A SYSTEM THAT IMPLEMENTS AUTOMATIC MEDICATION MANAGEMENT AND PASSIVE REMOTE MONITORING TO ENABLE INDEPENDENT LIVING OF HEALTHCARE PATIENTS

by

Corey McCall

A thesis submitted in partial fulfillment of the requirements
for the Honors in the Major Program in Computer Engineering
in the College of Engineering and Computer Science
and in the Burnett Honors College
at the University of Central Florida
Orlando, Florida

Fall Term 2010

Thesis Chair: Dr. Cliff C. Zou

ABSTRACT

Many healthcare patients are required to enroll in assisted living facilities, because they are not able to manage their complex medication regimens without the active assistance of a caregiver. This restricts their ability to live independently, and places a considerable burden on the healthcare system. This thesis describes the development of a system that implements automatic medication management and passive remote monitoring for outpatients. The goal of the system is to enable patients to live independently, while reducing medication noncompliance. The resulting prototype is a device that performs two essential functions: (1) to provide notifications and assistance to the patient when medication is to be taken, and (2) to provide passive monitoring of the patient's compliance to a remote caregiver.

ACKNOWLEDGMENTS

I would like to thank:

My advisor, Dr. Zou, for the opportunity to work on this project and to represent us at the IEEE Engineering in Medicine and Biology Conference in Argentina.

The rest of my thesis committee, Dr. Gonzalez, for guiding me through engineering research, and Dr. Ford, for teaching me the value of entrepreneurship and innovation.

My research partner, Branden Maynes, for supporting me and keeping us on track through the rigorous prototyping process.

My family, Terry McCall, Debbie McCall, and Chelsea McCall, and my girlfriend, Naomi Cowie, for their support and encouragement that empowered me to complete this project, and strive for a career in research.

My sponsors, the National Science Foundation, the National Collegiate Inventors and Innovators Alliance, and the Burnett Honors College, for funding this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
GPIO	General Purpose Input / Output
HIS	Hospital Information System
I/O	Input / Output
LCD	Liquid Crystal Display
LED	Light Emitting Diode
RFID	Radio Frequency Identification
RX	Receive
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
TX	Transmit
UART	Universal Asynchronous Receiver / Transmitter
UHF	Ultra High Frequency
UID	Unique Identifier
USB	Universal Serial Bus
V	Volts

CHAPTER 1: INTRODUCTION AND BACKGROUND

A large percentage of healthcare patients fail to comply with their prescribed medication schedules. This can result in hospital and nursing home admissions, serious injury, or death. It is difficult for some patients to adhere to a complex medication regimen, because it may be too complex, or the patient may be forgetful. The research presented in this thesis attempts to solve this important issue by developing an intelligent medication monitoring and notification system that can enable patients to follow prescribed medication schedules with minimal effort. This should enhance the lives of patients, and enable independent living.

The research presented is essentially an extension of research done by Intel's Proactive Health lab, published in 2003 and 2008 (described in [1] and [2] respectively). These papers describe a system that tracks a patient's medicine supply using Radio Frequency Identification (RFID) technology and a scale. When new medicine is received, the patient assigns a uniquely identifiable RFID tag to its container, and inputs the corresponding schedule information into the system. The medicine storage device is then able to notify the patient when a particular medicine is to be taken. The scale is used to check that the patient actually takes the medicine.

This system succeeds in providing a reminder to forgetful patients. However, it requires the user to manually assign and attach RFID tags to the medicine containers, and manually input schedule information whenever a new medicine is added. This system also lacks the ability to dispense individual medications, and to alert a patient's caregiver if medicine dosages are not taken correctly. Without these features, the most critical interactions with the system would likely have to be done by an able caregiver. This restricts use of the system to a traditional healthcare environment with attentive caregivers.

As an alternative to manual entry, the system developed in this thesis relies on prescription schedule information encoded on an RFID tag and attached at the patient's pharmacy. This information will be readily available, considering it is a subset of the information concurrently printed on the label. This allows the system to be easily configured by the patient by simply placing the new medicine container on the device. This system also implements the ability to dispense individual medicine containers using a novel motorized storage tray, eliminating the need for the patient to manually choose the correct medicine to take. And finally, this system implements the ability for a caregiver to passively monitor patient compliance from a remote location using a cellphone. These critical additions, combined with a more user friendly design, allow the revised system to be used by patients while living independently.

Overall, this project fills in many of the gaps of previous work, and provides a practical means of in-home medication monitoring and notification using current technology. The first part of this thesis (Chapters 2 and 3) describes background research, and the resulting theory and specifications that define the revised system. The second part (Chapters 4 and 5) describes the prototyped implementation of the revised system. The final part (Chapters 6 and 7) evaluates the prototype, and suggests further work to be done.

1.1 Background and Motivation

The research presented is done in response to two general problems in the healthcare system. The first is the inability of patients and healthcare providers to perfectly match medicine type, amount, and schedule with a patient’s requirement. Because of this, many patients are not taking their medicines as directed, leading to unnecessary disease progression, complications, lower quality of life, and mortality [3]. The second is the impending stress on the healthcare system. This stress is described in [4], by emphasizing that the growing ratio of patients to workers caused by the baby boomer population mushroom will likely cause the current healthcare system to fail within the next ten years.

Many patients would prefer to live independently. This would allow them to have more freedom, and relieve stress on the healthcare system [5]. However, many elderly patients are not able to do this, simply because they are not capable of following their own medication regimen. This is because it may be too complex, or the patient may be forgetful. Currently, the patient’s only option is admission to an assisted living facility, such as a hospital or nursing home, or requiring the attentive care of a volunteer caregiver [5].

Research presented in [6] and [7] discusses passive remote monitoring systems as an alternative to active caregiver assistance. The systems described focus on the use of monitoring sensors that only alert caregivers when their attention is required. This allows caregiver resources to be scheduled much more efficiently, reducing the patient’s burden on the healthcare system.

The development of a system that incorporates automatic medication management and passive remote monitoring to allow healthcare patients to independently manage their personal medications helps to solve these problems. The research presented includes the development of such a system. In this system, the patient’s medicine cabinet is replaced with a computerized storage device that is capable of providing notifications and assistance to the patient when medication is to be taken, and providing passive monitoring of the patient’s compliance to a remote caregiver.

1.2 Thesis Objective

This thesis aims to provide a solution to the previous mentioned problems by developing a revised system based on previous research described in [1] and [2]. The revised system implements automatic medication management and passive remote monitoring to manage complex medication schedules for outpatients. This should prolong independent living, eliminate medication noncompliance, and reduce the burden on the healthcare system. The following contributions are made: (1) A system is developed that allows healthcare patients to comply with complex medication regimens without the active assistance of a caregiver, (2) The prototype device that is built to implement the system is designed in such a way that it is marketable to anyone who takes medicine on a regular predefined schedule, and (3) All hardware and software design components associated with the device, including source code, are published so that the device can be reproduced and further development can be done.

1.3 Thesis Roadmap

This thesis proceeds as follows. Chapter 2 contains a literature review of relevant previous work. Chapter 3 describes the approach that is the basis of the research presented. Chapter 4 describes the design and construction of the prototype hardware and the associated costs. Chapter 5 describes the design of the prototype software and how the user interacts with it. Chapter 6 is an evaluation of the prototype. Chapter Seven concludes the thesis with a summary, an assessment of the completed project, and a list of further work to be considered. Three appendices are also included that cover the hardware schematics, a detailed budget for the prototype, and examples of the RFID tag encoding protocol to be implemented at pharmacies. Additionally, the full source code for the resultant prototype device is listed in [8].

CHAPTER 2: LITERATURE REVIEW

This chapter contains a literature review of relevant previous work. The first section, In-home Healthcare, describes the growing need for in-home healthcare. Section two, RFID in healthcare, describes the current state of RFID technology in the healthcare system. Section 3, In-home Medication Administration Devices, describes the current state of home-based medication devices. The final section is a summary that focuses on the gaps in previous work.

2.1 In-home Healthcare

The growing need for in-home healthcare is best described in [4]. This article describes the need to scale the healthcare system by developing technology that will promote "aging in place," the concept of allowing elderly healthcare patients to live at home during treatment. Providing healthcare to baby boomers is a "time bomb" that could cripple the healthcare industry in the near future. This is because it will be impossible to treat the high ratio of patients to workers using the current infrastructure-based healthcare paradigm. The article also describes the benefits of implementing more in-home technologies today. These include a reduction in cost compared to hospital-based healthcare and a reduction in the burden to caregivers.

Additional support for the higher efficiency of in-home care is found in [8]. This paper discusses how advances in wireless and other communications technologies have allowed elderly healthcare patients to receive personal treatment by caregivers and healthcare providers without requiring face to face attention. The resulting experiment trials conclude that the implementation of remote communication technology in healthcare monitoring allows patients to receive more attentive care at a reduced cost. For example, it is much easier for a caregiver to check on a patient by participating in a teleconference than by physically visiting their residence. This enables patients to experience the benefits of independent living while receiving the attentive care of a more traditional healthcare environment.

The reduction of healthcare cost due to passive remote monitoring is exemplified in [1] and [6]. These papers discuss the value of surveillance systems for in-home healthcare patients, specifically those with dementia. This value is derived from an increase in patient safety and autonomy, and a reduction in the burden to caregivers [8]. The surveillance systems described consists of multiple sensors that passively record the condition of the patients. The data is used to detect pathological behavior patterns, and warn caregivers of emergencies, such as in the case of a fallen or wandering patient. Being able to remotely and passively monitor patients allows caregivers to use their time more efficiently, reducing healthcare cost.

2.2 RFID in Healthcare

RFID is a technology that allows objects to be “tagged” with a small microchip that can be wirelessly identified by an associated reading device. The tags are small, inexpensive, and do not require a power source. When the reader is activated within a close proximity of the tag, the tag’s antenna harvests enough energy from reader’s transmission to respond with a signal indicating its unique identifier (UID). Some more advanced RFID tags can also store a small amount of data that can be accessed in the same way.

RFID has been successfully integrated into the healthcare system for several applications. The first of which is patient tracking, as described in [7]. This paper describes the Alzheimer Multi-Agent System, a system designed to intelligently create optimized working schedules for nurses in geriatric residences. The schedules are computed by observing patient location and incident history, and applying a case-based planning reasoning algorithm. Patient location information is collected by short-range RFID readers placed above doorways. These sensors scan patient RFID bracelets whenever they move from one room to another within the residence.

RFID is also used to accurately access medical information records in a hospital environment. This implementation is described in [9] and [10], in which RFID technology is used in conjunction with barcode technology to match patients to medicines. When a medicine dosage is ordered by a doctor through the Hospital Information System (HIS), the pharmacy automatically prints a barcode on the package. The barcode value is then matched to a patient identification value in the HIS database. When the nurse picks up the medicine and brings it to the patient, she scans the patient's RFID bracelet containing his identification value. The patient and medicine identification are then checked by the in-room computer and either a match or mismatch notification is provided. This system is meant to be a "second guard" to ensure that patients are given the correct medicine, while reducing costs compared to a completely RFID-based system.

It is important to note that RFID has not been completely adopted by the healthcare system for several reasons described in [12]. First, RFID tags are still too expensive and will not become as pervasive as barcodes until the cost becomes comparable. Second, RFID tags are not completely reliable. Current state of the art readers have a read rate of about 99.5%, which is not acceptable for many healthcare applications. Third, it is still unknown how the small amount of radiation associated with

RFID technology will affect patients, especially those with pacemakers or other sensitive devices. Another issue with current RFID technology, mentioned in [2], is that since RFID still under development, it cannot be fully standardized and may be difficult to scale over time.

2.3 In-home Medication Administration Devices

Several in-home medication administration devices have been developed by researchers and private companies. The first of these devices is described in [1], and shown in Figure 2.1. This report describes a system designed to monitor the medication regimens of forgetful healthcare patients. A patient initializes a medication bottle into the system by placing it on a platform with an integrated scale, and manually inputting the medicine's schedule information. The system then reads in the RFID tag placed on the bottle to identify it. The system can then notify the patient when the medicine is to be taken and show relevant dosage information on a graphical display. It can also detect whether or not the medicine has been taken by calculating the weight of the medicine inventory on the scale. According to the corresponding "Related Work" section, this is not the first system to enable smart medication monitoring in the home, but it is the first to do so in a practical way.

In [2], the system described in [1] is further developed by the same laboratory (shown in Figure 2.2). This paper describes the use of ultra high frequency (UHF) RFID and wireless sensor motes to add patient identification features, and allow the system to be expanded. When medicine is to be taken, the system checks to see if the patient is in the vicinity of the reader by polling a long range, UHF RFID tag worn by the patient. If the patient is present, a notification buzzer is activated and the patient is notified to take the medicine. The system is expandable because each component (medicine tray, base station, and patient presence sensor) communicates wirelessly, so new components can be added without any change in hardware, and only minor change in software.

These two systems were built to show off the full capabilities of a new wireless mote technology developed by Intel. Consequently, the systems are very complicated to operate, and are not useful to regular patients. Both systems are largely based on wirelessly communicating components, one of which depends on immature UHF RFID technology. The added complexity of a system architecture based on physically separated wireless mote nodes is not worth the benefit of potential expandability. If the device is to be expandable (such as for the purpose of adding more medicines than a single unit can hold), a duplicate, self-sufficient device could be added and the wireless mote technology could be implemented to centralize the medicine database. In order to make the device practical, simplicity should be incorporated over expandability and unnecessary features (i.e. patient proximity detection).

Another problem with these two systems is that they require a patient to put all medicine bottles on top of a scale in order for the system to measure the total weight of the medicine inventory. This makes the system unsuitable for patients with many medicines, and requires an expensive high-capacity scale. In addition, it requires the patient to manually pick up the right bottle from the cluster of bottles on top of the scale. This choice can be extremely dangerous to a patient who has poor vision or memory.

Another system, described in [12], and pictured in Figure 2.3, is an FDA approved, in-home medication dispenser developed by INRange Systems Inc. It can automatically dispense a given dosage of pills, and alert the patient when it is time to take a medicine. The user interacts with the device via an integrated touch screen computer, and the patient's doctor can remotely reconfigure it using a web-based application. This product, called EMMA, is currently for sale to customers through participating pharmacies.

Unfortunately, EMMA has two major problems. First, all pills need to be individually packed into a special-made cartridge. This can only be done by medicine manufacturers, and not pharmacies. This greatly increases medication cost, and limits its adoption to participating manufacturers. Second, it is only applicable for pill format medicines, and cannot be used with liquids or any other less common medicine formats. It is important that an in-home medication administration solution be compatible with all of the patient's medications. Otherwise, a patient will likely forget about medications that are not automatically managed.

Figure 2.1: An elementary in-home medication administration device[1]

Figure 2.2: A further developed in-home medication administration device[2]

Figure 2.3: An in-home medication administration device by INRange [3]

---

[1] Source: [1]

[2] Source: [2]

[3] Source: [12]

2.4 Summary

The papers in this chapter describe the growing need for in-home healthcare devices, and the current application of RFID and remote monitoring technology in the healthcare industry. They also show that, considering the similar research that has been done, the research presented contributes valuable new knowledge to the field.

The growing need for in-home healthcare devices is best described in [4] as the population growth of retirement-age Americans is projected and compared against that of working-age Americans. It is shown that in less than ten years, the current healthcare infrastructure will become overloaded and inevitably fail. Several of the other papers such as [1], [6], and [8] support the implementation of more in-home healthcare technology at the present time because it greatly increases the efficiency of caregivers and lowers the cost of healthcare. This cost not only includes money, but also the burden to caregivers, many of which are volunteering friends or family members.

RFID technology has already been implemented into several healthcare applications. For example, RFID tags can be used to track patients. This data can be used to efficiently schedule and place healthcare staff as described in [7], or keep track of wandering dementia patients as described in [6]. Another application of RFID is for tracking medicine. This can be done for inventory tracking purposes as described in [11], [1], and [2], or for patient to medicine matching as described in [9] and [10].

Three in-home medication administration devices are discussed in [1], [2], and [12]. However, none of them are practical enough for patients to use. The devices in [1] and [2] are too complicated, and require the patient to enter safety critical information manually. The device in [12] does not have an intuitive enough interface to be adopted by elderly patients who are uncomfortable with computers. The research presented fills in these gaps in previous work.

CHAPTER 3: PROBLEM AND APPROACH

This chapter describes the approach that is the basis of the research presented. The first section, Research Problem, describes the research problem, and the goal and contributions of this thesis. Section 2, Approach, is a description of how a prototype is implemented to solve the problem. The final section, Summary, is a summary of the problem and solution presented.

3.1 Research Problem

This section describes the research problem, and the goal and contributions of this thesis.

3.1.1 Problem Description

The research presented is done in response to two general problems in the current healthcare system. The first is the inability of patients and healthcare providers to perfectly match medicine type, amount, and schedule with a patient's requirement. This problem is described in [9], [10], and [11], in which RFID technology is proposed as a solution to match patients with medicine in a hospital environment. RFID is the ideal technology for identifying patients and medicines, because RFID tags are inexpensive and can be read wirelessly by devices that interface directly with the HIS. This allows healthcare providers to receive accurate confirmation of medicine type and dosage before administering it to the patient.

The second problem is the impending stress on the healthcare system described in [4]. The growing ratio of patients to healthcare workers can simply not be supported under the current paradigm of infrastructure-based care, such as that provided by hospitals and nursing homes. In [4], [6], and [7], the conversion to an "aging in place" based system is discussed as a solution to allow patients to live independently and receive passive remote monitoring by caregivers. The authors conclude that when patients are monitored from a remote location using passive sensors, it greatly reduces the effort required by caregivers. This is because attention is only needed when an incident is reported by the system. This allows caregivers to use their time more efficiently, reducing the overall cost of healthcare.

Improving the safety of patients and reducing the burden on caregivers and the healthcare system is a worthwhile goal in any context. More importantly, if the vitality of the current healthcare system is in serious danger, the problems presented must be solved in time to prevent the impending crisis. The development of technology that has the ability to further enable independent living of patients is an important step in the process of achieving this goal.

Many healthcare patients would prefer to live independently, but are not able to, because they are not capable of adhering to their complex medication regimen. This requires them to enroll in assisted living facilities, or receive active attention from a caregiver. If a system were developed that could enable them to independently comply with medication schedules, it would relieve a substantial burden on caregivers and the healthcare system as a whole.

3.1.2 Objective

This thesis aims to provide a solution to the previous mentioned problems by developing a revised system based on previous research described in [1] and [2]. The revised system implements automatic medication management and passive remote monitoring to manage complex medication schedules for outpatients. This should prolong independent living, eliminate medication noncompliance, and reduce the burden on the healthcare system.

3.1.3 Contributions

The following contributions describe the goals of this research.

- A system is developed that allows healthcare patients to comply with complex medication regimens without the active assistance of a caregiver.
- The prototype device that is built to implement the system is designed in such a way that it is marketable to anyone who takes medicine on a regular predefined schedule.
- All hardware and software design components associated with the device, including source code, are published so that the device can be reproduced and further development can be done.

3.2 Approach

This section describes the solution system in terms of its concept and requirements. The implementation of the prototype is discussed in Chapters 4 and 5.

3.2.1 Concept

In order to solve the previously mentioned problems, a solution system was developed. The system consists of a device that is meant to replace a patient's medicine cabinet. The enhanced storage device is capable of administering medications in a way similar to a live caregiver, and is capable of contacting an actual caregiver if noncompliance is detected. It also allows the patient to add, remove, and access medications using only the assistance of the device. The system should effectively allow healthcare patients to maximize the length of time that they are able to live independently. This is because medicine reminders and dosage monitoring can be done automatically by the system, and a caregiver can be notified of any issues.

3.2.2 Requirements

The following general requirements have been developed for the solution system in order to satisfy the thesis objectives.

3.2.2.1 Intuitive User Interface

The device shall feature an intuitive user interface that behaves in a way that is similar to the way the patient normally manages medications. This is important because behavioral change requirements hinder adoption, and elderly people do not generally adapt well to high-tech devices. This is an issue with the device presented in [12], since it requires the user to change their behavior from using a medicine cabinet, to interfacing with a high-tech "black box." This can be very intimidating to a user who has never used a computer before, or is generally uncomfortable with technology.

3.2.2.2 Medicine Container Distinction

The device shall be able to physically distinguish which medicine container that the user should take. This is important because all of the medicine containers are stored together, and they all may look similar (especially when looking down at the tops). This will allow the user to easily choose which medicine to take, save them time if there are a lot of medicines to choose from, and ensure that the correct medicine is chosen.

3.2.2.3 Dosage Intake Monitoring

The system shall be able to monitor the amount of medication that the patient ingests. This information is needed to ensure that the patient fully completes a medicine dosage, and to determine if the patient has overdosed.

3.2.2.4 Dynamic Dosage Scheduling

The system shall be able to create an optimized administration schedule based on the dosage frequencies of the medicines enrolled in the device. For example, if two medicines are enrolled in the device with daily frequencies of one and two, the total number of times the user should be required to take medicines is two. In this case, the frequency one medicine dosage can be taken at the same time as one of the frequency two dosages.

3.2.2.5 Unexpected Input Handling

The system shall be able to handle unexpected input from the user. For example, it should be able to recognize if a new medicine is added in an improper way, or if the user has taken medicine when they should not have. It is also important that the prototype constantly monitors which medicine the user is taking while they are taking it. For example, if the user is to take a certain medicine, they should not be instructed as to how much to take until they pick up the correct container.

3.2.2.6 Pharmacy Integration

The system shall be easy to integrate into a pharmacy. The pharmacy is the point at which medicine and schedule information is stored on each medicine container’s RFID tag. This allows for the information to be loaded automatically and accurately. This is a critical success factor for the system since, without it, the user or caregiver must enter medicine and schedule information into the system manually. This additional task is inconvenient, and prone to the risk of error associated with human input.

3.2.2.7 Direct Communication with Caregiver

The system shall be able to directly communicate with the caregiver in case a noncompliance incident is detected. This essential function allows most patients to live independently, since the caregiver can be alerted immediately at any time. The caregiver's contact information should be configurable so that it can be used with a professional, family member, or friend. This increases the market to users who are not under direct medical care, or would like a family member or friend to monitor their medication intake in case of an emergency.

3.2.2.8 Remote Care

The system shall be able to manage medications for the patient who is away from the device. It is unreasonable to assume that the patient will be near the device every time medicine needs to be taken. For example, he may want to go out to dinner, on a vacation, or somewhere where the device cannot be easily transported.

3.3 Summary

This chapter described the research problem, as well as the concept and requirements for the solution presented. The problem is that patients are being enrolled in assisted living facilities simply because they cannot manage their own medications. This causes unnecessary stress on the fragile healthcare system.

A system that implements automatic medication management and passive remote monitoring to manage complex medication schedules can enable many healthcare patients to live independently, while eliminating medication errors and reducing the burden on the healthcare system. The development of this system is the goal of the research presented. The contributions of the research are the system itself, and a fully documented design of the prototype.

The solution system is a device meant to replace an outpatient's medicine cabinet, in order to add assistive features and caregiver monitoring. In order to solve the research problem, the following requirements are associated with the system: intuitive user interface, medicine container distinction, dosage intake monitoring, dynamic dosage scheduling, unexpected input handling, pharmacy integration, direct communication with caregiver, and remote care.

CHAPTER 4: PROTOTYPE HARDWARE

This chapter describes the design and construction of the prototype hardware, and the associated costs. The first section, Concept, describes the overall concept of the prototype. Sections 2 and 3 describe the prototype in terms of system architecture and hardware. Section 4, Costs, is an analysis of development costs. Section 5, Assumptions, is a declaration of all assumptions made. The final section is a summary of the prototype hardware. Detailed hardware designs and a complete budget table are listed in appendices A and B respectively.

4.1 Concept

A complete prototype has been developed (Figure 4.1) according to the approach described in Chapter 3. The prototype is a fully functioning device that a patient can use to easily manage their medications and prescription schedules. The device itself is a mechanical tray that is controlled by a microprocessor-based computer system. The user interacts with the device through the tray itself and an external user interface panel. The device can also interact with a patient’s caregiver or pharmacist through an Internet connection. Patient operation is mostly automatic, and the interface is designed to be unobtrusive, and mimic a traditional medicine taking procedure. For instance, during normal operation, no button presses or intentional user feedback is required; the user just picks up a medicine container, takes the dosage on the label, and replaces it.

The system is primarily composed of five main components: a microcontroller, RFID reader, scale, user interface panel, and motorized tray platform. The microcontroller controls all processing, data storage, input/output (I/O), and telecommunications. The RFID reader provides the microcontroller with tagged medicine bottle information, including each medicine's dosage information and UID. The digital scale provides the microcontroller with weight measurements accurate enough to judge individual medicine units (i.e. pills) removed from a container. The user interface panel displays instructions and dosage information to the user and provides buttons to navigate any system menus. The motorized tray platform provides the ability to reposition the medicine bottles on the device. Using these components, the following features are implemented:

- The device can alert patients by alarm and cellphone text message when it is time to take medicine.
- The device can automatically tell if a patient has taken the right amount of dosage for each medicine.
- The device can automatically rotate the correct medicine bottle in front of a patient for him to take.
- The device can automatically alert a patient's healthcare provider or pharmacist (via email or text message) of overdose or underdose incidents.
- Data on a medicine bottle's RFID tag enables the device to automatically read the medicine's information without requiring manual input from the patient or caregiver.

Figure 4.1: Picture of the final prototype

4.2 System Architecture

The following sections describe the detailed system architecture of the prototype. It is important to note the primary use of generic prototyping components, specifically the Arduino platform, rather than designing and fabricating custom circuits. This allowed for the development of more features, less troubleshooting, and more simplified software development in higher-level programming language. A basic diagram of the system architecture is shown in Figure 4.2. Component wiring diagrams are shown in Figure 4.3 and Figure 4.4. A picture of the inside of the prototype with the components mounted and wired up is shown in Figure 4.5.

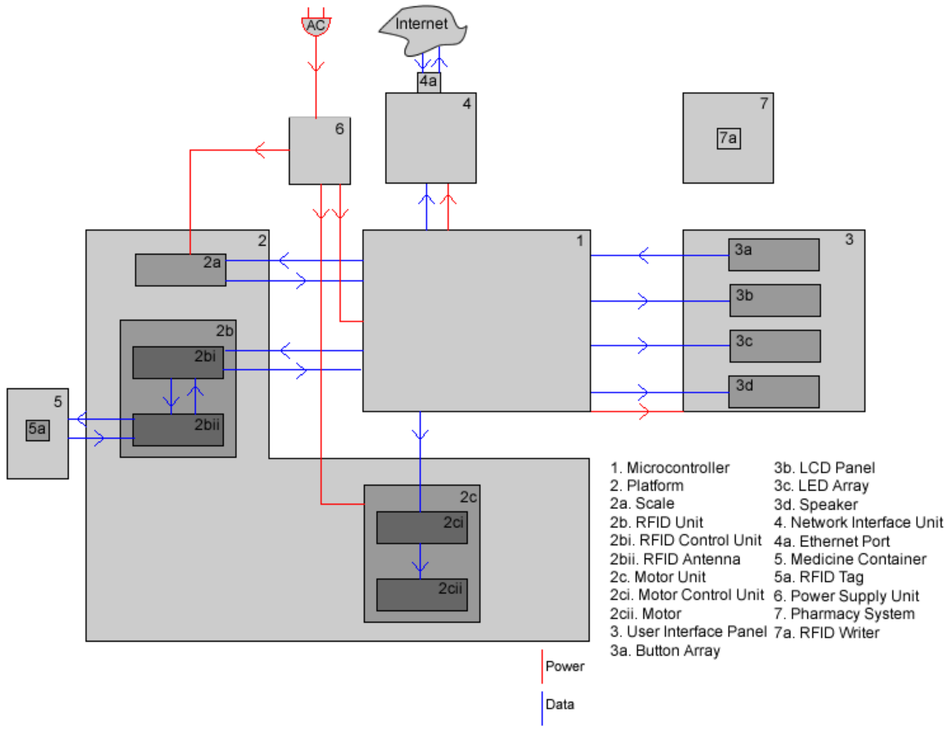


Figure 4.2: System architecture block diagram

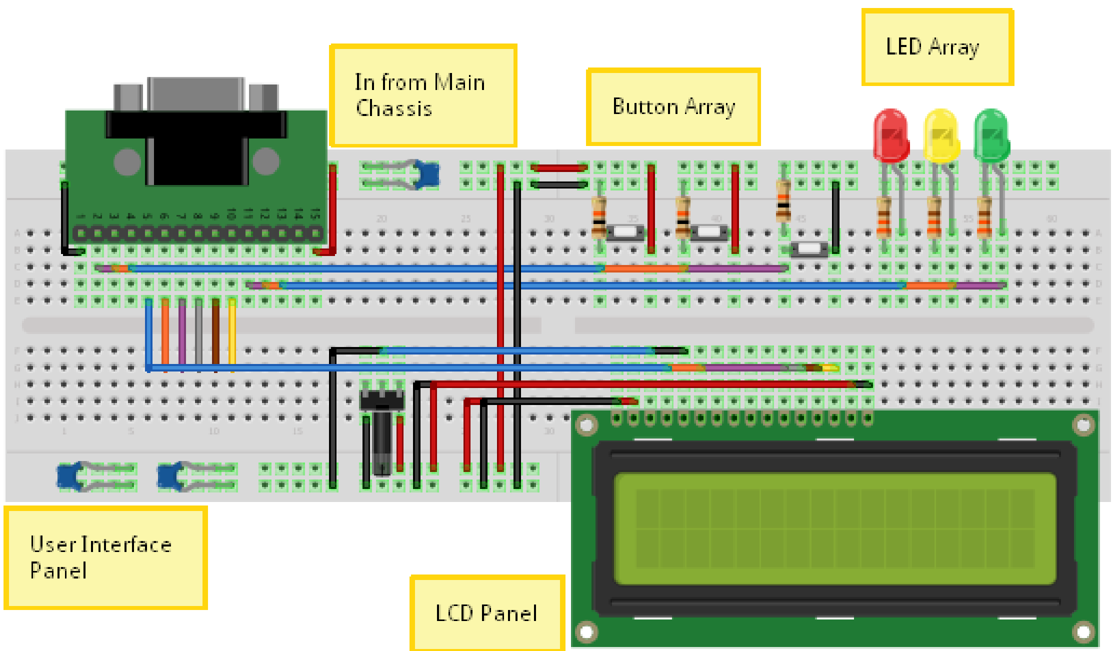


Figure 4.3: System architecture wiring diagram of user interface panel

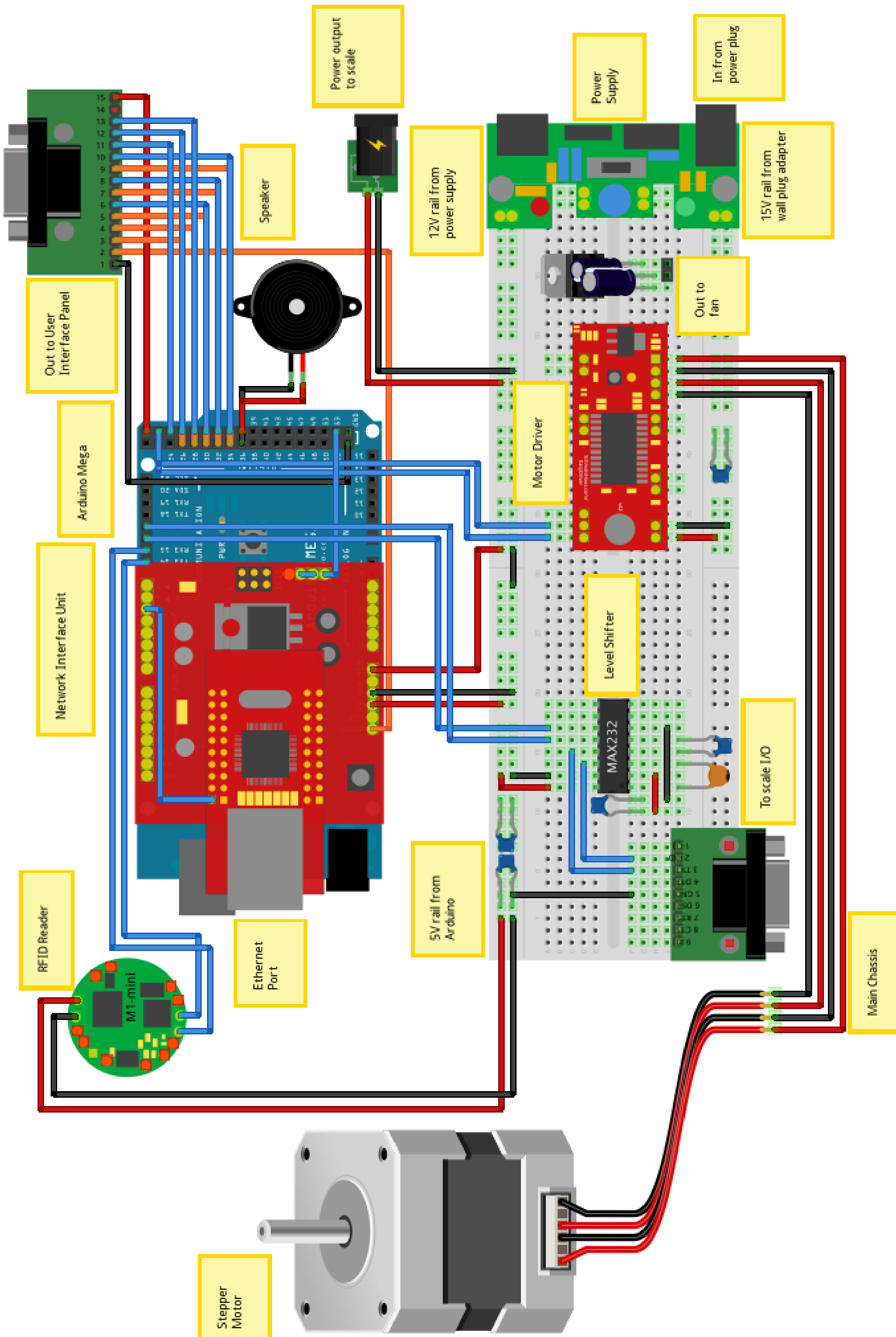


Figure 4.4 System architecture wiring diagram of main chassis unit

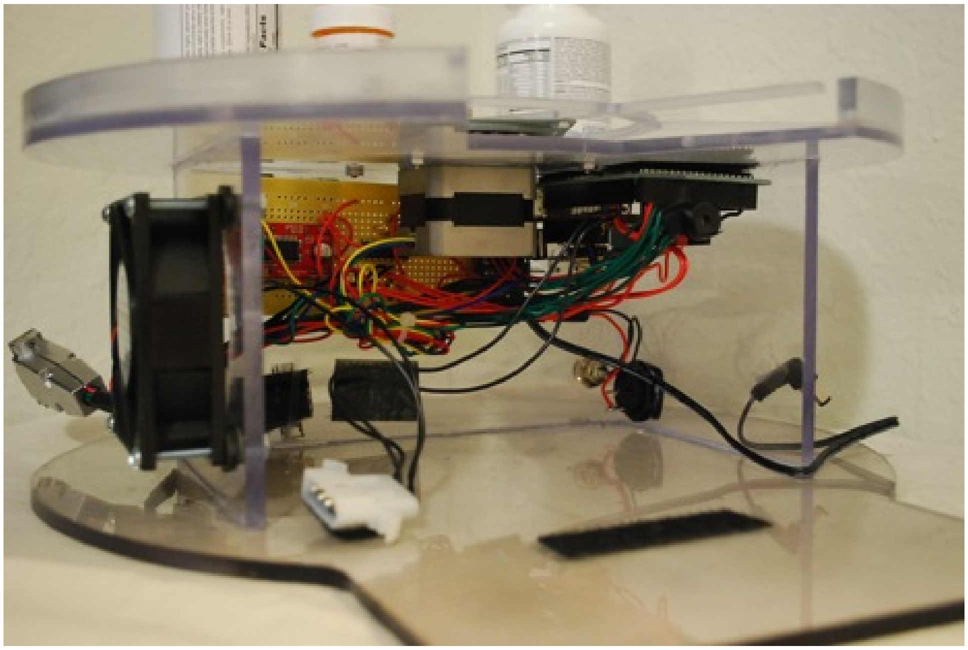


Figure 4.5: Picture of the inside of the prototype

4.2.1 Microcontroller

The microcontroller is the central processing unit for the device. It stores and executes the system software and interfaces with all of the peripheral devices. It is also responsible for backing up the system memory to persistent storage in case of a power failure or disconnect. The prototype uses the Arduino Mega (Figure 4.6), an open source prototyping board based on the Atmel ATmega1280 microcontroller [13]. The board includes all necessary hardware to run the microcontroller, easily access its I/O ports, provide regulated power to the microcontroller and peripheral devices, and provide a Universal Serial Bus (USB) connection to a PC for debugging and programming. This board was chosen because it is the only standard Arduino board that met the requirements for the prototype and it operates at 5 Volts (V) like the rest of the components. The table in Figure 4.7 compares the capabilities of the board with the prototype requirements, and the table in Figure 4.8 maps each of the board's used I/O ports to other system components. The board's hardware schematic is given in [14].

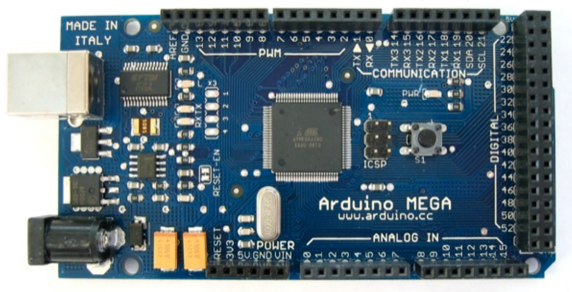


Figure 4.6: Picture of an Arduino Mega prototyping board[4]

Item	Required	Arduino Mega
GPIO Ports	20	60
UARTs	3	4
Code Space (Bytes)	47,200	128,000
EEPROM (Bytes)	445	4,000

Figure 4.7: Arduino Mega capabilities vs. prototype requirements

Arduino Port	Device
UART 0-1	USB Serial
GPIO 2	Network Interface Unit Reset
SPI 11-13, 55	Network Interface Unit SPI
UART 16-17	Scale Serial Connection
UART 18-19	RFID Unit Serial Connection
GPIO 22, 23	Motor Unit Driver Control

[4] Source: http://arduino.cc/en/uploads/Main/ArduinoMega.jpg

GPIO 24, 26	User Interface Panel Buttons
GPIO 25, 27, 29	User Interface Panel LEDs
GPIO 30-35	User Interface Panel LCD
GPIO 36-37	Speaker Input
Reset	User Interface Panel Reset Button

Figure 4.8: Arduino board I/O port map

4.2.2 Platform

The platform is the top surface of the device on which all of the containers are stored. It allows the containers can be repositioned, identified, and weighed. The platform includes three subcomponents, the scale, the RFID unit, and the motor unit. These components are discussed separately in the following sections.

4.2.2.1 Scale

A scale is integrated into the platform in order to weigh an individual container's weight before and after medicine consumption. It receives control input from the microcontroller and returns scale measurements to the microcontroller. The prototype uses an Acculab Vicon Portable Digital Balance VIC-303 with an RS-232 interface kit (Figure 4.9). This scale was chosen because of its low price, high precision, ample capacity, and computer interface. According to its specifications, it is precise to 1 milligram and has a capacity of 300 grams. This is excellent considering that very few pills weigh less than 100mg, and even large liquid medicine containers weigh less than 300 grams. It is important to note that the scale requires a 12 V input and cannot be powered by the 5 V regulated power output from the Arduino board.

The scale also has an RS232 serial interface. Since the voltage range of RS232 logic levels is much greater than that of the microcontroller, a MAX232 level shifting chip is used to interface the scale to one of the microcontroller's universal asynchronous receiver/transmitter (UART) ports. With the level shifter in place, two-way serial communication can be performed at the standard transistor–transistor logic (TTL) voltage levels of the microcontroller. The scale is connected to the device over 4 wires. Voltage and Ground are connected to a separate 12 V regulated power source, and the Transmit (TX) and Receive (RX) are connected to the microcontroller's UART port through the level shifter.

Figure 4.9: Picture of an acculab vicon portable digital balance VIC-303[5]

4.2.2.2 RFID Unit

The RFID unit is used to identify medicines, and to scan type and dosage information stored on a container's RFID tag. It receives control input from the microcontroller and returns scanned tag data to the microcontroller. The prototype uses a Skyetec

---

[5] Source: http://www.acculab.com/pix/rendered/220x245/acc0002d2-WH-F.jpg

SkyeModule M1-Mini RFID reader (Figure 4.10). This reader was chosen because it is small, has an integrated antenna, a short and reliable read range (to avoid wireless "collisions" with other tags), memory block read/write support, a TTL-level UART interface, and a well-documented data exchange protocol. The RFID reader is connected to the microcontroller on the device over 4 wires: Voltage, Ground, UART TX, and UART RX.

Figure 4.10: Picture of a SkyeModule M1-mini RFID reader[6]

4.2.2.3 Motor Unit

The motor unit provides the motion necessary to reposition the containers around the platform. It receives control input from the microcontroller. The prototype uses a 17HS4002-56M bipolar stepper motor (Figure 4.11), coupled with the EasyDriver 4.2 stepper motor driver circuit (Figure 4.12). The driver circuit is a prototyping board based on the Allegro A3967 motor driver chip. In this configuration, the motor can be digitally controlled by the microcontroller to accurately adjust its angular position within 0.225

[6] Source: http://media.digikey.com/photos/Skyetec%20Photos/SM-MN-00-HF-RC.JPG

degrees, with 120g*cm of torque. This is excellent considering that the minimum rotation maneuver for the prototype is 45 degrees, and it is not likely that more than a few containers will be large.

It is important to note that the motor requires a 15 V power supply and draws a large 280 milliamps of current under load. Therefore, it cannot be powered by the regulated power output from the Arduino board and must be attached to a more upstream, higher voltage power source. The stepper motor driver is connected to each of the 4 inputs on the stepper motor, and to the main device over 4 wires. Voltage and Ground are connected directly to the 15 V power adapter through the power supply circuit, and the direction and step ports are connected to two of the microcontrollers general purpose input/output (GPIO) pins. The board’s hardware schematic is given in [15].

Figure 4.11: Picture of a 17HS4002-56M stepper motor[7]

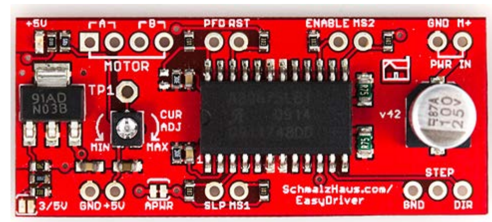


Figure 4.12: Picture of an EasyDriver 4.2 stepper motor driver[8]

4.2.3 User Interface Panel

The user interface panel displays and receives information to and from the user. It is separated from the platform device by a 14-wire tether. It consists of 4 parts, the button array, the Light Emitting Diode (LED) array, the Liquid Crystal Display (LCD) panel, and the speaker. These components are discussed separately in the following sections.

4.2.3.1 Button Array

The button array consists of 2 large general purpose buttons to navigate menus on the screen, and 1 small reset button for resetting the device if it malfunctions. The 2 general purpose buttons are connected to 2 of the microcontroller's GPIO ports in such a way that

---

[7] Source: http://static.sparkfun.com/images/products/09238-01.jpg

[8] Source: http://www.schmalzhaus.com/EasyDriver/EasyDriverV42_Top.PNG

when pressed, the port will read logic high, and when released, the port will read logic low. The reset button is connected directly to the microcontroller's reset port and reads logic low when pressed, and logic high when released. The diagrams for this circuit are found in **Error! Reference source not found.** and Figure 4.3.

4.2.3.2 LCD Panel

The LCD panel displays information to the user, including current medicine information, instructions, alert explanations, and system menus. The prototype uses a standard 20x4 HD44780 white on blue LCD (Figure 4.13) which can display up to 80 ASCII characters at one time. This display was chosen because it is large and bright enough for a person with poor vision to see, and it has a standard interface. It is connected to the microcontroller's regulated power and GPIO ports over 8 wires: Voltage, Ground, Register Select, Enable, Data Bit 4, Data Bit 5, Data Bit 6, and Data Bit 7. There is also a small potentiometer mounted behind the display to adjust the contrast.

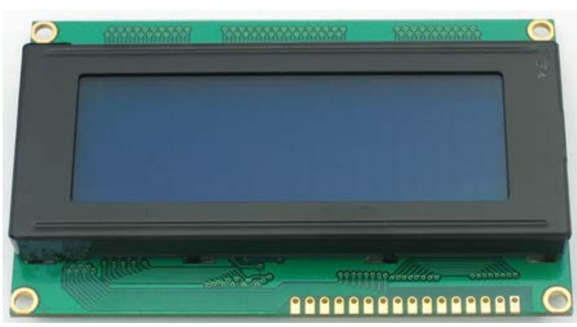


Figure 4.13: Picture of a 20x4 HD44780 white on blue LCD[9]

[9] Source: http://www.adafruit.com/images/medium/LCDblue204_MED.jpg

4.2.3.3 LED Array

The LED array displays information from the microcontroller concerning the system status. There are 3 LEDs: red, yellow, and green. The red LED indicates that the device is connected to power, yellow indicates that the system is busy repositioning the containers or accessing the connected network, and green indicates that attention is required from the user. Each LED is connected to the microcontroller over 2 wires: Ground and Positive. They can be individually switched on and off by toggling the corresponding Positive port to logic high or logic low respectively.

4.2.3.4 Speaker

The speaker emits audible alerts from the microcontroller when a medicine dosage is to be taken. The prototype uses a small piezoelectric buzzer connected to 2 of the microcontroller's GPIO ports. To generate sound, the microcontroller alternates the two ports between logic high and logic low, for a maximum voltage swing of 10 V. When this is done at a frequency of about 4 kHz, a loud buzzing sound is emitted that is adequate to alert a nearby patient.

4.2.4 Network Interface Unit

The network interface unit receives text message information from the microcontroller and transmits it, through the Internet, to the patient or a remote caregiver. It also receives time information from an Internet time server in order to keep its clock synchronized. The prototype uses an NKC Electronics Ethernet shield (Figure 4.14), which easily interfaces the Arduino with a Wiznet WIZ812MJ Ethernet module over a serial peripheral interface (SPI) connection. This board was chosen because it is the only compatible Ethernet interface available for the Arduino Mega. It is connected to the Internet through an active Ethernet line. It is connected to the microcontroller's regulated power and SPI ports over 6 wires: Voltage, Ground, SPI SCLK, SPI MOSI, SPI MISO, and SPI SS. The board's hardware schematic is given in [16].

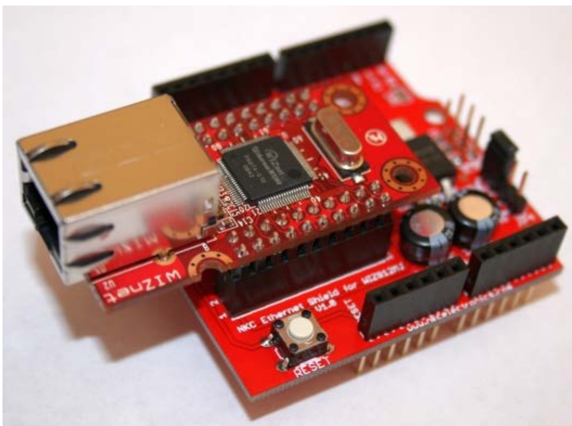


Figure 4.14: Picture of an NKC Electronics Ethernet shield[10]

4.2.5 Medicine Container with RFID Tag

The medicine container stores a single type of medicine for the user. This is a traditional medicine bottle or blister pack outfitted with an RFID tag on the bottom. The RFID tag contains a UID and a small amount of data. The prototype uses Texas Instruments Tag-it^{TM} HF-I RFID tags (Figure 4.15). They are ISO15693, operate at 13.56MHz (the same frequency as the reader), and contain a 64-bit UID and 256 bits of programmable data. The tags are about the size of a quarter, and as thin and flexible as a single sheet of paper. They also contain just enough data space to store the 212 bits of medicine information

[10] Source: http://www.vintagecomputercables.com/img/ethernet_shield_5.jpg

required by the data protocol that will be described in the CHAPTER 5:. This makes them ideal for our prototype.

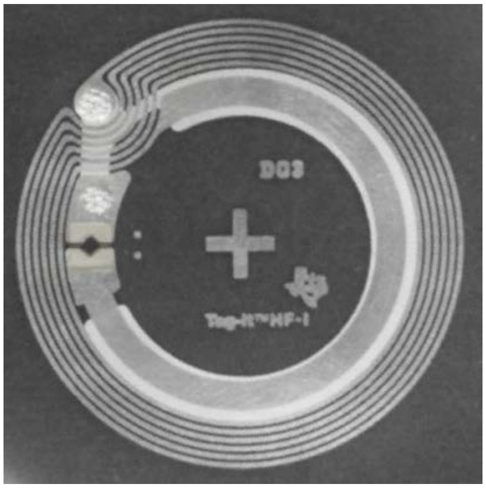


Figure 4.15: Picture of a Texas Instruments Tag-it™ HF-I RFID tag[11]

4.2.6 Power Supply Unit

The power supply unit provides power of the appropriate form and voltage to the Arduino, scale, and motor unit. It is connected directly to a standard 15 V 1.2 Amp DC power adapter. The prototype uses the Adafruit Adjustable Breadboard Power Supply (Figure 4.16). This supply was chosen because it can output the input voltage in addition to an adjustable voltage which is set to 12 V. The 15 V rail is connected to the motor control unit, and the 12 V rail is connected to the scale and Arduino board. All other components are powered through a 5 V power regulator on the Arduino board. The board's hardware schematic is given in [17].

[11] Source: http://media.digikey.com/photos/Texas%20Instr%20Photos/RI-I17-114A-01.JPG

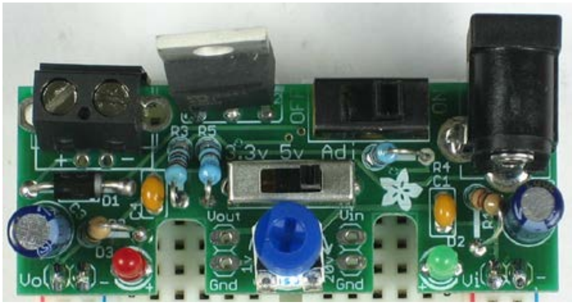


Figure 4.16: Picture of an Adafruit Adjustable Breadboard Power Supply[12]

4.2.7 Pharmacy Integration

In order for the prototype device at a patient's home to function, cooperation at the pharmacy is required to store prescription information on each medicine bottle. This is done by the addition of an RFID writer to the pharmacy's current computer system. The procedure for writing data to the tag is as easy as placing the tag within 6 inches of the reader platform for less than 1 second. The work done on this prototype does not encompass the development of this system. However, its implementation is necessary before real-world application is possible. For testing purposes, we used the DLP-RFID1 RFID writer (Figure 4.17) to manually write data to tags. This reader was chosen because it is compatible with our RFID tags and has a convenient USB interface.

[12] Source: http://www.adafruit.com/images/medium/bbpsup_MED.jpg

Figure 4.17: Picture of a DLP-RFID1 RFID writer[13]

4.3 Hardware

The following sections describe, in detail, the hardware design of the prototype. All custom parts made for the device components are made entirely out of LEXAN® polycarbonate sheets that were professionally cut at UCF's machine shop and assembled by hand. Each part was designed individually with SolidWorks 3D CAD modeling software. A textured assembly model of the final device is shown in Figure 4.18, and a basic hardware diagram is shown in Figure 4.19. Detailed dimensional drawings of each chassis component are included in Appendix A: for reference.

[13] Source: http://apple.clickandbuild.com/cnb/shop/ftdichip?imageID=182&op=imgLib-viewImage

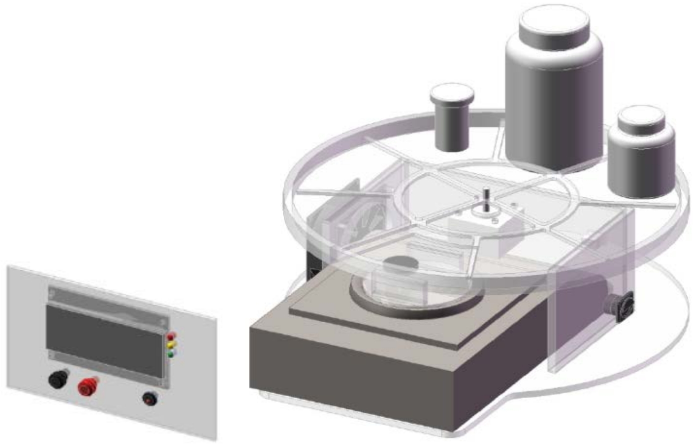


Figure 4.18: SolidWorks assembly model of the prototype

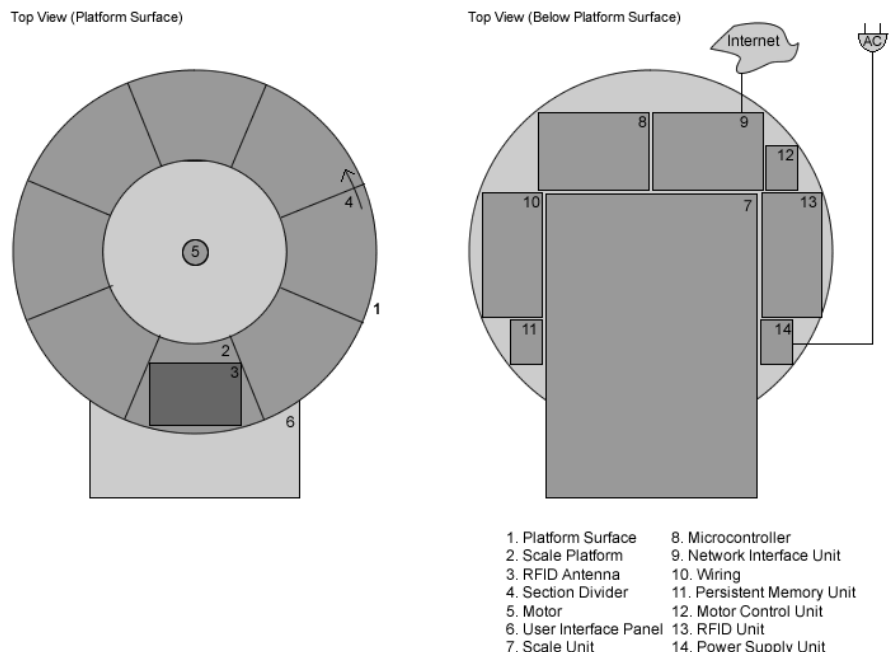


Figure 4.19: Basic hardware diagram

4.3.1 Main Chassis

The main chasses is a 14 and 1/4” by 12” by 5 and 3/16” structure that is designed is to mimic that of a Lazy Suzan revolving shelf traditionally used to store medicines. The scale and all electronics are placed inside of the structure, and a large circular tray is mounted to the top. A motor interfaces the tray and turns the section divider, which is capable of repositioning medicine containers placed on the tray. Figure 4.20 shows the main chassis by itself, and Figure 4.21 shows it with the scale and motor attached. The other holes cut into the sides are for mounting a cooling fan, power switch, and external Ethernet and USB ports. The following sections describe the tray platform and section divider in greater detail.

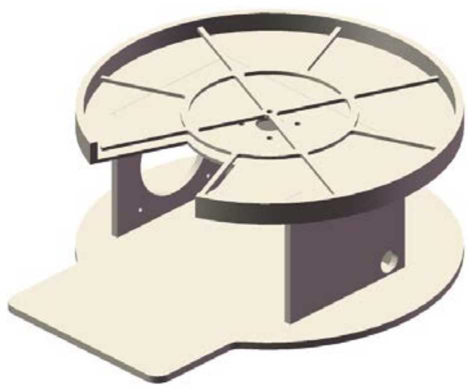


Figure 4.20: SolidWorks assembly model of the main chassis

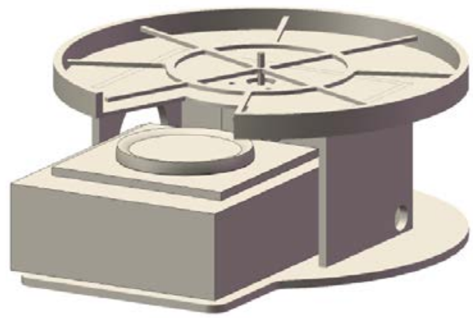


Figure 4.21: SolidWorks assembly model of the main chassis with motor and scale

4.3.1.1 Tray Platform

The tray platform (Figure 4.22) is a flat, circular section that provides a surface for medicine containers to be stored. It is 12” in diameter, 1/8” thick, and has a 5/8” high retaining wall that is 1/4” wide. The surface is smooth enough that containers can easily slide along it when pushed horizontally. In this design, there is enough area to store up to 8 circular containers along the perimeter with diameters of up to 3 inches each. An eighth of the platform is cut out of the edge in order to separate a single section. This is done so that containers can be weighed individually on the cutout section. This will be discussed further in the following section. There are also several holes cut out to facilitate mounting the tray platform to the rest of the device and the motor.

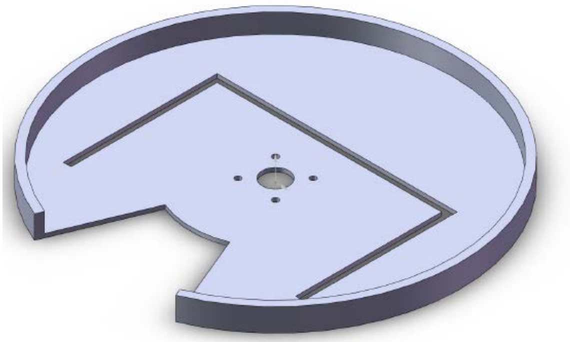


Figure 4.22: SolidWorks model of the tray platform

4.3.1.2 Section Divider

The section divider (Figure 4.23) is a star-shaped part that is mounted to the shaft of the stepper motor over the tray platform using a Lazy Suzan ball bearing. It is 11 and 1/4” in diameter, 1/8” thick, and effectively divides the tray platform into 8 strict sections defined by the outer section divider edge and the tray platform retaining wall. The interior sections are cut out to reduce the weight of the part. When the section divider is rotated, all containers constrained in its sections will be horizontally pushed to slide across the tray platform surface.

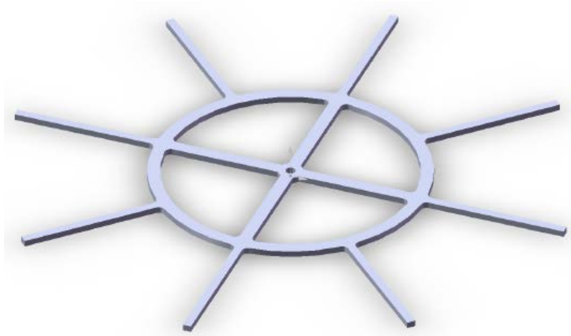


Figure 4.23: SolidWorks model of the section divider

4.3.2 Scale Platform

The scale platform is raised above a stationary high-precision scale that is able to measure exactly 1 of the 8 sections. When assembled with the main chassis, its surface is physically flush with the tray platform surface so that medicine containers can easily traverse across it. The scale platform assembly is shown by itself in Figure 4.24, and attached to the scale and RFID reader in Figure 4.25. The shape of surface is identical to the void section in of the tray platform. Therefore, when assembled with the main chassis and scale, the top surface of the device acts as one large uniform tray as seen in Figure 4.18. The RFID reader is positioned in such a way that it should read and only the tag of the container placed on the scale platform, without any interference from containers in other sections. The mounting hole is offset so that if a small container is pushed onto the scale platform, its tag will still stop above the reader when rotated counter-clockwise.

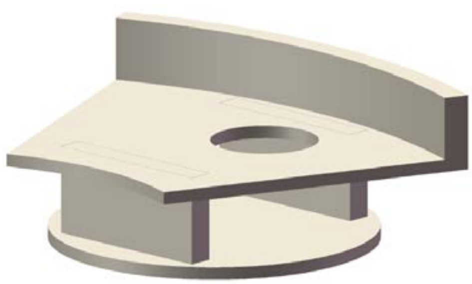


Figure 4.24: SolidWorks assembly model of the scale platform

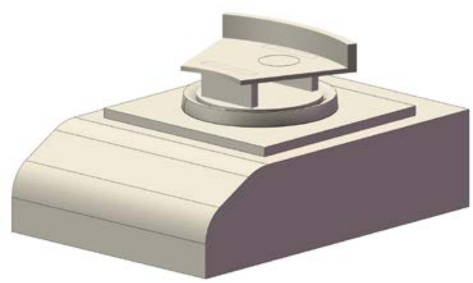


Figure 4.25: SolidWorks assembly model of the scale platform and RFID reader

4.3.3 User Interface Panel

The external user interface panel is 4 and 1/4” by 6 and 3/4”, and contains the button array, LCD panel, and LED array. In the final prototype, the panel was mounted into a 5” by 7” wood picture frame as shown in Figure 4.1. It is shown by itself in Figure 4.26, and attached to the other components in Figure 4.27. Figure 4.28 shows a basic diagram describing the user interface panel and its components.

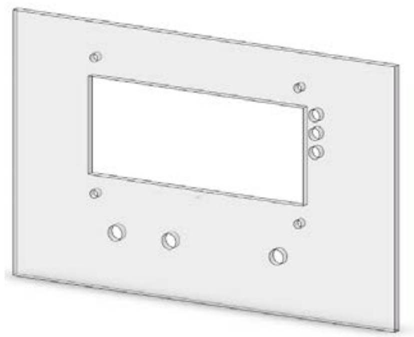


Figure 4.26: SolidWorks model of the user interface panel

Figure 4.27: SolidWorks model of the user interface panel with components attached

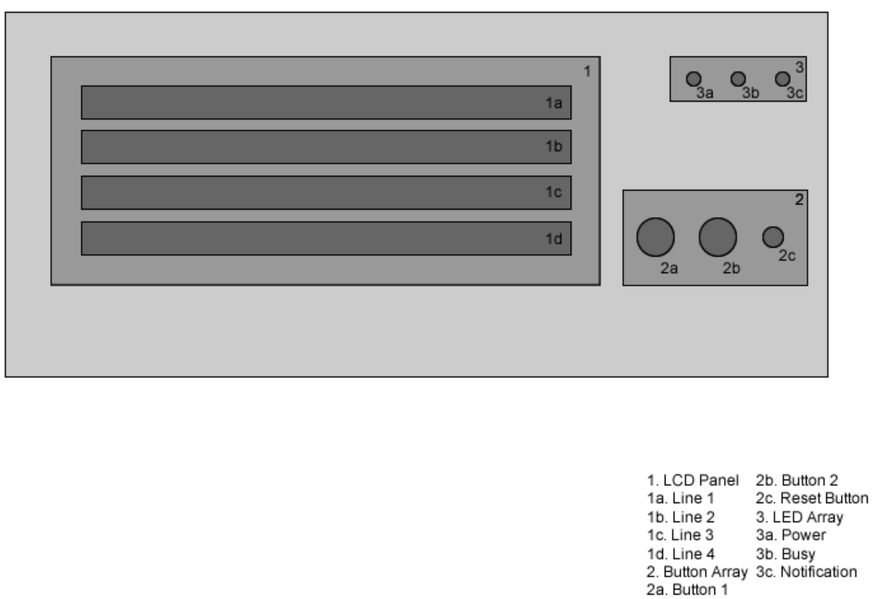


Figure 4.28: User interface panel hardware diagram

4.4 Costs

The estimated total cost of the parts necessary to build the prototype is calculated to be $1011.51, which includes 1 of the RFID writer required at each participating pharmacy. This cost is shown organized by category in Figure 4.29. A detailed per-item budget table with cost estimate sources is listed in APPENDIX B: for reference. The cost of a mass produced device could be reduced significantly from the prototype cost and is qualitatively discussed in the following sections.

Category	Price
System Components	$611.67
Other Electronics	$105.05
Materials	$164.30
Pharmacy Equipment	$130.49
TOTAL	$1,011.51

Figure 4.29: Prototype cost organized by category

4.4.1 Scale

The most expensive component of the prototype is the scale, which is costly because of its high capacity, high precision, and data interface. In mass production, it may be possible to design a custom scale for the device that would be less expensive to produce, fit the device better, and function better in the typical device conditions. This is because the scale was designed for laboratory use and includes many features that are not necessary. For example, the scale features a user interface panel that faces the back of the prototype when installed, and is completely unused.

4.4.2 RFID Writer

The RFID writer is expensive because it includes the RFID writer circuit, an enclosure, a USB interface with cable, and software for writing data to tags. It is ready to go right out of the box. In actual production, the RFID writer would need to be custom designed to interface the current pharmacy computer system in terms of hardware and software. This would lower the cost in the same way as the scale. The components used in the RFID writer cost much less than the purchase price. For example, the RFID reader used in the prototype device costs 34% less and includes identical writing functionality, only it does not include an enclosure or direct computer interface.

4.4.3 Materials

The custom parts for the prototype were built out of expensive Lexan® polycarbonate, which is great for prototyping since it is transparent, strong, and can be easily bonded, drilled, and tapped. The polycarbonate is only sold in large, standard-sized sheets, which left a large amount of unused material. In actual production, the prototype device could be redesigned to use a significantly less expensive plastic and the unused material could be used to fabricate more units, which would significantly increase yield and lower cost.

4.4.4 Electronics

The electronics used in the prototype are dominated by prototyping boards that are expensive because they include constructed circuit boards that are easy to interface. In many cases, such as the EasyDriver board used to drive the stepper motor, the cost of the needed chip is significantly less than that of the prototyping board. In this case, the actual driver chip alone only represents about 11% of the cost if purchased in individually [18].

Similar savings can easily be made by custom designing a single circuit board based on the prototyping board schematics listed in the Appendix **Error! Reference source not found.**. The board would only implement features and interfaces that were necessary and cost significantly less that the prototyping boards. This would also make the electronics much easier to produce and could be easily shaped to fit a future device design.

4.5 Assumptions

Several assumptions and limitations of the prototype hardware are recognized in the following sections. Significant limitations are readdressed in **Error! Reference source not found.**.

4.5.1 User Interface

Patients interface the prototype device in 2 ways, the user interface panel, and the tray platform. This limits the device’s usefulness to patients who are able to read the text on the LCD screen, and effectively remove and replace the small medicine containers in the tray platform sections. For instance, patients with poor eyesight or regular hand tremors will not likely be able to use the device. The device is also built around the concept of retrofitting traditional medicine containers with RFID tags. This means that patients are required to manually count the right amount of dosage from a medicine container according to the instructions displayed on the user interface panel and medicine bottle.

4.5.2 Reliability

Reliability is an extremely important factor in a critical medical device. This device relies on an uninterrupted supply of power, an open connection to the internet, and the messaging system used to deliver alert messages, specifically email and SMS. If any of these fail, the device is rendered dangerous considering that the patient is depending on its function.

4.5.3 Design

The hardware design of the device presented is an elementary prototype. It proves the concept of the research, but is not meant to be put into production. A production device would need to redesigned to be more efficient, robust, and aesthetically pleasing.

4.5.4 Restrictions

The hardware design of the device implies certain restrictions. For instance, the device is limited to Ethernet-based Internet connectivity, and is not compatible with a Wi-Fi or cellular data connection. Additionally, the medicines that are compatible with the device are restricted to a size and weight that is able to fit in the tray platform sections and be pushed by the motor. Lastly, the device does not include a way to initialize the position of the section divider when the device is powered on. Since the stepper motor is ignorant of its absolute angular position, it can only be adjusted relative to its current position. An additional sensor is needed to initialize its position.

4.6 Summary

This chapter described the prototype hardware concept, design, cost, and assumptions. The prototype device was developed according to the approach discussed in CHAPTER 3:, and is designed in a way that mimics a traditional Lazy Suzan medicine shelf. It implements network connectivity for remote notifications, RFID technology for medicine identification and prescription information, a scale to measure actual medicine dosages, and a novel motorized tray platform to distinguish particular medicine containers to the patient.

The prototype was able to be constructed quickly by using polycarbonate materials, prototyping boards, and off the shelf components instead of using the more efficient materials and custom components required in a production device. This increased cost and reduced robustness, but this device is intended to be a prototype and requires much redesign before it is ready to be put into production. In addition, the necessary pharmacy-based system required to produce compatible medicine containers is not developed in this project, but is necessary before production is possible.

The cost of the prototype was estimated to be $1,011.51. This is much higher than the potential mass production cost because of the expensive generic prototyping components and small quantities purchased. This cost also includes 1 RFID writer required by the pharmacy to produce compatible medicine containers. However, the actual cost to implement the system into pharmacies is largely unknown because it requires integrating hardware and software into the current information system.

CHAPTER 5: PROTOTYPE SOFTWARE

This chapter describes the design of the prototype software, and how the user interacts with it. The first section, Concept, describes the overall software concept. Section 2, User Operation, describes how the user interacts with the prototype. Section 3, Data Protocols, describes the custom data storage and communication protocols used. The final section, Description of Source Code, describes the software code and algorithms. This chapter deliberately neglects the software component required at the pharmacy, as it is outside the scope of this prototype.

5.1 Concept

The software is the most important component in the user interface design, and was developed according to the approach described in Chapter 3. It allows the patient operation of the prototype to be mostly automatic, as the interface is designed to be unobtrusive by mimicking the traditional medicine taking procedure. For instance, in normal operation, no button presses or intentional user feedback is required. The user simply picks up a medicine container, takes the dosage on the label, and replaces it. For cases in which the attention of the patient of caregiver is required, such as if the user forgets to take a medicine dosage or takes too much, the appropriate party is notified audibly, visually, or via remote text message. Figure 5.1 is a basic diagram that shows how the user interacts with the prototype, and where each of its main functions is integrated into the main process loop.

The user is required to initially configure the device with basic information such as the current time zone, daylight savings time status, and the telephone number of the patient and the caregiver. This is done through the LCD screen and two large buttons on the user interface panel. Whenever the device is powered on, it can automatically load the previous settings from before the device was powered off. The time is also automatically set at startup by a network time server on the Internet.

Medicine schedule information is read in from each medicine container's RFID tag when it is first placed on the device. This information includes the medicine name, dosage amount, dosage frequency, and any special instructions. The system automatically parses the dosage frequencies into a schedule that will allow the user to take any of their medicines up to four times per day. Whenever it is time for one or more medicines to be taken, the user is alerted by an audible alarm and text message. The prototype can then administer the medicines to the user one container at a time.

The user has the option of removing medicines from the device either permanently or temporarily. Temporary removal is an option if they wish to leave their home for a period of time, but still receive remote reminders and compliance checking. In this case, the device alerts the patient via remote text message when it is time to take any of the checked out medicines. When the medicines are returned to the device, it checks that the proper amount of each medicine was taken. The caregiver is then notified accordingly. The user also has the option of conveniently taking a medicine up to one hour earlier than the scheduled time.

The device also tracks the weight of all medicines in-between dosage times so that when the medicine container is reaccessed, it can be known if the patient has taken any unscheduled dosages. The system will also do a full inventory check once per day in case the patient adds a medicine to the tray, but forgets to place it on the scale platform to be identified. This allows the caregiver to be assured that all of the medicines are being constantly monitored by the system.

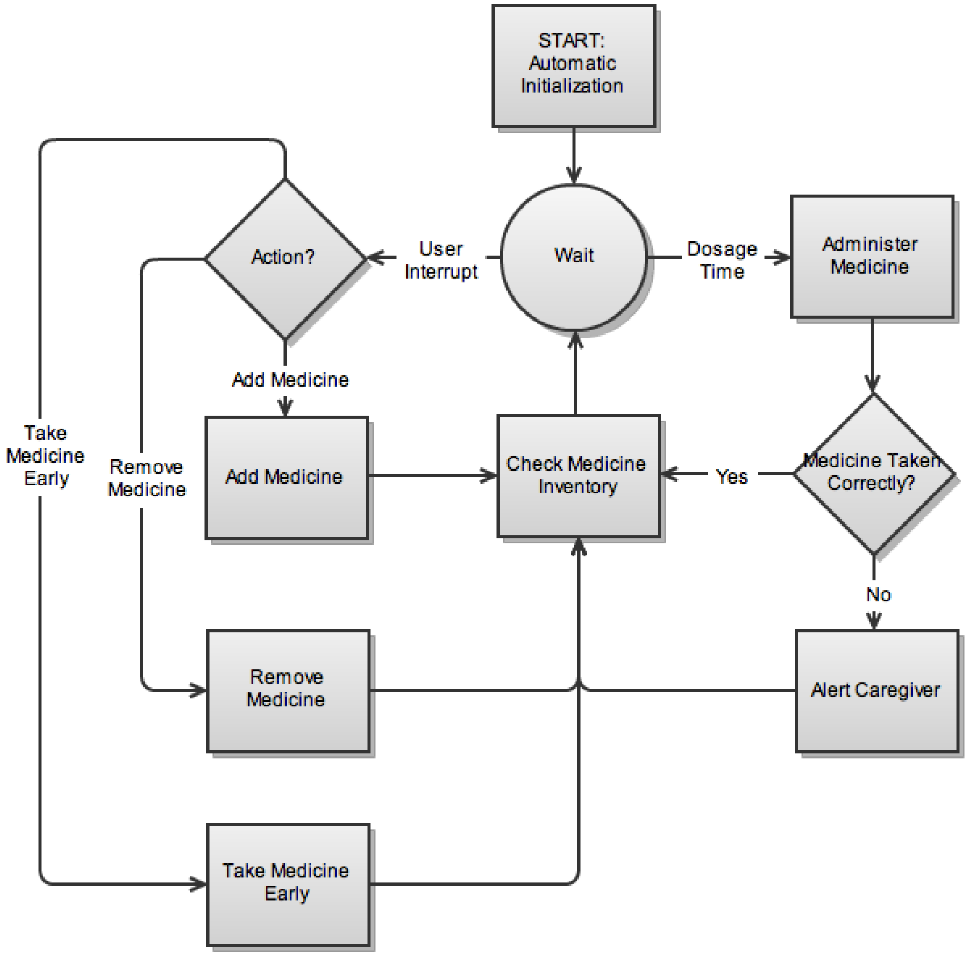


Figure 5.1: Basic flowchart describing the system's interaction with the user

5.2 User Operation

The following sections describe exactly how the user interacts with the device in key situations.

5.2.1 Initialization

When the device is powered on, an initialization sequence is activated. This occurs when either the user plugs in the device for the first time, relocates the device to another power outlet, or the device is reactivated after a power failure. It first attempts to load previous settings from the microcontroller's persistent memory. If both general purpose buttons are pressed at the same time during the first 5 seconds after power-on (during this time, the device displays a splash screen), the persistent memory will be entirely cleared.

Once the persistent memory is loaded, the user interface panel prompts the user for basic settings. If these settings are available from memory, they are loaded as defaults on the entry screen, and are automatically loaded if the user does not respond, such as if the device is restarted by a power failure while the user is away. Otherwise, the user is required to enter the following information into the system using the two general purpose buttons and the LCD display: time zone, daylight savings time status, caregiver cellphone number, and patient cellphone number. All four of these items must be entered. If they are not available in memory, the system will wait indefinitely for user input. When all of the settings are loaded, the device automatically connects to the Internet and sets the system time.

The system will then update its database of medicines by combining medicines previously on the device, medicines currently on the device, and medicines that have been checked out by the user. Each container on the device is also weighed to ensure that the current weight of each bottle is the same as before the device was powered off. Once the database is compiled and the containers are checked, the persistent memory is updated, and the caregiver is sent a message describing the initialization and any discrepancies found.

5.2.2 Adding a Medicine

When there is no medicine to be taken, the prototype assumes an idle state in which the scale platform empty. To add a medicine to the device, the user places the new medicine container onto the scale platform. Once the RFID reader identifies it, it is weighed, then moved by the section divider to clear the scale platform for any additional new medicines. This procedure requires no button input and mimics the traditional procedure to add a new medicine to a medicine shelf. Once the medicine has been added, the persistent memory is updated and the caregiver is notified.

The device must always maintain one blank section to accept medicine container input and zero the scale. For example, if there are n total sections, and $n - 1$ medicines in the medicine database (including medicines checked out by the user), new medicines are not accepted. In this case, the prototype will alert the user when they try to add a new medicine to a full device.

5.2.3 Removing a Medicine Permanently

To permanently remove a medicine from the system, the user takes the medicine container from the tray platform and places it onto the scale platform. The medicine will be identified and the system will prompt the user to press the black button to remove the medicine. When the user presses the button, the system waits for them to remove the container from the scale platform. Once it is removed, the persistent memory is updated, and the caregiver is notified. If the container is not removed, it is eventually added back into the medicine database.

5.2.4 Checking Out a Medicine

A medicine can be temporarily removed, or "checked out," so that the user can take it while they are away from home. To check a medicine out of the system, the same procedure as permanent removal is used, only the red button is pressed instead of the black button. This is the same procedure used when taking a medicine early, so it is important to note that when a medicine is eligible to be taken early, it must be taken before it can be checked out.

5.2.5 Taking a Medicine Normally

Medicine schedules are stored in one to four daily dosage slots, specifically 8:00am, 2:00pm, 8:00pm, and 2:00am. At each of these times, the system triggers a software interrupt that checks if any medicines should be taken. If so, the container is weighed, the user is alerted via text message, the alarm is sounded, and the green LED is activated to indicate that the system is ready to administer a medicine. If a particular medicine container cannot be weighed correctly, its dosage is skipped and the caregiver is alerted.

When the user approaches the device, the active medicine container is positioned at the front (the scale platform), and the medicine information is displayed on the user interface panel. When the user picks up the container, the panel shows how many units of the medicine he should take. The user can then attempt to take the described dosage and replace the container on the scale platform.

If the user has taken the right amount of medicine, no further action is required. If more needs to be taken, the display will instruct the user as to how many more units should be taken until either the user takes the remaining units, or fails three times. If the user does not take the right amount of a medicine, the caregiver is alerted via text message, and the display notifies the user of the failure. In all cases, once a medicine is done being administered, its final weight is recorded and the system moves on to the next medicine that should be taken in the current time slot until no more remain. Once all the medicines have been processed, the persistent memory is updated.

In addition, before any medicine is administered, it is weighed and compared to the most recently recorded weight. If any difference is detected, the caregiver is alerted. This is done in case the patient takes a medicine from the device before it is supposed to be administered.

5.2.6 Taking a Medicine Early

The user can take a medicine as early as one hour before the actual dosage time occurs. When any medicines are to be taken within one hour, the idle screen displays the number of medicines that can be taken and the green LED is activated. If the user wants to take a medicine early, they follow the same procedure as removing the medicine, only they press the red button when prompted.

5.2.7 Taking a Checked Out Medicine

When a medicine is checked out of the system, the prototype will send the user a text message every time the checked out medicine is to be taken. The message contains all of the information normally printed to the user interface panel. When the user returns to the device, they simply place the checked out bottle on the scale platform. The prototype will then reweigh the bottle to make sure that the right amount was taken. If an overdose or underdose was detected, the caregiver is notified. Once the administration is complete, the persistent memory is updated.

5.2.8 Idling

When there is no user input and no medicine is to be taken, the prototype remains in an idle state. During this time, the user interface panel displays a screen that says “Ready” and the current time. If there are any medicines to be taken within the next hour, the number of these medicines is also displayed along with the activation of the green LED.

During the idle state, the system constantly checks the scale platform for user input, as well as listens for a software interrupt to indicate when it is time to take medicines. Additionally, once per day, it will identify each medicine container on the platform and add any medicine that is not in the medicine database. This is done in case the user adds a medicine to the platform, but forgets to identify it on the scale platform first. The system will also print its status to the USB connected serial terminal every five minutes. This is useful for software debugging as well as for a caregiver who wants to see which medicines the machine is tracking.

5.3 Data Protocols

This section describes the protocols used to transmit and store data in the prototype.

5.3.1 Communication with RFID Reader

The microcontroller communicates with the RFID reader over a 9600 bps UART connection. Each byte is sent with eight data bits, no parity, and one stop bit. This is the standard configuration for the Arduino Serial library, so no adjustments are required.

The RFID reader interprets and sends data according to the SkyeTek Protocol Version 2. The protocol is described in detail in [19]. Figure 5.2 shows the basic format of the data exchange, and Figure 5.3 and Figure 5.4 show the structure of the request and response datagrams. Requests and responses are transmitted as American Standard Code for Information Interchange (ASCII) characters. The prototype only implements a couple of the request commands, specifically “SELECT_TAG” to read the RFID tag’s UID, and “READ_TAG” to read the data stored on the RFID tag.

HOST :	**CR**	**REQUEST**	**CR**						
READER:				**execute command**		**LF**	**RESPONSE**	**CR**	**LF**

Figure 5.2: Basic data exchange format for RFID reader[14]

REQUEST										
MSG LEN	**FLAGS**	**COMMAND**	**RID**	**TAG TYPE**	**TID**	**AFI**	**STARTING BLOCK**	**NUMBER OF BLOCKS**	**DATA**	**CRC**
8 bits	8 bits	8 bits	8 bits	8 bits	64 bits	8 bits	8 bits	8 bits	n*8 bits	16 bits

☐ **Mandatory Fields**

☐ **Optional Fields**

Figure 5.3: Format of request datagram for RFID reader[15]

RESPONSE					
MSG LEN	**RESP**	**RID**	**TAG TYPE**	**RESP DATA**	**CRC**
8 bits	8 bits	8 bits	8 bits	n * 8 bits	16 bits

☐ **Optional Fields**

☐ **Mandatory Fields**

Figure 5.4: Format of response datagram for RFID reader

[14] Source: [19]

[15] Source: [19]

5.3.1.1 SELECT_TAG command

For the SELECT_TAG command, the command code is 14, and no flags are required. For this command, the value 01 is also sent in the "TAG TYPE" field to indicate that the tag type is ISO15693. The final request string sent to check a tag's UID is: *<CR>"001401"<CR>*, where <CR> represents the literal ASCII value of a carriage return, 0x0D.

The response from the SELECT_TAG command is 14 for a successful read, and 94 if no tag is present. If the read was successful, the response is immediately followed by sixteen characters representing the hexadecimal value of the tags eight-byte UID. For example, if no tag is present, the response string is: *<LF>"94"<CR><LF>*, where <LF> represents the literal ASCII value of a line feed, 0x0A. If a tag is present, the response string is: *<LF>"140011223344556677"<CR><LF>*, where 0011223344556677 represents the present tag's UID in hexadecimal format.

5.3.1.2 READ_TAG Command

For the READ_TAG command, the command code is 24, and the flag is set to 40 to indicate that the intended tag's UID will be sent. This is done to prevent reading data from the wrong tag. The TAG TYPE field is set to 01 again, followed by the UID of the intended tag. This is followed by a one-byte "STARTING BLOCK" field, and an one-byte "ENDING BLOCK" field that indicate which and how many four-byte blocks of data to read from the tag. The final request string sent to read a tag's data is: *<CR>"40240011223344556660007"<CR>*, where 00112233445566 is the hexadecimal value of the intended tag's UID, and blocks 00 through 06 are requested to be read.

The response from the SELECT_TAG command is 24 for a successful read, and A4 if the intended tag is not found or cannot be read. If the read was successful, the response is immediately followed by the requested data. For example, if no tag is present, the response string is: *<LF>"A4"<CR><LF>*. If a tag is present, the response string is: *<LF>"2400000000000000000000000000000000000000000000000000000000"<CR><LF>*, where the string of fifty-six zeros represents the hexadecimal value of the data stored in the tag's first eight blocks. It is important to note that the system that was used to write data to the tags wrote to the data blocks in the reverse order. Because of this, the blocks are read in a reverse order as seen in the *readTagData()* function in Appendix E.

5.3.2 Communication with Scale

The microcontroller communicates with the scale over a 9600 bps UART connection buffered by an RS232 level shifter. Each byte is sent with seven data bits, even parity, and one stop bit. This is not the default configuration of the scale, and it must be set this way in the settings menu before it can be interfaced with the microcontroller. This is not the standard configuration for the non-configurable Arduino Serial library either, so special care must be taken to enforce the required parity scheme.

The scale interprets and sends data according to its own proprietary protocol. This protocol is described in detail in [20]. Figure 5.5 and Figure 5.6 show the structure of the request and response datagrams. Requests and responses are transmitted as standard ASCII characters. The prototype only implements a couple of the request commands, specifically "ZERO" to zero the scale, and "PRINT" to get the current weight of object on the scale.

5.3.2.1 ZERO Command

For the ZERO command, the command code is T, which is ASCII hexadecimal value 0x54. The binary representation of this value has three 1's, so another 1 must be added to the beginning of the byte before it is sent. This is to enforce even parity. This can easily be done since the scale was set to expect seven data bits, and the Serial library transmits eight bits at a time. This transforms the command code to 0xDF. This transformation must also be done with the <CR> character, transforming it from 0x0D to 0x8D. The <LF> character contains an even number of 1's, so it does not need to be adjusted. The final request string sent to zero the scale is: *0xDF 0x8D <LF>*. The ZERO command does not induce a response from the scale.

5.3.2.2 PRINT Command

For the PRINT command, the command code is P, which does not need to be adjusted for even parity. The final request string sent to get the weight of the object on the scale is: *"P" 0x8D <LF>*.

The response from the PRINT command is a formatted version of the weight of the object on the scale. For example, the response string is: *"   + 9999.999 g " <CR><LF>*, where + is the sign of the weight, 9999.999 is the quantity of the weight, and g is the measurement unit.

Format for Control Commands						
Format 1 (upper-case letter):	Esc	!	CR	LF		
Format 2 (lower-case letter):	Esc	!	#	_	CR	LF

Esc: Escape (optional)
!: Command character
_: Underline
CR: Carriage return (optional)
LF: Line feed (optional)

Figure 5.5: Format of request datagram for scale[16]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I	I	I	I	I	I	+	*	D	D	D	D	D	D	D	D	*	U	U	U	CR	LF
*	*	*	*	*	*	–		.	.	.	.	.	.	.	.		*	*	*		
						*		*	*	*	*	*	*	*	*						

I: ID code character
*: Space
D: Digit or letter
.: Decimal point
U: Unit symbol
CR: Carriage return
LF: Line feed

Figure 5.6: Format of response datagram for scale[17]

5.3.3 Medicine Data Protocol

Medicine information is stored on each container's RFID tag. The following is a description of each item:

- **UID:** An eight-byte unique identifier is hardcoded into each tag from the manufacture.
- **Medicine Name:** Twenty characters of the medicine name are stored. Each character is stored as its ASCII value.
- **Medicine Schedule:** A dosage frequency between one and four times per day is stored along with how many units should be taken at each time, and the weight of an individual unit. All of these values are stored as integers.

---

[16] Source: [20]

[17] Source: [20]

- **Special Instructions:** Special instructions can be assigned to medicines, including "take with food," "take with water," "stay home," and "do not take with Aspirin." Each of these four cases is stored as an integer that is set to 1 for true and 0 for false.
- **Entry Method:** An entry method is assigned. This method can be oral swallow, oral chew, oral liquid, nasal, injection, eye, ear, or anal. This value is stored as an integer between 0 and 7, consistent with the order listed.
- **Most Recent Dosage Information:** The most recent dosage information is stored. This information includes the date of the most recent dosage, and the weight of the container after that dosage was given. All of these values are stored as integers.
- **Presence:** The presence of the medicine container on the device is stored. This value is stored as an integer that is set to 1 for present and 0 for checked out.

5.3.4 Tag Data Organization

The medicine information stored on each tag is reasonably compressed into 26.5 bytes in order to fit all of the information into the tag's 32-byte capacity. For this reason, many values, such as the binary flags and small numerical values, are stored as half-bytes. Figure 5.7 describes how the data is organized into the tag's data block sections. Two full sample medications are listed in APPENDIX C:.

Field	Type	Range	Size (Bytes)	Position
Name	20x ASCII Character	[0, 255]	20	[00.00, 19.5]
Frequency	Integer	[1, 4]	0.5	20.0

Unit Quantity	Integer	[1, 15]	0.5	20.5
Unit Weight	6x Integer	[0, 9]	3	[21.0, 23.5]
Food Flag	Integer	[0,1]	0.5	24.0
Water Flag	Integer	[0,1]	0.5	24.5
Stay Home Flag	Integer	[0,1]	0.5	25.0
No Aspirin Flag	Integer	[0,1]	0.5	25.5
Entry Method Code	Integer	[0, 7]	0.5	26.0

Figure 5.7: Table describing medicine data organization of RFID tag

5.3.5 Main Memory Data Organization

The medicine information stored in main memory is expanded into whole bytes for easier access. These bytes are stored in a two dimensional array, *medicineData*. The array is long enough to hold $n - 1$ medicines, where n is the number of slots on the prototype. It is also wide enough to store each medicine's data in a 60-byte row. Figure 5.8 describes how the data is organized into the *medicineData* array.

Field	Type	Range	Size (Bytes)	Position
UID	16x ASCII Character	[0,255]	16	[0, 16]
Name	20x ASCII Character	[0,255]	20	[16, 35]
Frequency	Integer	[1, 4]	1	36
Unit Quantity	Integer	[1, 15]	1	37
Unit Weight	6x Integer	[0, 9]	6	[38, 43]
Food Flag	Integer	[0,1]	1	44
Water Flag	Integer	[0,1]	1	45

Stay Home Flag	Integer	[0,1]	1	46
No Aspirin Flag	Integer	[0,1]	1	47
Entry Method Code	Integer	[0, 7]	1	48
Most Recent Dose Year	Integer	[0, 99]	1	49
Most Recent Dose Month	Integer	[1, 12]	1	50
Most Recent Dose Day	Integer	[1, 31]	1	51
Most Recent Dose Time Code	Integer	[1, 4]	1	52
Presence Flag	Integer	[0, 1]	1	53
Most Recent Weight	6x Integer	[0, 9]	6	[54, 59]

Figure 5.8: Table describing medicine data organization in main memory

5.3.6 EEPROM Data Organization

Data is also stored in the microcontroller's persistent electrically erasable programmable read-only memory (EEPROM). The EEPROM keeps an uncompressed backup of the *medicine Data* array, the user's settings, and a special validation sequence. The validation sequence is simply the character string "PMM" stored at the end of the data. This data is stored in the linear address space of the EEPROM starting at address 00. Position is given in terms of the constant NUM_SLOTS, which is the number of slots on the device. Figure 5.9 describes how the data is organized in the EEPROM.

Item	Size (Bytes)	Position
Backup of *medicineData*	(NUM_SLOTS - 1)*60	[0, {(NUM_SLOTS - 1)*60 – 1}]
Backup of Patient's Phone	10	[(NUM_SLOTS - 1)*60, (NUM_SLOTS - 1)*60 + 9]
Backup of Caregiver's Phone	10	[(NUM_SLOTS - 1)*60 + 10, (NUM_SLOTS - 1)*60 + 19]
Backup of DST Flag	1	(NUM_SLOTS - 1)*60 + 20
Backup of Time Zone Code	1	(NUM_SLOTS - 1)*60 + 21
Validation Sequence	3	[(NUM_SLOTS - 1)*60 + 22, (NUM_SLOTS - 1)*60 + 24]

Figure 5.9: Table describing data organization of the EEPROM

5.4 Description of Source Code

The software code is written in Arduino, the C++-like language associated with the Arduino hardware platform. This section describes the flow of the code, and elaborates purpose and algorithms behind several key functions. The final code is fully commented, and is listed in [21], as it is too lengthy to include in the Appendix.

5.4.1 Software Flow

The code begins with the declaration of external libraries, constants, and global variables. Among these are the unit's MAC address, the IP address of the network time server, and the IP address of the outgoing mail server. The *setup()* function is then called by the bootloader to run once. *setup()* configures all of the I/O ports, initializes the *medicineData* array, runs the initialization sequence, and sets the software interrupt timers. These timers are provided by the *Time* library, and allow the system to periodically call a function. For example, this software periodically calls the function that administers medicines.

After *setup()* is complete, the bootloader calls the *loop()* function which repeats indefnately. *loop()* constantly updates an idle screen by calling *displayReady()* every time *loop()* iterates. In addition to calling *displayReady()*, *loop()* also checks for the presence of a medicine container by calling *handlePlatformInput()*, and checks for a software interrupt by calling *Alarm.delay()*.

If *handlePlatformInput()* detects the presence of a medicine container, the function determines the appropriate action to take. If the medicine is not recognized in *medicineData*, it attempts to add it by calling *addNewMedicine()*. If the medicine is recognized as checked out, it attempts to check it in by calling *checkInMedicine()*. If the medicine is recognized, but not checked out, the user is prompted to choose to either remove the medicine by calling *removeMedicine()*, check out the medicine by calling *checkOutMedicine()*, or take the medicine early by calling *takeMedcine()*. Which options are available depends on how much time is left before the next dosage.

If *Alarm.delay()* detects a software interrupt, it branches to either *medicineTime()*, *printStatus()*, or *checkPlatform()*, depending on which timer triggered the interrupt. *MedicineTime()* determines the current time code and administers all untaken medicines by successively calling *takeMedicine()*. It also removes any medicines that have been checked out for too long. If caregiver or user attention is required, a remote message can be sent using *sendNotification()*. *printStatus()* simply prints the status of the machine to a

USB-connected serial terminal for debugging. *checkPlatform()* loops through each medicine to check if any unauthorized dosages were taken or new medicines added.

Function	Location (Line Number)
setup()	101
loop()	170
displayReady()	2,046
handlePlatformInput()	2,415
Alarm.delay()	19
addNewMedicine()	1,522
checkInMedicine()	2,520
removeMedicine()	2,940
checkOutMedicine()	2,689
takeMedicine()	3.037
medicineTime()	3,629
printStatus()	3,772
checkPlatform()	3,859
sendNotification()	1,895

Figure 5.10: Locations of major functions in source code

5.4.2 Additional Function Explanations

The following sections elaborate on select functions that were not previously explained.

5.4.2.1 setTime()

setTime() is located at line 285, and is responsible for setting the system time. This is done by first getting the time zone and daylight savings time data from the user via the user interface panel or a previously stored value. The microcontroller then activates the network interface unit to obtain an internet protocol (IP) address via dynamic host configuration protocol (DHCP), and fetch the time from an Internet time server using *getNtpTime()*. For the prototype, the time is fetched from the time server at the National Institute of Standards and Technology (time.nist.gov / 192.43.244.18). Once the time is fetched, it is converted to the local time using the time zone and daylight savings time data given previously. If the time cannot be fetched, the system is halted. This is because the cause of failure was likely an unreliable or inactive network connection, which makes the system unusable.

5.4.2.2 refreshClock()

refreshClock() is located at line 1,070, and is responsible for displaying the time on the user interface panel. The function simply prints the formatted time to the bottom right corner of the screen. The format of the time is HH:MM$m, where HH represents the hour in twelve-hour format, MM represents the minute, and $ represents a or p, depending if it is currently morning or evening. *refreshClock()* is called very frequently by almost every function in order to insure that the value of the displayed clock is always current. Generally, the function is called in any loop that runs for an undetermined number of iterations.

5.4.2.3 setupDatabase()

setupDatabase() is located at line 1,285, and is responsible for loading all medicine information into the *medicineData* array when the device is powered on. It is run in the initialization sequence right after the data from the persistent memory has been read in. The following algorithm is implemented:

1. Initialize *medicineData* array to all blanks by setting the first byte of each row to 0x20.
2. If no previous medicine data was found on persistent memory, initialize the EEPROM to all blanks.
3. Load the EEPROM contents into the temporary array *tempMedicineData*.
4. Copy all checked out medicines in *tempMedicineData* to *medicineData,* then remove any medicines that have been checked out for longer than seven days.
5. Search the tray platform for medicines containers. Add any unknown medicines, and check in any known medicines.
6. Search through *tempMedicineData* for medicines that are not present in *medicineData*, and alert the caregiver of any missing medicines.
7. Update the EEPROM with the newly compiled *medicineData* array.

5.4.2.4 sendNotification()

SendNotification() is located at line 1,895 and is responsible for sending an email-based text alert to a patient or caregiver's cellphone. This is done by setting up a Telnet connection to a Simple Mail Transfer Protocol (SMTP) server. The SMTP server can then be used to send an email to a special address that is forwarded to a cellphone number. This model implies a couple of important restrictions.

Since the microcontroller is not as powerful as a traditional desktop computer, processor-intensive standard authentication technologies cannot be used. Most SMTP servers on the Internet require this type of authentication to prevent unauthorized users from sending messages. The prototype is connected to Yahoo's special SMTP server for mobile devices, which allows its clients to authenticate using pre-encoded credentials that can be calculated on a computer. For this reason, all messages from the prototype are sent via coreysmccall@yahoo.com, with password "google". The username and password must be encoded using the base64 encoding standard listed in RFC 2045. A sample conversation with the Yahoo SMTP server is shown in Figure 5.11, and the corresponding text message is shown in Figure 5.12. The bolded characters represent outgoing data.

The Yahoo SMTP server can send emails, not cellular phone text messages. For this reason, the phone number entered by the user is concatenated with the string "@txt.att.net". This will automatically forward the message to the given cellphone as a text message. It is important to note that this will only work for AT&T subscribers.

```
Steven:~ coreymccall$ perl -MMIME::Base64 -e 'print
encode_base64("\000coreysmccall\@yahoo.com\000google")'
AGNvcmV5c21jY2FsbEB5YWhvby5jb20AZ29vZ2xl
telnet 98.136.86.109 587 Trying 98.136.86.109...
Connected to smtp1-mob.biz.mail.vip.ac4.yahoo.com.
Escape character is '^]'.
220 smtp108-mob.biz.mail.ac4.yahoo.com ESMTP
EHLO MAILSERVER
250-smtp108-mob.biz.mail.ac4.yahoo.com
250-AUTH LOGIN PLAIN XYMCOOKIE
250-PIPELINING
250 8BITMIME
```

```
AUTH PLAIN AGNvcmV5c21jY2FsbEB5YWhvby5jb20AZ29vZ2xl
235 OK, go ahead
MAIL FROM: <COREYSMCCALL@YAHOO.COM>
250 OK , completed
RCPT TO: <9545793517@TXT.ATT.NET>
250 OK , completed
DATA
354 Start Mail. End with CRLF.CRLF
FROM: PATIENT <COREYSMCCALL@YAHOO.COM>
TO: <9545793517@TXT.ATT.NET>
SUBJECT: PATIENT OVERDOSE

YOUR PATIENT HAS OVERDOSED ON 'NAPROXEN'
.
250 OK , completed
```

Figure 5.11: Sample conversation with the Yahoo mobile SMTP server

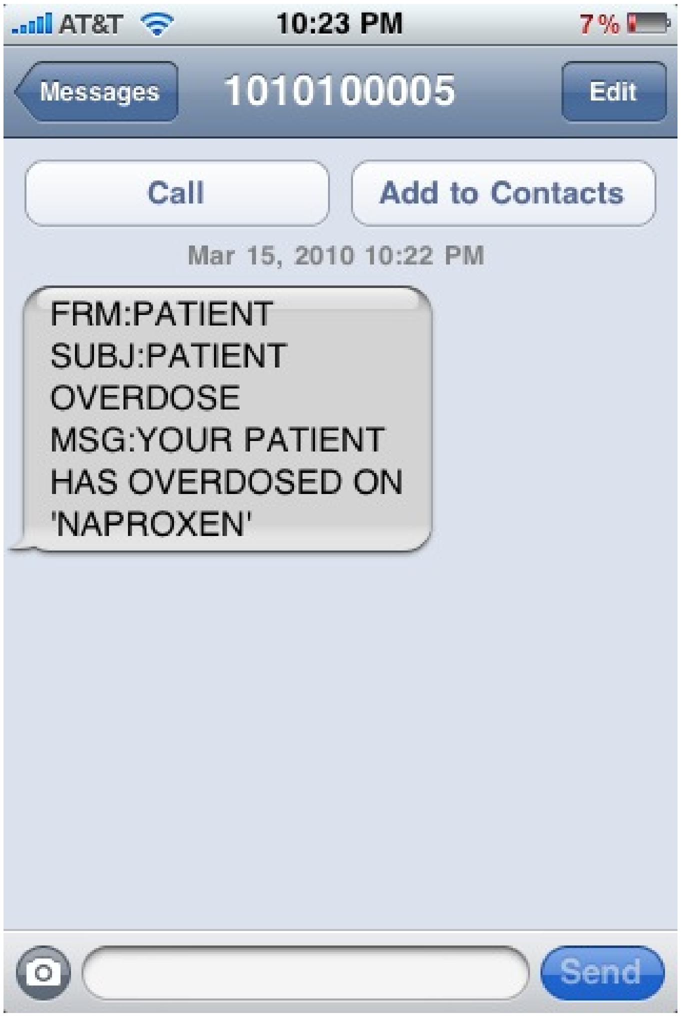


Figure 5.12: Screenshot of a text message received by the prototype

Summary

This chapter describes the prototype software concept, user interface, data protocols, and code. The software was developed according to the approach mentioned in Chapter 3, and is designed in a way that mimics the traditional medicine taking procedure. It was able to be written quickly, since it was written in the high-level Arduino programming language. The code is explained in detail and is listed in its entirety in Appendix E.

CHAPTER 6: PROTOTYPE EVALUATION

This chapter describes the evaluation of the prototype. The first section, Requirements Analysis, reviews the requirements in the context of the prototype. Section 2, Component Testing, describes the testing of key prototype components. Section 3, Functionality Gaps, describes unresolved functionality that is necessary to implement the system. The final section, Future Testing, is a list of uncompleted full system tests. Aside from the components in Section 2, the final prototype device is largely untested. The evaluation discussed in this chapter is limited to specific components and an analysis of the prototype design. The task of an exhaustive system test is left to future work.

6.1 Requirements Analysis

This section reviews the requirements listed in Chapter 3 in the context of the prototype.

6.1.1 Intuitive User Interface

An intuitive user interface is implemented in the prototype hardware and software design. The hardware of the prototype is built to mimic the style of a Lazy Suzan revolving shelf that is traditionally used to store medicines. Just as a normal shelf, medicines can be easily accessed on top of device.

The software of the prototype is also designed with an intuitive user interface that mimics the traditional medicine management procedure. For example, for a user to take a medicine, they simply pick up the container, take the dosage on the label and LCD screen, and place it back down on the device. No manual input is required. The other non-traditional functions of the device are designed to require the very minimum number of button presses.

6.1.2 Medicine Container Distinction

The prototype is able to distinguish individual medicine containers by rotating them across the platform to the front of the device using the section divider. This makes it obvious to the patient as to what medicine to take, because the correct medicine is always in the front of the device.

6.1.3 Dosage Intake Monitoring

Dosage intake monitoring is accomplished by the prototype using a scale integrated into the platform. The scale weighs each medicine container before and after the patient consumes a medicine dose in order to determine exactly how much he has ingested.

6.1.4 Dynamic Dosage Scheduling

The prototype implements optimized dynamic scheduling by parsing each medicine dosage frequency into one of four dosage times. For example, if a medicine is to be taken once, it is taken at 8:00am. If a medicine is to be taken twice, it is taken at 8:00am, and 8:00pm. If a medicine is to be taken three times, it is taken at 8:00am, 2:00pm, and 8:00pm. If a medicine is to be taken four times, it is taken at 8:00am, 2:00pm, 8:00pm, and 2:00am. With this static schedule, an optimal schedule is always created with the least amount of total dosage times.

6.1.5 Unexpected Input Handling

Unexpected input from the user is handled automatically by the prototype. The system checks for unexpected input at two times. First, before each medicine is administered, its current weight is compared to its weight after its most recent administration. If the container has not been tampered with, the weights should be the same. If medicine has been added or removed, it is recognized and reported by the system.

Unexpected input is also checked on a daily basis when each container on the tray is compared to the list of known containers. If any new containers are present, or known containers are missing, the system recognizes and reports them.

6.1.6 Pharmacy Integration

The prototype does not include a pharmacy integration component. However, the data protocol for the medicine container RFID tags is given. This data protocol should be implemented into custom integration software at participating pharmacies. For testing purposes, pharmacy compliance was simulated by manually writing valid data to the tags.

6.1.7 Direct Communication with Caregiver

The prototype is able to directly communicate with the caregiver using a network connection that is configured to send a cellphone text message. When the prototype device is initialized, the caregiver's phone number is given by the user, and saved to persistent memory. This allows the device to contact the caregiver if a noncompliance incident is detected.

6.1.8 Remote Care

Remote care is implemented in the prototype software by allowing the patient to "check out" medicines for up to one week. Checked out medicines do not instigate local alarm notifications on the device, but rather, they send full medicine dosage instructions to the patient's cell phone. Compliance is checked when the patient returns the containers to the device, where they are reweighed and the taken dosage calculated.

6.2 Component Testing

The following sections describe how the scale and RFID reader component integration was tested. The testing of these two components is essential because they are at the core of the system's functionality and were not designed specifically for this system as the other custom components were.

6.2.1 Scale

The scale was tested to insure that it is able to detect the displacement of any size pill, in any quantity. This was done by measuring the displacement of 1, 2, 3, 4, and 5 pills from an initial quantity of 5 and 250. This test was repeated for pills of 150mg each and 1500mg each. For the 1500mg pills, the maximum initial quantity was lowered from 250 to 50 so that a reasonable sized container could be used. The test was 100% successful. Figure 6.1 displays the results.

Condition	1 Pill	2 Pills	3 Pills	4 Pills	5 Pills
150mg, 5 pills	Pass	Pass	Pass	Pass	Pass
150mg, 250 pills	Pass	Pass	Pass	Pass	Pass

1500mg, 5 pills	Pass	Pass	Pass	Pass	Pass
1500mg, 50 pills	Pass	Pass	Pass	Pass	Pass

Figure 6.1: Table describing results of the scale test

6.2.2 RFID Reader

The RFID reader was tested to insure that it is able to detect a medicine container no matter where it is placed on the scale platform. This test was done by repeatedly reading the RFID reader's response as a medicine container was manually moved around the scale platform. The results showed that the container could only be read if the edge of its RFID tag was within 0.3 inches of the edge of the RFID reader. Because of the position of the reader (as shown in Chapter 4: and Appendix A), and the limited peripheral range of the antenna, a large portion of the scale platform us unreadable. This is a significant issue that should be addressed in future iterations of the prototype. To improve the peripheral range, the RFID reader may be moved downward in its mounting hole, or a larger antenna may be used. Figure 6.2 is a visual approximation of the unreadable area. Any tag that is completely enclosed in the shaded area cannot be read.

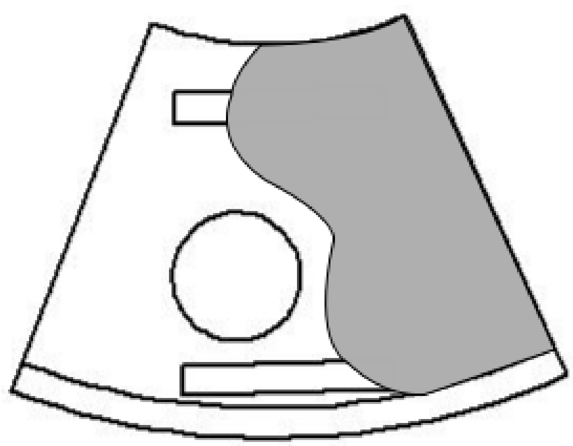


Figure 6.2: A visual representation of the unreadable area of the scale platform

6.3 Functionality gaps

The research presented covers a large portion of the system described in Chapter 3. However, the system is not fully developed, and functionality gaps were discovered during the evaluation. The following sections describe the three main functionality gaps.

6.3.1 Pharmacy Software and Integration

The pharmacy software and integration is outside of the scope of the research presented, although still essential to system functionality. In order for the system to be to be adopted, special software must be developed to integrate with the current pharmacy information system. The software should encode the printed medicine label's information on each container's tag according to the protocol described in Chapter 5. For testing purposes, pharmacy cooperation was simulated by manually encoding tags with medicine data using an RFID writer.

6.3.2 RFID Antenna Sensitivity

Upon testing the RFID reader circuit's integrated antenna, it proved to be inadequate for the prototype designed. In order to provide a reliable response to medicine container input, a custom antenna should be built that better suits the exact specifications of the particular device model. Unlike the current antenna that cannot detect medicine in certain areas of the scale platform, the revised antenna should be able to consistently read and only read a medicine container positioned in any part of the scale platform.

6.3.3 Section Divider Initialization Mechanism

The section divider of the current prototype does not have an initialization mechanism. Without this function, it must be manually aligned with the platform sections each time the device is powered on. If it is aligned incorrectly, it will never correct itself because the stepper motor that it is attached to can only control its relative position. An initialization mechanism should be built to accomplish this function. The design may consist of simply a small switch placed on the intersection of the inner rim of the platform retaining wall and the edge of one of the sections. If the switch is triggered when a section divider spoke passes by it, the microcontroller can be notified that the section divider is currently aligned.

6.4 Future Testing

The following sections describe tests that should be performed on the prototype before it progresses to its next iteration if further work is to be done. The tests were designed before prototype construction began, and are categorized in terms of device usability and reliability.

6.4.1 Usability Tests

Usability is determined by ability of uninvolved users to interact with the device, and the device's ability to intelligently manage medication schedules. The following tests should be conducted.

1. The user inputs a new medicine container into the device. The device should then save the medicine into the system and begin providing notifications.

2. The user permanently removes a medicine container from the device. The device should then remove the medicine from the system and discontinue providing notifications.

3. The user checks out a medicine container from the device. The device should then discontinue providing local notifications, but continue to provide remote notifications.

4. The user places a checked out medicine container on the device. The device should then start to provide local notifications.

5. When notified, the user consumes a medicine dose, and correctly replaces the container. The device should then clear the scale platform of the medicine container, and replace it with any remaining medicines to be taken.

6. When notified, the user consumes too much of the medicine. The device should alert the user and a remote caregiver.

7. When notified, the user consumes too little of the medicine. The device should prompt the user to consume more. If the user does not comply, the device should alert the user and a remote caregiver.

8. When notified, the user ignores the device. The device should alert a remote caregiver.

9. When notified, the user attempts to consume multiple medicines in sequence, because their scheduled consumption times are identical. The device should administer each medicine sequentially.

6.4.2 Reliability Tests

Reliability is determined by the ability of the device to function correctly and as expected according to the specifications mentioned in CHAPTER 3:. The following tests should be conducted.

1. The device functions for an extended time (greater than twenty-four hours). The device should provide notifications for all enrolled medicines during this time.

2. The user refuses to correctly replace the medicine container after consuming the dosage. The device should prompt the user to replace the container. If the user continues to be noncompliant, the device should alert the user and a remote caregiver.

3. The device is reset by pressing the reset button. The device should then rescan each of the medicines, and reconfigure itself automatically without further user interaction.

4. The device is arbitrarily disconnected from power. The device should then load recent activity and medicine information from persistent memory, rescan each medicine bottle, run the automatic configuration, and resume functioning normally. If any notifications have been missed, a remote caregiver should be alerted.

Summary

This chapter described the evaluation of the completed prototype. The prototype as a whole remains largely untested, with the exception of the scale and RFID reader. These two components were tested because they are at the core of the system’s functionality and were not designed specifically for this system. The scale worked perfectly, however the RFID reader suffered from a large section of the scale platform that could not be read. If further research is to be done , the issue with the RFID reader should be addressed in the next iteration of the prototype. Because the device remains largely untested, several test cases with expected outcomes are also included. These cases test the device in terms of its usability and reliability, and should be satisfied before the device is put into production.

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

This chapter concludes the thesis with a summary, a list of further work to be done, and an assessment of the completed project. The first section, Project Summary, summarizes the project in this thesis. Section two, Future Work, outlines suggested future work that should be done to progress the project. The final section, Conclusion, assesses the outcome of the thesis in terms of the initial goals.

7.1 Project Summary

In this thesis, a system is developed to track personal medication consumption and provide appropriate notifications to patients and caregivers. The system is based on a device developed in previous research done by Intel's Proactive Health lab in [2], in which certain features were added in order to enable the device to be used by patients who live independently. These features include a more intuitive hardware and user interface design, the ability to directly integrate with the healthcare system at the pharmacy level, the ability for the patient to interact with the device remotely, and the ability for a caregiver to passively monitor the patient's medication intake using a cellphone. With these enhancements, the system can be operated independently by the patient as a replacement to their traditional medicine cabinet.

The user interface improvements include hardware and software features. The prototype hardware is designed to mimic a traditional Lazy Suzan medicine shelf. The medicines are stored on top of the device on an enclosed tray with a "section divider" that can be controlled by the system to reposition the medicine containers. When a medicine is to be taken, the corresponding container can be moved to the front of the device so that the user does not have to search for it. The software is intuitively designed to mimic the traditional medicine taking procedure. In the normal case, the procedure to take medicine is simply to pick up the medicine container, take the stated dosage, and replace it on the tray.

The system relies on prescription schedule information encoded on an RFID tag attached to medicine bottles at the pharmacy. This allows the patient to automatically add new medicines to the device without any help from a caregiver. CHAPTER 5: describes the protocol for storing medicine information on the tags, but the actual software to implement the protocol at the pharmacy is not developed in this project.

Patients can also use the system while away from the medicine storage device. This is done by allowing the user to manually "check out" individual medications. When it is time to take a medicine that is checked out, the patient receives a cellphone text message with dosage instructions. Compliance is later verified when the medicine is placed back on the device.

The system also implements passive remote monitoring of the patient. Whenever the medicine inventory is changed (a medicine container may be added, lost, or removed), the caregiver receives an alert via cellphone text message. The caregiver also receives an alert when the wrong amount of a medicine is taken, or a dosage is ignored. Since the caregiver is only alerted when their attention is required, caregiver resources can be scheduled very efficiently.

The electronics and software of the prototype were developed using Arduino, a rapid prototyping microcontroller platform. This allowed the system to be prototyped very quickly, and allowed more time to refine the physical hardware components and incorporate more features. If the device is ever manufactured in a significant quantity, custom circuit boards and hardware interfaces should be developed.

The prototype is never fully tested in this thesis. Before further development is done, the system should be refined to pass the suggested tests given in Chapter 6. The full hardware and software designs are published with this thesis so that the prototype can be reproduced and further development can be done.

7.2 Future Work

The following sections describe suggested improvements to the system to be included in future work.

7.2.1 Abstract User Interface

The user interface of the prototype was designed to mimic traditional medicine taking procedures. However, the user interface still heavily relies on the LCD screen, which may be uncomfortable or difficult for an elderly patient to read. The LCD screen can be replaced with a more abstract output such as simple LED lights. For example, when medicines are to be taken, the corresponding sections would glow a particular color. Audible dosage instructions would then be given using a text-to-speech component.

7.2.2 Individual Medicine Monitoring

Individual medicine monitoring can be done to determine when a patient is low on a particular medicine or when one or more medicines are known to conflict. This would add more convenience and safety to the current system.

7.2.3 Cellular-based Networking

The current system relies on an Ethernet-based Internet connection to transmit messages to the caregiver and remote patient. This is inexpensive, buy very risky considering that the uptime of a patient's home Internet connection cannot be guaranteed. The Ethernet-based networking interface can be replaced with a cellular-based interface. This would provide a more reliable network connection to set the system time, and an almost perfectly reliable way to send SMS text messages.

7.2.4 Hardware Clock Updating

The user interface panel includes a clock that is constantly displayed in the corner of the LCD screen. In the prototype, the clock is manually updated by calling the *refreshClock()* during wait loops. This method is functional, but it does not guarantee the accuracy of the clock, and adds unnecessary complexity to the source code. The multiple function calls can be replaced with a single interrupt service routine that is run whenever a hardware timer is triggered (for example, at 1Hz). This would keep the clock constantly updated, and eliminate the numerous references in the source code.

7.2.5 Aperiodic Dosage Schedules

The current software and medicine data protocol are restricted to medicines being taken periodically on a daily basis. For example, medicines can only be set to be taken one, two, three, or four times per day. Many medication schedules are more complex than this. The software and medicine data protocol can be redesigned to allow for aperiodic dosage schedules. This would allow the system to be compatible with more medicines.

7.2.6 Remote Dosage Adjustment

EMMA, the in-home medication administration device described in [11], allows doctors or caregivers to remotely adjust medicine dosage schedules using the device's Internet connection. In order to adjust dosage schedules on the prototype, the medicine must be removed, relabeled by the pharmacy, and reinitialized into the system. The software can be redesigned to allow abrupt dosage adjustments by the remote caregiver. This would allow the system to be useful to patients with changing dosage schedules.

7.2.7 Network Multiple Machines Together

The in-home medication administration device described in [4] allows the system to be expanded by adding multiple wirelessly networked medicine trays. This is great for users with many medicines, or users who want to take medicines in different locations, but the complexity added by the additional tray is not worth the decrease in reliability. The prototype can implement a similar feature by allowing multiple independently-functioning devices to be networked together to act as a single system. This would allow the system to be used with a larger number of medicines without sacrificing reliability, since the individual devices could function independently if necessary.

7.3 Conclusion

The system that resulted from this research was designed according to previous work and research done on in-home healthcare, RFID in healthcare, and in-home medication administration devices. The novel features of the system, such as the hardware interface designed for the scale, the section divider mechanism used to position medicine containers, and the software protocols used to interface the system with the pharmacy provide a significant contribution to the research that has already been done in this area. The availability of the published prototype hardware and software components will also valuable to researchers if a similar or subsequent system is developed.

One significant problem was realized during the evaluation of the prototype. The scale platform, the hardware component that interfaces with the scale and RFID reader, was unable to detect medicine containers when they were placed in the region of the platform opposing the reader antenna. Because of the way that the reader is strategically positioned, read errors do not occur when the containers are positioned on the platform by the section divider. However, if the user places a medicine container on the platform, it is likely that it will be placed in the restricted region. In chapter 6, this issue is further described, and a solution is presented.

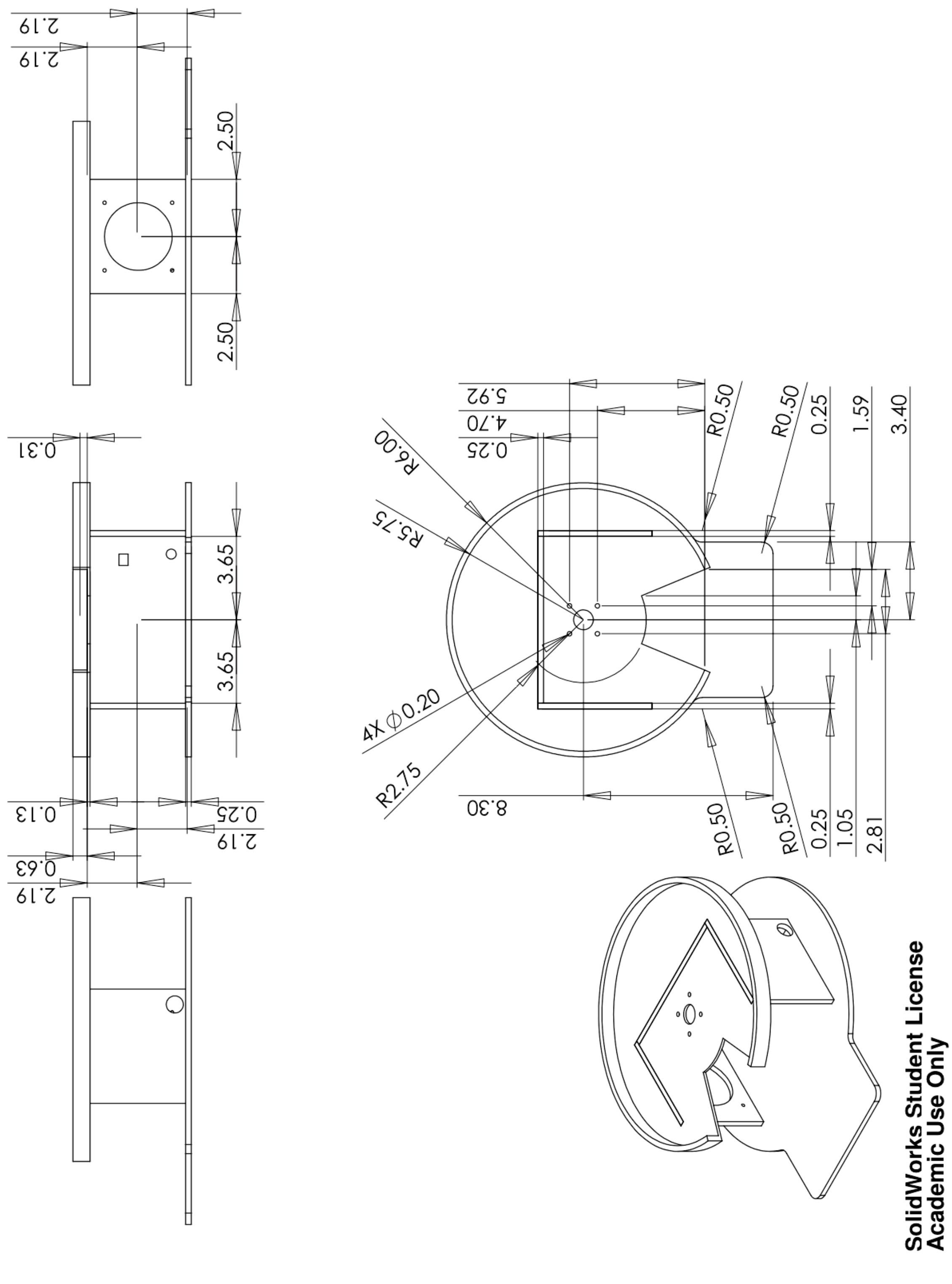
The effectiveness of the system relies on several assumptions. First, the device's user interface depends on the ability of the user to read text on the LCD screen, and effectively remove and replace the small medicine containers in the tray platform sections. Second, the device's remote monitoring and administration functions depend on an uninterrupted supply of power, an open connection to the Internet, and the messaging system used to deliver alert messages, specifically email and SMS. Third, the prototype is meant to prove the concepts in this research, and should not be immediately put into production. And finally, the design of the device hardware implies certain dependencies, specifically an Ethernet-based Internet connection, and medicine containers that are compatible with the device's medicine tray in terms of size and weight.

If this research is continued, the suggested system tests described in Chapter 6 should be conducted until the system has met satisfactory results. Once the tests have been completed, the system should be tested with humans as part of a clinical trial.

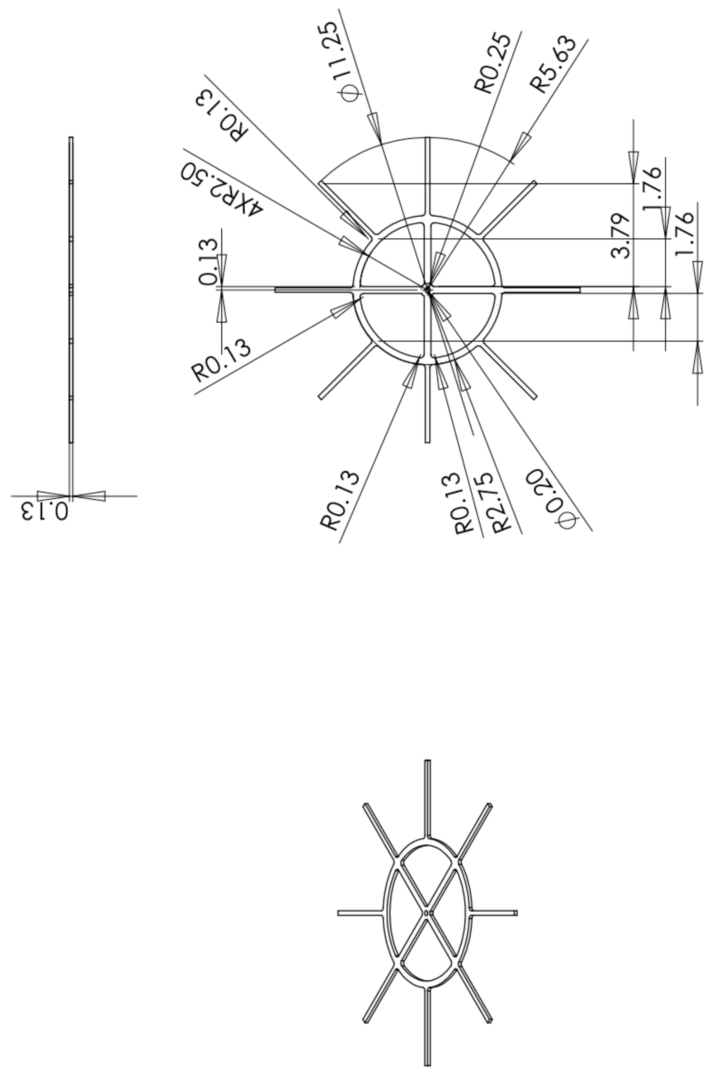
APPENDIX A:

CUSTOM PART DIMENSIONAL DRAWINGS

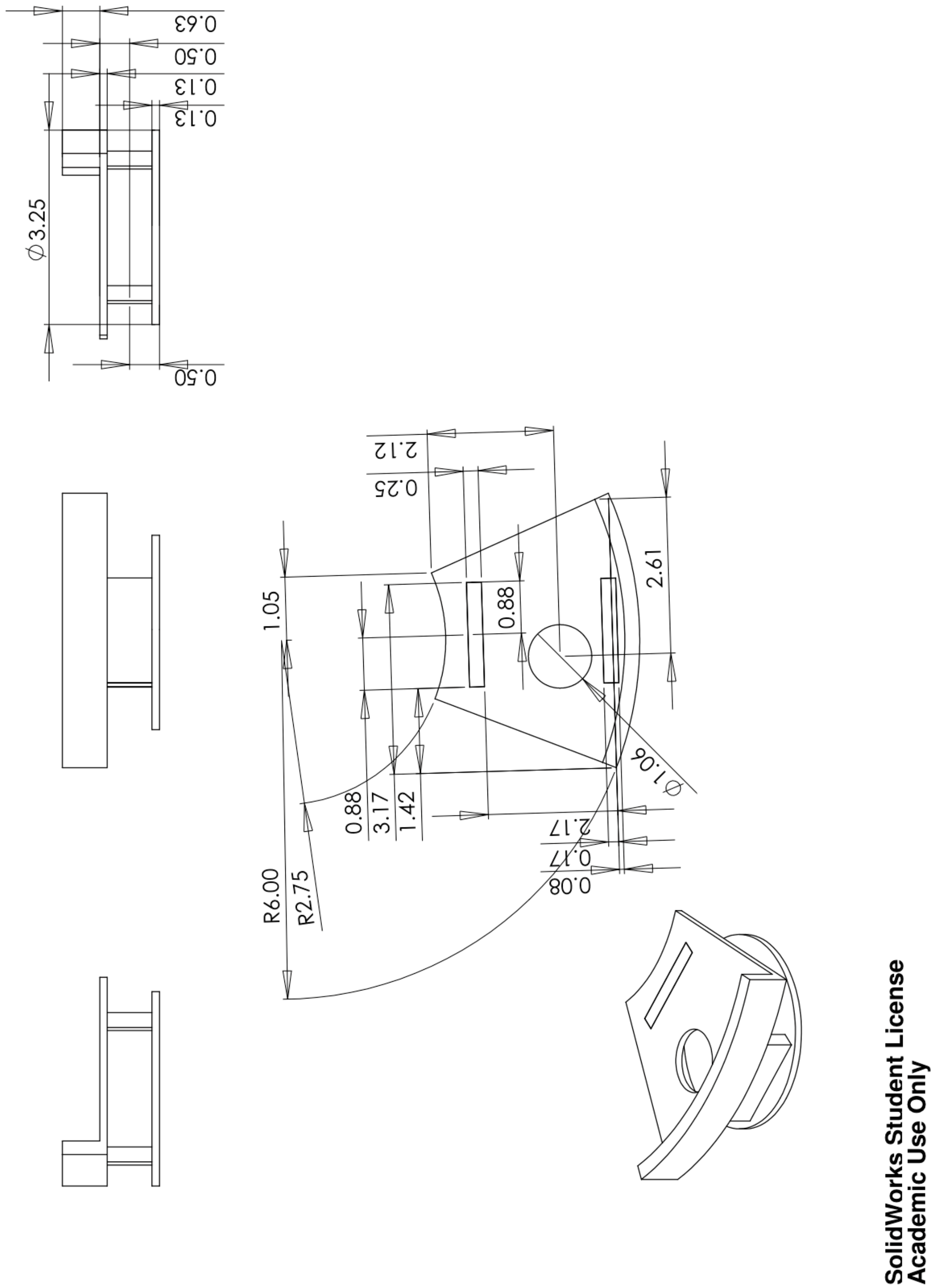
A.1 Main Chassis Assembly



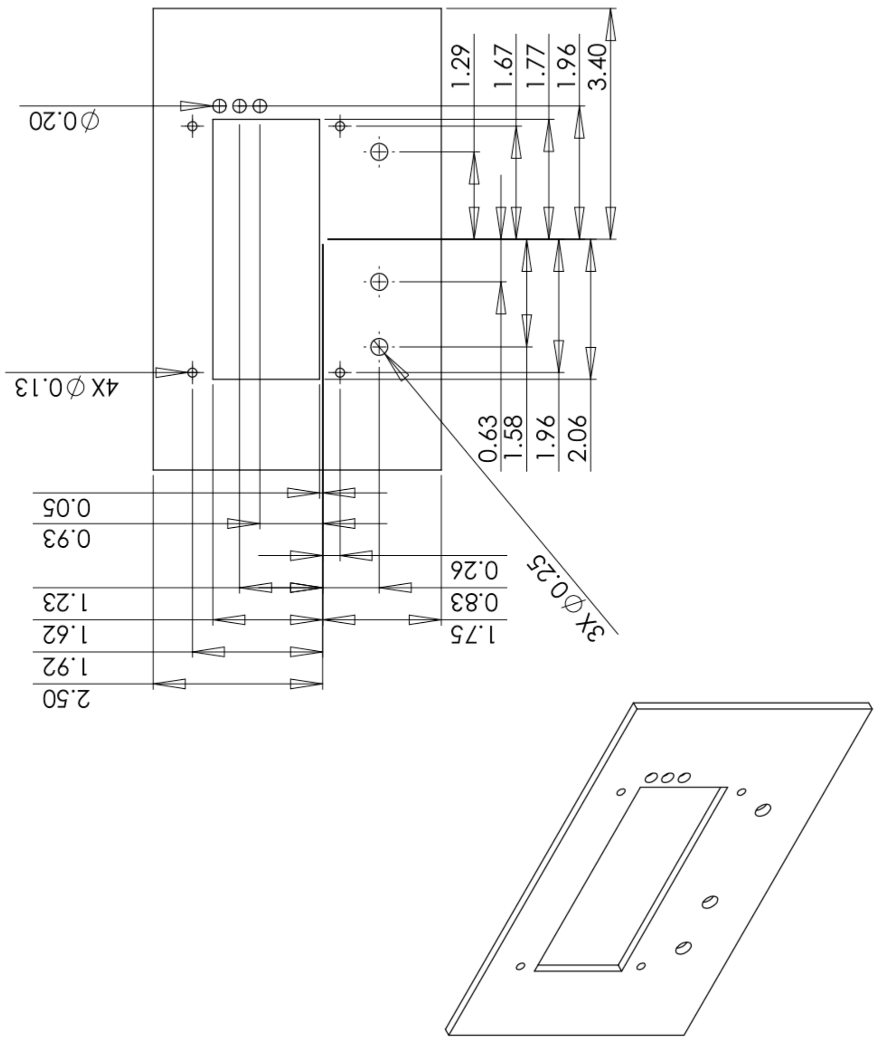
A.2 Section Divider



A.3 Scale Platform Assembly



A.4 User Interface Panel



APPENDIX B:

DETAILED BUDGET TABLE

Personal Medication Monitor Prototype Cost

does not include tax, shipping, or labor cost

*****includes equipment required at pharmacy***

Item	Price*	Source
System Components		
Digital Scale w/ Computer Interface	$354.95	http://www.itinscales.com/acculab_vicon_balances.htm
RFID Reader Circuit	$86.25	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=753-1009-ND
Microcontroller Board	$65.00	http://www.adafruit.com/index.php?main_page=product_info&cPath=17&products_id=191
Network Interface Card	$32.99	http://nkcelectronics.com/nkc-ethernet-shield-for-arduino-mega--duemilanove--diecimila-diy
x25 RFID Tag	$19.73	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=481-1099-1-ND
Power Adapter	$18.30	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=T381-P5P-ND
LCD Screen	$18.00	http://www.adafruit.com/index.php?main_page=product_info&cPath=37&products_id=198
Stepper Motor	$14.95	http://www.sparkfun.com/commerce/product_info.php?products_id=9238
Electronic Buzzer	$1.50	http://www.adafruit.com/index.php?main_page=product_info&cPath=35&products_id=160
TOTAL	$611.67	
Other Electronics		
Cooling Fan	$17.99	http://www.radioshack.com/product/index.jsp?productId=2102826
Power Supply Circuit	$15.00	http://www.adafruit.com/index.php?main_page=product_info&cPath=38&products_id=184
Stepper Motor Controller Board	$14.95	http://www.sparkfun.com/commerce/product_info.php?products_id=9402
Hookup Wire	$6.99	http://www.radioshack.com/product/index.jsp?productId=2049745
x2 Large Button	$6.78	http://www.radioshack.com/product/index.jsp?productId=2062496
x2 Prototyping board	$5.98	http://www.radioshack.com/product/index.jsp?productId=2102846
x3 Connector Shield	$5.97	http://www.radioshack.com/product/index.jsp?productId=2103993
Power Switch	$2.99	http://www.radioshack.com/product/index.jsp?productId=3016149
Power Plug	$2.99	http://www.radioshack.com/product/index.jsp?productId=2102486
Pin Headers	$2.50	http://www.sparkfun.com/commerce/product_info.php?products_id=116
4-pin Connector Male	$2.39	http://www.radioshack.com/product/index.jsp?productId=2103293
4-pin Connector Female	$2.39	http://www.radioshack.com/product/index.jsp?productId=2103250
9-pin Connector Male	$1.99	http://www.radioshack.com/product/index.jsp?productId=2102497

15-pin Connector Male	$1.99	http://www.radioshack.com/product/index.jsp?productId=2102601
15-pin Connector Female	$1.99	http://www.radioshack.com/product/index.jsp?productId=2102496
Solder	$1.95	http://www.sparkfun.com/commerce/product_info.php?products_id=9162
Serial Interface Chip	$1.95	http://www.sparkfun.com/commerce/product_info.php?products_id=316
x7 0.1µF Capacitors	$1.75	http://www.sparkfun.com/commerce/product_info.php?products_id=8375
Voltage Regulator	$1.59	http://www.radioshack.com/product/index.jsp?productId=2062599
Power Connector	$1.50	http://www.radioshack.com/product/index.jsp?productId=2103614
x3 LED	$1.05	http://www.sparkfun.com/commerce/product_info.php?products_id=533
Reset Button	$0.87	http://www.radioshack.com/product/index.jsp?productId=2062539
x3 10KΩ Resistor	$0.75	http://www.sparkfun.com/commerce/product_info.php?products_id=8374
x3 330Ω Resistor	$0.75	http://www.sparkfun.com/commerce/product_info.php?products_id=8377
TOTAL	**$105.05**	

Materials

Polycarbonate Plastic	$150.00	(estimate) http://www.professionalplastics.com/LEXANSHEET9034
Picture Frame	$9.99	http://www.target.com/Wood-Frame-Brown/dp/B0027UIL7U
Lazy Suzan Bearing	$1.49	http://www.vxb.com/page/bearings/PROD/LazySusan/Kit8882
x4 M3 screw	$1.12	http://www.lowes.com/pd_138543-37672-880729_0_?productId=3012884
x12 4-40 Screw w/ Nut	$1.02	http://www.lowes.com/pd_62032-37672-491275_0_?productId=3036645
x8 6-32 Screw w/ Nut	$0.68	http://www.lowes.com/pd_57977-37672-490413_0_?productId=3035920
TOTAL	**$164.30**	

Pharmacy Equipment**

RFID Writer	$130.49	http://apple.clickandbuild.com/cnb/shop/ftdichip?op=catalogue-product_info-null&productID
TOTAL	**$130.49**	

TOTAL	**$1,011.51**	

APPENDIX C:

RFID TAG DATA EXAMPLES

C.1 One a Day Vitamin

Field	*Value*	*Data*
Name	"One A Day Vitamin ”	<u>00.0 – 19.5:</u> 0x4F 0x6E 0x65 0x20 0x41 0x20 0x44 0x61 0x79 0x20 0x56 0x69 0x74 0x61 0x6D 0x69 0x6E 0x20 0x20 0x20
Frequency(per day):	1	<u>20.0:</u> 0x01
Unit Quantity	1	<u>20.5:</u> 0x01
Unit Weight (mg)	1500	<u>21.0 – 23.5:</u> 0x00 0x15 0x00
Food	1	<u>24.0:</u> 0x01
Water	1	<u>24.5:</u> 0x01
Stay Home	0	<u>25.0:</u> 0x00
No Aspirin	0	<u>25.5:</u> 0x00
Entry Method Code	0	<u>26.0:</u> 0x00
(Unused)	0	<u>26.5 – 27.5:</u> 0x00 0x00

Data Block (Address)	*Data*
00 (00 - 03)	0x4F 0x6E 0x65 0x20
01 (04 - 07)	0x41 0x20 0x44 0x61
02 (08 - 11)	0x79 0x20 0x56 0x69
03 (12 - 15)	0x74 0x61 0x6D 0x69
04 (16 - 19)	0x6E 0x20 0x20 0x20
05 (20 - 23)	0x11 0x00 0x15 0x00
06 (24 - 27)	0x11 0x00 0x00 0x00

C.2 Ibuprofen

Field	*Value*	*Data*
Name	“Ibuprofen ”	<u>00.0 – 19.5:</u> 0x49 0x62 0x75 0x70 0x72 0x6F 0x66 0x65 0x6E 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20
Frequency(per day):	4	<u>20.0:</u> 0x04
Unit Quantity	2	<u>20.5:</u> 0x02
Unit Weight (mg)	325	<u>21.0 – 23.5:</u> 0x00 0x03 0x25
Food	0	<u>24.0:</u> 0x00
Water	1	<u>24.5:</u> 0x01
Stay Home	0	<u>25.0:</u> 0x00
No Aspirin	0	<u>25.5:</u> 0x00
Entry Method Code	0	<u>26.0:</u> 0x00
(Unused)	0	<u>26.5 – 27.5:</u> 0x00 0x00

Data Block (Address)	*Data*
00 (00 - 03)	0x49 0x62 0x75 0x70
01 (04 - 07)	0x72 0x6F 0x66 0x65
02 (08 - 11)	0x6E 0x20 0x20 0x20
03 (12 - 15)	0x20 0x20 0x20 0x20
04 (16 - 19)	0x20 0x20 0x20 0x20
05 (20 - 23)	0x42 0x00 0x03 0x25
06 (24 - 27)	0x01 0x00 0x00 0x00

REFERENCES

[1] Ken Fishkin and Min Wang, "A Flexible, Low-Overhead Ubiquitous System for Medication Monitoring," Seattle, WA, 2003.

[2] Melody Moh, Loc Ho, Zachary Walker, and Teng-Sheng Moh, "A Prototype on RFID and Sensor Networks for Elderly Health Care," in *RFID Handbook: Applications, Technology, Security, and Privacy*, 1st ed., Syed Ahson and Mohammad Ilyas, Eds. Boca Raton, FL, 2008, pp. 311-328.

[3] National Council on Patient Information and Education, "Enhancing Prescription Medicine Adherence: A National Action Plan," 2007.

[4] Eric Ishman, "Inventing Wellness Systems for Aging in Place," *IEEE Computer*, pp. 34-41, May 2004.

[5] Upkar Varshney, "Pervasive Healthcare and Wireless Health Monitoring," *Mobile Networks and Applications*, vol. 12, pp. 113-127, 2007.

[6] Alison M. Kenner, "Securing the Elderly Body: Dementia, Surveillance, and the Politics of "Aging in Place"," *Surveillance & Society*, vol. 5, no. 3, pp. 252-269, 2008.

[7] Juan M. Corchado, Javier Bajo, Yanira De Paz, and Dante I. Tapia, "Intelligent Environment for Monitoring Alzheimer Patients Agent Technology for Health Care," *Decision Support Systems*, vol. 44, no. 2, pp. 382-396, 2008.

[8] Corey McCall. (2010, July) Personal Medication Monitor v0.9 Source Code. [Online]. http://cns.eecs.ucf.edu/PMM/download/McCall_PMM_v09.pde

[9] Olga Boric-Lubecke and Victor M. Lubecke, "Wireless House Calls: Using Communications Technology for Health Care and Monitoring," *IEEE Microwave* , pp. 43-48, September 2002.

[10] Pei R. Sun, Bo H. Wang, and Fan Wu, "A New Method to Guard Inpatient Medication Safety by the Implementation of RFID," *Journal of Medical Systems*, vol. 32, pp. 327-332, 2008.

[11] Fan Wu, Frank Kuo, and Lie-Wei Liu, "The Application of RFID on Drug Safety of Inpatient Nursing Healthcare," in *ACM International Conference on Electronic Commerce*, Xi'an, China, 2005, pp. 85-92.

[12] Huzaifa Al Nahas and Jitender S. Deogun, "Radio Frequency Identification Applications in Smart Hospitals," in *Twentieth IEEE International Symposium on Computer-Based Medical Systems*, Maribor, Slovenia, 2007, pp. 337-342.

[13] INRange Systems, Inc. (2010) Understanding EMMA. [Online]. http://www.inrangesystems.com/index.php?page=understanding-emma

[14] Arduino. (2009, March) Arduino Mega. [Online]. http://arduino.cc/en/Main/ArduinoBoardMega

[15] Arduino. (2009, March) Arduino Mega Schematic & Reference Design. [Online]. http://arduino.cc/en/uploads/Main/arduino-mega-reference-design.zip

[16] Brian Schmalz. (2009, August) EasyDriver v4.2 Eagle Files and BOM. [Online]. http://www.schmalzhaus.com/EasyDriver/EasyDriver_v42.zip

[17] NKC Electronics. (2010) NKC Ethernet Shield for Arduino MEGA Schematics. [Online]. http://www.vintagecomputercables.com/eagle/ethernet_shield.sch

[18] Limor Fried. (2010, October) Breadboard Supply Schematic. [Online]. http://www.ladyada.net/make/bbpsup/supply.sch

[19] Digi-Key Corporation. (2010) Digi-Key part #620-1140-2-ND. [Online]. http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=620-1140-2-ND

[20] SkyeTek, "SkyeTek Protocol: Host Communication Description for the SkyeRead family of RFID readers," 2003.

[21] Acculab, YADAP-USB Operating Instructions, 2005.