I. BACKGROUND OF THE BRAILLE SYSTEM

The Purpose of the System

The Palm Braille system was originally designed by Scott Stoffel for people, who like himself, are blind, deaf and have sensitivity and motor disabilities. Sensitivity disabled people do not have a good sense of touch so they cannot tell what an object is by feeling it. Motor disabled people cannot signal with their hands at a normal pace. People who are blind and deaf, and also have these two disabilities cannot use the regular Braille system since they are written very small and there is little space between each translation. Scott took care of this problem by designing a box using electromagnetic solenoids that stick up in the form of Braille representations.

The Original Palm Braille System

Figure 1 - The Original Braille System

The Original system (shown in figure 1) consists of a large Braille character cell and two software programs. The cell is made up of a rectangular box with six solenoids that stick out of small circular holes on the box's surface, forming the Braille representations of the keys that are struck on the keyboard. Each solenoid consists of a coil wrapped around a metal rod. With flow of current, the coil produces an electromagnetic field around the rod, drawing it up or down as needed. The data source of the cell is a parallel port connected into the back of a computer.

The Palm Braille Keyboard Translator and the Palm Braille reader make up the software package. The translator translates a message typed on a standard keyboard to its Braille representation. The reader reads text files and translates them into Braille one character at a time so that the user can read it.

The problems with this system are:

i. the dangerous amount of current the solenoids produce.
ii. the amount of hardware in the system which causes the system to be too large to be easily moved around.

The Previous Improvements

Last year, team EE-6 improved Scott Stoffel's system by replacing the solenoids with servos to reduce current, and by creating a wireless model to make the system more mobile. The system(shown in figure 2) consists of a keyboard, a transmission device, and a palm Braille box. The keyboard uses a transmitter, which consists of a device called the PAK VI, to send information through radio frequency technology to a receiver contained in the box. The palm Braille box contains the servos, a chip called the Basic Stamp, a receiver and a device called the Mini SSC II.

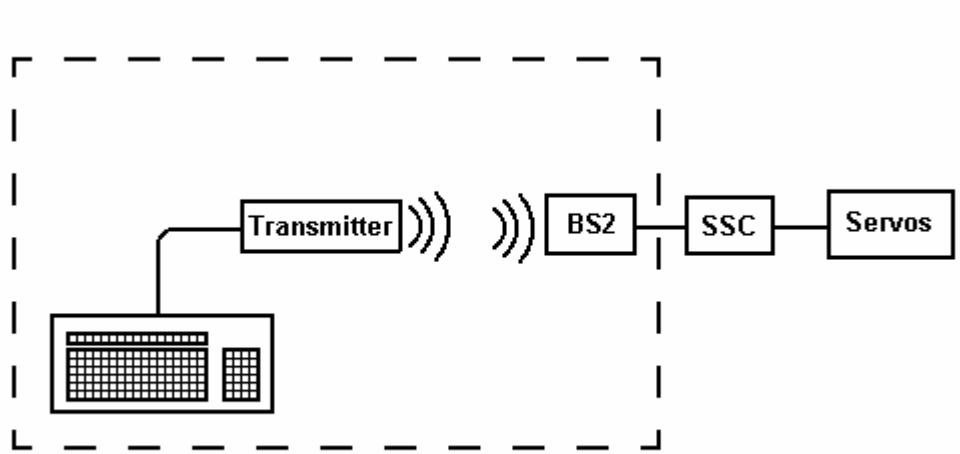


Figure 2 – Last year's modifications.

The Basic Stamp II: The Basic stamp II is a chip that is only able to run a computer program written in PBASIC (a simpler version of BASIC). This program takes the characters of the keyboard and translates it to their Braille representations. It is then inputted to the serial input port of the Mini SSC II.

Transmitter/Receiver: To make the system wireless, a transmitter was connected to the keyboard, and the receiver to the basic stamp in the Braille box. Information from the

keyboard is sent through the transmitter to the receiver, which passes the data to basic stamp.

The Mini SSC II and Servos: Since we used these devices in our system, we will discuss them in detail when we discuss our system.

II. Design Objective

This year, we decided to integerate the previous two systems, taking advanatge of the versatility of Scott's system and the mobility of the wireless system. We proposed to do this by replacing the desktop computer used in Scott's system with a PDA. The PDA has all the features of the desktop computer that are necessary for the system, and due to the PDA's small size, it has the mobility of the wireless system.

For the Braille box, we decided to use a similar design as the wireless system. We chose to use a mini SSC II and servos, however, the microprocessor from the PDA eliminated the need for the basic stamp. The system was not intended to be wireless, so it required a serial port connection from the PDA to the Braille box. This system (as shown in figure 3) would enable the user to send data from the PDA to the Braille box. This data would be read by the mini SSC, which would in turn activate the servos to move the pins to form the desired Braille character.

The PDA had to be programmed to transmit data to the mini SSC. We chose to use Y-Basic, a software designed by HotPaw Basic Inc, to program the PDA. There are many different kinds of PDAs we could have used, including Blackberry, Ipaq, WinCe and Palm, but we chose to use Palm for the following reasons:

i. Palm is more universal than other PDAs.

"Palm currently has 78% market share of the world market for PDA's. Does this mean that Microsoft Windows CE owns a 22% market share in the US? NO – Psion, Symbia, and Apple all share that remaining 22% along with Microsoft."

"Many current handset manufacturers, including Nokia and Qualcom, license the PALM OS". (Epinions.com – epinions.com)[1]

ii. Palm is cheaper than WinCe, its closest competitor.

As of October 2000, the most expensive versions of Palm – the Palm VII and Palm VIIx, which were already wirelessly enabled cost $399 and $499 respectively. The WinCe equivalent of those models at that time cost $549 - $599, plus an additional $200 for a modem for a device that enables the WinCe to be wireless.

iii. Palm has more attainable Software Development Kits (SDK's).

While searching for programming options for the PDAs, we were able to come up with many different tools that were applicable only to the Palm platform. These sources include the Palm OS SDK, the RoboPilot software by Taygeta, the Simplicity by Data Represenations, CodeWarrior lite for the Palm by Metrowerks and PRC-Tools for linux among many others.

After deciding on the Palm as our PDA, we chose the m500 version out of the various models available because:

- Its price is reasonable considering all its other attractive features.

- It is a recent model, so it will not be outdated easily.
- Its battery is rechargeable and it comes with a charger.
- Its memory is expandable.
- It is compact in size.
- It has an on-screen keyboard.

Figure 3 – The PDA Tactical Braille System

III. Design Process

The development of our system involved the interfacing of various hardware components. Two Braille boxes had to be built – one as a prototype and the other for our main system. Each Braille box contains the mini SSC II, which is connected to the PDA using an RS-232 serial cable. This makes it possible for data to be transmitted from the PDA to the mini SSC II. In general, the system consists of three main parts – the Braille box, the serial cable and the PDA. These, in turn, consist of several components, which are described below.

The Braille Box

Construction

The Braille box consists of 6 servos, 4 pieces of aluminum L-brackets (shown in figure 4), a mini SSC, 6 cotter pins, and a voltage regulator circuit. In constructing this box, we first ordered all the parts mentioned above and some screws, and then began putting them together.

Figure 4: L- Brackets

First we cut the L-brackets for each box. We made two of them 5 inches long, and the other two 6 inches long. The bottom of each servo was to be

placed on the longer brackets, and the top on the shorter brackets. We then drilled 3 pairs of holes in each bracket, i/8 of an inch in diameter. The pairs are 7/8 of an inch apart, and the holes in each pair are 7/16 of an inch apart. We also drilled two bigger holes (about twice the size of the others) ¼ of an inch away from the ends of the bracket.

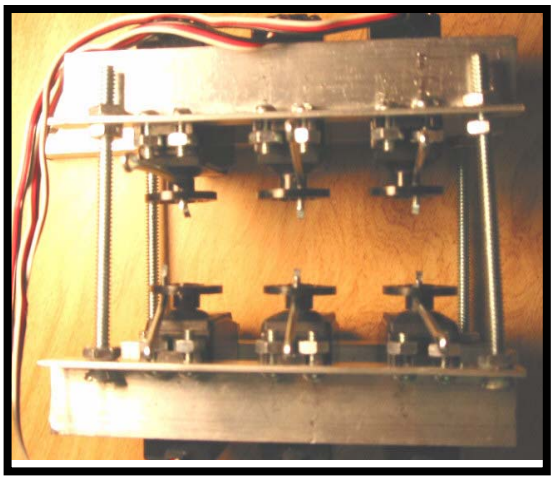


Figure 5: Completed servo mounting

We then proceeded to mount the servos. The top and bottom of each servo had two holes, which were the same distance apart as the holes in each pair on the brackets. Aligned the servo holes with the smaller bracket holes – 3 servos on each pair of brackets – and screwed the servos in. when all servos were mounted, we connected the four brackets together through the bigger holes, using 3-inch machine screws. This way, each three servos faced the other three with 2.5 inches between them, as seen in figure 5.

After mounting the servos on the brackets, we connected the mini SSC to the servos by placing the servo connectors on the mini SSC on the servo header, with the back wire facing the servos number. We then had to build a 5-volt voltage regulator, which would enable us supply 5-volts to the servos and 9volts to the mini SSC, using one 9-volt battery. The design of this circuit is explained in detail later on in this report.

The Servos

Figure 6: One Servo

The servos consist of three wires, which function as power, ground and input signal from the Mini SSC II. The Mini SSC II sends a pulse width of 1ms or 2ms to the servos, which causes them to rotate to a rise or fall position respectively. We attached pins to the servos and the angular motion of the servos will cause the pins to go up and down in the box. The pins would thus represent the Braille alphabet. These servos draw 95mA of current each to operate

The mini SSC

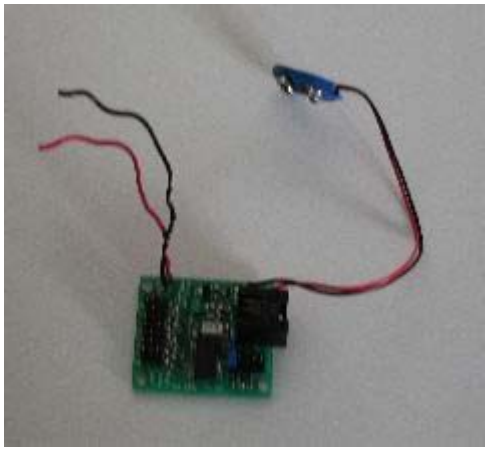


Figure 7 : The mini SSC

Mini SSC II stands for mini Serial Servo Controller version 2. This device receives data in a 3-byte format. The first byte is used for synchronization and has to be 255 base 10. The second byte tells the device which servo to rotate, and the third bytes tells the position the servos should move to. To send instructions to the Mini SSC, we have a simple format consisting of a sync byte (always ASCII 255), the servo number (0-254), and relative position (0-254, where 127 is centered). The SSC sends the appropriate three bytes (*unsigned chars* in C parlance) and send the specified servo control pulses that make it move to the commanded position. Servos are held in the last commanded position until instructed otherwise. For example if someone wants to send the Braille representation of 'A', to the Braille box, the SSC would receive a 255, 1, 254 base 10. The Mini SSC takes serial data 8 bits at a time, and outputs 8 servo control signals. The mini SSC has 2 power supply ends seen in figure 7 above. One of them is a 9-volt battery stacker, to power the SSC and the other is to be connected to the 5-volt servo power supply. We used a 9Volt battery and four AA (1.5V) batteries to power the mini SSC and servos respectively.

The RS-232 Serial Port

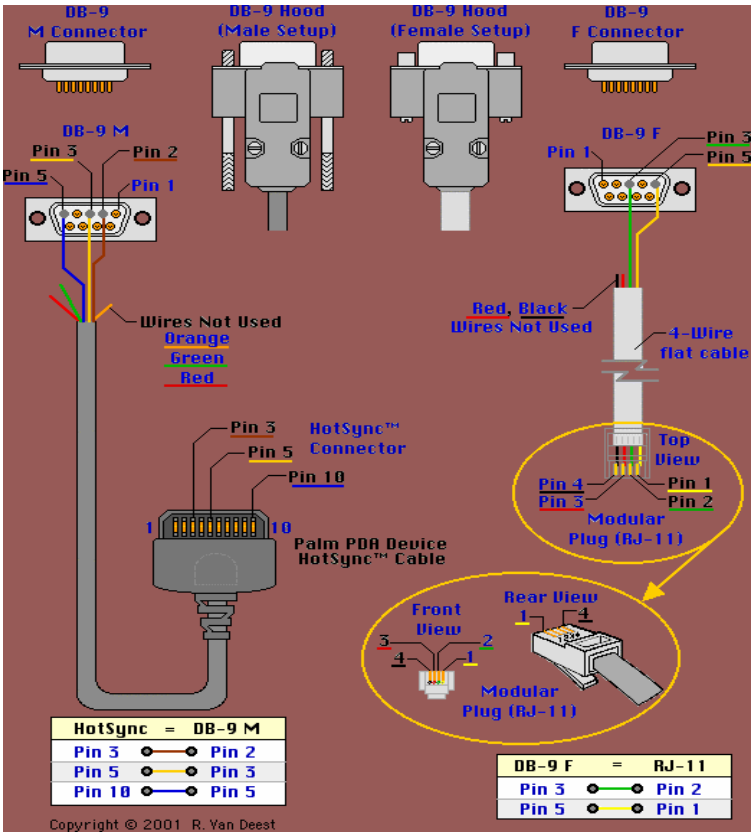


Figure 8 : Showing the connection pins of the RS-232 Serial Port
(Roger's Gadgets and gizmos – bpesolutions.com)[2]

The RS-232 (Registered Standard 232) is a set of protocols created for serial port connections (normally used to transfer data from computers to receiving devices). We used the RS-232 port to transfer information from the Palm to the Mini SSC II. Figure 13b shows the pin configurations of the serial port. The connection of the RS-232 to other components of the system will be discussed later in this report.

The PDA

Figure 9: Some of the features of the PDA

Features

Palm has various useful features, as shown in figure 9 above, but the ones explained below are those that are useful to the system.

Microprocessor: the PDA had a microprocessor which is just like that of any computer, but slightly smaller it capacity. This feature enables us to program the PDA directly, without using the basic stamp or any other data processor. This is advantageous because it reduces the contents and complexity of the Braille box.

HotSync: Palm version m500 has a "hotSync" feature that enables the user to directly transmit data from the desktop to the PDA and vice versa easily. This is a very useful feature because the palm had to be programmed to communicate with the mini SSC, and it was much easier to do this program on a desktop computer, and then transfer it to the

palm without any complications. Also, users of the system could do all their work on desktop computers and transfer them to the PDA using this feature.

Keyboard: Palm has a graffiti area, a built-in map (as shown in figure 10 below), and a stylus that enable users create and transmit data. Users could learn the representations of each character using the character map (these representations look almost like the characters themselves, making them easy to learn) and just write the messages on the screen using the stylus. Palm also has an onscreen keyboard that does not require the use of a graffiti map. We were able to program the palm to translate the standard keyboard characters to their ASCII equivalents.

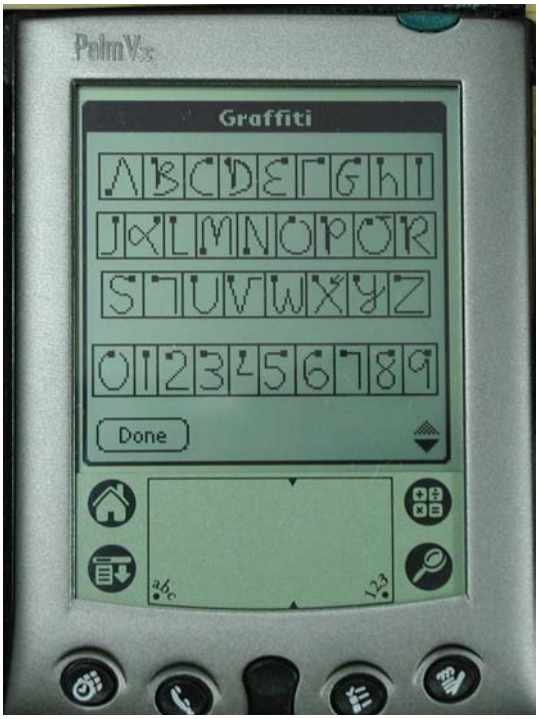


Figure 10: showing the graffiti area of the PDA

Power supply: Palm m500 comes with a rechargeable battery and a cradle charger (as shown in figure 11 below). This is convenient and economical because it completely eliminates the need for batteries for the PDA.

Figure 11: the cradle charger

The expansion card:

The palm has a slot that allows the user insert an expansion card (as shown in figure12) which functions like a floppy disk in a computer. This feature makes the system more versatile in functionality.

Figure 12: showing the expansion card slot

The overall hardware connection

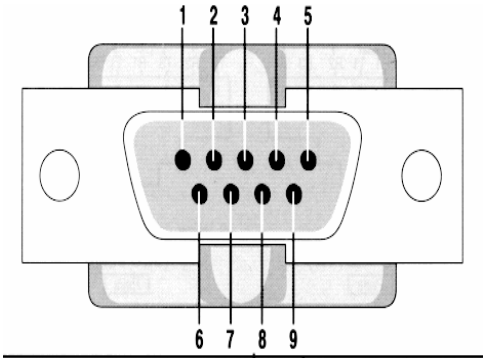
So far, we have discussed each hardware component and it's constituents. In order to complete the system, we had to put all these components together. We connected the mini SSC to the RS-232 serial port, and the serial port to the palm.

Mini SSC to RS –232 connections

The Mini SSC II input is a modular RJ-11 (phone plug) connection we used an RS-232 Female DB-9 (9 pin) connector . We connected the one side of the phone plug to the mini SSC using the connector, and then cut off the connector on the other side. The RJ-11

plug consists of 4 wires – red, black green and yellow. The red and black wires were not used in our connections. We soldered the yellow wire to pin 5, which is the ground pin of the DB-9 connector, and the green wire to pin 2, which is the data reception pin. Figure 8 above shows the RJ-11 and DB-9 connections, and figure 13b shows the pin configurations of the DB-9.

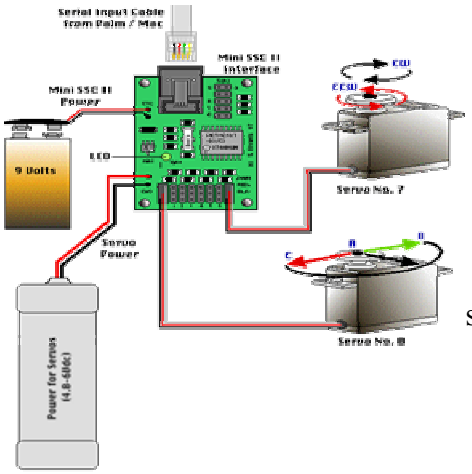
Figure 13a: DB-9 connector



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figure 13b: pin configurations of a DB-9 connector

Mini SSC to Servo connections



There are 8 servo ports on the mini SSC board and our system consists of 6 servos. We plugged each servo into one SSC port, and left two ports unused, as shown in figure 15.

Figure 14: mini SSC to servo connections
(Roger's Gadgets and gizmos – bpesolutions.com)[2]

PDA to mini SSC connection using the RS-232 serial port

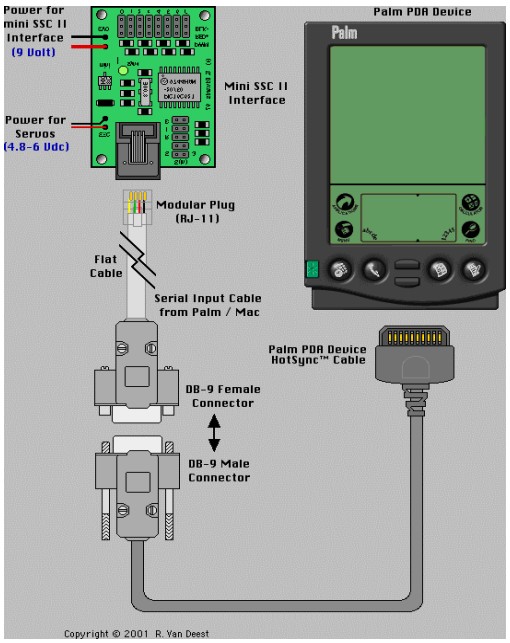


Figure 15: PDA to mini SSC connection

We had to purchase a Palm m500 compatible cable, in order to connect the mini SSC to the PDA because the cradle is too bulky to manage. The one end of the cable is a DB-9 male connector and the other is a Palm m500 connector (as seen in figure 15 above). We connected the male connector to the DB-9 Female connector that was connected to the mini SSC, and plugged in the m500 connector to the serial port of the PDA.

Figure 16 below shows the all the components of the systems, not packaged.

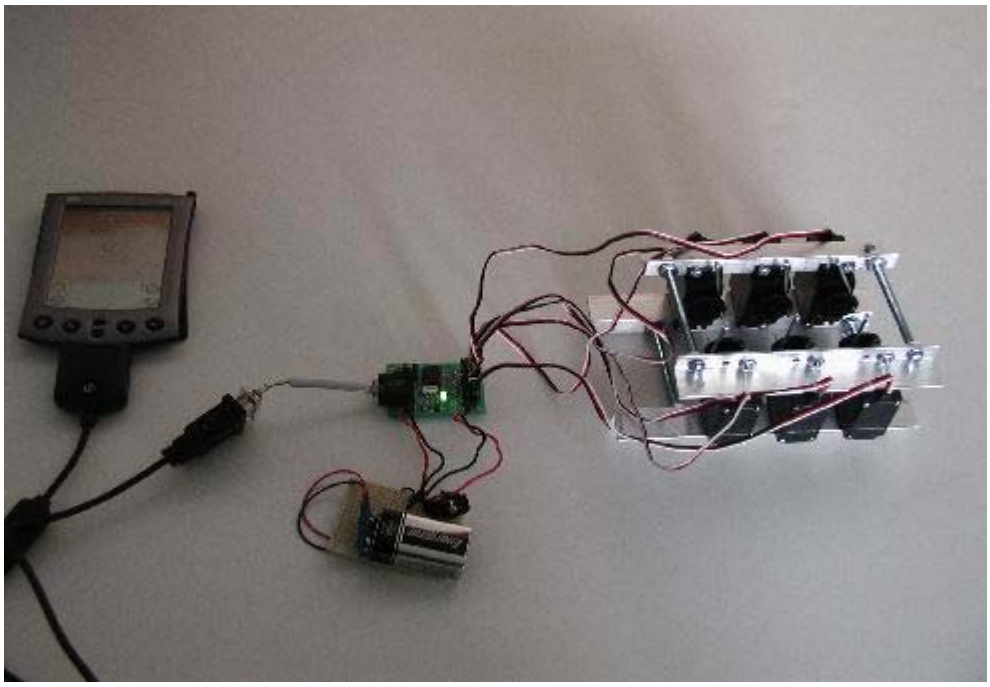


Figure 16: Complete components of the system

With the hardware components all connected, we were able to move on to our program code.

The Software

Software Used

We used a Palm platform compatible software called HotPaw Basic, which was developed and maintained by HotPaw Inc. We chose HotPaw Basic for the following reasons:

- Ability to develop on the palm pilot itself
- Reasonably Priced – cost $20
- Less complexity - no need for language interpretation

The software used a program called YBASIC, derived from the BASIC program.

The Program Outline:

The outline of the program was as follows:

- Create form elements
- Check to see if filename
- Check to see if number
- Check to see if capital letter
- Get servo numbers for current letter
- Send data to the SSC

How the program works:

Figure 17: The PDA screen during the communication process

When the program receives input (from a file or directly from the user), it checks the ascii code to find out if the input is a number or character and it's case (lower or upper case) if it's a character, draws the braille representation of the letter on the palm screen (as seen in figure 17 above), and sends the braille representation of the input to the mini SSC . If the input is an upper case letter or a number, the program first draws the upper case or number sign on the screen and sends it to the mini SSC before doing the same for the letter or number representation. The progran then waits one second to get the next letter or end the loop if there's no more input coming in. Figure 18 below shows the braille represenations of all numbers and characters, and below that is an illustration of how the program works. The full program can be found in appendix A of this report.

To Move the letter 'A' the program executes:

```
if((char < 91) and (char > 64))
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
```

```
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
        draw circle 25,95,5
        draw circle 25,110,5
        draw circle 25,125,5
        draw circle 45,95,5
        draw circle 45,110,5
        draw circle 45,125,5,7
        open "com1:",9600 as #5
        print #5,send$;
        sound 1,1,1
        close #5
        fn wait(1)
        char = char + 32
endif
```

which checks if it is a capital letter, redraws the screen and sends the capital letter sign to the mini ssc.

then it executes:

```
if(char = 97)
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
        draw circle 25,95,5,7
        draw circle 25,110,5
        draw circle 25,125,5
        draw circle 45,95,5
        draw circle 45,110,5
        draw circle 45,125,5
endif
```

which moves the servos that represent 'a'

fn wait (1)

next i

wend

end

this program segment waits 1 second, gets the next letter and ends the loop.

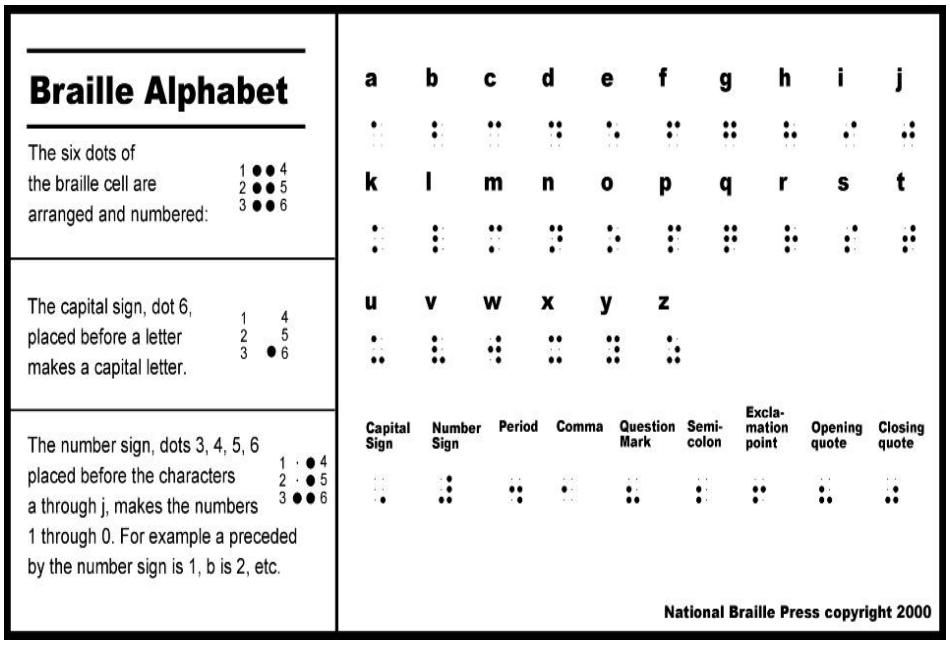


Figure 18: Braille Alphabet

The different versions of the program

Below we have illustrated the general steps we took in creating software to control the Braille system.

version	date	change
0.0.1	02.16.2003	algorithm for lowercase alphabet
0.0.2	02.18.2003	added algorithm for capital letters, numbers, and punctuation
0.0.3	02.20.2003	fixed bug with open/close quote prints current letter to screen instead of pop-up menu
0.1	02.22.2003	fixed letters s – z
0.1.1	03.22.2003	created algorithm to read strings from the command line
0.2	03.24.2003	created picture representation of which servos are high/ low made command line bigger to approx. 25 chars fixed close quote problem - tested 03.24.2003
0.3	03.30.2003	reads text from a file stored in memopad blanks out input line after sending a line of text to the braille box changed pos and poslo variables for larger range of motion.
0.3.1	04.10.2003	added splash screen and about box
1.0	04.16.2003	launches and exits to the palm menu registered creator id 'ee-6'

IV. POWER SPECIFICATIONS

The 9-volt battery supplies 2.8 amps for 1 hour. The mini SSC requires 10mA to operate. We used this information to calculate the amount of time the SSC could run for using the 9v battery.

(2.8 amps / .010 amps) * 1 hour = 280 hours of operation before battery is drained

The 4AA batteries supply 2.8 amps for 1 hour at 6-volts. Each of the six servos can use up to a maximum of 95mA to operate. We used this information to calculate the amount of time the servos could run for using the 6-volt battery supply.

(2.8 amps / (6 * .095 amps) * 1 hour = 4.9 hours of operation before are drained.

Theoretically, if the system draws max current constantly, the system would run for 280 hours before the 9-volt battery need to be replaced and 4.9 hours before the 4AA batteries need to be replaced.

V. HEALTH AND SAFETY CONSIDERATIONS

This project will require health and safety considerations since it is geared towards disabled people who are not necessarily engineers, and a significant amount of current flows inside the Palm Braille box with which the user will be in constant physical contact. The following precautions are mandatory when using the system:

1. The Palm Braille box may not be used near water.

VI. ETHICAL CONSIDERATIONS

Electrical Engineers have various ethical considerations to worry about when working projects, especially one, like ours, which is mainly geared towards the handicapped. We have to be careful to make sure the device is purposeful and not at all harmful. We are basically going to follow the ethical codes of the IEEE, which we have enclosed in Appendix B.

VII. BUDGET

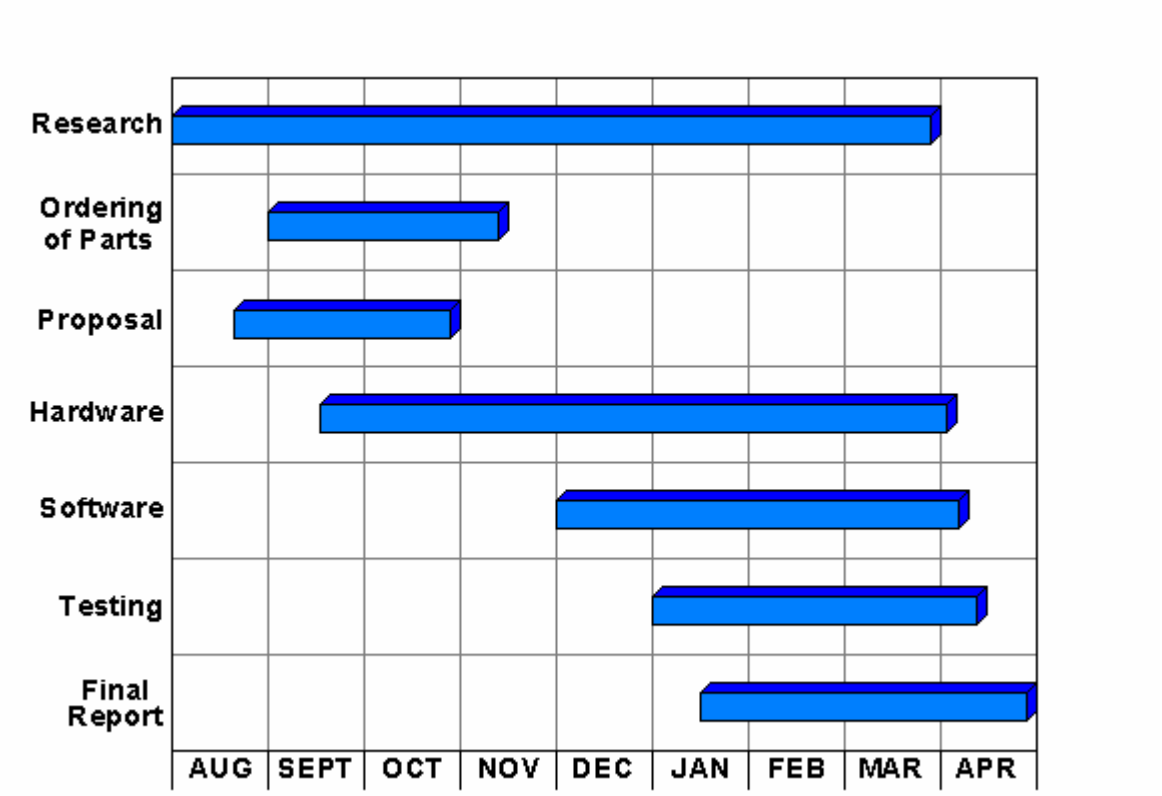
Item	Qty	Price	Total Price
Mini SSC II	2	$49	$98
Futaba Servos	12	$19	$228
Palm PDA(m500)	1	$200	$200
Plastic Casing	2	$8	$16
36” Aluminum Brackets	2	$2	$4
Cotter Pins(8 packs)	2	$0.40	$0.80
Palm Hotsync serial cable	1	$20	$20
Screws(pack 10)	10	$0.86	$8.60
Hot Paw Basic Program	1	$20	$20
Serial Ports	2	$2	$4
Batteries (9V, 4 AA)		$6	$6

Total Price $605.40

Sources:

- Temple University College of Engineering Senior Design Fund
- Tacticom Corporation

VIII. TIMELINE



We began research on this project in early August to get an idea of what needed to be done. We continued to research all the way through until the project was completed in April. Based on our research we ordered the necessary parts and shortly thereafter began to build the Braille boxes. Completing the hardware took one month longer than we anticipated due to finding a suitable power supply solution for the project. The software, testing and final report were completed on schedule.

IX. SUMMARY

In conclusion to this report, here are the basic changes we have made to the previously designed systems, and our systems advantages over others:

Changes	Advantages
• Replacement of the Keyboard with the stylus and graffiti area and the built in keyboard of the PDA	• Eliminates the keyboard – makes the system much more portable
• Replacement of the Basic Stamp with the microprocessor of the PDA	• Eliminates the Basic stamp, makes the Braille box lighter and simpler. • Enables users to add more programs to the system than just the character translation code.

The Palm Braille system is a very essential device for people who have vision, sensitivity and motor sensory disabilities. Remodeling it into the PDA Palm Braille System has made it more desirable and attainable because, as demonstrated above, the system will be much smaller and lighter, and, therefore, easier to carry about. Also the system is now more versatile and therefore functional to the user. PDAs are very "present day", so, apart form it's functionality, our system also gives a feeling of modern technology to its users.

X. REFERENCES

1. Epinions.com – epinions.com

2. Roger's Gadgets and gizmos – bpesolutions.com

XI. BIBLIOGRAPHY

Kaitell, Chris and others. Wireless Palm Braille System, Temple University, Philadelphia, Pennsylvania May, 2002.

Myers, Brad A., Kin Pou Lie., and Bo-Chieh Yang. Two Handed Input Using a PDA and a Mouse. CHI Letters. April 6, 2000. 41-48.

Reshko, G. Palm Pilot Robot Kit. November 5, 2001. <http://www-2.cs.cmu.edu/~reshko/PILOT/>

Acroname Easier Robotics. 2002. Acroname Incorporated. <http://www.acroname.com/robotics/>

Roberts Gadgets and Gizmos (projects) Palm PDA and Mini SSC II <http://www.bpesolutions.com/gadgets.ws/gproject2.html#anchor153922>

Scott Edwards Electronics, Inc. Mini SSC II Serial Servo Controller User's Manual 1999.

Parallax Inc. Basic Stamp Program Manual (version 1.9). 1998.

Palm, Inc. Palm m500 Handheld Series User's Manual. 2000.

Daniel, J.; Greenfield, S.; Schmalzel, J. "Serial Analysis with a Palm Organizer". IEEE Instrumentation and Measurement Technology Conference. May 4, 2000. 311-312.

Seyer, Martin D. RS-232 Made Easy:Connecting computers Printers, Terminals, and Modems.(2nd Edition) Englewood Cliffs, NJ: Prentice-Hall, 1991.

XII. APENDIX A – THE PROGRAM CODE

```
#tactical-v0.0.3.bas
draw -1
draw "Tactical Braille System v.1",15,23,1
form btn 75,95,50,20, "Send Data",1
form fld 80,55,40,12, "",1
draw "Letter ",5,57

#fixed bug with open/close quote
#02.20.2003

wake = 255
pos = 254
poslo = -128
quotecount = 0

while
x = asc(input$(1))
char = asc(s$(0))
print at 5, 75
print "Letter = " chr$(char)

if((char < 58) and (char > 47))
print "char < 58) and (char > 47"
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
# goto sent
        fn wait(1)
        char = char + 48
        if char = 96 then print "char = 96 "
endif

if((char < 91) and (char > 64))
print "char < 91) and (char > 64"
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
        fn wait(1)
        char = char + 32
endif

if(char = 97)
print "char = 97 "
```

```
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 98)
print "char = 98 "
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 99)
print "char = 99 "
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 100)
print "char =100"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 101)
print "char =101"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 102)
print "char =102"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif
```

```
if(char = 103)
print "char =103"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 104)
print "char =104"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 105)
print "char =105"
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif


if(char = 106)
print "char =106"
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 107)
print "char =107"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 108)
print "char =108"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
```

```
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 109)
print "char =109"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 110)
print "char =110"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 111)
print "char =111"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 112)
print "char =112"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 113)
print "char =113"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 114)
print "char =114"
```

```
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 115)
print "char =115"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 116)
print "char =116"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 117)
print "char =117"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 118)
print "char =118"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 119)
 "char =119"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif
```

```
if(char = 120)
print "char =120"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 121)
print "char =121"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 122)
print "char =122"
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 46)
print "char =46 "
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
endif

if(char = 44)
print "char =44 "
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 63)
print "char =63 "
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
```

```
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
endif

if(char = 59)
print "char =59 "
        send$ = chr$(wake) + chr$(1) + chr$(posilo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 33)
print "char =33 "
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 34)
print "char =34 "
        if((quotecount mod 2) = 0)
print "quotecount mod 2) = 0"
                send$ = chr$(wake) + chr$(1) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
                send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
                send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
        else
print "else quotecount mod 2) = 1"
                send$ = chr$(wake) + chr$(1) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
                send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
                send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
                send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
        endif
        quotecount = quotecount + 1
endif

if((char = 32) or (char = 45))
print "char =32 of 45"
        send$ = chr$(wake) + chr$(1) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(poslo)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(poslo)
endif

if(char = 43)
```

```
print "char =43 "
        send$ = chr$(wake) + chr$(1) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(2) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(3) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(4) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(5) + chr$(pos)
        send$ = send$ + chr$(wake) + chr$(6) + chr$(pos)
endif

open "com1:",9600 as #5
print "send = " send$
print #5,send$;
#sound 1800,100,64
#sound 1500,30,64
close #5
wend
end


#       0       //not used
# 1     4
# 2     5
# 3     6
# 7             //not used
```

XIII. APENDIX B – IEEE CODE OF ETHICS

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. To accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
3. To be honest and realistic in stating claims or estimates based on available data;
4. To reject bribery in all its forms;
5. To improve the understanding of technology, it appropriate application, and potential consequences;
6. To maintain and improve our technical competence and to undertake technological task for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of other;
8. To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;
9. To avoid injuring others, their property, reputation, or employment by false or

malicious action;

10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.