Digital Braille Watch

Client: Colton and Holly Albrecht

Advisor: Professor Walter Block

Team members:
Team leader – Joel Gaston
BSAC – Ryan Kimmel
BWIG – Robert Bjerregaard
Communications – Allison McArton

Date: December 11, 2008

Table of Contents

Abstract

Currently the only portable devices that allow for the visually impaired to tell time are audible watches that "speak" the time and an analog watches that have different dots at the 3, 6, 9, and 12 positions in which the individual uses touch discrimination to determine the time. While these methods work, audible watches can be disruptive and tactile analog watches can easily be misread. Digital Braille clocks that exhibit Braille numerals do exist, yet these are not portable and are meant to stay at a stationary location. Last semester, a Biomedical Engineering design team undertook a project to create a digital Braille watch for a blind client. Utilizing four vibro-motors, the team created a device that displayed the time in Braille using a computer and a personal measurement device. The goal for this project is to improve upon a digital Braille design and compact it into a small, portable system that can be read with minimal error. The device prototype now accurately displays the Braille representation of time through vibro-motors, and is completely contained in a compact design. An ATtiny24 AVR microcontroller controls the vibration motor output through the use of a custom program and circuit board. The vibration-motors themselves are mounted on viscoelastic foam at a distance of 3 cm apart, allowing for full two-point discrimination. Future work for this project includes improving the watch enclosure for everyday use, as well as additional functionality for displaying the date.

Problem Statement

The visually impaired currently rely on watches that audibly announce the time or tactile analog watches that must be carefully examined to tell time. These options can be disruptive, fragile, or prone or misreading. Our goal is to create a digital Braille watch that employs 24-hour notation to tell time, does not cause disruptions, and is robust enough to have a long service

life. The Braille display must be sized so that the user can accurately and reliably distinguish the different digits displayed.

Background

Problem Motivation

Colton Albrecht, who is visually impaired, currently has a watch that audibly announces the time. Since he is in school, he has noticed that his watch can be a distraction to the class. Thus, Colton has requested a watch that is not audible and that he can use without any outside assistance. With this device, Colton can gain the independence he wants while at school and around the house. This device would benefit all children who are visually impaired that have trouble with reading analog Braille watches.

Braille

As the client is visually impaired, he is very practiced at reading the Braille alphabet. Ideally the design will display the time using Braille representations of the digits zero through nine. A shown by figure 1, Braille numbers only use the top four units of a Braille cell, as opposed to the traditional six units.

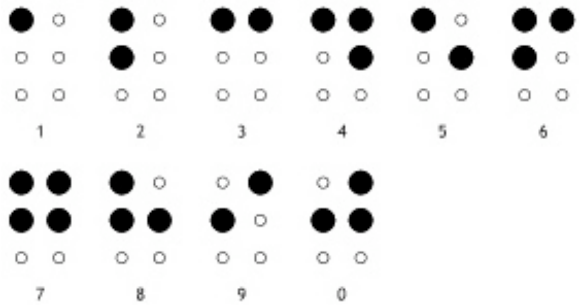

Figure 1: The Braille representation of digits zero through nine

http://www.pharmabraille.co.uk/images/braille-alphabet/braille-alphabet-numbers.gif

Competing/current products

There are a number of ways in which a visually impaired individual can determine the current time. The four most common methods include asking another person, reading analog Braille watches, audible watches, and digital Braille clocks. Though each of these methods allows the individual to ascertain the time, all have serious drawbacks.

If a visually impaired person needs to know the time, but does not own a device to aid in telling time, he or she can usually ask another person. This method of telling time may be sufficient in a few cases, but has obvious shortcomings. There are many instances during a normal day in which it would be both inconvenient and disruptive to inquire about the current time. Therefore, this method alone is not commonly used as the only way to establish the current time.

Two popular devices that are already available on the commercial market are the analog Braille watch and the audible watch. Analog Braille watches, shown in figure 2, are often chosen because of the relatively simple design and the relatively inexpensive cost. The design's major flaw becomes evident once the person tries to read the time, however. These watches are easily misread, as one must be able to accurately orientate the two differently sized hands on the watch face that already contains an arrangement of bumps signifying the hour.

Figure 2: An Analog Braille Watch

http://www.geocities.com/Eureka/Concourse/3294/cortblind.jpg

The other popular device is the audible watch. Most audible watches appear very similar to popular digital watches, but contain a button that activates an audio output of the current time.

The advantage of this design is that it is almost impossible to misread the time, because it is read aloud to the user. The disadvantage is that the sound that enables the user to tell time can also be distracting to others, especially in a quiet setting.

The final method for a visually impaired person to tell time is to use a digital Braille clock. The Kentucky Department for the Blind has already designed and built a "Braille Digital Clock Calendar" that displays the current time using six Braille cells – two for hours, two for minutes, and two for seconds. This design has two advantages, but one major disadvantage. The Braille Digital Clock Calendar is easy to read and is not distracting to others but is most definitely a clock, not a watch. The product measures 7x5 inches, and is one 1.5 inches thick. This is definitely not watch size. The parts for the device cost around $600, not including labor, a cost well out of our price range.

Design Constraints

The following requirements must be met throughout the design process:

1. The watch display must utilize traditional Braille numbering.
2. This must be able to fit on the wrist or in the pocket, so it must be small, and all of the components must be internalized.
3. Military time must be used – hours are to be displayed based on from 0 to 24 throughout the day.
4. The dots that are on the watch face must be positioned appropriately so the user can distinguish between them to be able to tell the time.
5. The watch must be quiet – there must not be a system that the watch announces the time, and all components must not make disruptive noise.

6. SAFE – the watch must not cause any harm to the user.
7. Aesthetics – The watch must be comfortable to the user.

Previous Design

Last semester, a BME 301 design team started this project. They came up with several different designs but ultimately decided upon a vibro-motor design. By using four small motors (as shown in Figure 3), the team had the ability to create a Braille representation time by turning on and off the motors in accordance the number desired. For example, the Braille portrayal of the number one is a single dot in the upper left, so to display this number; the upper left motor would vibrate while the others remain off.

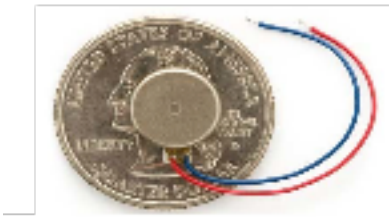

Figure 3: A shaftless vibrating motor that draws small amounts of power in order to vibrate.

http://www.sparkfun.com/commerce/product_info.php?produ cts_id=8449

In order to keep track of time and convert this digital time into a Braille depiction, the previous team developed a program in the language Delphi 5, similar to Pascal (Tang et al., 2008). To depict the computer program output on the vibrating motors, the team used a personal measurement device (PMD). A PMD has the ability to take in either digital or analog signals and create output with varying voltages (2008). They were able to successfully output digits to the vibrating motors using military time as to not confuse AM and PM times. They used two digit outputs for the hour and two for the minutes. In order to distinguish the four numbers, the team also incorporated a one second pause between the different displays. The final output of 7:35am would produce zero, pause, seven pause, three, pause, five through the different vibrations.

Redesign: Isolating Vibration

One of the main concerns with the design was the problem of isolating vibrations. During initial testing, it was found that isolating vibrations within one unit would be very difficult. The vibro-motors that are incorporated into the final design are actually the same as the motors found in many cell phones. One of these motors can cause an entire cell phone to vibrate. The goal for this design is to do the opposite of this – to have one motor vibrating, but isolate the vibrations so the rest of the watch, including the other three motors, is not disturbed. After trying various materials and methods for isolating vibrations, it was found that the best accessible material was a type of viscoelastic foam. The previous team had attempted to use this foam to isolate vibrations, but the problem with their design was that they had the motors too close to each other, so when one motor was activated, all four vibrated. In the current design, the motors are spaced much farther apart, and the user's wrist is actually used as a vibration dampener.

The first attempt at isolating vibrations was to embed two motors in two separate blocks of foam, so that each piece of foam contained only one motor. As you can see in figure 4, these separate blocks were then glued onto a piece of wood. When testing this design, it was found that vibrations from one motor would be transmitted down through the foam, through the wood, up into the other block of foam, and finally into the second motor. This caused the second motor to vibrate equally as much as the first motor, even though it was not activated. It was concluded that this

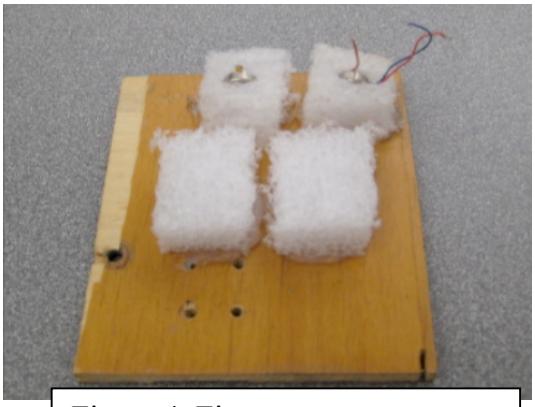

Figure 4: First attempt at isolating vibrations.

design would not work, and that new materials or new positioning of the motors would need to be used.

The second attempt involved two thick pieces of copper wire bent into the shape (Figure 5). A shock absorber commonly seen on tennis racquets was incorporated into the design in an effort to dissipate vibrations. The same problem occurred here. Vibrations were transmitted from one motor to the next. From here it was concluded that the design must not overly constrict the motion of the vibro-motors (embedding them in foam was not going to work), and also that a rigid structure, such as thick copper wire, would not work either. For these reasons it was decided to mount the motors on the outside of a piece of foam, more specifically, on a softer piece of viscoelastic foam.

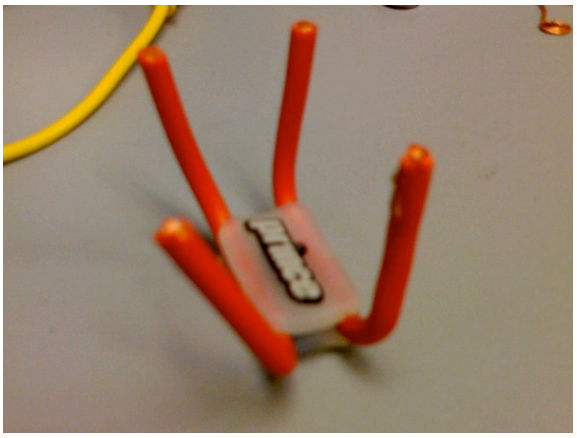

Figure 5: Second attempt at isolating vibrations.

The vibro-motors are on the exterior of the final design (Figure 6) for a couple of reasons. This makes accessing the motors hassle-free, but it also makes isolating the vibrations much easier. If the motors were embedded in the foam, much more of the vibration energy would be transmitted into the foam, so dampening these vibrations would become much more difficult. That is why it was decided that the motors be mounted on the outside of the viscoelastic foam.

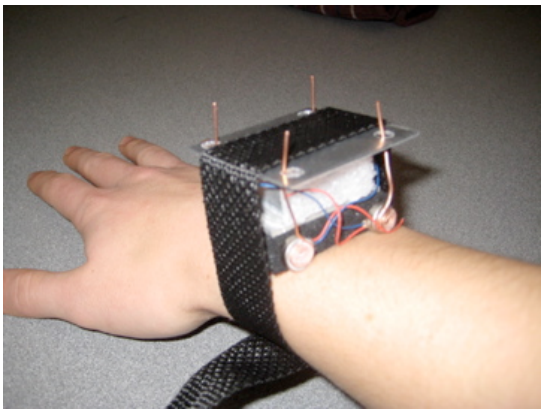

Figure 6: Final design.

Viscoelastic materials are both viscous and elastic when undergoing deformation. When a purely elastic material undergoes deformation, any energy put into the material comes right back out. In other words, no energy is dissipated. Viscoelastic materials dissipate energy when a load is applied, so energy is lost. Since energy cannot be created or destroyed, the energy has to go somewhere, and in the case of viscoelastic materials, the energy goes into molecular rearrangement within the material (Meyers and Chawla). This property of viscoelastic foam is exactly what is needed to isolate vibrations.

Our design incorporates a block of viscoelastic foam to help dampen vibrations, but after some testing it was realized that the foam alone would not be enough to isolate the vibrations. The second way vibrations are isolated is by using the user's wrist as a dampener. When the viscoelastic foam is pressed against the user's skin, much of the energy from the vibrations is transmitted into the user's wrist, and therefore is not able to reach the other motors and disrupt their signals. The final design has a strap that goes over the top of the watch so that the foam is pressed against the user's wrist. This means vibrations from any particular motor are isolated to the copper wire attached to that motor. The user is then able to distinguish which wires are vibrating, and which ones are not.

Unlike the mid-semester design, the final design does not incorporate a clock chip. Instead, a program was downloaded onto the microcontroller to control the time. Some microcontrollers have internal crystal oscillator to keep time, but they are usually on accurate to one tenth of a second. This circuit board has an external crystal oscillator that operates at 32.768 kHz. This allows the microcontroller to keep a time accurate to one thousandth of a second. Also, the clock chip was left out to avoid adding unnecessary parts to the circuit. The battery

would have to power the clock chip, as well as the microcontroller and vibro-motors. Without the clock, the battery will have a longer life, since it is supplying current to fewer devices.

Final Design

Circuitry

The final design uses a relatively simple circuit to connect the microcontroller to the vibration motors, as well as several other components. The microcontroller controlling the entire device is the Atmel ATtiny24 (PDIP/SOIC package), a surface mount 8-bit AVR microcontroller. The ATtiny24 uses 12 programmable I/O lines to control the circuit components, and has two more lines corresponding to voltage and ground inputs. This adds up to a fourteen pin package. The final design only uses 10 of the I/O lines; four for the motors, three for switches, two for the connection of a quartz crystal, and one for an LED located on the circuit board (Figure 7). The entire circuit is powered by a 3V Energizer 123 lithium photo battery, which can output more than 900mA for over 3 seconds when activated.

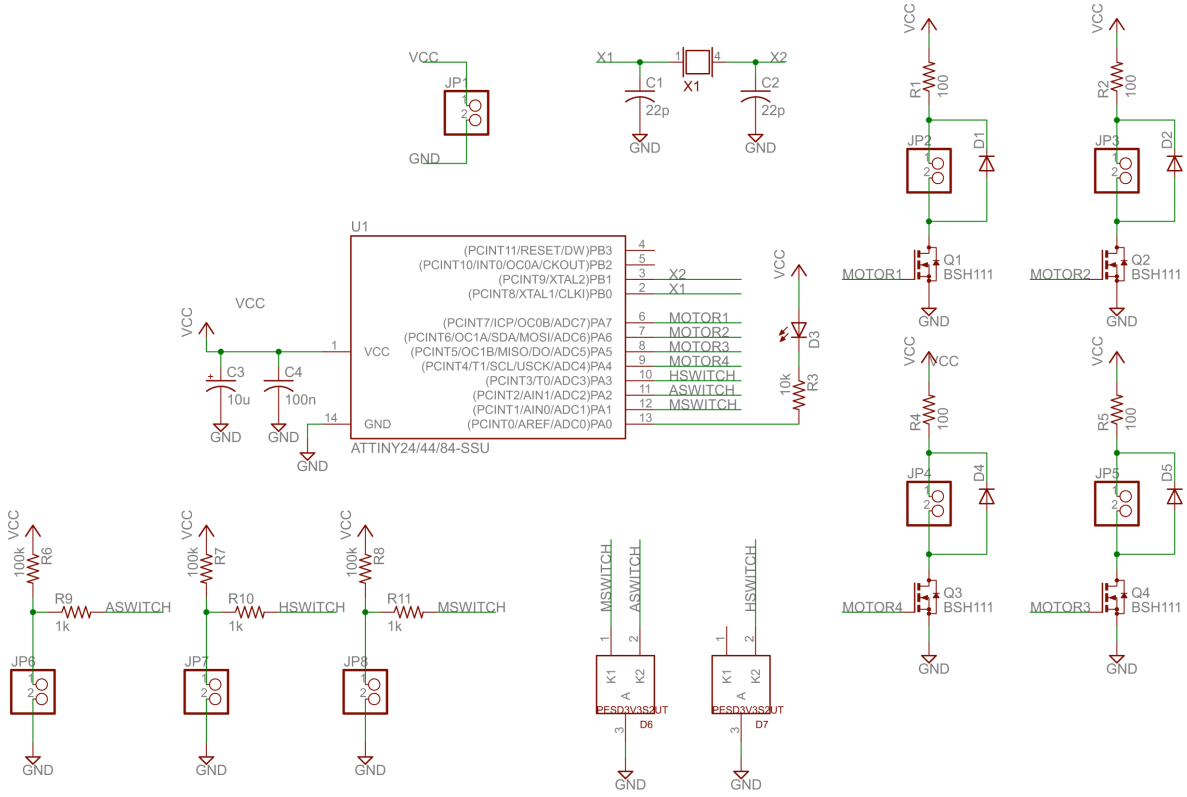

Figure 7: Circuit schematic.

The motor wires, which c tor, are soldered into the holes marked JP2, JP3, JP4, and g identical circuitry, are organized into "motor units". The connected in parallel to a resistor, MOSFET, and diode, all he motor. The resistor (R1 for motor 1 connected at JP2) is a 100 Ω 0805 surface mounted resistor, which is used to drop the current from the battery to 23mA. The resistor is also connected in series with the battery. The MOSFET (Q1 for motor 1) is a BSH111 SOT123 field effect transistor used to ensure that current does not flow into the microcontroller. The MOSFET drain, source, and gate are connected to the motor, ground, and microcontroller, respectively. The diode (D1 for motor 1) is a 100V SOT23 surface mounted diode, and is absolutely crucial for proper motor function. The motor, which basically acts as an inductor, will discharge current once the microcontroller input

is turned off. This diode, D1, will force the current to simply travel around the loop made by itself and the motor, effectively protecting the rest of the circuit from excessive current. This "motor unit" is repeated for each motor.

The microcontroller is also connected to three switches, which provide input to activate certain functions. The first function, activated by one switch, is to tell the microcontroller to display the current time. The other two switches are used to change the stored time in the watch, controlling the hours and minutes. The wires for the switches are soldered into the holes marked JP6, JP7, and JP8. The switches, like the motors, are organized into a basic "switch unit". The switch unit contains a switch connected in series to ground, and in parallel to two resistors (R6 and R9 for the activation switch in Figure 7). One of the resistors, R6, is a 100 kΩ 0805 surface mounted resistor which is also connected to the battery. This resistor serves the purpose of dropping the voltage from the battery to a reasonable level when the switch is activated. The other resistor, R9, is a 1 kΩ 0805 surface mounted resistor connecting the switch, in parallel, to the microcontroller. Finally, a diode (D6, k2 in Figure 7) is placed between R9 and the microcontroller, preventing electrostatic discharge from damaging it. This "switch unit" is repeated for each switch, as shown in Figure 7.

In order to accurately keep track of time, the microcontroller is connected to a 32.768 kHz quartz crystal. This crystal oscillation frequency corresponds to exactly one millisecond, which allows for extremely accurate time keeping. This crystal (X1) is connected in parallel to the microcontroller and two identical capacitors, C1 and C2 (Figure 7). These capacitors allow the crystal to oscillate at the correct frequency, and are absolutely crucial for accurate time tracking. Each capacitor is a 100 pF 0805 surface mount capacitor, and in addition to being connected to the quartz crystal, is also connected in series to ground.

Finally, the battery is connected via wires to the holes located at JP1. Two capacitors are also located in close proximity to the microcontroller, in order to minimize electrostatic discharge. The first capacitor, C3, is a simple 10 µF 0805 surface mount capacitor. The second capacitor, C4, is a 100 nF ceramic capacitor, and is used to absorb most of the electrostatic discharge. Both capacitors are connected in parallel to the microcontroller voltage input, as well as ground.

For the final prototype, the circuit schematic was used to make a printed circuit board (PCB), physically connecting all components. The final board layout is displayed in Figure 8. The actual board size is 1.4 inches long x 1.7 inches high by .062 inches high. The red lines are the traces connecting the components, like wires, and are located on the top of the board.

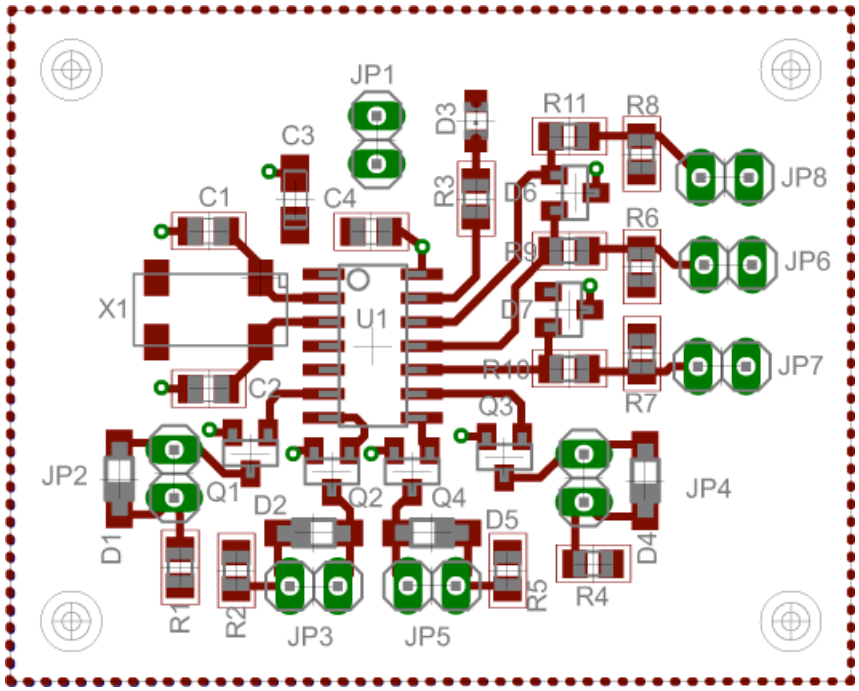

Figure 8: The physical circuit board layout, as viewed from the top. The concentric circles at the corners of the board show where the mounting holes are located.

This layout has several important features, the most important being copper plains. The entire top of the board, as outlined by the dashed red square, is connected to the positive battery terminal at JP1, effectively making it all one node. This allows each circuit component to easily

and efficiently acquire the power it needs. The entire bottom of the board, which cannot be seen in Figure 8, is used as a grounded copper plain and has no circuit components located on it. Once again, this makes connecting individual components to ground very easy, primarily by the use of a via. A via, which is indicated by the small green holes in Figure 8, is simply a physical hole in the board, connecting the top and bottom of the board.

Another important feature of the PCB is the placement of capacitors C3 and C4, which are located near the first pin of the microcontroller (U1 on the board). The close physical proximity of these capacitors greatly minimizes the possibility of electrostatic discharge reaching the microcontroller, and subsequently damaging it.

Programming

The previous team working on this project, created code in Delphi5, most commonly known as Pascal. While this language is very user-friendly and easy to code, it cannot be implemented for use in a microcontroller. A watch programmed with this language could have full time-telling functionality and would be very easy to develop code for it, however it would have to remain connected to a computer, defeating the overall purpose of a watch.

After speaking with two experts in the area (Mike Morrow and Erick Oberstar), it was deemed that an ATMEL microcontroller should be used in this project, which entails using either programming the microcontroller in the C language or coding in assembly language.

C is a high-level language, making it easy to communicate basic thoughts and processes into usable code. Assembly on the other hand, is a very low-level language, only one step above programming in binary(1's and 0's) and proves much more difficult for novice programmers to use. Instructions in this language are very rudimentary and involve hexidecimal notation to jump

from one command or register (memory slot for a variable) to another. After talking with Erick Oberstar, it was confirmed that the code for this project should be written in C.

After the language was decided upon, a compiler and assembler had to be chosen. The compiler and assembler allow the user to write the code on a computer, which will then be translated into a machine language, and written to the FLASH memory on the microcontroller. After researching several options, the C compiler software program CodeVision AVR was chosen. CodeVision AVR is a high-performance C compiler that allows for quick and efficient programming of microcontrollers (AVRFreaks, 2006). It also included a user-friendly graphical user interface for writing code to the STK500 development board. This application is available for use on all CAE computers, free for use by College of Engineering students.

Once the language and assembler/compiler were chosen, the actual programming of the microcontroller commenced. A development platform, needed to send information from the computer and write in on the chip, proved to be an integral part of initial coding. Instead of attempting to write out the code in one large chunk, small steps were taken to ensure basic processes could be successful developed.

Eight small LED's and eight push button switches are incorporated on the STK500 development board. By connecting the pins on the microcontroller to these, one can begin to develop by incorporating basic input/output methods by either turning lights on or off, or by pushing buttons. Due to CodeVision AVR's superior functionality, placing a few simple checks in boxes sets the pins on the microcontroller to be either input or output pins. Once the microcontroller was setup, initial programming to interface the chip began.

There are two separate processes for the program: one that keeps track of time and one that keeps track of whether or not the user-activated button has been pressed, which in turn

controls whether or not the time needs to be displayed. The different processes are referred to as "states" in programming jargon, which references the concept of finite state machines. In this philosophy, a program remains looping in one state, until a certain event (i.e. a button is pushed) and then jumps to another state in which then it can stay in that state or return to the original state (see figure 9).

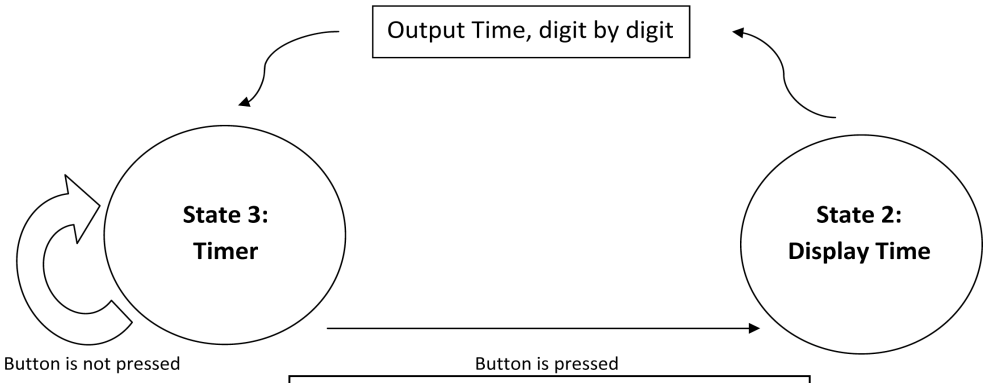

Figure 9: Finite state diagram for programming.

Using the basics of this philosophy, the program loops between two states: Timer and Display time. The function of Timer is simple; it keeps track of time. Display time, on the other hand, has three functions: display the output, turn the pushed button back to false, and return to the original state: Timer. This allows for continual looping, which saves power by eliminating the need to always have motors on.

Using these two programming loops allowed for the coding of one giant "while" loop. The global variables used were sec, H1, H2, M1, M2, and turnedOff. All variables are of the char type, which is an 8 bit variable, to save space on the microcontroller. Sec, keeps track of seconds, H1 keeps track of the hours' tens digit, H2, the hours' one digit, M1, the minutes' tens digit and M2, minutes' ones digit. These digits were set to the digits corresponding to the current time. By splitting the hour and minute digits into two variables, it provided an easy method for

converting the time into four digits and executing the different motor sequences. TurnedOff keeps track of whether or not the button is pushed and was initialized to 1, stating that it is false. Five different microcontroller pins were also used as variables. Four were used as outputs to display the digits and had to be referenced as Port instead of Pin: PortA.4, PortA.5, PortA.6, and PortA.7 to display the digits. For the pushbutton interface, the variable PinA.2 was initially set to false, and is only changed when the button is physically depressed.

In order to minimize code, the entire program is included in the interrupt, so that the program would interrupt every second and then consequently increment the seconds, minutes and hours variables. Two main 'if statements' were embedded in the interrupt section, which tested which state the program was currently in. Since the code is compiled by the machine so quickly, interrupting during displaying the time would not prove to be problematic.

If the button was pushed the variable turnedOff turned to true in which the code then entered an inner-loop, which displayed the time. Output pins 4, 5, 6, and 7 were turned on according to which digit needed to be displayed. Each variable, H1, H2, M1, and M2 was checked to see value it contained and then was displayed one at a time starting with H1 then pausing using the delay_ms function one second then displaying the next digit until all four digits were successfully displayed. The turnedOff button was then reset to true.

After the code was up and running, there were several bugs in the code. All digits were tested to ensure that they displayed correctly. Initially some digits displayed incorrectly so the code was changed accordingly. In addition the time was originally incrementing too quickly. In order to fix this problem, an oscilloscope was used. By hooking it up to an unused output pin, the device showed when the chip was interrupting (incrementing the seconds) and how long these intervals were. By using this method, it was determined that the microcontroller clock chip was

being interfaced incorrectly by the code due to the fact it assumed the clock chip was operating at a different frequency then it actually was. Once the correct frequency values were obtained and integrated into the program, the board was again hooked up to the oscilloscope which determined the program was interrupting at the correct interval: every 1 second.

Testing

The purpose behind testing the device was to develop a setup for proper two-point discrimination. First, a developing platform was made to determine the best separation of the motors (Figure 10).

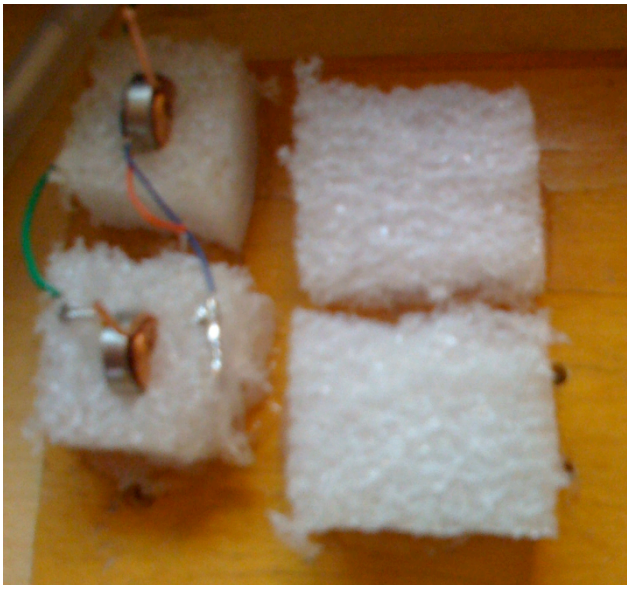

Figure 10: Setup for determining optimal separation between fingers.

With the consent of the client, it was decided upon to place the motors so two would transmit a vibration to the index finger and so two would transmit a vibration to the middle finder. This way, there is less confusion between which motor is vibrating, because there are only two per finger, and they are spaced far apart. The vertical separation between the motors is 3 cm.

Next, the power source needs to supply the proper amount of current to each motor. If the current is too small, it is difficult to feel when the motor is activated. However, if the current is too large, it is nearly impossible to distinguish a vibration in one motor from another. Tests concluded that the optimal amount of current to be sent to each motor should be between 22 and 25 mA (Figure 11). The corresponding voltage is 1 to 1.8 V dropped at each motor.

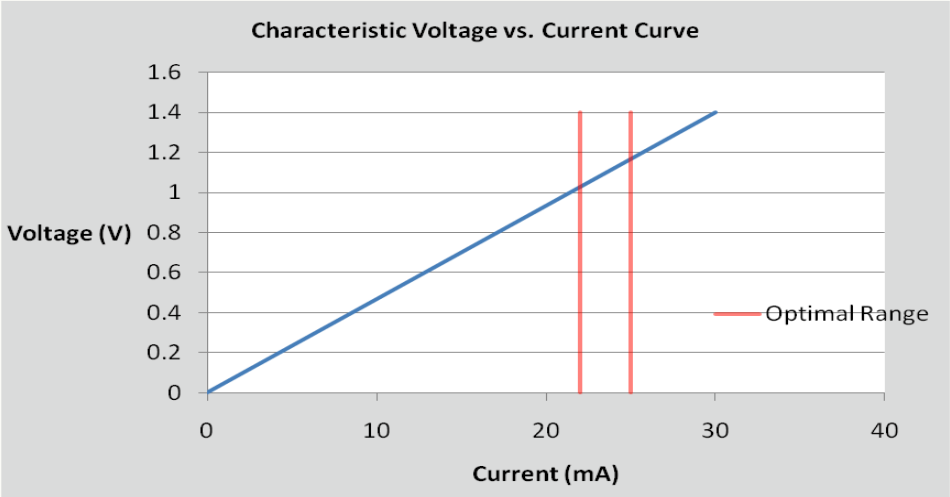

Figure 11: Testing voltage versus current for each individual motor. The "desired range" indicates the amount of current that should be place through the motor that gives an ideal vibration. This also shows the corresponding voltage for the input current.

The microcontroller code was tested with several different people as users. They were instructed which button to push and were also provided with a diagram on how to read the "Braille" displayed by the sample LED's. Although a bit tricky for those not familiar with Braille numbers, the users were able to successfully tell the time. Further calibrations of the pauses between the different digits were made to ensure that optimal time was given between the digits. This was tested by exposing people familiar to reading Braille, to the prototype and then having them recite the time. Those unfamiliar with Braille would have a hard time being able to figure out the time with quicker pauses. Finally the accuracy of the display was checked. After two hours of use, the watch was still precise to the minute.

Future Work

Now that the watch can keep track of time and work as one stand-alone unit, a future team can start adding functions to the watch. One function that needs to be added is the ability to adjust the time. One could do this by adding two buttons, one for hours, one for minutes, which have the sole purpose of setting the time. The time keeping program would need to be modified

slightly to recognize these input functions. With these time-altering buttons on the watch, there would be a possibility of unintentionally changing the time. To avoid accidentally altering the time when these extra buttons are bumped, a future team would have to program the watch so that it is only possible to alter the time after, for example, holding the two buttons down for a few seconds.

Another function that could be added is the ability to display the date. The watch can theoretically display the date using a sequence of numbers. This would need to be added to the time-keeping program, and either use the same prompt button, or add a second button that tells the watch to display the date. If the team decided to use the same button, they could have the watch display four numbers – two for month, two for date – following the four number time sequence. It would be slightly more difficult to integrate a separate button to be used for displaying the date, but this would be more convenient, easier to use, and less confusing for the user.

Finally, if this product is going to be commercially produced and widely used, the housing and overall aesthetics need to be improved. Currently, the motors are on the outside of the watch, which is not the most structurally sound design. Our design is definitely not weather resistant. Currently, the watch is rather large. The height of the watch could certainly be reduced. One way to do this would be to embed the circuit components into the viscoelastic foam. The copper wires could also be shortened slightly to further reduce the height. In addition, a flip cover could be added to keep the protruding copper wires from getting snagged on an article of clothing or other objects.

Appendix A: Product Design Specifications

Product Design Specification for BME 300 Group 33: Digital Braille Watch (As of October 22, 2008)

Group Members: Joel Gaston, Robert Bjerregaard, Alison McArton, Ryan Kimmel

Problem Statement:

The visually impaired currently rely on watches that audibly announce the time, or tactile analog watches that must be carefully examined to tell time. These options can be disruptive, fragile, and prone to misreading. Our goal is to create a digital Braille watch that employs 24-hour notation to tell the time, does not cause disruptions, and is robust enough to have a long service-life. The Braille display must be sized so that the user can accurately and reliably distinguish the different digits displayed.

Client Requirements:

- Digital military time watch
- Watch must fit on the wrist, or in a typical pocket
- Silent system
- Braille numbering must be used to display the time

1. Design Requirements

The device must meet all of the client requirements

a. *Performance requirements:* Audible noises must not be emitted from the watch

b. *Safety:* The watch cannot contain toxic materials, and all wires must be internal

c. *Accuracy and Reliability*: The watch must tell time to an accuracy of one minute

d. *Life in Service*: The watch must have a life-span of several years

e. *Shelf Life*: The watch must have a shelf life of 10 years

f. *Operating Environment*: The device must be able to operate in a dry environment and be able to endure water, humidity, and temperatures between -30 and 50 degrees Celsius

g. *Ergonomics*: The device must be able to be worn or carried as a pocket watch, and should not have any rough edges or loose electrical components

h. *Size*: The watch must fit in a typical pocket

i. *Weight*: The device must weigh at most .2 kilograms

j. *Materials*: The device must be comprised of non-toxic components

k. *Aesthetics*, *Appearance, and Finish*: The watch must be aesthetically pleasing

2. Product Characteristics

a. *Quantity*: Only one working prototype. Could be mass produced based on demand and performance.

b. *Target Product Cost*: Unknown at this time

3. Miscellaneous

a. *Standards and Specifications*: The client has a patent pending

b. *Customer*: The customer would like a device that physically displays the time using Braille digits. Possible functions include a power-saving function and a flip cover

c. *Patient Related Concerns*: None

d. *Competition*: Watches for the visually-impaired are commercially available, in both audible and analog Braille varieties

Appendix B: References

Block, W. Personal Interview. 10 Oct. 2008.

"DS1315 Phantom Clock Chip." 21 Oct. 2008.
<http://www.maximic.com/quick_view2.cfm/qv_pk/2692/t/al>

Morrow, M. Personal Interview. 10 Oct. 2008.

Tang, A., Raju, V., Boumeester, A., & Loherentz, A. (2008) "Digital Braille Watch."

Wayne D. Thompson, Kentucky Department for the Blind,
<http://www.blind.ky.gov/Technology/BrailleClock/clock_description.txt> Retrieved 24 Sep. 2008

Codevisionavr-standard. (2006). Message posted to
http://www.avrfreaks.net/index.php?func=viewItem&item_id=146&module=Freaks%20Tools

Gadre, D. (2000). *Programming and customizing the avr microcontroller* (1st ed.) McGraw-Hill Professional Publishing.

Oberstar, Erick (Personal communication. December 1, 2008).

Oberstar, Erick (Personal communication. December 2, 2008).

Oberstar, Erick (Personal communication. December 3, 2008).

8-bit Microcontroller with 2/4/8K Bytes In-System Programmable Flash. Jan. 2008.
<http://www.atmel.com/dyn/resources/prod_documents/doc8006.pdf>

AVR035: Efficient C Coding for AVR. Jan. 2004.
<http://www.atmel.com/dyn/resources/prod_documents/doc1497.pdf>.

AVR055: Using a 32kHz XTAL for run-time calibration of the internal RC. Jul. 2008.
<http://www.atmel.com/dyn/resources/prod_documents/doc8002.pdf>.

Stk500 User Guide. Mar. 2003.
<http://www.atmel.com/dyn/resources/prod_documents/doc1925.pdf>.

Stk505 User Guide. 2005.
<http://support.atmel.no/knowledgebase/avrstudiohelp/mergedProjects/STK505/STK505.htm>.

Appendix C: Code

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.8a Evaluation
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :  Braille Watch
Version :
Date    : 12/3/2008
Author  : Freeware, for evaluation and non-commercial use only
Company :
Comments:


Chip type         : ATmega8515
Program type      : Application
Clock frequency   : 8.000000 MHz
Memory model      : Small
External RAM size   : 0
Data Stack size   : 128
*****************************************************/

#include <mega8515.h>
#include <delay.h>

char sec = 0;
char H1 = 1; //hard code values
char H2 = 2;
char M1 = 4 ;
char M2 = 2;

char turnedOff=1;

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
 PORTB.0= ~PORTB.0;
      sec++;
      turnedOff = PIND.2;
      if(sec==60)
        {
          sec = 0;
           M2++;
         }

        if(M2==10)
        {
          M2=0;
          M1++;
        }

        if (M1 == 6)
        {
           M1 = 0;
           H2 ++;
           }

        if(H2 == 10)
        {
           H2 = 0;
           H1++;
         }

         if (H1 == 3)
```

```
{ H1=0; }

if(!turnedOff)
{
    turnedOff = 1; //set back to true
    PORTA.4 = 1;
   PORTA.5 = 1;
    PORTA.6 = 1;
    PORTA.7 = 1;

   delay_ms(1000);

   if(H1==0)
   {
     PORTA.4=0;
     PORTA.5=0;
     PORTA.7=0;
   }

   else if(H1 == 1)
   { PORTA.6=0;}

   else if (H1== 2)
   {
         PORTA.6=0;
         PORTA.7= 0;
   }

   //HOURS 1'S DIGIT DISPLAY

   delay_ms(1200);
   PORTA.4 = 1;
   PORTA.5 = 1;
    PORTA.6 = 1;
    PORTA.7 = 1;
    delay_ms(500);

   if(H2==0)
   {
    PORTA.4=0;
    PORTA.5=0;
    PORTA.7=0;
   }

   else if(H2 == 1)
   { PORTA.6=0;}

   else if (H2== 2)
   {
    PORTA.6=0;
    PORTA.7= 0;
   }

   else if(H2 == 3)
   {
      PORTA.4 = 0;
      PORTA.6 = 0;
   }

   else if(H2 == 4)
   {
   PORTA.4 = 0;
   PORTA.5 = 0;
   PORTA.6 = 0;
   }

   else if(H2 == 5)
   {
         PORTA.5 = 0;
```

```
      PORTA.6 = 0;
}

else if(H2 == 6)
{
  PORTA.4 = 0;
  PORTA.6 = 0;
  PORTA.7 = 0;
}

else if(H2 == 7)
{
 PORTA.4 = 0;
 PORTA.5 = 0;
 PORTA.6 = 0;
  PORTA.7 = 0;
}
else if(H2 == 8)
{
    PORTA.5 = 0;
    PORTA.6 = 0;
    PORTA.7 = 0;
}

else if(H2 == 9)
{
 PORTA.4 = 0;
 PORTA.7 = 0;
}

//Display Minutes tens digit

delay_ms(1200);
  PORTA.4 = 1;
    PORTA.5 = 1;
    PORTA.6 = 1;
    PORTA.7 = 1;

    delay_ms(800);
    sec++;

if(M1==0)
{
      PORTA.4=0;
      PORTA.5=0;
      PORTA.7=0;
}

else if(M1 == 1)
{ PORTA.6=0;}
else if (M1 == 2)
{
      PORTA.6=0;
      PORTA.7= 0;
}

else if(M1 == 3)
{
      PORTA.4 = 0;
      PORTA.6 = 0;
}

else if(M1 == 4)
{
     PORTA.4 = 0;
     PORTA.5 = 0;
     PORTA.6 = 0;
}
```

```
else if(M1 == 5)
{
    PORTA.5 = 0;
    PORTA.6 = 0;
}

//Display minutes one's digit

delay_ms(1200);
PORTA.4 = 1;   //reset motors to off
PORTA.5 = 1;
PORTA.6 = 1;
PORTA.7 = 1;

delay_ms(1200);

if(M2==0)
{
     PORTA.4=0;
     PORTA.5=0;
     PORTA.7=0;
}

else if(M2 == 1)
{ PORTA.6=0;}

else if (M2== 2)
{
     PORTA.6=0;
     PORTA.7= 0;
}

else if(M2 == 3)
{
     PORTA.4 = 0;
     PORTA.6 = 0;
}

else if(M2 == 4)
{
    PORTA.4 = 0;
   PORTA.5 = 0;
    PORTA.6 = 0;
}

else if(M2 == 5)
{
   PORTA.5 = 0;
   PORTA.6 = 0;
}

else if(M2 == 6)
{
   PORTA.4 = 0;
   PORTA.6 = 0;
   PORTA.7 = 0;
}

else if(M2 == 7)
{
   PORTA.4 = 0;
   PORTA.5 = 0;
   PORTA.6 = 0;
   PORTA.7 = 0;
}

else if(M2 == 8)
{
   PORTA.5 = 0;
```

```
            PORTA.6 = 0;
            PORTA.7 = 0;
        }

        else if(M2 == 9)
        {
            PORTA.4 = 0;
           PORTA.7 = 0;
        }

      delay_ms(1200);
        PORTA.4 = 1;   //reset motors to off
        PORTA.5 = 1;
        PORTA.6 = 1;
        PORTA.7 = 1;
      }
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=T State2=T State1=T State0=0
PORTA=0x00;
DDRA=0xF1;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Port E initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 62.500 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
```

```
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x03;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3d;
OCR1AL=0x01;
OCR1BH=0x00;
OCR1BL=0x00;


// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
EMCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

// Global enable interrupts
#asm("sei")

while (1)
    {
    // Place your code here
    };
}
```