GlucoMon: A Glucose Monitoring System for the Handspring Visor PDA

John K. Estell, Jeff Haar, Josh Lemke, Jeremy Saunier, Adam Smith
Electrical & Computer Engineering and Computer Science Department
Ohio Northern University

Introduction

Over 17 million individuals in the United States are affected with diabetes. While incurable, diabetes is manageable with proper monitoring. Currently, monitoring is performed through use of stand-alone blood glucose meters that allows a diabetic to monitor blood sugar levels on a periodic basis; the meter readings are either recorded into a log book or uploaded via proprietary software to a desktop computer. The purpose of this senior design project was to offer a new approach to patient self-monitoring through the development of a diabetes management system using the Handspring Visor Personal Digital Assistant (PDA). The design consists of a Springboard module containing blood glucose metering hardware and an accompanying software package that operates the module and allows for the storage and processing of data. The following sections outline the process followed by the senior design group in their development of the design.

Obtaining Information About Current Glucose Meters

The first step to determine how current glucose meters obtain a reading was to look online for documentation. This allowed the group to review several different types of meters; however, it did not lead to any conclusive information about how an actual reading of a glucose level was obtained from a blood sample. After struggling with the search for information, the group turned to contacting the companies directly. Dr. David Kisor, a professor in the College of Pharmacy at Ohio Northern University, offered to help the group contact several different companies in the hope that one of them would be willing to form a partnership. Unfortunately, large drug companies move very slowly and none of the companies was able or willing to provide the design group with the necessary information in time. At this point, it was decided to examine the patent documentation from the several glucose meters the design group had already reviewed. One patent was found which had specifications on how the glucose level was obtained. The patent for the Bayer Glucometer Elite[1] explicitly stated that the glucose reading could be obtained by applying 600 mV to the electrode containing the blood-testing strip, waiting five seconds, and then measuring the current flowing through the circuit. Figure 1 contains the graph of the response current to glucose concentration relationship as taken directly from the patent.

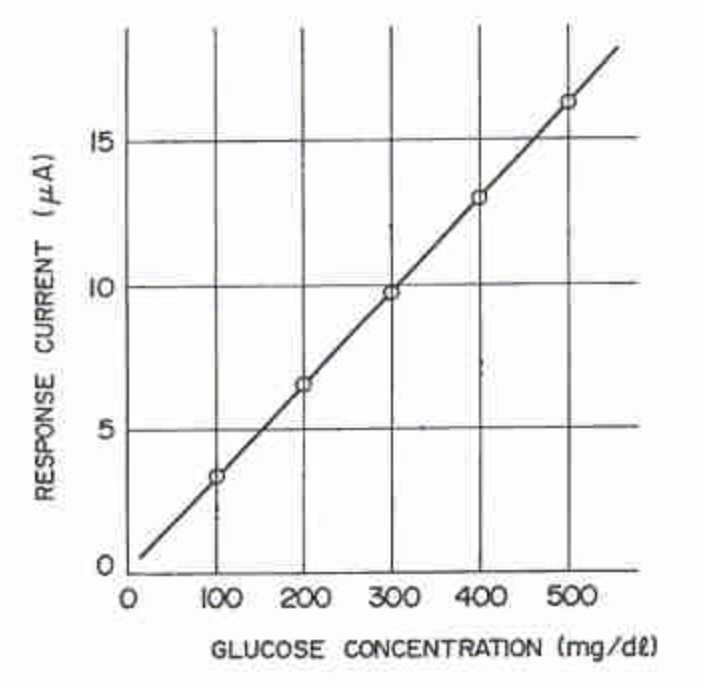

Figure 1: Raw Graph of Glucose-Current Relation from Bayer Glucometer Elite Patent

Designing the Prototype

Figure 1 provided the only detailed information that could be found; thus, it was used to generate a working prototype. It was assumed that the electrode was acting as a resistive element with higher glucose concentrations effecting a lowering of the resistance. A table was created in Excel to calculate the approximate resistances of the electrode based on the patent information. This table was also used to calculate the sensor resistance that could be used inline with the electrode that would not introduce significant error into the reading, yet offer a sufficient voltage drop across it for accurate measurement of the current response. By looking at available 1% resistor values and accepting a tolerance of ±1% of the average resistance (about 1MΩ), the value of 875 Ω was selected. The currents produced by the 600 mV source through the electrode were from 0 to 16 μA, which accounted for a voltage drop of 0 – 14 mV across the sensor resistor. Finally, to center the voltage applied to the electrode at 600 mV, the source voltage was increased to 607 mV, thereby making the final electrode voltage vary from 593 mV to 607 mV. The diagram of this completed circuit can be seen in Figure 2. The output voltage of this circuit is amplified, then sent to a 12-bit analog to digital converter (ADC). By using an amplification factor of 20 and a reference voltage of 1.15 volts, the glucose reading from the patent information in Figure 1 corresponds exactly with the digital value produced by the ADC. In other words, the digital output of the ADC is the glucose level of the blood sample in mg/dL.

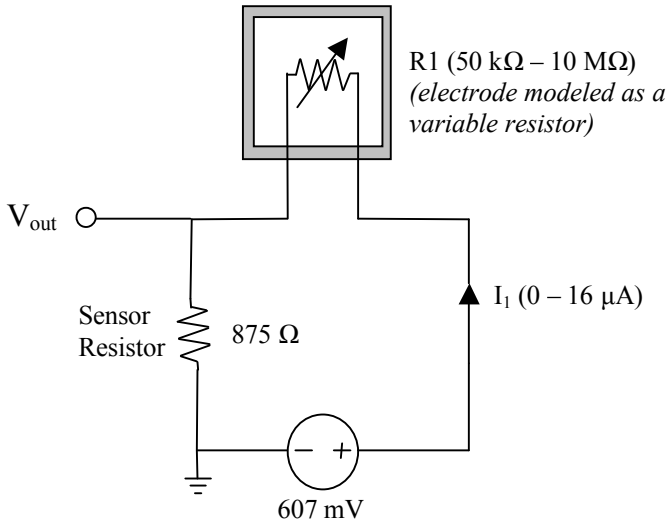

Figure 2: Analog Prototype Circuit

Choosing a Hardware Development Platform

In order to facilitate hardware development, a development board was needed. The development board needed to support at least a simple module creation. In addition, because of the limited ten-week design period, the board needed to be easy to use. The following features were required on the development board: low power CPLD, A/D converter, ROM, and development programs.

The team initially explored the option of building a development board. This option posed many challenges, including fitting the module into the Springboard slot of the Handspring Visor. Another major problem was the difficulty in programming the CPLD on the development board. Even with the ability to physically create and program a module, the design team had to consider the cost of purchasing all the needed components and the time needed to assemble the board. The second option was to find a commercially-available development kit. It was discovered that Insight Electronics offered a development board that fit the needs of the design team. The development board uses the Xilinx CoolRunner CPLD that is designed for low power applications. It also offers a JTAG programming interface to allow simple programming of the Xilinx chip, a 12-bit analog to digital converter, 8 MB Flash ROM, and 4 MB SRAM. Creating a custom development board would have simply duplicated the work that had been done creating the Insight development board; therefore, the Insight development board was purchased. Not only was the design simple, it made the development of the hardware faster because the design team could focus on the glucose meter development, and not the development board creation. There was, however, one major unforeseen drawback to the Insight development board. Insight had some problems with all of their boards, which delayed the delivery of the board by over two months. The delay left only three weeks in the design period to complete the hardware portion of the glucose meter. This setback kept the design in the prototype phase, as there was no additional time to finalize the design. Documentation of the development board[2] was obtained

before the final delivery; however, it was difficult to understand the hardware until it could be tested and implemented. Unfortunately, Insight has since discontinued the Springboard Development Kit due to the phasing out of the Visor product line by Handspring.

Hardware Development

The hardware design was implemented in VHDL and consisted of three major components: the ADC driver, the SRAM driver, and communications with the Handspring. Xilinx provided open source drivers for the ADC and the SRAM in the form of an Oscilloscope program[3]. The Xilinx-provided drivers were used in the development of the glucose meters hardware to obtain the analog reading.

The driver for the ADC had to be modified to meet the specifications of the glucose meter module. Fortunately, the Xilinx documentation provided a detailed explanation for the functionality of the oscilloscope program. Due to the low voltage range coming from the strip reader, a gain of 20 and a reference voltage of 1.15V were used to obtain the required resolution. To communicate with the software in the Handspring Visor, additional VHDL code was written to represent the different states of the hardware: strip insertion, blood application, and data conversion. In the conversion state, the software signals to the hardware to start the A/D conversion by using two different write commands. The first write command is used to reset all information in the ADC and to reset the address pointer in the SRAM. The second write command causes the ADC to start a new conversion. The software waits for 10 seconds to allow the blood sample to stabilize (as specified in the patent information) then performs the analog to digital conversion by making 32 consecutive readings, which are stored into the SRAM. Once the conversion is done, the software application can retrieve the information from the SRAM, average these values, and present the result to the user.

Software Requirements

The functional requirement for this design is a program that will allow patients to monitor their glucose levels using a Handspring Visor. Before development of the software began, the team defined additional functional requirements. First, it was determined that this software must be able to store related sets of data within the Handspring Visor's memory. For each data set, the values to be stored are the date and time of the specific reading, comments about how the patient is feeling, the type of reading (in relation to recent meals), and the glucose reading value. Once stored in the Visor, this data is to be retrieved, reviewed, and displayed in a graphical form.

Non-functional requirements are those requirements that must be met in order to satisfy the user and client in factors such as appearance, ease of use, and speed. The most important non-functional requirement of this software program was ease of use. Many diabetics see taking their glucose reading as a chore; thus, for diabetics to consistently monitor their levels, the testing program must be as easy to use as possible. Along with "ease of use" comes an intuitive GUI that follows the typical Palm OS style of buttons, menus, and other GUI objects. Satisfying these non-functional requirements is just as important as the functional requirements because they help make the program appealing and marketable.

Software Design

With the functional and non-functional requirements defined, the next step was to construct a flow chart of how the software should function. From the flow chart, the program design was organized into four sections: the opening section, the enter data section, the view data section, and the graph section. For each section, a set of screen displays was developed.

First, the opening section gives the user three main choices: enter data into the system, view current data in the system, or exit the program. This section presents the "home screen" that the program returns to whenever the user finishes an operation. When entering data, the initial screen allows for input of comments about how the user is feeling as well as the type of reading (relative to meals) being entered. Once the process of getting a reading has been started, the system waits for more user input such as the glucose strip insertion and blood application. If the process of obtaining the glucose reading is completed without error, the system will then display the data collected and return to the home screen. If there were an error obtaining the reading from the external hardware, the system would alert the user, then allow the user to either attempt another reading or manually input a glucose level. After data has been acquired by the system, the user can select to view the data from the main screen. A typical "view data" screen can be seen in Figure 3. This screen allows users to navigate though the database of glucose readings, allowing for the review and possible deletion of past values.

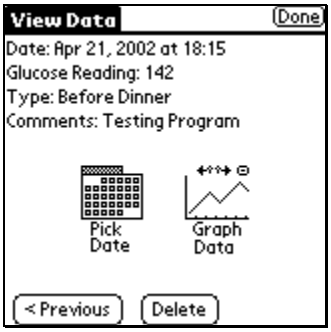

Figure 3: View Data Screen

From this screen, the user can select to either graph or display data. The graph gives the user the ability to see visual trends of their glucose values over a day, a week, or even a month. Figure 4 shows examples of day, week, and month graphs with sample data. The zoom in button (🔍) at the top of the screen allows the user to zoom in from month to week or from week to day. The zoom out button (🔍) allows the user to zoom out from day to week or from week to month.

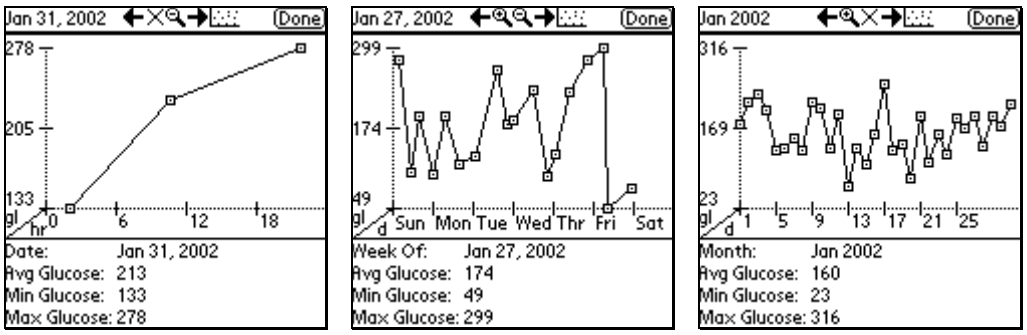

Figure 4: Day, Week, and Month Graph Views

Along with trending and other general statistical information about the day, week, and month, the graph allows users to click on an individual data point and obtain even more information. This additional information includes all information about the point when viewing data on the day and week view, as well as summarized daily values when viewing data on the month view. Samples of this information can be seen in Figure 5 for all three views.

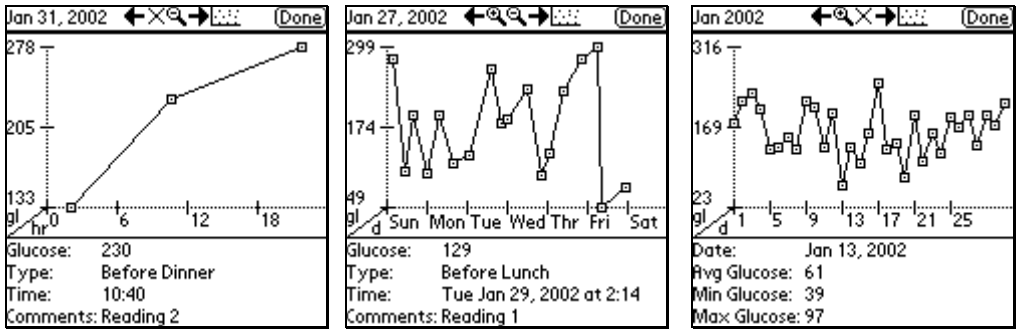

Figure 5: Day, Week, and Month Graphs after Selecting a Single Point

Another feature of the data graphs is the ability to move sequentially through the database using the arrows at the top of the screen. These arrows, seen in the screen shots contained in Figures 4 and 5, are dynamic in that they will only appear when there is additional data available either forward or backwards in time. Finally, the graphing screen includes the ability to not display the lines connecting the data points on the graph; this aids in the speed of the graph drawing and provides another choice for the user. When selected, the dot icon ( ) in the top right of the screen will toggle off the line display. When a graph without the lines is displayed, the dot icon is replaced with a graph icon ( ) to toggle back to a graph with lines.

Software Development

Two programming languages presented a viable option to perform the functionality needed in this design. The first environment evaluated was Metrowerks Code Warrior using C++ as a programming language. Initial development started on this platform. Code Warrior proved to have great functionality; unfortunately, it also had a very steep learning curve. The development of functional code in Code Warrior took too long, and compatibility with the development board was questionable. The second environment evaluated was Pocket C from OrbWorks Concentrated Software[4]. One benefit to this environment was that Pocket C was provided with the Insight development board. Pocket C was relatively easy to learn, and had the basic functionality needed to complete the design. Additionally, Pocket C proved to be a more friendlier programming environment, presenting a straight-line approach to program design.

The first step of the code development was creating the necessary toolkits that would be used to create the GUI. This toolkit would be the root of the system by handling virtually all user input. The toolkit includes functionality for: buttons, drop down list, text lines, and press-able bitmaps (using an image as a button). With the design of a flexible set of functions in the GUI toolkit, the design moved onto the layout of the main forms. These forms are simply separate functions that set up different buttons with the GUI toolkit and display information. The program is further organized into separate files: the main file that deals with general program control and functionality, a utilities file that contains the graphing functions necessary for the program, and a file containing functions to calculate information about dates. An additional file contains the I/O library required for interacting with the Springboard interface.

Software Verification

In order to verify the software, a program was created to simulate user input into the system through the random selection of three input values per day over a two year period. This allowed the data retrieval and graphing functionality of the program to be fully tested. This also allowed a large number of records to be stored in the database to assess possible problems with speed of access, size, and data display when working with a large data file. With over two years of information in the database based on the assumption of three glucose readings a day, the size of the database was 109 KB. Typically, PDAs have 8 MB or more of storage space, thus this amount of data can be easily stored. The speed of the program slightly decreased when the program searched in the database for a previous date; however, it is still gave results within a few seconds and gave completely acceptable performance.

To ensure that the program would work on different platforms the POSE (Palm Operating System Emulator) system was obtained and used. With POSE, the group was able to test the functionality of the program with multiple version of the Palm OS operating system including versions 3.0, 3.1, 3.3, 3.5, and 4.0. The software functioned correctly on all of these systems.

Finalizing the Software Development

The software development platform arrived much earlier than the hardware; thus, this part of the project was completed earlier than anticipated on the original project plan. Once the hardware design was completed, there were additional modifications that were made to finalize the software. The biggest change was adding hardware specific information within the program. Using this information, the program sets up the hardware and obtains the digital reading of the glucose level. Error checking was also added to the software during this process so that the reading would be assured to be within a specific range of 10-500 mg/dL (the common unit of blood glucose level measurement). This range was chosen to match the range indicated within the patent information. With the final modifications complete, the software was tested with the hardware and functioned to the design specifications.

In order to properly function, the final software package requires several files to be installed. Table 1 summarizes the required files and the memory that they require on the system.

Table 1: Necessary Programs to Install

File	Size (kb)	Function
GlucoMon	61	Main Program
IOLib	2	Needed For Access to Hardware
MathLib	50	Part of Pocket C, additional Math Frunction
PcktCDateLib	4	A Pocket C toolkit that allows access to PalmOS Calendar

When completely installed, the program required 117 KB of memory within the Handspring Visor (this excludes whatever memory is used for the database of patient values). This is small enough to fit within any current Visor on the market.

Results, Observations and Conclusions

The senior design group was successful in their development of a "proof of concept" device; the entire hardware system is shown in Figure 6. Part of the reason for their success was the open source nature of the Visor platform. The Handspring developer's web site[5] contains several valuable documents for developers, including the "Springboard Development Guide for Handspring Handheld Computers" which is invaluable for Springboard module development. Other links lead to lists of suppliers for Springboard housings, product technical specifications, application notes, emulators, and software development kit information. Another reason for the success of this project is the relative ease of assembling a working prototype due to the large size, when compared to other interfaces such as Compact Flash (CF) or Secure Digital (SD), of the Springboard interface. This allows students to build prototypes without the need for highly specialized miniaturization equipment.

As of the writing of this paper, Handspring is still marketing the Visor Pro and Visor Platinum models at inexpensive prices (for PDAs) through their web site[6]. Unfortunately, this platform is on its way out, as Handspring is now pushing their Treo line, which uses the SD expansion interface instead of the Springboard interface. While development of new interfaces for the

Figure 6. GlucoMon system showing software application, Springboard module, and test strip

Visor are, in most cases, no longer commercially viable, it is still an excellent platform for student projects, due in part to the wealth of design information that continues to be available. Additionally, as corporations unload the remainder of their Visor and Visor-related products, significant savings can be achieved through judiciously-timed purchases, which is always beneficial on a tight budget.

It turned out that this project was more "state-of-the-art" than initially realized. This project was finished in March of 2002, and had progressed as far as it could without having to deal with FDA-related issues for the marketing of medical devices. In July of 2002, TheraSense announced the FreeStyle Tracker Diabetes Management System[7], which is a glucose monitoring system integrated into a personal digital assistant; specifically, a Handspring Visor. The operation of their device is similar to the project presented here, with the ability to test and record glucose levels, graph the results over time, and act as a logbook. Given that this senior design project was conducted by a handful of students who were working with limited resources and facilities (when compared to an established company) while simultaneously taking several classes required for the completion of their engineering degrees, the results of their efforts compare very favorably to the TheraSense design. This senior design project was a very valuable experience, resulted in a successful design, and provided a roadmap for those who wish to conduct similar design projects.

Bibliography

1. Bayer, U.S. Patent No. 5,120,420 (June 9, 1992)
2. Xilinx, "Understanding the Insight Springboard Development Kit," XAPP359, July 11, 2001. Available: http://www.xilinx.com/xapp/xapp359.pdf
3. Xilinx, "Designing an Oscilloscope with the Insight Springboard Kit," XAPP149, September 25, 2001. Available: http://www.xilinx.com/xapp/xapp149.pdf
4. http://www.orbworks.com
5. http://www.springboard.com/developers/
6. http://www.springboard.com
7. http://www.therasense.com/tracker/

Biographical Information

JOHN K. ESTELL became Chair of the Electrical & Computer Engineering and Computer Science Department at Ohio Northern University in 2001. He received his BS (1984) degree in computer science and engineering from The University of Toledo and received both his MS (1987) and PhD (1991) degrees in computer science from the University of Illinois at Urbana-Champaign. His areas of interest include interface design and embedded applications. Dr. Estell is a member of ACM, ASEE, IEEE, Tau Beta Pi, and Eta Kappa Nu.

JEFF HAAR graduated from Ohio Northern University with the Bachelor of Science in Computer Engineering degree in 2002.

JOSH LEMKE graduated from Ohio Northern University with a Bachelor of Science in Computer Engineering degree in 2002.

JEREMY SAUNIER graduated from Ohio Northern University with a Bachelor of Science in Computer Engineering degree in 2002.

ADAM SMITH graduated from Ohio Northern University with a Bachelor of Science in Computer Engineering degree in 2002.