Development of a Prescription Drug Management System (SmartPill)

A Thesis
In TCC402

Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
Of the Requirements for the Degree

Bachelor of Science in Computer Engineering

by

Spence Green

Other Group Members:
Jonathan Kelley, Electrical Engineering
Brad Pinney, Electrical Engineering

March 23, 2004

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed_____

Approved: _____ (Technical Advisor)
Ronald D. Williams

Approved: _____ (TCC Advisor)
Claire Chantell

Preface

The SmartPill project was part of the Engineering in Context (EIC) program at the University of Virginia School of Engineering and Applied Science. The EIC program exposes students to the non-technical aspects of engineering. Organizational issues such as legal requirements as well as cultural and ethical issues are discussed at length in the curriculum. In addition, the program allows students to carry out their thesis work as part of a multidisciplinary team. Proposals are made in the spring of the third year. Teams are then formed to develop the accepted proposals during the fourth year. Funding for both the project and the EIC program was provided by Lockheed Martin Corp.

I would like to thank Jonathan Kelley and Brad Pinney, my fellow group members. In this paper, I have noted the portions of the project that these two individuals worked on. Professor Ronald D. Williams initially highlighted the problem addressed by SmartPill. I would also like to thank Clement Song, who co-authored the initial proposal. He had the first conversation with Professor Williams that led to the idea for this project. Finally, Professor Dan Bauer, the project's EIC advisor, was instrumental in the project's success. Our conversations with him frequently fell outside of the technical realm. We learned much more from him than just engineering.

Table of Contents

List of Figures

Abstract

Elderly, illiterate, uneducated, and otherwise impaired individuals have difficulty complying with prescription drug regimens. As a result, misuse is common among these types of patients. Researchers have documented a number of reasons for this problem, but the most common is confusion. SmartPill combats this problem in an innovative way. The product, which consists of an automated medication dispenser and special drug cartridges, reduces the risks associated with prescription drug use through the application of Information Technology (IT). Customer needs were determined through interviews and a review of relevant literature. This research confirmed that no product on the market today successfully automates the cataloguing of user medication and generation of a usage schedule. As a result, SmartPill's approach is novel. A number of organizational obstacles must be overcome, however, for the project to be successful. For example, the system would require most pharmacies to change their business practices.

A proof-of-concept was implemented to demonstrate the system's novel aspects: the electronic transfer and synthesis of prescription drug information. Commercial-off-the-shelf (COTS) parts were used to build the proof-of-concept. The parts were chosen after a rigorous concept selection process. To date, the proof-of-concept has met the system's design goals. Several recommendations for further system development are also made.

Glossary of Terms

1-Wire—The serial communication protocol used to communicate with Dallas Semiconductor iButtons.

Commercial-of-the-shelf (COTS)—Any product readily available in the marketplace, as opposed to components that must be custom made-to-order.

eXtreme Programming—A lightweight (agile) software design methodology that uses a "design a little, build a little" approach.

Gcc—An Open Source C compiler common on most Unix/Linux systems.

iButton—An electronic memory device developed by Dallas Semiconductor that has only two leads: data/power and ground. iButtons are attached to SmartPill cartridges.

JTAG—Acronym for Joint Test Action Group. An interface supported by some devices that allows program loading and in some cases, remote debugging.

MCU—Microcontrol unit (see MSP430).

Medication dispenser—A device that contains pills and releases them to a user at prescribed intervals.

MSP430—The microcontrol unit (MCU) used in the proof-of-concept.

Proof-of-concept—A limited version of the final or virtual product. The proof-of-concept demonstrates the novel system aspects: medication cataloguing, dynamic usage scheduling, and harmful combination detection.

Refactoring—One of eXtreme Programming's twelve fundamental principles. It involves rewriting and optimizing existing software code.

SmartPill cartridge—A pill container with attached electronic memory.

Virtual product—The final conception of the SmartPill system. The virtual product would be a final functional version of the proof-of-concept ready for manufacture and sale.

XP—See "eXtreme Programming."

1.0 Introduction

1.1 Statement of Thesis

Drug misuse due to either ignorance or accident is a growing problem. The National Institute on Drug Abuse (NIDA) lists prescription drug misuse by elderly patients as a major healthcare industry concern:

> The misuse of prescription drugs may be the most common form of drug abuse among the elderly. Elderly persons use prescription medications approximately three times as frequently as the general population and have been found to have the poorest rates of compliance with directions for taking a medication. [3]

The sheer quantity of prescription medications consumed by patients necessitates a technological solution. Bates and Gawande claim that the application of Information Technology (IT) is the best solution: "Safe [patient] care now requires a degree of individualization that is becoming unimaginable without computerized decision support" [1].

This paper details the development of SmartPill, a prescription drug management system that combats accidental drug misuse. The system facilitates electronic transmission of product information from a pharmacy to an end user. A fully automated medication dispenser receives that information and manages the user's drug regimen. My research focused on three aspects of the project: development of the software artifacts, the methodologies used to develop those artifacts, and hardware/software integration.

1.2 Problem Definition

Why do geriatric patients misuse prescription drugs? The project's first task was to determine the factors that lead to patient non-compliance. Professor Ronald D. Williams of the Electrical and Computer Engineering Department at the University of

Virginia originally alerted the project team to the problem. Professor Williams has worked as an Emergency Medical Technician (EMT) since 1991. On numerous occasions every year, he has been called to the home of an elderly person who has overdosed on his medications. Usually the person overdosed for one of several reasons:

- The patient forgot that he had already taken his medication at one point during the day. The patient then took his medication again and overdosed.
- The patient was taking so many medications simultaneously that he either became confused and took the wrong medications or took his medications in the wrong dosage.
- The patient's pharmacist or doctor did not detect a harmful medication combination in the patient's regimen. Presently, a pharmacist only knows about the medications that a patient has filled at that particular pharmacy, making it difficult for the pharmacist to detect harmful combinations.

As a result of this interview, the project team originally focused on geriatric patients. The next step was to determine how current prescription drug distribution practices lead to misuse. CVS and some of the other national franchises would not speak about the operation of their stores. Fortunately, the group found Amy Alderman, a former CVS employee, who provided the needed information. In an interview, she listed several factors that lead to misuse.

First, few patients read prescription product literature. The pharmacy's responsibility is limited to providing the literature and answering any questions. Ms. Alderman has watched patients discard medication literature before exiting the store. Furthermore, patients who fill multiple prescriptions simultaneously are overwhelmed by the sheaf of literature they receive [4].

Second, medication conflicts frequently arise when multiple doctors prescribe drugs for a patient. Pharmacists can detect these combinations, but only if they have a complete picture of the patient's drug regimen. Unfortunately, pharmacies only track

prescriptions filled at that particular pharmacy. For example, if a patient fills a prescription at CVS and then fills another at Walgreen's, the pharmacist at Walgreen's is not aware of the previous prescription. As a result, pharmacists frequently cannot detect harmful medication combinations in the patient's drug regimen. Some of the larger chains such as CVS have networked their own stores, but pharmacists still do not have the information they need to protect patients [4].

Third, most pharmacies do not create usage schedules for their patients. The user must synthesize all product information and create a daily regimen. This task becomes unmanageable for geriatric patients without assisted care. Ms. Alderman stated that illiterate, uneducated, and non-English speaking patients also have this problem [4].

Finally, pharmacists are almost completely unaware of the over-the-counter (OTC) and alternative medicines that a patient uses [4]. Consequently, the patient has no safeguard against conflicts between prescription drugs and OTC medications. Since OTC and alternative medicines can be purchased in many other places besides pharmacies, an end-user solution, i.e. a solution in the user's home, is the only practical way to prevent harmful medication combinations. In summary, there is a substantial need for an IT-based, prescription drug management system.

1.3 Literature Review

The research for this project fell into the following areas: customer needs, a novelty search, and device and concept selection. Customer needs were determined primarily through expert interviews. The novelty search involved reviewing United States patent literature and products currently on the market. The group conducted device and concept selection by first determining a set of performance and cost metrics

based on the SmartPill Product Specification [19]. Several commercial-off-the-shelf (COTS) components were then rated against the metrics. The highest-rated components were selected for the proof-of-concept.

Since my primary responsibility was software design, I also researched software development methodologies. Lightweight design has been an active research area over the last few years, so I decided to try eXtreme Programming (XP) for this project. Lightweight design involves a "design a little, build a little" approach, whereas conventional methods such as the waterfall and spiral methods require extensive up-front design *before* implementation [20:4]. Authoritative XP texts such as Kent Beck's *eXtreme Programming Explained* were reviewed at length. The research focused on adapting lightweight software design to a project with concurrent hardware development. Section 2.2 introduces and explains XP.

1.4 Rationale and Scope of Project

As the so-called "Baby Boomer" generation ages, the need for SmartPill will increase dramatically. A 1999 AARP study showed that two of the top three needs for assisted living patients were medication dispensing and medication reminders. Fifty-nine percent of all patients required the former while 50% required the latter [18]. Patients that cannot afford assisted living, which currently costs several thousand dollars a month, have an even greater need.

Although pharmaceutical management systems have existed for years, the SmartPill approach is novel in its degree of automation: it requires no user programming. Other systems require the user to enter product information. Research clearly shows, however, that the target user base cannot understand prescription drug information in the

first place. Furthermore, geriatric patients usually cannot perform complex tasks such as programming. SmartPill catalogues all medications dynamically and generates a usage schedule. This approach will eliminate the need for a knowledgeable caregiver to setup the system, a requirement of every other design on the market. Finally, the use of electronic product information allows the system to detect harmful medication combinations.

1.5 The SmartPill System

SmartPill users own an automatic medication dispenser that accepts SmartPill cartridges, pill storage containers with electronic product information. Pharmacists use a programming cradle and a software application to load information onto cartridges. The system has seven basic features:

- Cataloguing of medications available to the user dynamically when medicines are introduced to the system. No user input or programming is required.
- Audio and visual cues to guide the user through the process of taking medications.
- Automatic detection of potentially harmful medication combinations. The user is alerted through audio and visual cues.

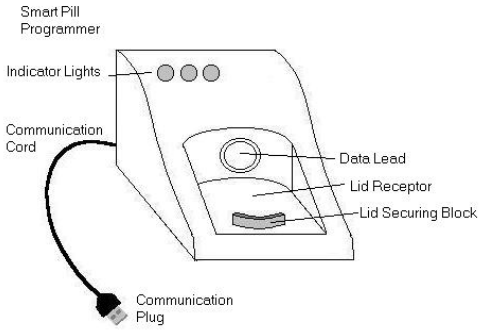


Figure 1--SmartPill pharmacist programming cradle. Drawing by Jonathan Kelley.

- Logging of the user's compliance history. SmartPill also prompts the user for a refill when the number of pills in a cartridge drops below 5%.
- A scheduling feature that is configured automatically when a new medication is introduced to the system. This feature prompts the user when, in what quantity, and how to take the medication.
- An automatic product information updating feature that does not rely on the Internet/web.
- Securing of all SmartPill cartridges to prevent user access at unauthorized times.

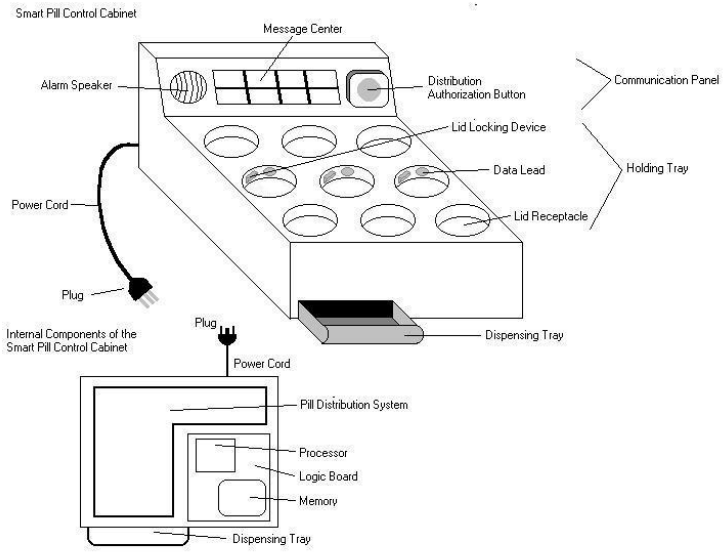


Figure 2--SmartPill automatic medication dispenser. Drawing by Jonathan Kelley.

1.6 Overview of Report

This report details the SmartPill system development. It summarizes the hardware-related activities and focuses on software development and system integration, my primary responsibilities. Section 2 lists the research conducted during the project. Section 3 covers the methods employed by the project team for project management, software design, implementation, and testing. In addition, this section covers the product development lifecycle. In section 4, the proof-of-concept's noteworthy aspects are explained. The proof-of-concept was the focal point of the project, so important design decisions and tradeoffs are discussed at length. The final two sections, 5 and 6, list the project results and conclusions. Most importantly, section 6.1 contains recommendations for future work. Several appendices supplement the information in the text.

2.0 Literature Review

The research conducted for this project fell into the areas of customer needs, a novelty search, and device and concept selection. The former two items related to the virtual product and determined the product features (see section 1.5). Those features were then prioritized and the most important items were implemented. Device and concept selection, which related to the proof-of-concept, was driven by the virtual product research. Application notes, datasheets, and part specifications were consulted. Finally, I conducted software development research in order to adapt lightweight design to the project. This section details the important research findings.

2.1 System Research

Automated pill dispensers first appeared in the mid-1980s. Since that time, inventors have developed a number of variations. Most of the relevant literature comes from two sources: patents and existing product documentation. In addition, several sources were used to determine customer needs.

2.1.1 Customer Needs

Professor Ron Williams, an Emergency Medical Technician (EMT), originally informed the group of the need for a better medication organization system. He noted that elderly persons often misuse medications for two reasons: confusion and ignorance [5]. The National Institute on Drug Abuse (NIDA), a subsidiary of the National Institutes of Health (NIH), corroborated Professor Williams' claims. Data taken from Veteran's Affairs Hospitals around the country shows that elderly persons take three times as many medications as the average adult and have the poorest rates of compliance [3].

Amy Alderman, a 4th Year Nursing Student at the University of Virginia and CVS employee, detailed current drug distribution methods (see section 1.2) [4]. Customer

needs were determined by analyzing the deficiencies in those methods. Recent journal literature has also addressed the misuse problem. Bates and Gawande investigated ways to improve patient safety [1]. They concluded, "If medicine is to achieve major gains in quality, it must be transformed, and information technology will play a key part, especially with respect to safety" [1]. Few standards existed for the application of Information Technology (IT) to healthcare, though, until Congress passed the Consolidated Health Informatics legislation in March 2003 [2]. This legislation provides the legal foundations and standards necessary for IT solutions to patient safety problems.

2.1.2 Novelty Search

Lewis and Roberts created one of the first automated pill dispensers in 1986 [12]. This device has served as a reference for almost every subsequent design. It consists of a detachable, rotating wheel with 12 storage compartments. A pharmacist can preload this wheel. Pills are mixed together in the compartments and dispensed at regular, programmed intervals (every two hours, four hours, etc.). An audio alarm alerts the user that pills have been dispensed. A major flaw is that dispensed, unconsumed pills build up in the base of the apparatus, making it easy for the patient to overdose. In addition, the patient must be competent enough to program the device.

Agans submitted a more complex design in 1992 [11]. A microprocessor controls this machine, which dispenses pills at finer intervals than the Lewis and Roberts design. Agans does not use a rotating wheel for dispensing, a major innovation. As a result, the dispenser has more compartments while still maintaining a small footprint. The mechanical dispensing mechanism is far more complex, though, than the elegant rotating wheel. The dispenser provides push buttons for programming and an audio/visual alert system.

Shaw solves many of the earlier designs' problems [10]. His system adds a "Dispense" button that the user must press to dispense medicine. Consequently, dispensed pills do not accumulate. This advanced system receives, stores, and processes prescription data, which the user enters with a folding keyboard. The top lid locks to prevent medication access at unauthorized times. Finally, Shaw adds a logging feature to track patient compliance.

Lim made the first attempt at "coaching" the user with his 1999 design [9]. This dispenser allows a caregiver or pharmacist to record audio usage instructions for each medication. In addition, the device has a communication subsystem that connects to the phone network. If the patient does not comply with his regimen, the system can alert an off-site caregiver. This device, however, does not account for human error. The audio instructions can become disassociated with the corresponding medication.

Two designs have been submitted in the past year. Lim improved his design by solving a previously unaddressed problem: securing loose pill bottles [8]. All previous designs require the patient to transfer pills from bottles to the dispenser. Excess pills remain in the pill bottles. The system cannot track these pills and for a patient taking multiple medications, the bottles can become lost. Lim's new design locks the loose bottles in a compartment. Hubicki also takes a novel approach by providing compartments for medications in non-pill form (liquid, powder, etc.) [13]. His design incorporates a mobile paging unit that prompts the user to take medications. Hubicki simplifies programming of the device to simply moving analog switches. He also notes that programmable systems in the prior art "do not appeal to many senior citizens who have limited incomes and little experience in programming high-tech equipment" [13].

One inventor approached the development of a medication organization system without using a storage cabinet. Sagar submitted a system with an electronic pill bottle cap [14]. The cap has lights that indicate when a medication should be taken. The lights utilize color coding to instruct the user. The cap also has a sensor that tracks compliance. Sagar's most important innovation involves the electronic transfer of product information. The system retrieves information from a database and automatically programs the bottle cap, eliminating the need for manual programming. The medications are not automatically dispensed, however, meaning that the user must still pick pills out of loose bottles at designated times.

Several relevant products are currently available on the market. The Talking-Rx is a small device that attaches to the base of a pill bottle [16]. A pharmacist or caregiver can then record audio instructions. The user has a small receiver that plays the instructions. E-pill of Wellesley, MA also sells several dispenser models including the MD.2 [17]. This product features 60 pill containers, audio/visual usage prompts, and a lockable cabinet. In addition, it can call a caregiver over the telephone to report compliance and refills.

No product in the prior art both eliminates user programming *and* secures loose pill bottles. Furthermore, the only system that does not require manual programming, Sagar's design, needs network access. Patients in rural areas may not have such connectivity. Finally, no extant system detects harmful medication combinations. In summary, by solving previous design flaws, SmartPill minimizes patient risk like no other product.

2.1.3 Proof-of-Concept

Vendor part documentation aided device selection and supported development. Section 4.1 describes the proof-of-concept parts selection process at length. Most of the documentation related to the Texas Instruments MSP430 microcontroller (MCU) and the Dallas Semiconductor iButton products.

The *MSP430x4xx Family User's Guide* is the authoritative MSP430 document [23]. This extensive document contains schematics, an instruction set description, and block diagrams. TI also provides datasheets for the two particular MCUs used in the project: the MSP430F413 and MSP430F449 [40] [41]. The mspgcc compiler documentation was also helpful [42]. The compiler developers provide efficiency tips and document JTAG debugging problems. In addition, this guide contains invaluable instructions that guide the user through the involved mspgcc installation process. Finally, the Olimex website contains information about the MSP430 development boards [28]. Olimex provides schematics, datasheets, and sample assembly code for both the MSP430-413STK and MSP430-449STK boards.

Dallas Semiconductor supplies extensive documentation for its iButton products. Most of the documents consulted during this project related to iButton software interfacing. The encyclopedic *Book of iButton Standards* contains a detailed description of the iButton design [43]. It explains the merits of the product, technical specifications, and the differences between the different iButton models (NVRAM, EPROM, and EEPROM). The document also details the 1-Wire communication protocol. Several other papers supplement the information in this guide.

White Paper #2 describes the different files in the Public Domain Kit (see section 4.4.1), including which files are needed to build the protocol stack [44]. It serves as a

good, non-technical introduction to the software. The "1-Wire Software Resource Guide" (Application Note 155) is a more comprehensive technical document [24]. This helpful guide describes every method in the Public Domain Kit and explains each layer in the protocol stack. Most importantly, it lists the methods that must be implemented to port the software to different hardware architectures. Application Note 126, "1-Wire Communication through Software," gives an example implementation of those methods [22]. It also describes the 1-Wire bus timing constraints. Used in conjunction, Application Notes 155 and 126 can guide a user through the process of porting the Public Domain Kit to various unsupported hardware platforms.

Two other iButton references were used during software development. Application Note 187 describes an algorithm for searching a 1-Wire network for new devices [45]. iButtons use a master/slave bus architecture and a large number of slaves can be chained onto the bus. The master must search the network to find the unique read-only memory (ROM) numbers of each new slave. Application Note 2420 was also helpful. This paper shows how to port the Public Domain Kit to a Microchip PICmicro microcontroller [6]. The PICmicro MCU is similar to the TI part used in the SmartPill dispenser. As a result, the procedure described in this paper was adapted to the MSP430 porting process.

2.2 Software Development

Lightweight (agile) software design has been an active research area over the last few years. The group wanted to try lightweight design for both their own education and to investigate how it worked with parallel hardware development. Several methodologies exist including eXtreme Programming (XP), Feature-driven Development (FDD), and

Crystal. Of these agile methods, XP is the most fully formed [20:4]. Kent Beck's *eXtreme Programming Explained* is the authoritative XP text. Beck, along with Ward Cunningham and Ron Jeffries, developed XP at Chrysler in the late 1990s. Many books have been written on the topic over the last several years including Steinberg and Palmer's recent *Extreme Software Engineering: A Hands-On Approach*, which serves as textbook for a college-level XP course. The information in these two books guided SmartPill software development. This section gives a brief overview of XP.

XP is based on the notion of "just-in-time design" and "…is based on incremental development and continuous integration" [20:94] [20:12]. It uses an evolutionary "design a little, build a little" approach that relies on continual feedback from the client [20:4]. Steinberg and Palmer write, "It [XP] accepts as given that the requirements for a system will change and brings a customer onto the development team to make quick decisions…." [20:4]. For Beck, XP is distinguished by short feedback loops, an incremental planning approach, a schedule that responds to changing business needs, reliance on automated tests, reliance on communication, and reliance on an evolutionary design process [25:xvii]. Most software projects use front-loaded design, such as the waterfall or spiral method, in which most design work is done before coding. XP, on the other hand, leverages emergent behavior by relying on small, localized decisions that evolve into an optimal design.

Beck articulates XP's twelve fundamental principles: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer, and coding standards [25:54]. Several of these principles require further elaboration. First, "small releases"

means delivering code to the client at the end of every development iteration (usually two weeks). This principle creates a short feedback loop between the client and the developer. XP literature refers to the "small releases" principle as "you aren't going to need it", or YAGNI. In short, the system should never be over-designed for unanticipated needs [20:98].

Next, XP strongly emphasizes testing. If a feature cannot be tested, then it does not work [25:45]. More importantly, XP requires an unorthodox "test and code" approach. Tests are created *before* code is written, thus requiring programmers to think about what the code does before writing it. Tests are compiled into an automated test suite that the system must always pass. JUnit, an automated testing tool created by Beck and Erich Gamma, is the *de facto* standard [20:57].

Refactoring is another unique XP principle. When writing new code, the developer should always find the simplest working solution. Later, working code is *refactored*, or rewritten for optimization. The code itself is subject to an evolutionary process. Finally, pair programming is the most noteworthy—and controversial—XP principle. Code is always written in groups of two, with a "pilot" and a "navigator." The pilot writes the code, while the navigator consults documentation, designs, and checks the pilot's work. At any time, the pilot and navigator can switch roles. The rationale for this scheme is that two people can find defects faster than one, but three people is probably too many at a single workstation. Some developers, though, are hesitant to work with a partner. As a result, this principle is frequently ignored.

To design a system, XP developers create user stories on 3 in. by 5 in. index cards. Each story captures a way in which the system is used. The client must accept

each story. Stories are then broken into tasks and assigned to programmers for implementation. New stories are added to the system as needed. Appendix A lists the stories used in this project.

Any XP development effort should focus on three critical issues. First, dissemination of information within the group is critical; "XP depends on timely, rapid communication" [20:87]. Email is usually the preferred communication medium. Second, "No code sits unintegrated for more than a couple of hours. At the end of every development episode, the code is integrated with the latest release and all the tests must run at 100%" [25:97]. Finally, the development group should always have one set meeting time and place per week to plan and coordinate [20:87]. In summary, the development team should work for a rhythm: "learn/test/code/release" [25:99].

3.0 Methods and Approach

Over three academic semesters, the SmartPill project went from problem identification to implementation. A well-defined process directed the development effort and helped record the group's progress and accomplishments. This section describes the three-phase development process. It also details the project management techniques used and how those techniques evolved in response to changing needs. The section also illustrates lightweight design in practice. Finally, it lists the laboratory equipment.

3.1 Project Management

Two important aspects of project management are *tracking* progress and *planning* future direction. Tracking is important for two reasons. First, it leads to opportunities for improvement [31:15]. One of XP's main virtues is a short feedback loop. This feedback loop requires accurate, detailed information. Without the management infrastructure to make information flow possible, XP is severely crippled. Second, new individuals cannot be added to the project without detailed documentation. For example, a virtual product development effort would require several mechanical engineers. Without a description of the design decisions that had been made, these engineers could not efficiently do their job.

Planning is critical to any project's success. Ulrich and Eppinger write that the timing of milestones "…anchors the schedule of the overall development project" [31:15]. The SmartPill project had finite time constraints. By setting milestones, the group narrowed the project's scope. The planning activity helped the group decide what could be accomplished and ensured that those tasks would be completed on time.

3.1.1 Tracking

All group members recorded design decisions, meeting minutes, and other pertinent information in engineering logbooks. The other tracking tools used during the project changed between the Research and Design (see section 3.2.2) and Implementation (see section 3.2.3) phases.

During the Research and Design phase, the group used eproject, a web-based project management suite [32]. eproject features a logged email list, a document revision control tool, and a calendar. Whenever a change occurs (such as the posting of a new document), the system notifies group members via email. This feature greatly enhanced communication within the group. eproject had allowed free academic licenses, but it discontinued that service in January 2004. Consequently, the group looked elsewhere for a low-cost alternative.

Information Technology and Communication (ITC) at the University of Virginia provided eproject's services at no cost. ITC supplied a logged email list through the school's listserver. Several gigabytes of storage on a Sun Microsystems Solaris server were also provided. The server hosted a Concurrent Versions System (CVS) repository, which provided version control for both documents and source code [33]. A Gantt chart stored in the CVS repository replaced the group calendar.

Mantis version 0.16, an Open Source error/issue tracking system, was also used during Implementation [34]. Mantis features a web-based front-end and a MySQL database back-end. The MySQL back-end was chosen for data transportability. Issues related to software, hardware, and documentation were posted on the website. Responses were added to the postings until the issues were resolved.

3.1.2 Planning

Planning occurred at the beginning of each phase. A portion of each weekly meeting was also devoted to readjusting milestones according to the current stage of the project. Each phase's milestones were compiled into Gantt charts. During the Proposal and Research and Design phases, milestones corresponded to deliverables. Tasks required for deliverable completion were then arranged ahead of the milestone. Since fixed milestones anchored the schedules of those two phases, the calendars were fairly rigid. The Implementation phase, guided by XP, was more fluid.

At the start of Implementation, user stories were created from the Software Requirements Document, Hardware Requirements Document, and Product Specification [35] [36] [19]. The group collectively estimated the time required for each story and then grouped stories into one-week iterations. The story groups were then prioritized and compiled into a Gantt chart. Finally, each story was broken into individual tasks and assigned to different group members. Initially, the group's time estimations were inaccurate. For example, iButton software source porting (see section 4.4.1), which was estimated to take about 10 man-hours over one iteration, actually took 4 iterations and around 30 man-hours. Consequently, iterations were adjusted frequently at the beginning of the Implementation phase.

3.2 Project Phases

3.2.1 Proposal

The Proposal phase's main objective was identification of a problem that required a technical solution. Professor Williams, as mentioned previously, articulated the problems with geriatric prescription drug use (see section 1.2). After the problem was identified, some background research was conducted. A National Institutes of Health

(NIH) report on prescription drug abuse was the linchpin [3]. Next, potential solutions were brainstormed. The group considered several concepts based on complexity, cost, and most importantly, estimated time to complete a proof-of-concept. Finally, the group wrote a proposal that articulated the problem, outlined a solution, and listed the anticipated project requirements (see Appendix B). The proposal was selected from a field of fourteen entries by a three-member faculty panel to receive funding. Jonathan Kelley and Brad Pinney then joined the project.

Deliverable	Description
Project Proposal	Detailed project description and proposal consisting of a problem description, solution strategy, contextual analysis (organizational and cultural project aspects), project requirements, and verification strategy.

Table 1--Proposal phase deliverable listing.

3.2.2 Research and Design

The Research and Design phase contained five parts: customer needs, concept selection, requirements specification, up-front design, and implementation planning. Several important deliverables, discussed here, were created. Table 2 lists other secondary deliverables.

The Statement of Work and Product Specification were two critical documents. The Product Specification was based largely on customer needs (see section 2.1.1). This technical document listed all virtual product parameters. The Statement of Work showed which parts of the virtual product would be implemented for the proof-of-concept. It also listed all project activities for the final two phases of the project.

The concept selection activity was another important part of this phase. Several competing SmartPill designs were evaluated using criteria such as intuitive use, cost, and ease of integration into the existing pharmaceutical distribution infrastructure [37]. The

medication dispenser solution surpassed the other designs (one of which involved placing wireless sensors on pill bottles). Next, Hardware and Software Requirements documents were prepared as part of the up-front design activity. These documents, which stated *what* the system had to do, were also based on customer needs. Finally, Detailed Hardware and Software documents, which served as blueprints during Implementation, were created based on the requirements.

Deliverable	Description
Concept Selection	Contains the concept selection matrices used to evaluate several virtual product designs.
Contextual Analysis	Analysis of SmartPill's social, ethical, and organizational aspects.
Detailed Hardware Design	SmartPill hardware blueprint.
Detailed Software Design	SmartPill software system blueprint.
Hardware Requirements	Explains what the system hardware must do.
Preliminary Design Review (PDR) Report	Project status report for the PDR, which took place in mid-November 2003.
Product Specification	Contains the virtual product technical details.
Project Review II Report	Project status report for the final Research and Design Project Review.
Software Requirements	Explains what the system software must do.
Statement of Work	Lists proof-of-concept features and all Research and Design and Implementation development activities.

Table 2--Research and Design phase deliverable listing.

3.2.3 Implementation

The proof-of-concept was created and evaluated during this segment, which consisted of both implementation and verification steps. The latter step is slightly misleading, though, since software verification in XP occurs throughout implementation. Of the documents created, the Development and Test Schedule, which served as both a plan and a record of Implementation activities, was the most noteworthy.

The Development and Test Schedule's three main parts were the user story listing, the acceptance test listing, and the task log. One or more acceptance tests were created for each XP user story. The task log, which was updated weekly, listed both future activities and recorded past tasks. For each activity in the task log, the due date,

number of man-hours required to complete, and status were recorded. Later in the semester, a risk analysis section was added to the Development and Test Schedule (see Table 5). Two other important documents, the Functional and Verification Test Procedures, were created later in the phase.

The Functional Test Procedure would be used during production. The document listed 14 tests that would verify a product's operation immediately after it left the assembly line. Table 3 shows one of those tests. The Verification Test Procedure, on the other hand, would be used on a control group of virtual products after development completion. The Verification Test Procedure lists a detailed test, including the procedure and required equipment, for each item in the Product Specification.

Test 4	***Description***
Test Item Description	Pill bottle receptacle communication with microcontroller.
Test Objective	Ensure that microcontroller can read every receptacle.
Reference Documents	Hardware Requirements Software Requirements
Test Description	Insert a preprogrammed cartridge into the first slot. Then serially insert a cartridge with preprogrammed conflicting medication description into each other slot. The test is successful when the conflict signal is communicated to the tester in each case.
Test Equipment	Two SmartPill cartridges loaded with conflicting prescription information.

Table 3--A test description from the Functional Test Procedure. [37]

Deliverable	**Description**
Development and Test Schedule	Contains a user story listing, test equipment list, acceptance test list, task log, and risk analysis.
Functional Test Procedure	Lists a procedure for testing products immediately after manufacturing.
Proof-of-Concept	Functional system prototype.
User's Manual	Virtual product usage instructions.
Verification Test Procedure	A procedure for verifying that the product meets its specification after development.

Table 4--Implementation phase deliverable listing.

3.3 Applied Lightweight Design

The group made several modifications to XP during software development. First, since time was a limiting factor, the group prioritized the functionality that would be

implemented during the semester. Five tiers were defined (with one as the highest tier), with the lower tiers containing features that were not critical to the proof-of-concept:

Tier 1
- Catalogue medicines available to a user dynamically when medicines are introduced to the system. This cataloguing should require no user input or programming.
- Alert the user to any potentially harmful medication combinations.

Tier 2
- Provide a scheduling feature that is configured automatically when a new medication is introduced to the system. This feature prompts the user when, in what quantity, and how to take the medication.

Tier 3
- Log the user's compliance history and prompt the user to obtain refills when necessary.

Tier 4
- Lock all loose pill bottles in place.

Tier 5
- Make language-based/paper-based product information and education optional by relying solely on a combination of audio and visual prompts.
- Provide an automatic product information updating feature that does not rely on the Internet/web.

Second, the group classified each user story as a hardware component, a software component, or both. This activity provided a basis for dividing labor among the group members. Finally, the group identified several key technical risks.

	Risk Description	Abatement Strategy
1	Software code size	Ordered a new board with the maximum amount of memory the msp430 architecture can address.
2	Hardware defect or software defect?	When implementing a hardware feature, test that feature extensively before integrating with software (and vice versa). For hardware/software features, define tests that best isolate the hardware and software components of that feature.

Table 5--The risks associated with the onboard software development activity. [30]

After the process was defined, the next step was to determine the overall architecture of the system. Since the SmartPill dispenser is heavily I/O dependent, an architecture that polled for events seemed suitable. Events were defined for various actions and a main loop that polled devices and handled events was constructed. This architecture, which is quite robust, also merged well with XP. Feature changes and new functionality could be quickly implemented.

The final two up-front design steps involved specifying the system data formats and outlining the software classes using Class Responsibility Collaboration (CRC) cards. iButtons can store data as "files", so the format and contents of several files were specified. Table 6 shows an example file specification. Table 7 contains brief descriptions of the other files stored on SmartPill cartridges.

Filename	prod.00
Max size	TBD
Contents	(1) National Drug Code (4 bytes). (2) Patient Number (4 bytes) (3) Number of pills in bottle (1 byte)

Table 6--Product Information file format. Note that the data is packed along byte boundaries. [30]

File	Filename	Contents
Dispense Times	time.00	List of dispense times.
Pills per Dispense Time	pill.00	Pills to be dispensed at the times listed in time.00
Product Conflicts	cflt.00	List of medications that conflict with this medication.
Product Information	prod.00	National Drug Code, patient number, and number of pills in bottle.
Updates and Special Features	updt.00	Product updates for the whole system.
Usage Information	usag.00	Usage options for each dispense time in time.00

Table 7--Listing of the files stored on SmartPill cartridges. The data is aligned along byte boundaries in each file. [30]

Since the C language does not support object-oriented design natively, groups of functions were separated into different files to mimic classes. Each file/class was specified using a Class Responsibility Collaboration (CRC) card. Although XP stresses "just-in-time design," it was important to think about the overall structure of the system and the methods that each system component would need. Changes were made later during the implementation process, but the overall coherency of the design remained intact due to the up-front design work. Table 8 depicts an example CRC card.

Class:	hardware
Responsibility:	• Provides software interface to LEDs, audio alarm, and dispense button.
Attributes:	None
Services:	alarm_on() : short alarm_off() : short led_on(int num) : short led_off(int num) : short led_test(int num) : short dispense_pill(short bottle_num, short num_pills, int options) detect_dispense() : short

Table 8--Hardware CRC card. Alterations were made to the CRC cards weekly, but the partitioning of the system remained intact. [30]

4.0 Proof-of-Concept

Proof-of-concept implementation was the final phase of the SmartPill product development lifecycle. The proof-of-concept served two purposes. First, it demonstrated the system design's feasibility. Although not a complete implementation of the virtual product, this prototype showed that the system's novel aspects could function as designed. Second, the proof-of-concept served as a tangible artifact that could be demonstrated for focus groups, potential investors, and intellectual property consultants.

Several limitations constricted the proof-of-concept's scope. Money and time were the most obvious constraints. The total project budget (see Appendix C) amounted to only a fraction of the funding needed to fully develop the idea. Furthermore, the development team consisted of full-time students, which limited the amount of time they could spend on the project. As a result, the group focused on the novel aspect of the project: automating the medication dispenser. The mechanical pill dispensing apparatus was not implemented. Although an integral part of the SmartPill product, several acceptable dispensing designs already exist (see section 2.1.2). In the end, the SmartPill system would probably be integrated onto—and thus automate—an existing dispenser.

This section lists the development materials. Summaries of both the hardware and mechanical development efforts are also included. Finally, software development, my primary area of responsibility, receives a rigorous treatment.

4.1 Materials

All of the materials used during development were commercial-off-the-shelf (COTS) products. COTS materials have several virtues. First, by leveraging economies of scale, COTS products usually cost less than custom components. This factor was

crucial because of both the project's limited budget and the desire to minimize the virtual product's cost. Furthermore, COTS materials usually have short lead times. With only a few exceptions, most of the hardware components listed below can be procured at a good electronics store. The software development environment can be downloaded for free from the Internet [26] [27].

4.1.1 Microcontrol Unit (MCU)

The dispenser required an onboard processing unit. Mr. Harry Powell, a University of Virginia technical staff member, was able to focus the search for an appropriate MCU [7]. Some of the major requirements were low-power, availability of software programming tools, and amount of free I/O pins. Two potential units were selected: the Motorola 68HC16 series and the Texas Instruments MSP430F4xx series. These parts are comparable in terms of word size (16-bit), addressable memory, and power consumption. Furthermore, both have short lead times, extensive documentation, and are used for a wide range of embedded applications. For large orders, the unit price of both parts can be less than $10. A concept selection matrix was prepared to select one of the parts and two determining factors emerged: software tools and available development boards.

Motorola supports several commercial software options for their part, including Metrowerks Codewarrior. Alternatively, patches for GNU's gcc compiler are available on the Internet for certain parts in the series. The Open Source projects maintaining these patches are not very active, though. Texas Instruments, on the other hand, provides a free integrated development environment (IDE), the MSP430 IAR Kickstart package [29]. An active, well-documented port of gcc for the entire MSP430 series also exists for free download [26]. The latter option was attractive for the group, as all members had

experience with gcc and the other GNU C development tools. The MSP430 options were deemed better in terms of both cost and documentation.

The other determining factor was the availability of development boards. Several preconfigured boards exist for both lines of MCUs, but Olimex provides an MSP430 development board that features a preconfigured Dallas Semiconductor 1-Wire port. Dallas iButtons had already been selected as the SmartPill cartridge memory unit, so this feature was extremely attractive. No similar boards were available for the Motorola MCU. As a result of these two factors, the TI MSP430 was selected as the onboard MCU.

4.1.2 Olimex Development Board

Olimex provides a line of TI MSP430 development boards that have the following features:

- Preconfigured liquid crystal display (LCD)
- Four momentary state buttons
- Preconfigured Dallas 1-Wire port
- JTAG interface (for real-time debugging)
- RS-232 serial port
- Several free, general purpose I/O pins

Two different boards were purchased for development: MSP430-449STK and MSP430-413STK. Figure 3 and Figure 4 show the two boards:

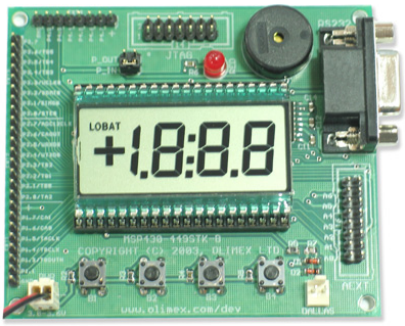


Figure 3--Olimex MSP430-449STK development board. The Dallas port is at the bottom right of the board. [28]

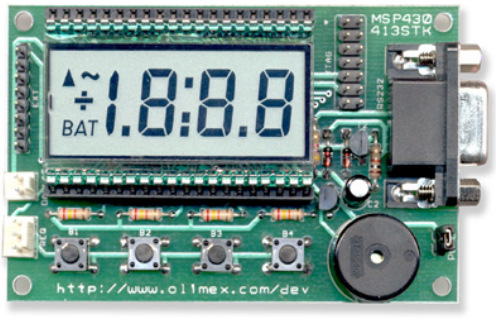


Figure 4--Olimex MSP430-413STK development board. [28]

The 449STK board, which contains a TI MSP430F449 MCU, was selected as the primary development board. Memory size was the main reason for this choice. The 449 board contains the maximum amount of memory that the MSP430 architecture can address: 60KB of ROM and 2048 bytes of RAM. The 413STK board was used as a backup. Since both parts shipped from Bulgaria and had long lead times, it was important to have a backup board.

4.1.3 Dallas Semiconductor iButton

The group considered several possibilities for the SmartPill cartridge memory. The first option was a magnetic stripe, similar to the technology found on credit cards. By using a magnetic stripe, the cartridges would not need any electronics. Furthermore, magnetic stripes cost only pennies. This technology can only store a few bytes of data, though, and magnetic fields common in many environments can erase that data [43:3]. SmartPill required several kilobytes of storage and data integrity was critical, so magnetic stripes were not suitable.

The second possibility was a standard EEPROM or NVRAM memory chip. Many different products exist which vary in terms of packaging, storage capacity, and power consumption. This solution was also cost-effective, with most parts priced at less than one dollar. Packaging was a major drawback, though, since most devices have several pins that must be connected. The group wanted a solution that had fewer pins so that the cartridge could slide into the medication dispenser with minimal user effort. Furthermore, the packaging on most memory chips was neither tamper- nor water-proof.

Radio frequency (RF) tags were another option. Pill bottles could communicate with each other wirelessly and share information. This option's main drawbacks were the

inability to secure loose bottles and track compliance. Users would still need to pick individual pills out of each bottle.

The final possibility was Dallas Semiconductor's iButton line of products. These devices use the 1-Wire (serial) protocol, which requires only two pins: Data/Power and Ground. Furthermore, iButtons come in water-proof, tamper-resistant metal packaging. Dallas provides a wide range of development materials including readers, PC-connectivity parts, and software.

The most important iButton feature, though, is the ability to create 1-Wire networks over a single bus line. The medication dispenser contains room for multiple cartridges. The onboard system must communicate with all cartridges at any time. If ordinary memory chips had been used, some of the MCU's address space would have been consumed by the external memory chips. This was not a desirable solution, since the code size of the onboard software was not known in advance. The 1-Wire protocol, though, provides the capability to communicate with all connected iButtons over a single bus line *without* using any MCU address space (see section 4.4.1). This feature was the primary motivation for using iButtons. Figure 6 shows the unique iButton packaging:

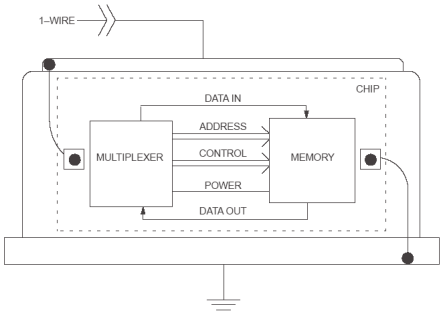


Figure 5--iButton block diagram. Opcodes on the 1-Wire bus control device's operation. [43:4]

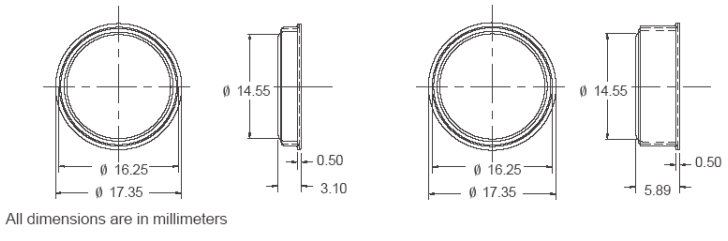


Figure 6--Dimensions of the iButton MicroCan package. [43:12]

Figure 7--This DS1402D-DR8 reader, which has two receptacles for 1-Wire network testing, connects to a PC via a port adapter.

Dallas provides free iButton samples, so EPROM, EEPROM, and NVRAM variants of the iButton product were obtained at no cost. The NVRAM iButton variant (DS1992L and DS1993L) was selected after testing revealed that it was the easiest to program. Several iButton readers (DS9092T, DS9092L, and DS1402D-DR8), RS-232 port adapters (DS9097U-S09), and circuit board mounting kits (DS9094FS and DS9106S-GN) were also procured. Several of the DS9092L readers were soldered together and then connected to the Dallas port on the Olimex development board (see section 4.2.2).

4.1.4 Software Development Environment

Two compilers were evaluated for the TI MSP430 software development environment. TI supports the IAR Kickstart suite, which is available for free download [29]. This Windows-based IDE includes a compiler, assembler, and debugger. Unfortunately, the software was difficult to configure in Windows 2000 and Windows XP. The group encountered problems with the JTAG parallel port interface as well as the assembler.

The other option was the Open Source mspgcc package, also available for free [26]. mspgcc is a patched version of GNU's ubiquitous C compiler, gcc. The package also includes a proxy server for JTAG debugging, an msp430 version of gdb (GNU's debugger), and several other tools. The library of header files included with the package is also an important benefit. Extensive documentation exists for mspgcc and its software libraries. Fedora Core 1, Red Hat's desktop Linux distribution, was used to test the package. mspgcc was configured quickly and all group members were familiar with

other variants of the tool, so the package was selected as the software development environment.

4.1.5 Fast Prototyping

Jonathan Kelley used AutoCAD 2004 to model the medication dispenser. A SmartPill cartridge was also created. A Stratasys Dimension fast-prototyping tool then used these models to create physical prototypes out of acrylonitrile butadiene styrene (ABS), a plastic resin.

4.1.6 Miscellaneous Materials

An IBM ThinkPad A20m laptop computer was used as a mobile laboratory. This system was dual-booted with Windows XP Professional and Fedora Core 1. The Olimex development boards and iButton readers were connected to the parallel and serial ports of this machine. The laptop allowed the group to operate independently of a fixed laboratory location. To complete the proof-of-concept, other miscellaneous electronic components such as light-emitting diodes (LED), resistors, and capacitors were used.

4.2 Hardware Development

Hardware development centered on enabling the Olimex boards' features and augmenting those features with several other components. The former area required more work than anticipated. The board shipped with no documentation and some of the information posted on the Olimex website was incorrect. TI's *MSP430x4xx Family User's Guide* was an invaluable resource during development [23].

4.2.1 MSP430 I/O Interface

The MSP430 architecture provides convenient access to its general purpose I/O pins. "MSP430 devices have up to six digital I/O ports implemented, P1 - P6. Each port has eight I/O pins. Every I/O pin is individually configurable for input or output

direction, and each I/O line can be individually read or written to" [23:8-2]. Figure 8 shows the different ports and the different connections to those ports on one of the Olimex development boards.

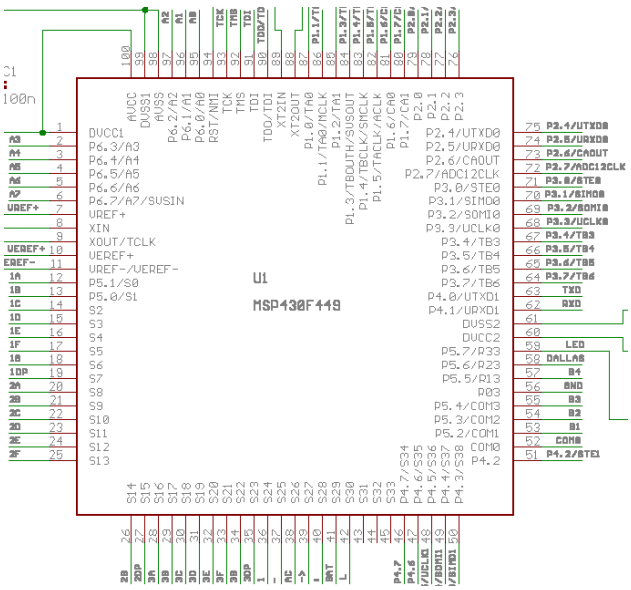


Figure 8--Olimex MSP430-449STK schematic showing the connections to the MSP430's various I/O ports. Note that the Dallas iButton 1-Wire port is connected to port 5, pin 6 (P5.6) on the far right.

Each port is mapped to several eight-bit registers in memory. Each bit in the registers corresponds to one of the eight port pins. The registers used in this project were INPUT, OUTPUT, and DIR. The DIR register specifies the direction of each pin. Bits are set to 0 for input or 1 for output. After setting the proper bits, the INPUT and OUTPUT registers can be used. The INPUT registers are read-only. When a digital signal appears on the pin, the corresponding bit in the INPUT register is set to the value of that signal. The user can write to the OUTPUT registers. For example, if bit 0 in the port 3 OUTPUT (P3OUT) register is set to 1, then a 1 signal appears on port 3, pin 0. The mspgcc software libraries define different C macros that allow intuitive control of these pins. The following C code segment demonstrates typical I/O pin usage:

```
#include <msp430x44x.h>       /*Defines msp430x449 macros */


int i;
P3DIR = 0x00;  /* Set all port 3 pins to input*/
i = P3IN;      /* Read the port 3 pins */
i &= 0x01      /* Mask upper 7 bits to look at pin 0.
                  i now equals the value of port 3, pin 0.*/
...
P3DIR = 0xFF   /* Set all port 3 pins to output*/
P3OUT |= 0x01  /* Set port 3, pin 0 to 1*/
```

Figure 9--MSP430 I/O pin control example. C code like this appears in the SmartPill onboard software.

The LCD, alarm buzzer, Dallas port, LEDs and buttons on the Olimex board are all connected to the MCU's I/O ports. By manipulating the port registers, these devices can be controlled in software.

4.2.2 MSP430 Clock Subsystem

SmartPill's scheduling feature depends upon an accurate 24-hour clock. MSP430 devices provide at least one configurable counter that can be used for such a purpose. Timer_A, an asynchronous, 16-bit timer/counter with three capture/compare registers, is available on all MSP430x4xx parts [23:11-2]. It has four configurable modes and a selectable clock source. The counter triggers interrupts that can be processed in software.

Timer_A can be set to count up to a value stored in register CCR0, as shown in Figure 10. When Timer_A reaches that value, it triggers an interrupt and then counts up again from zero. Both of the Olimex boards contain a 32,768 Hz crystal oscillator that functions as the MCU auxiliary clock (ACLK). Consequently, if the value 32768 is stored in CCR0 and the clock source is set to ACLK, then an interrupt is triggered every second. Variables in software can be adjusted in the interrupt handler to implement the 24-hour clock.

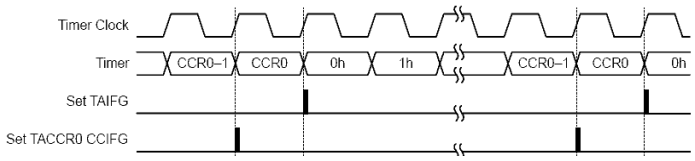


Figure 10--Timer_A "up" mode behavior. When Timer_A equals the value in CCR0, an interrupt is triggered and the timer resumes counting from 0. [23:11-6]

4.2.3 iButton Network

Figure 11 depicts the Dallas port layout on the Olimex boards. Pin 1 is grounded while pin 2 serves as the power/data lead. One of the 1-Wire protocol's main virtues is that multiple iButtons can connect to the same power/data lead. Each slot on the SmartPill medication dispenser required an iButton reader, so several DS9092L parts were simply soldered together and connected to the Dallas port. Resistance in the wires becomes significant if too many readers are soldered together, but this problem was not an issue for the small number of readers used in this project.

4.3 Mechanical Development

The plastic prototypes discussed in section 4.1.5 were modified using a Dremel tool. Holes were cut at the base of each cartridge slot for the iButton readers. Openings were also created for the different external LEDs on the device. Finally, the Olimex MSP430-449STK board was located underneath the prototype. The cartridges were also modified. Holes were created for the iButtons. Furthermore, the cartridges were sanded to fit in the dispenser prototype.

4.4 Software Development

The remainder of this chapter describes the onboard system's software. The topics covered are those that have been completed by March 23, 2004. Mundane topics such as libraries used are not discussed. The high-level functionality of this system is not complete, so only the lower-level system parts receive lengthy treatment.

4.4.1 iButton Software Development

Dallas Semiconductor provides iButton software drivers based on an Open Systems Interconnection (OSI) protocol stack architecture [21]. The software has been ported to several platforms including Windows and Java Virtual Machine (JVM). Fortunately, Dallas also provides a Public Domain Kit (version 1.02 was used for this project) for other architectures. This kit includes full Network, Transport, File, and Device layer implementations. It requires hardware-specific Session and Link layer implementations. Consequently, this development activity involved implementing those two layers.

Dallas iButtons require a platform with a bidirectional communication port (open-drain output and weak pull-up on the line) and the capability to generate an accurate and repeatable 1µs delay. Furthermore, communication operations must not be interrupted during generation [22:1]. The MSP430 has bidirectional communication ports and the Olimex development board already had a weak pull-up on the line connected to the Dallas port (see Figure 11).

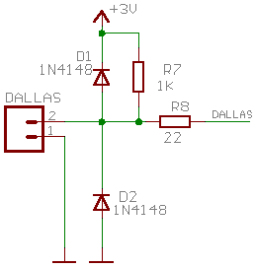


Figure 11--Layout of the Dallas port on the Olimex development board. This circuit provides a weak pull-up. The line labeled "Dallas" connects to one of the MSP430's I/O pins.

The latter two requirements, though, must be implemented in software. A 1µs delay can be synthesized by writing a method that, when called, "busy waits" for the desired time period. This method depends upon the clock rate of the processor and the number of machine cycles required for instruction execution. The MSP430's adjustable clock rate further complicated the issue. The Olimex board uses a 32,768 Hz crystal oscillator to drive the MSP430's auxiliary clock

(ACLK). The clock frequency of the processor (MCLK) is then multiplied by some user-defined integer factor. The schematic in Figure 12 shows how the MSP430 generates its clock signal. The crystal oscillator is farthest to the left between XIN and XOUT.

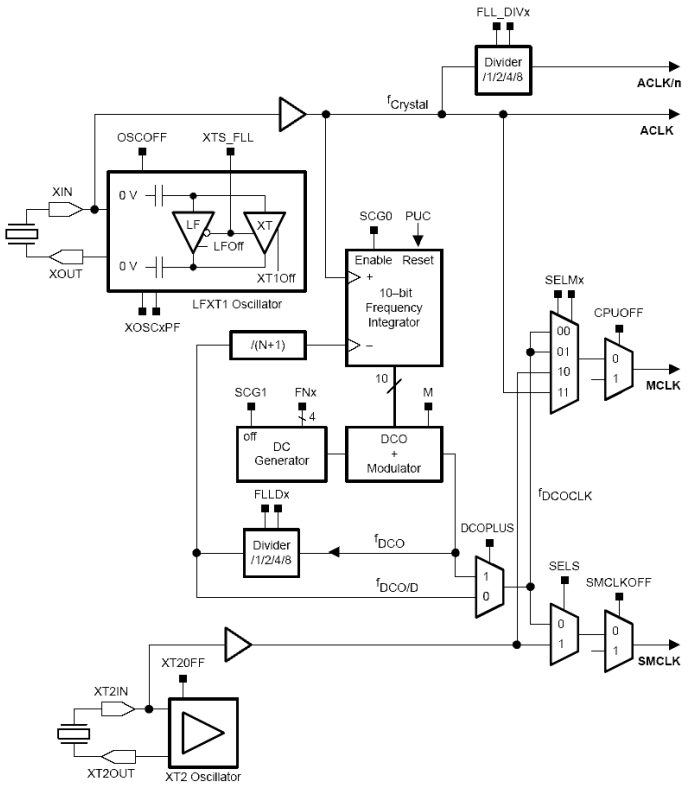


Figure 12--The MSP430's Frequency Locked Loop (FLL). Note the MCLK signal that clocks the microcontroller. [23]

The microcontroller uses 32 as the default clock multiplier, resulting in an operating frequency of 1.048576MHz. The only other information needed to write the delay function was the number of machine cycles required to execute instructions. The MSP430 can perform register operations in a single cycle [23:3-2]. Function call overhead requires an additional eight instructions, resulting in the following equation:

$$\frac{1}{MCLK} \bullet (n+8) = delay\ (\mu s)$$

Equation 1--MSP430 delay equation. *n* represents the number of register operations executed while busy-waiting.

A method was implemented in assembly language that simply decrements an integer input parameter until that parameter equals zero. Consequently, three machine cycles are consumed per iteration (jump operations require two cycles). Equation 1 then becomes:

$$\frac{1}{MCLK} \bullet (3n + 8) = delay\,(\mu s)$$

Equation 2--Final MSP430 processor delay equation. This equation factors in three machine cycles per loop iteration.

After the delay method was implemented, it was possible to implement the Link-level methods in the iButton protocol stack. These methods included owTouchReset and owTouchBit, which reset the 1-Wire bus and perform read/write operations, respectively [24]. A specific algorithm is used to communicate over the bus, as shown in Table 9:

Operation	Description	Implementation
Write 1 bit	Send a '1' bit to the 1-Wire slaves (Write 1 time slot)	Drive bus low, delay 6µs Release bus, delay 64µs
Write 1 bit	send a '0' bit to the 1-Wire slaves (Write 0 time slot)	Drive bus low, delay 60µs Release bus, delay 10µs
Read bit	Read a bit from the 1-Wire slaves (Read time slot)	Drive bus low, delay 6µs Release bus, delay 9µs Sample bus to read bit from slave Delay 55µs
Reset	Reset the 1-Wire bus slave devices and ready them for a command	Drive bus low, delay 480µs Release bus, delay 70µs Sample bus, 0 = device(s) present, 1 = no device present Delay 410µs

Table 9--1-Wire operations table [22:1].

Figure 13 shows the above information represented as waveforms. The bus is controlled by manipulating the I/O pin connected to the circuit in Figure 11.

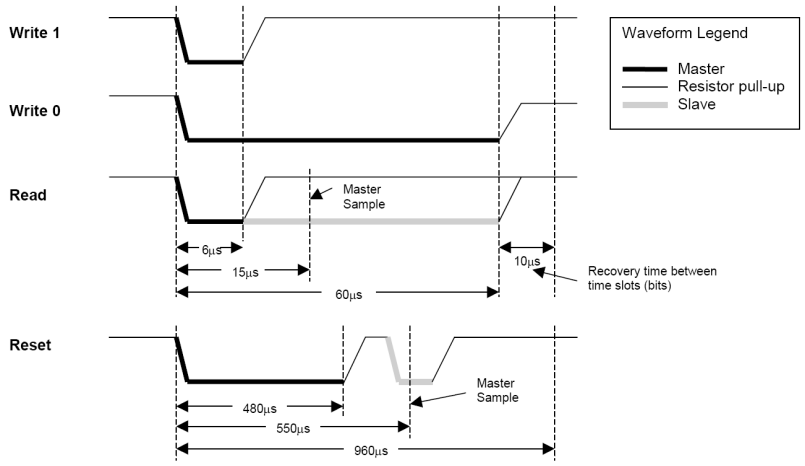


Figure 13--1-Wire waveforms. Note that the bus uses a weak pull-up, so letting the wire float high releases the bus. [22:2]

Once the Link-level methods were implemented, the rest of the Public Domain Kit methods were quickly added. Since memory was a major constraint, unnecessary methods were removed.

4.4.2 iButton Data Storage

A major design challenge involved transferring data from a 32-bit PC platform to the 16-bit MCU. Table 10 shows the different sizes of C primitive types on the two architectures used in the SmartPill system.

	TI MSP430x4xx	Intel 32-bit x86 (IA-32)
char	1 byte	1 byte
short	1 byte	2 bytes
int	2 bytes	4 bytes
long	4 bytes	8 bytes

Table 10--Primitive type sizes for the two different architectures used in the SmartPill system.

In order to eliminate incompatibilities between the two architectures, all types are broken into individual bytes. The data is then stored along byte boundaries in character arrays on the SmartPill cartridges. Each "file" described in section 4.4 is actually a character array. Note that both the MSP430 and IA-32 are little-endian architectures, so

no reordering at the bit level is necessary. General purpose methods were written that "pack" a variable number of bytes in an array starting at a given byte. Arrays are only packed on the IA-32 platform. These arrays are then read when an iButton is detected by the onboard software. Since the file formats were standardized in terms of bytes, the data could be unpacked from the arrays and then type-casted as the appropriate primitive type on the MCU. For example, four bytes (an int) are stored in a character array of length four on the IA-32 platform. That array is then unpacked on the MCU and the four bytes are type-casted as a long.

The byte-packing scheme also had another important benefit: efficient use of storage space. The initial storage design involved converting primitive types to strings and storing that data. In that scenario, one byte is used to store one alphanumeric character, thus wasting a significant amount of memory. Since each iButton only contains a few kilobytes of memory, a more efficient design was necessary.

4.4.3 Medication Dispensing

The medication dispensing algorithm utilizes the non-volatile memory on each SmartPill cartridge. If product information were always read into RAM, then that memory would be exhausted in some cases. The system would also need to rebuild its product database whenever the power went out. Consequently, a better solution is to maintain a pointer to the next dispense time in memory. Since the Dispense Times file in Table 7 is sorted into chronological order, the software simply checks each cartridge at a regular interval to see if dispensing should occur. That check occurs every minute in the proof-of-concept software. Although 1-Wire reads introduce time delays, performance is not critical in this system.

When the system detects that medication should be dispensed, it creates a dispense time structure, as shown in Table 11. The system maintains a buffer of dispatched dispense times. When the dispense buffer is not empty, the system sounds an audio alarm. The user must then depress the "Dispense" button on the medicine cabinet. When that button is depressed, the alarm ceases and the buffer is emptied, with product information printed to the dispenser's LCD.

Field	Size	Description
nat_drug_code	4 bytes	The medication's National Drug Code.
pills_to_disp	1 byte	The number of pills to be dispensed at this time.
options	2 bytes	Usage options from the Usage Options file corresponding to this dispense time.
dispatch_time	2 bytes	The time in minutes corresponding to when this struct was placed in the Dispense Buffer.
bottle_num	1 byte	This dispense time's corresponding bottle.

Table 11--Dispense time data structure.

4.4.4 Input/Output Specification

The software system is heavily I/O dependent. In dealing with geriatric patients, though, the number of inputs and outputs had to be minimized and simplified. A design was finalized that had only three major user inputs: cartridge insertion, a "Dispense" button, and clock set buttons. Any user capable of operating an alarm clock can use SmartPill. Table 12 lists the proof-of-concept's inputs and outputs.

Signal	Type	Action
Bottle inserted into system.	Input	• LED on that pill slot set to ON.
Bottle removed from system.	Input	• LED on that pill slot set to OFF.
Depress both min set and hour set buttons and hold for 10 seconds.	Input	• System resets to factory default state.
Depress hour set button	Input	• Hour component of current time incremented.
Depress min set button	Input	• Min component of current time incremented.
Dispense button depressed.	Input	• Alarm deactivated. • The medication slot number is printed to the LCD, followed by a five second pause, followed by the number of pills to be dispensed. This procedure occurs for each item in the Dispense Buffer. • The usage LEDs are illuminated according to that dispense time's usage options. This procedure occurs for each item in the Dispense Buffer.
Dispense Event	Output	• Alarm activated.
Pill Refill	Output	• LED on that medication's slot set to FLASH. • SmartPill cartridge is unlocked.
Power activated.	Input	• Power LED turns on.
Power deactivated.	Input	• Power LED turns off. • Contents of both the Dispense buffer and the Bottle Database are saved to non-volatile memory.
Product Conflict	Output	• LEDs on conflicting medications' slots set to FLASH. • Alarm activated.

Table 12--SmartPill system I/O specification.

5.0 Results

This section summarizes the proof-of-concept's implementation status as of March 23, 2004. Development is ongoing since the project deadline is not until May 2004. Most of the system features are partially implemented. To date, 5 of the 21 eXtreme Programming (XP) user stories have been implemented and tested (see Appendix A for a listing of these stories). Another seven stories are partially complete. The results can be summarized as follows:

- The Texas Instruments MSP430 microcontroller (MCU) detects the presence of SmartPill cartridges. A DS9092L reader facilitates the connection between the MCU and the cartridge's memory unit.
- The medicine cabinet can read product information stored on the connected bottle. Furthermore, the algorithms for storing this information on the cartridges have been implemented.
- The medicine cabinet successfully requests and reads the unique read-only memory (ROM) number of each cartridge over the 1-Wire bus.
- The MSP430's counters and crystal oscillator have been used to implement an accurate clock (see section 4.2.2).
- The LCD displays on both Olimex development boards function properly. The clock mentioned above has been displayed correctly on the displays.
- Additional hardware development work has been completed. The Olimex boards' buttons have been configured and tested. The buzzer on each board functions properly.
- The fast-prototyping tool (see section 4.1.5) completed the medicine cabinet shell and several SmartPill cartridges.

Presently, development is behind schedule. Two major issues have delayed the group's progress. First, iButton software development has taken longer than expected. When compiled, the Public Domain Kit's code size exceeded the capacity of the MSP430-413STK board. As a result, the group could not continue until the MSP430-449STK board arrived. Since most of the medicine cabinet's functionality depends on the input from the cartridges, this obstacle has impeded much of the software development effort. Second, Olimex shipped both boards with no documentation and the information

on their website was largely incorrect. Consequently, the group had to blindly "guess-and-check" its way through much of the low-level software development.

6.0 Conclusions

At present, the project results are inconclusive. Since development has been delayed and the project deadline has not passed, there are no artifacts to verify and validate. As a result, this section describes the future work necessary to implement the virtual product. It also lists some social and ethical considerations.

6.1 Future Work Recommendations

In order to move SmartPill to market, a full virtual product prototype must be implemented. The proof-of-concept demonstrated the SmartPill's novel features: medication cataloguing, dynamic usage scheduling, and harmful combination detection. The virtual product should show that the system as a whole can effectively combat prescription drug misuse. All tiers of functionality (see section 3.3) should be implemented. Most importantly, the mechanical pill dispensing apparatus should be designed and developed. When completed, the virtual product prototype should work "out-of-the-box."

Development of the SmartPill support infrastructure should also commence. Each pharmacy will need the mechanism depicted in Figure 1 to load product information onto SmartPill cartridges. This mechanism must have accompanying software that interfaces with both the pharmacist's customer database and a remote database containing SmartPill-formatted product information. Obviously, the latter database must also be developed.

SmartPill should be phased in gradually. This paper describes the final vision of the product, but such a product would require a significant organizational change. Insurance companies would need to decide whether to reimburse the cost of the

cartridges and/or the dispenser. The system, particularly the cartridges, might require government approval. Pharmacies would need SmartPill's support infrastructure. The cartridges would require new distribution channels. As a result, product iterations should incrementally add the more involved features. For example, one initial version of the product might read all product information from a Smart Card or from the Internet rather than from cartridges. As pharmacies embraced the product and started offering SmartPill cartridges, that functionality could be added. In any case, vendors must be convinced of the system's unique value for the product to gain acceptance.

Another possibility might involve licensing SmartPill's novel features to existing vendors. For example, e-pill has a full line of medication dispensers that would greatly benefit from SmartPill's electronic product information approach [17]. A patent should be obtained if this course of action is pursued.

6.2 Social and Ethical Considerations

The SmartPill project has largely positive social effects. It reflects the societal and cultural belief that human life has value. By organizing and managing a patient's usage schedule, SmartPill reduces the risks associated with the prescription drug use. In addition, it provides a "last line of defense" by detecting harmful medication combinations. With the plethora of new drugs introduced to the market each year, these features will become increasingly important. Finally, SmartPill benefits an impaired segment of society. This demographic's size will grow as the "Baby Boomer" generation ages.

7.0 References

[1] Bates, David W. and Atul A. Gawande. "Patient Safety: Improving Safety with Information Technology." *The New England Journal of Medicine* 348(25) (2003): 2526-2534.

[2] Consolidated Health Informatics Homepage. United States Government. 25 Oct. 2003 <http://www.whitehouse.gov/omb/egov/gtob/health_informatics.htm>.

[3] "Prescription Drugs: Abuse and Addiction." National Institute on Drug Abuse. 2 Apr. 2003 <http://www.nida.nih.gov/PDF/RRPrescription.pdf >.

[4] Alderman, Amy. Personal Interview. 3 Sept. 2003.

[5] Williams, Ronald D. Personal Interview. 10 Sept. 2003.

[6] "1-Wire Communication with a Microchip PICmicro Microcontroller." Application Note 2420. Dallas Semiconductor Corp.

[7] Powell, Harry. Personal Interview. 17 Sept. 2003.

[8] Lim, James. *Programmable Automatic Pill Dispenser*. 2003. U.S. Pat. 6,510,962.

[9] Lim, James. *Programmable Automatic Pill Dispenser with Pawl Indexing Mechanism*. 1999. U.S. Pat. 5,915,589.

[10] Shaw, Thomas J. *Automatic Pill Dispensing Apparatus*. 1997. U.S. Pat. 5,609,268.

[11] Agans, Rita M. *Medicine Reminder and Dispenser*. 1992. U.S. Pat. 5,159,581.

[12] Lewis, Kermit E. and Arthur S. Roberts, Jr. *Automatic Pill Dispenser and Method of Administering Medical Pills*. U.S. Pat. 4,573,606.

[13] Hubicki, Joseph T. *Automated System and Method for Dispensing Medications for Low Visions Elderly and Blind Individuals*. 2003. U.S. Pat. 6,615,107.

[14] Sagar, Richard Bryan. Koninklijke Philips Electronics N.V. *Bottle-Cap Medication Reminder and Overdose Safeguard*. 2003. U.S. Pat. 6,604,650.

[15] Picerno, Virginia L. *Pill Bottle and Dispensing Cap Combination*. 1996. U.S. Pat. 5,484,089.

[16] Talking-Rx. 2000 Millennium Compliance Corporation. 25 Oct. 2003 <http://www.talkingrx.com>.

[17] e-pill Medication Reminders. 2003 e-pill, LLC. 25 Oct. 2003 <http://www.epill.com>.

[18] Citro, Jeremy and Sharon Hermanson. "Assisted Living in the United States." March 1999. AARP. 17 Mar. 2004 <http://research.aarp.org/il/fs62r_assisted.html>.

[19] "Product Specification: A Prescription Drug Management System (SmartPill)." 2003 SmartPill. Document #R00002.

[20] Steinberg, Daniel H. and Daniel W. Palmer. *Extreme Software Engineering: A Hands-On Approach*. Upper Saddle River, NJ: Prentice Hall, 2004.

[21] Dallas Semiconductor iButton Site http://www.ibutton.com.

[22] "1-Wire Communication Through Software." Application Note 126. Dallas Semiconductor Corp.

[23] *MSP430x4xx Family User's Guide*. Revision C. 2003 Texas Instruments Inc.

[24] "1-Wire Software Resource Guide Device Description." Application Note 155. Dallas Semiconductor Corp.

[25] Beck, Kent. *eXtreme Programming Explained*. Reading, MA: Addison-Wesley, 2000.

[26] mspgcc. 23 Mar. 2004 <http://mspgcc.sourceforge.net/>.

[27] The Fedora Project. 2003 Red Hat, Inc. 23 Mar. 2004 <http://fedora.redhat.com/>.

[28] Olimex Ltd. 2004 OLIMEX Ltd. 23 Mar. 2004 <http://www.olimex.com/>.

[29] Downloads for MSP430 Ultra-Low Power Microcontrollers. 2004 Texas Instruments Inc. 23 Mar. 2004 <http://focus.ti.com/docs/toolsw/folders/print/msp430freetools.html>.

[30] "Detailed Software Design. A Prescription Drug Management System (SmartPill)." 2003 SmartPill. Document #R00010.

[31] Ulrich, Karl T. and Steven D. Eppinger. *Product Design and Development*. 2nd ed. Boston: McGraw-Hill, 2000.

[32] eproject. 2003 eProject Inc. 23 Mar. 2004 <http://www.eproject.com>.

[33] Concurrent Versions System. 2002 CollabNet, Inc. 23 Mar. 2004 <http://www.cvshome.org>.

[34] Mantis. 23 Mar. 2004 <http://www.mantisbt.org>.

[35] “Software Requirements. A Prescription Drug Management System (SmartPill).” 2003 SmartPill. Document #R00004.

[36] “Hardware Requirements. A Prescription Drug Management System (SmartPill).” 2003 SmartPill. Document #R00005.

[37] “Concept Selection. A Prescription Drug Management System (SmartPill).” 2003 SmartPill. Document #R00003.

[38] “Functional Test Procedure. A Prescription Drug Management System (SmartPill).” 2004 SmartPill. Document #D03.

[39] “Development and Test Schedule. A Prescription Drug Management System (SmartPill).” 2004 SmartPill. Document #D01.

[40] “MSP430x41x Mixed Signal Microcontroller.” Revision F. 2003 Texas Instruments Inc.

[41] “MSP430x43x, MSP430x44x Mixed Signal Microcontroller.” Revision C. 2003 Texas Instruments Inc.

[42] “mspgcc: A Port of the GNU Tools to the Texas Instruments MSP430 Microcontrollers.” 2003 mspgcc. 23 Mar. 2004. <http://prdownloads.sourceforge.net/mspgcc/mspgcc-manual-20031127.pdf?download>.

[43] *Book of iButton Standards*. Dallas Semiconductor Corp. 1 Dec. 2003 <http://pdfserv.maxim-ic.com/en/an/appibstd.pdf>.

[44] “Using the 1-Wire Public-Domain Kit.” White Paper #2. Dallas Semiconductor Corp.

[45] “1-Wire Search Algorithm.” Application Note 187. Dallas Semiconductor Corp.

Appendix A: XP User Stories

	User Story	Revision	Classification
1	When the user inserts a bottle into an open pill slot, the system detects that the bottle has been added.	1/26/2004	Hardware/Software
2	The system can read the information stored on the connected bottle.	1/26/2004	Hardware/Software
3	Each slot on the medicine cabinet is assigned a static number. **REVISION**: The 1-wire net assigns one iButton to be the master and additional buttons as slaves. Each button's unique ROM serial number is used to identify it on the bus.	1/26/2004	TBD
4	The system has the capability to poll slots on the medicine cabinet in order to take inventory of what bottles are available.	1/26/2004	Software
5	Using the information stored on each pill bottle, the system creates a usage schedule that integrates all currently connected medications. This action occurs dynamically with no user input other than inserting the bottle into the system.	1/26/2004	Software
6	The system updates the usage schedule during operation of the system. If a bottle is removed, that medication is removed from the schedule. If a bottle is added, that medication is added to the schedule.	1/26/2004	Software
7	Medications are organized in the usage schedule by dispense times on a 24-hour clock. When a dispense time is *active* (one or more medications should be dispensed), then the system activates the alarm.	1/26/2004	Hardware/Software
8	When the alarm is activated, it does not turn off until the dispense button is depressed by the user.	1/26/2004	Hardware/Software
9	When the user depresses the dispense button, the following actions occur: • The user's compliance is logged in the compliance database. An entry is added stating the dispense time, medications dispensed, and the time when the alarm was sounded. • For each medication, the medication and number of pills dispensed is written to the LCD screen (proof-of-concept).	1/26/2004	Hardware/Software
10	The system keeps a running count of the pills available in each bottle. When a medication is dispensed, the count is decremented.	1/26/2004	Software
11	When the running count of pills for a slot equals 10%	1/26/2004	Hardware/Software

	of the original amount, the system prompts the user to refill the medication flashing the LED on that pill slot. When the running count equals 0, the system unlocks the bottle.		
12	The system will have an LED on each pill slot. The system will be able to toggle three states for the LED: on, off, and flashing. In the flashing state, the LED will toggle between on and off every half second.	1/26/2004	Hardware
13	The system will maintain a compliance database that lists the time and number of pills for each time a medication is dispensed. This database can be downloaded via the system's serial port.	1/26/2004	Hardware/Software
14	When a medication is introduced into the system, the system will be able to detect harmful combinations between all currently connected medications. This feature involves comparing the list of connected medications to the list of harmful medications stored on each pill bottle.	1/26/2004	Software
15	The LED on each pill slot operates as follows: • OFF—Slot not occupied. • ON—Slot occupied and functioning properly. • FLASHING—Refill necessary or product conflict. This function should be implemented in software.	1/26/2004	Hardware/Software
16	When a bottle is introduced into the system, that bottle is locked in place mechanically by the system. Only the system can release the bottle from the slot (proof-of-concept). Bottle release only occurs when the medication is empty.	1/26/2004	Hardware
17	The system shall maintain an onboard clock. The time can be retrieved from this clock, the clock can be set, and the clock may be tested to ensure correct operation.	1/26/2004	Hardware/Software
18	The system shall have an LCD screen capable of printing text. For the proof-of-concept, this screen will show the type of medication and number of pills dispensed. When pills are not being dispensed, the screen shows the current time. For the virtual product, the only function of this screen is to show the current time.	1/26/2004	Hardware/Software
19	For each currently connected bottle, the system shall maintain product updates for that bottle. These updates are retrieved from other bottles connected to the system. The product update storage space for each bottle is treated by the system as if it is actually on the bottle itself.	1/26/2004	Software

20	A 12-hour clock will be displayed on the LCD screen when the system is not dispensing medications. This clock has a PM indicator. The clock can be set using two buttons: an hour set and a min set. When the hour set button is depressed, the hour is incremented by 1. When the min set button is depressed, the minute is incremented by 1. When the minute component is 59 and the min set button is depressed, the hour component is NOT incremented.	2/24/2004	Hardware/Software
21	The system has four usage LEDs: "With Food," "With Water," "With Sleep," and "No Alcohol."	2/24/2004	Hardware
22	Each pill slot has some mechanism (like a pressure sensor) by which the microcontroller can detect that a cartridge has been added to that slot. This feature is necessary because the slot cannot be determined using the 1-wire net alone.	2/24/2004	Hardware
23	If the system is turned off, the contents of the Dispense Buffer and the Bottle List should be stored to non-volatile memory. The system should check non-volatile memory when the power is restored to see if there is data there. If so, the system should verify the data by checking it against currently connected SmartPill cartridges before loading that data.	2/24/2004	Software
24	If the hour set and min set buttons are held down for 10 consecutive seconds, the system resets to the factory default state (see Detailed Software Design).	2/24/2004	Software
25	The system should be powered by a standard 120V AC wall socket. The system should also have use two AA batteries as a backup power supply that is used when the power cord is not connected to the 120V AC source.	2/24/2004	Hardware

Appendix B: ENGR302 Project Proposal

A Pharmaceutical Prescription, Distribution, and Self-Organizing System[1]

Spence Green
University of Virginia, SEAS
wsg6f@virginia.edu

Clement Song
University of Virginia, SEAS
cs2bs@virginia.edu

1. Abstract

In a recent report by the National Institute on Drug Abuse (NIDA), prescription drug misuse by elderly patients was listed as a major concern in the healthcare industry:

> The misuse of prescription drugs may be the most common form of drug abuse among the elderly. Elderly persons use prescription medications approximately three times as frequently as the general population and have been found to have the poorest rates of compliance with directions for taking a medication (Prescription Drugs 6).

Clearly there is a need for to improve and simplify the prescription, distribution, and use of medications. We propose a medicine **Prescription**, **Distribution**, and **Self-organizing** (PDS) system that will authenticate all prescriptions, eliminate incorrect or fraudulent distribution, and decrease the risk to the patient. Such a system would save lives every year.

Currently, a patient receives a paper prescription from a medical professional and takes the prescription to a pharmacy. The pharmacist will then distribute the medication with literature on the particular drug. The first problem with this system is the transaction method, which makes it relatively easy to obtain prescription drugs fraudulently. The NIDA also reported an alarming trend:

> In 1999, an estimated 4 million people—almost 2 percent of the population aged 12 and older—were currently (use in past month) using certain prescription drugs non-medically . . . In addition, health care professionals—including physicians, nurses, pharmacists, dentists, anesthesiologists, and veterinarians—may be at increased risk of prescription drug abuse because of ease of access, as well as their ability to self-prescribe drugs (Prescription Drugs 4).

Our system will secure the prescription system by electronically encrypting prescription information on smart cards and authenticating all transactions.

The other major problem that our system addresses is drug misuse. By placing wireless sensors on medication bottles, patients can physically place bottles together to determine the safety of use. The wireless sensors will automatically detect any dangerous combinations and alert the user. In addition, inexpensive LCD displays will be available to access product information, which is also stored in the sensor package.

[1] The proposal's original formatting is preserved in this Appendix.

Figure 1—PDS conceptual system outline.

2. Problem Description

a. Definition

This product addresses two main problems in the areas of drug prescription, distribution, and consumption:

1) Prescription fraud—Paper prescriptions with the doctor's name, contact information, and drug information are used for drug prescription today. Paper prescriptions are easy to forge, though. Our solution increases the complexity of fraudulently obtaining drugs by several orders of magnitude. By authenticating doctor and patient information and electronically encrypting the actual prescription, our solution secures the prescription process.
2) Decrease improper drug use—Many patients use multiple medications simultaneously. Product literature usually runs several pages per drug. In addition, this information is often difficult to decipher. By storing all of the usage information on the bottle and ensuring proper use via "smart" sensor networks, we decrease the patient's risk.

b. Target Customer

Our target customers fall into three main categories:

1) Doctors and hospitals—These customers will need the drug prescription system components (see section 5.1).
2) Pharmacies—Drug distributors will need the technology to decrypt and authenticate the prescriptions. In addition, they will need a machine that can load drug information onto each pill bottle and enable the sensor device (see section 5.2).
3) Patients and insurance companies—Any individual that visits a healthcare professional will need a smart card that carries prescription information. The doctor could distribute this card or the patient could purchase a reusable card. The total cost of this system is passed down to the patient (and their insurance provider), so the patient also pays for the rest of the system when he purchases a drug.

3. Problem Context and Significance

This system saves lives. Every year, patients use drugs in non-prescribed combinations or for non-medical purposes. Although we can never prevent patients from abusing legally obtained drugs, we can greatly reduce an individual's risk in using medications through the use of "smart" technology.

3.1 Technical

There are a myriad of technical challenges facing this system. We must solve problems in the following areas:

- Security/reliability—This is the greatest technical challenge our system faces. We must safeguard each pill bottle from tampering. Liability is always a major issue in healthcare, so we must create a system that will be reliable *almost without exception*.
- Miniature sensor devices—Since the sensor package must fit on a pill bottle, there are extreme size constraints. The sensor system should not noticeably increase either the size or weight of the average pill bottle.
- Power consumption—The sensor package will need a power supply that will last for months. Since size is a major constraint, we must limit the overall energy consumption of the sensor package.
- Context-aware computing—Each sensor must have the capability to dynamically "discover" proximate pill bottles.
- Ad hoc networking—When multiple pill bottles come in close proximity to one another, an ad hoc network is formed between the sensor packages on each bottle. Facilitating these interconnections seamlessly is a major project design challenge.
- Wireless connectivity—We will equip each sensor package with an RF transmitter. We must select a small and energy efficient transmitter. The RF protocol must also allow us to perform ad hoc networking.
- Embedded real time systems—We will need to do some processing (data comparison) on the pill bottle which will require a small, embedded real-time system. Real-time systems have strict constraints on operation, so we must balance computing power (which implies a physically larger system) with size.
- Packaging—We will need to package the embedded system so that it will fit onto a standard pill bottle.

3.2 Organizational

We have identified several critical organizational aspects of this project. First, the largest challenge PDS faces is in the legal arena. Any medical device must win government approval before it can even be marketed. We must carefully research government standards for this type of product. We will need legal advisors to help us follow all regulations. In addition, we need legal experts to draft disclaimers and all other necessary documentation to limit our liability. In short, this project cannot succeed unless we follow all legal regulations.

The second organizational aspect is the drastic impact this product will have on the healthcare industry. Pharmaceutical companies will need to reformat their product literature to fit on our devices (we could provide this service, though). Institutions such as medical schools will need to train doctors about the proper use of PDS. Practicing medical professionals (including pharmacists) will also need proper training. Each patient will need to understand the operation of the system from an end-user perspective. A stated goal of this system is to move all complexity away from the patient, so this type of user should not need formal training.

The final organizational aspect of this system is infrastructure. This is a revolutionary system and there is not a single pharmacy or medical office in the world (that we know) that supports anything like it. Once the system is developed, we must facilitate deployment on a scale that will rapidly make the system ubiquitous. This is a major corporate distribution problem.

Clearly there are major organizational challenges for this system. We believe that these challenges are more significant than the technical problems facing the system. We are confident, however, that none of these obstacles are insurmountable.

3.3 Cultural

Clearly the reduction of health risk and the preservation of life are PDS's positive cultural aspects. The largest cultural challenge we face is acceptance. Elderly patients usually take more medications than younger people. Unfortunately, this segment of the population is also the most reluctant to embrace new technologies. We must take great care to emphasize the safety and reliability of this system. In addition, we must emphasize the benefits of PDS since it adds cost to medications. Prescription drugs are already breathtakingly expensive, so we must make both the patient and the patient's insurance provider believe that the extra dollar they spend is worth it.

4. Team requirements

We need personnel from several disciplines to complete this project. We have divided the project team into a Board of Advisors, who will provide expert support, and an Engineering Team, which will implement the physical system.

Board of Advisors

- Legal counsel (1)—Since we must deal with government rules and regulations, we need legal assistance during development. We also need a strategy for limiting our liability as developers and vendors of the system.
- Marketing (1)—We need marketing and business research to understand the user requirements and the best way to sell this system.
- Technical Advisor (1)—Professor Ronald D. Williams, Associate Professor of Electrical and Computer Engineering, has offered to advise us on the technical aspects of this project. He has extensive experience in building embedded systems including some embedded medical devices.

Engineering Team

- EE/CpE (1)—We need an individual to design and implement the sensor package for the pill bottles.
- CS/CpE (1)—Simulation software for the prototype system must be developed (see section 6). We need an engineer with experience in computer simulations and graphical interfaces.
- ME (1)—The physical pill bottle, as well as an attachment mechanism for the sensor package, must be developed by a mechanical engineer with experience in the area of materials science.

With three diligent engineers and three knowledgeable advisors, we believe this system can be developed successfully in one year.

5. Design Strategy

There are three components in this system: the prescription system, the distribution system, and the self-organizing pill bottle network system. Different design strategies and concerns exist for each of these components.

5.1 Prescription System

This system will allow doctors to digitally prescribe medications for patients. The customers can then take the digital prescriptions on a smart card to the pharmacist. Finally, the pharmacist will deliver the prescribed medicine to the patient.

Design Objectives:

The doctor will make the prescription on his computer or PDA. A smart card reader will then flash the encrypted electronic prescription onto a smart card. Each doctor will have a unique identification tag for authentication when the pharmacist decrypts the prescription. The integrity of the digital prescription must be guaranteed and immune from attacks.

Involved Disciplines:

Electrical Engineering, Computer Science

Design Procedure:

We will develop a custom prescription application that utilizes commercial encryption technology. We will utilize standard software engineering practices such as risk analysis, requirements analysis, and system specification to develop the software artifact. We will also purchase a smart card and smart card reader that plugs into a PC or PDA.

Design concerns:

Which authentication method and encryption technique should we use? What information needs to be stored on the customer's card? How will we assign unique identification codes to each doctor?

5.2 Distribution System

Pharmacists will utilize this component. Upon receiving the prescription from a customer, the pharmacist can decrypt the prescription data and authenticate both the doctor and patient information. After transaction authentication, drug product information is transferred onto a pill bottle and the medicine is distributed to the customer.

Design Objectives:

The pharmacist will be able to verify that the prescription is for a particular customer. The entire process should be largely automated. The software system must be reliable and secure, as this is a mission-critical application. The imprinting process should also be authenticated and secured.

Involved Discipline:

Mechanical Engineering, Electrical Engineering, Computer Science

Design Procedure:

We will develop a custom authentication and distribution application that utilizes strong encryption technology. We will utilize standard software engineering practices such as risk analysis, requirements analysis, and system specification to develop the software artifact. We will also utilize a commercially available smart card and smart card reader. We will not create an independent machine to imprint the pill bottles. Instead, we will simulate this data transfer by "flashing" the memory hardware on the sensor package.

Design concerns:

What information will we imprint on the pill bottle? What format will this data be stored in?

5.3 Self-Organizing Pill Bottle Network

Pill bottles contain information about the medicine, instructions, and precautions associated with the drug. When a pill bottle is placed among other pill bottles, it will communicate with the rest of the pill bottles and check if any precautions have been violated, such as incompatible drugs combinations. In the event of a violation, the system will alert the patient (initially via LEDs). In addition, the user can view a simple LCD screen to obtain product information (the inexpensive LCD will be sold independently).

Design Objectives:

Each pill bottle will contain a durable, embedded sensor package to store product information and facilitate ad hoc networking. This device must satisfy strict cost, power, and profile constraints. The networking of all the pill bottles should happen seamlessly. All pill bottles owned by an individual user should carry a shared family identification code. In addition, the user should be able to view product information on an inexpensive LCD screen. All functionality should be intuitive and simple to use. Finally, the information on the pill bottles should be secure and preserve the privacy of the owner.

Design Procedure:

Development will center on a sensor package consisting of a processor, memory, and short-range RF transceiver. We will also implement a service discovery protocol to identify other devices. Finally, we will develop an attachment mechanism for the sensor package.

Design Concerns:

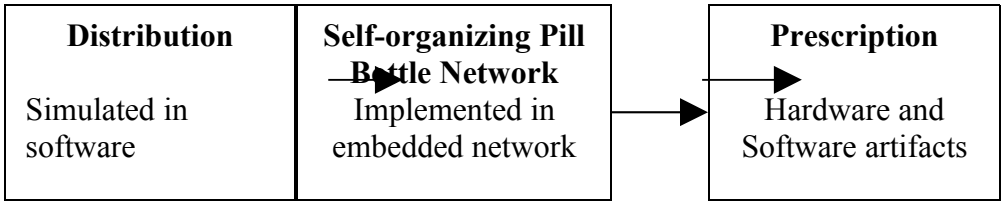
What is an acceptable lifetime for the sensor? How will we satisfy the low-profile requirements? How much memory will we need? How much computational power will we need? What are some other methods to signal the users when violation occurs? Do we lock the cap of the bottle when an alarm is triggered?

Involved Discipline:

Electrical Engineering, Computer Science

6. **Prototype Product Vision**

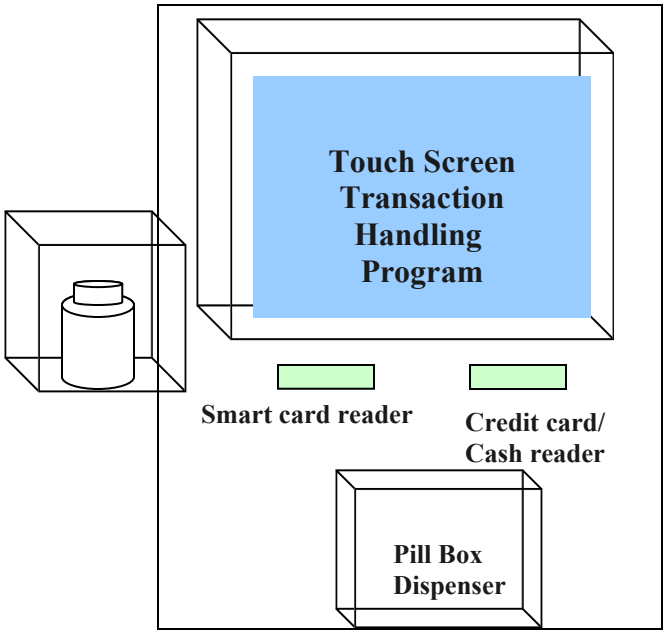
Figure 2—



The prototype PDS system that will be developed as proof-of-concept.

For this stage of the project, we will create the medical prescription software. Next, we will transfer the electronic prescription to a smart card. The distribution software will then simulate both the distribution of the medication and imprinting of the pill bottle. Finally, a physical wireless embedded network will be implemented to handle violation checking and signaling.

7. **Future Possibilities**



If our product becomes successful, we could eventually see a fully automated Medicine Dispensing/Refill Machine. A person could walk up to the PDS machine and insert a smartcard, which would have the prescription information on it, into the smartcard reader. With the help of some other authentication method, such as a password, this person could then see the prescription that has been stored on their card. If the patient wanted to purchase the prescription, then he could insert a credit card or cash into the machine. The machine would automatically fill the prescription and imprint the bottle with all the product data. The assembled pill bottle would then be dispensed from the Pill Box Dispenser.

8. Proposed Budget

Phase 1: System prototype budget
Goals: Use available resources from the department to build an embedded network with three devices.

7.1.1.1 Item	Quantity	Cost
Off-the-shelf printed circuit board	3	$50
Low power RF link	3	$50
LCD display	1	$50
Programmable Logic Device (PLD)	3	$300
IP module license	/	$400
Smartcard	1	Donation?
Total:	/	$1000

Funding Source: Lockheed Martin

Phase 2: Full System prototype budget

Goals: Use professional development tools to implement the whole system

Item	Quantity	Cost
Smartcard development kit	1	$6000
Embedded processor development kit	1	$10,000
Printed Circuit Board Design package	1	$5,000
Low power RF link	3	$50
LCD display	1	$50
Miscellaneous cables and tools	/	$200
Total:	/	$21300

Funding Source: National Collegiate Inventors and Innovators Alliance (NCIIA)

9. Verification

Goals:

1. Document the development and management processes.
2. Develop multidisciplinary design strategy.
3. Produce requirements analysis, system specification, and source code for the software components of the system.
4. Implement the embedded network system, along with all the embedded software.
5. Produce simulation tools to test the system software and hardware

Verification Procedure:

Step 1: Verify that all the critical development processes are sufficiently documented and all the software and hardware artifacts are built.

Step 2: Verify that members of the multidisciplinary team have adequately contributed to the development process.

Step 3: Run the software and hardware through simulation and verify the system behavior complies with the specification.

Step 4: Test the system for compliance with government standards and regulations.

Step 5: Perform extensive usability testing with a broad range of patients, focusing on elderly people.

10. Conclusion

We firmly believe that the hardware and software artifacts for this project can be developed in one year. Professor Williams will serve actively on our advisory board. There are various intellectual property opportunities in this project and we will pursue patents on several aspects of this system. We believe PDS addresses a growing problem in our society today and we are excited about the future of this project.

11. References

"Prescription Drugs: Abuse and Addiction." National Institute on Drug Abuse. http://www.nida.nih.gov/PDF/RRPrescription.pdf <2 Apr. 2003>

Smart card Technology. http://www.raisonance.com/files/pdf/SmartCardPriceEuro.PDF <2 Apr. 2003>

12. Appendix (Resumes)

Appendix C: Project Budget

Smart Pill Budget						
Updated: 2/10/2004						
Item	**Received?**	**Purchase Date**	**Description**	**P/N**	**Supplier**	**Quantity**
1	X	11/5/2003	Motorola Processor	68HC908LK24	Newark InOne	1
2	X	11/5/2003	TI Processor	MSP430F449	DigiKey	1
3			LED array	276-1622	RadioShack	1
4			Buzzer	273-066	RadioShack	1
5			Momentary Pushbutton	275-1548	RadioShack	1
6	X	10/31/2003	External Memory (iButton)			
7	X	10/31/2003	iButton Port	DS9092R	Dallas Semi.	1
8	X	10/31/2003	Password-Protected 32kB EEPROM Bu	DS1977	Dallas Semi.	4
9	X	10/31/2003	iButton Plastic Card Mounts	DS9093RA	Dallas Semi.	1
10	X	10/31/2003	iButton Plastic Card Mounts	DS9093RB	Dallas Semi.	1
11	X	10/31/2003	64 kB Memory iButton	DS1996L	Dallas Semi.	2
12	X	10/31/2003	iButton Halo	DS9106S-GN	Dallas Semi.	1
13	X	11/12/2003	iButton Blue Dot Receptor	DS1402D-DR8	Dallas Semi.	1
14	X	11/12/2003	iButton Hand-Held Probe	DS9092GT	Dallas Semi.	1
15	X	11/13/2003	Universal 1-Wire COM Port Adapter	DS9097U-S09	Dallas Semi.	1
16	X	12/7/2003	JTAG Programming Board	MSP430-413STK	Olimex	1
17	X	12/7/2003	JTAG Board Cable	MSP430-JTAG	Olimex	1
18	N/A	12/7/2003	JTAG Shipping Charge	N/A	Olimex	1
19			Battery Backup	TBD		1
20			AC power (AC/DC converter)	TBD		1
21	X	1/16/2004	Mastering AutoCAD 2004 (book)	N/A	Amazon.com	1
22	X	1/19/2004	IBM Thinkpad	N/A	retrobox.com	1
23	X	1/19/2004	Wireless networking card	8410-WD	Computers4SUR	1

Total:

Funds

Available Funds: