Digital Braille Watch

Nick Anderson – Team Leader
Andrew Hanske – BWIG
Nick Thate – Communicator
Billy Zuleger – BSAC

May 5, 2010

Client: Holly and Colton Albrecht
Advisor: Dennis Bahr

University of Wisconsin – Madison : Biomedical Engineering

Table of Contents

Abstract

The visually impaired rely on the Braille system to read and write. However, there isn't a current watch design that utilizes Braille. A watch of this type would allow the visually impaired to read the time accurately, discretely and efficiently. Current watch designs for the visually impaired include an analog tactile watch and an audio watch. However, the analog tactile watch is difficult to read and breakable while the audio watch is disruptive and draws attention to the user. Since the current methods are inadequate, a Digital Braille Watch was designed. The final prototype uses eight rotating disks with four Braille dots apiece. These disks can be rotated to 90, 180 or 270 degrees to orientate in the correct position and display the correct time. The final prototype was constructed and programmed with an Arduino Duemilanove USB board, eight servo motors, and various circuit elements. These components were then placed in an acroylonitrile-butadiene-stryene (ABS) case, which contained a display window through which the time could be read. Initial testing was completed, confirming the benefits of the digital Braille watch design. Future work will consider minimizing size and power consumption, along with allowing visually impaired individuals to test the device.

Background

Problem Statement

In order to independently determine the time, the visually impaired currently depend on audio or tactile analog watches. However, audio watches are disruptive, while the analog tactile watches are often fragile and difficult to read. Our goal is to develop a digital Braille watch that will efficiently display the time without the issues of the current technologies. This watch should display military time, be accurate and reliable, and utilize the standard Braille numerals.

Braille Basics

The Braille system is the most common writing and reading method used by the visually impaired. Each character is composed of a three row by two column grid of potential dot positions. Raised dots are placed in the six positions, using various combinations to denote different alphabetical and numerical characters. The reader identifies the characters by running his or her finger over the raised dots.

Certain size specifications exist to ensure that Braille is easy to read. Braille dots should be raised 0.019 inches off of a surface, with a base diameter of 0.057 inches. Distances between dots within a given character and between adjacent characters should be 0.092 inches and 0.245 inches center-to-center, respectively [1].

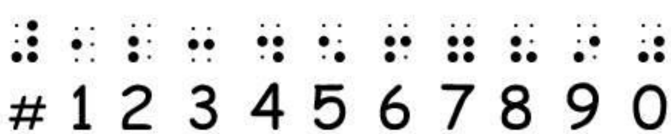


Figure 1 – Nemeth Braille numerals 0-9
Image courtesy of Dotless Braille:
http://www.dotlessbraille.org/AboutBraille.htm

An interesting note about Braille numerals is that they only use the bottom four positions of the three by two grid (Figure 1). Since the watch design only needs to display numerals, the complexity of the design can be reduced by displaying each number using a two by two grid.

Current Methods

There are two main categories of watch products currently on the market for the visually impaired: audio watches and analog tactile watches. Audio watches, informally called talking watches, function by verbally relaying the time to the user whenever the user presses a button (Figure 2). This method is very effective in communicating the time; however, it can be very disruptive and potentially embarrassing for the user. The time is announced to anyone in the vicinity and can attract

Figure 2 – The Audible Watch announces the time to the user when prompted
Image courtesy of Tel-Time: http://ecx.images-amazon.com/images/I/41hK4nPA-zL._SL500_AA280_.jpg

unwanted attention to the visually impaired individual. Analog tactile watches, on the other hand, are silent (Figure 3). They function much like traditional analog watches, except in this case in order to tell the time the user must touch the face of the watch and feel where the hands are located. There are also raised markings on the watch that indicate the positions of the numbers; however, there is no standard format for these markings and they vary from product to product. The client has informed us that these watches can be difficult to read and come with a learning curve when they are first used. Also, the hands of these watches are exposed while the user is telling the time, and therefore they can be easily broken or damaged.

Figure 3 – The visually impaired touch the hands of the tactile analog watch to tell the time
Image courtesy of Independent Living Aids, LLC:
http://www.timebooth.com/wordpress/wpcontent/uploads/2009/05/braillehitouchwatchax7.jpg

Figure 4 – Haptica Braille Watch design by David Chavez
Image courtesy of Tuvie Design of the Future:
http://www.tuvie.com/haptica-braille-watch-concept/

In addition to these currently available watch products, there is a watch that has recently been designed called the Haptica Braille Watch (Figure 4). This design features a set of 16 rotating disks that circulate Braille dots in and out of the display to assemble the desired Braille numerals. Each disk contains a single Braille dot that is moved in concert with the other disks to display the time in Braille. This concept was created by David Chavez in 2008. Chavez is not an engineer and, to our knowledge, has not yet created a prototype for his design[2].

Design Criteria and Considerations

Design Specifications

Our clients for this project are Holly and Colton Albrecht. Colton is Holly's visually impaired son; together, they came up with the idea for the digital Braille watch. As such, the project will be created in accordance to their wishes and specifications. Their main requirements are that the design is able to correctly display the current time in standard Braille, utilize military time, and operate without any noise. The watch must not be dangerous to the user, so moving parts and electronic components must be contained properly. It has to be accurate within the minute whenever it is connected to a power source. Holly doesn't require that the prototype is any particular size; rather she is looking for a proof of concept. However, the watch should be designed so that it would be possible to scale down to watch-size in the future. For more information on the product design specifications, see Appendix A.

Funding

Since caring for a visually impaired child can be financially taxing, it is difficult for the client to provide funding for this project. As a result, outside sources were contacted to try to offset the financial burden on Holly and Colton. Based on advice from Dennis Bahr, a budget proposal (Appendix B) was written and sent to the Madison Institute of Electrical and Electronics Engineers (IEEE). The proposal asked the IEEE, which Dennis is a member of, for $500 towards the completion of the prototype. The team hopes that the IEEE will grant sufficient funding for this project so that none of the clients money has to be used.

Vibrating Dots

The task of designing a Braille watch has been presented to two previous BME design groups; this design is a continuation of what our predecessors have accomplished. It features four vibrating motors that vibrate in sequence to communicate the time (Figure 5). When the user presses a button on the watch, the four motors vibrate to signify one Braille numeral, and, after a short pause, they vibrate again to denote the second numeral. This process is repeated until all four numerals have been relayed to the user. A major obstacle that the previous groups faced was the challenge of separating the vibrations of the four motors in order to provide accurate readings. The last group accomplished this by placing the motors on the outside of the watch and inserting a piece of viscoelastic foam between the motors and the display. Strands of copper wire were wrapped around each motor and led up around the foam to the display. The user would feel the tips of these wires to tell the time.

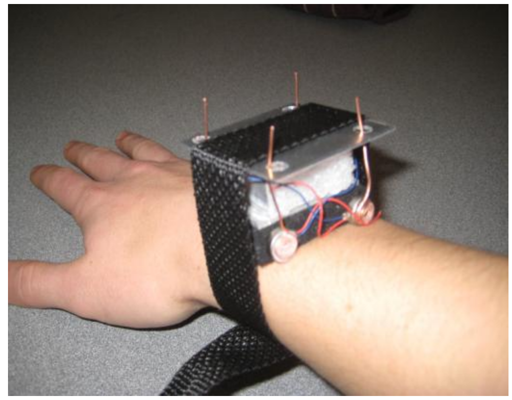


Figure 5 – Vibrating dots prototype created by past BME design team
Image courtesy of BME 200 Fall 2008 Design Team

Image courtesy of Arduino:
http://arduino.cc/en/uploads/Tutorial/forLoop_schem.pngchf_08/secure/

The fall of 2008 team that worked on this project was able to construct a working prototype of this design; however, it has some major draw backs. Colton informed us that many visually impaired people have increased sensitivity to touch in their hands. This is due to the fact that many of them use their fingers to detect subtle tactile differences, including those encountered while reading Braille. As a result, the vibrations of this prototype have an over-stimulating effect. Also, because of the size of the watch, the user must use their entire hand to read the device. This ends up creating an experience that is very different from actual Braille reading, which only requires one finger tip. In addition to posing this challenge, the design is also relatively power hungry. Watches are devices that are expected to run for long periods of time while using limited amounts of power. The act of generating vibrations is very power-intensive compared to the other mechanisms of a normal watch. This means that this watch would not last very long compared to other watches, and its power source would have to be replaced or recharged more frequently. After meeting with Holly and Colton, they informed us that they did not approve of this design and they want us to take a different course of action with this project[3].

Actuating Dots

As suggested by Holly, the design search began by looking at HumanWare's BrailleNote system (Figure 6). This device has an 18 character refreshable display and is used as a link for the visually impaired to the digital world. To form the display each Braille dot is a pin that actuates up or down through a flat surface (Figure 7). This gives a familiar Braille configuration similar to feeling bumps on a sheet of paper.

Figure 6 – HumanWare's Braille Note is used by the visually impaired to read computer screens
Image courtesy of HumanWare
http://www.anu.edu.au/disabilities/atproject/BrailleNote/index.php

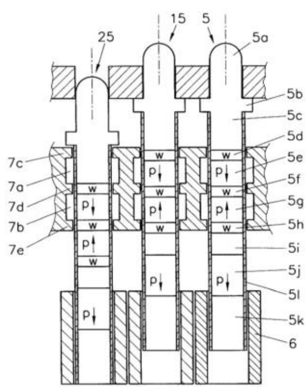


Figure 7 – Actuating dots mechanism
Image courtesy of Litschel, Dietmar, and Schwertner

After doing patent research for possible mechanisms to drive the pins up and down, it was found that many solutions have been proposed. Some of these included attaching the pins to a spring mechanism or an electromagnetic solenoid[4]. These mechanisms are highly complicated and mechanically intense. For the display, 16 individual actuating mechanisms were needed to display the four numerals, making for a complicated product. This complexity would make it exceedingly difficult to both create a prototype and to eventually be able to scale it down to actual wristwatch size. Also, since the pins would be driven up and down many times, this mechanism would also use quite a bit of power; causing it to need a power source larger then those commonly found in wristwatches.

Rotating Disks

This design uses eight rotating disks to form the required Braille numerals. Each disk would have two raised dots, which could be configured to form the top or bottom half of the character cell (Figure 8). When read by the user, this design would create the sensation of classic Braille.

This design was inspired by the Haptica Braille Watch concept proposed by David Chavez. Although the designs share the concept of having disks rotate in and out of the display area, this design has a much easier mechanism. By placing two dots on each disk instead of one, the amount of disks needed to form the display is reduced from 16 to 8. This reduction of moving parts has several critical benefits. There would be considerably less power consumption for each reconfiguration. In addition, it would be easier to scale down a prototype to wristwatch size.

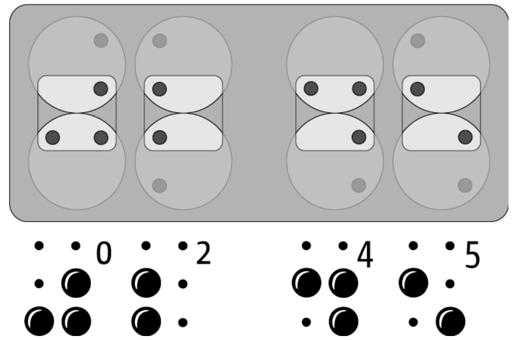


Figure 8 – By a rotation of 90º, 180º or 270º, the disks can display the correct time

Design Evaluation

The vibrating dots, actuating dots, and rotating disks designs were evaluated on a scale of one to ten and weighted for a variety of design criteria (Table 1). The most important criteria were given more weight in the matrix, which include ergonomics, aesthetics, accuracy, and design simplicity. These aspects were determined to be the most important design characteristics since they are critical in terms of the ease of use and effective functionality of the final product.

Table 1 – The design matrix displays the design evaluation on a scale of one to ten (one = poor, ten = excellent) and is weighted on a variety of design criteria for all three design concepts.

Weight	Design Aspects	Vibrating Dots	Actuating Dots	Rotating Disks
0.05	Prototype Cost	8	6	7
0.15	Aesthetics	4	7	9
0.25	Ergonomics	3	8	9
0.05	Safety	10	9	9
0.10	Durability	9	6	8
0.15	Accuracy	7	10	10
0.15	Design Simplicity	9	4	7
0.10	Scalability	8	6	7
1	**Total**	**6.35**	**7.10**	**8.45**

Ergonomics was weighted most heavily in the design matrix since the two current methods and the previous teams' designs were deficient in this area. Also, it was decided that ergonomics is the most important criteria in functionality and success of a watch design. The tactile analog watch is difficult for the user to read, while the audio watch doesn't allow the user to discretely check the time. The vibrating motors design provided an over-stimulation, making it almost impossible to read. The actuating disks and rotating disks designs both provide an easy way for the user to discretely and accurately check the time. Aesthetics also was weighted heavily because the watch shouldn't draw attention to the visually impaired person using it. Since it's a watch, the design must provide an accurate display of the time. The design must also be simple in order to reduce cost, increase durability, and enhance performance. After evaluating the designs, it was determined that the rotating disks design scored the highest and, therefore, was selected as the design to pursue.

Prototype

Servos

Figure 9– A high torque full turn GW servo

Image courtesy of Acroname Robotics: http://www.acroname.com/robotics

Radio-controlled (RC) high torque full turn servos were used to rotate the disks (Figure 9). These servos were 1.6" X 0.78" X 1.5" and are power efficient due to their pulse-width modulation control. The small size of the servos allows them to be easily orientated in a way that corresponds with the design (Figure 10). Pulse-width modulation means that a servo requires a short electric pulse in order to rotate a desired angle. For example, many servos require 3V for a duration of 1.5 milliseconds in order to rotate ninety degrees. The power efficiency and small size of servos is what makes them ideal for this project. However, for prototyping purposes, one of the greatest

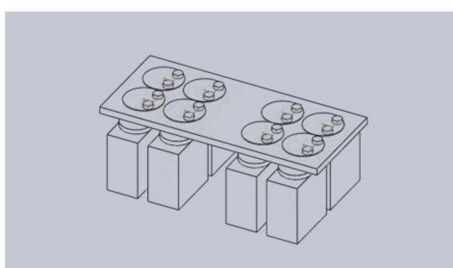


Figure 10 – The rotating disks design will include eight servos, which each control a different disk in the display

attributes of servos is that they can be controlled by a microprocessor[5].

Disks

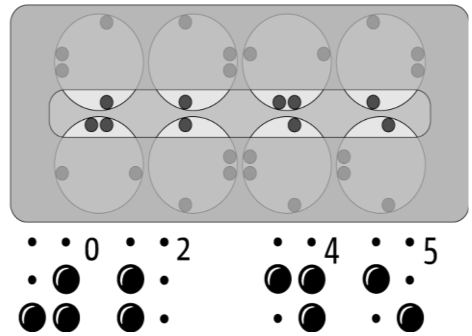


Figure 11– The new four-dot disk design brings the dots closer together

The original two-dot disk design (Figure 8) was modified to a four-dot design (Figure 11). The new design allows the dots to be closer together and better resemble traditional Braille character size. To fabricate the disks, a 7/8" diameter acroylonitrile-butadiene-stryene (ABS) rod was cut into 1/4" tall disks. Size 0 Phillips screws were then inserted into small, pre-tapped holes to form the dots. Additionally, a hole was drilled into the bottom of each disk so they could be easily attached to the servos. A complete fabricated disk can be seen in Figure 12.

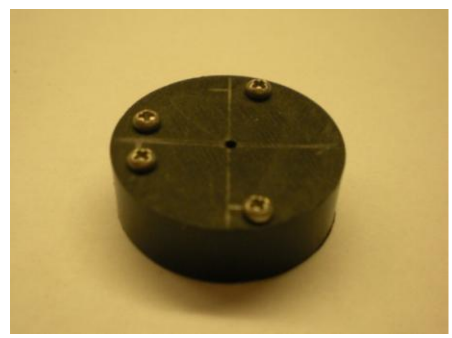


Figure 12– The Braille disk cut from an ABS rod with size 0 Phillips screws as dots

Programming

The prototype was programmed using an Arduino Duemilanove USB Board (Figure 13). Arduino is an open-source computing platform based on an input/output board. It functions using an ATMega328 microcontroller that implements the wiring programming language. Wiring is an open source form of Java specifically used for electronics with input/output boards. By downloading the Arduino-0017 and Wiring-0022 programs, code could then be uploaded to the Arduino Board via an USB cord[6].

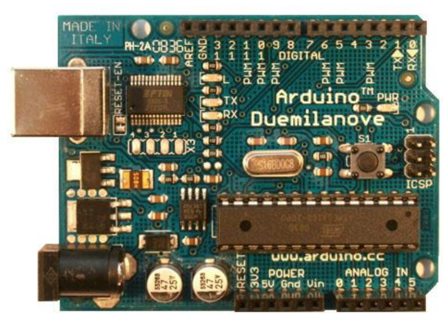


Figure 13 – The Arduino Duemilanove will be used to program the watch
Image courtesy of Arduino:
http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

The selected timing chip was a DS1307 Real Time Clock Module (Figure 14). This chip easily interfaces with the Arduino when connected in the configuration seen in Figure 15. By downloading the

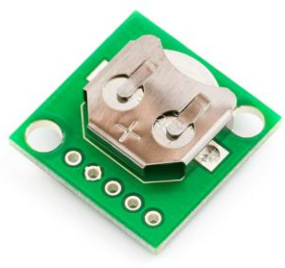


Figure 14– DS1307 Real Time Clock Module

Image courtesy of Active Robots: http://www.active-robots.com/products/components/real-time-

code found in Appendix C, the timing chip was initially programmed so that it contained the correct time and date. This clock module was helpful for this project because after it was initially set, it kept the correct time and date, accurate to within a second. It is internally powered, so it is able to function even when the Arduino is not connected to a power source.

The code for the final prototype interfaced eight full turn servos with the Real Time Clock Module. Using the same configuration seen in Figure 15, the Arduino received the correct date and time from the timing chip and sent the appropriate signal to each of the eight servos. This signal rotated each servo to the desired angle, leading to the configuration that displayed the correct numerals.

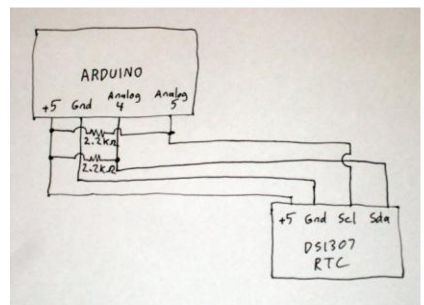


Figure 15– Circuitry for the timing chip

Image courtesy of Hobby Robotics: http://www.glacialwanderer.com/hobbyrobotics/?p=12_schem.png

In order to achieve the desired functionality, two buttons were also attached to the Arduino. The first button changed the mode of the watch, while the other updated the watch. The watch has three modes: the first displays the time in hours and minutes, the next shows only the minutes and seconds, and the final mode displays the month and day. When the update button is pushed, the updated information of the respective mode is displayed. The servos do not move unless this button or the mode button is pressed, preventing the constant movement of the servos and saving power. The complete code of the prototype can be seen in Appendix D.

Circuitry

The constructed circuit contained eleven elements: a timing chip, two buttons and eight servos. Much of the circuitry was determined by looking at example projects. Figure 15 demonstrates the wiring used for the timing chip, while Figure 16 is a representation of how each of the buttons was configured. Each of the servos contained a ground, power and input wire. The power wire was connected to the 5V pin on the Arduino, while the ground wire was connected to the ground pin. The input wire for each servo was connected to a different input pin on the Arduino.

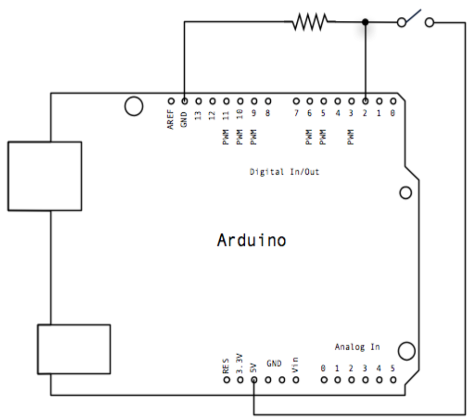


Figure 16– Circuitry for reset button

Image courtesy of Arduino: http://arduino.cc/en/uploads/Tutorial/button_schem.png

Proto-shield

The proto-shield (Figure 17) used was a soldering board built specifically for the Arduino board. By soldering stackable female connectors onto the proto-shield, the pins on the Arduino board were brought to the top of the proto-shield. The eight servos, mode button, update button, and the timing chip were then soldered to the proto-shield using the wiring described in the Circuitry section.

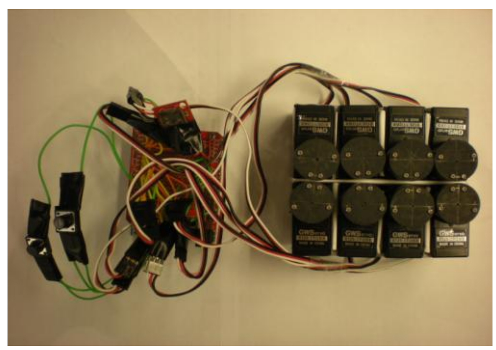


Figure 17– The proto-shield with all of its circuitry elements connected to the Arduino board

Casing

The Arduino microcontroller and servos were housed in a two-piece 6.125" x 4.620" x 2.255" ABS casing. A 1/2" wide cut along the center of the cover was made to form the display window, and two 5/16" diameter holes were drilled to the left of the display window. The top hole was for the update button, while the other was for the mode button. An additional cut was made in the back of the casing so that the USB port of the Arduino is exposed, allowing for powering and reprogramming of the device.

Assembly

When assembling the watch, minimizing its size and efficiently utilizing the space of the casing was crucial. In order to achieve this, the servos had to be modified. The servos originally contained mounting attachments (Figure 9), which were cut off to allow the servos to be placed closer together. In addition, the servo wires exited from the front side of the servo. This made it impossible to have the Braille disks touching and reduced the quality of the display. To overcome this obstacle, holes were cut in the bottom of the servos and the wiring was led beneath them.

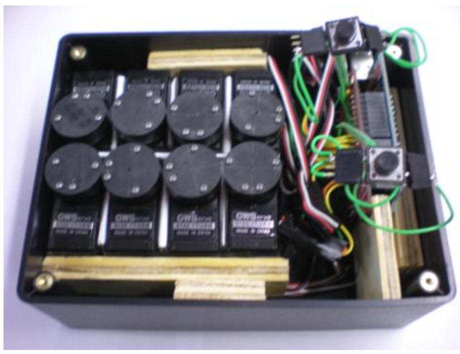


Figure 18– The assembled prototype, seen without the cover

The disks were about one millimeter wider than the width of the servo. In order to properly space the servos, a 2 mm piece of folded paper was placed between them. The servos were then taped together into one cohesive block to ensure that the display would be stable. The Braille disks were adhered to the servos using Gorilla Glue. The Arduino, proto-shield, and servos were held in place inside the casing by wedging 1/3” thick pieces of plywood into the container. This successfully kept all of the components in place while still allowing for easy disassembly. The complete assembly of the prototype within the casing can be seen in Figure 18.

Testing and Results

In order to gain useful feedback on the relative merits of the design, a survey was sent to the Wisconsin Center for the Blind and Visually Impaired. Colton attends the school, so several of his teachers were contacted to ask if they or the students would be willing to answer a few brief questions about the Digital Braille Watch project. The questions gauged the user’s familiarity with current time-telling methods, and they were asked to rate their level of agreement with several statements regarding the Digital Braille Watch and the competition. The full attachment sent to the school can be seen in Appendix E.

All the respondents said that they were familiar with either the tactile analog or audible watch devices. While there were mixed responses regarding whether or not the current methods are adequate, only 14.3% of those surveyed thought that the Digital Braille Watch would not be an improvement upon the existing time-telling methods (Figure 19). When asked to comment on the design, one of the respondents stated, "I strongly encourage the development of a digital watch for those whose personal needs or interests would welcome such a device." This further demonstrates the need for a watch similar to the prototype that has been created.

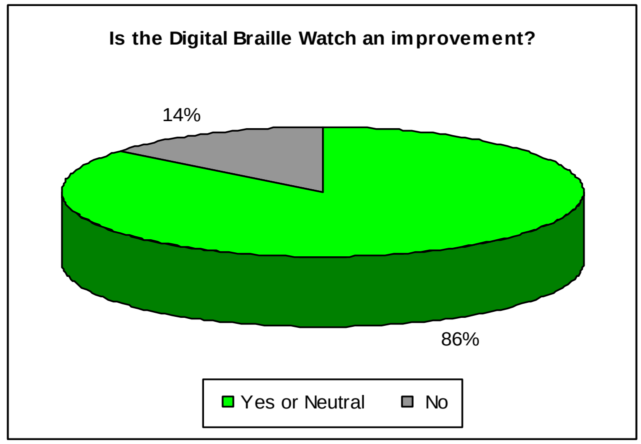


Figure 19– Pie chart demonstrating that the Digital Braille Watch is an improvement upon the current devices

Management and Planning

At the beginning of the semester, the Gantt chart found in Appendix F was created as a work and time management tool. It was roughly followed throughout the semester to ensure that the project was on track and deliverables were finished on time. All expenses were recorded throughout the semester as well. The projects total cost was $413.64. A detailed list of the expenses can be found in Appendix G.

Future Work

Using eight rotating disks that configure to form Braille numerals, the Digital Braille Watch prototype is able to effectively display the current time in Braille for visually impaired individuals. Since Holly and Colton emphasized a proof-of-concept, this prototype was largely a success. However, there are still various improvements that can be implemented to optimize the current design.

Perhaps the largest and most obvious concern is to minimize the size of the watch. The Digital Braille Watch prototype is housed in a 6.125" x 4.620" x 2.255" casing, which is much larger than a wristwatch. The mass of the watch must be reduced so it is comparable to the size of standard wrist or pocket watches available on the market. In order to accomplish this, the servos must be eliminated. The grouping of eight servos took up the majority of room in the casing, and small enough servos are not available to create a reasonably sized pocket or wristwatch. As such, a different mechanism for disk rotation would have to be used. The Arduino microcontroller is also too large to use in an ideal Braille watch. By using a microcontroller, it was easy to interface the timing chip with the servos, but this could likely be accomplished with different circuitry methods that would take up less space.

In addition to size, the power consumption of the watch must also be minimized so that it can run for extended periods of time without being connected to a USB or AC supply. The watch should be able to run off of an internal battery supply, much like a standard digital or analog watch. While the real time clock module is effective in communicating the time, it also has significant drift. This means that the timing mechanism is not perfect, which can lead to an inaccurate time display after an extended period of time. A more accurate time-keeping mechanism would be to integrate a WWVB radio-receiver. The time is sent to this chip via a radio signal, ensuring that it remains accurate over long durations of time[7].

In addition to these hardware changes, additional features could be added to the watch, such as an alarm clock or stopwatch, making the watch more similar to standard digital watches on the market. By allowing more people to test the prototype, a greater understanding of their desires in a watch and opinions on further features can be established. Finally, the watch should be adapted so that its components can be easily be mass-produced and assembled. When altered for mass assembly, the human error encountered in machining and assembling the device will be eliminated.

Conclusion

A fully functional Digital Braille Watch was successfully created using a rotating disk design. The final prototype met all of the major design specifications, and, based on the preliminary survey results, it was well received by a sample of visually impaired students. The design shows promise for becoming a marketable product that addresses all of the shortcomings of the current time-telling devices available for the visually impaired. The prototype is effective in proving the functionality of the design, but in order for it to be commercially ready, there needs to be several improvements. These include minimizing the size of the design, changing the watch's power source, and enhancing its function. This innovative prototype demonstrates that it is possible to create a Digital Braille Watch that is silent, is easy to read and improves the day-to-day life of the visually impaired.

References

1. “Size and Spacing of Braille Characters.” *Braille Authority of North America*. n.d. 27 Jan. 2010. <http://www.Brailleauthority.org/sizespacingofBraille/>.

2. "Haptica Braille Watch Concept". *Tuvie Design of the Future*. 2009. 25 Jan. 2010. <http://www.tuvie.com/haptica-Braille-watch-concept/>.

3. "Braille Watch". *UW-Madison Biomedical Engineering Design Courses – Project Pages*. 2008. 11 Feb. 2010.

4. Litschel, Dietmar, and Schwertner. Device for Representing Relief Items. Caretec GmbH, assignee. US Patent 6109922. 2000.

5. “What’s a Servo?” Seattle Robotics Society .n.d. 26 Feb. 2010. <http://www.seattle robotics.org/guide/servos.html>.

6.“Arduino Duemilanove.” *Arduino*. n.d. 2 Mar. 2010. <http://arduino.cc/en/Main/ ArduinoBoardDuemilanove?action=diff>.

7. “WWVB Radio Controlled Clocks.” National Institute of Standards and Technology: Time and Frequency Division. 2010. 30 Mar. 2010. <http://tf.nist.gov/stations/radioclocks.htm >.

Appendix A: Product Design Specifications

Project Design Specifications—Digital Braille Watch

February 17, 2010

Team: Nick Anderson, Nick Thate, Andrew Hanske, Billy Zuleger
Client: Holly and Colton Albrecht
Advisor: Dennis Bahr

Problem Statement:

To determine the time, the visually impaired currently depend on audio or tactile analog watches. However, audio watches are disruptive, while the analog tactile watches are often difficult to read and fragile. Our goal is to develop a digital Braille watch that will efficiently display the time without the issues of the current technologies. This watch should display military time, be accurate and reliable, and utilize the standard Braille numerals.

Client Requirements:

- Digital military time display
- Silent and without vibrations
- Time in standard Braille

Design Requirements:

1) Design Requirements
    a) *Performance requirements:* See Client Requirements above
    b) Safety: All electronics must be contained and the watch must not contain hazardous materials
    c) *Accuracy and Reliability*: The watch must accurately display military time within the minute
    d) *Life in Service*: The watch must be able function continuously while connected to a power source
    e) *Shelf Life*: Not specified for prototype
    f) *Operating Environment*: The device must be able to operate reliably in a dry environment
    g) *Ergonomics*: The watch should not contain rough edges or loose components and the display surface should be easy to read
    h) *Size*: The prototype does not need to be watch-sized but should be scalable
    i) *Weight*: See Size Requirement
    j) *Materials*: The device must comprise of non-toxic components
    k) *Aesthetics, Appearance, and Finish*: The watch should be aesthetically pleasing
2) Product Characteristics
    a) *Quantity*: One working prototype
    b) *Target Product Cost*: $100 or less when mass-produced

3) Miscellaneous
   a) *Standards and Specifications*: Must display time according to the standard Braille language
   b) *Customer*: The customer would like a device that physically displays the time using Braille digits
   c) *Patient Related Concerns*: None
   d) *Competition*: Audio and tactile analog watches are commercially available for the visually-impaired

Appendix B: Budget Proposal

February 25, 2010

Dear Madison Institute of Electrical and Electronics Engineers,

I'm writing to you on behalf of my engineering team to request funding for our Biomedical Engineering (BME) design project. The goal of our project is to design a digital Braille watch that the visually impaired can use to accurately and reliably read the time.

Our clients, who consist of a mother and her son who is visually impaired, introduced this idea to the UW Madison BME department over two years ago. Current watches that exist for the visually impaired include an analog tactile watch and an audio watch. However, our clients expressed that the analog watch can be difficult to read and that the audio watch can be disruptive. Upon further research, we discovered that many of the visually impaired have experienced similar problems.

Our idea is to design a watch that consists of eight disks. The disks will be oriented in two rows of four, and each will have two Braille dots on its surface. Each disk will be attached to and rotated by a servo, each of which will receive a signal from the same microprocessor. The microprocessor will be used to send the appropriate signal to each servo. This will correctly position the disks and, thus, display the correct time. We are currently working on a SolidWorks drawing that clearly demonstrates this design, and we would be willing to send this to you as soon as it is completed.

As parenting a visually impaired child can be financially straining, it is difficult for our client to provide funding for this project. I'm in turn hoping that you would be willing to support our team in assisting this great cause. We estimate that it will cost us approximately $500 to construct a working prototype. Any dollar amount would be greatly appreciated, but sufficient funding would ensure that we could be successful on this urgent project.

On behalf of my team, I would like to thank you for your time. We hope you are interested in supporting this wonderful cause and look forward to hearing from you soon.

Sincerely,

Nick Anderson

Appendix C: Real Time Clock Module Code

```
/*
 * RTC Control v.01
 * by <http://www.combustory.com> John Vaughters
 * Credit to:
 * Maurice Ribble - http://www.glacialwanderer.com/hobbyrobotics for RTC
DS1307 code
 *
 * With this code you can set the date/time, retreive the date/time and use
the extra memory of an RTC DS1307 chip.
 * The program also sets all the extra memory space to 0xff.
 * Serial Communication method with the Arduino that utilizes a leading CHAR
for each command described below.
 * Commands:
 * T(00-59)(00-59)(00-23)(1-7)(01-31)(01-12)(00-99) -
T(sec)(min)(hour)(dayOfWeek)(dayOfMonth)(month)(year) - T Sets the date of
the RTC DS1307 Chip.
 * Example to set the time for 02-Feb-09 @ 19:57:11 for the 3 day of the
week, use this command - T1157193020209
 * Q(1-2) - (Q1) Memory initialization  (Q2) RTC - Memory Dump
 */

#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68  // This is the I2C address


// Global Variables

int command = 0  // This is the command char, in ascii form, sent from the
serial port
int i;
long previousMillis = 0;        // will store last time Temp was updated
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
byte test;

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
  return ( (val/10*16) + (val%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
  return ( (val/16*10) + (val%16) );
}

// 1) Sets the date and time on the ds1307
// 2) Starts the clock
// 3) Sets hour mode to 24 hour clock
// Assumes you're passing in valid numbers, Probably need to put in checks
```

```
for valid numbers.

void setDateDs1307()
{

   second = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
   minute = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
   hour  = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
   dayOfWeek = (byte) (Serial.read() - 48);
   dayOfMonth = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
   month = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
   year= (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.send(0x00);
   Wire.send(decToBcd(second));    // 0 to bit 7 starts the clock
   Wire.send(decToBcd(minute));
   Wire.send(decToBcd(hour));   // If you want 12 hour am/pm you need to set
                                   // bit 6 (also need to change
readDateDs1307)
   Wire.send(decToBcd(dayOfWeek));
   Wire.send(decToBcd(dayOfMonth));
   Wire.send(decToBcd(month));
   Wire.send(decToBcd(year));
   Wire.endTransmission();
}

// Gets the date and time from the ds1307 and prints result
void getDateDs1307()
{
  // Reset the register pointer
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0x00);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  // A few of these need masks because certain bits are control bits
  second     = bcdToDec(Wire.receive() & 0x7f);
  minute     = bcdToDec(Wire.receive());
  hour       = bcdToDec(Wire.receive() & 0x3f
  dayOfWeek  = bcdToDec(Wire.receive());
  dayOfMonth = bcdToDec(Wire.receive());
  month      = bcdToDec(Wire.receive());
  year       = bcdToDec(Wire.receive());

  Serial.print(hour, DEC);
  Serial.print(":");
  Serial.print(minute, DEC);
  Serial.print(":");
  Serial.print(second, DEC);
  Serial.print("  ");
  Serial.print(month, DEC);
  Serial.print("/");
```

```
  Serial.print(dayOfMonth, DEC);
  Serial.print("/");
  Serial.print(year, DEC);

}


void setup() {
  Wire.begin();
  Serial.begin(57600);

}

void loop() {
     if (Serial.available()) {      // Look for char in serial que and
process if found
      command = Serial.read();
      if (command == 84) {      //If command = "T" Set Date
       setDateDs1307();
       getDateDs1307();
       Serial.println(" ");
      }
      else if (command == 81) {  //If command = "Q" RTC1307 Memory Functions
        delay(100);
        if (Serial.available()) {
         command = Serial.read();
         if (command == 49) {
          Wire.beginTransmission(DS1307_I2C_ADDRESS);
          Wire.send(0x08); // Set the register pointer
         for (i = 1; i <= 27; i++) {
             Wire.send(0xff);
            delay(100);
         }
         Wire.endTransmission();
         getDateDs1307();
         Serial.println(": RTC1307 Initialized Memory");
         }
         else if (command == 50) {    //If command = "2" RTC1307 Memory Dump
          getDateDs1307();
          Serial.println(": RTC 1307 Dump Begin");
          Wire.beginTransmission(DS1307_I2C_ADDRESS);
          Wire.send(0x00);
          Wire.endTransmission();
          Wire.requestFrom(DS1307_I2C_ADDRESS, 64);
          for (i = 1; i <= 64; i++) {
             test = Wire.receive();
             Serial.print(i);
             Serial.print(":");
             Serial.println(test, DEC);
          }
          Serial.println(" RTC1307 Dump end");
         }
        }
```

```
    }
  Serial.print("Command: ");
  Serial.println(command);   // Echo command CHAR in ascii that was sent
  }

  command = 0;                   // reset command
  delay(100);
}
```

Appendix D: Final Prototype Code

```
//
// Maurice Ribble
// 4-17-2008
// http://www.glacialwanderer.com/hobbyrobotics

// This code tests the DS1307 Real Time clock on the Arduino board.
// The ds1307 works in binary coded decimal or BCD.  You can look up
// bcd in google if you aren't familior with it.  There can output
// a square wave, but I don't expose that in this code.  See the
// ds1307 for it's full capabilities.
#include <Servo.h>
#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;
Servo servo7;
Servo servo8;

const int  buttonPin1 = 11;    // the pin that the pushbutton is attached to
const int  buttonPin2 = 12;    // the pin that the pushbutton is attached to
int buttonPushCounter2 = 0;   // counter for the number of button presses
int buttonState1 = 0;         // current state of the button
int buttonState2 = 0;         // current state of the button
int lastButtonState1 = 0;     // previous state of the button
int lastButtonState2 = 0;     // previous state of the button
int onePress = 0;

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
  return ( (val/10*16) + (val%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
  return ( (val/16*10) + (val%16) );
}

// Stops the DS1307, but it has the side effect of setting seconds to 0
// Probably only want to use this for testing
/*void stopDs1307()
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
```

```
  Wire.send(0);
  Wire.send(0x80);
  Wire.endTransmission();
}*/

// 1) Sets the date and time on the ds1307
// 2) Starts the clock
// 3) Sets hour mode to 24 hour clock
// Assumes you're passing in valid numbers
void setDateDs1307(byte second,        // 0-59
                   byte minute,        // 0-59
                   byte hour,          // 1-23
                   byte dayOfWeek,     // 1-7
                   byte dayOfMonth,    // 1-28/29/30/31
                   byte month,         // 1-12
                   byte year)          // 0-99
{
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.send(0);
   Wire.send(decToBcd(second));    // 0 to bit 7 starts the clock
   Wire.send(decToBcd(minute));
   Wire.send(decToBcd(hour));      // If you want 12 hour am/pm you need to
set
                                   // bit 6 (also need to change
readDateDs1307)
   Wire.send(decToBcd(dayOfWeek));
   Wire.send(decToBcd(dayOfMonth));
   Wire.send(decToBcd(month));
   Wire.send(decToBcd(year));
   Wire.endTransmission();
}

// Gets the date and time from the ds1307
void getDateDs1307(byte *second,
          byte *minute,
          byte *hour,
          byte *dayOfWeek,
          byte *dayOfMonth,
          byte *month,
          byte *year)
{
  // Reset the register pointer
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  // A few of these need masks because certain bits are control bits
  *second     = bcdToDec(Wire.receive() & 0x7f);
  *minute     = bcdToDec(Wire.receive());
  *hour       = bcdToDec(Wire.receive() & 0x3f);  // Need to change this if
12 hour am/pm
```

```
  *dayOfWeek  = bcdToDec(Wire.receive());
  *dayOfMonth = bcdToDec(Wire.receive());
  *month      = bcdToDec(Wire.receive());
  *year       = bcdToDec(Wire.receive());
}
void setTime(Servo servo1, Servo servo2, int num){
  if (num == 0){
   servo1.write(157);
   delay(50);
   servo2.write(77);
  }
  if (num == 1){
   servo1.write(104);
   delay(50);
   servo2.write(132);
  }
  if (num == 2){
   servo1.write(104);
   delay(50);
   servo2.write(157);
  }
  if (num == 3){
   servo1.write(77);
   delay(50);
   servo2.write(132);
  }
  if (num == 4){
   servo1.write(77);
   delay(50);
   servo2.write(104);
  }
  if (num == 5){
   servo1.write(104);
   delay(50);
   servo2.write(104);
  }
  if (num == 6){
   servo1.write(77);
   delay(50);
   servo2.write(157);
  }
  if (num == 7){
   servo1.write(77);
   delay(50);
   servo2.write(77);
  }
  if (num == 8){
   servo1.write(104);
   delay(50);
   servo2.write(77);
  }
  if (num == 9){
   servo1.write(157);
```

```
    delay(50);
    servo2.write(157);
   }
 }
void setup()
{
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  Wire.begin();
  Serial.begin(9600);

  // Change these values to what you want to set your clock to.
  // You probably only want to set your clock once and then remove
  // the setDateDs1307 call.
  second = 0;
  minute = 58;
  hour = 9;
  dayOfWeek = 4;
  dayOfMonth = 13;
  month = 4;
  year = 10;
  //setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
  servo1.attach(2);
  delay(100);
  servo2.attach(3);
  delay(100);
  servo3.attach(4);
  delay(100);
  servo4.attach(5);
  delay(100);
  servo5.attach(6);
  delay(100);
  servo6.attach(7);
  delay(100);
  servo7.attach(8);
  delay(100);
  servo8.attach(9);
  delay(100);
  servo1.write(132);  // set servo to mid-point
  delay(100);
  servo2.write(132);  // set servo to mid-point
  delay(100);
  servo3.write(132);  // set servo to mid-point
  delay(100);
  servo4.write(132);  // set servo to mid-point
  delay(100);
  servo5.write(132);  // set servo to mid-point
  delay(100);
  servo6.write(132);  // set servo to mid-point
  delay(100);
  servo7.write(132);  // set servo to mid-point
  delay(100);
```

```
  servo8.write(132);  // set servo to mid-point
}

void loop()
{
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

  getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
&year);
  Serial.print(hour, DEC);
  Serial.print(":");
  Serial.print(minute, DEC);
  Serial.print(":");
  Serial.print(second, DEC);
  Serial.print("  ");
  Serial.print(month, DEC);
  Serial.print("/");
  Serial.print(dayOfMonth, DEC);
  Serial.print("/");
  Serial.print(year, DEC);
  Serial.print("  Day_of_week:");
  Serial.println(dayOfWeek, DEC);
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  if (buttonState2 == lastButtonState2){
    onePress = 0;
  }
  if (buttonState2 != lastButtonState2 && onePress == 0){
    buttonPushCounter2++;
    onePress = 1;
  }
  if ((buttonPushCounter2 % 3) == 0){
    if (buttonState1 != lastButtonState1) {
      setTime(servo1, servo2, ((hour - hour % 10) / 10));
      delay(50);
      setTime(servo3, servo4, (hour % 10));
      delay(50);
      setTime(servo5, servo6, ((minute - minute % 10) / 10));
      delay(50);
      setTime(servo7, servo8, (minute % 10));
      delay(50);
    }
  }
  if ((buttonPushCounter2 % 3) == 1){
    if (buttonState1 != lastButtonState1) {
      setTime(servo1, servo2, ((minute - minute % 10) / 10));
      delay(50);
      setTime(servo3, servo4, (minute % 10));
      delay(50);
      setTime(servo5, servo6, ((second - second % 10) / 10));
      delay(50);
      setTime(servo7, servo8, (second % 10));
      delay(50);
```

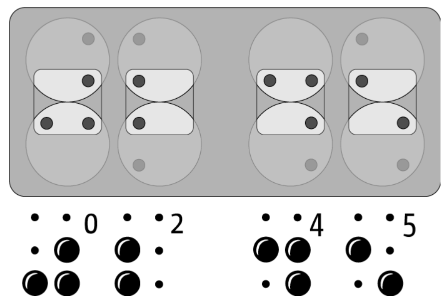
```
    }
  }
  if ((buttonPushCounter2 % 3) == 2){
    if (buttonState1 != lastButtonState1) {
      setTime(servo1, servo2, ((month - month % 10) / 10));
      delay(50);
      setTime(servo3, servo4, (month % 10));
      delay(50);
      setTime(servo5, servo6, ((dayOfMonth - dayOfMonth % 10) / 10));
      delay(50);
      setTime(servo7, servo8, (dayOfMonth % 10));
      delay(50);
    }
  }
}
```

Appendix E: Survey

Digital Braille Watch for the Visually Impaired – Survey

The purpose of this survey is to test the effectiveness and necessity of a design created by an UW-Madison Biomedical Engineering design team for their semester design project. Please read the following problem statement and design description and then fill out the following questions. Thank you!

Problem Statement: In order to determine the time, the visually impaired currently depend on audio or tactile analog watches. However, audible watches are disruptive, while the analog tactile watches are often difficult to read and fragile. Our goal is to develop a digital Braille watch that will efficiently display the time without the issues of the current technologies. This watch should display military time, be accurate and reliable, and utilize the standard Braille numerals.



The disks rotate to display the correct time.

Design Description: To eliminate the current issues experienced with the audible and tactile analog watches, a digital Braille watch has been designed. This watch displays the time in Braille numerals via disks that rotate corresponding to the correct time. This watch is silent and accurate. The watch has added functionality so that it displays the date when a button is pressed. The watch also contains an internal timing chip, so the watch will never have to be set even if the power supply must be replaced.

An example of the final design

Please answer the survey questions by clicking on the following link:
http://www.surveymonkey.com/s/KX52NTJ

Appendix F: Gantt Chart

	January		February				March			April						May
	22	29	5	12	19	26	5	12	19	26	2	9	16	23	30	7
Project Research and Development																
Researching																
Brainstorming																
Design Matrix/Cost Estimation																
Design Selection																
Ordering Materials																
Prototyping																
Testing																
Final Prototype																
Deliverables																
Progress Reports																
PDS																
Mid-semester Presentation																
Mid-semester paper																
Final Presentation																
Final Paper																
Meetings																
Client																
Advisor																

Appendix G: Expenses

Date	Item	Per Unit Cost	Extended Cost
3/12/10	Arduino Duemilanove	$30.00	$37.95
3/12/10	High Torque Full Turn Servo (x2)	$18.00	$40.95
3/14/10	High Torque Full Turn Servo (x6)	$18.00	$123.95
3/17/10	Real Time Clock Module	$19.95	$24.35
3/25/10	High Torque Full Turn Servo (x2)	$18.00	$43.95
3/25/10	Mini Push Button Switch	$0.35	$0.35
3/25/10	Momentary Push Button Switch	$0.50	$0.50
3/25/10	Arduino ProtoShield Kit	$16.95	$24.67
4/08/10	High Torque Full Turn Servo (x1)	$18.00	$22.95
4/10/10	ABS Rod, 7/8” Diameter, Black	$6.04	$10.29
4/10/10	Polycase DC-46P (x2)	$7.25	$33.45
4/14/10	Phillips Screw, 0-80 Thread, 1/8”	$8.41	$8.41
4/14/10	Slotted Screw, 0-80 Thread, 1/8”	$7.55	$11.80
4/15/10	Momentary Push Button Switch (x2)	$0.50	$5.41
4/22/10	Break Away Female Headers	$1.50	$1.50
4/22/10	Break Away Headers – Straight (x2)	$2.50	$23.16
		TOTAL	$413.64