Prototyping the new-guard portable device for radiation detection

Aaron Ponzio
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/2315

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
PROTOTYPING THE NEW-GUARD PORTABLE DEVICE FOR RADIATION DETECTION

by

Aaron Ponzio
Bachelors of Science in Computer Engineering
University of Nevada, Las Vegas
2006

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Electrical Engineering
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2008
Thesis Approval
The Graduate College
University of Nevada, Las Vegas

April 16, 2008

The Thesis prepared by
Aaron Ponzio

Entitled
Prototyping The New-Guard Portable Device For Radiation Detection

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Prototyping the New-GUARD Portable Device for Radiation Detection

by

Aaron Ponzio

Dr. Shahram Latifi, Examination Committee Chair
Professor of Electrical and Computer Engineering
University of Nevada, Las Vegas

A novel and efficient radiation detection algorithm fused together with a new generation detection unit will produce an effective detector to battle field radiation measurement problems and reduce field surveyor work hazards. The SPRT (Sequential Probability Ratio Test) algorithm helps increase reliability and speed of radiation detection; the new generation detector improves the detector’s flexibilities, applications, and functions. A prototype system of the New Generation User Adaptable Radiation Detector (New-GUARD) is developed and analyzed to determine the system feasibility of usage, development, and safety for radiation detection. Development stages include the implementation of the Graphical User Interface (GUI), building the ideal hardware components for the detection unit, and the integration of the two to form the complete New-GUARD system. The New-GUARD GUI is created using the Visual Basic .NET programming language along with the .NET Compact Framework and Windows Mobile 6 SDK for all Windows Mobile based devices. The hardware portion is implemented using a microcontroller that sends data out to the GUI via a wireless transmission.
medium. New-GUARD system performance metrics are provided to show real-time processing capabilities. Lastly, alternative New-GUARD hardware designs as well as a future plan to eliminate human presence entirely from radiation fields are discussed.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. iii

LIST OF FIGURES .................................................................................................................... vii

LIST OF TABLES ....................................................................................................................... viii

ACKNOWLEDGEMENTS .......................................................................................................... ix

CHAPTER 1  INTRODUCTION .............................................................................................. 1
  1.1 Thesis Outline ..................................................................................................................... 2

CHAPTER 2  BACKGROUND ............................................................................................... 4
  2.1 Algorithm Background ....................................................................................................... 4
    2.1.1 Sequential Probability Ratio Test (SPRT) ................................................................. 5
  2.2 SPRT for radiation detection ............................................................................................. 6
    2.2.1 Hypothesis settings ................................................................................................... 10
    2.2.2 Alarm computation ................................................................................................... 11
    2.2.3 Dynamic background updating .............................................................................. 12
    2.2.4 Maximum Likelihood Estimator (MLE) ................................................................. 13
    2.2.5 ilap .......................................................................................................................... 14
  2.3 GUI and hardware integration ......................................................................................... 15
  2.4 Multi-threading ................................................................................................................ 17

CHAPTER 3  GUI IMPLEMENTATION OF RADIATION DETECTION DATA .............. 20
  3.1 Programming environment .............................................................................................. 20
    3.1.1 Windows Mobile Device ......................................................................................... 21
    3.1.2 Visual Basic.NET ...................................................................................................... 21
    3.1.3 .NET Compact Framework 2.0 ............................................................................... 22
    3.1.4 Windows Mobile Software Development Kit (SDK) ....................................... 22
    3.1.5 Windows Mobile device application development ........................................... 23
  3.2 General software system structure .................................................................................. 23
  3.3 Core SPRT algorithm ..................................................................................................... 25
  3.4 Class SPRT implementation ......................................................................................... 28
    3.4.1 SPRT algorithm output ............................................................................................. 29
  3.5 GUI implementation ....................................................................................................... 30
    3.5.1 Parameter entry ......................................................................................................... 32
      3.5.1.1 GC and NC tabs ........................................................................................ 32
      3.5.1.2 Settings tab ................................................................................................... 33
      3.5.1.3 Background tab ......................................................................................... 33
3.5.1.4 Output tab ................................................................. 34
3.5.1.5 Input tab ................................................................. 36
3.5.2 Visual display capabilities ........................................... 36
  3.5.2.1 Data tab .......................................................... 36
  3.5.2.2 Graphs tab ......................................................... 38
3.5.3 Actions .................................................................... 39
  3.5.3.1 Alarms ............................................................. 40
    3.5.3.1.1 Sounds .................................................. 40
    3.5.3.1.2 Displays ................................................. 40
    3.5.3.1.3 Text-messaging ........................................ 41
  3.5.3.2 Sending out data to a pre-existing network .............. 41
  3.5.3.3 Radiation mapping ............................................ 42
3.6 Multi-threading .......................................................... 44

CHAPTER 4 HARDWARE AND INTEGRATION ..................... 47
  4.1 Ideal detector ............................................................ 48
  4.2 Hardware ................................................................. 48
  4.3 Integration ............................................................... 53
    4.3.1 Authentication .................................................. 53
    4.3.2 Headers .......................................................... 54
    4.3.3 Parsing ............................................................ 57
    4.3.4 Conversion ....................................................... 58
      4.3.4.1 Time string conversion .................................. 58
      4.3.4.2 GPS string conversion .................................. 58
    4.3.5 Movement correction ......................................... 60
  4.4 Timing ........................................................................ 60

CHAPTER 5 REAL-TIME ANALYSIS AND FUTURE WORK .... 62
  5.1 SPRT algorithm response time ...................................... 65
    5.1.1 Other performance metrics .................................. 66
  5.2 Other Implementations ................................................ 67
    5.2.1 VHDL implementation ........................................ 68
      5.2.1.1 Algorithm Flow ......................................... 68
      5.2.1.2 VHDL structure ......................................... 68
  5.3 Future work .............................................................. 73
    5.3.1 Mobile robot development ................................... 74
    5.3.2 Movement algorithm .......................................... 78

CHAPTER 6 CONCLUSION .................................................. 77

REFERENCES ................................................................. 80

VITA .................................................................................. 84
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Presentation of ilap.</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Single-threaded system</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Multi-threaded system</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>The New-GUARD software GUI system block diagram</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>SPRT algorithm logic flowchart</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Creating instances of the SPRT class for different radiation intensity values</td>
<td>29</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Sample array with multiple entries containing instances of the sample class</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Layout of the SPRT graphical user interface</td>
<td>32</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Layout of the Settings tab</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Layout of the Background tab</td>
<td>34</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Layout of the Output tab</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Layout of the Input tab</td>
<td>36</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>Entries on the GUI data grid</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>Sample detailed statistics</td>
<td>38</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>Gamma profile graph</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.13</td>
<td>Gamma alarm profile graph</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.14</td>
<td>New-GUARD output options to pre-existing networks and devices</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3.15</td>
<td>Alarm level radiation field mapping</td>
<td>43</td>
</tr>
<tr>
<td>Figure 3.16</td>
<td>New-GUARD system threading diagram</td>
<td>46</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Hardware and software integration block diagram</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Block diagram of hardware system tied together by a microcontroller</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>PCB of New-GUARD hardware unit</td>
<td>52</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Hardware unit inside enclosure with attached buttons and LCD</td>
<td>52</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>New-GUARD system implementation via header data</td>
<td>55</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Original Las Vegas data set mapped out via Google maps</td>
<td>57</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Integration thread functionality and thread diagram</td>
<td>61</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Windows mobile device, HP iPAQ hw6515, used for testing and integration of the New-GUARD system</td>
<td>62</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Profile of the radiation dataset used for testing</td>
<td>64</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>State machine of the VHDL SPRT algorithm</td>
<td>70</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Table simulation of the VHDL SPRT algorithm</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Waveform simulation of the VHDL SPRT algorithm</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>New-GUARD system attached to a mobile unit</td>
<td>74</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>Tabulated data example</td>
<td>34</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Debugged data example</td>
<td>35</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Example header data from a perimeter in Las Vegas, Nevada</td>
<td>56</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Response time 1 using specific Imin/Imax settings</td>
<td>65</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Response time 2 using specific Imin/Imax settings</td>
<td>66</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>New-GUARD system response times for key components</td>
<td>67</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my advisor, Dr. Shahram Latifi, for his encouragement and guidance throughout the process of my graduate work and preparation of my thesis.

I would like to express my warm thanks to Dr. Yuan for giving me the opportunity to work on the SPRT radiation detection project at both the university and the Nellis Air Force Base Remote Sensing Lab. His ideals and concepts have provided good basis for completing my thesis.

I wish to thank Dr. Selvaraj for his guidance in engineering concepts Dr. Venkatesan Muthukumar, for his essential help in classes as well as research projects over the years and Dr. Laxmi Gewali for being involved in my thesis completion.

During this work I have collaborated with many faculty members and colleagues for whom I have great regard, and I wish to extend my sincere thanks to all those who have helped me with my work in the Electrical and Computer Engineering Department of University of Nevada, Las Vegas.

I owe my loving thanks to my family and friends for sticking with me all these years while I finished all my schooling. Without them, I would not have had the proper support in finishing my education.
CHAPTER 1

INTRODUCTION

The increase in security threats around the world has created a greater emphasis for radiation detection studies. Currently, many departments and national laboratories have dedicated a great deal of time and effort to develop new detection instruments, which enable quick detection of radiation sources and at the same time decrease human exposure to radiation [1]. The primary concern in the radiation field has been the safety of the field surveyor. These technicians are exposed to a significant amount of risk when taking the radiation measurements. Considerable research has been conducted to understand the hazards of radiation exposure to human health [2]. Occupational exposure to radiation is governed by the concept of maximum permissible dose (MPD) with the understanding that all exposures should be kept as low as reasonably achievable (ALARA) and in accordance with the International Commission on Radiological Protection (ICRP) and the U.S. National Council on Radiation Protection and Measurements (NCRP) [3].

A new generation detector unit incorporated with an efficient detection algorithm implementation will produce an excellent device to reduce radiation detection hazards. Deployment of hand-held radiation detectors has been comprehensively studied [4], and device features to help and protect a field surveyor when taking measurements are developed in many national research laboratories [5]. These features are essential in
giving the technician more control and flexibility over the measuring unit. Moreover, having additional information combined with the technician's experience and knowledge gives them a safer tool to perform radiation detection. There have been many engineering projects implemented using GUI-based systems, instead of a text-based approach [3-5]. A graphical user interface allows for considerably more information to be effectively presented to the technician [6].

In this thesis, the New-GUARD (New Generation User Adaptable Radiation Detector) prototyping is presented to determine its viability. The New-GUARD system is an integration of the SPRT GUI software and the portable hand-held radiation detector hardware.

1.1 Thesis Outline

In this thesis, the New-GUARD system is prototyped. The first part of the thesis presents the topics about the implementation of the SPRT GUI software portion of the New-GUARD system. This is followed by a discussion about the ideal hardware pieces of the system. The latter part describes the integration of the hardware and software components to complete the New-GUARD system. The thesis concludes with a discussion about the testing done to obtain the real-time performance metrics of the New-GUARD system, as well as future additions to achieve efficiency and eliminate human presence entirely from radiation fields.

The thesis is organized into six chapters. Chapter 1 gives the introduction. Chapter 2 introduces the algorithm used and background information. Chapter 3 presents the SPRT GUI implementation. Chapter 4 describes the hardware and integration. Chapter 5 is
comprised of results, analysis, and future additions to the system. Chapter 6 contains the conclusion of the work.
CHAPTER 2

BACKGROUND

Before addressing the implementation of the New-GUARD system, the radiation detection algorithm used in the New-GUARD system is explained. After that, an overview of the current needs for GUI integration with hardware is explained and examples are given. Lastly, multi-threading principles to enhance programming capabilities are discussed.

2.1 Algorithm Background

Many algorithms have been developed for various applications related to radiation detection. A comprehensive survey of the available algorithms has been performed to obtain substantial understanding of the working principle. The Sequential Probability Ratio Test (SPRT) provides a better platform for the development of an algorithm that emphasizes sample sizes and measurement time limitations. SPRT has been proven successful in many applications [7-9] and is a well known optimal hypothesis testing technique that minimizes the expected sample size for a given error probability [10].
2.1.1 Sequential Probability Ratio Test (SPRT)

Wald's SPRT method has the advantage of handling sequential sampling data [11]. Let \( f(x, \theta) \) be the distribution of a random variable \( x \). \( H_0 \) represents the hypothesis that \( \theta = \theta_0 \), and \( H_1 \) the hypothesis that \( \theta = \theta_1 \). Therefore, the distribution of \( x \) is given by:

\[
\begin{align*}
  f(x, \theta_0) & \quad \text{when } H_0 \text{ is true} \\
  f(x, \theta_1) & \quad \text{when } H_1 \text{ is true}
\end{align*}
\]

Successive observations on \( x \) are denoted by \( x_1, x_2, \ldots, x_m \). For any positive integral value \( m \) the probability that a sample \( x_1, x_2, \ldots, x_m \) is obtained is given by:

\[
\begin{align*}
  P_{1m} &= f(x_1, \theta_1) \cdots f(x_m, \theta_1) \quad \text{when } H_1 \text{ is true} \\
  P_{0m} &= f(x_1, \theta_0) \cdots f(x_m, \theta_0) \quad \text{when } H_0 \text{ is true}
\end{align*}
\]

(2.10) \hspace{2cm} (2.11)

The SPRT for testing \( H_0 \) against \( H_1 \) is defined by the following steps:

1. Two positive constants are chosen \( A \) and \( B \) (\( B < A \)).

2. At each stage of the experiment (at the \( m \)th trial for any integral value \( m \)), the probability ratio \( P_{1m}/P_{0m} \) is computed.

3. If

\[
B < \frac{P_{1m}}{P_{0m}} < A
\]

(2.12)

then the experiment is continued by taking an additional observation.

4. If

\[
\frac{P_{1m}}{P_{0m}} \geq A
\]

(2.13)

then the process is terminated with rejection of \( H_0 \) (acceptance of \( H_1 \)).

5. If
\[
\frac{P_{l_m}}{P_{0_m}} \leq B
\]  \hspace{1cm} (2.14)

then the process is terminated with the acceptance of \( H_0 \).

The constants \( A \) and \( B \) are to be determined so that the test will have the prescribed strength \((\alpha, \beta)\). \( A \) and \( B \) satisfies the inequalities:

\[
A \leq \frac{1-\beta}{\alpha} \hspace{1cm} (2.15)
\]

\[
B \geq \frac{\beta}{1-\alpha} \hspace{1cm} (2.16)
\]

For practical computation purposes, the logarithmic ratio of \( \frac{P_{l_m}}{P_{0_m}} \) is used. The reason for this is that the logarithmic ratio represents the sum of the \( m \) terms, i.e.:

\[
\log \frac{P_{l_m}}{P_{0_m}} = \log \frac{f(x_1, \theta_1)}{f(x_1, \theta_0)} + \ldots + \log \frac{f(x_m, \theta_1)}{f(x_m, \theta_0)} \hspace{1cm} (2.17)
\]

2.2 SPRT for radiation detection

Since Wald’s SPRT method has the advantage of handling sequential sampling data [11], experiments were run to prove it be a solid solution to sequential radiation detection [12-13])

In the case of radiation detection, data is collected via a type of Geiger counter unit, as in radiological survey situations [14]. The counter unit will attain a value such as a gamma, neutron, alpha, or beta count. Each sample will have two possibilities associated with it:

1. Null hypothesis \((H_0)\): denoted as a sample coming from a low radiation background.
2. Alternative Hypothesis ($H_1$): denoted as a sample coming from a high radiation source plus the background region.

The idea is that if the alternative hypothesis ($H_1$) is accepted, then there is pertinent radiation in the field. A radiation alarm should then be calculated, alerts should go off, and the appropriate figures should be notified of the situation. On the other hand, if the null hypothesis ($H_0$) is accepted, it is safe to move on to the next sample in the field.

For each sample, a decision making process is employed based on SPRT. The three options are as follows:

1. Accept Null Hypothesis and define the sample as being from a low radiation background.
2. Accept Alternative Hypothesis and define the sample as being from a higher radiation source plus background.
3. Request one more measurement $x_i$ be made over the same sample, which would be from the same location.

Type I and type II errors are associated with the decisions. They are defined as:

$$\alpha = P(H_1 \mid H_0) \quad (2.18)$$

$$\beta = P(H_0 \mid H_1) \quad (2.19)$$

These are defined as test strengths. The concept is that false positive means that $H_1$ is accepted when actually $H_0$ is true where as false negative is defined when $H_0$ is accepted when actually $H_1$ is true. Next, the probability density functions (PDF) which correspond to $H_0$ and $H_1$ are: $f_0(x)$ and $f_1(x)$, respectively. Note that all sample measurements are independent and the following are the measurements of the same location or sample that have been made so far:
Therefore, the Logarithmic Likelihood Ratio (LLR) is formed by:

\[ \Lambda_n = LLR(x_1, x_2, ..., x_n) = \log \frac{\prod_{i=1}^{n} f_1(x_i)}{\prod_{i=1}^{n} f_0(x_i)} = \sum_{i=1}^{n} \log \frac{f_1(x_i)}{f_0(x_i)} \]  

(2.20)

Conditions are set to determine which decision is chosen. They are as follows:

1. If \( \Lambda_n > B \) accept \( H_1 \)
2. If \( \Lambda_n < A \) accept \( H_0 \)
3. If \( A \leq \Lambda_n \leq B \) request additional sample

Where \( A \) and \( B \) are constants defined as:

\[ A = \log \frac{\beta}{1 - \alpha} \]  

(2.21)

\[ B = \log \frac{1 - \beta}{\alpha} \]  

(2.22)

The SPRT algorithm for radiation detection generates a decision based on a certain number of data points of each sample. The data points, denoted as \( \{x_1, x_2, x_3, ..., x_n\} \), correspond to a specific time that the samples were collected. There are two types of schemes based on which direction to move within the data collected: the FIFO (First-in/First-out) and LIFELO (Last-in/First-elected/Last-out) schemes. Under the FIFO scheme, the oldest data point in the system is first chosen for subsequent analysis using SPRT. Once the analysis is completed or the decision is made, the oldest data point is dropped out of the system and a new sample is drawn. The LIFELO scheme uses the most recent data point from a given spatial location in the sequence for starting the subsequent sequential analysis. If needed, the next youngest data point is chosen to join the analysis. This process is continued until a conclusion is made about the most recent
location or data point. The LIFELO scheme is used for real-time sequential analysis of continuous spatial survey data. It offers reduced response time and spatial errors for decision-making for time-and-spatial critical environmental contamination surveys. The FIFO is therefore not used as much because of its delayed response. At the time a new sample is drawn, the analysis either cannot tell immediately about the sample, or makes an incorrect statement about the sample.

Once the direction is determined, the LLR is computed and the decision process begins. It works on the idea of iterations, where parameters are set to define the minimum and maximum number of data points required to make a conclusion about a specific location, called sample size. There is both a minimum and maximum sample size, denoted by \( \text{imin}/\text{imax} \), for radiation and background regions. These parameters are set as the following:

- \( \text{imin}/\text{imax} \): smaller values for source regions to keep technician safe from large exposure times.
- \( \text{imin}/\text{imax} \): higher values for background regions, as being in a background region is not harmful to a technician.

Once the \( \text{imin}/\text{imax} \) values are set, the LLR calculation follows. The first step is to calculate the background mean and standard deviation. Next, compute the same for the first data point in the sample. Then keep acquiring data points until the chosen minimum number of data points are acquired. After this, compute the LLR and compare to the \( A \) and \( B \) values. The criteria for a decision are as follows:

- If \( \text{LLR value} < A \), then sample is from the background \( (H_0) \) region.
• If $LLR$ value $> B$, then sample is from the source ($H_i$) region and alarm value can then be calculated.

• If $A \leq LLR$ value $\leq B$, then between background and source values. More processing is required.

In the case that the $LLR$ value is between the $A$ and $B$ values, more data points are required to make a decision. If the maximum number of samples is reached with the $LLR$ value still being in between $A$ and $B$, a decision is made based on the following formulas:

$$\begin{align*}
\text{If } \Lambda_{\text{max}} &\leq \frac{A+B}{2} \text{ accept } H_0 \\
\text{If } \Lambda_{\text{max}} &> \frac{A+B}{2} \text{ accept } H_1
\end{align*}$$

2.2.1 Hypothesis settings

The hypothesis settings can be changed in the SPRT algorithm. The two distributions associated with the $H_0$ and $H_i$ decisions are the Normal $N(\mu, \sigma^2)$ distribution and the Poisson $p(\mu) = N(\mu, \mu)$ distribution. There are four available combinations that can be used in the SPRT algorithm. They are described as follows:

1. NN: $H_0$ and $H_i$ both using normal distribution ($\mu_i > \mu_0$)
   - This combination is usually used for a low background region with a low source region. For example, this choice would be good using Neutron counts for nuclear waste surveys.

2. NP: $H_0$ uses the normal distribution while $H_i$ uses the Poisson distribution ($\mu_i > \mu_0$)
- This combination is usually used for a low background region and a high source. For example, this would suffice for using Neutron counts for Special Nuclear Material (SNM) surveys.

3. PN: $H_0$ uses the Poisson distribution while $H_1$ uses the normal distribution ($\mu_1 > \mu_0$)

- This combination is very rare and never comes up. The choice only used for experimenting purposes.

4. PP: $H_0$ and $H_1$ both using the Poisson distribution ($\mu_1 > \mu_0$)

- This combination is usually used for high background and very high source regions. For example, looking for gamma counts in radiation field surveys is choice for this combination.

2.2.2 Alarm computation

The last step in the SPRT algorithm is alarm computation. The alarm computation will only occur if the LLR is calculated and $H_1$ is accepted, as it will be zero otherwise. The Linear Departing Coefficient (LDC) is used:

$$LDC = \left[ \frac{\mu_1 - \mu_0}{\sigma_0} \right]$$  \hspace{1cm} (2.23)

where $\rho (0 < \rho \leq 1)$ is a proportional constant, or scaling factor rho. If the constant equals 1, the LDC indicates the number of standard deviations the source region is from the mean of the background region. Since the LDC alarm level is significantly bigger than the desired alarm level, the Log Departing Coefficient (LGC) is used instead:

$$LGC = \left[ \log_b(Max[1, \frac{\mu_1 - \mu_0}{\rho \sigma_0}]) \right]$$  \hspace{1cm} (2.24)
where $\rho (0 < \rho \leq 1)$ is a proportional constant, or scaling factor rho. Base $e$ and 2 are common bases to use for radiation alarm computation in the SPRT algorithm.

2.2.3 Dynamic background updating

The SPRT algorithm has the option for dynamic background updating. Dynamic background updating is used to adjust the reference background level via an automated process. The other method employs a fixed, or static, background value ($\mu_0$) to compute the final alarm level value. The background statistics (mean, standard deviation, and number of samples) is determined by taking a number of samples at a known background location. This process starts before the radiation detection survey occurs. The preliminary background calculation uses the following prediction formula for the background mean:

$$\mu_k = \left( \frac{k-1}{k} \right) \mu_{k-1} + \frac{SPC}{k} \tag{2.25}$$

where $k$ is the number of background samples taken, $SPC$ is the intensity of the last sample being measured and $\mu_{k-1}$ is the previous sample's background mean. This process will continue until the technician has finished calculation of the background for a particular location. A typical background calculation will last from 1 minute to 5 minutes, using between 60 samples and 300 samples (sampling at 1 sample per second). The resulting background mean value is denoted as ($\mu_0$) and then the technician can start the radiation detection process. In order to make this process more flexible, the dynamic background updating is employed.

The reason for adding this flexibility is to allow a technician to have the background updating process be an automated procedure. Therefore, according to the region
surveyed, the background will be updated automatically. This automated procedure will happen when three conditions occur:

1. When $A \leq LLR\ value \leq B$

2. The maximum number of samples for a decision is reached. This means that when the I value (or data point number) has reached its $i_{\text{max}}$ value.

3. When $LLR\ value < \left( \frac{A + B}{2} \right)$ and the alarm value equals 0

When this occurs, the background value will be updated using equation 2.25 and the background mean will be set to the new background mean ($\mu_0 = \mu_k$). Since this process is a sensitive procedure, cares are taken to limit the mean, hence eliminating the error of background becoming near source level. Therefore, a threshold is applied to not allow the new background value to reach a source value. The threshold value, denoted as $\mu_{\text{thresh}}$, can either be set by a technician or calculated. The condition is as follows:

$$\mu_{\text{thresh}} > \mu_k$$

(2.26)

If the newly calculated mean is larger than the threshold calculated or set, then the original mean $\mu_0$ becomes the background mean again. If not, the newly calculated background mean $\mu_k$ will be used for further computations.

### 2.2.4 Maximum Likelihood Estimator (MLE)

A New Maximum Likelihood Estimator (MLE) for Normal Approximation of Poisson Distribution $N(\mu, \mu)$ is also employed in the SPRT algorithm. The MLE is defined as:

$$\mu_{\text{MLE}} = \sqrt{T_n + \frac{1}{4} - \frac{1}{2}}$$

(2.27)
where

\[ T_n = \frac{1}{n} \sum_{i=1}^{n} X_i^2 \]  \hspace{1cm} (2.28)

If the Maximum Likelihood Estimator is turned on, it will be employed wherever a Poisson distribution is used. As a direct result, the MLE will be executed in the NP, PN, and PP cases.

2.2.5 Ilap

An additional parameter is used in the original SPRT algorithm, called ilap. Ilap is only used in FIFO algorithm and is used as an overlapping factor, which treats the sampling sequence as a continuous sequence. Figure 2.1 shows correct simulation results of the ilap parameter.

As shown in Figure 2.1, if ilap is negative, the distance between the start of one sample and next is the absolute value of ilap. If ilap is positive, the distance between the start of one sample and next is \( n - \text{ilap} \), where \( n \) is denoted as the data points of each sample.
2.3 GUI and hardware integration

Contemporary times require for a transition away from text-based software. New graphical user interfaces are required to coincide with new integration between hardware and software systems.

Text-based software is very difficult to understand, especially when there are complicated procedures to follow. Relying solely on the written word provides the most opportunities for misunderstanding by an operator or technician. The developer must rely exclusively on written word to disseminate work instructions for the operator to execute. The only way to battle this is by having extensive training and documentation. However, with text-based software it is difficult to describe what an operator should look for without a diagram for reference [6].

A transition between text-based software and a graphical user interface is needed. The first step is to get rid of paper documentation and include it electronically. Embedded training techniques, like multimedia tutorials, should follow. The GUI should be useable as soon as delivered, which eliminates the need for extensive training and paper documentation.

The idea is to reduce misunderstandings and inconsistent perceptions by the operator, while executing instructions using current multimedia capabilities. The use of multimedia objects reduces errors. If a picture paints a thousand words, the canvas created with a video or continuous sequence of multimedia objects could be unbelievable [6]. The graphical user interface should be useable and independent, should use off-the-shelf technology, and should use a common application interface style.
A GUI has a major advantage in that considerably more information can be effectively presented to the user. Menus and status can be made visually appealing, and therefore more effective. Using a mouse or pointer device to initiate responses is more effective than having the user transfer attention between the screen and keyboard.

More specifically, shapes can be used to describe system components and colors used to convey the state of those components. For example, alarm conditions become apparent when flashing a particular component different colors and textual information about a problem can be presented by pop-up windows [15]. Visually, this can make it much simpler for the user to interpret information and algorithm output.

Studies have been performed on porting pre-existing engineering solutions into GUI systems. A GUI model for the selection and design of stress-grading systems of high voltage and radiation proved to improve speed, ease of use, productivity, and range of use in the design and analysis of high voltage systems [16]. A friendly interface and a highly efficient graphical and analytical environment for real-time electromagnetic transient simulation was developed to increase productivity by making the process of assembling circuits, monitoring simulations, and analyzing results quick and easy due to fully providing a graphical and analytical environment for the system. In addition, the system was made flexible by allowing it to be ported to different operating systems [17]. Robots are assembled and deployed easily by a newly developed GUI system [18]. A GUI, written in MATLAB, is used to control the BS2 microcontroller and its functionalities [19]. Lastly, SCADA/EMS graphical user interfaces are developed so that visual presentation of information to the operators and the mode of operation of the various functions are increased. Advanced functions to facilitate SCADA/EMS operations are
made easily available via the GUI [20]. These engineering developments are made easier using GUI systems and they all comprise of integration between hardware and software. This is essential in contemporary times.

2.4 Multi-threading

A thread is an independent flow of control within a process, composed of a context (which includes a register set and a program counter) and a sequence of instructions to execute [21]. A thread is essentially a code sequence that runs independently. This permits a program to work on multiple tasks in a parallel manner [23]. In a single-threaded system, there is only one flow of control (one thread) through the program instructions. Therefore, only one instruction at a time can be seen and executed. Figure 2.2 illustrates this procedure.

![Figure 2.2. Single-threaded system](image)

If tasks can be executed independently of each other, the use of multiple threads of control is possible. As a result, a multithreaded program has two or more flows of control (threads) [21]. Concurrent thread execution (or concurrency) means that two or more threads are in progress at the same time. By doing this, significant performance gains are achieved. For example, slow applications, such as file I/O, can be performed on
a separate thread while the main thread continues execution [23]. A program with threads is called asynchronous programming. Figure 2.2 illustrates this procedure:

![Multi-threaded system diagram](image)

Figure 2.2. Multi-threaded system

There are many benefits of threads in application developments. These include increased throughput, better response time, conservation of system resources, faster operations, and a natural programming structure [21]. The idea is that performance benefits are realized through concurrent and/or parallel thread execution. The actual speedup over using a single-threaded program is calculated as follows [21]:

\[
\text{Speedup} = \frac{1}{(1 - M)} + \frac{M}{N}
\]  

(2.29)

where

- \(N\) = number of processors used in the system.
- \(M\) = percentage of the total execution time in parallel

The point is to be allowed to do several tasks at one time in a program. Using multiple threads within an application allows you to separate out the tasks that are
performed by the user interface (UI) from the tasks that are carried out in the background. Organizing tasks in this manner allows the UI to remain responsive to user input while the background processing takes care of the data processing [24]. The .NET platform allows the options for asynchronous programming. Any method created can be called asynchronously. It allows this by a processed referred to as synchronous delegate invocation, as a delegate is used to allocate a worker thread and call a method to execute on the thread [21].

In addition, threads can be invoked in the .NET compact framework for multithreaded mobile applications using the System.Threading namespace. The important highlights are as follows [25]:

- **Background Threads**: A thread may now be made background thread by setting the IsBackground property to true.
- **Aborting Threads**: Calling the newly included Abort method results in an exception being raised and the thread terminating normally.
- **Thread Coordination**: A thread can now wait for another thread to complete by calling the latter's Join method.
- **Thread naming**: Threads can be given a name (only once).

In addition, the compact framework supports asynchronous delegate support as discussed before.
CHAPTER 3

GUI IMPLEMENTATION OF RADIATION DETECTION DATA

This chapter presents the implementation of an efficient graphical user interface for radiation detection purposes.

The organization of this chapter is as follows: Section 3.1 describes the programming environment; Section 3.2 shows the general structure of the software system; Section 3.3 lays out the SPRT algorithm used in the software system; Section 3.4 studies the class SPRT implementation; Section 3.5 presents the GUI implementation, and Section 3.6 views the multi-threading functionalities.

3.1 Programming environment

The core of the New-GUARD system is developed using Visual Basic .NET., the Windows Mobile Software development kit (SDK), and the .NET Compact Framework 2.0. Visual Basic.NET is an object-oriented programming included in Visual Studio 2005. The Windows Mobile SDK allows for developing applications on all Windows Mobile devices (cell phones, PDAs, etc). Lastly, the .NET compact framework is used, via managed code, as an environment for running the SDK developed programs on the Windows Mobile devices.
3.1.1 Windows Mobile Device

A Windows Mobile Device is a handheld device powered by the Windows Mobile platform. The Windows Mobile platform is a rich developer platform that allows for the building of third-party software directly onto the device [26]. Device-specific actions using the windows mobile platform are making phone calls, retrieving email, scheduling, internet actions, text-messaging, Microsoft Office integration, and other processing capabilities.

The Windows Mobile platform is available on a variety of different devices from a variety of different operators. Around 44 OEM providers have devices with the Windows Mobile software on it such as Motorola, Palm, Dell, HP, Palm, and I-mate. Windows Mobile devices are offered on both GSM and CDMA networks [27].

3.1.2 Visual Basic .NET

Visual Basic .NET provides the easiest, most productive language and tool for rapidly building Windows applications. Visual Basic .NET comes with enhanced visual designers, increased application performance, and a powerful integrated development environment (IDE). It also supports creation of applications for wireless, Internet-enabled hand-held devices [28]. Its features include the ability to create powerful windows and web based applications, full object-oriented constructs, simplified deployment options, powerful data access mediums, improved coding, direct platform access, COM interoperability, code reusability, upgrade options, and mobile applications development. These features make Visual Basic.NET a solid platform to develop a graphical user interface in.
3.1.3 .NET Compact Framework 2.0

The .NET Compact Framework is a hardware-independent environment for running programs on resource-constrained computing devices such as personal digital assistants (PDAs), mobile phones, and set-top boxes. It runs on top of the Microsoft Windows CE operating system, which allows applications that are universal among all devices running Windows CE based operating systems [29].

The .NET Compact Framework is available for Visual Basic .NET. Visual Basic .NET includes all the capabilities that allow writing and debugging a device application (called Smart Device Extensions). Tools and techniques are available for developing normal application when writing device applications too. The .NET Compact Framework is a subset of the .NET Framework class library and also contains classes exclusively design for it. Instead of using the .NET Framework class libraries, the Compact Framework specific libraries are used [30]. It inherits the full .NET framework architecture of the common language runtime and managed code execution [31].

3.1.4 Windows Mobile Software Development Kit (SDK)

The Windows Mobile SDK is used to develop applications for Windows Mobile devices using managed code with the .NET Compact Framework 2.0 installed on them. The SDK allows a development environment for building, testing and deploying applications for the Microsoft Windows Mobile platform [26]. By using the Windows Mobile software platform, innovative applications for mobile devices are able to be developed. The platform offers features such as data connectivity, rich API support such as Bluetooth and the Pocket Outlook Object Model (POOM), an extensive range of
programming models, and device resources such as multithreading. This allows for fast
time-to-market or rapid application development [26].

3.1.5 Windows Mobile device application development

Developing a Windows Mobile application and deploying it onto a device requires
some important steps. The general procedure is as follows:

1. Install a copy of Visual Studio 2005 (VB .NET)
2. Install the Windows Mobile SDK on system
3. Install the .NET Compact Framework 2.0 on the target device
4. Develop application on system and deploy application on target device

3.2 General software system structure

The New-GUARD software system is written in Visual Basic .NET while the
Windows Mobile SDK is used to port the Visual Basic .NET code over to a mobile
device, like a PDA or windows mobile cell phone. The Compact Framework 2.0 is
installed on the target device to allow the code to run on the Windows Mobile platform.
The SPRT algorithm is implemented as a separate entity that communicates with the
Visual Basic .NET graphical user interface. The SPRT portion receives parameter data
from the GUI, and then processes/sends back the results to the GUI. In turn, the GUI
takes the statistical output from the SPRT portion and outputs it via tables, graphs, and
other built-in GUI functions. This process is shown by Figure 3.1.
The technician has the ability to enter the SPRT algorithm parameter data by interacting with the GUI. As a result, the technician has power to determine how the radiation detection procedure performs. For a novice technician, there are optimal default settings set up for startup of the program. Saved setting files can also be imported based on known conditions.

The input to the GUI system is simple, a gamma neutron or other radiation intensity value. The GUI takes the intensity value and, based on the technician entered parameters, generates a SPRT output. The output is then used by the GUI to generate an action to the technician. The outputs, or actions, of the GUI include displaying the SPRT data in an easily readable fashion to the technician, alerting the technician if an alarm exists, and
other functions that allow the technician to understand the characteristics of the radiation data.

Currently, the mediums for input to the GUI portion include text file, serial, or Bluetooth data. All the data comes in one line at a time at any specified interval. The default interval is one sample per second.

The GUI can also output the SPRT results to another device or to a pre-existing GSM network. It can output to one other device via serial port or multiple devices using Bluetooth. In addition, multiple devices can send SPRT data using a GSM network by SMS messaging for further actions or processing of data.

3.3 Core SPRT algorithm

The core algorithm currently used in the SPRT GUI system is the SPRT radiation detection algorithm. The algorithm can be changed at anytime for future flexibility. There are two versions of the SPRT implemented in the GUI system, the FIFO (First-in/First-out) and LIFELO (Last-in/First-elected/Last-out) schemes. Under the FIFO scheme, the oldest data point in the system is first chosen for subsequent analysis, such as for computing population mean and sequential probability ratio test (SPRT). Once the analysis is completed or the decision is made, the oldest data point is dropped out of the system and a new sample is drawn. On the other hand, under the LIFELO scheme, the most recent data point from a given spatial location in the sequence is first selected for starting the subsequent sequential analysis. If needed, the next youngest data point is chosen to join the analysis. This process is continued until a conclusion is made about
the most recent location or data point. Figure 3.2 describes the logic via a flowchart of the implemented SPRT algorithm.

Figure 3.2. SPRT algorithm logic flowchart using the FIFO and LIFELO schemes.

The solid line represents FIFO scheme while dashed line represents the LIFELO scheme.
The basic algorithm is as follows:

1. Compute background: calculate mean and standard deviation

2. Choose hypothesis settings (NN, NP, PN, PP)

3. Attain 11 samples and store them inside a buffer

4. Grab new data, store it at beginning of buffer and compute mean and standard deviation
   a. If FIFO algorithm, start from the end of the buffer (oldest data point) and traverse towards the beginning of the buffer (newest data point)
   b. If LIFELO algorithm, start from the beginning of the buffer (newest data point) and travel towards the beginning of the buffer (oldest data point)

5. Choose imin/imax values
   - If data leans toward source, set imin/imax to the source imin/imax values
   - If data leans toward background, set imin/imax to the background imin/imax values

6. Compute LLR or take another data point
   - If the number of data points taken is larger than the minimum number of samples, compute LLR
   - If the number of data points taken is smaller than the minimum number of samples, go to step 4

7. Check LLR value
   - If LLR < A, Alarm = 0
   - If LLR > B, perform Alarm calculation
   - If A ≤ LLR ≤ B and number of data points taken = imax, compute alarm
• If \( A \leq \text{LLR} \leq B \) and number of data points taken < \( \text{imax} \), go to step 4

8. Push out oldest data point and wait for next sample. Go to step 4 if new sample arrives.

### 3.4 Class SPRT implementation

The SPRT class is the structure, or backbone, of the SPRT portion of the GUI. All the parameters are stored inside the class. The idea is that once an instance of the SPRT class is created, it will have its own set of SPRT parameters that can be changed. This is an object-oriented approach that allows for multiple parameter sets for different radiation intensity types (alpha, beta, gamma, neutron, etc). The class contains SPRT operations that run based on the selected SPRT parameters. These operations run the SPRT algorithm on the parameter set and return a resulting output. The output contains all the SPRT alarm data for further processing by the GUI portion of the system.

This idea is shown in Figure 3.3.
Figure 3.3. Creating instances of the SPRT class for different radiation intensity values

The parameters held within the SPRT class include: imin/imax values, alpha values, beta values, rho values, base values, background settings, source settings, dynamic background and ceiling settings, file, Bluetooth, and serial port settings, algorithm type settings, mode settings, MLE options, and work path and file logging parameters.

The operations available within the SPRT class include: a class constructor, fill_background, fill_source, swap_buffer, shift_buffer, SPRT_calculate_sample, and calculate background. The class constructor sets all the default values for an instance of an SPRT class, fill_background fills a buffer with background data, fill_source fills a buffer with source data, swap_buffer swaps the data inside the buffer, SPRT_calculate_sample calculates the SPRT output of a specific sample, and calculate background calculates the background information of a region of interest.

3.4.1 SPRT algorithm output

The main SPRT class takes incoming data and runs the SPRT algorithm based on the parameters set inside a specific instance of the class. The output is stored within this
instance in a class specifically for individual samples. Each SPRT output corresponds to one sample and is stored in its own entry in a sample array. As each sample comes in, a new sample and its corresponding SPRT output is dynamically allocated to a new instance of the sample class. This is demonstrated in Figure 3.4.

![Figure 3.4. Sample array with multiple entries containing instances of the sample class](image)

Each sample includes the following SPRT output results: radiation intensity value, number of samples for decision, sample mean value, sample mean square value, sample standard deviation value, sample LLR value, sample imin/imax value, sample time stamp value, sample GPS string value, sample compass string value, sample accelerometer value, sample longitude/latitude values, and sample generated alarm value. This resulting output is then taken by the GUI portion of the system and processed accordingly.

### 3.5 GUI implementation

The GUI portion of the New-GUARD system is used to make it easier for a technician to read and interpret radiation data/output from the SPRT algorithm. The
SPRT algorithm receives the technician inputted settings from the GUI and outputs the radiation statistics. The output is then communicated over to the GUI portion for further processing. GUI processing functions include displaying the radiation output via graphs and tables, alarming the technician using sounds, vibrations, visual displays, and text messaging support, and other important functions like text message (SMS) data post processing and real-time radiation field mapping using Google Maps.

The SPRT GUI is defined as its own class. It is a graphical front end that contains interface controls such as VB.NET forms, labels, text boxes, list boxes, data grids, buttons, and graphs. The GUI is organized based on layers of tabs. Tabs are ideal for a Windows Mobile device because their screens are smaller, so an abundance of information needs to be spread out. For expandability purposes, tabs can be added on the fly for new tasks for expandability purposes. A hierarchy of tab controls is therefore used. The VB.NET controls are placed amongst specific tabs to make it easier for a technician to control the functionalities of the New-GUARD system.

The GUI piece is split up into different categories containing their own tabs that indicate a different task: SPRT parameter entry, visual display capabilities, and actions. The parameter entry tab sections comprise of settings the technician can change that interact with the SPRT algorithm, the visual display tabs show the resulting SPRT statistical output in an easily readable form for the technician, and the actions take care of functionalities that occur based on the calculated SPRT algorithm output. Figure 3.5 shows the layout of the SPRT graphical user interface, generated by the Windows Mobile 6 Software Development Kit (SDK).
Figure 3.5. Layout of the SPRT graphical user interface

3.5.1 Parameter Entry

The parameter entry section contains all tabs that allow the technician to input or alter SPRT core algorithm settings in an easy fashion. In addition, input and output settings are allowed to be entered here.

3.5.1.1 GC and NC tabs

The GC and NC tabs have identical information stored in them. The only difference is that one takes care of gamma count settings and the other neutron count settings. A
SPRT class is instantiated for each of these classes and routed to the corresponding tab inputs.

3.5.1.2 Settings tab

Under the settings tab, SPRT settings are inputted or altered. Settings such as the SPRT hypothesis, confidence, scaling, imin/imax, and background/source mean and standard deviation are changed here. Once entered these settings are tied to their corresponding SPRT class. The SPRT class, or core algorithm, then runs based on these settings. Figure 3.6 shows the layout of the settings tab.

![Hypothesis Settings](image)

**Figure 3.6. Layout of the Settings tab**

3.5.1.3 Background tab

The background tab takes care of general background algorithms settings. Settings such as algorithm type (FIFO or LIFELO), mode (regular or debug), MLE, and dynamic background settings can be altered here by a technician. Figure 3.7 shows the layout of the background tab.
Algorithm Settings

- FIFO
- LIFELO

[Image: Figure 3.7. Layout of the Background tab]

3.5.1.4 Output tab

The output tab takes care of where the resulting output is placed after SPRT calculations. Currently, the SPRT data can be outputted to a text file, serial port, or Bluetooth interface.

There are two types of output logging, standard tabulation and detailed debug mode. Standard tabulation is used for easy integration of SPRT data into a database while the debug mode is used to analyze, in great detail, the trends of data. Table 3.1 shows a tabulated data example while Table 3.2 shows a debug data example.

Table 3.1. Tabulated data example

<table>
<thead>
<tr>
<th>K</th>
<th>GC</th>
<th>XN</th>
<th>SIG</th>
<th>LLR</th>
<th>AL</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>250</td>
<td>290.333</td>
<td>14.0665</td>
<td>8.54282</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>188</td>
<td>348</td>
<td>363.667</td>
<td>33.3217</td>
<td>5.83022</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>189</td>
<td>299</td>
<td>370.33</td>
<td>23.029</td>
<td>6.97798</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 3.2. Debugged data example

<table>
<thead>
<tr>
<th>Sample #</th>
<th>1</th>
<th>( \mu ) = 294.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>i: 1</td>
<td></td>
<td>( x_{in} = 307.000000 )</td>
</tr>
<tr>
<td>i: 2</td>
<td></td>
<td>( x_{in} = 269.000000 )</td>
</tr>
<tr>
<td>i: 3</td>
<td></td>
<td>( x_{in} = 298.000000 )</td>
</tr>
<tr>
<td>i: 4</td>
<td></td>
<td>( x_{in} = 278.000000 )</td>
</tr>
<tr>
<td>i: 5</td>
<td></td>
<td>( x_{in} = 296.000000 )</td>
</tr>
</tbody>
</table>

\( \text{alarm: 0.000000} \) \( H_{r} = 7.624578 \)

Note that not only will the SPRT GUI log the SPRT statistics of the data coming in, but it will also log all background data. Figure 3.8 shows the layout of the output tab on the GUI.

**File Settings**

- **Output File**
  - dataout.txt

- **Log File**
  - logfile.dat

---

**File Settings**

<table>
<thead>
<tr>
<th>File</th>
<th>Serial</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Background</td>
<td>Output</td>
</tr>
</tbody>
</table>

| GC | NC | Input | Background | Comm |

---

Figure 3.8. Layout of the Output tab
3.5.1.5 Input tab

The main input tab deals with where the data (samples) are coming from. Currently, data can come through file, serial, or Bluetooth mediums. Each input medium has specific settings that can be changed. In the future, there will be a way to generate random data as one of the input choices. Figure 3.9 shows the layout of the Input tab.

**General I/P Settings**

- File
- Serial
- Random

Figure 3.9. Layout of the Input tab

3.5.2 Visual display capabilities

The SPRT core algorithm outputs radiation statistical data. The data is hard to understand in raw text format, so graphical interpretation of the data is important. Therefore, the GUI portion of the New-GUARD system has extensive visual display capabilities using both tables and graphs.

3.5.2.1 Data tab

The data tab contains a grid that stores all incoming sample data in table format. The data stored in the grid is based on the sample array data explained earlier. Each sample
will have SPRT data stored within it and displayed on the data grid. Figure 3.10 is an example of entries on the data grid.

<table>
<thead>
<tr>
<th>K</th>
<th>Value</th>
<th>LLR</th>
<th>Alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>284</td>
<td>7.2704803</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>5.6257749</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>281</td>
<td>5.6342805</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>312</td>
<td>4.5722900</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>279</td>
<td>4.9022955</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>293</td>
<td>5.6503870</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>291</td>
<td>7.6193876</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>291</td>
<td>7.6193876</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.10. Entries on the GUI data grid

Currently, there are three data grids present for the GC and NC tabs: SPRT data, SPRT debug, and SPRT loc. The following is a description of these data grids:

- **SPRT data:** Displays columns for the Kth sample, the GC/NC value, expected alarm value, SPRT calculated alarm value, and time stamp of the data coming in.

- **SPRT debug:** Displays columns for the Kth sample, the GC/NC value, the SPRT LLR calculated value, the SPRT calculated alarm value, and time stamp of the data coming in.

- **SPRT loc:** Displays columns for the Kth sample, the GC/NC value, the GPS string, the Accelerometer string, the Compass string, and the time stamp of the data coming in.
In addition, double clicking entries in the data grid pulls up more detailed information about the sample, which is useful to a technician. The detailed sample data is pulled from the specific index of the sample array. Figure 3.11 illustrates an example of this procedure.

![Sample detailed statistics](image)

Figure 3.11. Sample detailed statistics

3.5.2.2 Graphs tab

The graphs tab contains all graphs for the incoming SPRT data. Currently, two graphs are present for GC and NC data, data profile and alarm profile. The data profile is a connected scatter plot graph of the sample number vs. the GC/NC values while the alarm profile is a graph of the sample number vs. the alarm value. Figure 3.12 and 3.13 illustrate an example of a gamma profile and gamma alarm profile scatter plot graph.
3.5.3 Actions

The SPRT core algorithm calculates incoming data and outputs it to the GUI portion of the New-GUARD system. The SPRT output is analyzed by the GUI and actions are
invoked based on certain characteristics of the output, like alerting the technician using visual, sound, or text-messaging based methods, sending out data to a pre-existing network, or radiation level mapping.

3.5.3.1 Alarms

The SPRT core algorithm generates an alarm level per sample. If the alarm level is high enough, an action is generated by the GUI indicating to the technician the presence of a radiation source. The New-GUARD system uses three types of alarms: sound, display, and text-message notifications.

3.5.3.1.1 Sounds

When a radiation alarm is calculated by the SPRT core algorithm, a sound goes off to alert the technician of nearby danger. The technician can specify which sound to use based on differing radiation types and alarm levels. This is implemented by creating a universal sound class for use with all Windows Mobile devices. In order to use the class, it is instantiated with a filename of the sound needed to be used for the alarm and then executed whenever a sample generates a high enough alarm level. Currently, the New-GUARD system uses two different alarm wavs for GC and NC alarm types. A sound is played if the SPRT core algorithm generates an alarm value of 1 or higher.

3.5.3.1.2 Display

In addition to sounds playing, colors are displayed on the Windows Mobile display based on the value of the generated alarm. Currently, three different colors are used: yellow, orange, and red, where yellow indicates an alarm level of 1, orange indicates an alarm level of 2, and red indicates an alarm level of 3 or higher. The color will be displayed at the time of alarm and set back to normal until another sample is calculated.
3.5.3.1.3 Text-messaging

When an alarm is set off, sounds are played and displays are invoked automatically. Alarm text-messaging is a setting that can be turned on and off for the New-GUARD system. If text-messaging support is turned on, a SMS text message will be sent out using a pre-existing GSM network to any specified phone number(s). The following data is currently sent out via SMS text message when an alarm is generated:

\[[GK, GC, GMU, GSIG, GAL, GPS \text { string}, ACC, COMP]\]

where

- GK is the Kth sample,
- GC is the radiation intensity value coming in,
- GMU is the sample mean,
- GSIG is the sample standard deviation,
- GAL is the sample alarm,
- GT is the sample time stamp,
- GPS is the GPS string,
- ACC is the accelerometer string,
- COMP is the compass string.

3.5.3.2 Sending out data to a pre-existing network

The New-GUARD system includes the functionality to send data to a pre-existing network. Currently, the system allows for two mediums of transport: Bluetooth and SMS text messaging. Any of the data being transmitted throughout the New-GUARD system (SPRT algorithm output, GPS information, etc) can be sent out for further processing at a central location. Figure 3.14 shows this procedure.
This feature is advantageous in the event that the New-GUARD device is destroyed. Measures would still be able to be taken based on the previous data transmitted to a base station or to another device.

3.5.3.3 Radiation mapping

The SPRT GUI can map radiation data based on the algorithms alarm output level and latitude/longitude data. If mapping is turned on, the GUI will map location markers with their corresponding alarm levels in real-time. The SPRT GUI currently has the following alarm marker mapping:

Alarm level 0: blue
Alarm level 1: yellow
Alarm level 2: orange
Alarm level 3 or higher: red
Figure 3.15 shows an illustration of this procedure.

As shown in the figure, a car is driven in a certain perimeter and the SPRT algorithm is run on the data, which is stored with corresponding location information. 1700 data points were used in this real-time test run.

The algorithm for mapping SPRT data is as follows:

1. When a new data point comes in, run SPRT core algorithm on data point to generate an alarm value.
2. Store alarm value and retrieve GPS longitude and latitude data for that point.
3. Create marker on map based on the following criteria:
   a. If alarm level = 0, make marker blue
   b. If alarm level = 1, make marker yellow
c. If alarm level = 2, make marker orange

d. If alarm level is 3 or higher, make marker red

4. Wait for data point and then loop steps 1-3

3.6 Multi-threading

The GUI’s functions were developed to run in parallel. Therefore, independent threads are created for every critical process, such as NC and GC graph/table outputs. This means that the operator can perform other operations, such as save comments, while core functions are running in the background. This is essential in increasing system performance. The following are the core threads invoked in the SPRT GUI application:

- **GC SPRT data:** This thread takes care of all incoming GC data; including running the SPRT algorithm on new data, graphing, and updating data tables. The thread will run at all times until technician stops the execution of the GUI.

- **NC SPRT data:** This thread takes care of all incoming NC data; including running the SPRT algorithm on new data, graphing, and updating data tables. The thread will run at all times until technician stops the execution of the GUI.

- **Alarm:** When an alarm is generated, a thread is created by one of the main data threads. This thread takes care of all the alarm functions; including sending out alarm text messages and generating alarm sounds. When an alarm is produced, this thread will send the text message to a specified number, generate a sound, or do both. The alarm threads are given highest priority.
• **Output logging:** This thread takes care of all output logging; including writing SPRT data to file streams and serial/Bluetooth mediums. The thread will run at all times unless output logging is turned off.

• **Mapping:** This thread takes care of mapping the SPRT alarm data along with location information on a mapping medium, like Google Maps. The thread will run at all times unless the mapping functionality is turned off.

• **Integration:** This thread takes care of preparing new data coming in from different devices for further processing by the main data threads. It will integrate different types of devices with the GUI system. The thread will run at all times until technician stops the execution of the GUI.

Figure 3.16 shows the threading diagram of the New-GUARD system.
Figure 3.16. New-GUARD system threading diagram
CHAPTER 4

HARDWARE AND INTEGRATION

Radiation detection is classified as a hazardous task. The technicians taking the radiation measurement are exposed to the radiation for a long duration of time. Reduction of human intervention while detecting radiation is highly desirable. This is the main drive towards developing a new generation hand held detector which will eventually reduce the need for humans to take measurements. The concept is strikingly simple: use a detector to obtain radiation data, log the information, process, and send it out to a remote location. The information logged and sent out includes radiation data, location information, alarm, and comments. The idea is to implement all features onto a single device with self-correcting capabilities.

This chapter presents integration of an efficient graphical user interface and hardware for radiation detection purposes.

The organization of this chapter is as follows: Section 4.1 describes the concept of an ideal detector, Section 4.2 shows the hardware needed for the detector, Section 4.3 describes the integration process between hardware and software for the system, and Section 4.4 describes timing issues in the system.
4.1 Ideal Detector

The ideal radiation detector is an integrated system comprising of a radiation detector, global positioning system (GPS) unit, self-correcting system (compass and accelerometer), wireless communication, and a GUI interface to tie everything together. The system is an integration between hardware and software components. Figure 4.1 shows this process.

![Figure 4.1. Hardware and software integration block diagram](image)

As shown, the hardware inputs are sent over to an integration function. The integration function takes the hardware inputs and sends it over to the GUI in a readable fashion. The idea is that hardware inputs are converted and sent out in a standardized data format to the GUI portion of the New-GUARD system. In turn, the GUI portion parses, formats, and then processes the data. This is the heart of the New-GUARD system.

4.2 Hardware

The hardware portion of the New-GUARD system has the following components:

i. Radiation detector

The New-GUARD system requires a radiation detector to be integrated with the system. The radiation detector communicates via serial or Bluetooth and will provide the
hardware with a radiation intensity value. The detector can detect all forms of radiation, such as gamma, beta, and alpha radiation.

ii. Global Positioning System (GPS)

Tagging the positional data together with the detection data is important for both on-site processing and post-processing. The latitude and longitude information of an area gives the availability to map the radiation strength of the area relative to the position. This capability provides better visualization for future processing and actions, as the information can be converted into graphical representation.

iii. Accelerometer

An accelerometer enables the technician to measure the acceleration of the New-GUARD system. It measures the acceleration in both the x and y directions. This helps in the correction process.

iv. Compass

A compass enables the technician to locate his/her relative position. Also, the angle reading on the compass helps self-correcting computation process. This helps in the correction process.

v. Bluetooth

The collected data has to be sent in real-time to the central processor for further analysis. This enables other technicians to monitor the detection process remotely. Furthermore, when using robots to replace humans for taking readings, it is essential to get a real-time reading of the sample strength.

vi. LCD screen
The LCD screen is used to view the data being sent to the integration function of the New-GUARD system. It is mainly used for debugging purposes and could be taken out of the system in the future.

vii. Memory

All data that is sent out to the integration piece of the New-GUARD system is saved in memory, as only the critical information is sent elsewhere for further processing. The data saved to the memory could be used for future data analysis if needed. In addition, in case of communication failure, the memory acts as a safeguard. SD memory is used in the New-GUARD system to provide sufficient memory to the device to hold all pertinent information.

The hardware system is tied together using a microcontroller. All devices are connected with the microcontroller and programmed to communicate with each other. Figure 4.2 shows this procedure.
The microcontroller is the core of the hardware system. It is a type of microprocessor emphasizing high integration, low power consumption, self-sufficiency, and cost-effectiveness [33]. It is coded to interact with the hardware and process data accordingly. It is programmed using the C language because using C for a large embedded project reduces the non-recurring engineering costs by a factor of three or more [33]. The hardware unit, assembled by the senior design team Rogelio Esparza and Alexander Harris, is shown in figures 4.3 and 4.4.
Figure 4.3. PCB of New-GUARD hardware unit

Figure 4.4. Hardware unit inside enclosure with attached buttons and LCD
4.3 Integration

The next step after creating the hardware unit and software SPRT piece is integrating the two together. The concept is to obtain location and gamma information from the hardware unit and use it coherently with a Windows Mobile device with the SPRT GUI installed. This is the heart of the New-GUARD radiation detection system.

The integration portion is a developed standard written out based on the output of the hardware components. That standard is written into a header file and sent out via Bluetooth or serial communication to a system ready to read and process it. The New-GUARD system has the functionalities installed in the GUI software to read and process the header information in a meaningful way. The main steps include:

1. **Authentication.** The integration communication requires authorization before it accepts data for security purposes.

2. **Attain header data.** The entire header will be sent and retrieved via serial or Bluetooth communication.

3. **Parsing the information.** The header will be parsed into different data elements based on a comma delimited format.

4. **Conversion.** The data elements will be converted into proper formats for further processing using the SPRT GUI.

5. **Movement correction.** The device will correct itself based on movement.

4.3.1 Authentication

For security purposes, the integration piece requires authorization before it accepts data. Therefore, a form of handshaking takes place between the integration pieces and the hardware unit using a secret encrypted code. Once the code is validated, data transfer
and processing is initiated. There is no need for handshaking for a direct serial connection.

4.3.2 Headers

The hardware device has capabilities of sending out data via serial and Bluetooth interfaces. While both communication mediums will work the same, Bluetooth was chosen and tested with New-GUARD project.

In order for the hardware’s data to be read correctly, a header of its readings are formed. This way, as long as any device knows the header format of the data being communicated from the hardware unit, it can process the data how it pleases. The hardware unit of the New-GUARD system has the following header format:

| ID | GC  | Time | GPS_1 | GPS_2 | GPS_3 | GPS_4 | COMP | ACC_1 | ACC_2 | *
|----|-----|------|-------|-------|-------|-------|------|-------|-------|------

Where

- **ID**: identification number of the device sending out the data
- **GC**: gamma value of the data point being sent out
- **Time**: timestamp of when the data point is sent out
- **GPS_1**: latitude value \((ddm.mmmm)\)
- **GPS_2**: latitude north or south value
- **GPS_3**: longitude value \((ddm.mmmm)\)
- **GPS_4**: longitude east or west value
- **COMP**: compass value
- **ACC_1**: acceleration value in the x direction
- **ACC_2**: acceleration value in the y direction
- *****: termination symbol
Therefore, Figure 4.5 shows how the New-GUARD system is implemented.

![Diagram](image)

Figure 4.5. New-GUARD system implementation via header data

As shown, the hardware outputs its header data at a specified interval serially and the Windows Mobile device at the other end will grab the header and process the data using its SPRT GUI functions. Table 4.1 shows an example data using headers from driving around a perimeter in Las Vegas, Nevada.
Table 4.1. Example header data from a perimeter in Las Vegas, Nevada.

<table>
<thead>
<tr>
<th>ID</th>
<th>GC</th>
<th>Time</th>
<th>GPS_1</th>
<th>GPS_2</th>
<th>GPS_3</th>
<th>GPS_4</th>
<th>COMP</th>
<th>ACC_1</th>
<th>ACC_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>500</td>
<td>16:10:58</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>189</td>
<td>3</td>
<td>0*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:10:59</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>188</td>
<td>0</td>
<td>2*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:11:00</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>189</td>
<td>0</td>
<td>3*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:11:01</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>188</td>
<td>-1</td>
<td>2*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:11:02</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>189</td>
<td>6</td>
<td>-4*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:11:03</td>
<td>36.06.34</td>
<td>N</td>
<td>115.08.30</td>
<td>W</td>
<td>191</td>
<td>10</td>
<td>8*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:16:36</td>
<td>36.06.37</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>185</td>
<td>3</td>
<td>-12*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:16:37</td>
<td>36.06.37</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>187</td>
<td>19</td>
<td>-12*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:18:19</td>
<td>36.06.39</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>217</td>
<td>5</td>
<td>-1*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:18:20</td>
<td>36.06.39</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>217</td>
<td>8</td>
<td>-7*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:18:21</td>
<td>36.06.39</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>220</td>
<td>12</td>
<td>-3*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:18:22</td>
<td>36.06.39</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>220</td>
<td>9</td>
<td>-3*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:18:23</td>
<td>36.06.39</td>
<td>N</td>
<td>115.08.26</td>
<td>W</td>
<td>219</td>
<td>6</td>
<td>-7*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:40</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.15</td>
<td>W</td>
<td>198</td>
<td>3</td>
<td>-16*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:41</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.14</td>
<td>W</td>
<td>192</td>
<td>0</td>
<td>-23*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:42</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.14</td>
<td>W</td>
<td>190</td>
<td>0</td>
<td>-18*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:43</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.14</td>
<td>W</td>
<td>192</td>
<td>0</td>
<td>-6*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:44</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.13</td>
<td>W</td>
<td>192</td>
<td>-3</td>
<td>-13*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:45</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.13</td>
<td>W</td>
<td>193</td>
<td>-4</td>
<td>-17*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:27:46</td>
<td>36.06.00</td>
<td>N</td>
<td>115.07.12</td>
<td>W</td>
<td>193</td>
<td>-4</td>
<td>-22*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:32</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>227</td>
<td>3</td>
<td>-18*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:33</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>228</td>
<td>8</td>
<td>-17*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:34</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>227</td>
<td>13</td>
<td>-20*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:35</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>228</td>
<td>12</td>
<td>-21*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:36</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>226</td>
<td>10</td>
<td>-14*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:37</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>226</td>
<td>8</td>
<td>-16*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:38</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>226</td>
<td>5</td>
<td>-21*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:35:39</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.14</td>
<td>W</td>
<td>226</td>
<td>7</td>
<td>-23*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:37:20</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.31</td>
<td>W</td>
<td>228</td>
<td>9</td>
<td>-15*</td>
</tr>
<tr>
<td>S1</td>
<td>500</td>
<td>16:37:29</td>
<td>36.06.52</td>
<td>N</td>
<td>115.08.31</td>
<td>W</td>
<td>224</td>
<td>8</td>
<td>-13*</td>
</tr>
</tbody>
</table>

Figure 4.6 shows how the entire original data set looks on a standard Google Map, without setting alarm levels to each data sample.
4.3.3 Parsing

The header data comes in byte by byte in a comma delimited format. Therefore, the header is parsed in order to extract the appropriate data for further SPRT GUI processing. The parsing function creates an array of entries and stores each data variable that is separated by a comma in its correct location in that array. Currently, the New-GUARD system has 11 entries corresponding to 11 pieces of important information. They are the 11 header entries described previously:

Figure 4.6. Original Las Vegas data set mapped out via Google Maps
4.3.4 Conversion

After the header information is stored in an array, it is converted into a meaningful format for further processing by the GUI system. The simple conversions include ID (A0) being converted into string format, the GC (A1) value converted to integer, the compass value (A7) converted to degrees, and the accelerometer (x,y) (A8-A9) values converted to an integer format. The more complicated conversions are the GPS and time stamp string conversions.

4.3.4.1 Time string conversion

The array entry A2 is used for the time string. It has the following format:

\[ xx:yy:zz \]

where

- **xx**: indicates hours
- **yy**: indicates minutes
- **zz**: indicates seconds

The string is separated by the same parsing technique described before, except using colons (:) as delimiters. Once the data is separated out, it can be put together however the GUI program wishes. The GUI works with time stamps from only using hours, minutes, and seconds to using time stamps with more detailed information, like **year:month:day:hour:minutes:seconds:milliseconds**.

4.3.4.2 GPS string conversion

The array entries A3 through A6 must be used to find the needed GPS format to pass over to the GUI for further processing. If the proper format is not given to the GUI, the mapping functionalities will produce inaccurate results. When the GPS data is fed from
the hardware unit to the GUI it needs to be converted from \texttt{ddmm.mmmm} to the \textbf{decimal degrees} format for proper mapping. The following algorithm is used to convert the hardware GPS string into a format processed by the GUI:

1. Attain new GPS string containing all GPS entries: \texttt{[A3, A4, A5, A6]}
2. \texttt{A3} contains 2 values delimited by commas for the latitude. Separate them into \texttt{[A3(0), A3(1)]}
3. \texttt{A3(0) = ddmm} and \texttt{A(1) = .mmmm}. Set the integer portion to be \texttt{dd}
4. Set the decimal portion = \( \frac{\texttt{mm.mmmm}}{60} \)
5. Lastly, set the decimal degrees = integer portion + decimal portion
6. Repeat steps 1-5 using \texttt{A5} for the longitude decimal degree value

Once the decimal agree format is found for each longitude and latitude, a compensation for North, South, East, and West must be made. The algorithm for this is as follows:

1. Attain latitude decimal degree value
2. If \texttt{A4 = S} (south), make the latitude decimal degree value:
   \[
   \textit{latitude\_decimal\_degree} * -1
   \]
3. If \texttt{A4 = N} (north), make change to the latitude decimal degree value
4. If \texttt{A6 = W} (west), make the longitude decimal degree value negative:
   \[
   \textit{longitude\_decimal\_degree} * -1
   \]
5. If \texttt{A6 = E} (east), make no change to the longitude decimal degree value

Consequently, there will be only two values, latitude and longitude, for further GUI processing and mapping.
4.3.5 Movement correction

The New-GUARD system has the functionality of correcting the technician based on where he/she is holding the unit. This means if the unit deviates from its original location while being held, it will inform the technician. In addition, if the unit is moved too fast, the technician will be informed based on the accelerometer readings.

4.4 Timing

Timing is important when it comes to synchronization. The hardware has its own clocking device so data is packed into a header and sent out based on its clock. Currently the New-GUARD system hardware unit sends out a header packet every second. This means that the integration software written on the SPRT GUI side must retrieve this data whenever it’s available. The way it does this is by threading principles.

In the SPRT GUI there is a separate integration thread that is invoked at startup and runs at all times. The thread runs continuously and checks for new data via the Bluetooth or serial ports. When new data arrives at these ports, it runs the integration functions described before (parsing, data conversion, etc) and then the resulting data is sent to the main GUI thread which takes care of the rest of the data processing. The main GUI program grabs data from the integration thread when it is finished processing the previous sample and therefore is ready to process a new radiation sample point. This process is described by Figure 4.7.
As shown, the hardware unit sends out concatenated header data every second. The integration piece runs concurrently and collects the data as it comes. In turn, when the main SPRT core is ready to process another data point, it acquires the most current sample from the integration thread and outputs a result to the main GUI.
In this section, the New-GUARD system is used to obtain results. The experiments were set up to show that the New-GUARD system is able to be used in real-time for radiation detection by a field surveyor. The data for the testing was collected using radiation samples at the Nellis Remote Sensing Laboratory in Las Vegas, Nevada. The Windows Mobile device used for testing and integration is called the HP iPAQ hw6515.

Figure 5.1 Windows mobile device, HP iPAQ hw6515, used for testing and integration of the New-GUARD system
The HP iPAQ has the following important specs:

- Modes: GSM 850 / GSM 900 / GSM 1800 / GSM 1900
- Weight: 5.8 oz (164 g)
- Dimensions: 4.65" x 2.8" x 0.71" (118 x 71 x 18 mm)
- Battery Life: Talk: 4-168 hours
- Battery Type: Lilon (1200 mAh / user-changeable)
- Display: Type: LCD (Color TFT/TFD)
- Colors: 65,536 (16-bit)
- Resolution: 240 x 240 pixels
- Processor: 312 MHz Intel XScale PXA272
- Memory: 55 MB (built-in, flash shared memory)

The algorithm used in this real-time processing test was the LIFEO scheme, which had the following parameters:

- Test strengths: $\alpha = 0.05$ & $\beta = 0.01$
- Threshold value: $a_{val} = -4.55388$, $b_{val} = -2.98568$ & $c_{val} = -0.784097$
- Hypothesis = NN
- $\rho = 1$, base = 2.71827
- Hypothesis = NN
- Dynamic background = OFF
- MLE = OFF

The background mean and standard deviation are calculated by the New-GUARD system by allowing it to take the background radiation values for an interval of five minutes. At the remote sensing lab, the background statistics are as follows:
• Background Mean: 322.547112
• Background Standard Deviation: 66.322044
• K = 300

In all cases, the SPRT algorithm response time is measured to be less than 1 millisecond, so the real-time response time is dependent on how many samples are available per second. Currently, the New-GUARD system receives one sample per second.

The data used in these experiments is taken by having sources shielded in a region. When an alarm level condition is needed, the source is unshielded and measurements are taken. Therefore, the experiment setup is as follows: non-source region while source is being shielded, source region when source is unshielded, and back to non-source region when source becomes shielded again. Figure 5.1 shows the profile of this procedure.

![Figure 5.1](image1.png)

**Figure 5.1. Profile of the radiation dataset used for testing**

As shown, the source is unshielded shortly after sample 40 and shielded sometime after sample 60.
5.1 SPRT algorithm response time

The first metric measured is the response time of the SPRT core algorithm based on specific settings. The New-GUARD system achieves a new sample every second. This test measures how far back in the buffer the algorithm needs to go to make a decision. Table 5.1 shows the response time results using the following settings for imin/imax.

- \text{Imin source} = 1, \text{Imin background} = 1
- \text{Imax source} = 8, \text{Imax background} = 11

Table 5.1. Response time using specific imin/Imax settings

<table>
<thead>
<tr>
<th>Source GC range</th>
<th>Background to Source (samples)</th>
<th>Source to Background (samples)</th>
<th>Alarm Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-600</td>
<td>2</td>
<td>1</td>
<td>1-2</td>
</tr>
<tr>
<td>600-800</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>800-1000</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1000-1200</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1200-1400</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1400-1600</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1600-1800</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1800-2000</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2000-2200</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>3000-3200</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4000-4200</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5000-5200</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

As shown, the response time for the New-GUARD on detecting radiation is virtually instantaneous, as it only takes the time allocated for attaining a new sample to make a decision.

- \text{Imin source} = 1, \text{Imin background} = 2
- \text{Imax source} = 8, \text{Imax background} = 11
Table 5.2. Response time 2 using specific Imin/Imax settings

<table>
<thead>
<tr>
<th>Source GC range</th>
<th>Background to Source (samples)</th>
<th>Source to Background (samples)</th>
<th>Alarm Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-600</td>
<td>2</td>
<td>3</td>
<td>1-2</td>
</tr>
<tr>
<td>600-800</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>800-1000</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1000-1200</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1200-1400</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1400-1600</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1600-1800</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1800-2000</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2000-2200</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3000-3200</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4000-4200</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5000-5200</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

In this case, the response time for the New-GUARD on detecting from background to source is almost instantaneous. However, since the Imin parameter is set to 2, there will be a little delay in detecting a background region coming from a source region. This delay is not harmful, as no danger will come to the technician walking from a source to a background region. As the Imin/Imax parameters change, the response time will also change. Therefore, the technician must set the parameters corresponding to the region he/she is surveying.

5.1.1 Other performance metrics

The New-GUARD system works in real-time. The main delay is dependent on the sampling rate of the radiation detector. At the Nellis Air force base, the radiation detector had the capability of sampling data between 1 to 3 samples per second, meaning a new GC or NC value would be ready to process at those times. Table 5.3 presents response time of key components of the New-GUARD system.
Table 5.3. New-GUARD system response times for key components

<table>
<thead>
<tr>
<th>Description</th>
<th>Response time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware concatenation and transfer</td>
<td>1,000 (dependent on detector)</td>
</tr>
<tr>
<td>SPRT algorithm</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Integration cycle (parsing, conversion, correction)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Sounds (playing .wav file)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Display (graphing per incoming data)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Display (invoking colors)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Table (new entry per incoming data)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Output logging</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Text message (SMS) alarms</td>
<td>300</td>
</tr>
<tr>
<td>Mapping (per incoming data)</td>
<td>5-6</td>
</tr>
</tbody>
</table>

The table shows that the New-GUARD system detects radiation real-time. Most of the functionalities of the New-GUARD system were too quick (less than 1 ms) to gauge, except text message alarms and the mapping procedures. Text messaging has a delay because it has to pass through a pre-existing network before it reaches its destination. The mapping was tested using 1700 data points, taking between 5-6 ms per data point. This delay is attributed to the drawing functions of the Google Maps platform.

5.2 Other Implementations

Currently, the New-GUARD system works well for real-time processing. It uses the .NET Compact Framework as a hardware-dependent environment for running programs.
on Windows Mobile devices. However, this is an off-the-shelf approach. For a more optimal solution, the implementation should be done strictly in hardware. The size of the unit would then be reduced significantly. In addition, hardware level programming results in a more efficient design [34]; therefore, a VHDL code for the SPRT algorithm was written for future hardware implementation.

5.2.1 VHDL implementation

The VHDL code is implemented by using FSMs (Finite State Machines). This way, reduction can be done by performing Logic Synthethis techniques, where truth tables are decomposed. ROM size reduction techniques will yield a more efficient FPGA (Field-Programmable Gate Array) design.

5.2.1.1 Algorithm Flow

The data coming in for analysis will come in whenever it is available. The algorithm will attain the data and process it via the SPRT LIFELO algorithm. The data is stored in an internal memory buffer. The VHDL algorithm flow is as follows:

1. Create a buffer vector of size 11. Fill the buffer with 11 data points.
2. Start processing on the newest data point in the buffer.
3. Attain and output SPRT results.
4. Wait for new data point. Once a new data point comes in, insert the new data point and shift out the old data point from the buffer.
5. Go back to step 2.

5.2.1.2 VHDL structure

The VHDL SPRT algorithm is designed to work on an FPGA. Hardware input and output signals are therefore designed. They are as follows:
• Buffer: the size 11 buffer for the radiation data (input / output).

• Alarm: The alarm value. This will be the value determining if there is radiation or not in the area (output).

• CLK: The clock (input).

• Datain: The signal for dealing with incoming radiation data (input).

• Clear: The signal for clearing all data (input).

The buffer uses the hardware memory to store data. The alarm signal is outputted to an external hardware unit for further actions. The CLK is used for synchronization. The Datain signal is the newest radiation data value coming into the hardware system. Lastly, the clear signal allows for a reset of the hardware system.

The signals traverse the hardware system based on a FSM. Only the necessary states to execute the SPRT algorithm are designed:

• Initialization: This state calculates all the background and source statistics.

• Set Background parameter: This state sets the background sampling amount. It changes the used imin/imax values to the background’s imin/imax values.

• Set Source parameter: This state sets the source sampling amount. It changes the used imin/imax values to the source’s imin/imax values.

• LLR calculation: This state calculates the LLR value.

• Alarm calculation: This state determines the alarm level of the sample.

• Alarm off: This state sets the alarm state to OFF. This state loops back to the initialization stage.

• Alarm on: This state sets the alarm state to on and invokes an alarm.
- Decision state: This state is implemented if a decision can’t be made. It will come here to make the final decision whether the new samples are a background or a source.

Figure 5.3 shows the developed FSM:

![State machine of the VHDL SPRT algorithm](image)

This procedure runs based on clock pulses. It deals with the SPRT core computations. There is one other procedure running in parallel within the VHDL SPRT system: the buffer handling process.
Driven by the datain signal, the buffer handling process is only invoked when a new data point comes into the system. There are two states that run in this parallel process:

- **Fill buffer**: This is the state that fills the buffer initially. This state will be invoked until 11 data points are inside of the buffer. Afterwards, it will change the state to the shift buffer state.

- **Shift buffer**: This is the state that shifts new data in and old data out of the buffer. The idea is that once 11 data values are in the buffer, if a new data value comes in, the last data in the buffer is shifted out and the new data is placed at the beginning of the buffer.

The program was tested and works reliably. Figure 5.4 and 5.5 are simulations of the VHDL SPRT algorithm.
Figure 5.4 Table simulation of the VHDL SPRT algorithm

| Name | Value | Si | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 | 220 | 240 | 260 | 280 | 300 | 320 | 340 | 360 | 380 |
|------|-------|----------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| `i`  | state | unit | &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift| &shift|
| `#` | bufler | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` | `bufler` |
| `#` | bufler_temp | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` | `bufler_temp` |
| `#` | buflerNTU | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` | `buflerNTU` |
| `#` | buflerNTL | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` | `buflerNTL` |
| `#` | buffer_D | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` | `buffer_D` |
| `#` | alarm | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` | `alarm` |

Figure 5.5 Waveform simulation of the VHDL SPRT algorithm
The simulations show that the VHDL implementation works as it should. The first step was to attain data to fill the buffer with 11 elements. The buffer state remains in the fill buffer state until 11 new data points arrive. Once the buffer is filled, the buffer state is changed to the shift buffer state. The clock is now invoked and the SPRT calculations are now able to occur. The program then moves to the set background state, as the data points resemble background data (data is between a GC of 200-400). From there, the program moves to the LLR state, and then the LLR value is calculated. Next, the state is changed to the alarm calculation state, where the alarms for the data points are calculated. This state determines that there is no alarm so the program reverts back to the initialization state. The buffer is then at the shift buffer state, so the new data point comes in at the first spot and the oldest data point is shifted out. This changes the distribution of the data so that it now falls into the source nature (GC values of 1,000). As can be seen, an alarm is triggered with this new data point and the program ends up back at the initialization state.

5.3 Future Work

The New-GUARD system is a solid and innovative solution to current needs in radiation detection. The prototype is ready for any technician to do a radiation field survey no matter his/her expertise. However, there is always room for improvement and additions for any device.

The main improvement needed would be to try to reduce the size of the unit. This would be solved by implementing the SPRT algorithm on a separate chip, like a Field-programmable gate array (FPGA) or an Application Specific Integrated Circuit (ASIC).
The VHDL implementation of the SPRT algorithm previously described is a perfect solution to this problem. In addition, implementing the device purely in hardware would add to the efficiency of the system.

Lastly, adding to the functionalities of the unit is important to its continuing success in the battle against radiation. A nice feature would be to eliminate the need for a technician in the field. The strategy is to keep all humans safe from radiation damage. This can be done by deploying the hardware unit onto a mobile robot.

5.3.1 Mobile robot development

The New-GUARD system can work as a standalone unit. The only thing it is missing is the functionality of movement. Attaching the unit to a smart mobile unit will eliminate the need for any human presence in a radiation field. Figure 5.6 describes this procedure.
As shown, the New-GUARD system will be attached to another microcontroller, which is attached to a mobile unit. The idea is that the microcontroller will control the movement of the unit using directional commands. The New-GUARD system will also send out its data to a pre-existing GSM network for either radiation notification or further processing.

The scheme is that the mobile unit, or robot, will be sent out into a possible radiation field. The unit will then move throughout the radiation field while sending out SPRT radiation data, location information (GPS, accelerometer and compass) to a GSM network, and mapping out the radiation field. The GSM network will either take the data and do some post-processing, or log the data for future use. In addition, the GSM network will also have the ability to send commands back to the unit, allowing the functionality of changing the behavior of any part of the new system. This includes parameters of the SPRT algorithm, the movement of the robot, and other inherit functionalities of the New-GUARD system.

5.3.2 Movement algorithm

The robot needs to move throughout a radiation field in an intelligent manner. Therefore, it is best that the robot moves towards the highest radiation areas as fast as possible. This way, the central station will know quickly the most important problematic areas in a region. The proposed movement algorithm is as follows:

1. Check radiation parameters in specified directions
2. Log data
3. Determine strongest potential source
4. Move towards strongest potential source
5. Map out region and send radiation info to central station via GSM network
6. Repeat steps 1-4

The robot will then move in an intelligent manner throughout a radiation field, sending out all meaningful data back to the base station. Most importantly, a technician would not be required on site, eliminating potential human danger.
CHAPTER 6

CONCLUSION

The development and prototyping of the New-GUARD system was presented. This includes the key hardware components, the implementation of the software, and the integration of the two. It was shown that the New-GUARD system was an innovative and effective new generation detector unit in detecting radiation hazards. The SPRT algorithm, using the LIFELO scheme, was proven to be a robust radiation detection algorithm.

The software implementation of the New-GUARD system was discussed. A graphical user interface (GUI) with ample features was developed to make the radiation detection process more readable, flexible, appealing, and effective as an engineering solution. The GUI system was coded using the Visual Basic .NET coding language, the Windows Mobile SDK for code deployment to a mobile device, and the .NET compact framework to run the generated code on a specified device.

The key hardware components for the New-GUARD were explained. These include a radiation detector, global positioning system (GPS), compass, accelerometer, Bluetooth interface, LCD, and memory. All hardware components are tied together by a microcontroller and communicate with the outside world via a Bluetooth or serial interface.
Integration between the hardware and software components defines the heart of the New-GUARD system. The hardware unit sends out a header file to the GUI portion for SPRT core algorithm processing and GUI output. Timing has also proven to be an important factor. The New-GUARD system improves the processing throughput by using threading principles.

Evaluation of the performance metrics shows that the New-GUARD system works in real-time. The only notable delay is based on the sampling rate of the radiation detector connected to the hardware unit. Most system operations take less than one millisecond to run.

Future work includes improving and adding on to the New-GUARD system. The aim is to make the system more efficient and innovative. One way is by implementing the system strictly onto a hardware unit, like an Application Specific Integrated Circuit (ASIC), to improve upon both area and speed of the old system. Using the hardware description language VHDL, a behavioral/RTL structure of the new SPRT algorithm is implemented via a finite state machine and a buffer management system. Lastly, a solid addition to the New-GUARD system would be to eliminate the need of any human presence in a radiation field. Since the New-GUARD system works as a standalone unit, the solution would be to make the unit mobile. This could be done by attaching the New-GUARD system onto a mobile unit driven by its own microcontroller, delegating all movement commands. The microcontroller would be given directional commands based on a smart algorithm determining the highest potential source in a region. The system would move towards a radiation field, mapping out all points of interest, and sending out
all import data to a pre-existing network. The New-GUARD system will prevent a great number of lives from being harmed due to radiation hazards.
REFERENCES

[1] Radiation Detection Instrumentation at the Department of Energy,

[2] Understanding Risk Analysis,
ID=14418

v34, pg 260-265, 1991

[4] K.J.Hofstetter, "Environmental radiation monitoring technology: Capabilities and


[6] C. Barker, “Moving From Text-based to Graphical User Interface”, Lockheed Martin,
2004

Hutchison and Ross Inc.

Communications” v3, Computer Science Press, Rockville

Transaction in Communication, v 38, pg 165-171, 1990


[27] “Windows Mobile 6 Documentation,”


[29] “.NET compact framework 2.0,”


[31] “Using the .NET compact Framework,”


[33] “Hardware Vs. Software,”
   http://www.embedded.com/story/OEG20011009S0072

[34] “FPGA programming step by step,”
VITA

Graduate College
University of Nevada, Las Vegas

Aaron Ponzio

Home Address:
59 Huntfield, Henderson, Nevada 89074

Degrees:
Bachelors of Science in Computer Engineering, 2006
University of Nevada, Las Vegas

Publications:


Thesis Title: Prototyping the New-GUARD Portable Device for Radiation Detection

Thesis Examination Committee:
Chairperson, Dr. Shahram Latifi, Ph.D.
Committee Member, Dr. Henry Selveraj, Ph.D.
Committee Member, Dr. Venkatesan Muthukumar, Ph.D.
Graduate Faculty Representative, Dr. Laxmi Gewali, Ph.D.