pass for this test to pass	Yes
CO.3	1) Load firmware to output to Serial monitor the following data: -If left slider scroll is detected, output message “LeftScrollUp:X” or “LeftScrollDown:X” depending on scroll up or scroll down. X will be replaced by <u>lowercase</u> English alphabets which will increment for every scrollup and decrement for every scrolldown. -If right slider scroll is detected, output message “RightScrollUp:X” or “RightScrollDown:X” depending on scroll up or scroll down. X will be replaced by <u>uppercase</u> English alphabets which will increment for every scrollup and decrement for every scrolldown. -If both sliders are pressed for greater than 2 seconds, output message “SMSModeOff: 2) Connect device to Serial Monitor and Reset Device 3) Scroll up from ‘a’ to ‘z’ using left slider 4) Scroll down from ‘z’ to ‘a’ using left slider 5) Scroll up from ‘A’ to ‘Z’ using right slider 6) Scroll down from ‘Z’ to ‘A’ using right slider 7) Press on both sliders for greater than 2 seconds. 8) Repeat step 7 for nine times	Serial Monitor output indicates that: -LeftScrollUp gesture detected 25/25 times -LeftScrollDown gesture detected 25/25 times -RightScrollUp gesture detected 25/25 times -RightScrollDown gesture detected 25/25 times -SMSModeOff gesture detected 10/10 times	Yes

CO.4	Program control board to query battery's state of charge every 5 seconds and set it to turn off one cell when state of charge has fallen below 20%. The microcontroller will output the state of charge to debugging console to allow for visual confirmation.	One cell will stop moving when the debugging shows that the state of charge has fallen below 20%.	Yes
SE.1	1) Load firmware to output to Serial monitor the following data: Continuous print out of the following string in a new line: "xxxxxxxxxxxx" where each x is replaced by the state of a capacitive pad (1 =touched and 0= not touched) 2) Connect device to Serial Monitor and Reset Device 3) Touch all the capacitive pads with uncovered finger (no gloves) either sequentially or at the same time	Serial Monitor output indicates that: -Touch detected for 12/12 capacitive pads	Yes
RM. 1	1) Load firmware to retrieve data one character at a time into an internal array and also record the time-stamp in milliseconds since the start of the program till the point the character was retrieved. After all data has been retrieved, output each character and associated time-stamp to Serial Monitor. 2) Insert micro-SD card with a .txt file containing the data consisting of the 26 characters: abcdefghijklmnopqrstuvwxyz 3) Connect device to Serial Monitor and Reset Device	Serial Monitor output indicates that: - all data in the file are read properly - latency between two SD card reads must not exceed 133ms	Yes
RM. 2	Memory integrity is guaranteed by manufacturer. Not practical to test ourselves.	Not applicable.	
WT. 1	1) Load firmware to output to Serial monitor the following data: Any data that is received over the Bluetooth serial port 2) Connect device to Serial Monitor and Reset Device 3) Pair the Bluetooth transceiver with a mobile device 4) Send the following 160 character long message: "Thequickbrownfoxjumpedoverthelazydogthequickbrownfox jumpedoverthelazydogthequickbrownfoxjumpedoverthelazyd ogthequickbrownfoxjumpedoverthelazydogthequickbrownfox " When the mobile device is placed in close proximity of the device 5) Repeat step 4 for greater distances with increments of 0.5 meters	Serial Monitor output indicates that: - the Bluetooth data can be received for distances up to 1 meters (three lines of the broadcasted text should be present in the Serial monitor corresponding to close proximity, 0.5 meters and 1 meters)	Yes
PS.1	Begin with a fully charged battery. Program control board to query battery's voltage and state of charge every 5 seconds and print values to the debugging console. Program sense board to continuously cycle through *Pride and Prejudice* which is stored on the microSD card. Plot results in Matlab.	Voltage never drops below 5V and the state of charge never goes to 0% for an entire hour. Motion of servos is sharp and never jerky.	Yes

PS.2	The battery would have been depleted by the end of the test for PS.1 Otherwise continue test until servos have stopped moving. Proceed to charge battery and perform test for PS.1. a. Measure voltage across battery via an oscilloscope throughout the entire charging period. Plot the voltage over time in Matlab to check for inability to be recharged back to 7.4V.	Test for PS.1 still pass. a. No voltage spikes during charging and the battery voltage is ~8V when fully charged.	Yes

Appendix C – Firmware

B.1. Sense Board

```
//CAPACITIVE
#include "mpr121.h"  //for capacitive IC
#include <Wire.h>    //I2C communication with capacitive IC
int irqpin = A3;  // Digital 2
int touchStates[12] = {0,0,0,0,0,0,0,0,0,0,0,0}; //to keep track of the previous touch states

//SD CARD
#include <SD.h>
File file;
char* name = "BK1.TXT"; // Create an array that contains the name of our file.
char* bkName[10] = {"BK1.TXT", "BK2.TXT", "BK3.TXT", "BK4.TXT", "BK5.TXT", "BK6.TXT", "BK7.TXT",
"BK8.TXT", "BK9.TXT", "BK10.TXT"};

//COMMUNICATION WITH CONTROL
#include <SoftwareSerial.h>
SoftwareSerial mySerial(A0, A1);  //rx:A0 tx:A1
char flag=0xff;

//SMS
char SMS[160];         //160 is the size limit to any SMS
int length = 0;        //length of SMS

int mode; //STATE: 0 for text, 1 for sms
int touched = 0; //this state is entered when it is touched and left when touch is removed
int origin; //this is the original point where it was touched, refreshed only when touch is removed
int currTextPos, currBookPos, currSmsPos, prevTextPos, prevBookPos, prevSmsPos;
int prevCenter, currCenter, prevprevTextPressed, prevTextPressed, currTextPressed, prevBookPressed, currBookPressed;
boolean exitSmsState; int exitSmsCenter; unsigned long exitSmsMillis;

void setup(){
  pinMode(irqpin, INPUT);
  digitalWrite(irqpin, HIGH); //enable pullup resistor

  Serial.begin(9600);
  mySerial.begin(9600);

  //Capacitive
  Wire.begin();
  mpr121_setup();
  delay(100);
  Serial.println("Ready...");

  //SD
  pinMode(10, OUTPUT);
  if (!SD.begin(10)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  currTextPos = 0;  currBookPos = 0;
  prevTextPressed = 0;  prevBookPressed = 0;
  currTextPressed = 0;  currBookPressed = 0;
  exitSmsState = false;
  mode =0;
}

void loop(){
```

```
getNumber();
int numPads =0;
int whichPads =0;
float center;
for (int i=0; i < 12; i++){
  //Serial.print(touchStates[i]);
  if((touchStates[i]==1)&&(i>0)&&(i<11)){
    numPads++;
    whichPads+=i;
  }
}
currTextPressed = touchStates[0]; currBookPressed = touchStates[11];
currCenter = whichPads/numPads;

if(touched ==0){
  if((currTextPressed==1)&&(prevTextPressed==0)) {touched =1; origin =currCenter;}
}
else{
 if(currTextPressed==0) touched =0;
}


if(touched){
  if((currCenter-origin)>1){
    origin = currCenter;
    if(mode==0){
      char temp = getSD(currTextPos);
      if(temp!=-1){
        mySerial.write(0xff);
        mySerial.write(temp);
        currTextPos++;
      }
    }
    if(mode==1){
      if(currSmsPos<length){
        mySerial.write(0xff);
        mySerial.write(SMS[currSmsPos]);
        currSmsPos++;
        //mySerial.write(0xff);
        //mySerial.write('u');
        //delay(10);
      }
    }
  }
  else if((currCenter-origin)<-1){
    origin = currCenter;
    if(mode==0){
      if(currTextPos>=0){
        mySerial.write(0xff);
        mySerial.write(getSD(currTextPos));
        currTextPos--;
      }
    }
    if(mode==1){
      if(currSmsPos>=0){
        mySerial.write(0xff);
        mySerial.write(SMS[currSmsPos]);
        currSmsPos--;
        //mySerial.write(0xff);
        //mySerial.write('d');
        //delay(10);
      }
      }
    }
  }
  if(currBookPressed){
    if((currCenter>prevCenter)) {currBookPos++; name = bkName[(currBookPos/4)%10]; currTextPos = 0;}
    if((currCenter<prevCenter)&&(currBookPos>=0)) {currBookPos--; name = bkName[(currBookPos/4)%10]; currTextPos = 0;}
  }
  prevCenter = currCenter;
  prevTextPressed = currTextPressed;
  prevprevTextPressed = prevTextPressed;
  prevBookPressed = currBookPressed;

  if(exitSmsState){
    if((millis()-exitSmsMillis)>3000) {
      Serial.println("exit SMS mode");
      mode = 0;
      exitSmsMillis = millis();
    }
    if(((currCenter-exitSmsCenter)>2)||((currCenter-exitSmsCenter)<-2)) exitSmsState = false;

  }
  else{
    if((touchStates[0]==1)&&(touchStates[11]==1)){
      exitSmsMillis=millis();
      exitSmsCenter=currCenter;
      exitSmsState=true;
    }
  }

  if (Serial.available()>=2){
    if(Serial.read() == 0xff){
      length = Serial.read();
      for(int i=0; i<length; i++){
        while(!Serial.available());
        SMS[i]=Serial.read();
        mySerial.write(0xff);
        mySerial.println(SMS[i]);
      }
      mode =1;  //change to sms mode
      currSmsPos = 0;

    }
  }

  //Serial.print('\t');
  //Serial.print(currTextPos);
  //Serial.print('\t');
  //Serial.print(currBookPos);
  //Serial.println();
  delay(10);
}


//reads touch inputs and updates states
void getNumber()
{
  if(!digitalRead(irqpin)){
      //read the touch state from the MPR121
      Wire.requestFrom(0x5A,2);
      byte LSB = Wire.read();
      byte MSB = Wire.read();
      uint16_t touched = ((MSB << 8) | LSB); //16bits that make up the touch states

      for (int i=0; i < 12; i++){  // Check what electrodes were pressed
        if(touched & (1<<i)){
          touchStates[i] = 1;
        }else{
          touchStates[i] = 0;
        }
      }
  }
 // else{Serial.print("same");  }
 delay(5);
}

char getSD(int pos)
{
  char alphabet;                    // To store the current character
  file = SD.open(name, FILE_READ);  // Open the file
  file.seek(pos);                   // Position seek in file.
  alphabet=file.read(); // Get the byte in the file at seek location.
  file.close();                     // Close the file
  return alphabet;
}
```

B.2. Control Board

```
//SOFTWARE SERIAL

#include <SoftwareSerial.h>

SoftwareSerial mySerial(A0, A1);  //rx:A0 tx:A1



// LIPO POWER GAUGING

#include <Wire.h>

#define MAX17043_ADDRESS 0x36  // R/W =~ 0x6D/0x6C

#define V_RES 1/400 // 1/800 for 17043; 1/400 for 17044

float batVoltage;

float batPercentage;

unsigned long curr_time, prev_time;

int low_count = 0;



//SERVO
#include <Servo.h>
// Create servo object to control a servo.
Servo left1;  //1:topleft
Servo left2;  //2:topright
Servo left3;  //3:midleft
Servo left4;  //4:midright
Servo left5;  //5:botleft
Servo left6;  //6:botright
Servo right1; //1:topleft
Servo right2; //2:topright
Servo right3; //3:midleft
Servo right4; //4:midright
Servo right5; //5:botleft
Servo right6; //6:botright

// Calibrated up positions.
#define left1up 63  //1:topleft
#define left2up 66  //2:topright
#define left3up 57  //3:midleft
#define left4up 52  //4:midright
#define left5up 60  //5:botleft
#define left6up 57  //6:botright
#define right1up 61 //1:topleft
#define right2up 58 //2:topright
#define right3up 64 //3:midleft
#define right4up 54 //4:midright
#define right5up 57 //5:botleft
#define right6up 50 //6:botright

// Calibrated down positions.
#define left1down 37  //1:topleft
#define left2down 44  //2:topright
#define left3down 36  //3:midleft
#define left4down 36  //4:midright
#define left5down 39  //5:botleft
#define left6down 37  //6:botright
#define right1down 34 //1:topleft
#define right2down 34 //2:topright
#define right3down 37 //3:midleft
#define right4down 30 //4:midright
#define right5down 39 //5:botleft
#define right6down 30 //6:botright

//1 byte chars that store the
char currLeft, currRight, prevLeft, prevRight, changeLeft, changeRight;
char brailleMap[26] = {
	0x01, 0x05, 0x03, 0x0B, 0x09, 0x07, 0x0F, // 'a' to 'g'
	0x0D, 0x06, 0x0E, 0x11, 0x15, 0x13, 0x1B, // 'h' to 'n'
	0x19, 0x17, 0x1F, 0x1D, 0x16, 0x1E, 0x31, // 'o' to 'u'
	0x35, 0x2E, 0x33, 0x3B, 0x39 // 'v' to 'z'
};

void setup(){
	Serial.begin(9600);
	//Serial.println("Hi I am ATmega!");

	// set the data rate for the SoftwareSerial port
	mySerial.begin(9600);
	//mySerial.println("The other ATmega says Hi!");

	Wire.begin();  // Start I2C
	delay(100);
	configMAX17043(32);  // Configure the MAX17043's alert percentage
	qsMAX17043();  // Restart fuel-gauge calculations

	// Attaches servo objects to pins.
	left1.attach(5); left2.attach(6); left3.attach(7);
	left4.attach(4); left5.attach(2); left6.attach(3);
	right1.attach(9); right2.attach(11); right3.attach(13);
	right4.attach(12); right5.attach(8); right6.attach(10);
	pinMode(A2, OUTPUT);  //right cell enable
	pinMode(A3, OUTPUT);  //left cell enable
	digitalWrite(A2, HIGH);
	digitalWrite(A3, HIGH);

	delay(500);
	left1.write(left1up); left2.write(left2up); left3.write(left3up);
	left4.write(left4up); left5.write(left5up); left6.write(left6up);
	right1.write(right1up); right2.write(right2up); right3.write(right3up);
	right4.write(right4up); right5.write(right5up); right6.write(right6up);

	delay(500);
	left1.write(left1down); left2.write(left2down); left3.write(left3down);
	left4.write(left4down); left5.write(left5down); left6.write(left6down);
	right1.write(right1down); right2.write(right2down); right3.write(right3down);
	right4.write(right4down); right5.write(right5down); right6.write(right6down);

	prevLeft = 0x00;
	prevRight = 0x00;
	currLeft = 0x00;
	currRight = 0x00;
}

void loop(){
	curr_time = millis();
	if(curr_time - prev_time >= 5000){
		prev_time = curr_time;
		batPercentage = percentMAX17043();
		// Multiply by V_RES since voltage data is quantized.
		batVoltage = (float) vcellMAX17043() * V_RES;

		Serial.print(curr_time);
		Serial.print("\t");
		Serial.print(batPercentage, 3);  // Print the battery percentage
		Serial.print("\t");
		Serial.println(batVoltage, 3);  // print battery voltage

		if(low_count < 10){
			if(batPercentage <= 15)
				low_count++;
		}else digitalWrite(A3, LOW); // Disable left servo to conserve power.

		qsMAX17043();  // Restart fuel-gauge calculations
	}

	if (mySerial.available() >= 2){
		if(mySerial.read() == 0xff){
			prevLeft = currLeft;
			currLeft = currRight;
			prevRight= currRight;
			int input = mySerial.read();
			if((input>=65)&&(input<=90)) currRight = brailleMap[input-65];
			else if((input>=97)&&(input<=122)) currRight = brailleMap[input-97];
			else currRight = 0x00;

			//debugging stuff
			/*Serial.write(input); Serial.write('\t');
			Serial.print(input); Serial.write('\t');
			Serial.print("prevLeft:");
			for(int i=0; i<6; i++){Serial.print((prevLeft>>i)&1);}
			Serial.print("    currLeft:");
			for(int i=0; i<6; i++){Serial.print((currLeft>>i)&1);}
			Serial.print("    prevRight:");
			for(int i=0; i<6; i++){Serial.print((prevRight>>i)&1);}
			Serial.print("    currRight:");
			for(int i=0; i<6; i++){Serial.print((currRight>>i)&1);}
			Serial.write('\n');
			*/
			//debugging stuff
		}
	}

	changeLeft = currLeft^prevLeft;
	changeRight = currRight^prevRight;

	// tell servo to go to position in variable 'pos'
		if((changeLeft>>1)&1) {if((currLeft>>1)&1) {left1.write(left1up); /*Serial.println("left1 up");*/}  else
{left1.write(left1down); /*Serial.println("left1 down");*/}}
		if((changeLeft>>0)&1) {if((currLeft>>0)&1) {left2.write(left2up); /*Serial.println("left2 up");*/}  else
{left2.write(left2down); /*Serial.println("left2 down");*/}}
		if((changeLeft>>3)&1) {if((currLeft>>3)&1) {left3.write(left3up); /*Serial.println("left3 up");*/}  else
{left3.write(left3down); /*Serial.println("left3 down");*/}}
		if((changeLeft>>2)&1) {if((currLeft>>2)&1) {left4.write(left4up); /*Serial.println("left4 up");*/}  else
{left4.write(left4down); /*Serial.println("left4 down");*/}}
		if((changeLeft>>4)&1) {if((currLeft>>4)&1) {left5.write(left5up); /*Serial.println("left5 up");*/}  else
{left5.write(left5down); /*Serial.println("left5 down");*/}}
		if((changeLeft>>5)&1) {if((currLeft>>5)&1) {left6.write(left6up); /*Serial.println("left6 up");*/}  else
{left6.write(left6down); /*Serial.println("left6 down");*/}}
		delay(30);
		if((changeRight>>1)&1) {if((currRight>>1)&1) {right1.write(right1up); /*Serial.println("right1 up");*/}  else
{right1.write(right1down); /*Serial.println("right1 down");*/}}
		if((changeRight>>0)&1) {if((currRight>>0)&1) {right2.write(right2up); /*Serial.println("right2 up");*/}  else
{right2.write(right2down); /*Serial.println("right2 down");*/}}
		if((changeRight>>3)&1) {if((currRight>>3)&1) {right3.write(right3up); /*Serial.println("right3 up");*/}  else
{right3.write(right3down); /*Serial.println("right3 down");*/}}
		if((changeRight>>2)&1) {if((currRight>>2)&1) {right4.write(right4up); /*Serial.println("right4 up");*/}  else
{right4.write(right4down); /*Serial.println("right4 down");*/}}
		if((changeRight>>4)&1) {if((currRight>>4)&1) {right5.write(right5up); /*Serial.println("right5 up");*/}  else
{right5.write(right5down); /*Serial.println("right5 down");*/}}
		if((changeRight>>5)&1) {if((currRight>>5)&1) {right6.write(right6up); /*Serial.println("right6 up");*/}  else
{right6.write(right6down); /*Serial.println("right6 down");*/}}

	/*
	///////////////////////////////////////////////////////////////
		if (Serial.available() > 0) {
				// read the incoming byte:
			int blah = Serial.read();
			if(blah ==97) {left6up--; Serial.print(left6up);}
			if(blah ==98) {left6up++; Serial.print(left6up);}
	}


		left1.write(left1up);          // tell servo to go to position in variable 'pos'
		left2.write(left2up);
		left3.write(left3up);
		left4.write(left4up);
		left5.write(left5up);
		left6.write(left6up);
		delay(30);
		right1.write(right1up);          // tell servo to go to position in variable 'pos'
		right2.write(right2up);
		right3.write(right3up);
		right4.write(right4up);
		right5.write(right5up);
		right6.write(right6up);
		delay(30);

		////////////////////////////////////////////////////////////////////
		*/
}

/*
vcellMAX17043() returns a 12-bit ADC reading of the battery voltage,
as reported by the MAX17043's VCELL register.
This does not return a voltage value. To convert this to a voltage,
multiply by 5 and divide by 4096.
*/
unsigned int vcellMAX17043(){
	unsigned int vcell;

	vcell = i2cRead16(0x02);
	vcell = vcell >> 4;  // last 4 bits of vcell are nothing

	return vcell;
}

/*
percentMAX17043() returns a float value of the battery percentage
reported from the SOC register of the MAX17043.
*/
float percentMAX17043(){
	unsigned int soc;
	float percent;

	soc = i2cRead16(0x04);  // Read SOC register of MAX17043
	percent = (byte) (soc >> 8);  // High byte of SOC is percentage
	percent += ((float)((byte)soc))/256;  // Low byte is 1/256%

	return percent;
}

/*
configMAX17043(byte percent) configures the config register of
the MAX170143, specifically the alert threshold therein. Pass a
value between 1 and 32 to set the alert threshold to a value between
1 and 32%. Any other values will set the threshold to 32%.
*/
void configMAX17043(byte percent){
	if ((percent >= 32)||(percent == 0))  // Anything 32 or greater will set to 32%
		i2cWrite16(0x9700, 0x0C);
	else{
		byte percentBits = 32 - percent;
		i2cWrite16((0x9700 | percentBits), 0x0C);
	}
}

/*
qsMAX17043() issues a quick-start command to the MAX17043.
A quick start allows the MAX17043 to restart fuel-gauge calculations
in the same manner as initial power-up of the IC. If an application's
power-up sequence is very noisy, such that excess error is introduced
into the IC's first guess of SOC, the Arduino can issue a quick-start
to reduce the error.
*/
void qsMAX17043(){
	i2cWrite16(0x4000, 0x06);  // Write a 0x4000 to the MODE register
}

/*
i2cRead16(unsigned char address) reads a 16-bit value beginning
at the 8-bit address, and continuing to the next address. A 16-bit
value is returned.
*/
unsigned int i2cRead16(unsigned char address){
	int data = 0;

	Wire.beginTransmission(MAX17043_ADDRESS);
	Wire.write(address);
	Wire.endTransmission();

	Wire.requestFrom(MAX17043_ADDRESS, 2);
	while (Wire.available() < 2)
		;
	data = ((int) Wire.read()) << 8;
	data |= Wire.read();

	return data;
}

/*
i2cWrite16(unsigned int data, unsigned char address) writes 16 bits
of data beginning at an 8-bit address, and continuing to the next.
*/
void i2cWrite16(unsigned int data, unsigned char address){
	Wire.beginTransmission(MAX17043_ADDRESS);
	Wire.write(address);
	Wire.write((byte)((data >> 8) & 0x00FF));
	Wire.write((byte)(data & 0x00FF));
	Wire.endTransmission();
}
```