RFID-Assisted wireless sensor networks for cardiac tele-healthcare

Laura J. Celentano
RFID-Assisted Wireless Sensor Networks for Cardiac 

Tele-healthcare

by

Laura J. Celentano

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Fei Hu
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
November 2007

Approved By:

_____________________________________________________________
Dr. Fei Hu
Primary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________________________
Dr. Marcin Lukowiak
Secondary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________________________
Dr. Muhammad Shaaban
Secondary Advisor – R.I.T. Dept. of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: RFID-Assisted Wireless Sensor Networks for Cardiac Tele-healthcare

I, LAURA J. CELENTANO, HEREBY GRANT PERMISSION TO THE WALLACE MEMORIAL LIBRARY TO REPRODUCE MY THESIS IN WHOLE OR PART.

_________________________________
Laura J. Celentano

_________________________________
Date
Dedication

For my family, who have made me into all that I am, and for the friends that have supported me while finding my way.
Acknowledgements

I would like to thank Dr. Fei Hu for his support and guidance, both academically and professionally. His advising on this work has opened up new opportunities for me in the fields of radio frequency identification and wireless sensor networks. I would also like to thank my other two esteemed committee members, Dr. Marcin Lukowiak and Dr. Muhammad Shaaban.

I would like to especially thank Glenn Ramsey for his proof reading of this thesis.
Abstract

As the baby boomers head into old age, America will see a dramatic increase in the number of elderly patients admitted to healthcare facilities, such as nursing homes. Due to this rising elderly population, it will be difficult for nursing home personnel to monitor all patients at once. One way to cut down on the amount of supervision by the staff is for patients to administer their own medication. This leads to new problems though, as a patient incorrectly administering one of their many medications could lead to a disastrous end. Technology to wirelessly transmit a patient’s electrocardiogram (ECG) has also been implemented to reduce supervision. Wireless transmissions are infamous for their error rate, but the ECG is a sensitive signal where every second of data matters and cannot tolerate such losses. Additionally, such existing networks employ an expensive communication infrastructure. Due to this healthcare crisis, the ability for a device to remotely monitor a patient’s medication intake and transmit accurate ECG readings, while being cost efficient, is a major innovation.

To combat this crisis, this thesis focuses on a multi-hop wireless sensor network (WSN) composed of many wearable sensors, one for each patient, that host a radio frequency identification (RFID) reader and are capable of RF communication. Each wearable device is also assumed to contain an ECG sensor, though this was not implemented in this work. The system is responsible for two distinct features. The first is remotely supervised patient medication intake via RFID and a central workstation/database. The second is the accurate remote transmission of a patient’s ECG using the extended Kalman filter (EKF) for wireless error recovery.
# Table of Contents

Thesis Release Permission Form ....................................................................................................... ii  
Dedication ........................................................................................................................................ iii  
Acknowledgements ........................................................................................................................ iv  
Abstract ........................................................................................................................................... v  
List of Figures ..................................................................................................................................... viii  
List of Tables ................................................................................................................................... x  
List of Equations ........................................................................................................................... xi  
Glossary ........................................................................................................................................ xiv  

## Chapter 1
Introduction ................................................................................................................................. 1

## Chapter 2
Background ..................................................................................................................................... 4  
2.1. RFID ....................................................................................................................................... 4  
2.2. The Discrete Kalman Filter ..................................................................................................... 8  
2.3. Wireless Sensor Networks ...................................................................................................... 16  
2.4. Work Environment ................................................................................................................ 17  
2.4.1 TinyOS ............................................................................................................................... 17  
2.4.2 nesC ................................................................................................................................... 17  
2.4.3 MySQL ............................................................................................................................. 18  
2.4.4 MATLAB .......................................................................................................................... 18

## Chapter 3
RFID via Mica2Dot and Medical Software System ..................................................................... 19  
3.1. RFID via Mica2Dot ................................................................................................................ 19  
3.2. Medical Software System .................................................................................................... 23  
3.2.1 RIT Medi-Write ................................................................................................................. 24  
3.2.2 MySQL Database .............................................................................................................. 26
Chapter 4  RFID via Mica2 with Multi-hop ................................................................. 34
  4.1. Mica2 Advantages ......................................................................................... 34
  4.2. Mote-Reader Interface .................................................................................... 38
  4.3. Multi-hop Communication ............................................................................. 41
  4.3.1 TinyOS Active Messages ............................................................................. 42
  4.3.2 MultiHopRouter Multi-hop Protocol ......................................................... 45
  4.4. Multi-hop Medi-View System ........................................................................ 48

Chapter 5  The Extended Kalman Filter ................................................................. 51
  5.1. Discrete Kalman Filter Alterations ............................................................... 51
  5.2. ECG Modeling ................................................................................................ 57
  5.3. Implementation ............................................................................................... 60

Chapter 6  Conclusion ............................................................................................. 64

Bibliography ........................................................................................................... 67
List of Figures

Figure 2.1: Taipei Medical University Hospital LBMS Infrastructure ................................. 5
Figure 2.2: Network Infrastructure of eShepered ............................................................... 6
Figure 2.3: ScripTalk Reader and Smart Label ................................................................. 7
Figure 2.4: Packet ECG data loss/errors due to wireless link.......................................... 9
Figure 2.5: The discrete Kalman filter cycle ................................................................. 10
Figure 2.6: A complete view of the Kalman filter’s recursive nature ............................ 15
Figure 2.7: Generic mote layout ....................................................................................... 16
Figure 2.8: Mica2 and Mica2Dot motes ........................................................................... 17
Figure 3.1: SkyeTek M1-Mini RFID reader ................................................................. 19
Figure 3.2: SkyeTek M1-Mini RFID reader pin-out ....................................................... 21
Figure 3.3: Crossbow Mica2Dot pin-out .......................................................................... 21
Figure 3.4: Crossbow MIB510 gateway board ................................................................. 23
Figure 3.5: SkyeTek M1 reader ....................................................................................... 24
Figure 3.6: Medi-Write screenshot .................................................................................. 25
Figure 3.7: Entire Medi-Write system ............................................................................. 26
Figure 3.8: Database screenshot ...................................................................................... 27
Figure 3.9: Screenshot of status messages ..................................................................... 28
Figure 3.10: Medi-View state diagram .......................................................................... 30
Figure 3.11: The given medication is not in the database .............................................. 32
Figure 3.12: Patient attempting to take medication that is not theirs ............................ 32
Figure 3.13: Patient is out of pills .................................................................................. 32
Figure 3.14: The patient is about to overdose ............................................................... 32
Figure 3.15: Medi-View system overview diagram ......................................................... 33
Figure 3.16: Entire Medi-View system ............................................................................. 33
Figure 4.1: MDA100CB DAC board ................................................................................ 39
Figure 4.2: Connected MDA100CB and M1-Mini ........................................................... 40
Figure 4.3: Multi-hop communication illustration ......................................................... 41
Figure 4.4: SendMsg and ReceiveMg interfaces .............................................................. 42
Figure 4.5: Single-hop packet format ............................................................................... 43
Figure 4.6: Send and Intercept interfaces ......................................................................... 44
Figure 4.7: Multi-hop packet format ............................................................................... 44
Figure 4.8: MultiHopRouter configuration ..................................................................... 47
Figure 4.9: Experiment field and mote placement ............................................................ 49
Figure 4.10: Entire multi-hop Medi-View system ............................................................ 50
Figure 5.1: A complete view of the extended Kalman filter’s recursive nature .............. 56
Figure 5.2: Example of a normal ECG trace .................................................................. 57
Figure 5.3: Normal ECG signal ....................................................................................... 60
Figure 5.4: ECG from 5.3 with simulated wireless errors ................................................ 61
Figure 5.5: Filtered ECG from 5.3 ................................................................................... 62
Figure 5.6: Filtered abnormal ECG with arrhythmia ....................................................... 63
Figure 5.7: Filtered abnormal ECG with arrhythmia ....................................................... 63
List of Tables

Table 2.1: Discrete Kalman filter time update equations ................................................. 14
Table 2.2: Discrete Kalman filter measurement update equations ................................. 14
Table 3.1: Effective range for ISO 15693 tags using internal antenna ............................. 20
Table 3.2: Possible SELECT_TAG command responses .................................................. 22
Table 3.3: Correlation between Medi7Write field names and database field names ........ 27
Table 4.1: Radio physical properties comparison ............................................................ 35
Table 4.2: System core comparison .................................................................................. 35
Table 4.3: Power supply comparison .............................................................................. 37
Table 4.4: MDA100CB prototyping area layout .............................................................. 40
Table 5.1: Description of the variables in 5.5 through 5.10 ............................................. 53
Table 5.2: EKF time update equations ............................................................................. 56
Table 5.3: EKF measurement update equations ............................................................... 56
Table 5.4: Typical parameters of the synthetic ECG model ............................................. 59
Table 5.5: rddata parameters and execution command ..................................................... 61
List of Equations

\[ P_{\text{received}} \propto \frac{1}{d^2} \] \hspace{0.5cm} (2.1) ................................................................. 8

\[ x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \] \hspace{0.5cm} (2.2) ................................................................. 10

\[ z_k = Hx_k + v_k \] \hspace{0.5cm} (2.3) ................................................................. 11

\[ \hat{x}_k^\cdot = Ax_{k-1} + Bu_{k-1} \] \hspace{0.5cm} (2.4) ................................................................. 12

\[ \hat{x}_k = \hat{x}_k^\cdot + K(z_k - H\hat{x}_k^\cdot) \] \hspace{0.5cm} (2.5) ................................................................. 12

\[ e_k^\cdot = x_k - \hat{x}_k^\cdot \] \hspace{0.5cm} (2.6) ................................................................. 12

\[ e_k = x_k - \hat{x}_k \] \hspace{0.5cm} (2.7) ................................................................. 12

\[ P_k^\cdot = E[e_k^\cdot e_k^\cdot^T] \] \hspace{0.5cm} (2.8) ................................................................. 13

\[ P_k^- = AP_{k-1}A^T + Q \] \hspace{0.5cm} (2.9) ................................................................. 13

\[ P_k = E[e_k e_k^T] \] \hspace{0.5cm} (2.10) ................................................................. 13

\[ P_k = (I - K_k H) \cdot P_{k-1} \cdot (I - K_k H)^T + K_k R \cdot K_k^T \] \hspace{0.5cm} (2.11) ................................................................. 13

\[ x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \] \hspace{0.5cm} (5.1) ................................................................. 51

\[ z_k = h(x_k, v_k) \] \hspace{0.5cm} (5.2) ................................................................. 52

\[ \hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \] \hspace{0.5cm} (5.3) ................................................................. 52

\[ \hat{z}_k = h(\hat{x}_k^-, 0) \] \hspace{0.5cm} (5.4) ................................................................. 52

\[ x_k \approx \hat{x}_k^- + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \] \hspace{0.5cm} (5.5) ................................................................. 52

\[ z_k \approx \hat{z}_k + H(x_k - \hat{x}_k^-) + Vv_k \] \hspace{0.5cm} (5.6) ................................................................. 52
\[ A_{[i,j]} = \frac{\partial f_{[i]}^{[j]}}{\partial x_{[i]}} (\hat{x}_{k-1}, u_{k-1}, 0) \] (5.7) .......................................................... 52

\[ W_{[i,j]} = \frac{\partial f_{[i]}^{[j]}}{\partial w_{[i]}} (\hat{x}_{k-1}, u_{k-1}, 0) \] (5.8) .......................................................... 52

\[ H_{[i,j]} = \frac{\partial h_{[i]}^{[j]}}{\partial x_{[i]}} (\hat{x}_{k}, 0) \] (5.9) .......................................................... 53

\[ V_{[i,j]} = \frac{\partial h_{[i]}^{[j]}}{\partial v_{[i]}} (\hat{x}_{k}, 0) \] (5.10) .......................................................... 53

\[ \tilde{e}_{x_k} \equiv x_k - \hat{x}_k \] (5.11) .......................................................... 54

\[ \tilde{e}_{z_k} \equiv z_k - \hat{z}_k \] (5.12) .......................................................... 54

\[ \tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k \] (5.13) .......................................................... 54

\[ \tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k \] (5.14) .......................................................... 54

\[ \hat{e}_k = K_k \tilde{e}_{z_k} \] (5.15) .......................................................... 54

\[ \hat{x}_k = \hat{x}_k^- + \hat{e}_k \] (5.16) .......................................................... 55

\[ \hat{x}_k = \hat{x}_k^- + K_k \tilde{e}_{z_k} \] (5.17) .......................................................... 55

\[ \hat{x}_k = \hat{x}_k^- + K_k (z_k - \hat{z}_k) \] (5.18) .......................................................... 55

\[ x' = a\alpha - a\eta \] (5.19) .......................................................... 58

\[ y' = a\eta + a\alpha \] (5.20) .......................................................... 58

\[ z' = -\sum_{i\in\{P,Q,R,S,T\}} a_i \Delta \theta_i \exp\left(-\frac{\Delta \theta_i^2}{2b_i^2}\right) - (z - z_0) \] (5.21) .......................................................... 59

\[ \alpha = 1 - \sqrt{x'^2 + y'^2} \] (5.22) .......................................................... 59

\[ \Delta \theta_i = (\theta - \theta_i) \text{mod}(2\pi) \] (5.23) .......................................................... 59
\[ \theta = \arctan 2(y, x) \] (5.24) ................................................................. 59

\[ -\pi \leq \arctan(y, x) \leq \pi \] (5.25) ................................................................. 59
# Glossary

<table>
<thead>
<tr>
<th>C</th>
<th>Cardiac arrhythmia</th>
<th>Any of a group of conditions in which the electrical activity of the heart is irregular or is faster or slower than normal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>DoD</td>
<td>Department of Defense – United States federal department charged with coordinating and supervising all agencies and functions of the government relating directly to national security and the military.</td>
</tr>
<tr>
<td>E</td>
<td>ECG</td>
<td>Electrocardiogram – an electrical recording of the heart, primarily used in the investigation of heart disease.</td>
</tr>
<tr>
<td></td>
<td>E-ZPass</td>
<td>An electronic toll collection system.</td>
</tr>
<tr>
<td>G</td>
<td>Galvanometer</td>
<td>An instrument for detecting and measuring electrical currents.</td>
</tr>
<tr>
<td>N</td>
<td>nesC</td>
<td>Network embedded systems C – a programming language designed specifically to build applications for the TinyOS platform. It is an</td>
</tr>
</tbody>
</table>
extension of the C programming language where components are “wired” together to create a program.

**R**

**RFID**  Radio Frequency Identification – an automatic recognition technology utilizing radio frequency communication.

**S**

**SQL**  Structured Query Language – a computer language designed for the retrieval and management of data in relational database management systems, database schema creation and modification, and database object access control management.

**T**

**TinyOS**  An open source operating system which targets wireless sensor networks. It has a component-based architecture and is able to operate within the severe memory constraints imposed by a sensor network.

**W**

**WLAN**  Wireless Local Area Network – a type of local-area network that uses high-frequency radio waves rather than wires to communicate between nodes.
Chapter 1 Introduction

According to the U.S. Bureau of the Census [1], the number of adults age 65 to 84 is expected to double from 35 million to nearly 70 million by 2025 when the youngest Baby Boomers retire. A recent study found that almost one third of U.S. adults, most of whom held full-time jobs, were serving as informal caregivers – mostly to an elderly parent [2]. As this burden becomes too great and more elderly patients head to nursing homes, it will become very important to build a remote telehealth monitoring system that can continuously, automatically, accurately, and cost effectively monitor such things as a patient’s medication intake and heartbeat.

Telehealth monitoring can be defined as “mobile computing, medical sensor, and communications technologies for health-care.” This represents the evolution of e-health systems from traditional desktop “telemedicine” platforms to wireless/mobile configurations [3]. While patient monitoring has conventionally been assigned to trained medical care personnel, putting some responsibility in the hands of the patient could alleviate a considerable amount of the staff’s workload. This would free medical professionals from a tedious task and allow them to center their attention on more demanding medical emergencies.

Industry has taken notice of the need for remote medical monitoring and several companies have come out with products to remotely and wirelessly monitor a patient’s ECG. Unfortunately, wireless transmissions have an error rate many times that of a traditional wired network. While there’s a desire to decrease the one-on-one time staff has with each patient, it must done safely. A loss in medical data typical of that seen in wireless transmissions cannot be tolerated – each piece of cardiac data could carry
important medical information. For example, a sudden heart failure may produce an abnormal ECG signal that lasts for only a few seconds. The transmission error of such a segment of ECG data is disastrous to the capture of, and response to, sudden heart failure events.

Based on these motivations, remote medication and accurate ECG monitoring, there have been a number of attempts to develop medical systems similar to the proposed work in this thesis, but they have fallen short. While these systems monitor the dispensing of medication to patients using RFID, they still rely on the labor of medical staff. In addition, such systems employ an expensive communication infrastructure. For nursing homes that are regional areas, expensive wireless local area networks (WLAN) should be avoided. Instead, a low-cost, short-distance telehealth system is preferred. Other devices on the open market that use RFID to manage medication are not part of any network and do not allow monitoring by healthcare personnel. Finally, in the systems that wirelessly monitor a patient’s ECG, error recovery is not performed even though transmission errors are likely and cannot be tolerated.

This thesis offers several contributions. First is a wearable medication monitoring platform which hosts an RFID reader and is capable of RF communication (Chapter 3). The patient will be able to scan any of their medication bottles containing an RFID tag and wirelessly transmit the attempted drug application to a central workstation. The workstation queries a central database which contains the proper administration for the given medication. If for any reason the patient is improperly taking this drug, they will be alerted by a red LED toggling on their wearable device. Additionally, the healthcare employee manning the workstation will also receive an alert message indicating a patient
has attempted to improperly take a drug. It is then at their discretion whether to call the
patient to follow-up or to physically check on them.

To avoid a high cost network while still attaining long range distance from the
central workstation, the network employs multi-hop communication, using each patient’s
wearable sensor as a hop (Chapter 4). By using patients as hops, rather than statically
placed sensors, the same effect can be achieved while making more efficient use of
resources. Finally, an EKF is applied in MATLAB to real ECG signals with simulated
data losses to achieve more reliable ECG transmissions (Chapter 5).

The rest of this document is organized in the following way. Chapter 2 provides
some background information on the topics of RFID, the discrete Kalman filter, as well
as other works supporting this thesis. As mentioned above, Chapters 3, 4, and 5 offer
more insights regarding the implementation of the wearable platform using Crossbow’s
Mica2Dot mote and the medication supervision software, implementing the system on
Crossbow’s superior Mica2 mote with multi-hop communication, and using the extended
Kalman filter to recover wireless ECG transmission losses, respectively. Chapter 6
provides some final remarks on this work and its future outlook. Recommendations are
also made for any subsequent work.
Chapter 2  Background

This chapter provides some foundational information on the subjects of RFID and its use in tele-healthcare, the Kalman filter, WNSs, and the supporting work environment. While it is not meant to provide a thorough discussion on any of the aforementioned topics, it will serve as a basis of understanding for the underlying work and research that went into preparing this thesis.

2.1. RFID

RFID is a method of storing and retrieving data, similar to a barcode. With RFID, the electromagnetic or electrostatic coupling in the RF portion of the electromagnetic spectrum is used to transmit signals. An RFID system consists of a reader and any number of tags. The reader contains an antenna and a transceiver, which both work to read the radio frequency and transfer the information to a processing device. The tag, or transponder, is an integrated circuit containing RF circuitry and information to be transmitted. There are generally two types of tags, passive and active.

Passive tags do not have their own internal power supply. Instead the minute electrical current induced in the antenna by the incoming radio frequency signal from the reader provides just enough power to supply the tag and provide a response. While this tag has a longer lifetime and is inexpensive, its read range is shorter than a tag with its own internal power supply [4]. The typical read range of a traditional passive tag ranges from a few inches to 30 feet [4]. Active tags are able to achieve a longer read range since they do possess an internal power supply and do not solely depend upon the RF field of
the reader. Unfortunately this also means they have a limited lifetime due to their battery and are quite expensive [4]. It is for this reason passive tags were chosen for this application – the read range would be small since a patient is scanning the medication bottle over their wearable sensor directly. To further cut down on costs, a primary concern of the healthcare industry, the tags have a longer lifetime, which reduces the need for replacement.

RFID has been used to replace the barcode in supply chain/object mobility monitoring applications in many organizations such as Wal-Mart, E-ZPass, and the U.S. Department of Defense (DoD) [5]. Tele-healthcare corporations have seen the success and usefulness of RFID and are now beginning to incorporate it into healthcare scenarios to alleviate errors and to cut down on costs.

A Location-Based Medicare Service (LBMS) was implemented in Taipei Medical University Hospital which used RFID tags to locate both patients and hospital assets with successful results [5]. The infrastructure of this system can be seen in Figure 2.1.

![Figure 2.1: Taipei Medical University Hospital LBMS Infrastructure](image)
Exavera’s eSheperd uses RFID over WLAN to track patients, staff, and supplies, including medication dispensed to patients by the staff [6]. The infrastructure of this system can be seen in Figure 2.2.

![Figure 2.2: Network Infrastructure of eShepered](image)

En-Vision America has created a new way to provide prescription information to the user using RFID with their product, ScripTalk, seen in Figure 2.3 [7]. When a patient using a ScripTalk reader submits a prescription, the pharmacy software prints and programs an auxiliary smart label using a dedicated, small-footprint printer. The smart label, which stores prescription information, is placed onto the prescription container by the pharmacist. In the home, the patient uses a hand-held ScripTalk reader that speaks out the label information using speech synthesis technology.
Unfortunately, the first two of these systems do not put the actual medication intake into the hands of the patient; they all rely on staff dispensing. Additionally they require WLAN access points and an expensive communication infrastructure. While the last system mentioned does put medication administration into the hands of the patient, it is not part of a network and therefore cannot be supervised by staff.

The RFID based medication monitoring system proposed in this work does not require an expensive communication infrastructure. Instead it will use an ad-hoc network consisting of wireless sensors which will host RFID readers. These sensors will be worn by all patients and will be used to scan their medication bottles which will contain RFID tags. The tag read will be transmitted to a central workstation which contains a central database. By referencing the database it will be determined if the medication should be taken or not. If the medication should not be taken, for any reason, the patient will be alerted with a red LED toggling on their sensor. Additionally, an alert will appear on the staff supervised central workstation citing the patient, the time of the incorrect application, and what the medication was. The sensors will utilize multi-hop
communication so that they can communicate with the workstation even when they are out of transmission range. Through this thesis contribution staff members no longer need to make rounds to supervise patients taking their medication, it can be done remotely.

### 2.2. The Discrete Kalman Filter

For the purposes of this work, the wearable devices are assumed to be equipped with ECG sensors. It is also assumed that ECG interpretation software will be running on the central workstation where all data are being sent. The wireless error rate is around 2% ~ 10% [8]. This is much higher than traditional wired networks, such as cable-based medical networks, which only have $10^{-9}$ error rate. As mentioned earlier, data errors cannot be tolerated because each piece of cardiac data could carry important medical information. There are several factors which can cause ECG data errors or even data loss. They are wireless signal energy loss during propagation and radio reflection/diffraction/scattering.

Wireless signal energy loss during propagation can be attributed to the well-known fact that the received wireless signal strength decreases when distance from the sender increases. This can be seen in Equation 2.1. This attenuation limits the range of the radio signal.

$$P_{\text{received}} \propto \frac{1}{d^2} \quad (2.1)$$
A radio wave may change its direction when hitting objects. Radio reflection/scattering damages wireless signals because an erroneous packet will be discarded by any receiver, which causes packet loss. Figure 2.4 shows one section of a collected ECG data series. It clearly shows the data missing at 0.05 s – 0.07 s and possible data errors at 0.09 s – 0.11 s. Even though ECG values are typically different at 0.01 s of time scale, three nearly duplicate values at 0.09 s – 0.11 s are still seen, which indicates data errors.

![Figure 2.4: Packet ECG data loss/errors due to wireless link](image)

Based on this knowledge it is clear an error recovery scheme must be implemented in order to obtain a reliable remote ECG. For this work the extended Kalman filter was chosen to recover wireless data loss. Before delving into the EKF, it would be sensible to describe its more simplified version, the discrete Kalman filter.

The discrete Kalman filter is named after Rudolph E. Kalman, who in 1960 published his famous paper describing a recursive filter that estimates the state of a dynamic linear system from a series of incomplete and noisy measurements. The discrete Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error.
covariance \[9\]. The discrete Kalman filter estimates a process by using a form of feedback control. The filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the discrete Kalman filter fall into two groups, time update equations and measurement update equations. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations \[9\]. A visualization of this can be seen in Figure 2.5 \[9\].

\[ x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \]  \hspace{1cm} (2.2)

For the discrete Kalman filter, the system is assumed to be linear. If it is non-linear it should be linearized, as will be seen later in Chapter 5. The linear system has a state vector called \(x\). For example, if attempting to determine the orbit of a satellite, the state vector's components would include the satellite's position and velocity. The value of \(x\) would like to be known at various times. First, assume at time \(k\) the state vector, \(x_k\), satisfies the dynamic equation seen in Equation 2.2 \[10\].
From Equation 2.2, \( A \) is an \( n \times n \) matrix which relates the state at the previous time, \( k - 1 \), to the state at the current time, \( k \). Here \( A \) was assumed to be constant, but this may not always be the case. The \( n \times l \) matrix \( B \) relates an optional control input, \( u \), to the state [9]. The variable \( \omega \) is process noise, which is assumed to be zero mean white noise. This means \( \omega \) is not correlated with any other random variables and is especially not correlated with past values of \( \omega \). The covariance matrix of \( \omega \) is \( Q \). The covariance is the measure of how much two variables vary together – the more often they differ in the same direction, the more positive the covariance, and the more often they differ in opposite directions, the more negative the covariance [10].

It is also assumed that measurements are taken at time \( k \), represented by \( z_k \), where the measurements are a linear function of \( x \). The equation for \( z_k \) can be seen below in Equation 2.3 where \( H \) is an \( m \times n \) matrix that relates the state to the measurement, \( z_k \). Again, \( H \) was assumed constant, but this may not always be the case. The variable \( v_k \) is the measurement noise and is assumed to be white noise with zero mean, and its covariance matrix is \( R \). The discrete Kalman filter is an algorithm used to estimate \( x_k \) from the measurements taken [10].

\[
z_k = Hx_k + v_k
\]  
\[
(2.3)
\]

The goal is to compute an estimate of the state vector, \( x_k \), where the estimate is called \( \hat{x}_k \). The a priori state estimate will be called \( \hat{x}_{k^-} \), which is taken at time \( k \), and uses knowledge of the process prior to time \( k \) [10]. This can be seen in Equation 2.4 below.

11
\[
\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}
\]  

(2.4)

Additionally, the *a posteriori* state estimate will be called \(\hat{x}_k\), which is taken at time \(k\) given measurement \(z_k\). The goal is to find an equation that computes an *a posteriori* state estimate, \(\hat{x}_k\), as a linear combination of an *a priori* estimate, \(\hat{x}_k^-\), and a weighted difference between an actual measurement, \(z_k\) and a measurement prediction, \(H\hat{x}_k^-\). This can be seen in Equation 2.5. The difference, \(z_k - H\hat{x}_k^-\), in Equation 2.5 is called the measurement innovation, or the residual. The residual reflects the discrepancy between the predicted measurement and the actual measurement. Clearly then, a residual of zero means that the two are in complete agreement [10].

\[
\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)
\]  

(2.5)

The *a priori* and *a posteriori* estimate errors, seen in Equations 2.6 and 2.7, respectively, lead to the centerpiece of the discrete Kalman filter – the computation of the estimate error covariance matrices for the *a priori* and *a posteriori* estimates [10]. These are represented by \(P_k^-\), seen in Equations 2.8 and 2.9, and \(P_k\), seen in Equations 2.10 and 2.11, respectively.

\[
e_k^- \equiv x_k^- - \hat{x}_k^-
\]  

(2.6)

\[
e_k \equiv x_k - \hat{x}_k
\]  

(2.7)
\[ P_k^- = E[e_k^- e_k^-^T] \quad (2.8) \]

\[ P_k^- = A P_{k-1} A^T + Q \quad (2.9) \]

\[ P_k = E[e_k e_k^T] \quad (2.10) \]

\[ P_k = (I - K_k H) \cdot P_k^- \cdot (I - K_k H)^T + K_k R \cdot K_k^T \quad (2.11) \]

The \( K \) seen in Equations 2.5 and 2.11 represents the Kalman gain. It is an \( m \times n \) matrix that is chosen such that it minimizes the \textit{a posteriori} error covariance seen in Equation 2.9. This minimization can be accomplished by first substituting Equation 2.5 into the definition for Equation 2.7. This result is then substituted into Equation 2.10 and the indicated expectations are performed. After this, the derivative of the trace of the result with respect to \( K \) is set equal to zero, and \( K \) is solved for [9]. The resulting minimization can be seen in Equation 2.12, as well as rearranged in Equation 2.13.

\[ K_k = P_k^- H^T \cdot (H \cdot P_k^- H^T + R)^{-1} \quad (2.12) \]

\[ K_k = \frac{P_k^- H^T}{H P_k^- H^T + R} \quad (2.13) \]

Referring to Equation 2.13, as the measurement error covariance, \( R \), approaches zero, the actual measurement, \( z_k \), is “trusted” more, while the predicted measurement,
$H\hat{x}_k^-$, is trusted less. On the other hand, as the \textit{a priori} estimate error covariance, $P_k^-$, approaches zero, the actual measurement is trusted less, while the predicted measurement is trusted more [9].

Returning back to the feedback loop analogy made earlier, Tables 2.1 and 2.2 highlight and summarize the time update equations and the measurement update equations, respectively.

Table 2.1: Discrete Kalman filter time update equations

<table>
<thead>
<tr>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}<em>k^- = A\hat{x}</em>{k-1} + Bu_{k-1}$</td>
</tr>
<tr>
<td>$P_k^- = AP_{k-1}A^T + Q$</td>
</tr>
</tbody>
</table>

Table 2.2: Discrete Kalman filter measurement update equations

<table>
<thead>
<tr>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$</td>
</tr>
<tr>
<td>$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$</td>
</tr>
<tr>
<td>$P_k = (I - K_k H) \cdot P_k^- (I - K_k H)^T + K_k R \cdot K_k^T$</td>
</tr>
</tbody>
</table>

Referring to Table 2.2, the first task during the measurement update is to compute the Kalman gain, $K$. The next step is to actually take a measurement to obtain $z_k$, and then to generate an \textit{a posteriori} state estimate by incorporating the measurement. The final step is to obtain an \textit{a posteriori} error covariance estimate. After each time and measurement update pair, the process is repeated with the previous \textit{a posteriori} estimate used to project or predict the new \textit{a priori} estimates. This recursive process is illustrated in Figure 2.6 [9]. The equation for $P_k$ in Figure 2.6 differs from Equation 2.11.
Equation 2.11 is numerically stable and yields correct answers even when $K$ is inaccurate. On the other hand, the equation for $P_k$ in Figure 2.6 is a simplification that is not numerically stable. While it is an accurate simplification based on the equation for $K$ in Equation 2.12, even the smallest error in computing $K$ can lead to horrific errors if the Figure 2.6 equation is used to compute $P_k$. This was a real problem in the 1960's and caused many problems in Kalman filter design.

![Figure 2.6: A complete view of the Kalman filter’s recursive nature](image)

While an ECG is a non-linear signal and therefore cannot use the standard Kalman filter, it can be linearized and have the extended Kalman filter applied to it. This will enable the lost pieces of wireless data to be predicted and the entire ECG signal to be reconstructed. The differences in the discrete Kalman filter and the EKF, as well as linearizing an ECG signal to be used with the EKF, will be covered in Chapter 5.
2.3. Wireless Sensor Networks

Sensor networks are highly distributed networks of small, lightweight wireless nodes. Each node consists of three subsystems: the sensor subsystem which senses the environment, the processing subsystem which performs local computations, and the communication subsystem. This layout can be seen in Figure 2.7. Research on WSNs was originally spurred by military applications such as battlefield surveillance and guidance systems of intelligent missiles. Sensors can be found in office buildings, warehouses, and hospitals, similar to the application in this work. The applications of sensor networks are endless, limited only by the current technology and human imagination [11].

![Figure 2.7: Generic mote layout](image)

Figure 2.8 shows the two different types of sensor nodes, most often referred to as motes, in this research work. Currently, their price and size are not conducive for their target applications. The goal is eventually to be able to produce cheap “smart dust” sensors that can be deployed in large quantities.
2.4. Work Environment

2.4.1 TinyOS

TinyOS is an open source operating system which targets WSNs. It was primarily developed by the University of California, Berkeley in cooperation with Intel Research and is written in nesC. It has a component-based architecture and is able to operate within the severe memory constraints imposed by a sensor network. The version of TinyOS used in this thesis was 1.1.0-1.

2.4.2 nesC

Network embedded systems C (nesC) is a programming language designed specifically to build applications for the TinyOS platform. It is an extension of the C programming language where components are “wired” together to create a program. It was also developed by the University of California, Berkeley and Intel Research. The key feature of the language is the small sized code it produces. This is why it is able to run on a resource constrained system, like a WSN.
2.4.3 MySQL

Created by MySQL AB, MySQL is arguably the most popular open source structured query language (SQL). It is a relational database management system, meaning data is stored in the form of tables and the relationship among the data is also stored in the form of tables. It is available for most all platforms and libraries for accessing MySQL databases are available in all major programming languages with language-specific APIs [12]. The version of MySQL used in this thesis was MySQL 5.0 Server for Windows. All associated patient medication data was stored in the database.

2.4.4 MATLAB

Matrix Laboratory (MATLAB) was created by The MathWorks, Inc. and allows for easy matrix manipulation and algorithm implementation. It allows for simple user interface creation, making it popular among developers. The MATLAB language is proprietary and referred to as M-code or just M. It is intended to “perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran” [13]. The copy of MATLAB used in this thesis was MATLAB 7.0.1 and it was used to simulate the EKF which filtered an ECG signal with simulated data losses.
Chapter 3   RFID via Mica2Dot and Medical Software System

This chapter is split into two sections. The first section focuses on the initial integration of RFID into a WSN while the second section discusses the creation of the medical software that was to go along with the system.

3.1. RFID via Mica2Dot

The RFID reader chosen for this work was SkyeTek’s M1-Mini. It is advertised as “the world’s smallest, self-contained multi-protocol 13.56 MHz” low power RFID reader [14] and sports a radius of a mere 12.7 mm. The M1-Mini can be seen in Figure 3.1.

![Figure 3.1: SkyeTek M1-Mini RFID reader](image)

The reader requires an input voltage between 3.2 V – 6 V. Its current consumption is 60 mA when scanning a tag and 15 mA when idle. ReaderWare, an
open-architecture software suite residing on all SkyeTek’s modules, provides intelligence for the RFID reader. It has an internal antenna which provides it with a read range of approximately 50.8 mm, but this also depends on the tag type. It has the ability to attach a standard 50 Ohm external antenna for improved read-range as well. It is capable of reading and writing to tags based on ISO 15693, 14443A, and 18000-3 air interface protocols. The effective read range for varying ISO 15693 tag types using the internal antenna, as done in this work, can be seen in Table 3.1.

<table>
<thead>
<tr>
<th>ISO 15693 Tag Dimensions</th>
<th>Effective Range for Internal Antenna</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 mm x 76 mm</td>
<td>5.0 cm</td>
</tr>
<tr>
<td>38 mm x 22.5 mm</td>
<td>3.5 cm</td>
</tr>
</tbody>
</table>

Table 3.1: Effective range for ISO 15693 tags using internal antenna

The WSN itself includes a number of node motes, Crossbow’s Mica2Dot, which will be equipped with these RFID readers, and one base station mote. The Mica2Dot acts as the host system by sending the M1-Mini commands, as well as receiving and interpreting its response to the given command. The Mica2Dot has a 12.5 mm radius and a 433 MHz multi-channel radio transceiver for wireless communications. It requires an input voltage between 2.7 V – 3.3 V. The overlapping input voltages enable both reader and mote to run off the same power supply. The pin-out for the M1-Mini can be seen in Figure 3.2 and the pin-out for the Mica2Dot can be seen in Figure 3.3.
Figure 3.2: SkyeTek M1-Mini RFID reader pin-out

Figure 3.3: Crossbow Mica2Dot pin-out
As clearly shown in Figure 3.2 and Figure 3.3, the M1-Mini was designed to interface directly with Crossbow’s Mica2Dot mote over UART. Both the M1-Mini reader and the Mica2Dot mote were run at a baud rate of 9600 bits per second, the default setting for the reader, in order to communicate. The mote was programmed with an individual node ID and continuously checked for a tag in the reader’s read range every 2000 ms via polling. The mote sent the instruction code for the SELECT_TAG command to the reader using SkyeTek’s ASCII protocol. This command read the ID number of the RFID tag in the read range and delivered it to the mote.

Communication is the biggest expenditure of power in a WSN, and so this should be minimized at all costs. Therefore, only when a successful response was received from the reader at the mote, indicating a tag ID was read, was the tag ID number sent to the awaiting base station, Crossbow’s Mica2 mote. Table 3.2 indicates the possible reader responses to the SELECT_TAG command. The node mote and base station mote had a communication range of approximately 150 m in an enclosed area.

<table>
<thead>
<tr>
<th>Response Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x14</td>
<td>SELECT_TAG pass</td>
</tr>
<tr>
<td>0x1C</td>
<td>SELECT_TAG LOOP activate</td>
</tr>
<tr>
<td>0x94</td>
<td>SELECT_TAG fail</td>
</tr>
<tr>
<td>0x9C</td>
<td>SELECT_TAG LOOP cancel</td>
</tr>
</tbody>
</table>

Table 3.2: Possible SELECT_TAG command responses

As mentioned, Crossbow’s Mica2 mote served as the base station for this work. In this application, the base station’s role was to receive and process the RFID tag ID sent
by the Mica2Dot mote, as well as the mote’s individual software programmed ID. Any Mica2 mote can function as a base station when it is connected to a gateway board. The MIB510 board provides a serial interface for both programming and data communication. The MIB510 can be seen in Figure 3.4. Using the RFID tag ID and mote ID, the medical software written for this project was able to determine if that medication intake was appropriate for that patient. If it was an incorrect medication intake for the patient, the base station sent a command back to the original mote to toggle its red LED.

![Crossbow MIB510 gateway board](image)

Figure 3.4: Crossbow MIB510 gateway board

### 3.2. Medical Software System

The software portion of this project involves two separate applications. The first is called RIT Medi-Write, while the second is called RIT Medi-View. Medi-Write allows a physician to program an RFID tag, while Medi-View alerts the medical staff and patient when an incorrect medication application was attempted. Both were created using Microsoft Visual C++ 6.0.
3.2.1 RIT Medi-Write

When using Medi-Write, the practitioner will use SkyeTek’s M1 RFID reader, which follows the same protocol as the M1-Mini, but has native support for an RS232 host interface and a supply voltage of 1.8 V to 5.0 V. The M1 can be seen in Figure 3.5.

![SkycTek M1 reader](image)

As stated before, Medi-Write allows a practitioner to fill out all prescription information on an RFID tag, which is to be applied on a medication bottle. The tag will contain the patient’s name, the name of the prescription, the quantity of medication in the bottle, the dose size, the doses needed per day, and the software programmed node.
(reader) ID, which would be printed on the unit if this system were to be manufactured. A screenshot of the Medi-Write application can be seen in Figure 3.6.

![Figure 3.6: Medi-Write screenshot](image.png)

The practitioner places the RFID tag over the reader, fills in all the fields with the previously mentioned information, and hits the “Write Tag” button. To check that all information was appropriately entered, press the “Read Tag” button and the fields will be filled with the data they previously entered. If it is found a mistake was made after reading back the tag information, the practitioner can simply correct the appropriate field and re-write to the tag. The status box above the buttons informs the practitioner whether
the read or write has failed or completed successfully. The entire system setup for running Medi-Write can be seen in Figure 3.7.

![Figure 3.7: Entire Medi-Write system](image)

### 3.2.2 MySQL Database

Behind the scenes, when the “Write Tag” button is pressed a new entry will be placed in the database with all the information supplied by the practitioner, plus the RFID tag’s ID. The tag ID is stored under the database field name tagID. There is also a field
in the database, doseToday, for how many doses of that medication were taken for the current day. This is set to 0 when a new entry is added. A screenshot of the current database contents can be seen in Figure 3.8.

Figure 3.8: Database screenshot

The correlation between the field names in Medi-Write and the field names in the database this information will be stored under can be seen in Table 3.3. Re-writing to the tag will erase the previous entry for it in the database and enter a new entry with the correct data. The database will never retain two entries for the same tag.

<table>
<thead>
<tr>
<th>Medi-Write Field Name</th>
<th>Database Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Name</td>
<td>Name</td>
</tr>
<tr>
<td>Prescription</td>
<td>Rx</td>
</tr>
<tr>
<td>Quantity</td>
<td>QTY</td>
</tr>
<tr>
<td>Dose Size</td>
<td>doseSize</td>
</tr>
<tr>
<td>Doses per Day</td>
<td>doseDay</td>
</tr>
<tr>
<td>Reader ID</td>
<td>readerID</td>
</tr>
</tbody>
</table>

Table 3.3: Correlation between Medi-Write field names and database field names
3.2.3 RIT Medi-View

The Medi-View application is the real software centerpiece. It is in charge of alerting the healthcare personnel member at the central workstation of an inappropriate medication application. Additionally, it must instruct the base station mote to send an alert to the original sending patient mote. Since this application’s only graphical requirement is to provide alerts to the personnel at the workstation, the program is comprised of a simple terminal window, which displays various status messages, and pop-up windows, which appear when an inappropriate medication application was attempted. A screen shot of these status messages can be seen in Figure 3.9.

![Screenshot of status messages](image)

Figure 3.9: Screenshot of status messages

The terminal window displays status messages on the system. It relays to the user if it is successfully connected to the serial port and the database. Regardless of what the software is receiving over the serial port, it updates the database at the beginning of each
new day, 12 AM. It must reset the doseToday field in the database to 0 for every database entry. When this occurs, a message is also displayed in the terminal window. Finally, when the program does receive data via the serial port, it will display the tag and mote ID in the terminal window.

In addition to the pop-ups, there is an audible “beep” generated when inappropriate medication was attempted to be taken. The pop-ups and beeps are the only elements of the software the personnel truly need be concerned with, not the status messages in the terminal window. Once the base station mote has received a tag and mote ID, it sends it via serial communication, provided by the gateway MIB510 board, to the workstation. When the workstation has this information it can reference the database to ensure the patient is properly taking their medication. First, an account of the program’s actions will be provided for a correctly taken medication. This will then be followed by the situations where the medication was taken inappropriately. It should be stated this program is continually polling to see if a tag and mote ID combo has been received. A state diagram of the program can be seen in Figure 3.10.

In the event a patient is correctly taking their medication, the database doseToday field will be incremented by 1, and the QTY field will be decremented by whatever the value of the doseSize field is. In this case, no information is sent serially back to the base station. This is because an alert does not need to be sent to the patient.
Connect to the serial port

[Connection Fails]

Print error message to terminal window

[Connection Successful]

Print success to terminal window and connect to the database

[Connection Fails]

Print error message to terminal window

[Connection Successful]

Print success to terminal window and check if a new day has begun

[Yes]

Reset doseToday in database and print message to terminal window

[No]

Waiting for data from base station mote

[Data received]

Parse data into mote ID and tag ID and print both to the terminal window

[No data received]

Check if the tag is in our system

[Yes]

Alert patient and display pop-up

[No]

Check the patient is only taking their own medication

[Yes]

Check that the medication isn't empty

[No]

Alert patient and display pop-up

[Yes]

Check that the patient isn't about to overdose

[No]

Alert patient and display pop-up

[Yes]

Close database connection

Figure 3.10: Medi-View state diagram
There are several situations in which the patient and healthcare personnel will receive an alert. They include the given medication not being in the database, a patient attempting to take medication that is not their own, if the patient is out of pills, and finally, if they have taken more than the required doses of that medication for the day. For each of these errors, a pop-up will be displayed containing a time stamp, the patient who is incorrectly taking their medication, and the reason the application was incorrect.

It is possible to determine if a medication is in the database by looking up the RFID tag ID under the tagID field in the database. If it does not exist in the database it was never entered into the system, and an alert should be sent. To check if a patient is taking medication that is not their own, look up the tag ID in the database. If this entry doesn’t have a value in the readerID field of the database that matches the patient’s mote ID, the patient is attempting to take medication that is not theirs and an alert must be sent. It can easily be determined if a patient is out of pills by checking the QTY field in the database for the corresponding tag ID received. If the value in the QTY is 0, an alert should be sent so that the prescription may be refilled. Finally, and perhaps most importantly is the check to ensure a patient is not about to overdose on their medication. The doseToday field in the database should be queried for the corresponding tag ID. If the value in this field is equal to the value in the doseDay field, the patient should not be taking anymore medication. If an attempt is made an alert must be sent to keep the patient from overdosing. An example pop-up message for each of the previously described situations can be seen in Figure 3.11 through 3.14.
Figure 3.11: The given medication is not in the database

Figure 3.12: Patient attempting to take medication that is not theirs

Figure 3.13: Patient is out of pills

Figure 3.14: The patient is about to overdose
This has detailed all the components of the RIT Medi-View system. An illustrated overview of the entire system can be seen in Figure 3.15. The entire physical system setup needed to run Medi-View can be seen in Figure 3.16.

Figure 3.15: Medi-View system overview diagram

Figure 3.16: Entire Medi-View system
Chapter 4  RFID via Mica2 with Multi-hop

This chapter has two areas of focus. The first is switching the wireless mote platform from the Mica2Dot over to the Mica2. This was done for several advantageous reasons. The second area is the incorporation of multi-hop communication into the pre-existing system. Such a communication scheme allows the network to transmit over a regional area, such as a nursing home, while avoiding an expensive WLAN system.

4.1. Mica2 Advantages

The Mica2 and Mica2Dot represent the de facto standard platforms for sensor networks, differing only in form factors and slightly different core resources. They come from the same family of Crossbow motes, but the Mica2Dot is from the MPR5x0 series, while the Mica2 comes from the MPR4x0 series. Both are third generation platforms and as such are relatively stable. While all platforms used operated using a 433 MHz transceiver, a 900 MHz version is also available. Table 4.1 below gives a side-by-side comparison of both platforms’ radio properties.

By inspecting this table, it is clear that both motes possess the same physical radio. This is what allowed communication between the two motes, as described in Chapter 3. The differences between these two motes can be seen in the system core comparison, shown in Table 4.2. It is these differences, along with availability differences, which lead to the Mica2’s superiority.
### Table 4.1: Radio physical properties comparison

<table>
<thead>
<tr>
<th></th>
<th>Mica2</th>
<th>Mica2Dot</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Radio</strong></td>
<td>CC1000</td>
<td>CC1000</td>
</tr>
<tr>
<td><strong>Frequency Band</strong></td>
<td>315-916 MHz</td>
<td>315-916 MHz</td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
<td>38.4 kbps</td>
<td>38.4 kbps</td>
</tr>
<tr>
<td><strong>Setup Time</strong></td>
<td>&lt;50 msec</td>
<td>&lt;50 msec</td>
</tr>
<tr>
<td><strong>TX Powerctrl</strong></td>
<td>30 dB</td>
<td>30 dB</td>
</tr>
<tr>
<td><strong>TX Power</strong></td>
<td>+/- 10 dBm</td>
<td>+/- 10 dBm</td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td>-101 dBm</td>
<td>-101 dBm</td>
</tr>
<tr>
<td><strong>Modulation</strong></td>
<td>FSK</td>
<td>FSK</td>
</tr>
<tr>
<td><strong>Antenna</strong></td>
<td>wire</td>
<td>wire</td>
</tr>
<tr>
<td><strong>Outdoor Range</strong></td>
<td>150 m</td>
<td>150 m</td>
</tr>
<tr>
<td><strong>Channels</strong></td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 4.2: System core comparison

<table>
<thead>
<tr>
<th></th>
<th>Mica 2</th>
<th>Mica2Dot</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microcontroller</strong></td>
<td>ATmega128l</td>
<td>ATmega128l</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>8-Bit</td>
<td>8-Bit</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>7.3728 MHz</td>
<td>4 MHz</td>
</tr>
<tr>
<td><strong>Program Memory</strong></td>
<td>128 kB</td>
<td>128 kB</td>
</tr>
<tr>
<td><strong>Data Memory</strong></td>
<td>4 kB</td>
<td>4 kB</td>
</tr>
<tr>
<td><strong>Storage Memory</strong></td>
<td>512 kB</td>
<td>512 kB</td>
</tr>
<tr>
<td><strong>External IO</strong></td>
<td>51</td>
<td>18</td>
</tr>
<tr>
<td><strong>On-Board Sensors</strong></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>UI Components</strong></td>
<td>3 LEDs</td>
<td>1 LED</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>1856 mm²</td>
<td>492 mm²</td>
</tr>
</tbody>
</table>
Referring to Table 4.2, one of the noted differences between the motes is their number of user interface (UI) components. The Mica2Dot only comes equipped with one red LED. This means its one light must be reserved for alerts to the patient that medication is being incorrectly taken. Ideally, the system should be able to give the patient confirmation that a tag read was sent. Without this, a level of confidence is removed from the system, as the patient can’t be sure that their medication intake was actually sent to the base station mote. On the other hand, the Mica2 is equipped with three LEDs, one green, one yellow, and one red. For this platform, it was decided that the node mote’s yellow LED should toggle when it is sending to the base station. This allows the patient to confirm that their intake was sent, as well as receive notification in the event they incorrectly administered a drug. There is one light left over which could be used during additional development.

Another noted difference is the Mica2Dot’s diminished capacity for peripherals when compared to the Mica2. The Mica2Dot has only 18 input/output (IO) pins, but the Mica2 has 51. While the system implemented in this work did not require more IO pins than that provided by the Mica2Dot, it is envisioned that this wearable device will have many more requirements other than RFID medication supervision. As mentioned previously, the system should at the very least also possess an ECG sensor, though this was not implemented here. This additional sensor would of course require additional IO pins on the mote. It is also likely, due to further development, a designer would like to add another sensor to measure a different physiological signal. This again would require the availability of more IO pins on the mote. Clearly, as additional functionality is added
to the system, more IO pins need to be available. The Mica2 is far more extendable, making it a better choice for this system.

The information in Table 4.3 outlines the differences in the motes’ power supplies. While both motes have an external power supply range of 2.7 V – 3.3 V, the battery packs for each require different batteries. The Mica2Dot requires a 3 V CR2032 coin cell battery, while the Mica2 requires 2 AA batteries. From the incurred costs of this work, on average a single coin cell battery costs as much as 4 AA batteries. This means for the same cost, double the amount of Mica2’s could be powered when compared to the Mica2Dot. Additionally, AA batteries have a lifetime of approximately 2700 mA-hours, while coin cell CR2032 batteries only have a lifetime of approximately 225 mA-hours. Since the considerations for this work include minimizing costs for the already expensive healthcare industry, the Mica2 is a more economical choice.

<table>
<thead>
<tr>
<th></th>
<th>Mica2</th>
<th>Mica2Dot</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Battery Power</strong></td>
<td>2 AAs</td>
<td>1 CR2032</td>
</tr>
<tr>
<td><strong>External Power Supply</strong></td>
<td>2.7 V – 3.3 V</td>
<td>2.7 V – 3.3 V</td>
</tr>
<tr>
<td><strong>Battery Supply Cost</strong></td>
<td>$2</td>
<td>$5</td>
</tr>
</tbody>
</table>

Table 4.3: Power supply comparison

Yet another key difference between the two motes is one that, again, cannot be seen in Table 4.2. Crossbow has stated that the Mica2Dot’s UART is not stable [15], and recommends using the Mica2 as a base station, since it will always be utilizing the UART to communicate serially with the PC. For this application, all motes in the system, not
just the base station, utilize their UART to communicate with their attached RFID reader. Naturally this makes the Mica2 a better choice for reliability.

Additionally, as of October 2007, Crossbow announced they will no longer offer or support the Mica2Dot [16]. Therefore, if this system is to receive any future development, additional Mica2Dot motes will not be available for purchase. This work was started before Crossbow had made this announcement, but a system cannot be based on obsolete hardware. This reason alone would be enough to make the switch from the Mica2Dot to the Mica2 mote platform.

The final difference to be noted is the discrepancy in the CPU speeds of the Mica2 and Mica2dot, seen in Table 4.2. While the Mica2Dot’s CPU only runs at 4 MHz, the Mica2’s CPU runs at almost double that, 7.3728 MHz. This means the Mica2’s processing time is shorter and faster compared to the Mica2Dot for the same task. Therefore, in applications where the mote is required to perform computations, the Mica2 is a better choice. Multi-hop communication, discussed later in this chapter, requires each node to find a path to the base station if it cannot communicate with it directly. The algorithm required for path establishment and discovery demands a high CPU clock cycle if it is to be done quickly. For multi-hop communication, the Mica2 is clearly the better choice.

4.2. Mote-Reader Interface

Unlike the Mica2Dot, the Mica2 was not created to directly interface with the M1-Mini. Instead, additional hardware needed to be purchased to break out the Mica2’s IO pins. This work utilized Crossbow’s MDA100CB digital acquisition (DAC) board,
which connected to the Mica2 via its mezzanine connector. A picture of the DAC board can be seen in Figure 4.1. The MDA100 series sensor boards have a precision thermistor, a light sensor/photocell, and general prototyping area.

![Figure 4.1: MDA100CB DAC board](image)

The prototyping area supports connection to all eight channels of the mote’s analog to digital converter (ADC0–7), both USART serial ports, and the I²C digital communications bus. The prototyping area also has a series of 45 unconnected solder holes that are used for breadboard of circuitry to connect other sensors and devices to the mote. Table 4.4 shows the prototyping area layout. Using this table, the M1-Mini was connected to the MDA100CB. The M1-Mini’s RX pin was connected to the MDA100CB’s TX, while the M1-Mini’s TX pin was connected to the MDA100CB’s RX. Power and ground were also connected between the MDA100CB and M1-Mini. A picture of the two connected can be seen in Figure 4.2.
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>VCC</td>
<td>VCC</td>
<td>VCC</td>
</tr>
<tr>
<td>2</td>
<td>OPEN</td>
<td>OPEN</td>
<td>USART1_CK</td>
<td>INT3</td>
<td>ADC2</td>
<td>PW0</td>
</tr>
<tr>
<td>3</td>
<td>OPEN</td>
<td>OPEN</td>
<td>UART0_RX</td>
<td>INT2&quot;</td>
<td>ADC1&quot;</td>
<td>PW1&quot;</td>
</tr>
<tr>
<td>4</td>
<td>OPEN</td>
<td>OPEN</td>
<td>UART0_TX</td>
<td>INT1</td>
<td>ADC0</td>
<td>PW2</td>
</tr>
<tr>
<td>5</td>
<td>OPEN</td>
<td>OPEN</td>
<td>SPI_SCK</td>
<td>INT0</td>
<td>THERM_PWR</td>
<td>PW3</td>
</tr>
<tr>
<td>6</td>
<td>OPEN</td>
<td>OPEN</td>
<td>USART1_RX</td>
<td>BAT_MON</td>
<td>THRU1</td>
<td>PW4</td>
</tr>
<tr>
<td>7</td>
<td>OPEN</td>
<td>OPEN</td>
<td>USART1_TX</td>
<td>LED3</td>
<td>THRU2</td>
<td>PW5</td>
</tr>
<tr>
<td>8</td>
<td>OPEN</td>
<td>OPEN</td>
<td>I2C_CLK</td>
<td>LED2</td>
<td>THRU3</td>
<td>PW6</td>
</tr>
<tr>
<td>9</td>
<td>OPEN</td>
<td>OPEN</td>
<td>I2C_DATA</td>
<td>LED1</td>
<td>RSTN</td>
<td>ADC7</td>
</tr>
<tr>
<td>10</td>
<td>OPEN</td>
<td>OPEN</td>
<td>PWM0</td>
<td>RD</td>
<td>PWM1B</td>
<td>ADC6</td>
</tr>
<tr>
<td>11</td>
<td>OPEN</td>
<td>OPEN</td>
<td>PWM1A</td>
<td>WR</td>
<td>OPEN</td>
<td>ADC5</td>
</tr>
<tr>
<td>12</td>
<td>OPEN</td>
<td>OPEN</td>
<td>AC+</td>
<td>ALE</td>
<td>OPEN</td>
<td>ADC4</td>
</tr>
<tr>
<td>13</td>
<td>OPEN</td>
<td>OPEN</td>
<td>AC-</td>
<td>PW7</td>
<td>OPEN</td>
<td>ADC3</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>VCC</td>
<td>VCC</td>
<td>VCC</td>
</tr>
<tr>
<td>15</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>16</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>17</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Table 4.4: MDA100CB prototyping area layout

Figure 4.2: Connected MDA100CB and M1-Mini
4.3. Multi-hop Communication

As previously stated, the indoor communication range for the Mica2 is approximately 150 m. The application environment in which this system should be applied, a nursing home, is clearly more spacious than this distance. Therefore, a method of communicating information between the base station mote and a patient must exist even when they are separated by more than 150 m. This can be achieved by using a hop-to-hop, or in this case patient-to-patient, communication scheme where information from patient A, outside the base station’s range, is routed through patient B, who is within both patient A’s range and the base station’s range. An illustration of this can be seen in Figure 4.3. It assumed that all patients will not be 150 m away from the base station mote at the same time and that each patient will never be more than 150 m away from another patient.

![Figure 4.3: Multi-hop communication illustration](image)

Before discussing multi-hop communication, it is important to understand how single hop communication occurs under TinyOS. This occurs through active messages, the TinyOS networking primitive.
4.3.1 TinyOS Active Messages

The primitive, active messages (AM), is a simple message-based networking abstraction. Depending on the platform, the abstraction differs to work with the specific networking stack. This work utilized the Mica2 platform, so it will be featured here. The default media access used for all radio communication is carrier sensed multiple access (CSMA).

The macro component called GenericComm encapsulates the TinyOS network stack. It provides the AM communication interfaces SendMsg and ReceiveMsg. The contents of both of these interfaces, send and sendDone for SendMsg, and receive for ReceiveMsg, can be seen in Figure 4.4. Applications and the network stack exchange fixed size message buffers through pointers. Upon a successful call to send, GenericComm returns the provided buffer to the sender and signals the sendDone event.

```c
interface SendMsg { // single-hop networking
    command result_t send(uint16_t addr, uint8_t len,
                          TOS_MsgPtr msg);
    event result_t sendDone(TOS_MsgPtr msg, result_t success);
}

interface ReceiveMsg { // single-hop networking
    event TOS_MsgPtr receive(TOS_MsgPtr m);
}
```

**Figure 4.4: SendMsg and ReceiveMsg interfaces**

While senders provide their own storage for messages, GenericComm only provides limited receive buffering, requiring a component to return a buffer to the radio stack when it receives a message [17]. The stack uses this buffer to hold the next
message as it arrives. Due to this limited buffering, a component usually consumes and returns the buffer for later use. In the event a component exhausts all its buffering, it must determine what should be dropped. Components that need to accumulate several packets before processing must allocate buffers statically and locally, rather than relying on the network stack.

The single-hop packet’s format can be seen in Figure 4.5. The first two bytes of a received packet are used to identify the destination of the packet. The next three bytes hold the ID of the message handler that is to be invoked on the packet, the length of the message, and the message’s group ID, respectively. The AM component first checks that the address matches the local address and then it invokes the listed handler, passing on the remaining bytes of the packet. In the event the message is bound for a handler that is not present on the receiving mote, the packet is ignored.

![Figure 4.5: Single-hop packet format](image)

Active messages are slightly modified for multi-hop networking implementations in TinyOS. To begin with, these implementations use the Send and Intercept interfaces, shown in Figure 4.6. The getBuffer command in the Send interface allows for packet encapsulation, while the Intercept event is signaled when a packet to forward is received. These interfaces enhance the AM abstraction by supporting multi-hop communication.
The message format for multi-hop routing is different as well. The format, seen in Figure 4.7, allocates seven additional bytes before the data field, and allows a maximum of 4 hops. The intermediate hops of the route are stored in the four bytes $R_1, R_2, R_3,$ and $R_4,$ while the number of hops left is stored in one byte, $N.$ The two remaining bytes are used to store the source node of the packet, $S,$ and the handler ID to be invoked once the message arrives at its destination, $H_f.$

![Multi-hop packet format](image)

- **N** - Number of Hops
- **H_f** - Destination Handler
- **R_1, R_2, R_3, R_4** - Route Hops
- **S** - Sending Node
- **D_0, D_1** - Payload

The TinyOS-1.1 release and later include library components that provide ad-hoc multi-hop routing for sensor-to-sensor network applications. The implementation uses a
shortest-path-first algorithm with a single destination node, the base station, and active
two-way link quality estimation. The multi-hop communication scheme used in this
work was based on this pre-existing configuration, known as MultiHopRouter, which is
further explained in the next section.

4.3.2 MultiHopRouter Multi-hop Protocol

The MultiHopRouter protocol is founded on tree-based routing. Tree based
routing is primarily based on two pieces of information: a parent node identifier and a
hop-count (the parent’s hop-count plus one). A routing tree is built via local broadcasts
from the root, in this case the base station node, followed by selective retransmission
from its descendents. A node routes a packet by transmitting it with the parents as the
designated recipient. The parent does the same to its parent, until the packet reaches the
root of the tree, the base station node.

The TinyOS MultiHopRouter protocol, which is widely used by the sensor
network community, is based on the shortest-path algorithm that forms a spanning tree so
that the path from any mote in the sensor field to the sink uses the least number of hops.
Route control messages are periodically broadcast from each node in the network to
estimate the routing cost and monitor link quality. MultiHopRouter uses the least number
of hops as the primary metric with link quality as a tiebreaker. It provides support for
retransmission and output queuing.

For route establishment, a spanning tree is built by the base station broadcasting
itself as the root of the tree. Every 5 seconds, a node within the base station’s range
broadcasts a packet which includes its hop count from the base station and its node ID.
Once a neighboring node gets that packet, it chooses its parent based on the minimal hop count, with link quality as a tiebreaker. Any data packet received is forwarded on if the given node is not the destination. This is a one way tree traversal, with communication only going from the nodes to the root.

If a node wants to send a message to the base station, it should use the Send.getBuffer command to get a pointer to the data region of a packet. Afterward, the Send.send command should be called in order to send the packet to the next hop. Upon the reception of a message, the AM component signals the Receive interface if the destination is the local node or the Intercept interface otherwise. If the node is not the base station, it should use Send.send to forward the packet.

The MultiHopRouter multi-hop implementation consists of two core modules in TinyOS, MultiHopEngineM and MultiHopLEPSM. These are wired together in a single configuration, MultiHopRouter [18]. Figure 4.8 provides an overview of the configuration.

MultiHopEngineM provides the overall packet movement logic for multi-hop functionality [18]. Using the RouteSelect interface, it determines the next hop path and forwards it out the parameterized SendMsg port. The mechanics of this module are independent of route selection. It only requires that the RouteSelect and RouteControl interfaces be available from the algorithmic component.

MultiHopLEPSM provides the link estimation and parent selection (LEPS) mechanisms for the multi-hop implementation [18]. The module monitors all traffic received at the node via the Snoop port and directly receives single-hop route update messages, AM_MULTIHOPMSG. These update messages may be sent from neighbors
within the single-hop range. Internally, this module manages the nearest available neighbors and decides the next hop destination based on shortest path semantics. Presently, the destination is identified as the node with TOS_LOCAL_ADDRESS set to 0. By default, the module sends a route update message once every 10 seconds and re-computes after 50 seconds (5 route update messages). MultiHopLEPSM may be interchanged with other modules that implement different selection algorithms.

MultiHopRouter connects MultiHopEngineM and MultiHopLEPSM with other necessary components [18]. The configuration exports the Receive, Send, Intercept, and
Snoop as Intercept ports to applications. The SendMsg port of MultiHopEngineM is wired to the QueuedSend library components for queuing outbound packets, both forwarded and locally originated. The ReceiveMsg and SendMsg ports of MultiHopLEPSM are wired to the AM_MULTIHOPMSG parameter of the communication provider for the purpose of exchanging single-hop route updates with neighbors.

4.4. Multi-hop Medi-View System

While MultiHopRouter ensured packets would be successfully delivered to the base station, a method of sending alerts from the base station to the correct patient needed to be constructed. The first step is to have the patient node send its mote ID, a 16-bit unsigned integer, along with the RFID tag ID, an 8-bit unsigned integer array of length 16, as its message payload. When this is received at the base station the appropriate processing takes place to determine if the medication is being taken properly. In the event there was an incorrect application, the base station sends a message with the node’s ID, also a 16-bit unsigned integer, as its payload. By examining this value, the patient node can determine if the message it received was intended for it. If so, the red LED on the mote will toggle, if not, that node will forward the packet on, until it gets to the appropriate destination node.

In this work one base station was used along with three nodes. Multi-hop communication was tested inside an enclosed building. Two of the nodes were only able to directly communicate with one another, while the third was able to directly communicate with one other node and the base station. The nodes were separated by
approximately 150 m each. The experiment field and mote placement can be seen in Figure 4.9. It was found that from mote power-up to route discovery the delay was approximately 90 seconds.

![Figure 4.9: Experiment field and mote placement](image)

The entire multi-hop communication Medi-View system, which utilized Mica2 motes as the patient nodes, can be seen in Figure 4.10. Clearly the Medi-Write system did not need to be altered since it is simply a means of programming the RFID tags.
Figure 4.10: Entire multi-hop Medi-View system
Chapter 5  The Extended Kalman Filter

The final piece of this work required that changes be made to the discrete Kalman filter in order to use its non-linear counterpart, the extended Kalman filter. A Kalman filter that linearizes about the current mean and covariance is referred to as an EKF. Additionally, an ECG was modeled with governing equations which were supplied to the filter.

5.1. Discrete Kalman Filter Alterations

For the EKF, the state vector is now governed by the non-linear difference equation seen in Equation 5.1. Here, \( f \) represents the non-linear function that relates the state at the previous time step, \( k - 1 \), to the state at the current time step, \( k \). An optional driving function may be included as \( u_{k-1} \), and \( w \) represents the zero-mean Gaussian process noise. The measurement equation has also changed to Equation 5.2. Similarly, \( h \) is the non-linear function that relates \( x_k \) to the measurement \( z_k \), while \( v_k \) is the measurement noise. Individual noise values, \( w_k \) and \( v_k \), are not known at each time step and there is no way to predict noise. Instead, an \( a \) priori estimate, from Equation 5.3, will replace Equation 5.1 and a measurement estimate, from Equation 5.4, will replace Equation 5.2. In Equation 5.3, \( \hat{x}_{k-1} \) is some \( a \) posteriori estimate of the state from the previous time step.

\[
x_k = f(x_{k-1}, u_{k-1}, w_{k-1})
\]  

(5.1)
\[ z_k = h(x_k, v_k) \]  \hfill (5.2)

\[ \hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \]  \hfill (5.3)

\[ \hat{z}_k = h(\hat{x}_k^-, 0) \]  \hfill (5.4)

New linearizing equations are needed to estimate a process with non-linear relationships. The governing equations seen in Equations 5.5 and 5.6 linearize about Equations 5.3 and 5.4, respectively. The equations for the matrices \( A, W, H, \) and \( V \), can be seen in Equations 5.7 through 5.10, respectively. For simplicity, the time subscript \( k \) is not used in the notation for these matrices even though they are in fact different at each time step. The information in Table 5.1 details what each variable represents.

\[ x_k \approx \hat{x}_k^- + A(x_{k-1} - \hat{x}_{k-1}) + W_{k-1} \]  \hfill (5.5)

\[ z_k \approx \hat{z}_k + H(x_k - \hat{x}_k^-) + V_{k} \]  \hfill (5.6)

\[ A_{i,j} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0) \]  \hfill (5.7)

\[ W_{i,j} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0) \]  \hfill (5.8)
\[ H_{[i,j]} = \frac{\partial h_{[i]}(\hat{x}_k, 0)}{\partial \hat{x}_{[j]}_{[1]}} \]  

(5.9)

\[ V_{[i,j]} = \frac{\partial h_{[i]}(\hat{x}_k, 0)}{\partial v_{[j]}} \]  

(5.10)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_k )</td>
<td>Actual state vector</td>
</tr>
<tr>
<td>( z_k )</td>
<td>Actual measurement vector</td>
</tr>
<tr>
<td>( \hat{x}_k )</td>
<td>An a priori state estimate at time ( k )</td>
</tr>
<tr>
<td>( \hat{z}_k )</td>
<td>Approximate measurement vector at time ( k )</td>
</tr>
<tr>
<td>( \hat{x}_k )</td>
<td>An a posteriori state estimate at time ( k )</td>
</tr>
<tr>
<td>( w_k )</td>
<td>Random variable representing process noise</td>
</tr>
<tr>
<td>( v_k )</td>
<td>Random variable representing measurement noise</td>
</tr>
<tr>
<td>( A )</td>
<td>The Jacobian matrix of partial derivatives of ( f ) with respect to ( x )</td>
</tr>
<tr>
<td>( W )</td>
<td>The Jacobian matrix of partial derivatives of ( f ) with respect to ( w )</td>
</tr>
<tr>
<td>( H )</td>
<td>The Jacobian matrix of partial derivatives of ( h ) with respect to ( x )</td>
</tr>
<tr>
<td>( V )</td>
<td>The Jacobian matrix of partial derivatives of ( h ) with respect to ( v )</td>
</tr>
</tbody>
</table>

Table 5.1: Description of the variables in 5.5 through 5.10

A new prediction error and measurement residual can now be defined. These can be seen in Equations 5.11 and 5.12, respectively. The measurement residual reflects the discrepancy between the predicted measurement, \( \hat{z}_k \), and the actual measurement, \( z_k \). Unfortunately, in practice one does not have access to \( x_k \) in Equation 5.11, since it is the actual state vector attempting to be estimated. One does have access to \( z_k \) in Equation 5.12 though, since it is the actual measurement that one is using to make an estimate. By
utilizing Equations 5.11 and 5.12, approximate linear governing equations for an error process can be written. These can be seen in Equations 5.13 and 5.14. It should be noted $\varepsilon_k$ and $\eta_k$ represent new independent random variables having zero mean and covariance matrices $WQ_kW^T$ and $VR_kV^T$, respectively. Recall $Q$ and $R$ are the process noise covariance and measurement noise covariance matrices, respectively. Additionally, $\tilde{e}_{x_k}$ has a covariance matrix of $E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T]$.

\begin{align*}
\tilde{e}_{x_k} &\equiv x_k - \hat{x}_k^- \\
\tilde{e}_{z_k} &\equiv z_k - \hat{z}_k \\
\tilde{e}_{x_k} &\approx A(x_{k-1} - \hat{x}_{k-1}^-) + \varepsilon_k \\
\tilde{e}_{z_k} &\approx H\tilde{e}_{x_k} + \eta_k
\end{align*}

(5.11)  (5.12)  (5.13)  (5.14)

It can be seen that Equations 5.13 and 5.14 are linear, and resemble the original discrete Kalman filter equations in Equations 2.2 and 2.3, respectively. This suggests there is a better way to estimate the error prediction of Equation 5.14. This new estimate of $\tilde{e}_{x_k}$, called $\hat{e}_k$, can be found by using Equation 5.12 and a second hypothetical Kalman filter. The Kalman filter equation used for this estimate can be seen in Equation 5.15.

\[
\hat{e}_k = K_k \tilde{e}_{z_k}
\]

(5.15)
Once Equation 5.15 is attained, it can be used in Equation 5.16 to obtain an *a posteriori* state estimate for the original non-linear process. By substituting Equation 5.15 into Equation 5.16, Equation 5.17 is obtained. Finally, by substituting Equation 5.12 into Equation 5.17, Equation 5.18 is achieved, which will be used for the measurement update in the EKF. Note that $\hat{x}_k$ and $\hat{z}_k$ come from Equations 5.3 and 5.4, and the Kalman gain, $K$, comes from Equation 2.12, with the appropriate substitution for the measurement error covariance.

$$\hat{x}_k = \hat{x}_k^- + \hat{e}_k$$  \hspace{1cm} (5.16)

$$\hat{x}_k = \hat{x}_k^- + K_k \tilde{e}_z_k$$  \hspace{1cm} (5.17)

$$\hat{x}_k = \hat{x}_k + K_k \left( z_k - \hat{z}_k \right)$$  \hspace{1cm} (5.18)

The complete set of EKF equations can be seen in Table 5.2 and Table 5.3. The tables separately group those equations used for time updates and those for measurement updates, respectively. The appropriate subscript has been attached to the matrices $A$, $W$, $H$, and $V$ to reinforce the notion that they are different at each time step, and therefore must be recomputed each time. Just as with the discrete Kalman filter, the time update equations for the EKF project the state and covariance estimates from the previous time, $k - 1$, to the current time, $k$. The entire cyclic process of predicting and correcting this prediction can be seen in Figure 5.1 [9]. Just as in Chapter 2, the equation for $P_k$ in
Figure 5.1 utilizes a simplification that is not numerically stable and as such is not referred to in this chapter.

\[ \hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \]
\[ P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \]

**Table 5.2: EKF time update equations**

\[ K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \]
\[ \hat{x}_k = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \]
\[ P_k = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \]

**Table 5.3: EKF measurement update equations**

![Diagram of EKF recursive nature](image)

**Figure 5.1: A complete view of the extended Kalman filter’s recursive nature**
An important feature of the EKF is that $H_k$ in the equation for the Kalman gain, $K_k$, serves to correctly emphasize or magnify only the relevant component of the measurement information. $H_k$ affects the Kalman gain so that it only magnifies the portion of the residual, $z_k - h(\hat{x}_k^-)$, that does affect the state.

### 5.2. ECG Modeling

With the EKF modifications made, all that is left is to determine the equation which governs an ECG and apply it to the filter. Produced by an electrocardiograph, the signal is constructed by measuring electrical potentials between various points of the body using a galvanometer. Figure 5.2 shows an example of a normal ECG trace, which consists of a P wave, a QRS complex and a T wave. Understanding the interpretation of this signal is not in the scope of this work.

---

**Figure 5.2: Example of a normal ECG trace**
While this work mainly focuses on using the EKF to recover wireless transmission errors, it can simultaneously be used to correct baseline wander of the ECG. The baseline is the resting potential of the ECG between the P, QRS, and T waves. Baseline wander occurs whether the ECG is sent wirelessly or not, as it is due to slipping or moist electrodes on the body. Since the baseline of an ECG is used to diagnose many different cardiac diseases, it is important to receive an accurate portrayal of this part of the signal. For instance, an ST segment below the baseline implies shortage of blood flow and oxygen.

McSharry et al. have proposed a synthetic ECG generator, which is based on a non-linear dynamic model [19]. This model has several parameters, $P, Q, R, S,$ and $T,$ which come from the ECG and makes it adaptable to many normal and abnormal signals. The dynamic model consists of a three dimensional state equation, which generates a trajectory with the coordinates $(x, y, z)$. These equations may be seen in Equations 5.19 through 5.21. The variables $\alpha, \Delta \theta_i,$ and $\theta$ are given in Equations 5.22 through 5.24. Note that Equation 5.24 is the four quadrant arctangent of the real parts of the elements of $x$ and $y$, with the bounds given in Equation 5.25. The variable $\omega$ is the angular velocity of the trajectory as it moves around the limit cycle. The baseline wander of the ECG signal has been modeled with $z_0$. Some typical parameters for the synthetic ECG model can be seen in Table 5.4.

\[
x' = \alpha x - \omega y \quad \text{(5.19)}
\]

\[
y' = \alpha y + \omega x \quad \text{(5.20)}
\]
\[ z' = -\sum_{i\in\{P,Q,R,S,T\}} a_i \Delta \theta_i \exp\left( -\frac{\Delta \theta_i^2}{2\delta_i^2} \right) - (z - z_0) \]  \hspace{1cm} (5.21)

\[ \alpha = 1 - \sqrt{x^2 + y^2} \]  \hspace{1cm} (5.22)

\[ \Delta \theta_i = (\theta_i - \theta_j) \mod (2\pi) \]  \hspace{1cm} (5.23)

\[ \theta = \arctan 2(y, x) \]  \hspace{1cm} (5.24)

\[ -\pi \leq \arctan(y, x) \leq \pi \]  \hspace{1cm} (5.25)

<table>
<thead>
<tr>
<th>Index (i)</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (Sec.)</td>
<td>-0.2</td>
<td>-0.05</td>
<td>0</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>( \theta_i ) (rads.)</td>
<td>-\pi/3</td>
<td>-\pi/12</td>
<td>0</td>
<td>\pi/12</td>
<td>\pi/2</td>
</tr>
<tr>
<td>( a_i )</td>
<td>1.2</td>
<td>-5.0</td>
<td>30.0</td>
<td>-7.5</td>
<td>0.75</td>
</tr>
<tr>
<td>( b_i )</td>
<td>0.25</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 5.4: Typical parameters of the synthetic ECG model

The three dimensional trajectory which is generated consists of a circular limit cycle which is pushed up and down when it approaches one of the \( P, Q, R, S \) or \( T \) points. In fact, each of the components of the ECG waveform has been modeled with a Gaussian function, which is located at a specific angle. This can be seen in Equations 5.19 through 5.21 by neglecting the baseline wander term, \( z - z_0 \), and integrating the \( z' \) equation. The projection of the three dimensional trajectory on the \( z \) axis gives a synthetic ECG signal.
5.3. Implementation

With the changes necessary for the EKF documented and a modeling equation for the ECG completed, these two elements simply need to be put together to filter a noisy wireless signal. From here, this work utilized the accomplishments of Sameni et al., who were able to linearize the ECG model and apply the EKF to it in MATLAB [20]. Their application was non-invasive fetal ECG extraction. This same work can applied in a completely different area, wireless data recovery.

The actual ECG data came from the Physionet.org PhysioBank database of physiological signals [21]. A sample of one of these normal ECG signals can be seen in Figure 5.3. While the signal does have the baseline wander typical of all ECG signals, since it wasn’t transmitted wirelessly there are no missing pieces of data. Therefore, the work of Sameni et al. had to be modified to remove data from these real ECG signals in order to simulate wireless data loss.

![Figure 5.3: Normal ECG signal](image-url)
As stated in Chapter 2, the error rate of wireless transmissions is in the range of 2% – 10%. For this work, data loss was simulated at 10%. Figure 5.4 shows a zoomed in view of the original signal in Figure 5.3 along with the signal which has 10% of all its data point removed. The MATLAB program, rddata.m, took a number of parameters for this filtering process, all of which can be seen in Table 5.5, in addition to the execution command itself.

![Figure 5.4: ECG from 5.3 with simulated wireless errors](image)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dir</strong></td>
<td>The directory where the Physionet ECG resides</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>The name of the ECG data file</td>
</tr>
<tr>
<td><strong>Samp</strong></td>
<td>The number of samples to take for the ECG file</td>
</tr>
<tr>
<td><strong>Psamp</strong></td>
<td>The periodic sample size to apply error loss to</td>
</tr>
<tr>
<td><strong>Err</strong></td>
<td>The number of data points to remove from the periodic sample size</td>
</tr>
</tbody>
</table>

**Command**

```
rddata('Dir', 'File', 'Samp', 'Psamp', 'Err')
```

**Table 5.5: rddata parameters and execution command**

61
The normal ECG signal filtered by the EKF can be seen in Figure 5.5. The final filtered signal, seen in green, is centered at 0 and its simulated wireless data loss has been recovered. The baseline wander removal was performed in the signal seen in red. Two filtered abnormal ECG signals can be seen in Figure 5.6 and 5.7, where both patients had a cardiac arrhythmia.

By analyzing all three filtered wave forms, it can be seen that whether the ECG is categorized as normal or abnormal, the wireless data loss can be recovered. This is all due to the parameterized software, which allows the $P$, $QRS$, and $T$ waves to be specified individually. Hardware related to ECG reading, interpretation, and recovery was not implemented in this work. Clearly though, it can be seen that this method of using the EKF to recover data lost due to a wireless transmission proved successful in simulation.

![Figure 5.5: Filtered ECG from 5.3](image-url)
Figure 5.6: Filtered abnormal ECG with arrhythmia

Figure 5.7: Filtered abnormal ECG with arrhythmia
Chapter 6  Conclusion

The objective of this thesis was to reduce healthcare costs by creating a means to remotely monitor medication intake of nursing home patients. Since technology to remotely monitor a patient’s ECG already exists, this work sought to provide a reliable ECG signal to the healthcare professional monitoring the patient. While technological advancements have resulted in changes to many aspects of daily life, there is still a significant gap between the existing solutions and the needs in the medical field. One of the most pressing issues in medical care today is the high cost of healthcare and health insurance. Although many patients are capable of following their medication regimen on their own, thousands of dollars are spent on extra staffing to administer and supervise a patient’s medication intake. Thanks to the recent developments in wireless sensor networks and RFID, such costly expenditures may be avoided.

This thesis is an endeavor to suggest a solution utilizing these technologies and provide a remote medication monitoring system designed for the medical environment. This system provides continuous medication intake monitoring without the exhaustion of any manpower. In fact, it is intended to give support to the current healthcare environments and free up medical professionals for more urgent functions. By automating the medication monitoring process, the most up to date information for all patients is made available at all times.

This system is composed of two major components. Wearable mobile platforms are distributed to patients, based on wireless sensor network technology. These mobile platforms are responsible for gathering patient medication intake using an RFID monitoring system. The gathered data is transmitted wirelessly over radio to the
receiving base station. The base station is connected to a workstation where the data is processed using the software created specifically for this work, RIT Medi-View. The outcome determined by this software is sent over radio back to the patient, to inform them if they are incorrectly taking their medication.

The second part of the system is based on TinyOS’s multi-hop communication protocol, MultiHopRouter. This allows patient nodes to communicate with the base station even if they are out of its transmission range. For a regional area like a nursing home, the target environment for this application, expensive WLAN infrastructures should be avoided. By utilizing multi-hop communication, a large transmission range can be achieved without sacrificing cost minimization.

In addition to these functionalities, a simulation was provided to ensure accuracy and reliability when remotely monitoring ECG signals. The data contained in an ECG is highly sensitive, since every second of information is so meaningful. A disastrous event like heart failure could begin to occur in mere seconds. This information cannot be lost due to its wireless transmission. It is imperative a remote ECG monitoring system employ a means for error recovery. The error recovery technique implemented here is the extended Kalman filter, which is designed for non-linear applications, such as an ECG signal.

Finally there are future expansion possibilities which would greatly improve the system. Currently, the RIT Medi-View software displays a modal dialogue box when alerting healthcare personnel of an inappropriate medication application. This modal dialogue box halts the system from displaying any other alerts that may be received until the current one is closed. Information is not lost, but its display is delayed. By simply
switching to a non-modal dialogue box this could be avoided. Since this work sought to achieve a simple proof of concept for this medication system, the extra time was not allocated towards this. This is an important expansion however, because the intention of this system is to decrease the amount of time required for medical response to patients in need.

Additionally, future work could also include improving the multi-hop communication scheme. Currently, when a new node enters the network, it takes 90 seconds for it to establish a path to the base station due to link quality estimation. Again, since this can delay response time it should be investigated for future versions of this system. Lastly, while the discrete Kalman filter is an optimal filter, the extended Kalman filter is no longer in the field of linear estimation theory and thus cannot provide an optimal estimate.

Although there have been many research efforts in both the fields of medical asset monitoring and remote ECG monitoring, most of them stay theoretical at best. This thesis marks an attempt to bridge the two research fields by providing a product that is realizable and would directly benefit the consumers in the medical field.
Bibliography


[18] TinyOS “Multihop Routing,”


[21] PhysioNet ECG database. Available from: