

Spring 2013

Pulse patency and oxygenation sensing system development to detect g-induced loss of consciousness

Herman Pretorius

University of New Hampshire, Durham

Follow this and additional works at: <https://scholars.unh.edu/thesis>

Recommended Citation

Pretorius, Herman, "Pulse patency and oxygenation sensing system development to detect g-induced loss of consciousness" (2013). *Master's Theses and Capstones*. 775.
<https://scholars.unh.edu/thesis/775>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.

**PULSE PATENCY AND OXYGENATION SENSING SYSTEM
DEVELOPMENT TO DETECT G-INDUCED LOSS OF CONSCIOUSNESS**

BY

**HERMAN PRETORIUS
B.S., University of New Hampshire, 2011**

THESIS

**Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of**

**Masters of Science
in
Electrical Engineering**

May, 2013

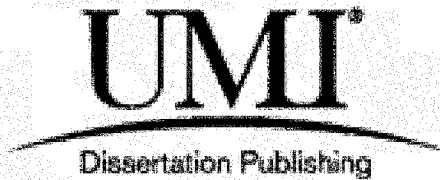
UMI Number: 1523777

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1523777

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code.

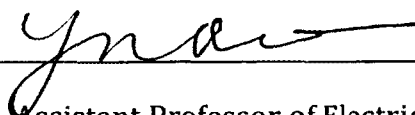


ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

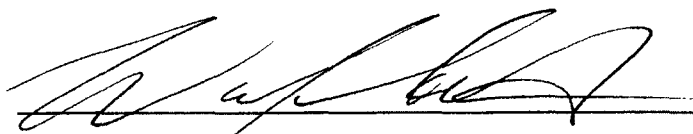
This thesis has been examined and approved



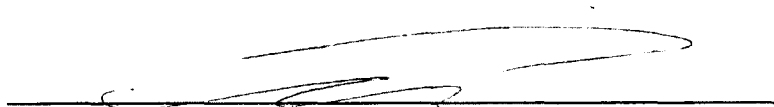
Thesis Director, Dr. John LaCourse, Chairman and Professor of Electrical and
Computer Engineering



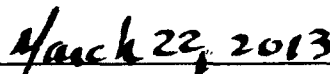
Dr. Qiaoyan Yu, Assistant Professor of Electrical and Computer
Engineering



Dr. Wayne Smith, Senior Lecturer of Electrical and Computer
Engineering



1LT Timothy Plante, Aerospace & Operational Physiologist, U.S. Air Force



Date

DEDICATION

To: My love, Christina.

Without you, this wouldn't have been done.

Thank you

ACKNOWLEDGEMENTS

First and foremost, I want to thank Dr. LaCourse and company for their support and professional guidance. To all of you, thank you. I will forever remain envious of your professionalism and knowledge.

I want to thank my family for their unrelenting love and support over my entire academic career. Through all the highs and lows, your support, guidance, and sacrifice gave me the opportunity to do this, and for that I will forever be grateful. To Christina, whose patience and love never wore down, thank you. There is no question the road was long and some of the obstacles were large, but there is nothing that can't be accomplished standing on all your shoulders.

To Timothy Plante for the idea and your support. Your assistance will always be appreciated along with our friendship. Thank you.

To the entire ECE department Faculty and Ms. Kathy Reynolds. Without any of you, this would not have been possible.

Sofie, you made the book.

Here is to the next chapter and whatever challenges that may bring...

Cheers!

FORWARD

“Now, of course, people ask me all the time, they say to me, “What is the secret to success?” ...There are some people that just like to kick back and coast through life and others want to be very intense and want to be number one and want to be successful. And that’s like me. I always wanted to be very intense; I always wanted to be number one. I took it very seriously, my career...And so this intensity always paid off for me, this commitment always paid off for me. So here are some of the rules:

The first rule is: Trust yourself...What I mean by that is, so many young people are getting so much advice from their parents and from their teachers and from everyone. But what is most important is that you have to dig deep down, dig deep down, and ask yourselves, who do you want to be? Not what, but who...And, I’m talking not what your parents or your teachers want you to be, but you. I’m talking about figure out for yourselves what makes you happy, no matter how crazy it may sound to other people...

Rule number two is: Break the rules. We have so many rules in life about everything. I say break the rules. Not the law, but break the rules...It is impossible to be a maverick or a true original if you’re too well behaved and don’t want to break the rules. You have to think outside the box. That’s what I believe. After all, what is

the point of being on this earth if all you want to do is be like by everyone and avoid trouble?...

Rule number three: Don't be afraid to fail. Anything I've ever attempted, I was always willing to fail...You can't be paralyzed by fear of failure or you will never push yourself. You keep pushing because you believe in yourself and in your vision and you know that it is the right thing to do, and success will come. So don't be afraid to fail...

Which brings me to rule number four, which is: Don't listen to the naysayers. How many times have you heard that you can't do this and you can't do that and it's never been done before? Just imagine if Bill Gates had quit when people said it can't be done...I hear this all the time. As a matter of fact, I love it when someone says that no one has ever don't this before, because then when I do it that means that I'm the first one that has done it. So pay no attention to the people that say it can't be done...

And that brings me to rule number five, which is the most important rule of all: Work your butt off. You never want to fail because you didn't work hard enough. I never wanted to lose a competition...because I didn't work hard enough. I always believed in leaving no stone unturned. Mohammed Ali, one of my great heroes, had a great line in the '70's when he was asked, "How many sit-ups do you do?" He said, "I don't count my sit-ups. I only start counting when it starts hurting. When I feel pain, that's when I start counting, because that when it really counts." That's what makes you a champion...No pain, no gain. So many of those lessons that I apply in life I have learned from sports, let me tell you, and especially that one. And let me tell you, it is

important to have fun in life, of course. But, when you're out there partying, horsing around, someone out there at the same time is working hard. Someone is getting smarter and someone is winning. Just remember that. Now, if you want to coast through life, don't pay attention to any of those rules. But if you want to win, there is absolutely no way around hard, hard work...Just remember, you can't climb the ladder of success with your hands in your pockets.

And that takes me to rule number six, which is a very important rule: it's about giving back. Whatever path that you take in your lives, you must always find time to give something back, something back to your community, give something back to your state or to your country...And let me tell you something, reaching out and helping people will bring you more satisfaction than anything else you have ever done...

So let me tell you, as you prepare to go off into the world, remember those six rules: trust yourself, break some rules, don't be afraid to fail, ignore the naysayers, work like hell, and give something back."

Arnold Schwarzenegger

University of Southern California

May 15, 2009

(Source: www.graduationwisdom.com)

TABLE OF CONTENTS

| | |
|--|-----|
| DEDICATION | iii |
| ACKNOWLEDGEMENTS..... | iv |
| FORWARD | v |
| TABLE OF CONTENTS | vii |
| LIST OF FIGURES | x |
| LIST OF TABLES | xiv |
| ABSTRACT | xv |
| INTRODUCTION | 1 |
| CHAPTER 1: G-INDUCED LOSS OF CONSCIOUSNESS..... | 9 |
| A HISTORICAL PROBLEM | 9 |
| MODEL OF HYDROSTATIC PRESSURE AND HUMAN PHYSIOLOGY | 11 |
| INTRAOCULAR PRESSURE AND VISUAL SYMPTOMS | 13 |
| GLOC PREVENTION | 15 |
| GLOC DETECTION SYSTEM: INTRODUCTION..... | 16 |
| LOSS OF CONSCIOUSNESS MONITORING SYSTEM..... | 19 |
| SUPERFICIAL TEMPORAL ARTERY | 21 |
| PLETHYSMOGRAPHY AND PULSE PATENCY..... | 22 |
| BLOOD OXYGENATION | 24 |
| THESIS HYPOTHESIS | 27 |
| SYSTEM DESIGN: BOPS..... | 28 |
| CHAPTER 2: METHODS..... | 29 |
| INTRODUCTION..... | 29 |
| PULSE PATENCY | 30 |
| TRIPLE AXIS ACCELEROMETER | 33 |
| ANALOG SIGNAL CONDITIONING FRONT END | 36 |
| BUFFERS | 38 |
| SALLEN-KEY HIGH PASS FILTER..... | 39 |
| GAIN STAGES..... | 46 |
| SALLEN-KEY LOW PASS FILTER..... | 48 |
| DC LEVEL SHIFT | 53 |
| SYSTEM CIRCUIT AND FREQUENCY RESPONSE..... | 56 |
| MATLAB AND MICROPROCESSOR..... | 57 |
| MATLAB AND MICROPROCESSOR ALGORITHM: OVERVIEW | 58 |
| SAMPLING RATE (FREQUENCY)..... | 60 |
| PEAK DETECT METHODS | 62 |

| | |
|---|-----|
| MOVING AVERAGE ALGORITHM | 67 |
| PULSE DETECTION CRITERIA AND VARIANCE | 69 |
| AUTO-PILOT WARNING AND TIME CONSTRAINTS | 72 |
| FINAL PEAK DETECT BLOCK DIAGRAM | 74 |
| PULSE PATENCY SIMULATION | 75 |
| SIGNAL SELECTION | 76 |
| WAVEFORM GENERATOR AND DECAY WINDOWS | 77 |
| SIGNAL INTERPOLATION AND DECIMATION | 83 |
| PROBLEMS WITH WAVEFORM GENERATOR | 84 |
| SIMULINK MODEL | 85 |
| PULSE PATENCY PHYSICAL MODEL | 93 |
| VALVE CHARACTERISTICS | 95 |
| HUMAN ELEMENT TESTING | 98 |
| PULSE OXIMETER | 99 |
| | |
| CHAPTER 3: RESULTS | 103 |
| INTRODUCTION..... | 103 |
| DESIGN VERIFICATION..... | 104 |
| ALGORITHM VERIFICATION: SIMULATION..... | 112 |
| ALGORITHM PERFORMANCE: PHYSICAL MODEL..... | 117 |
| PHYSICAL MODEL: MATLAB | 118 |
| PHYSICAL MODEL: ARDUINO R3 UNO | 120 |
| HUMAN MODEL: MICROPROCESSOR AND MATLAB | 124 |
| SIMULATION: G-PROFILE SIMULATION..... | 131 |
| VARIATION IN MOVING AVERAGE ARRAY SIZE..... | 132 |
| PHYSICAL MODEL: CONTROLLED NOISE ANALYSIS | 140 |
| PULSE OXIMETER | 143 |
| PULSE OXIMETER LOCATIONS | 144 |
| PULSE PATENCY AND PPG CORRELATION..... | 148 |
| | |
| CHAPTER 4: DISCUSSION AND FUTURE WORK..... | 150 |
| OVERVIEW..... | 150 |
| PULSE PATENCY SENSOR..... | 151 |
| BLOOD OXYGENATION | 153 |
| MICROPROCESSOR ALGORITHM | 155 |
| FUTURE WORK | 158 |
| | |
| LIST OF REFERENCES..... | 161 |
| | |
| APPENDIX A: WAVEFORM GENERATOR CODE..... | 164 |
| APPENDIX B: LINEAR AND EXPONENTIAL DECAY WINDOW CODE..... | 169 |
| APPENDIX C: MATLAB ALGORITHM CODE: USER INPUT MOVING AVERAGE ARRAY..... | 171 |
| APPENDIX D: MATLAB ALGORITHM CODE: SELF-INITIATING MOVING AVERAGE..... | 176 |
| APPENDIX E: ARDUINO ALGORITHM CODE | 181 |

LIST OF FIGURES

| | |
|--|----|
| FIGURE 1 – VECTORIAL DEPICTION OF G-INDUCED FORCES ON PILOT | 2 |
| FIGURE 1.1 – PHYSIOLOGICAL EFFECT OF +Gz ACCELERATION | 12 |
| FIGURE 1.2 - +Gz ACCELERATION VERSUS TIME AND TYPICAL VISUAL SYMPTOMS..... | 14 |
| FIGURE 1.3 – EFFECT OF RECLINED SEAT ON PILOT BLOOD PRESSURE | 15 |
| FIGURE 1.4 – SEQUENTIAL GLOC DETECTION SYSTEM | 17 |
| FIGURE 1.5 – G-WARS AUTO RECOVERY SYSTEM..... | 18 |
| FIGURE 1.6 – HEAD LEVEL HUMAN PHYSIOLOGY | 21 |
| FIGURE 1.8 – SHENDER ET AL.: rSO ₂ RESPONSE WITH +Gz ACCELERATION | 24 |
| FIGURE 1.9 – TRIPP ET AL.: RESULTS SHOWING REGIONAL TISSUE OXYGENATION AS A FUNCTION OF GLOC STAGES..... | 25 |
| FIGURE 1.10: TRIPP ET. AL: NOSE OXYGENATION MONITOR AND MELKER ET AL NOSE OXYGENATION MONITOR..... | 26 |
| FIGURE 1.11 – MMA7361 THREE AXIS ACCELEROMETER..... | 27 |
| FIGURE 2.1 – OVERALL SYSTEM BLOCK DIAGRAM | 30 |
| FIGURE 2.2 – BLOCK DIAGRAM OF PULSE PATENCY SENSOR | 31 |
| FIGURE 2.3 – AN EXPONENTIAL DECAY WINDOW, A LINEAR DECAY WINDOW, AND AN IMMEDIATE LOC WINDOW USED FOR TESTING AND SIMULATING LOC TO ENSURE RELIABILITY IN ALGORITHM..... | 32 |
| FIGURE 2.4 – BEST PLACEMENT OF TRIPLE AXIS ACCELEROMETER | 34 |
| FIGURE 2.5 – BLOCK DIAGRAM OF DISCUSSED BLOCK IN BOLD..... | 38 |
| FIGURE 2.6 – SCHEMATIC OF BUFFER..... | 38 |
| FIGURE 2.7 – BLOCK DIAGRAM OF DISCUSSED BLOCK IN BOLD..... | 39 |
| FIGURE 2.8 – SALLEN-KEY HIGH PASS BUTTERWORTH FILTER TOPOLOGY..... | 39 |
| FIGURE 2.9 – POLES AND ZEROES OF FOURTH ORDER HIGH PASS FILTER..... | 41 |

| | |
|--|----|
| FIGURE 2.10 – FREQUENCY RESPONSE OF FIRST STAGE | 44 |
| FIGURE 2.11 – FREQUENCY RESPONSE OF SECOND STAGE..... | 45 |
| FIGURE 2.12 – COMBINED TRANSFER FUNCTION FREQUENCY RESPONSE | 46 |
| FIGURE 2.13 – BLOCK DIAGRAM OF DISCUSSED BLOCK IN BOLD | 46 |
| FIGURE 2.14 – BLOCK DIAGRAM OF DISCUSSED BLOCK IN BOLD | 48 |
| FIGURE 2.15 – SALLEN-KEY LOW PASS BUTTERWORTH FILTER TOPOLOGY | 49 |
| FIGURE 2.16 – POLES AND ZEROES OF FOURTH ORDER LOW PASS FILTER..... | 50 |
| FIGURE 2.17 – FREQUENCY RESPONSE OF FOURTH ORDER LOW PASS FILTER | 52 |
| FIGURE 2.18 – BLOCK DIAGRAM OF DISCUSSED BLOCK IN BOLD | 53 |
| FIGURE 2.19 – DC LEVEL SHIFT CIRCUIT..... | 53 |
| FIGURE 2.20 – SIMPLIFIED SCHEMATIC OF INPUT..... | 54 |
| FIGURE 2.21 – FINAL SIGNAL CONDITIONING PULSE PATENCY SENSOR | 56 |
| FIGURE 2.22 – OVERALL SYSTEM FREQUENCY RESPONSE | 57 |
| FIGURE 2.23 – CONTINUOUS VS. DISCRETE 300BPM PULSE PATENCY MEASUREMENTS | 59 |
| FIGURE 2.24 – NORMALIZED FREQUENCY SPECTRUM OF 1HZ AND 5HZ HEART RATE FROM PHYSICAL MODEL | 61 |
| FIGURE 2.25 – 1 CYCLE OF A 10HZ SINE WAVE..... | 63 |
| FIGURE 2.26 – PEAK OF A 1 CYCLE 10HZ SINE WAVE | 64 |
| FIGURE 2.27 – 10 Hz SINE WAVE ERROR WITH SIMULATED ADC ERROR..... | 65 |
| FIGURE 2.28 – MOVING AVERAGE DATA DURING SIMULATION..... | 67 |
| FIGURE 2.29 – 1 Hz PULSE PATENCY SIGNAL FROM PHYSICAL MODEL | 70 |
| FIGURE 2.30 – FINAL CODE BLOCK DIAGRAM | 74 |
| FIGURE 2.31 – STORED OUTPUT VERSUS HEART BEAT | 76 |
| FIGURE 2.32 – OUTPUT SIGNAL COMPARISON BETWEEN SIMULATION AND PHYSICAL MODEL | 77 |
| FIGURE 2.33 – 60BPM AND 180BPM SIMULATION | 78 |
| FIGURE 2.34 – GENERATED SIGNAL MULTIPLIED BY WINDOW AND RESULTING OUTCOME | 81 |
| FIGURE 2.35 – GENERATED SIGNAL MULTIPLIED BY WINDOW AND RESULTING OUTCOME | 82 |
| FIGURE 2.36 – SIGNAL INTERPOLATION AND DECIMATION | 83 |
| FIGURE 2.37 – SIGNAL INTERPOLATION AND DECIMATION WITH ONE LOW PASS FILTER..... | 84 |

| | |
|--|-----|
| FIGURE 2.38 – FREQUENCY SPECTRUM OF WAVEFORM GENERATOR AT 3 HZ | 85 |
| FIGURE 2.39 – OUTPUT SIGNAL FROM 2 HZ PHYSICAL MODEL SIGNAL | 86 |
| FIGURE 2.40 – RC CIRCUIT WITH OUTPUT TAKEN ACROSS CAPACITOR..... | 87 |
| FIGURE 2.41 – SIMULINK OUTPUT SIGNAL FOR SIMULATION | 88 |
| FIGURE 2.42 – FREQUENCY SPECTRUM OF SIMULATION VERSUS PHYSICAL MODEL..... | 88 |
| FIGURE 2.43 – SIMULINK MODEL WITH ALL TRANSFER FUNCTIONS | 90 |
| FIGURE 2.44 – AUTOMATIC GAIN CONTROL BLOCK DIAGRAM | 91 |
| FIGURE 2.45 – AUTOMATIC GAIN CONTROL SIMULATION..... | 92 |
| FIGURE 2.46 – PHYSICAL MODEL SETUP..... | 94 |
| FIGURE 2.47 – RESPONSE OF ACCELEROMETER DUE TO VALVE OPENING AND CLOSING..... | 96 |
| FIGURE 2.48 – CURRENT SENSE RESISTOR VOLTAGE VERSUS FUNCTION GENERATOR VERSUS ACCELEROMETER OUTPUT | 97 |
| FIGURE 2.49 – NONIN OEM EVALUATION KIT | 101 |
| FIGURE 3.1 – EXAMPLE HUMAN PULSE PATENCY WAVEFORM..... | 105 |
| FIGURE 3.2 – FREQUENCY SPECTRUM OF HUMAN PULSE PATENCY WAVEFORM..... | 105 |
| FIGURE 3.3 – SIMULINK SIMULATED 75BPM HEART RATE | 106 |
| FIGURE 3.4 – FREQUENCY SPECTRUM OF SIMULATED WAVEFORM..... | 107 |
| FIGURE 3.5 – PHYSICAL MODEL SIMULATION OF A 75BPM HEART RATE..... | 107 |
| FIGURE 3.6 – FREQUENCY SPECTRUM OF A 75BPM PHYSICAL MODEL WAVEFORM | 108 |
| FIGURE 3.7 – PHYSICAL MODEL VERSUS SIMULINK SIMULATION NORMALIZED FREQUENCY SPECTRUM FOR 1 HZ (60BPM) PULSE PATENCY MEASUREMENT | 109 |
| FIGURE 3.8 – PHYSICAL MODEL VERSUS SIMULINK SIMULATION NORMALIZED FREQUENCY SPECTRUM FOR 2 HZ (120BPM) PULSE PATENCY MEASUREMENT | 110 |
| FIGURE 3.9 – PHYSICAL MODEL VERSUS SIMULINK SIMULATION NORMALIZED FREQUENCY SPECTRUM FOR 3 HZ (180BPM) PULSE PATENCY MEASUREMENT | 110 |
| FIGURE 3.10 – PHYSICAL MODEL VERSUS SIMULINK SIMULATION NORMALIZED FREQUENCY SPECTRUM FOR 4 HZ (240BPM) PULSE PATENCY MEASUREMENT | 111 |
| FIGURE 3.11 – PHYSICAL MODEL VERSUS SIMULINK SIMULATION NORMALIZED FREQUENCY SPECTRUM FOR 5 HZ (300BPM) PULSE PATENCY MEASUREMENT | 111 |
| FIGURE 3.12 – SIMULATED 1 HZ PULSE PATENCY WITH PEAK-TO-PEAK VOLTAGE | 113 |

| | |
|--|-----|
| FIGURE 3.13 – MOVING AVERAGE ARRAY AVERAGE ACROSS ALL SIMULATED PULSE PATENCY WAVEFORMS | 114 |
| FIGURE 3.14 – MINIMUM PEAK AND MAXIMUM PEAK DERIVATIVES | 115 |
| FIGURE 3.15 – SELF INITIATING MOVING AVERAGE ARRAY AVERAGE PER DETECTED PULSE PATENCY BEAT | 115 |
| FIGURE 3.16 – 1 HZ PULSE PATENCY WAVEFORM GENERATED BY PHYSICAL MODEL | 117 |
| FIGURE 3.17 – MOVING AVERAGE ARRAY AVERAGE AT EACH DETECTED PEAK..... | 119 |
| FIGURE 3.18 – 5 HZ PULSE PATENCY PHYSICAL MODEL RESULTS FROM ARDUINO PROCESSING . | 121 |
| FIGURE 3.19 – 4 HZ PULSE PATENCY PHYSICAL MODEL RESULTS FROM ARDUINO PROCESSING . | 122 |
| FIGURE 3.20 – 3 HZ PULSE PATENCY PHYSICAL MODEL RESULTS FROM ARDUINO PROCESSING . | 123 |
| FIGURE 3.21 – 2 HZ PULSE PATENCY PHYSICAL MODEL RESULTS FROM ARDUINO PROCESSING . | 123 |
| FIGURE 3.22 – 1 HZ PULSE PATENCY PHYSICAL MODEL RESULTS FROM ARDUINO PROCESSING . | 124 |
| FIGURE 3.23 – SNAPSHOT OF PULSE PATENCY WAVEFORM FROM HUMAN SUBJECT | 125 |
| FIGURE 3.24 – ARDUINO RESPONSE TO ACCELEROMETER OUTPUT FROM HUMAN SUBJECT | 126 |
| FIGURE 3.25 – MOVING AVERAGE FOR TRIAL 1 | 128 |
| FIGURE 3.26 – MOVING AVERAGE FOR TRIAL 2 | 129 |
| FIGURE 3.27 – MOVING AVERAGE FOR TRIAL 3 | 129 |
| FIGURE 3.28 – MOVING AVERAGE FOR TRIAL 4 | 130 |
| FIGURE 3.29 – MOVING AVERAGE FOR TRIAL 5 | 130 |
| FIGURE 3.30 – ALARM ENGAGING AFTER SET TIME SINCE LAST PULSE DETECT | 132 |
| FIGURE 3.31 – SIMULATED 120BPM (2 Hz) PULSE PATENCY WAVEFORM | 133 |
| FIGURE 3.32 – INITIAL MOVING AVERAGE DATA FOR SIMULATED PULSE PATENCY WAVEFORM . | 134 |
| FIGURE 3.33 – SIMULATED PULSE PATENCY WITH LINEAR DECAY..... | 135 |
| FIGURE 3.34 – MOVING AVERAGE ARRAY RESPONSE TO PULSE PATENCY LOSS | 136 |
| FIGURE 3.35 – ENHANCED VIEW OF MOVING AVERAGE ARRAY RESPONSE | 136 |
| FIGURE 3.36 – SIMULATED LOSS OF PULSE PATENCY..... | 137 |
| FIGURE 3.37 – MOVING AVERAGE RESPONSE TO PULSE PATENCY LOSS | 137 |
| FIGURE 3.38 – ENHANCED VIEW OF PULSE PATENCY LOSS | 138 |
| FIGURE 3.39 – SIMULATION OF EXPONENTIAL PULSE PATENCY DECAY..... | 139 |

| | |
|--|-----|
| FIGURE 3.40 – MOVING AVERAGE ARRAY RESPONSE | 139 |
| FIGURE 3.41 – HUMAN MODEL PULSE PATENCY AND DETECTION OF PEAKS..... | 141 |
| FIGURE 3.42 – PULSE PATENCY SIGNAL WITH ADDED NOISE..... | 142 |
| FIGURE 3.43 – FOREHEAD POSITIONED BLOOD OXYGENATION RESPONSE..... | 144 |
| FIGURE 3.44 – STA POSITIONED BLOOD OXYGENATION RESPONSE..... | 144 |
| FIGURE 3.45 – POST-AURICULAR POSITIONED BLOOD OXYGENATION RESPONSE | 145 |
| FIGURE 3.46 – FOREHEAD PPG RESPONSE TO HEAD MOVEMENT | 145 |
| FIGURE 3.47 – STA PPG RESPONSE TO HEAD MOVEMENT | 146 |
| FIGURE 3.48 – POST-AURICULAR PPG RESPONSE TO HEAD MOVEMENT | 146 |
| FIGURE 3.49 – FOREHEAD RESPONSE TO OXYGEN DEPRIVATION | 147 |
| FIGURE 3.50 – STA RESPONSE TO OXYGEN DEPRIVATION | 147 |
| FIGURE 3.51 – POST-AURICULAR RESPONSE TO OXYGEN DEPRIVATION | 147 |

LIST OF TABLES

| | |
|--|-----|
| TABLE 2.1 – VALUES FOR FOURTH ORDER SALLEN-KEY HIGH PASS FILTER..... | 43 |
| TABLE 2.2 – VALUE FOR FOURTH ORDER SALLEN-KEY LOW PASS FILTER..... | 52 |
| TABLE 2.3 – CALCULATED VALUES FOR DC LEVEL SHIFT CIRCUIT | 55 |
| TABLE 3.1 – RESULTS FROM ALGORITHM VERIFICATION | 116 |
| TABLE 3.2 – PHYSICAL MODEL OUTPUT ANALYZED BY ALGORITHM | 119 |
| TABLE 3.3 – PERCENT ERROR OF RESULTS FROM TABLE 3.2..... | 120 |
| TABLE 3.4 – FIVE TRIALS OF MODEL RESULTS..... | 127 |
| TABLE 3.5 – PERCENT ERROR OF RESULTS FROM TABLE 3.4..... | 128 |

ABSTRACT

PULSE PATENCY AND OXYGENATION SENSING SYSTEM DEVELOPMENT TO DETECT G-INDUCED LOSS OF CONSCIOUSNESS

by

Herman Pretorius, BSEE

University of New Hampshire, May, 2013

A fighter pilots greatest strength is the weakness of his or her opponent. Commonly, this strength comes down to the maneuverability of the aircraft, particularly the ability to out-climb. Since the 1980's, the thrust produced by these engines have the ability to drain the pilots head of blood causing a state of unconsciousness due to the overwhelming forces of gravity for upwards of 30 seconds; often times having fatal outcomes.

This thesis explores the feasibility of detecting of blood flow by means of arterial wall expansion (pulse patency) and blood oxygenation using a microprocessor to continually monitor the signals from this two part sensor where by insight into the development of a g-induced loss of consciousness sensing system can be developed. Results indicate greater than 90% accuracy pulse patency detection using an accelerometer. Simulation and physical models were used as well as human testing to develop a blood oxygenation and pulse patency sensor, or BOPS.

INTRODUCTION

Human physiology continues to be the hindering factor in aircraft performance. Each year, fighter aircraft become faster and more agile; however, these advancements are hindered by the pilots' physiology [1]. As a pilot conducts maneuvers, his or her body is subjected to inherent gravitational forces (G-Force). G-induced loss of consciousness (G-LOC), an outcome of high +Gz (Figure 1) maneuvers, has the potential of causing an incapacitation state in pilots for an average of *24 seconds* followed by an undefined period of time where the pilot suffers from impaired judgment and fine motor skills [2] [3]. Countermeasures, such as Anti-G suits, Anti-G Straining Maneuvers (AGSM), and seat positioning, have allowed pilots to push the limits of acceleration up to 9G's; however, these countermeasures only increase the pilot's G-tolerance but do not prevent the onset of G-LOC [4]. Because G-LOC is a transient functional state, its role in fatal crashes is often hard to ascertain [1].

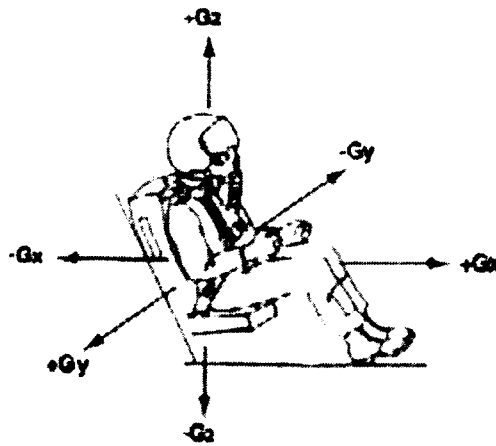


Fig.1: Vectorial depiction of gravitational forces on pilot [1]

Acceleration in the +Gz direction causes blood to flow from the upper extremities to the lower extremities, or head-to-foot direction. In the past 30 years, there has been significant research done in the development of physiological sensors to detect this effect on the human body and is primarily focused on the development of non-invasive sensors so as not to obstruct or hinder the pilot during flight.

Various approaches to monitor loss of consciousness were proposed and the concept system was given a name of Loss of Consciousness Monitoring System, or LOCOMS. These methods included eye blink rate, head slumping, strength or absence of grip on stick, head level arterial pulsations, eye level blood pressure, arterial oxygen saturation percentage, pulse pressure, spectral shift in frequency of electroencephalogram (EEG), and pilot response to voice synthesized interrogation [5]. The original development of the LOCOMS was to incorporate all of the physiological signals into an "Expert System" that would assume control of the

airplane given a G-LOC condition of the pilot. [5]. As the primary goal is reliability, the chance of all of these physiological signals indicating an error would be highly unlikely.

Given the numerous influences that could increase eye blink rate, head slumping, or stick grip, those methods of monitoring G-LOC were abandoned. The EEG yielded the most reliable results, however, the difficulty of reliably and unobtrusively obtaining the correct signals caused research to approach the problem differently [6]. The different approach was to monitor arterial pulsations, pulse pressure at eye level, and arterial oxygen saturation. These three physiological responses, if monitored accurately, are strongly correlated with the physiological symptoms of pilots during G-LOC and are highly accessible for non-invasive monitoring.

E.H. Wood observed that after 5-7 seconds, given a sustained acceleration, a loss of circulatory pulsations at ear level occurs [5]. In a study conducted with baboons, it was shown that cerebral blood flow always decreased rapidly to zero *prior* to G-LOC [7]. In an attempt to measure cerebral blood flow, plethysmography (PPG) data was taken at two different locations: the ear lobe and the nose [8]. Both of these methods showed correlations between G-LOC and blood loss. However, both systems required pilots don the instrumentation on their being; therefore, not achieving the goal of creating an unobtrusive sensor.

A prominent artery in the forehead, the Superficial Temporal Artery (STA), is not only highly accessible, but is an ideal target since it directly branches off the

carotid artery [9]. David Sarnoff Laboratories developed an array of ultra-sensitive antennae to detect arterial wall expansion, known as pulse patency [10]. The system yielded promising results, but was highly sensitive to motion artifact, such as helmet movement [5]. A similar system was designed known as the Pulse Pressure Monitoring System (P3). The P3 system was constructed similarly but used an array of piezo electric benders rather than resonating antennas. In an attempt to remove motion artifact, the piezo array was rigidly constructed. It was fully capable of detecting pulse patency but lacked resolution and sensitivity at high +Gz levels where pulse patency measurements are low [11].

Various studies using pulse oximeters have proved their ability to detect G-LOC. When testing the nose mounted blood oximeter, the studies showed an accurate tracking of SpO₂ which showed “significant drop offs at 7G’s” [8]. In a study of spatial disorientation in pilots, results show immediate drop-offs of SpO₂ during centrifuge testing [12]. The Naval Aerospace Medical Research Laboratory used reflectance pulse oximeters positioned on the forehead to monitor SpO₂. Reflectance pulse oximeters proved to be faster than finger transmission pulse oximeters to rapid changes in SpO₂. However, given the placement of the oximeter on the forehead, claims were made that major improvements in diminishing noise, curbing motion artifact, and improving reliability are needed [13].

A primary goal of this thesis is to develop a pulse patency sensor that has high sensitivity while maintaining a high level of reliability. It is envisioned that an accelerometer placed over the STA will meet the conditions sensitivity and

reliability while, at the same time, be non-invasive and unobtrusive to the pilot during flight. Three-axis accelerometers have adjustable sensitivity that will allow optimum calibration of the sensor. To optimize the sensitivity and reliability, the sensor will be simulated and modeled using modern engineering tools such as MATLAB and various Data Acquisition units (DAQ's). The ideal goal is to reach 100% accuracy given outside factors such as motion artifact while being able to measure pulse patency down to zero. However, as pulse patency approaches zero, the signal-to-noise ratio decrease causes a decrease in reliability. Therefore, appropriate algorithms will be used to ensure highest levels of calibration.

A secondary goal of the thesis is to effectively detect and analyze blood oxygenation saturation level, SpO₂ in various locations to determine the best location for reliability and reduction of motion artifact. Since SpO₂ drops off significantly, it won't be a focus to detect the blood oxygenation with high precision, but in a location where there exists a minimization of noise. One area that will be considered is the post-auricular region. The posterior auricular artery is a small branch directly off the external carotid artery that may provide improved insight to carotid blood flow than a probe on the forehead [9].

A G-LOC detection sensor must have the ability to sense the onset of cerebral hypoxia quickly and provide the appropriate feedback for pilot decision-making. Algorithms for both technologies will be developed in order to minimize the effect of noise and motion artifact, while attempting to detect an onset of hypoxia within 6-7 seconds [3]. Too many false alarms would eventually train the aircrew to ignore the

warning, rendering the system all but useless, while insufficient sensitivity may cost the military valuable lives and assets [13].

In order to verify the system design and algorithms, the sensor was tested in numerous methods. First, the algorithms will undergo a simulation process. Using simulation, the algorithms were verified before trying to diagnose problems during testing. Since pulse patency is dependent on acceleration, the simulation will be able to vary the heart rate while simulating loss of patency using three different windows. The first and second windows will be a linear and exponential decay window, which will allow for the simulation of a continuous onset of +Gz forces. Since G-tolerance is different for every individual, there is no direct correlation between how pulse patency decays and at what rate. These windows will be fully adjustable so that the user can vary the total pulse patency force loss (vertical scale) and the onset of +Gz forces (horizontal scale), which is inversely related to the pulse patency force (the slope or exponential decay rate).

After the system was verified in simulation, a physical model was developed to simulate pulse patency. The physical model provided a better insight into actual system and algorithm performance in real time. This model was constructed using air valves, tubing, and the accelerometer. By varying the air pressure (PSI) being pushed through the tubing, +Gz acceleration can be simulated.

Finally, the human element was tested. Once the algorithms are calibrated during simulation and the physical model, it was tested on a human subject. This will be the final step in the thesis. Accuracy analysis will be done along with a

motion artifact analysis. Due to the many complexities of the human body, this portion of the thesis will be a feasibility study of the implantation of such a sensor. The results of this thesis will guide future research into the advantages and disadvantages of using such technologies to monitor G-LOC.

Chapter 1 of this thesis is the background information needed to understand the complex issue of G-LOC. This chapter will include a brief history of G-LOC along with a view into the physiological aspects of G-LOC. This chapter will include further review of previous works that have been briefly described.

Chapter 2 of the thesis will be a description and theory of the methods used to conduct the research of the thesis. This will include the tools used for the simulation, physical modeling, and the human testing. This will also include insight into the oximeters used and the methods used for analysis on proper placement and analysis of the oximeters.

Chapter 3 will be an in-depth report on the testing and results conducted for both the pulse patency and blood oxygenation sensor placement. Results will include verification of simulation and physical model techniques along with accuracy analysis of algorithm detection. This chapter will also be an analysis of the placement of the oximeters. The results in this chapter will indicate where the oximeter can be placed to accurately monitor pilot SpO₂. Insight into motion artifact at different locations will be analyzed along with its effect on proper oxygenation readings.

Chapter 4 will be the conclusion of the thesis to include insight and recommendations for future research. Final analysis of the results will be reviewed here. The advantages and disadvantages of such a system will be discussed along with potential methods that would resolve some of the disadvantages.

CHAPTER 1:

G-INDUCED LOSS OF CONSCIOUSNESS

A Historical Problem

Standing on top of a sand dune in Kitty Hawk, North Carolina on December 17th, 1903, two brothers were guessing heads or tails. The winner of this coin toss would be the first to climb into a machine that would have an enormous impact in history. Orville and Wilbur Wright, owners of bicycle shop, astonished spectators as they flew upwards of 59 seconds that day in the Wright Flyer, a 12 horsepower push prop biplane made of light wood and sail [14] [15]. What they knew was that they changed the course of mankind, but what they couldn't have imagined is what our present day technology has been able to do with their dream.

Soon after Orville and Wilbur had their first flight, the governments of other countries started the production of their first airplanes. It wasn't long after the Wright brothers' flight that World War I started when the military took the concept of flight and produced spy and bomber planes using push or pull propeller aircraft. In 1939, during the early years of World War II, the first functional jet airplane took flight by a German pilot named Erich Warsitz. Due to the declining war industry in

Germany and the lack of experienced pilots, the first functional jet fighter wasn't introduced until 1942 [16].

As these pilots were conducting flight maneuvers, regardless of how basic, their bodies are subject to gravitational forces. Of all the three dimensions that gravity can exert a force, forces in the +Gz direction (head to foot) bring about the greatest concern. Too great of acceleration or too long of an acceleration in the +Gz direction could cause G-Induced Loss of Consciousness (GLOC) [17]. Pilots started reporting symptoms of GLOC as early as World War I as they were pulling away from earth after bombing runs [18].

Human physiology has been the limiting factor in aircraft since the introduction of flight due to symptoms such as fatigue, hypoxia, and disorientation. It wasn't until the fourth generation of aircraft (the 1980's) that human physiology became an operationally significant limiting factor. Aircrafts, with greater thrust and wing load capacity, were able to sustain accelerations of 9G's or more. A "G" is a reference to the amount of gravitational strain that is applied to the body. At 9G, for example, the force of gravity is 9 times that of the standing person on the surface of earth. Even though these levels of performance can be dangerous for pilots, they are also crucial for their survival. The key to survival during a "dogfight," a common term used in reference to air-to-air combat, is preying on the weakness of the opposing aircraft. Therefore, a powerful engine on an aircraft that will allow it to out-maneuver or climb its opponent may be its only key to survival [1].

Although this has proven necessary for survival in many cases, GLOC has been the documented cause of many crashes since the 1980's. Between 1982 and 1998, the U.S. Air Force reported a total of 26 lost aircraft due to GLOC, and a total of 402 reported episodes of GLOC between 1985 and 1998 [2]. Since GLOC is a transient state and currently undetectable, the documented number of accidents involving GLOC could certainly be much more! For example, evidence is available that indicates that there is a 33% chance of fatality and/or loss of aircraft if a student pilot experience a GLOC event [19].

Model of Hydrostatic Pressure and Human Physiology

As pilots pull back on the stick, which raise the nose of the aircraft, the aircraft and pilot are now subjected to +Gz forces. The harder the pilot pulls back on the controls, the +Gz acceleration increases. As the acceleration continues in the +Gz direction, the vector force produced by gravity becomes increasingly greater than the force produced by the heart in order to pump blood throughout the body causing the blood to pool in the lower extremities (Figure 1.1) [17].

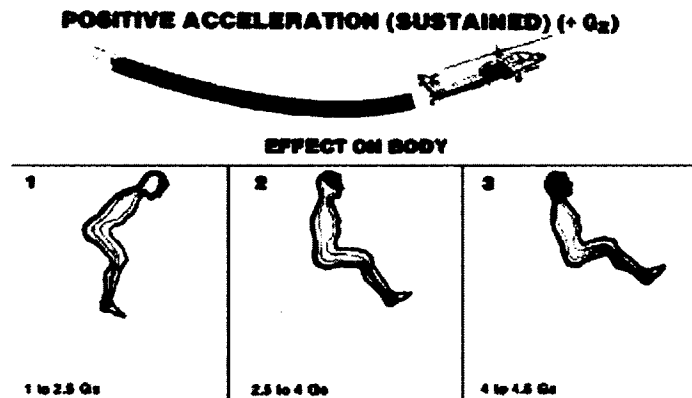


Figure 1.1: Physiological Effect of +G_z acceleration

The simplest way to understand basic cardiovascular effects of G forces is for the human body can be modeled as a hydrostatic column. Within the hydrostatic column, the important parameters are the height of the column, the pressure within the column, and the density of the fluid being affected [20]. The hydrostatic pressure at any given point can be modeled by Equation 1.1.

$$P = h * d * G$$

Equation 1.1: Hydrostatic Column equation

Where P is the pressure in mmHg, h is the height of the column in mm, d is the density of the fluid in Hg, and G is the gravitational force. The average distance between the heart and the brain is approximately 340mm [20]. The density of blood with respect to mercury (Hg) is $\frac{1}{13.6}$ [20].

When an individual is lying flat against the ground, perpendicular to the gravitational vector of earth, the difference in pressure between the head and the

heart is zero. However, if standing parallel to that force, the resulting difference between the head and the heart would be:

$$P = (340mm) * \left(\frac{1}{13.6} Hg\right) * 1G = 25mmHg$$

Equation 1.2: Values substituted into Equation 1.1

Assuming average systolic blood pressure of 120mmHg, the resulting brain level blood pressure would be 95mmHg. In a study done by Cammarota, the blood pressure at head level of the pilot seated at 1G was 96mmHg [21]. Therefore, at a force of +5Gz, the hydrostatic pressure is equal to 125mmHg, which exceeds the blood pressure at heart level, and the lack of blood at head level will cause unconsciousness. For the readers information, all blood pressures talked about throughout the entirety of the thesis will focus on systolic blood pressure, and not diastolic.

Intraocular Pressure and Visual Symptoms.

It is common for pilots to experience visual symptoms prior to GLOC. The average intraocular blood pressure is between 12 and 20mmHg [22]. The intraocular blood pressure is the blood pressure that the retinal and ophthalmic artery must overcome in order to supply oxygen to the eye [20]. Assuming a head level blood pressure of 95mmHg:

$$\frac{95\text{mmHg} - 15\text{mmHg}}{25\text{mmHg}/G} \cong 4G's$$

Equation 1.3: Simple G-tolerance calculation

A pilot not wearing any G-preventive equipment will experience visual symptoms at a max of 4G's depending on G-tolerance followed by blackout with a further of +1Gz, Figure 1.2 [1]. Typically the fighter pilot will go through various stages of visual symptoms with +Gz onset starting with a gray out, dimming of visual fields, and finally, blackout [2] [17]. The loss of vision does not mean that the pilot is incapacitated, however. If the pilot sustains the acceleration, he or she will be subject to GLOC. With faster +Gz onset, visual symptoms can occur at lower acceleration. If the onset is fast enough, the pilot will become incapacitated without any visual warning.

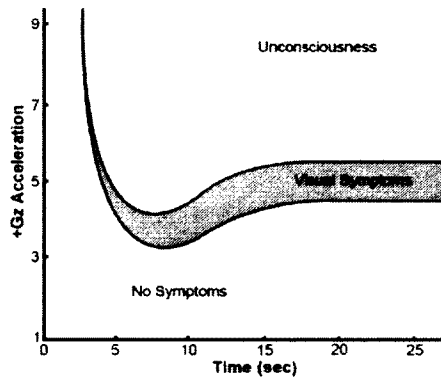


Figure 1.2: +Gz acceleration versus time and typical physiological symptoms

GLOC Prevention

In order to combat the effects of +Gz acceleration, certain technologies and techniques have been developed in order to increase the G-tolerance of the pilot. These developments to increase the G-tolerance of pilots include anti-G suits, semi-reclining seats, and voluntary straining maneuvers accompanied by positive pressure breathing [1].

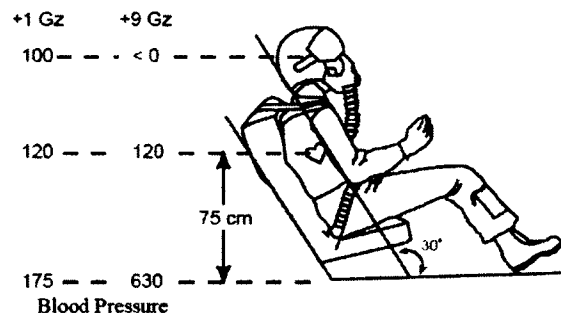


Figure 1.3: Effect of reclined seat on pilot blood pressure

Seat positioning is a simple, yet effective way of increasing a pilots' G-tolerance. By reclining the seat of the aircraft by 30 degrees, the distance between the heart and the head is decreased, as illustrated in Figure 1.3. Referencing to Equation 1.1, this minimizes the differential blood pressure between the heart and the head. The reclined seating position in the aircraft can contribute to a 0.5 to 0.75 increase in G-tolerance among average aircrew members. In a past study, the G-tolerance of an average aircrew member in a reclined F-16 seat was 5.2G's, an increase from the standard 4G to 4.5G tolerance [2].

The anti-G suit was developed during World War II and changed little during the following 70 years. The anti-G suit is constructed of two layers of non-stretch material with rubber bladders at the abdomen, calf, and thigh levels [1]. As +Gz acceleration increases, these bladders pressurize and assist in keeping the blood in the upper extremities. The anti-G suit can add another 1G to the pilots' tolerance.

During pilot training, pilots are exposed and extensively trained on individual GLOC anticipation and prevention. One of techniques taught throughout training is the anti-G straining maneuver (AGSM). The AGSM consists of numerous steps that the pilot performs during +Gz maneuvers. These include the following factors: anticipating the G, tensing all lower body muscle groups, breathe holding, interval breathing at peak G, and not relaxing until the G is unloaded [2]. A good AGSM can add another 3.5G's to the pilots' tolerance.

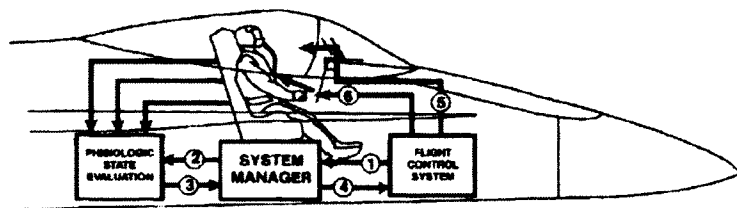
With all of these totaled, one can see that a 9G maneuver is a big challenge for most aircrew. Even though a pilot may be able to conduct such a maneuver, there is little to no safety margin at such acceleration.

GLOC Detection System: Introduction

Although the GLOC prevention technologies and techniques allow pilots to sustain some of the maneuvers from the aircraft, it doesn't prevent GLOC, it only increases the G-tolerance of the pilot. With GLOC incidents still occurring today and the evermore increasing costs of aircrew and aircraft, GLOC detection systems are

just as important now as ever. In order to save the pilot and the aircraft from loss due to GLOC, a system needs to be developed in order to detect the onset of GLOC and initiate an auto-recover system, such as autopilot) upon accurate detection of GLOC. The auto-recovery system would need a very important characteristic: reliability. Too many false alarms would eventually train the aircrew to ignore the warnings rendering the system useless. Conversely, a lack of sensitivity in the system may cost the military lives and assets [13].

Similar to the anti-G suit, a method of predicting GLOC was proposed based off of +Gz acceleration data. Using this data, a computer would determine that the acceleration profile of the aircraft has exceeded that of the “average” pilot, and would initiate an auto-recovery system. This system was least encumbering to the pilot due to the lack of biomedical sensors needed. However, due to the varying G-tolerance profile of each individual pilot, this auto-recovery system is likely to prohibit the development of highly reliably monitoring based solely on G-time history [3].



1. G-TIME HISTORY, AIRCRAFT STATE
2. +Gz STRESSOR METRIC
3. PHYSIOLOGIC RESPONSE TO STRESSOR
4. PILOT STATE OF CONSCIOUSNESS
5. G-LIMIT / AUTORECOVERY WARNINGS
6. AIRCREW RECOVERY STIMULI

Figure 1.4: Sequential GLOC Detection System [3]

Cammarota et al. determined that, in order to ensure utmost reliability, numerous biomedical sensors needed to be implemented. These sensors, working together with an acceleration profile monitor, provide the best hope for detecting GLOC (Figure 1.4) [3]. The idea of using multiple sensors is to minimize the chance of one “false positive” sensor assuming control of the aircraft, causing distrust on behalf of the pilot. Cammarota et al. proposed a system called “G-WARS”, a title based off of the tactic of using your opponent’s disadvantage, +Gz acceleration, as your advantage (Figure 1.5). A similar system was proposed in 2003, but it included items such as ground avoidance, pilot ejection, and auto-pilot algorithms [23].

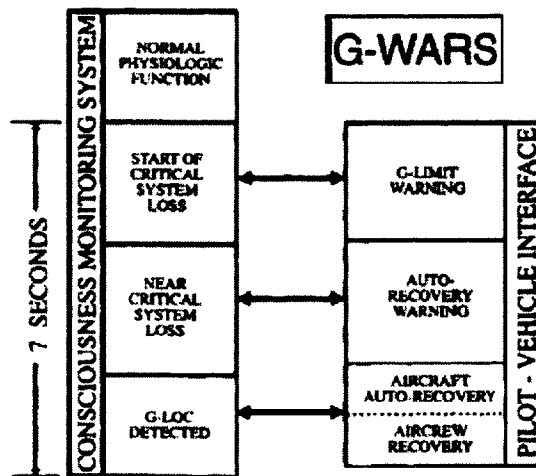


Figure 1.5: G-WARS auto-recovery system [3]

All of these systems allowed the pilot to resume control of the aircraft or delay the auto-recovery feature. As the pilot approached GLOC, the auto-recovery system would interrogate the pilot via voice interrogation, granting the pilot the

opportunity to maintain control of the aircraft [5]. This was due to several reasons. First, since pilots' G-tolerance varies, certain pilots may be able to push the aircraft closer to its full envelope than others. However, the primary reason for this, and a very important point, is the "limit." The "limit" refers to the point at which the auto recovery system assumes control of the aircraft. Based off of visual symptoms, if the auto-recovery feature was to engage when the pilot's vision was at gray out, the first stage of visual symptoms, the aircraft would never be able to fly close to its performance envelope [3]. This would not only be an enormous waste of research costs and resources, but it would pose a serious hazard to pilots trying to avoid enemy aircraft.

Loss of Conscious Monitoring Systems

The loss of conscious monitoring system, or LOCOMS, can be considered part of the overall G-WARS proposal. LOCOMS was an effort at the Armstrong Aerospace Medical Research Laboratory (AAMRL) as an initial approach to monitoring the pilots GLOC state [5]. LOCOMS was designed to monitor several physiological signals, including: eye blink rate, head slumping, strength or absence of grip on stick, head level arterial pulsations, eye level blood pressure, arterial oxygen saturation percentage, spectral shift in electroencephalogram (EEG), presence and quality of AGSM, and pilot response to voice synthesized interrogation. It also included aircraft quantitative data such as anti-G suit functionality, +Gz exposure time above 4G's, onset rate of exposure, and duration of exposure [5].

Research was conducted based off of the LOCOMS model of GLOC monitoring. Many proposed methods were retired early for various reasons. Eye blink rate, head slumping, and strength or absence of grip on stick were determined to be unreliable due to there being several reasons excluding GLOC for any of these symptoms to occur. All of these methods relied on the muscle flaccidity that occurs during an unconscious state.

The EEG was, according to AAMRL, the most reliable indicator of GLOC [5]. This method relied on unobtrusively monitoring brain signals and detecting a spectral shift with the onset of GLOC. At the U.S. Air Force School of Aerospace Medicine, a researcher by the name of Lewis observed the same shift to lower frequencies (Delta or 0.5 to 4.0Hz) occurred with subjects under GLOC to patients that are unconscious due to anesthesia [6]. The issue, however, was successful monitoring of the EEG. The overwhelming complexity of the brain makes unobtrusive monitoring of signals a daunting task. Government small business innovation research grants (SBIR) were, as late as 2005, being approved to develop reliable optical sensors for EEG monitoring [24].

The final physiological data not yet addressed from LOCOMS are: head level arterial pulsations, eye level blood pressure, and arterial oxygen saturation percentage. Blood pressure is defined as “the pressure exerted by the blood on the inner walls of the arteries, being relative to the elasticity and the diameter of the vessels and the force of the heartbeat” [25]. In order to fully measure blood pressure at eye level, the pilot would need to don a transducer and some form of tourniquet

[26]. This method would inherently restrict blood flow to the head causing a lower G-tolerance in the pilot.

Superficial Temporal Artery

The superficial temporal artery (STA) is a prominent artery in the head. As seen in Figure 1.6, the STA branches off from the external carotid artery. At ear level, the STA branches off into the frontal and parietal branches, which supply blood to the front and rear portions of the head, respectively. The STA is highly researched due, not only to its accessibility, but also due to the frontal branch being a blood oxygen supply to the eye.

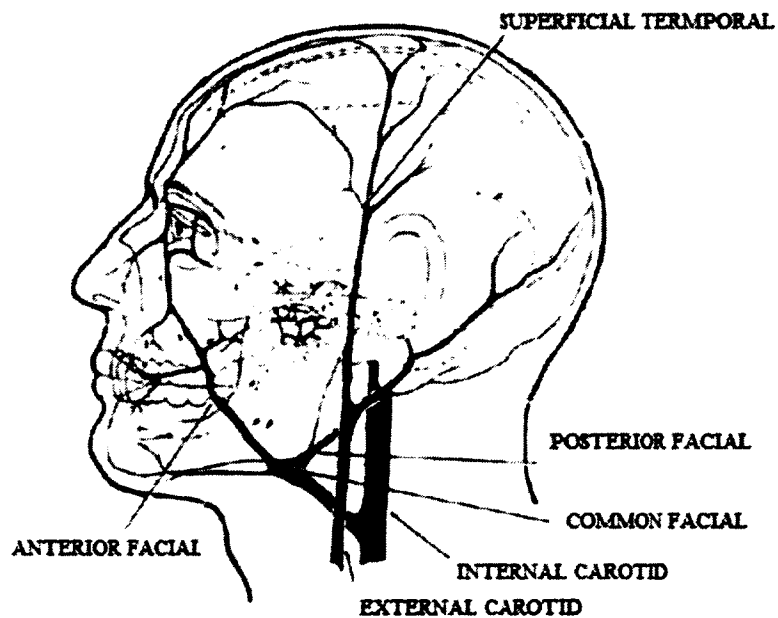


Figure 1.6: Head level human physiology (Source: www.armymedical.tpub.com)

Plethysmography and Pulse Patency

An accurate pulse patency or plethysmography measurement will yield results indicating the presence of blood at the point of measurement. The time difference of 5-8 seconds between loss of blood flow and GLOC allows the system enough time to process the data, initiate voice interrogation and potentially initiate an auto-recovery system [27]. Both pulse patency and plethysmography correlate with GLOC and blood pressure, but have suffered development issues. As blood pressure decreases due to sustained +Gz acceleration, so will the presence of blood in the upper extremities and the force of arterial wall expansion [11].

Plethysmography is the measurement of changes of volume of organs particularly changes resulting from blood flow [28]. The plethysmograph can be acquired in several methods. First, a simple pulse oximeter acquires blood oxygenation. A reading, of any kind, from a pulse oximeter would suggest the presence of blood. Pulse oximeters, mentioned later in the chapter, are highly susceptible to noise and/or would require the pilot to don a sensor. Secondly, there are several circuits that are standard blocks for simple plethysmograph readings [29]. These circuits require highly accurate positioning of the instrumentation and with minimal highly accessible arteries at head level. Using these circuits for plethysmograph would require the pilot to don the instrumentation and yield the potential of human error.

Pulse patency is a measurement of the expansion and contraction of vessel walls due to the flow of blood. Pulse patency has been an option explored since the

mid-1980's. David Sarnoff Laboratories, in 1986, constructed an array of dielectric patch antennas to monitor the pulse patency of the individual wearing the helmet on the occipital branch of the superficial temporal artery [5]. Their design was implemented in a helmet and tested. The results showed detection of arterial wall expansion, but they were also highly sensitive to helmet movement (motion artifact).

In order to counteract the motion artifact, LaCourse et al. designed a similar system called the Pulse Pressure Monitoring System (P3) using a piezo-electric bender array [11]. The system was protected from outside noise via a plastic casing that isolated the piezo array from outside forces (i.e. helmet pressure). This allowed for precise STA monitoring without the influence of helmet movement. However, due to the minimum force required to bend the piezo-electric bender, the system had low resolution at high +Gz acceleration. As discussed previously, as +Gz acceleration increases, eye level blood pressure decreases at a rate of 25mmHg/G. Pulse patency will also decrease with the onset of +Gz forces due to less blood flowing through the STA. Eventually, the force produced by the arterial wall expansion will not be able to overcome the force required to actuate the piezo-electric bender. Therefore, at high +Gz acceleration, the P3 system would suggest that blood flow has stopped prior to it actually stopping, causing unreliability.

Blood Oxygenation

As blood flow to the head decreases as +Gz acceleration increases, regional blood oxygenation, rSO₂, also decreases [12]. A study done by Shender et al. shows a significant drop in rSO₂ values with the onset of +Gz acceleration, (Figure 1.8). Tripp et al. tested subjects in a centrifuge and monitored rSO₂ values with the onset of different stages of GLOC (Figure 1.9). The two studies show a very similar response to tissue oxygenation as a function of +Gz acceleration and/or stages of GLOC. Results indicate a minimum drop to 85% of the baseline value with the onset of GLOC [19] [30].

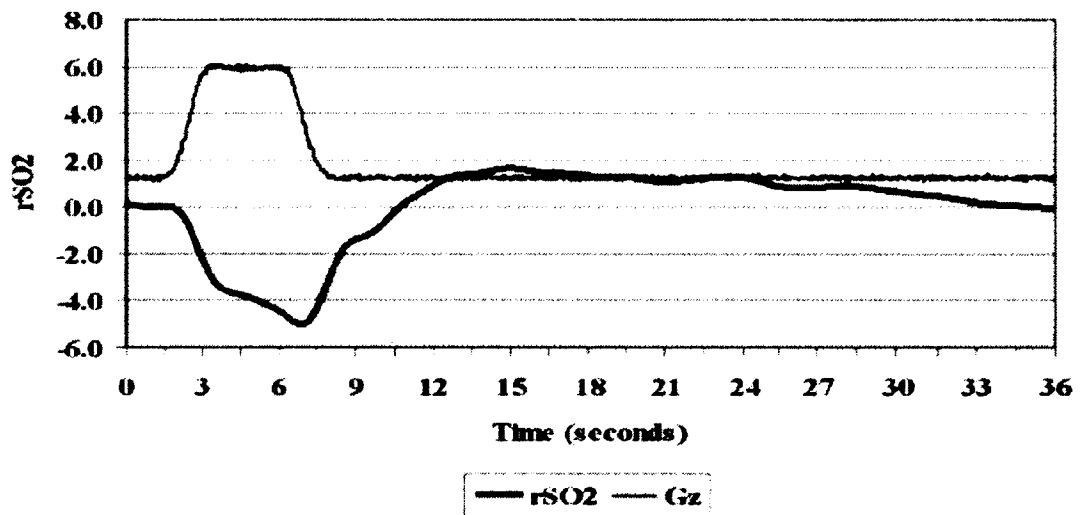


Figure 1.8: Shender et al.: rSO₂ response with the onset of +Gz acceleration [12]

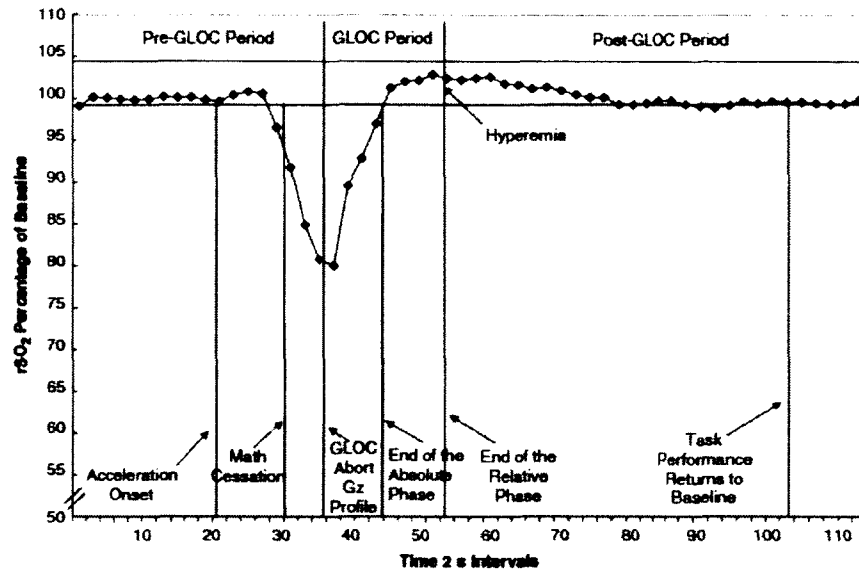


Figure 1.9: Tripp et al. results showing regional tissue oxygenation as a function of GLOC stages [19]

The use of pulse oximeters to monitor blood oxygenation was originally part of the LOCOMS system. Tripp et al. (1987) integrated a nasal blood oxygenation monitor into the facemask of a pilot's oxygen mask [8]. Similarly, Melker et al. patented a nasal blood oxygenation monitor in 2008 that was slightly less obtrusive to the pilot (Figure 1.10) [9]. However, both of these methods required the pilot to don the sensor at all times providing an inconvenience to the pilot.

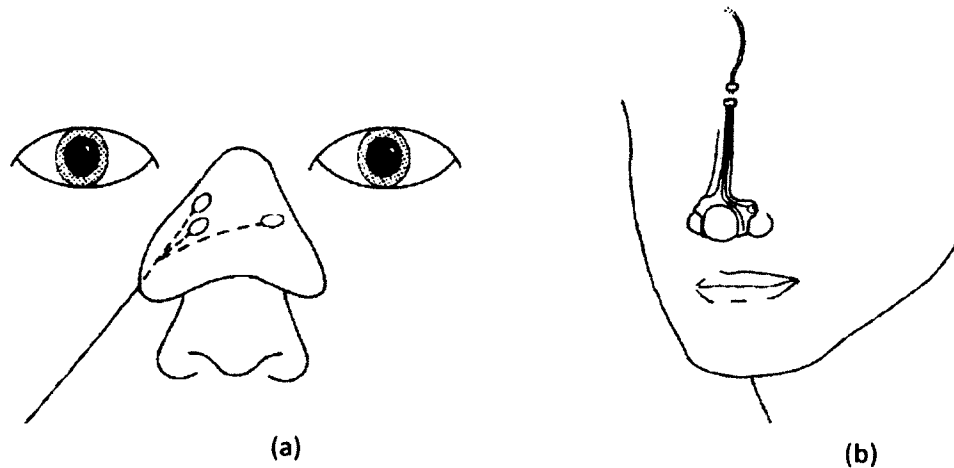


Figure 1.10: (a) Tripp et al. Nose Oxygenation Monitor [8] (b) Melker et al. Nose Oxygenation Monitor [9]

These studies also relate to rotary wing pilots. It is not uncommon for rotary wing pilots, especially in regions of the world that are mountainous or high in elevation, to achieve altitudes of 15000 feet or more [13]. Without any oxygen mask support, these pilots are also susceptible to cerebral hypoxia. The Naval Aerospace Medical Research Laboratory tested the Nonin OEM III reflective pulse oximeters on pilots under simulated altitude exposures in 2010 [13]. It is important to note that none of these pilots were exposed to +Gz acceleration. Their results yielded a strong correlation between finger pulse oximetry and the Nonin OEM III reflective pulse oximeter placed on the forehead ($r = .96$) [13]. However, Simmons et al. reported a high susceptibility to noise and motion artifact.

Thesis Hypothesis

In order to overcome the resolution at high +Gz acceleration, it is hypothesized that the use of an accelerometer will produce more accurate results (Figure 1.11). Three-axis accelerometers have been advanced in recent years. Accelerometers have become very common in technologies such as smart phones and tablets. Their small size and adjustable sensitivity allow for calibration of an accurate pulse patency measurement tool while considering the overall weight of the sensor due to the high G atmosphere. The methods used to select, test, and process the data from the accelerometer is further discussed in Chapter 2.

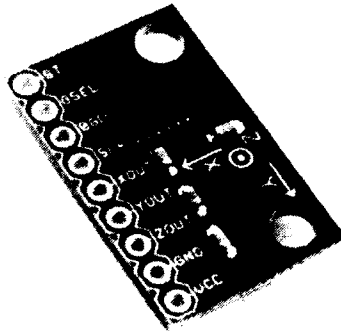


Figure 1.11: MMA7361 Three-axis accelerometer (Source: www.sparkfun.com)

In this thesis, the Nonin OEM III will also be used to test blood oxygenation. In Chapter 2 the methods of testing the pulse oximeter will be discussed. The focus of the testing will be minimization of noise and motion artifact. It is hypothesized that the same, or similar results can be obtained by the Nonin OEMIII pulse oximeter at different locations, such as the post-auricular region, on the head since rSO_2

decreases by a minimum of 15% of its baseline value with the onset of GLOC, the “best” location will be analyzed and tested.

System Design: BOPS

The design and development of a prototype system will be presented. The system will incorporate an accelerometer to measure pulse patency and a Nonin OEMIII blood oxygenation sensor to measure spot oxygen level, or SpO₂, in the head. It is important to note a few observations of this study. It is not required to monitor every single pulse 100% correctly, every time. Since blood flow stops and GLOC ensues 5-8 seconds later, there is time for a computer to resample, analyze all physiological signals, and initiate voice interrogations. Both of the signals, pulse patency and blood oxygenation will be sampled continuously. With the use of moving average algorithms, the chance of a false positive causing an unreliable system is diminished.

The design and calibration of this sensor will be referred to as the Blood Oxygenation and Patency Sensor, or BOPS.

This is a feasibility study of the integration of existing equipment to provide a potential solution to the problem and the results should be taken as such. The hope of the thesis is that it will serve as future guidance to advancement in the system design and enhance the chances of solving a 30 year old problem.

CHAPTER 2:

METHODS

Introduction

Chapter 2 describes the design for the development of the blood oxygenation and pulse patency sensor, or BOPS (Figure 2.1). This chapter explains thoroughly the design of BOPS, including items such as selection of all electronic components, the microprocessor selection, and design algorithms. The overall system includes a MM7361L triple-axis accelerometer to measure pulse patency and a Nonin OEM III blood oximeter to determine blood oxygenation. Both signals are processed through an Arduino UNO R3 microprocessor.

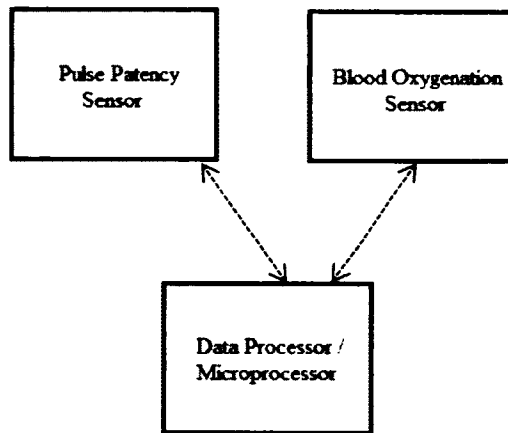


Figure 2.1: Overall System Block Diagram

Pulse Patency

The pulse patency measurement was a three-step design process. The major unknown component of the pulse patency measurement is the microprocessor algorithm. The order of succession of the three-step process is as follows: simulation, physical modeling, and human testing. Since the accelerometer is intrinsically responsive to motion, it is important to develop an algorithm to detect the correct peaks and be able to monitor (or count) the peaks in order to determine existing blood flow in the STA.

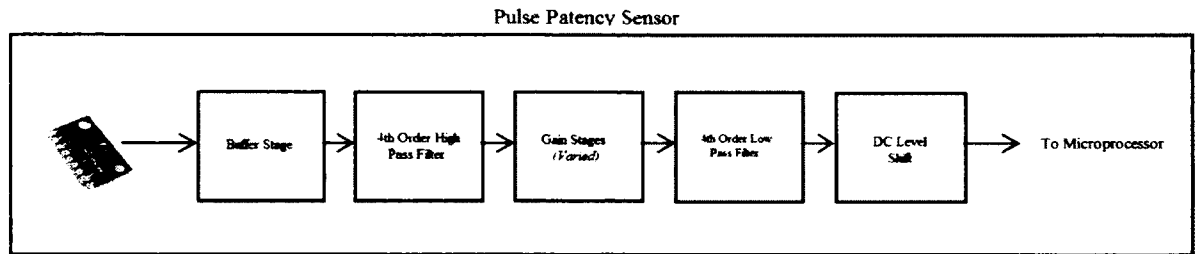


Figure 2.2: Block Diagram of Pulse Patency Sensor

In Figure 2.2, the overall block diagram of the Pulse Patency Sensor is presented. The sensor includes five main blocks to ensure a noise-free signal that accurately depicts the accelerometer's acceleration. The resulting signal is processed at the microprocessor. These blocks includes: the accelerometer, a buffering stage, a high pass filter stage, varying gain stages, a low pass filter, and a DC level shifter. Finally, the overall system circuit and frequency response is shown.

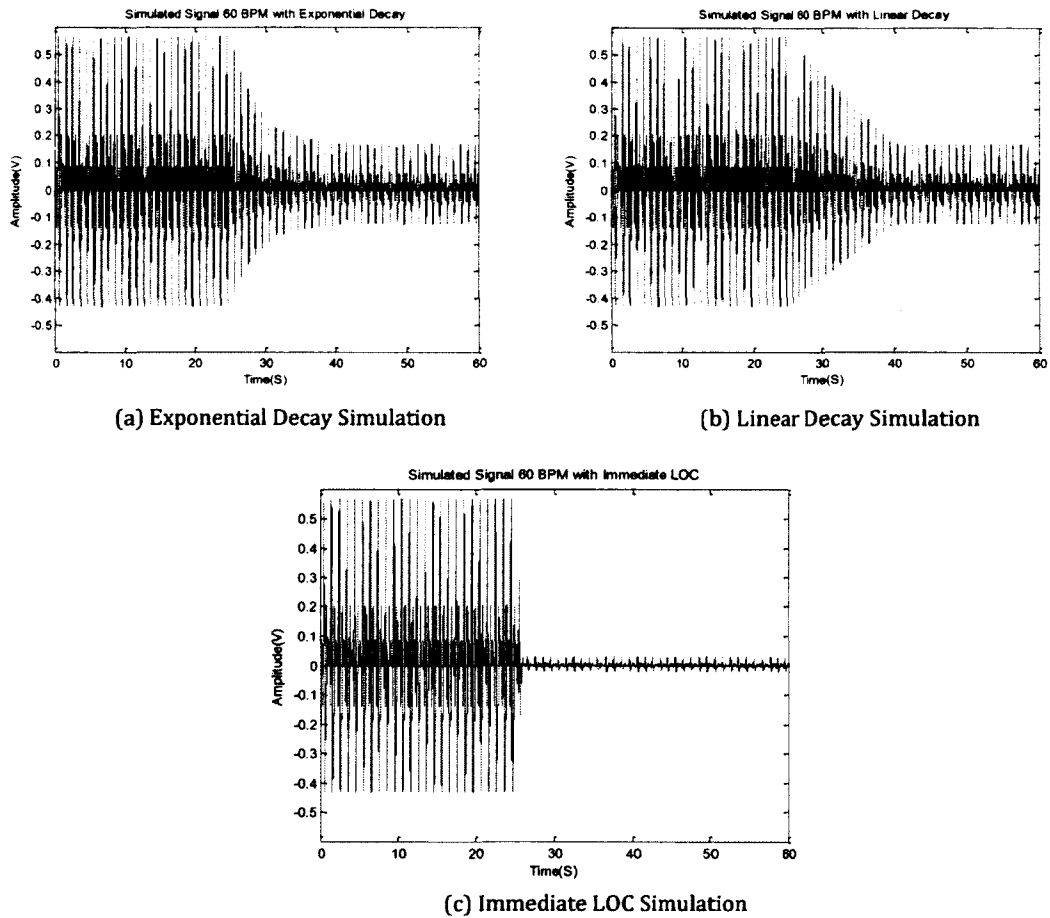


Figure 2.3: An exponential window (a), a linear window (b), and an immediate LOC (c) window used for testing and simulating LOC with pulse patency to ensure reliability in algorithm

It is important to note, as well, that there is no data on how pulse patency actually reacts with the onset of G-acceleration. It is known that the pulse patency decreases as G-acceleration increases, but there isn't any data suggesting how the outward force decays; whether it is linearly, exponentially, etc. (Figure 2.3). Therefore, different models of pulse patency decay, in the simulation portion of

the design process, will be tested using the identical algorithm from the microprocessor and the results will be discussed in follow-on Chapters.

Triple Axis Accelerometer

The accelerometer that was selected for the study was FreeScale's MMA7361L Triple-Axis Accelerometer. Accelerometers output a voltage that is proportional with acceleration along a particular axis. In the case of two-axis accelerometers, these devices measure acceleration along the x and y axis. This is of no use to the study because the impulse force created by the expansion of the arteries due to blood flow is in the z-axis direction since it is perpendicular to the surface (i.e. the forehead). It is possible to position the accelerometer so that either the x-axis or y-axis is perpendicular to the forehead. Due to the construction and packaging of accelerometers, this would cause the accelerometer to be obtrusive and would require mechanical devices to keep the accelerometer perpendicular to the forehead at all times. A z-axis accelerometer can lay flat along the surface of the forehead allowing for a spring and platform to place the sensor on the pilot internal to the helmet allowing for unobtrusive monitoring of pulse patency.

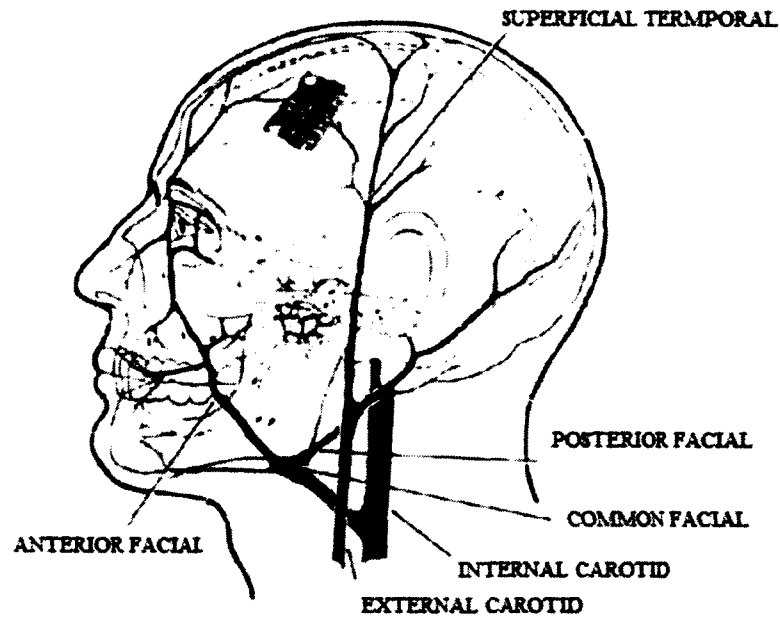


Figure 2.4: Best Placement of Triple Axis Accelerometer

The accelerometer is unique in many facets. The primary feature is that this particular accelerometer is G-selectable. The user can select between two G selectable measuring ranges, 1.5G and 6G. For this design, this was ideal. Since there is no data on the acceleration on the artery walls due to blood flow, a G-selectable accelerometer allowed for some experimentation opportunities. At 1.5G selection, the accelerometer outputs 800mV/G and at 6G selection, the accelerometer outputs 206mV/G.

Having G-select ability allows for several outcome results. Since the frontal branch of the STA tracks from the ear across the temple and onto the forehead, there is potential for motion artifact noises due to skeletal muscular movements along with gross human movement including head shaking. Such an example of

motion artifact is that of the jaw clench. If one clinches his or her jaw, there is a muscular reaction right along the temple. Ideally, the accelerometer could be placed on top of a small board, similar to the breakout board that could just be simply incorporated into a helmet that would alleviate the need for the pilot to precisely place the accelerometer along his or her STA. However, with the jaw clench causing motion in the temple area where the accelerometer would be located, there is the potential of motion artifact.

Having G-select ability would minimize such an issue. Since the mechanical frequency, which is different than the electrical signals produced by the electromyography of the temple muscle, is a low frequency, its natural acceleration would be lower than that produced by the impulse frequency of blood flow. Therefore, a company, like FreeScale, could design an accelerometer that has a certain "threshold" (i.e. G-selection), that would be higher than that of the mechanical acceleration of the temple muscle but lower than that of the impulse force produced by the impulse blood flow. However, since the design and production of such a product is not possible for this thesis, the use of both G-selections on this accelerometer will be used and analyzed.

Placement of the accelerometer along the STA is very crucial. In order for the accelerometer to create an output voltage, it naturally needs to accelerate in the z-direction. For this to happen, the expansion of the artery walls needs to expand away from the forehead meaning that it has to have support from the skull in order to do so. The skull in front of the ear, at the location along the STA prior to the

branch split, has a “valley”. If the accelerometer were placed here, it would cause the artery wall to expand towards the skull, and not outwards; thus, causing no acceleration, or output voltage, from the accelerometer. Therefore, it is imperative to place the accelerometer along the STA where the distance between the skull and skin, or soft tissue, is minimal (Figure 2.4).

Analog Signal Conditioning Front End and Electronic Component Selection

The National Institute of Health (NIH) reported that the average heart rate for children over 10 and adults, to include seniors, is 60-100 beats per minute (BPM). For well-trained athletes, who include most pilots, the resting heart rate is 40-60 BPM [31]. A 60 beats per minute heart rate translates into 1Hz frequency. Under stress or exercise, it is not uncommon for a heart rate to climb into the 200 BPM range. Therefore, the bandwidth of interest for the study is between 60 BPM to 300 BPM, or 1 Hz to 5 Hz.

The analog front end is a common bio-amplifier design circuit. It consists of front end buffers, followed by a fourth order high pass filter. The high-pass filter is followed by multiple gains stages and the final stage is a second order low-pass filter. In addition to the bio-amplifier, a DC level shifter had to be designed to ensure that the output voltage range is between 0 and 5V for the on-board Analog to Digital Converter (ADC) on the microprocessor. Both the high-pass filter and low-pass filter is a Sallen-Key topology Butterworth filter with unity gain. Unity gain was used for both filters due to the desire to minimize passive components throughout the

circuit. With 10% tolerance resistors and 20% accurate capacitors, the goal was to minimize the amount of error produced by the circuit and the sensor front end. Precision passive components weren't used due to the availability and low cost of the higher tolerance components. However, the components were measured on an impedance meter and placed in series or parallel to get as close to the calculated value as possible. This also applied to the gain stages, further discussed later. All calculated values of the resistors were either rounded to the closest standard value resistors or resistors were combined to get the value. The filter calculations were done based upon the availability of standard value resistors and selective capacitor values.

All filters and buffers were constructed using Linear LT1057 Dual JFET Input Precision High Speed Operational Amplifiers (Op Amps). These were chosen due to their incredibly high input resistance of $10^{12}\Omega$ and low output impedance (in $m\Omega$) allowing for minimal attenuation between stages of the bioamplifier in the case of long leads between stages or parts of the bioamplifier.

The power for each stage was taken off of two 9V batteries; one for the positive voltage and the other for the negative voltage. Since the voltage range of the ADC is only 5 volts, to reduce power consumption, two power regulators were emplaced, the LM7805 and the LM7905. The LM7805 is a positive 5V linear voltage regulator that was used for the positive voltage rail and the LM7905 is a negative 5V linear regulator. In addition, the accelerometer needed to be powered. Only taking a

maximum voltage of 3.2 volts, a LM317 adjustable voltage regulator was used to drop the voltage down from 5 volts to 3.2 volts.

Buffers

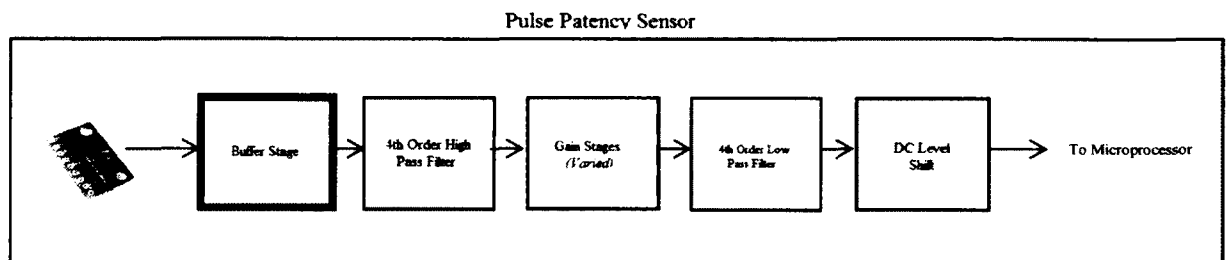


Figure 2.5: Block Diagram of Discussed Block in bold

The buffer, as seen in Figure 2.5, is a common input stage to analog conditioning stages. By emplacing a buffer as the input stage (Figure 2.6) , the high input impedance of the LT1057 minimizes the losses, or attenuation, of the signal between the source and the conditioning front it. It is also advantageous to utilize the buffer stage because it also ensures a low output impedance of that stage to the filtering stage.

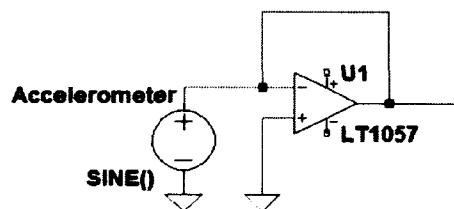


Figure 2.6: Schematic of buffer

Sallen-Key High-Pass Filter

Following the buffer stage, a 4th order high-pass filter was designed and constructed to remove any DC drift caused by the accelerometer or buffer (Figure 2.7). The Sallen-Key topology (Figure 2.8) is a commonly used filter configuration. The high-pass filter is a filter that will filter out lower frequencies and pass those frequencies of interest. In measuring heart rate, the minimum pass frequency should be 1Hz. It is important to note, that since this is a fourth order, the transfer functions will multiply together in order to yield the overall transfer function of the system.

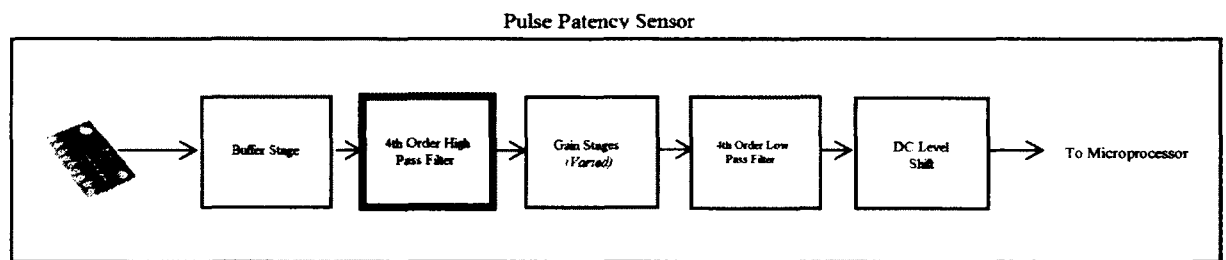


Figure 2.7: Block Diagram of Discussed Block in bold

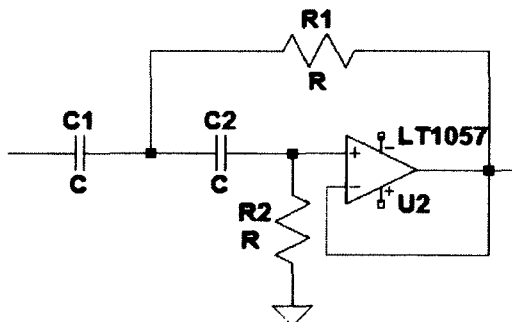


Figure 2.8: Sallen-Key High-Pass Butterworth Filter Topology

The transfer function of a second order system can be represented by

Equation 2.1:

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{\omega_0^2}{s^2 + \left(\frac{\omega_0}{Q}\right)s + \omega_0^2} \text{ where } Q = \frac{1}{2\zeta}$$

Equation 2.1: Generic Transfer Function equation

Where ω_0 is the frequency of interest, ζ is the dampening ratio, and Q is the quality factor.

Using nodal analysis, Kirchoff's current law, and solving for V_{out} with respect to V_{in} , the transfer function of the high pass filter can be seen in Equation 2.2:

$$\frac{V_{out}}{V_{in}} = H(s) = \frac{s^2}{s^2 + \left(\frac{1}{R_2C_1} + \frac{1}{R_2C_2}\right)s + \frac{1}{R_1R_2C_1C_2}}$$

Equation 2.2: Transfer function of Sallen Key High Pass Filter

The following step is to properly place the poles of the system. There will be two pairs of complex conjugate poles on the left hand side of the s-plane with a distance from the origin equal to that of the natural frequency. Seen in Figure 2.9 are the poles and zeros of the fourth order high-pass filter, with H1 being the poles of the first stage and H2 as being the poles for the second stage.

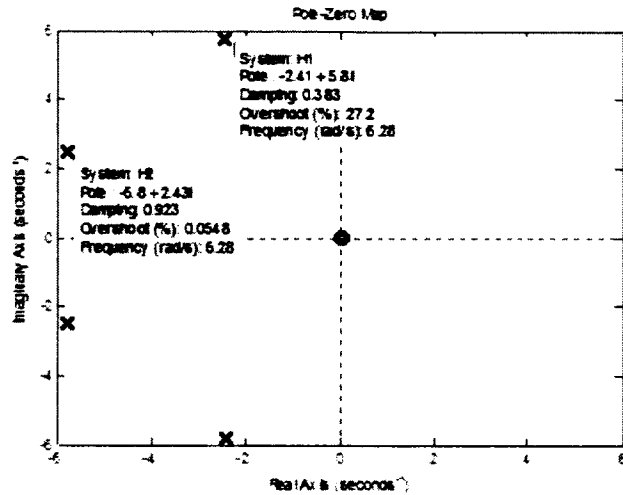


Figure 2.9: Poles and Zeroes of Fourth Order High Pass Filter

Plugging in the poles into the denominator to get the characteristic equation equals:

$$\text{For } H1: (s + 2.41 + 5.8j) * (s + 2.41 - 5.8j) = s^2 + 4.82s + 39.49$$

$$\text{For } H2: (s + 5.8 + 2.43j) * (s + 5.8 - 2.43j) = s^2 + 11.6s + 39.55$$

Equation 2.3: Characteristic Equation of the First and Second Stage

What remains is equating the terms of Equation 2.2 and Equation 2.3. Eventually, there will remain four unknowns and two equations. Therefore, the following equations are step by step simplification equations to show how each variable is solved for. Equating terms from Equation 2.1 and Equation 2.2 yields:

$$\text{Pass Frequency} = \omega_0 = 2 * \pi * 1 = \frac{1}{\sqrt{R_1 R_2 C_1 C_2}}$$

Equation 2.4: Pass Frequency Calculation of each Stage

$$\frac{\omega_0}{Q} = \left(\frac{1}{R_2 C_1} + \frac{1}{R_2 C_2} \right)$$

Equation 2.5: Quality Factor and Frequency equation

Each individual stage will have independent Q values due to their unique pole placement. The product of the Q values for each stage will be 0.707, which is the desired Q value for a Butterworth Filter. Equating the terms from Equation 2.3 and 2.5, the Q values for the first and second stages are:

For H1:

$$\frac{\omega_0}{Q} = 4.82 \therefore Q = 1.304$$

For H2:

$$\frac{\omega_0}{Q} = 11.6 \therefore Q = .542$$

Substituting Equation 2.4 into Equation 2.5 yields:

$$Q = \frac{\sqrt{R_1 R_2 C_1 C_2}}{R_1 (C_1 + C_2)}$$

Equation 2.6: Standalone Quality Factor Equation

Since there are two equations and four unknowns, if $C_1 = C_2 = C$, then the equations can be minimized further to yield:

$$\omega_0 = \frac{1}{C\sqrt{R_1R_2}}$$

Equation 2.7: Reduced Frequency Equation

$$Q = \frac{1}{2} \sqrt{\frac{R_2}{R_1}}$$

Equation 2.8: Reduced Quality Factor Equation

Substituting terms from Equation 2.7 and Equation 2.8 to solve for R_1 and R_2 yields:

$$R_1 = \frac{1}{2\omega_0QC} \text{ and } R_2 = \frac{2Q}{\omega_0C}$$

Equation 2.9: Resistor Value Calculations for Sallen-Key High Pass Filter

Equation 2.9 is a simplified calculation to solve for the resistor values of the 4 pole Butterworth filter. It is important to use the appropriate Q values for each stage. Using these values, a pass frequency of 1Hz, and a capacitor size of 1μF, all parameters for the circuit can be determined.

| Stage | Pass Frequency | Quality Factor (Q) | Capacitor (C ₁ & C ₂) | Resistor (R ₁) | Resistor (R ₂) |
|---------|----------------|--------------------|--|----------------------------|----------------------------|
| Stage 1 | 1Hz | 1.5708 | 1μF | 100 kΩ | 100 kΩ |
| Stage 2 | 1Hz | .5420 | 1μF | 147 kΩ | 172 kΩ |

Table 2.1: Values for Fourth Order Sallen-Key High-Pass Filter

Plugging in the values for the variables seen in Table 2.1, the transfer function for each stage can be calculated and are displayed in Equation 2.10 and Equation 2.11. The characteristic equation of each transfer function is identical to that seen in Equation 2.3.

$$H_1(s) = \frac{s^2}{s^2 + 4.817s + 39.48}$$

Equation 2.10: Transfer Function of the First Stage

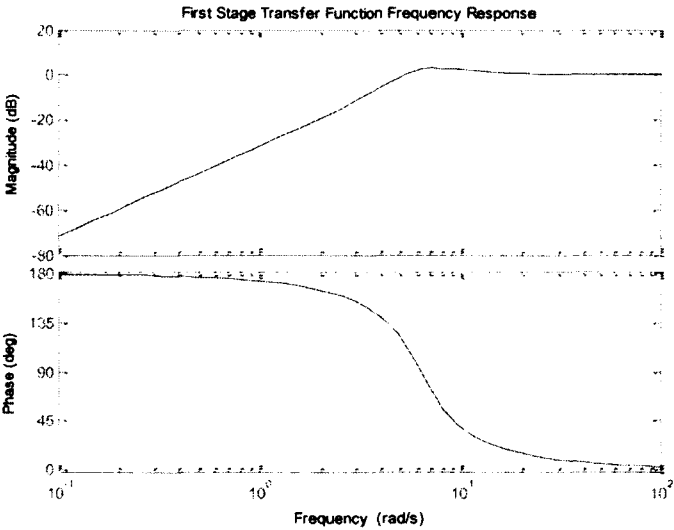


Figure 2.10: Frequency Response of First Stage

$$H_2(s) = \frac{s^2}{s^2 + 11.59s + 39.48}$$

Equation 2.11: Transfer Function of the Second Stage

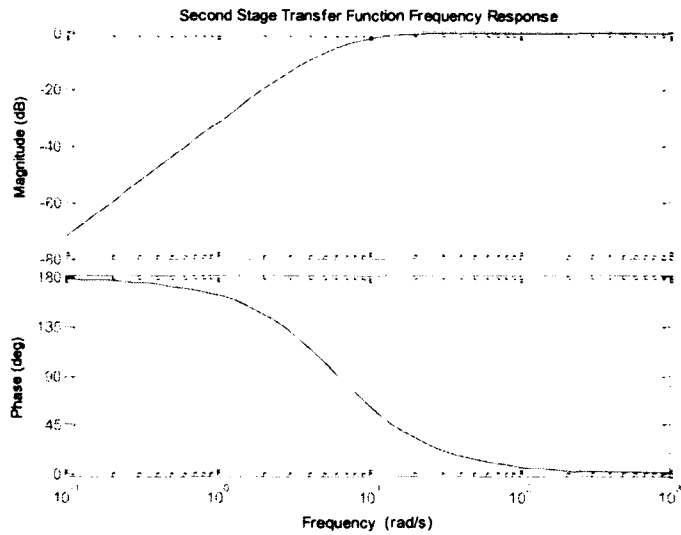


Figure 2.11: Frequency Response of Second Stage

After both transfer functions are derived, the combination of the two transfer functions (Equation 2.12) are what equate to the entire Butterworth filter. As seen in Figure 2.10, the frequency response of the first stage of the transfer function is under damped, whereas the second stage is over damped. Combined, however, these two stages equation to a slightly over-damped system (Figure 2.12) with a dampening ratio and quality factor equal to 0.707.

$$H_1(s) * H_2(s) = H_{tot}(s) = \frac{s^4}{s^4 + 16.41s^3 + 134.8s^2 + 647.8s + 1559}$$

Equation 2.12: Overall High Pass Filter Transfer Function

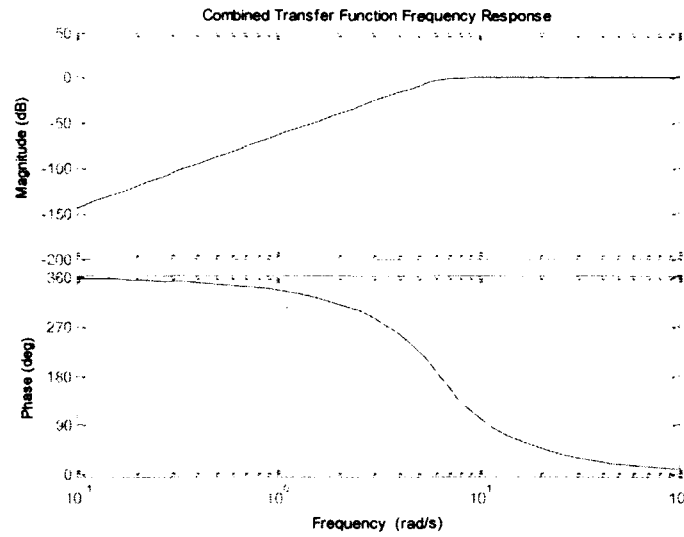


Figure 2.12: Combined Transfer Function Frequency Response

Gain Stages:

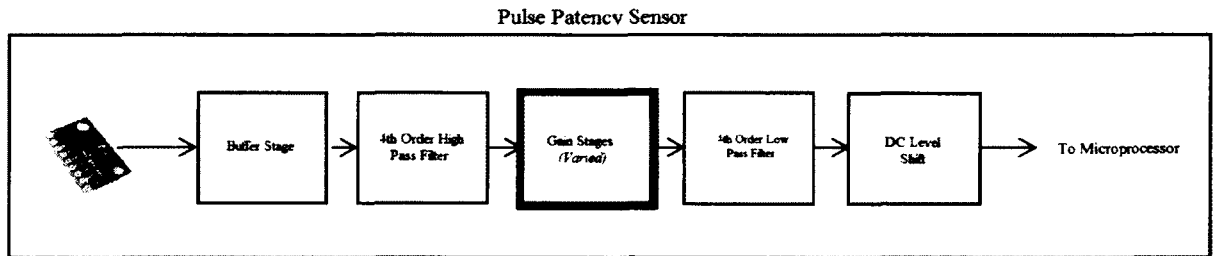


Figure 2.13: Block Diagram of Discussed Block in bold

The gain stages (Figure 2.13) followed the high-pass filter stage. Typically, an inverting or non-inverting Op Amp design would work, but due to the desire to minimize passive components, INA128 instrumentation amplifiers were used for the gain. If either the inverting or non-inverting gain configuration was used, there

could be a gain error of upwards of 20% given that both resistors in either configuration have a tolerance of 10%. By grounding the positive input terminal, the INA-128 was always in fully differential mode. The gain of the INA-128 is adjusted by one external resistor which allows for ease of adjustment during testing (Equation 2.13).

$$\text{Gain} = 1 + \frac{50k\Omega}{R}, \text{ where } R \text{ is the external resistor}$$

Equation 2.13: Gain of an INA-128 Instrumentation Amplifier

The succession of the three gain stages were adjusted accordingly during each phase of the testing, excluding the simulation part. During the physical model, a model setup to simulate the expansion and contraction of the STA through the use of latex tubing and air compressors, for example, the gain of the system had to be lowered compared to that of the human body due to the force of the air pressure being greater than that of the force of the blood. If the gains were to have remained the same, the output signal would have railed out against either supply voltage.

Sallen-Key Low-Pass Filter

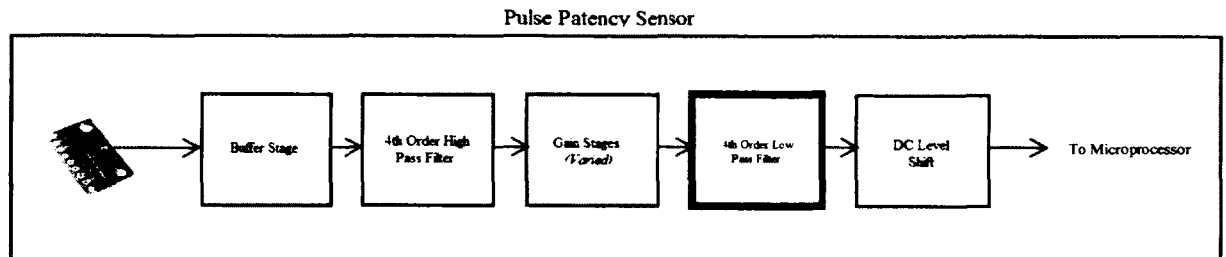


Figure 2.14: Block Diagram of Discussed Block in bold

A fourth order low-pass filter was used as the second to last signal conditioning stage prior to the ADC. The fourth order low-pass filter is a Butterworth filter that decays all frequencies higher than the cutoff frequency ($f = 10 \text{ Hz}$) at a rate of 40dB per decade. The low-pass filter is a very important final piece for several reasons. First, it attenuates the higher frequencies that are not of interest and have the ability to cause false readings or negatively affect frequencies of interest. Prominent noise, and an issue during testing, is 60Hz noise. Secondly, the algorithm that was used for processing of data relied on taking the derivative between data points. The fastest any given point could change would be at the highest frequency present in the signal. By choosing a low cutoff frequency for the low-pass filter, a lower sampling frequency can be used, which alleviates the need for more memory and processing power. If they highest pass frequency allowed is 10Hz, then 60Hz is also filtered out. This low-pass frequency allows the signal of interest to retain a minimum of 2 harmonics along with using a reasonable sample frequency with the given amount of storage available on the Arduino R3 microprocessor.

$$\text{slope} = \frac{\Delta Y}{\Delta X} = \frac{\sin(2\pi f * t) - \sin(2\pi f * (t - t_s))}{t_s}$$

Equation 2.14: Calculated ideal maximum slope between two data points

Where f is the highest frequency present in the signal and t_s is the sampling time interval. This equation is further discussed in the algorithm section. Equation 2.14 is an important consideration pertaining to the setting the cut off frequency of the low-pass filter.

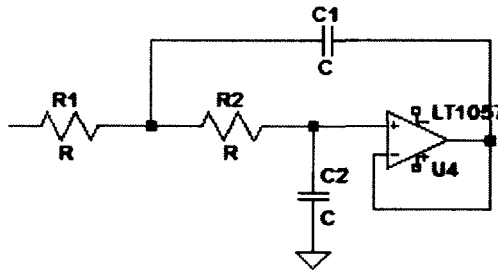


Figure 2.15: Sallen-Key Low-Pass Butterworth Filter Topology

Using nodal analysis on Figure 2.15 and in a similar process to that of a previous subsection *Sallen-Key High-Pass Filter* (Page 39), the derivation of all variables is shown below. It is important to reference Equations 2.1 for the following as the standard form for a second order system differential equation will be needed to equate the variables.

$$H(s) = \frac{1}{R_1 R_2 C_1 C_2} \frac{1}{s^2 + \left(\frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} \right) s + \frac{1}{R_1 R_2 C_1 C_2}}$$

Equation 2.15: Sallen-Key Low-Pass Filter Transfer Function

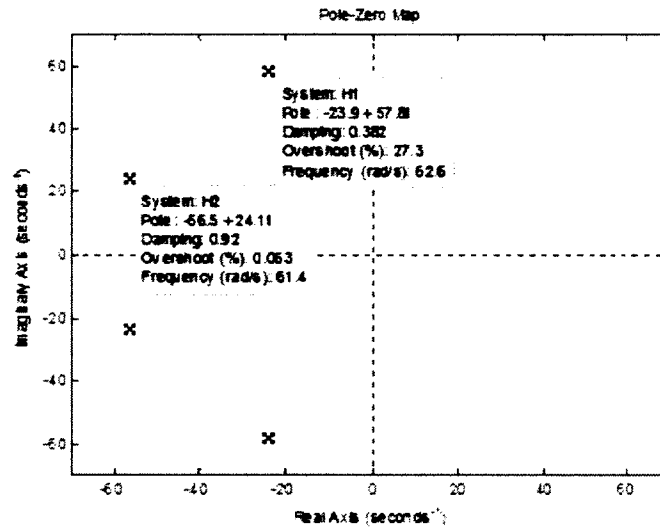


Figure 2.16: Poles and Zeroes for Fourth Order Low Pass Filter

Utilizing the pole placements seen in Figure 2.16 and solving for the characteristic equation of the transfer function with the complex conjugate pole locations equals:

$$(s + 23.9 + 57.8j) * (s + 23.9 - 57.8j) = s^2 + 47.78s + 3915$$

$$(s + 56.5 + 24.1j) * (s + 56.5 - 24.1j) = s^2 + 112.9s + 3796$$

Equation 2.16: Characteristic Equation of Low-Pass Filter Transfer Functions

Equating the terms in Equation 2.1 and Equation 2.15, yields:

$$\text{Cutoff Frequency} = \omega_0 = 2 * \pi * 10 = \frac{1}{\sqrt{R_1 R_2 C_1 C_2}}$$

Equation 2.17: Cut Off Frequency Equation

Since this is once again two second order transfer functions being combined to form the fourth order filter, the Q values will vary. When equating the terms from Equation 2.16 and Equation 2.18, the Q value for the first stage is calculated to be 1.32 and 0.5565 for the second stage. Together, these two stages form the maximally flat Butterworth filter.

$$\frac{\omega_0}{Q} = \frac{1}{R_1 C_1} + \frac{1}{R_2 C_1}$$

Equation 2.18: Frequency and Quality Factor Calculation

Substituting Equation 2.17 into Equation 2.18:

$$Q = \frac{\sqrt{R_1 R_2 C_1 C_2}}{(R_1 + R_2) C_2}$$

Equation 2.19: Quality Factor Equation

Once again, this leaves two equations and four unknowns for each stage causing it to be an underdetermined solution. However, the ratio presented in Equation 2.20 will allow for the calculation for all resistor and capacitor values seen in Figure 2.15.

$$\frac{C_2}{C_1} \leq \zeta^2 \text{ where } \zeta = \frac{1}{2Q}$$

Equation 2.20: Capacitor Ratio

If the terms are equated from Equation 2.15, the proper circuit component values can be calculated. For this part of the pulse patency sensor, the values of all the components are presented in Table 2.2. Figure 2.17 shows the frequency response of the 4th order Sallen key low-pass filter.

| Stage | Cutoff Frequency | Quality Factor | Resistor (R1) | Resistor (R2) | Capacitor (C1) | Capacitor (C2) |
|-------|------------------|----------------|---------------|---------------|----------------|----------------|
| 1 | 10Hz | 1.57 | 470kΩ | 270kΩ | 10μF | 27nF |
| 2 | 10Hz | .5565 | 910kΩ | 200kΩ | 54nF | 27nF |

Table 2.2: Values for Fourth Order Sallen-Key Low Pass Filter

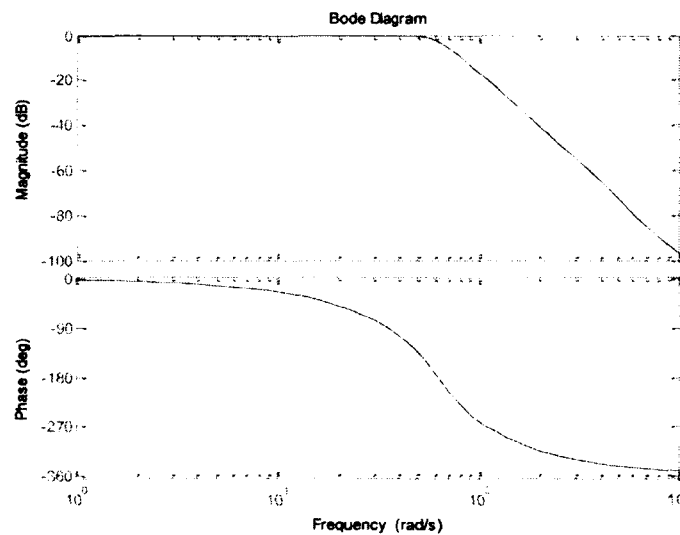


Figure 2.17: Frequency Response of 4th Order Low Pass Filter

DC Level Shift

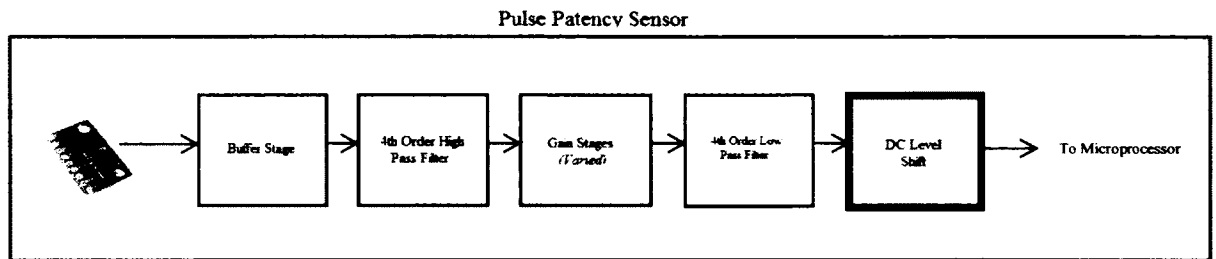


Figure 2.18: Block Diagram of Discussed Block in bold

The DC level shift (Figure 2.18) circuit is a very important part of the circuit due to the ADC on the Arduino R3 board. The ADC onboard the Arduino only reads positive values between 0 and 5 volts. Ideally, since there is both a positive and negative swing to the pulse patency signal, the DC level should accurately be between 0 and 5 volts at 2.5 volts. In order to accurately implement this, a DC level shift circuit needs to be used (Figure 2.19).

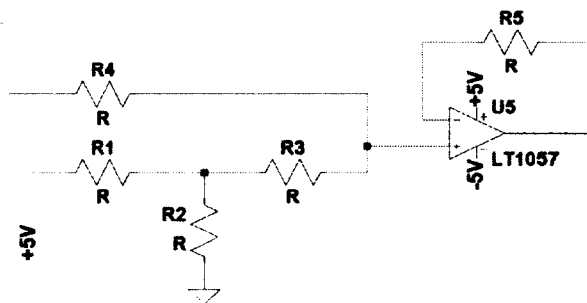


Figure 2.19: DC Level Shift Circuit

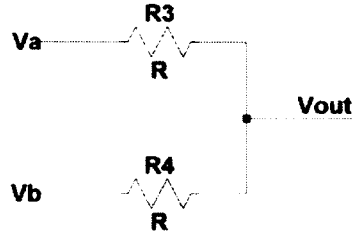


Figure 2.20: Simplified Schematic of Input

Figure 2.19 shows the overall circuit for the DC level shift. The 5 volt input to the circuit is taken off of the voltage regulated power lines for all the circuitry. If the circuit is minimized, as seen in Figure 2.20 where V_{out} is the positive terminal of the LT1057 Op Amp, V_{out} is equal to:

$$V_{out} = 2.5 = \frac{\frac{V_a}{R_3} + \frac{V_b}{R_4}}{\frac{1}{R_3} + \frac{1}{R_4}}$$

Equation 2.21: Output Voltage Between the two Input Voltages

V_a is the AC output voltage from the low pass filter with no DC component so its variable equals zero. V_b is a DC input from the voltage divider set up by R_1 and R_2 , so it can be adjusted accordingly. Equation 2.21 can be simplified to:

$$V_{out} = 2.5 = \left[\frac{R_3}{R_3 + R_4} \right] V_b$$

Equation 2.22: Simplified Calculation of Input Voltages

$$V_b = 5 \left[\frac{R_2}{R_1 + R_2} \right]$$

Equation 2.23: Input Voltage Divider Equation

If there were any amplification needed to the AC portion of waveform, R_5 can be adjusted according to the non-inverting amplifier gain equation. Since that is not necessary, R_5 is just shorted to provide unity gain.

| Component | R₁ | R₂ | R₃ | R₄ |
|------------------|----------------------|----------------------|----------------------|----------------------|
| Value | 10kΩ | 40kΩ | 10MΩ | 6MΩ |

Table 2.3: Calculated Values for DC Level Shift Circuit

System Circuit and Frequency Response

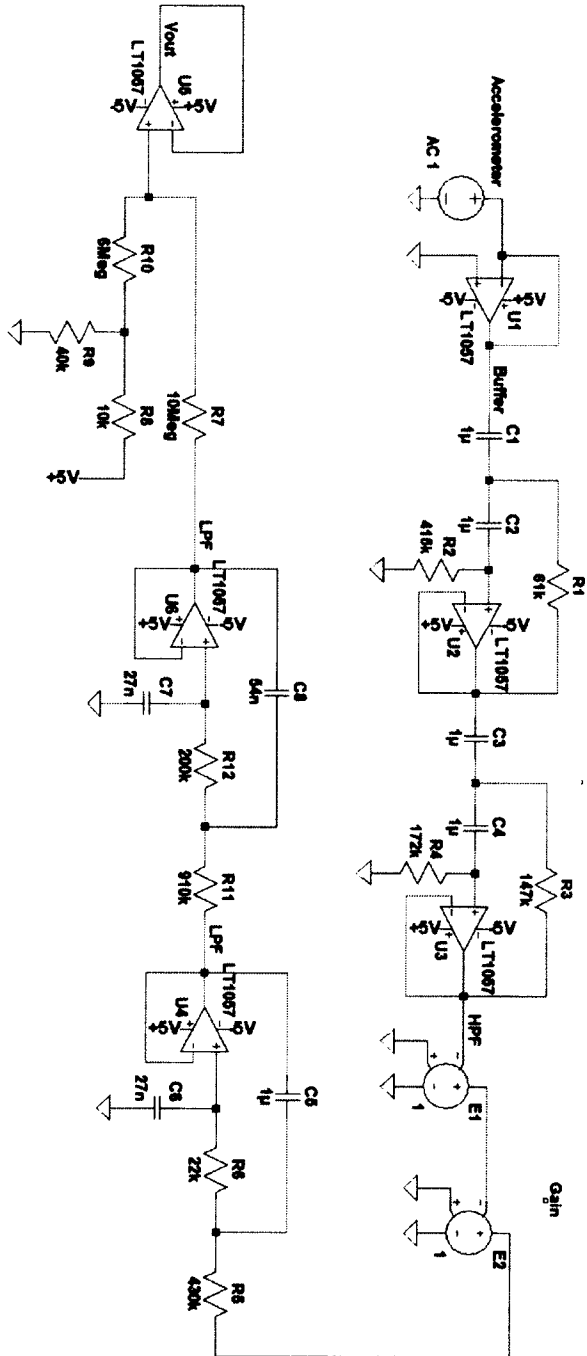


Figure 2.21: Final Signal Conditioning Pulse Patency Sensor

Figure 2.21 shows the final overall schematic of the pulse patency sensor. Circuit components E1 and E2 in Figure 2.21 are gain adjustors and represent the gain stages (refer to Page 46). Figure 2.22 shows the frequency response of the overall system.

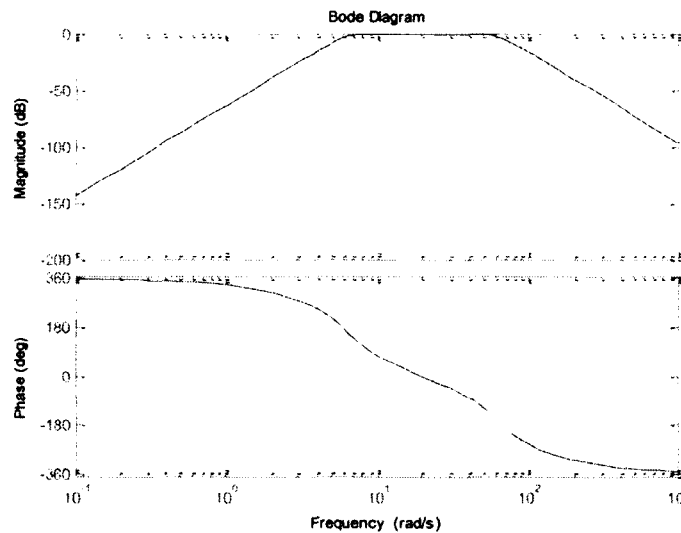


Figure 2.22: Overall System Frequency Response

MATLAB and Microprocessor

MATLAB was used extensively throughout the testing portions of the experimentation. The version of MATLAB that was used was R2012a. Although this is the current version of MATLAB, previous versions could be used to run the code. There are no new features in version R2012a that would make the code presented in the Appendix incompatible with previous versions of MATLAB. MATLAB was used to verify the algorithm developed to be coded on the microprocessor. By using

a USB compatible digital scope, a thumb drive was used to transport the data to the computer to be analyzed by MATLAB. Ideally, the importing of data from the digital scope into MATLAB is identical to that of the information presented to the microprocessor after the ADC.

The microprocessor that was used for data analysis was an Arduino Uno - R3 SMD. Arduino is an open-source physical computing platform based on a simple input/output board. The Uno-R3 has an ATmega328 microcontroller with 14 digital input/output pins, 6 analog pins, 32kbytes of flash memory, and a clock speed of 16MHz. The onboard ADC has 10 bits of resolution and a maximum of a 5 volt input voltage. This allows for 1024 values between 0 and 5 volts. Therefore, the ADC yields approximately a $\pm 5\text{mV}$ error in signal conversion. Through the analog signal conditioning front end, the gain of the system can be adjusted to minimize the impact of a 5mV error on the ADC. The gain of the system is varied throughout the experimentation due to the differences simulation, physical modeling, and human testing.

MATLAB and Microprocessor Algorithm: Overview

Once the signal makes it through the analog front end, through the on-board ADC, to the microprocessor, it now becomes a digital signal (Figure 2.23). The continuous output from the back end of the low-pass filter becomes an array of data points.

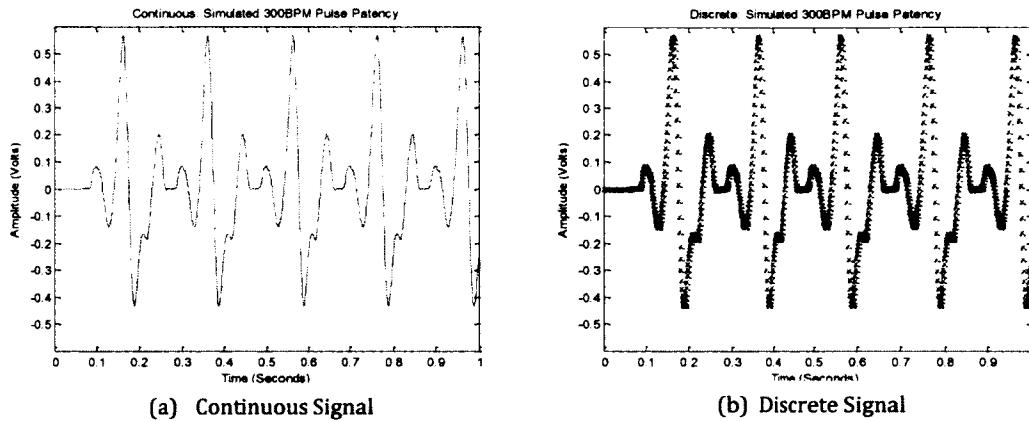


Figure 2.23: Continuous vs. Discrete 300BPM Pulse Patency Measurements

At a very fundamental level, digital signal processing comes down to accurate data sampling. The Nyquist theorem states that the minimum sampling rate needed to accurately reproduce a signal is twice the highest frequency, or bandwidth (B), present in the signal (Equation 2.24).

$$\text{Sampling Frequency} = f_s = 2 * B$$

Equation 2.24: Nyquist Sampling Theorem

Accurate sampling frequency is further discussed in the next section

Sampling Frequency (Page 60).

After sampling the signal, in order to detect a pulse, a peak detect algorithm needs to be implemented. There were two techniques analyzed in determining a peak. First, a method that will be referred to as “Max in an Array Method”, sample points are imported into an array, and, through the use of a *for-loop*, run through to determine the highest number. That number would be stored as the peak. More data would be imported into the same array and, once again, would be run through the

same *for-loop* to determine the max. This would happen over and over again until a peak was determined, and then a similar, but opposite, process would be used to detect a minimum peak.

The second method to determine a peak is to utilize two data points as two points along a straight line. Using the derivative, or slope, of the two points, one can determine the peak by calculating the slope and triggering once it is equal to zero. This method doesn't require excessive memory usage, but could be more susceptible to noise. These two methods are further discussed in section titled *Peak Detect Methods* (page 62).

Further sections include moving average algorithm (page 67), pulse detection criteria and variance (page 69), auto-pilot warning and time restraints (page 72), and the final coding block diagram is shown on page 74. All the code for the algorithm, both for MATLAB and for the Arduino are presented in the Appendix.

Sampling Rate (Frequency)

As Nyquist theorem states, the sampling frequency needs to be at least twice the bandwidth of the signal. Since the heart rate of interest is between 60 BPM and 300 BPM, or 1 Hz to 5 Hz, it may cause misunderstanding to conclude that 5 Hz is the bandwidth of the signal. In fact, the bandwidth of the signal is the highest frequency present in the highest frequency signal. Therefore, the bandwidth of

interest would be the highest frequency present in the 5Hz waveform, as depicted in Figure 2.24.

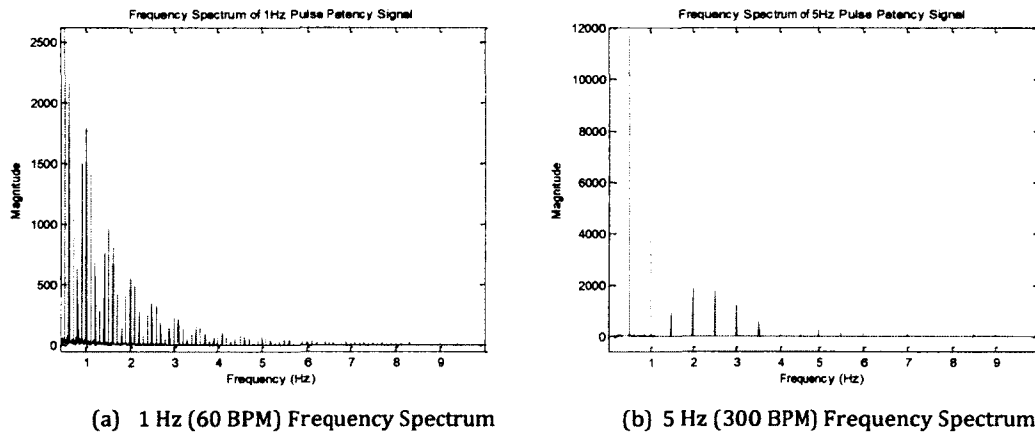


Figure 2.24: Normalized Frequency Spectrum of 1 Hz (a) and 5 Hz (b) Heart Rate from Physical Model

Both plots in Figure 2.24 are normalized frequency spectrum plots taken from the Physical Model testing. In the plots, one can see that there is higher frequency content present in higher frequency heart rates. Therefore, the sampling rate needs to be adjusted according to the 5 Hz frequency spectrum. Using this and wanting to preserve up to the second harmonic, a minimum sampling rate of 20Hz needs to be used.

However, in order to utilize the slope method, the more points present along the peaks, the closer to zero, ideally, the slope will become. If the “Maximum in an Array Method” was used, a lower sampling frequency may be used. However, if higher signals do become present, they may be aliased as lower frequency signals, and may cause a false peak readings.

Peak Detect Methods

To “pack” the signal with as many data points as possible, a slope calculation needs to be determined. The maximum difference a data point can change from the previous one is a function of the highest frequency present in the signal. Since the highest frequency present in the signal is 10 Hz, we can use Equation 2.14:

$$\text{slope} = \frac{\Delta Y}{\Delta X} = \frac{\sin(2\pi f * t) - \sin(2\pi f * (t - t_s))}{t_s}$$

By inserting the proper values, the maximum slope difference between two data points can be determined. In the case of a 5Hz signal:

$$f = 10 \text{ Hz}$$

$$t = \text{time at maximum peak or } \frac{\pi}{2} \text{ radians} = \frac{1}{4} * \frac{1}{f} = 0.0250$$

Equation 2.25: Maximum Peak of a 10Hz Sine Wave

$$t_s = \frac{1}{f_s}$$

Equation 2.26: Sampling Interval Equation

Using “trial and error”, the correlation between sampling frequency and maximum slope between data points at a peak can be determined. Due to hardware limitations on the Arduino R3, a sampling frequency of 1 kHz was used throughout the design process including: simulation, physical modeling, and human element.

Plugging all these values into Equation 2.14, this equates to a maximum slope difference, at the peak, of 2 (refer to the Appendix for MATLAB calculation code).

Assuming there is no noise and no error caused by the microprocessor, this would be true. However, this is not the case. The microprocessor's ADC has a ± 5 mV error which is a problem. Without any noise in the system, for example, at the peak of the signal, one data point is *peak + 5mV* and the next point is *peak - 5mV*. With a sampling frequency of 1kHz, this causes a slope change of 10. This would not meet the criteria and the microprocessor would not detect the peak which could lead to further problems. In order to minimize this effort, additional coding was needed to be added into the algorithm.

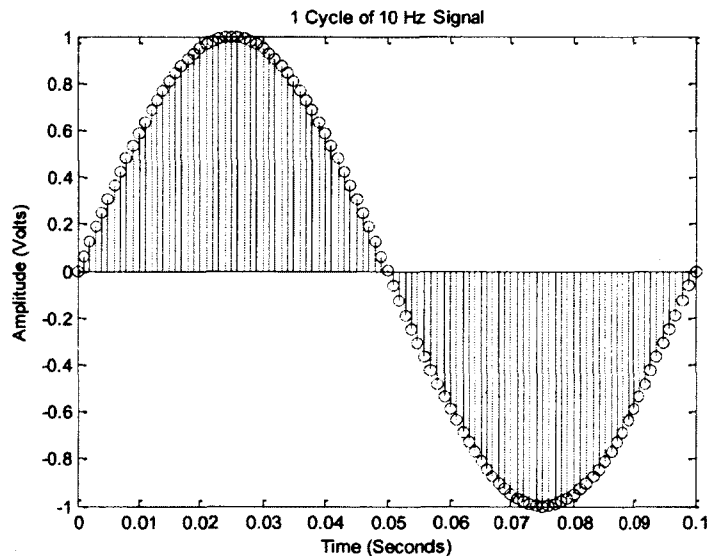


Figure 2.25: 1 Cycle of a 10Hz Sine Wave

Figure 2.25 is 1 cycle of a 10Hz signal with a sampling rate of 1kHz.

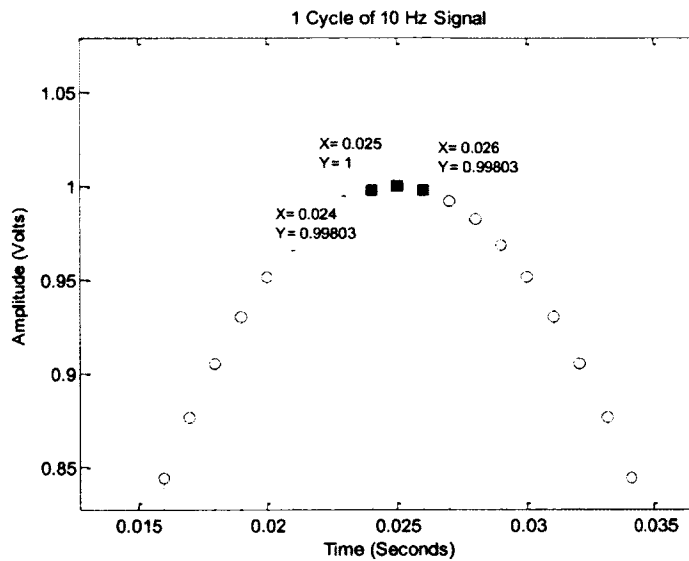


Figure 2.26: Peak of the 1 Cycle 10Hz Sine Wave

In Figure 2.26, the peak of the 10Hz sine wave (Figure 2.25) is seen along with pointers marking the peak and subsequent data points to the left and the right. The slope between either the left or right points or the peak is 2, as calculated. In the new slope calculation, an array is initiated with 3 indices. This array imports the first value, averages the second and third value, and finds the slope between the first value and the average of the second and third value.

$$\text{Array} = [\text{val1} \text{ val2} \text{ val3}]$$

$$\text{Slope} = \text{abs} \left\{ \frac{\text{val1} - \frac{\text{val2} + \text{val3}}{2}}{t_s} \right\}$$

Equation 2.27: New Slope Calculation to Minimize Error

Using Figure 2.26:

$$\text{Array} = [.99803 \ 1 \ .99803]$$

$$\text{Slope} = \text{abs} \left\{ \frac{.99803 - \frac{1 + .99803}{2}}{.001} \right\} = .97$$

In a ADC worst case scenario, as previously stated, the incoming signal may look like this:

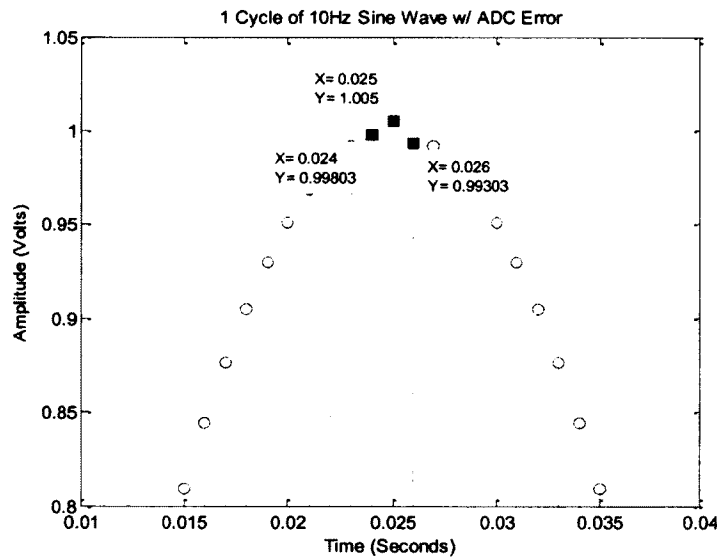


Figure 2.27: 10 Hz Sine Wave with Simulated ADC Error

In Figure 2.27, a simulated ADC maximum error is shown with the peak having added 5mV and the next point subtracted 5mV. The slope between the first and second point is 7 and between the second and third point is 12. Using the initial slope algorithm, this would cause the microprocessor to not consider this a peak. With the new algorithm with the array it would be as follows:

$$\text{Array} = [.99803 \ 1.005 \ .99303]$$

$$\text{Slope} = \text{abs} \left\{ \frac{.99803 - \frac{1 + .99303}{2}}{.001} \right\} = 1.52$$

Using this algorithm, the ADC error can be minimized and eliminated. This not only affects the error produced by the ADC but minimizes the impact of outside noise. However, this algorithm will only help with outside noise, it can't be relied on to null it. Precautions are taken with peak detection criteria to assist in detecting false positives.

The max detect array has been mentioned several times throughout the description of the peak detect algorithm. This technique was tested as the initial procedure for detecting the peak. However, with the implementation of the moving average algorithm, this technique was discarded. The peak detect method is highly susceptible to noise. A high frequency impulse would cause a potentially large value to be implanted into the moving average algorithm, raising the average (perhaps past the value of all follow on peaks). Although using the slope is also susceptible to noise, it is better to not detect a peak, then to allow a large, incorrect peak into the moving average algorithm. A high frequency noise signal would cause the slope method to not detect the peak. However, with such a slow signal and such a large number of samples, it is expected that the follow on data points would meet the slope criteria and detect the peak.

Moving Average Algorithm

The moving average array is a very important piece of the overall algorithm. Every subsequent pulse is based off of the average of the moving average array. The moving average array is initiated by the microprocessor and consists of two very important concepts. The first is the importation of allowable peaks. The second is the size of the array. Both of these have to be designed correctly in order for the moving average array to work correctly. In a simulation example, a 60BPM pulse patency signal was generated using Simulink and the moving average was constantly being plotted. The results are shown in Figure 2.28.

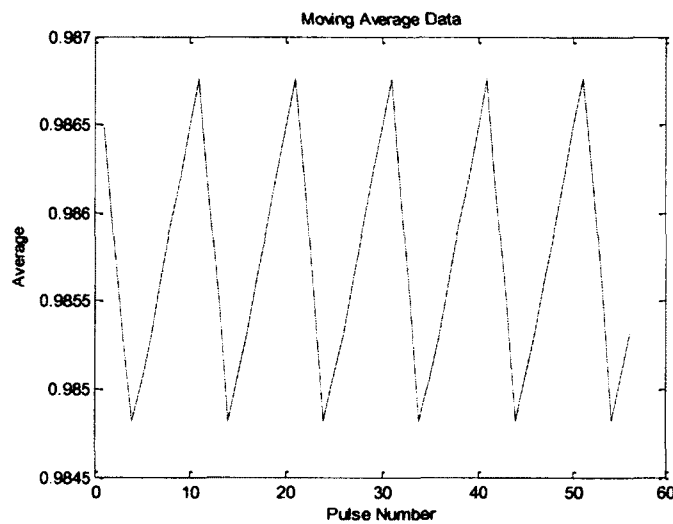


Figure 2.28: Moving Average Data during Simulation

When the moving average array is initiated by the microprocessor, it starts off with an *index* (in this case: 5) size array of zeros. Each zero is representative of the peak difference between the positive and negative peak of the pulse. Before the

moving average array is implemented, the first 100 detected peaks assist in initiating the moving average array. The first five peaks are imported into the array and the follow on ninety-five peaks are compared to the initial five peaks. Then, an *if-else* statement runs each follow-on peak through the array and compares it to that of each value present in the array. If it is greater than the first value, for example, then it replaces the first index of the array and exits the loop. The array, at the end of the first hundred peaks, consists of the five biggest peak differences. This may seem like a large amount of peaks, considering the presence of noise, it equates to approximately the first five seconds. Notice in Figure 2.28, that the peak number ends at 55. This is due to the first five peaks being used for the moving average array.

Initiating the moving average array off of the five largest peak differences is highly susceptible to noise. During this time, with the presence of noise, it may cause a large number to be part of the initial moving average array. Therefore, it is important to pick a large amount of indices. There is an evident cost-benefit to this. The benefits of having a large array allows for a “noise peak” to be present in the array and not throw off the actual average of the peaks. For example:

$$\text{Moving Average} = \text{avg}[2 \ 2 \ 5 \ 2 \ 2] = \frac{2 + 2 + 5 + 2 + 2}{5} = 2.6$$

This causes a 30% error, which is fixed by the variance variable, explained in the next section. This issue could also be fixed in another method. Prior to implementing the array and analyzing all follow-on peaks off of the moving average

algorithm, the array can “run through” itself again and self-eliminate the noise. If the array finalizes, as seen above, the array can very simply replace the 5 with the average of the other four variables.

The size of the array has a cost associated with it. The array is slower to respond to high acceleration. This means that when the pilot accelerates and causes the pulse patency to lower, the moving average of the array, in a similar fashion as the noise, responds slower. Therefore, the “perfect balance” needs to be tested using the algorithm in a centrifuge to determine the best possible size.

As each pulse gets determined based off of the moving average array, its peak-to-peak difference is imported into the array and the oldest value in the array is thrown out. In this manner, the moving average array is constantly being updated and is able to track each pulse.

Pulse Detection Criteria and Variance

After the slope is calculated and the peak is determined, there needs to be certain criteria to meet in order for the microprocessor to determine that it is a peak. Using strict criteria would assist in eliminating the maximum amount of noise possible. Seen in Figure 2.29 is an output signal taken from the physical model. There are certain characteristics of the waveform. First, however, it is important to note that the secondary pulse after each primary pulse is due to the air valve

shutting off and is the transient response of the spring holding the accelerometer down (further discussed in the Physical Model section).

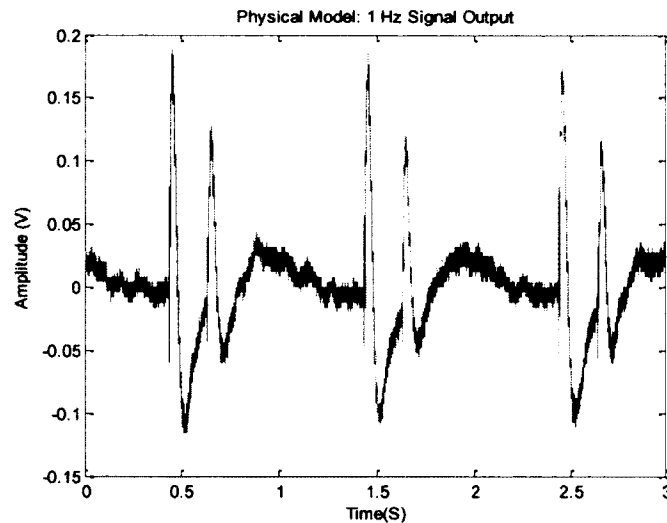


Figure 2.29: 1 Hz Pulse Patency Signal from Physical Model

Each initial waveform of the pulse patency physical model has similar characteristics. First, as the blood passes through the arteries, there is an initial spike in the positive direction. As the blood flow stops, there is an exponential decay as the accelerometer returns back to its initial state. This decay can be modeled as an electronic RC response. It is also observed that the peak-to-peak time difference is fairly minimal, easily under 0.2 seconds. And, finally, the observation that all the peaks remain the same. This is a very important observation because there are two ways to filter out noise. One way, is time-based and the second is amplitude-based. Further observations include minor high frequency noise riding on a much lower frequency between each pulse. However, that is fairly negligible due to a moving average array.

So, in order to effectively detect a peak, the algorithm needs to have strict qualifications for a pulse. First, each peak needs to have a slope of less than two. When the peak criteria are met and there is a high peak and a low peak, numerous time-based qualifications needs to be met. The first being the time difference between the positive and negative peak, which was determined to be less than 0.2 seconds. The second is the time difference between that peak and the previous peak. At 300BPM, which equates to 5 beats per second, the time difference between two peaks has to be a minimum of 0.2 seconds apart as well. With an added “buffer zone”, that number can be lowered down to 0.15. The final qualification is the moving average array.

Since not all pulses will be the exact same amplitude, a variance variable needs to be added to the average of the moving average array. This variance needs to be large enough to detect all peaks, including potential pulse patency drop off methods (refer to Figure 2.3). Through testing, it was determined that the best variance to use was 85%. So, if the array is:

$$\text{Moving Average Array} = [2 \ 2 \ 2 \ 2 \ 2]$$

$$\text{Average} = 2$$

$$\text{With Variance: min} = 1.7; \text{max} = 2.3$$

If the peak is between 1.7 and 2.3, then it gets determined to be a pulse, and the value of the peak difference is imported into the moving average array. At an 180BPM heart rate, this would allow for a pulse patency maximum exponential

decay time constant of 2 or maximum linear decay slope of 0.5. Utilizing the variance variable allows for changing moving average array but allows a greater noise margin (Equation 2.28).

$$\text{Exponential Decay} = 1 - e^{-\frac{t}{\tau}} = .15 \therefore \tau = 2$$

$$\text{Linear Decay} = \frac{\Delta Y}{\Delta X} = \frac{.15}{.33} = .5$$

Equation 2.28: Calculated Decay Rates vs. Variance

Auto-pilot Warning and Time Restraints

Establishing time restraints for pulse patency detection is a crucial part of the overall design. Since time is constantly being monitored with each incoming point being time stamped, by simply keeping track of the time difference between the last peak and the current time, a simple auto-pilot warning can be given. Similar to previously proposed ideas, this auto-pilot can be a visual or verbal warning. It is not within the scope of the thesis to determine the proper warning system, but it is important to understand the qualitative data.

Since the approximate time difference between loss of pulse patency and GLOC is 5-7 seconds, the time restraint before the warning needs to be less than the given time window. It is also important to note that it should be more than one or two seconds as well. This will give the system time to resample the signal and

continuing monitoring for pulses. Therefore, for testing, a time of four seconds was chosen before the warning was set, which, for lab purposes, was just a flashing LED.

Figure 2.30 shows the entire final peak block diagram about the previously discussed topics that encompass the peak detect algorithm.

Final Peak Detect Block Diagram

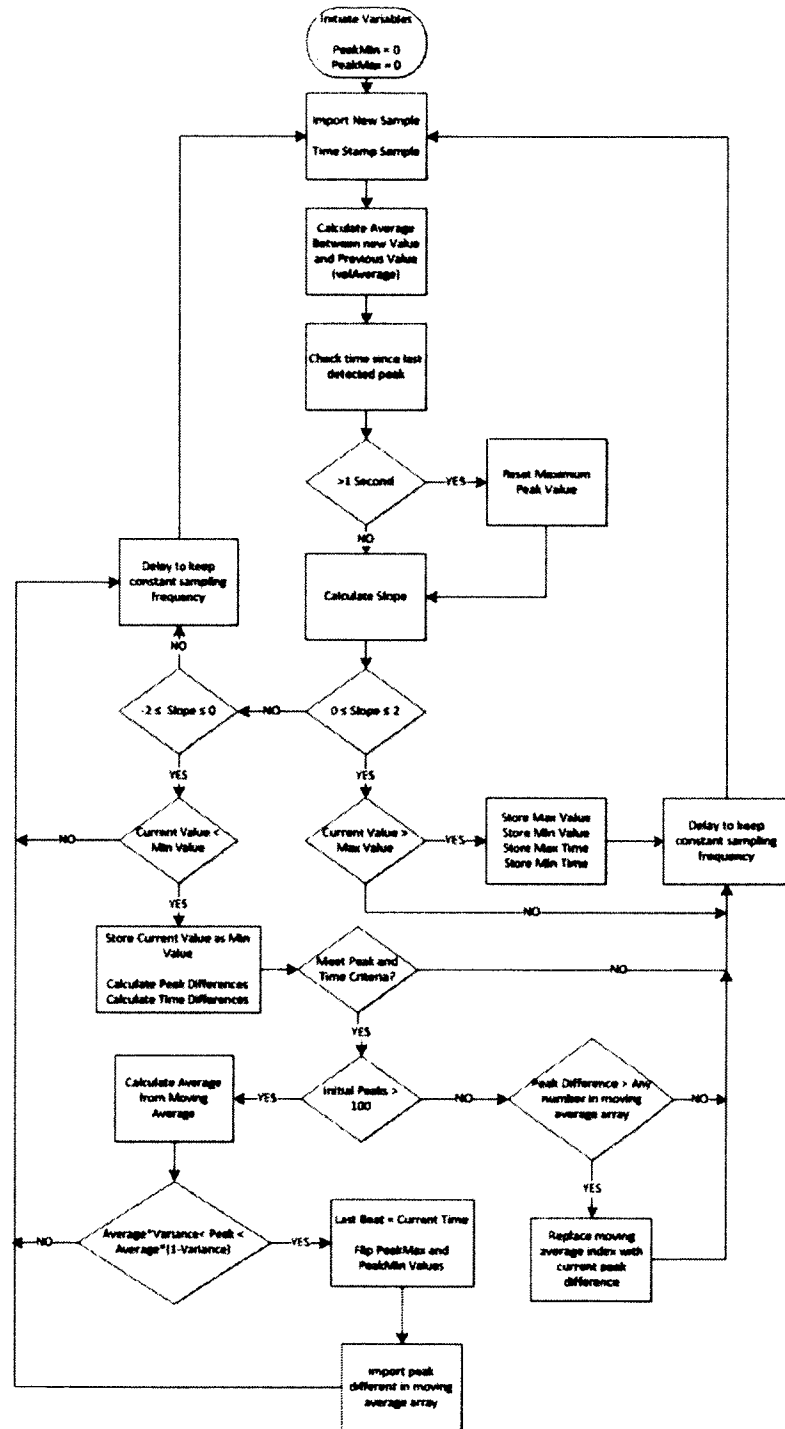


Figure 2.30: Final Code Block Diagram

Pulse Patency Simulation

The simulation portion of the design is an important part of the design. Simulation allows for accurate portrayal of expected lab results in a controlled environment. This section does not discuss circuit simulation, however important that is, but more pulse patency signal and detection algorithm techniques. Using an accurate pulse patency signal, a MATLAB algorithm could be designed to detect peaks in a discrete environment. The algorithm can then be duplicated into Arduino and the results can be expected to be similar.

The waveform generator simulation generates an output with important signal characteristics. Naturally, because it is a digital signal, the amplitude of the waveform can easily be adjusted by just multiplying the array by a scalar value. This allows for accurate representation of the actual waveform. Other important characteristics include: frequency adjustment for testing with a full range of heart beat frequencies, varying decaying values to sample all possible ways pulse patency may decay during +Gz acceleration, and varying sampling frequency signal generation. By allowing varying sampling frequencies, the results can be determined by using different sampling frequencies and their effect on the overall algorithm.

The waveform generator worked well for lower heart rate frequencies. Since the waveform generator was taken from a human subject and just repeated, it was a highly accurate representation of the expected results. However, the waveform generator wasn't accurate at high frequencies (from 3Hz to 5Hz). A MATLAB Simulink model was constructed to simulate an accurate depiction of the expected

waveform. This model produced a highly accurate output waveform compared to that eventually produced by the human model.

Signal Selection

Signal selection is very important. The more accurate the signal that is being simulated, the greater chance that the algorithm tested during simulation will match that to the physical model testing and the human element testing. In order to get the most accurate signal possible, the pulse patency sensor was designed and constructed and its output was monitored on a digital oscilloscope. The accelerometer was placed on the STA of a human subject and the output was stored (Figure 2.31).

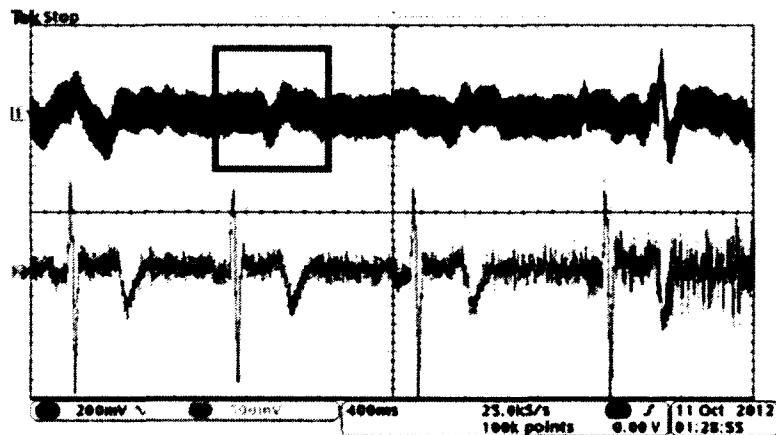


Figure 2.31: Stored output (Top) versus heart beat (Bottom)

Figure 2.31 shows the pulse patency of the human subject versus the heart beat (shown as the bottom signal in Figure 2.31). The approximate 200mS delay between the heart beat and the pulse patency signal is due to the 340mm distance the blood has to travel from the heart to the head [20]. The part of the top signal

located in the highlighted box was determined to be an accurate representation of the pulse patency waveform. As seen, all the signals have the same characteristics. It was determined that the second and the third pulse patency waveform were nearly identical compared to the first and fourth, which is why the second one was chosen.

The signal was isolated in MATLAB and filtered using a digital IIR filter. The resulting signal, which was simulated, versus a physical model is shown in Figure 2.32 a and b, respectively.

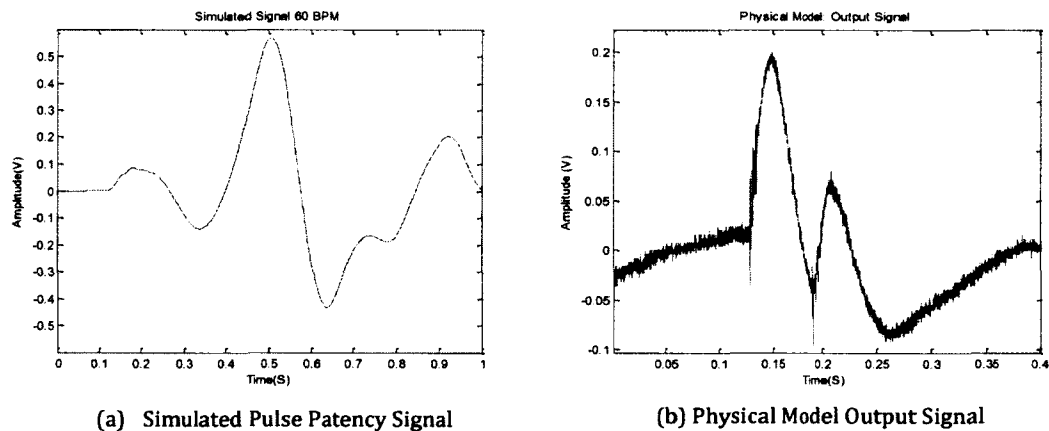


Figure 2.32: Output Signal Comparison between Simulation (a) and Physical Model (b)

Waveform Generator and Decay Windows

The simulation allows for heart rate adjustment, up to 300BPM. According to the frequency of the heart rate, the signal was repeated and time scaled accordingly to accommodate the user input. Figure 2.33 a and b, respectively, shows the first five seconds of a 60BPM heart rate and a simulated 180BPM heart rate.

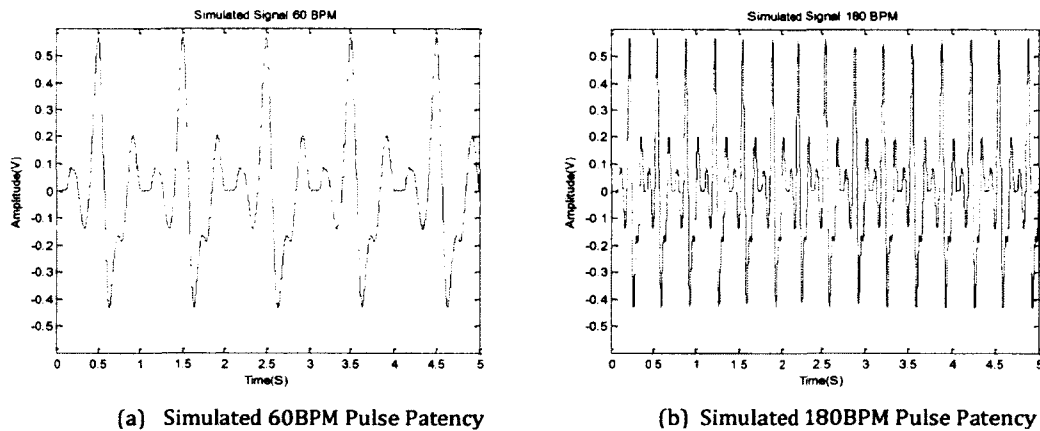


Figure 2.33: 60BPM (a) and 180BPM (b) Simulation shown

After the signal is generated, it is stored in MATLAB as a large array of numbers. However, since the appropriate sampling frequency is a major part of the algorithm, the amount of numbers within the array needed to be precise. Since this is MATLAB, the generated waveform had to be interpolated and decimated according to the appropriate sampling frequency. The amount of numbers in the array can simply be calculated by:

$$\text{Array Length} = \text{Generated Signal Length} * \text{Intended Sampling Frequency}$$

Equation 2.29: Array Length Calculation

When the signal is correctly interpolated and decimated, certain window functions can be applied. This is done by inputting a *g-scale_amp* variable that allows the user the amount of scaling is going to be done in reference to the amplitude of the produced signal. If *g-scale_amp* = 1, then the signal will remain the same. However, *g-scale_amp* is equal to 0.5, and then it will scale the signal by 0.5. The simulation also allows the user to add start and stop times for each window

function. This allows for a controlled simulation of a +Gz maneuver. The exponential window function allows also for a decay rate adjustment. The time constant for the exponential decay window defaults to the difference between the start time and stop time divided by four.

```
%%G scale is the ratio of expected patency between initial
G condition and
%%final G condition. (i.e. assume patency at 2G is 1/2 of
that at 1G,
%%therefore g_scale_amp will equal .5.
g_scale_amp = .5;

%%Input in Seconds
g_time_start = 25;           %%When G Manuever is
conducted
g_time_stop = 35;          %%When G Manuever is
finished
```

Code 2.1: Allows the user to input scaling factor along with Start and Stop times


```

g_time_start = (g_time_start/s_length)*length(time);
g_time_stop = (g_time_stop/s_length)*length(time);

%%Create the Line Window Function
g_line_window(1:g_time_start) = 1;

for z = 1:(g_time_stop - g_time_start)
    g_line_window(g_time_start + z) = ...
        g_line_window(g_time_start+ z - 1) - (1 -
g_scale_amp) / ...
        (g_time_stop - g_time_start);
end

g_line_window(g_time_stop:length(time)) = g_scale_amp;

%%Create the Exponential Window Function
%%Exponential decay 0<g_decay<1

g_decay = 4/(g_time_stop - g_time_start);

g_exp_window(1:g_time_start) = 1;

for l = 1:(g_time_stop - g_time_start)
    g_exp_window(g_time_start + l) = (1-g_scale_amp)*exp(-
1*g_decay*l)...
        + g_scale_amp;
end

g_exp_window(g_time_stop:length(time)) = g_scale_amp;

```

Code 2.2: Creates an Linear and Exponential Window Function

After the code produces the windowing functions, the generated output gets multiplied by the window of interest and produces the final outcome, reference Figures 2.34 and 2.35.

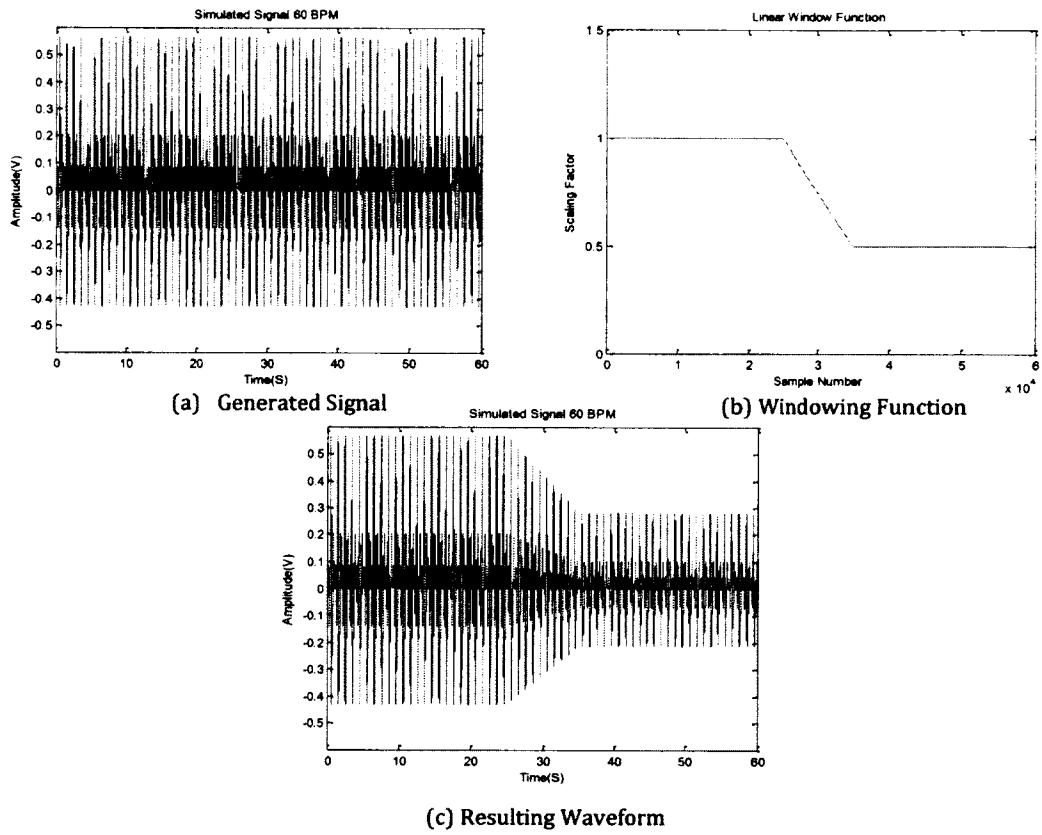


Figure 2.34: Generated Signal (a) multiplied by Window (b) and resulting outcome (c)

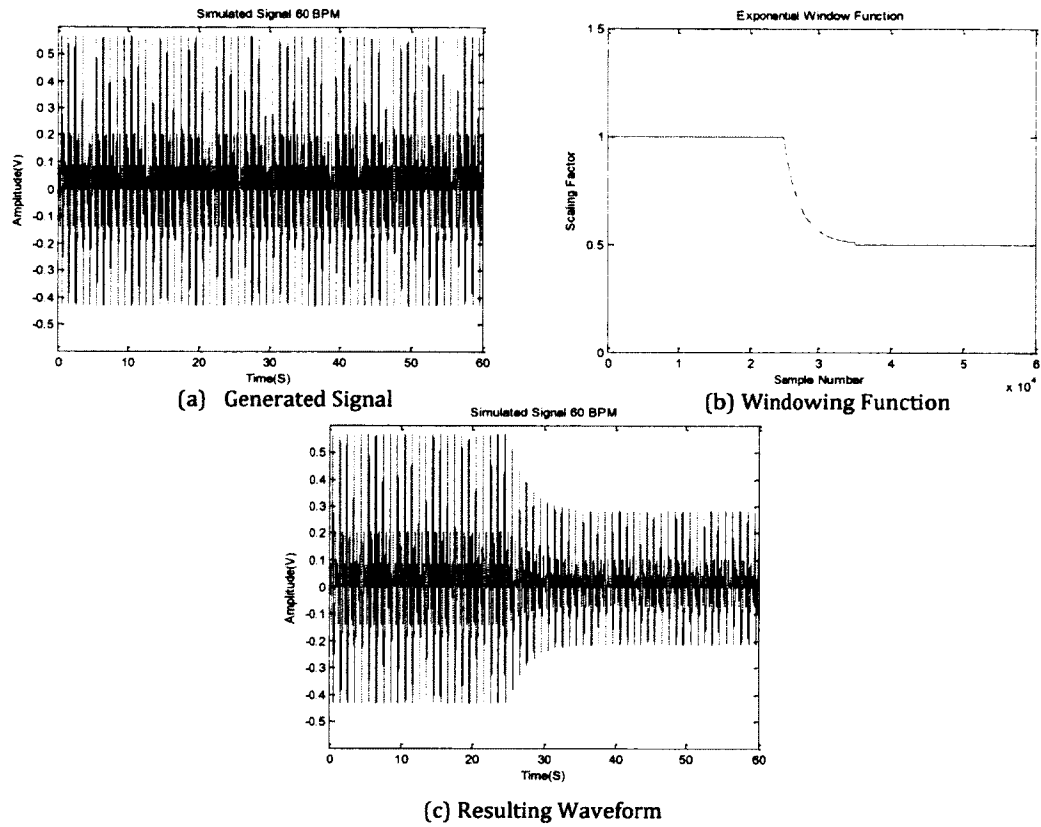


Figure 2.35: Generated Signal (a) multiplied by Window (b) and resulting outcome (c)

To create an immediate loss of pulse patency, the stop value of either window function can just be set to be a fraction of a second greater than that of the start value.

Signal Interpolation and Decimation

The simulated signal had to be interpolated (upsampled) and decimated (downsampled) in order to get the correct amount of sample points per unit time, or sampling frequency. By both interpolating and decimating a signal, simulated signal can be adjusted to meet any sampling frequency. Accurate signal interpolation and decimation has to occur with proper filter settings as to not disrupt the initial simulation signal.

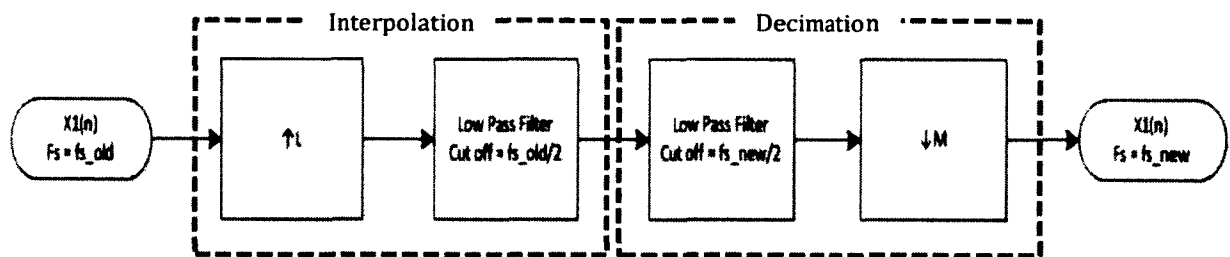


Figure 2.36: Signal Interpolation and Decimation

Represented by Figure 2.36, the signal is first interpolated. This is done by zero-padding the initial signal by $L-1$ zeroes in between sample points. Followed by the zero-padding is a low-pass filter with cut-off frequency equal to half the initial, f_{s_old} , sampling frequency and a filter gain of L . After the signal is fully interpolated, it passes through another low-pass filter with a cut off equal to half of the intended, f_{s_new} , sampling frequency and a gain of M . Finally, the signal is decimated, or desampled, by a factor of M .

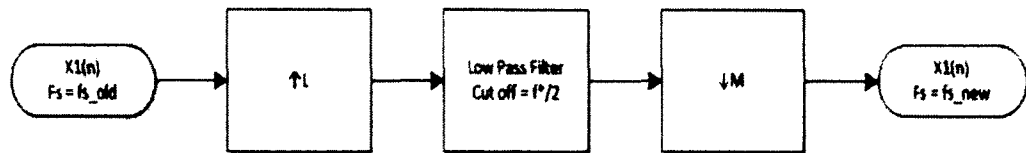


Figure 2.37: Signal Interpolation and Decimation with one Low Pass Filter

The two low-pass filters can be combined into just one filter. However, this step requires a precaution. As seen in Figure 2.37, the cut off frequency of the low pass filter is represented by f^* . The low-pass filter cut off frequency should be equal to half of the *lowest* sampling frequency, whether that is the initial, f_{s_old} , sampling frequency or the intended sampling frequency, f_{s_new} sampling frequency.

Problems with Waveform Generator

Although the waveform generator allowed for the “actual” pulse patency signal to be repeated, it didn’t provide the best possible signal. As mentioned earlier, the sampling frequency needs to be precise due to the slope calculation between data points. At higher frequencies, the waveform generator would have harmonics that would far exceed that of the cut off frequency of the filters and the signal harmonics of interest.

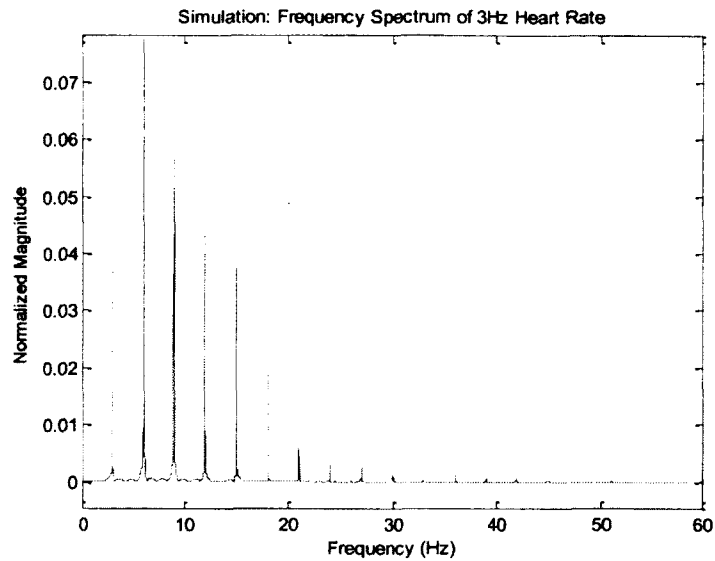


Figure 2.38: Frequency Spectrum of Waveform Generator at 3Hz

Figure 2.38 shows the frequency spectrum of a 180BPM, or 3Hz, pulse patency signal produced by the waveform generator. As seen, the frequency spectrum retains up to the fifteenth harmonic at 45Hz. The maximum slope difference between two points when the highest frequency is 45Hz and the sampling frequency is 1kHz is 40. In order to get the slope back down to 2 or 3, the sampling frequency would have to be around 15kHz. The bandwidth only gets higher with higher simulated pulse patency waveforms, causing bigger error. Therefore, the Simulink model was designed.

Simulink Model

The Simulink model provided the most accurate depiction of the actual pulse patency measurement that was attained when the human model was tested. The

error for the waveform generator wasn't discovered until the physical model testing was completed. So, in order to understand the process used to develop the Simulink model, results from the physical model testing need to be shown (Figure 2.39).

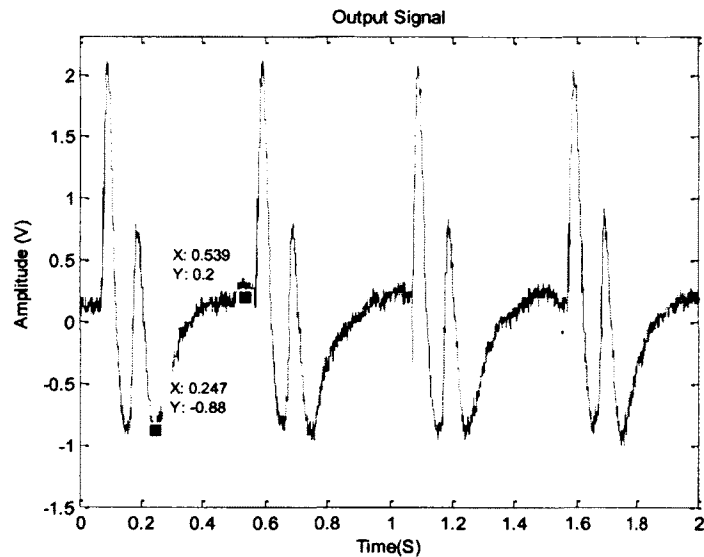


Figure 2.39: Output signal from 2Hz Physical Model Signal

Seen in Figure 2.39 is a 2Hz pulse patency waveform from the physical model. In the figure, there are two data points shown at time equal to 0.539 seconds and time equal to 0.247 seconds. The waveform between those two data points can be simulated as a typical voltage drop across the capacitor in an RC circuit (Figure 2.40). The time constant (τ) would equal to a quarter of the difference between the two values, or 0.75.

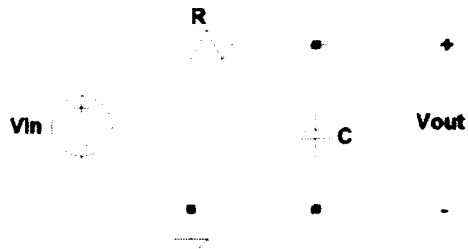


Figure 2.40: RC Circuit with Output taken across Capacitor

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{RCs + 1} \text{ where } \tau = RC$$

Equation 2.30: RC Time Constant Calculation

The input of the transfer function was two pulse waves of which one was phase delayed by the width of the initial. The first pulse was a positive square wave function with a duty cycle of 10% and amplitude of 1. This created the first initial peak, similar to that of Figure 2.39. The second square wave was phase delayed by 10% of the period, had amplitude of -1 and a duty cycle of 10%. The overall circuit produced an accurate depiction of a pulse patency waveform (Figure 2.41) with an very similar frequency spectrum (Figure 2.42)

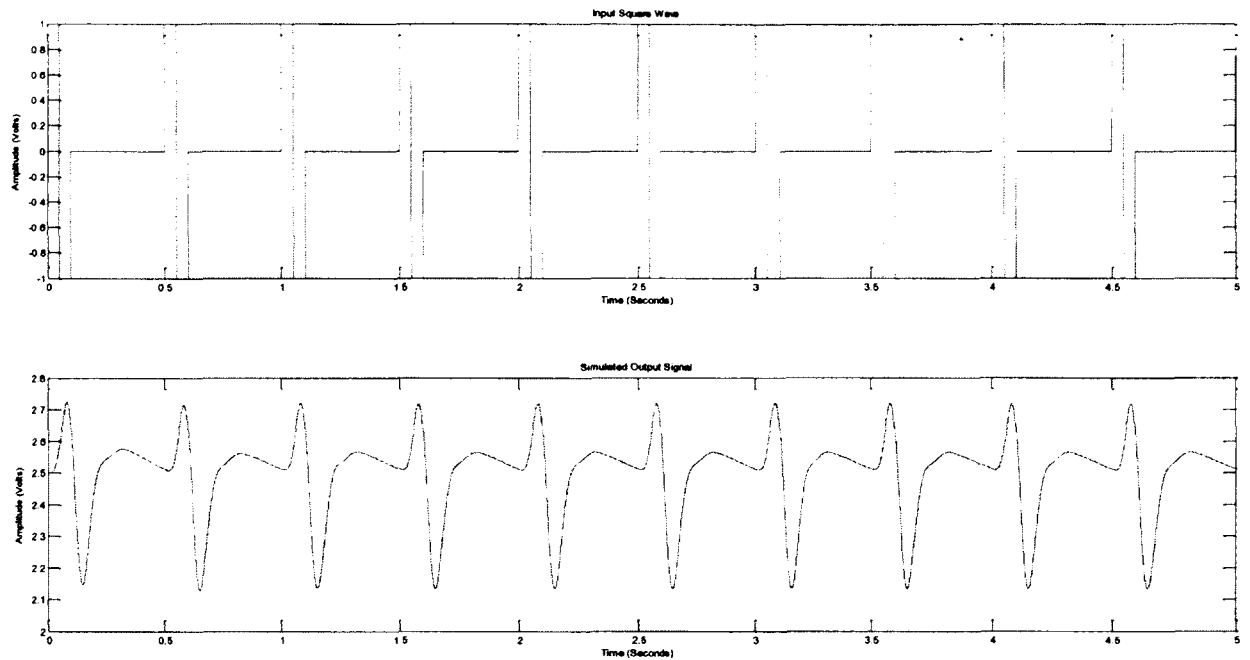


Figure 2.41: Simulink output signal for Simulation

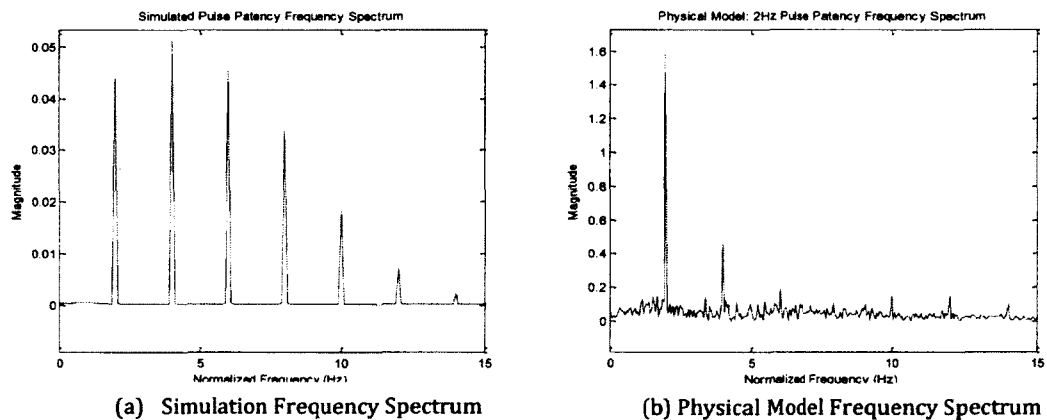


Figure 2.42: Frequency Spectrum of Simulation (a) versus Physical Model (b)

The follow on blocks in the Simulink model (Figure 2.43) are the transfer functions of the high-pass filter and low-pass filter. A DC offset was added at the end through a summation junction. Simulink also allows for the exporting of signal values to the MATLAB Workspace along with sample time step adjustment

(eliminating the need to do signal interpolation or decimation). This allows the signal to still be multiplied by the appropriate pulse patency decay windows to test the algorithm. Throughout all frequencies, this model proved to be the most accurate. All algorithm results were tested according to this output.

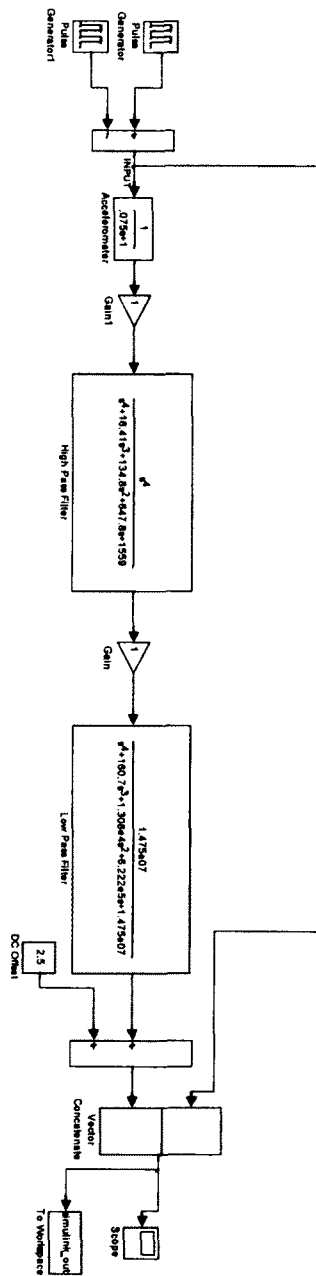


Figure 2.43: Simulink Model with all Transfer Functions

Along with this model of the analog front end circuitry, an automatic gain control (AGC) block was simulated in Simulink to perform the same task as the moving average algorithm (Figure 2.45). An automatic gain control circuit is typically used in communications devices, particularly in AM radio. Essentially, the AGC takes a reference voltage and modulates the amplitude of the incoming signal to match that of the reference signal; by making a weaker signal stronger and a stronger signal weaker (Figure 2.44). By monitoring the gain of the signal, which increases as pulse patency decreases in order to maintain a constant amplitude, a trigger is set once the gain is too high.

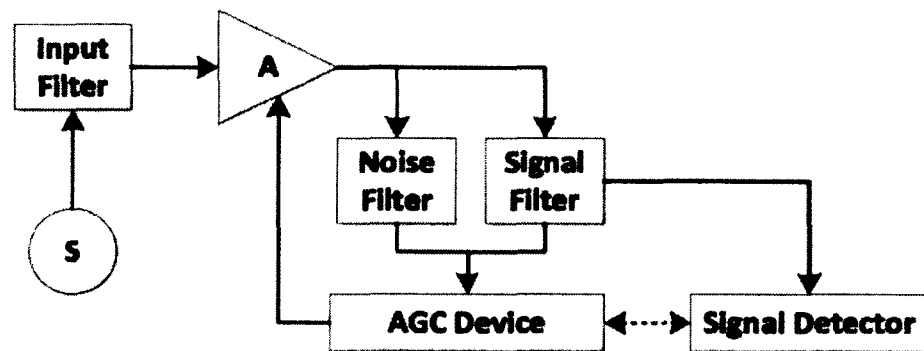


Figure 2.44: Automatic Gain Control Block Diagram

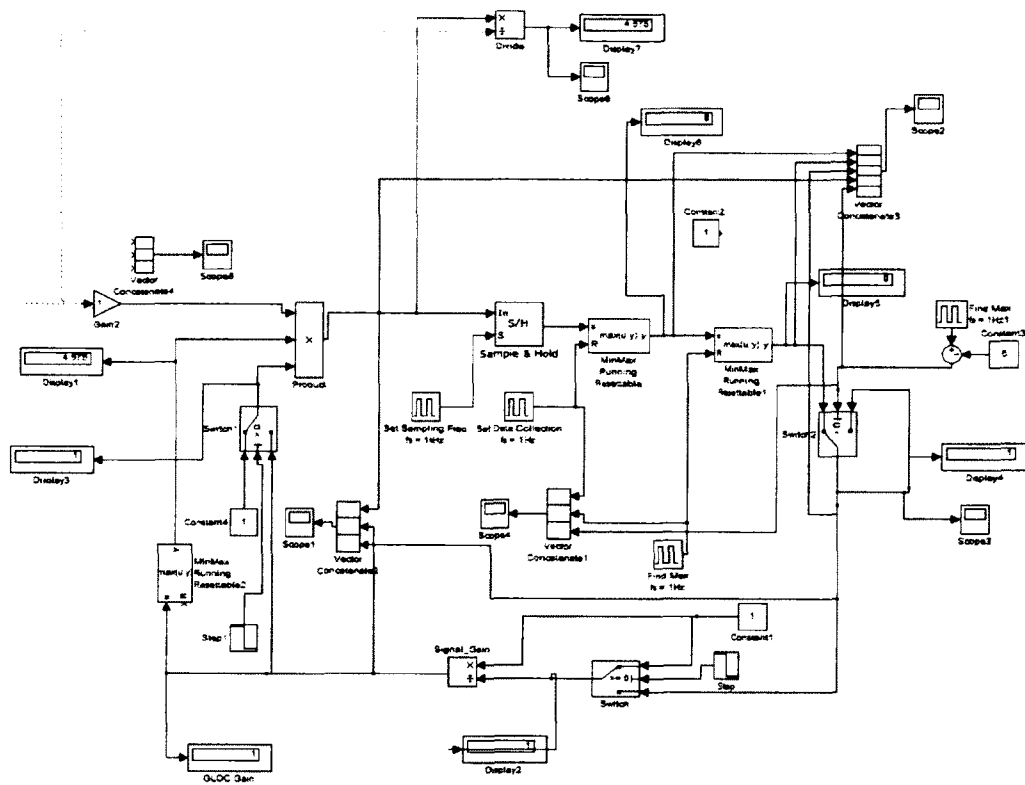


Figure 2.45: Automatic Gain Control Simulation

Once the signal was created by the simulation block in Figure 2.43, a windowing function would be applied to it in the MATLAB workspace. That signal would be then reimported into Simulink into Figure 2.45 along the dotted line seen in the top left of the figure. Finally, the gain could be monitored in order to maintain a constant amplitude.

Pulse Patency: Physical Model

The physical model is an important part of the design process. After the simulation was conducted and a working algorithm was coded, the physical model allowed further adjustment to what can be expected during human testing. The purpose of the physical model is to simulate the human body as accurately as possible. The results from the physical model allows for further design adjustment to meet the prospective needs of the human testing design phase.

Since blood is just a fluid flowing through an artery, there are numerous ways of simulating it. One method would be to utilize a similar fluid, like water, and pump it through expandable tubing to get the correct simulation. However, this process relies on accurate valve control of low pressure water along with the amount of water being pushed through the tubing.

Another method is to utilize air as the fluid. Air valves and air compressors are very common items. Also, and a very important note, is the impulse response of the valve opening has to be ideal. If a fluid, like water would be used, the liquid, unless it was highly pressurized, wouldn't flow in an impulse like waveform. However, due to the quickness of an air valve, this error is eliminated.

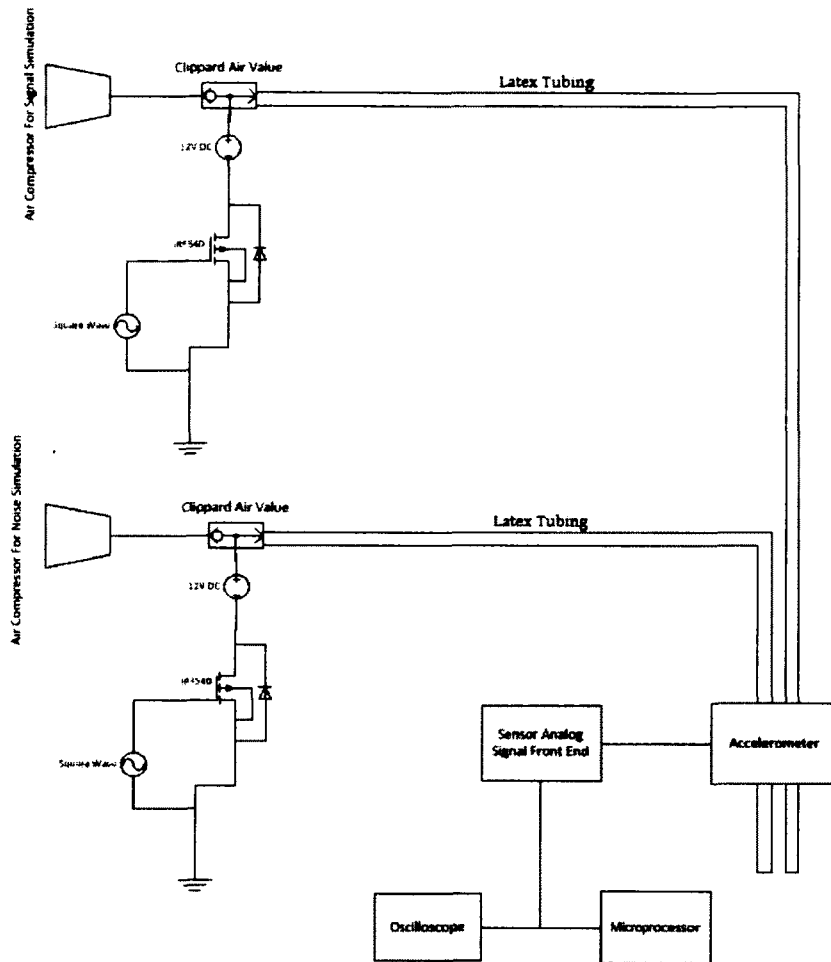


Figure 2.46: Physical Model Setup

The overall system, seen in Figure 2.46, is how the physical model was setup and includes the following: two air compressors, two solenoid air valves, two function generators, two IRF540 power NMOS transistors, a DC supply source, and the pulse patency sensor setup. The IRF540 power NMOS transistors enabled for control switching of the air valves given a pulse waveform. The function generators were unique due to having a pre-programmed pulse waveform. The amplitude of

the function generator was adjusted (5 volt pulse waveform with 2.5 volt DC offset) accordingly to overcome the threshold voltage of the IRF540.

The air compressors that were used were Craftsman air compressors with a 3 gallon tank and 1HP motor. It was adjusted to pump 40PSI worth of air pressure through the Clippard air valves and latex tubing (with measurements 1/8" X 1/32" X 25'). The function generators, actuating the IRF540 Power transistors, were BK Precision 4084 20MHz function generators. The entire tubing length was 3 feet in length with the accelerometer placed half way down the length of tubing. The end of the tubing was left open ended. The end of the tubing can be restricted, but should not be closed off.

Having two air compressors and air valves allowed for the simulation of the pulse patency signal along with the simulation and analysis of system noise. By engaging the second compressor, the testing, calibration, and analysis of the *variance* variable could be conducted. The latex tubing allowed for proper simulation of the STA due to its ability to both expand and contract. Figure 2.32 suggests that the two signals, one from the simulation signal (taken original from the human body) and the physical model signal are very similar.

Valve Characteristics

The air valve had several characteristics that both helped the study in causing a more accurate representation, but also hindered the study by causing

impulse responses seen by the accelerometer. In order to ensure that the signal produced by the accelerometer was representative of the latex tubing expanding and contracting, it is imperative to ensure that there is no ringing caused by the valve opening and closing. As shown by Figure 2.47, a 0.2Hz pulse waveform is shown as the top signal and the response by the accelerometer is shown as the second signal. It is evident that the closing of the valve causes a second transient response by the accelerometer. This is an issue at low frequencies (0+ Hz to 2Hz) due to the duty cycle of the waveform. At higher frequencies (3 Hz to 5 Hz), the two transient responses overlap causing only a single pulse at the output of the waveform.

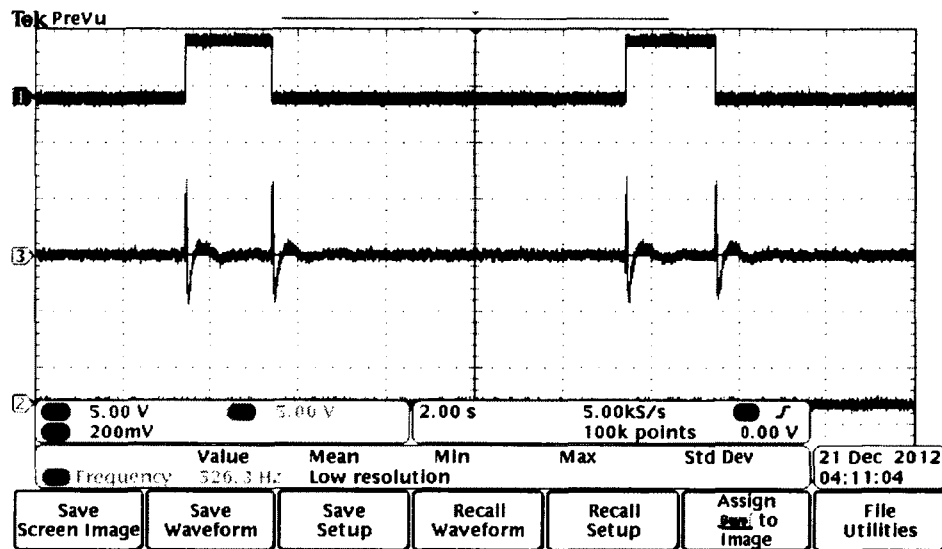


Figure 2.47: Response of sensor (lower) due to valve opening and closing (upper)

As seen by Figure 2.47 there is no ringing caused by the valve opening and closing. In addition, since the impulse responses for the valve opening and closing are identical, it is evident that the momentary oscillation is caused by the expanding

and contracting of the latex tube. This was confirmed also by inserting a 1Ω current sense resistor between the power supply and valve and measuring the voltage across the resistor. By doing this test and analyzing the ringing of the valve opening and closing, it could be concluded that any ringing that was present in the data was not due to the mechanical motion of the valve.

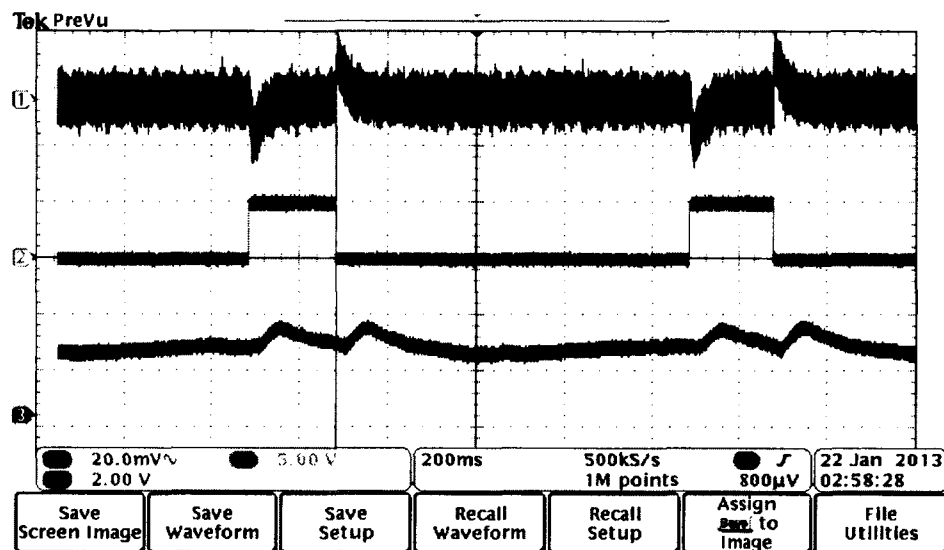


Figure 2.48: Current Sense Resistor Voltage (upper) vs. Function generator output (middle) vs. Accelerometer output (lower)

Figure 2.48 confirms that there is no ringing caused by the valve opening and closing by showing the current through the sense resistor (top signal), the 1 Hz pulse waveform (middle signal), and the accelerometer response (bottom signal). Once again, at 1Hz it is evident that the signal produced by the function generators pulse wave causes two outputs by the accelerometer. This causes the output to count twice, which is an error, but since this doesn't happen within the human body, it can be ignored, assuming the first pulse is detected. During the physical model and

human testing, the output was visualized using an onboard LED located on the Arduino board.

Human Element Testing

The human element testing portion of the design allowed for real life analysis of the overall design. It is important to get an idea of the accuracy of monitoring pulse patency on a human and areas in which the sensor can improve. Throughout the simulation and physical modeling, noise was added to the system to try to understand the tolerances and limitations of the device, but without any human element results, those prior experiments would not produce any good qualitative data.

During the human testing, the pulse patency sensor was placed along the author's STA in the location presented in Figure 2.4. The horizontal location can be approximated as half the distance between the eye and ear, directly on top of the STA. The accelerometer was secured using head phones and the spring constant from the head phones to keep the accelerometer in place. The analog front end gains were adjusted so that the signal became evident (close to 1 volt peak-to-peak under resting conditions). Note that the gain will vary from individual to individual depending on how prominent the STA is in the forehead of the individual. The signal deflection should have a minimum peak voltage of 1 volt, but not more than 2. The DC offset circuit places the DC output of the signal at 2.5 volts and the ADC can't read higher than 5 volts. To ensure that the entire signal always gets read by the

ADC with a buffer area for DC drift or low frequency offset, it is recommended that the signal be closer to 1 volt peak.

In order to increase heart rate, physical exercises were done including push-up to muscle failure. The sensor was immediately placed on the STA at the prescribed position and the output was analyzed.

The output was both monitored with the microprocessor that had an LED output to blink every time there was a pulse. It was also stored on a thumb drive to be processed by MATLAB and further analyzed there.

Pulse Oximeter

The NONIN OEMIII is a commercially available pulse oximeter. The OEMIII module, combined with the Nonin model 8000R Reflectance Pulse Oximeter were used for the testing. This pulse oximeter was chosen for a specifically to provide insight into the testing that was conducted by the US Air Force for BlackHawk pilots at altitude (refer to Chapter 1). The identical sensor was used during this testing in an attempt understand and improve on the results that the Air Force study reported. Naturally, during the testing, there was no access to an altitude chamber to simulate 18000 feet. However, major problems during the testing published in the report by the Air Force were motion artifact and signal-to-noise ratios. The testing done by the Air Force involved placing the Nonin 8000R pulse oximeter on

the forehead. They mentioned briefly that placing the pulse oximeter on an artery on the forehead caused significant noise issues.

The Nonin 8000R pulse oximeter package comes equipped with everything needed to run testing through the laptop. The package includes: a USB Evaluation Board, USB Adapter Cable, OEMIII Module, 8000R Reflectance Pulse Oximeter, and CD-ROM installation disks. It is important to note that the OEMIII module is a small module designed for the purpose of incorporating into sensors (i.e. BOPS System). The entire system can be seen in Figure 2.49.

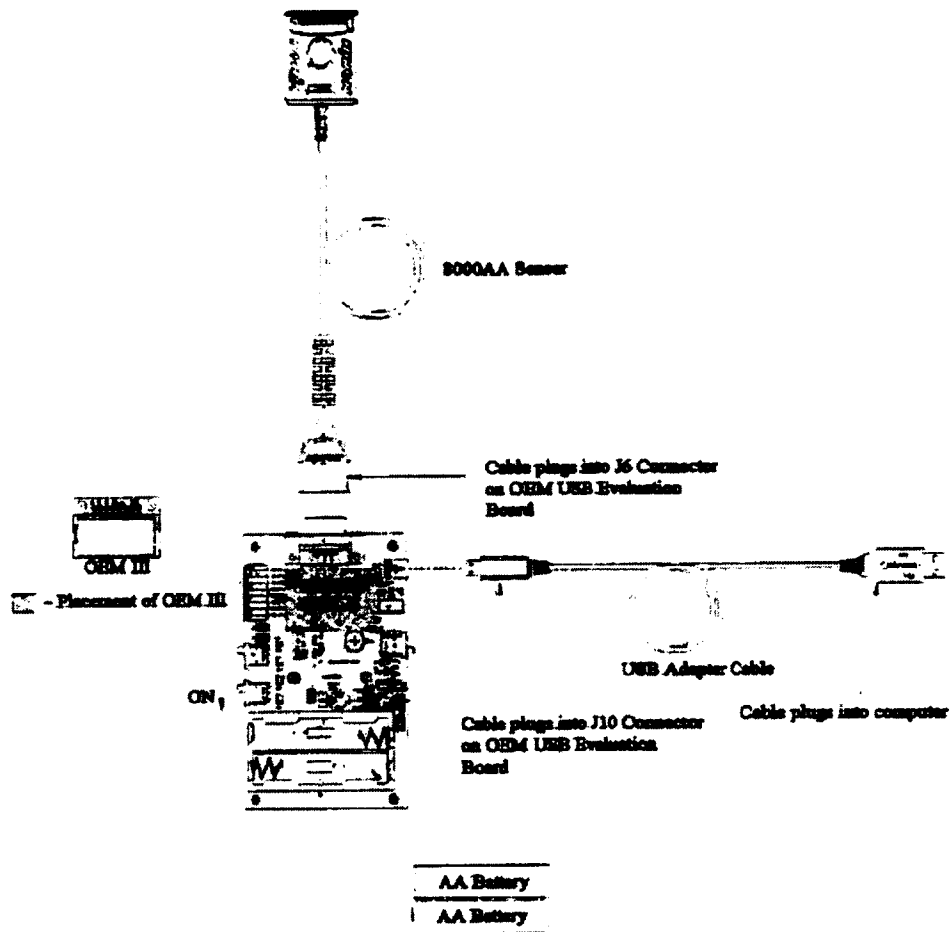


Figure 2.49: Nonin OEM Evaluation Kit (source: www.nonin.com)

The OEMIII is a standalone microprocessor that computes the ratio of red light to IR light. Each sensor is specifically tuned based off of the calibration curves of the company's product. The OEMIII module can be directly connected to the 8000R sensor using a 36 Pin connector that also supplies power to the product. The OEMIII module outputs a serial data stream that is fully compatible with the

Arduino R3 allowing for sampling of pulse oximetry in the four second window between the seizing of pulse patency and the resampling of blood oximetry.

The intent of the testing was to identify the weaknesses in the pulse oximeter. As this is a feasibility design, knowledge in weaknesses and strengths both in sensing pulse patency with an accelerometer and blood oxygenation with the NONIN 8000R sensor is vital. The NONIN 8000R sensor was placed at different location along with the forehead. Those locations include the mid-forehead region, the temple area along the STA, and the post-auricular region. This allowed for data collection at different locations that could be easily used for in-helmet placement of such a sensor.

The testing that was done to the sensor included the sensors response to motion artifact, including head motion and forehead muscle flexing, along with responses to decreases in blood oxygenation. Since there isn't accessibility to controlled oxygen chambers, the author held his breathe for a minute in order to simulate oxygen deprivation. The responses due to lack of oxygen were tracked in each case.

Since blood oxygenation is documented to drop up to 10% during GLOC, it is not necessary to track blood oxygenation to a precise percentage. These tests were done with that concept in mind.

CHAPTER 3:

TESTING AND RESULTS

Introduction

It was initially hypothesized that an accelerometer is able to accurately depict the expansion and contraction of the arterial wall (pulse patency) and is able to be act as the main component to a multiple sensor system (blood oxygenation included) in order to detect or predict G-induced loss of consciousness. Throughout the entirety of the thesis, chapter by chapter, different aspects have been discussed in order to fully explain the problem, the methods to achieve results that will either approve or disprove the initial hypothesis, and now, finally, the results will be presented.

This chapter goes through all of the results obtained through the three phases of design: the simulation phase, the physical modeling phase, and the human body modeling. All of the results are presented and briefly discussed in order to give the reader a thorough understanding of what is being presented. The pulse patency results extensively go through each portion of the design and include verification of robustness and accuracy related to that of the human model. Pulse oximetry

results are presented after all the pulse patency results. These results include placement analysis of the reflectance pulse oximeter along with qualitative data on motion artifact at each location and comparison results to finger pulse oximetry.

All of the code for MATLAB and for the Arduino is attached in the appropriate Appendices.

Design Verification

Verification of all design processes is imperative when testing the peak detect algorithm. The simulation and physical model design processes were specifically designed in order to properly simulate pulse patency on the human temple. The signal produced both by the simulation and the physical model had to match, not only how they appeared, but in frequency content. Since proper sampling frequency is so important to detecting the peak, having the same bandwidth in the simulation and physical model as the human model is critical.

In order to verify the simulation and the physical model, human samples needed to be taken. Naturally, trying to attain a heart rate of 300BPM in a lab is unrealistic. Therefore, numerous samples were taken from the human body (Figure 3.1) and their frequency spectrum was analyzed (Figure 3.2). These heart rates were matched both in simulation and on the physical model and their frequency spectrum was compared.

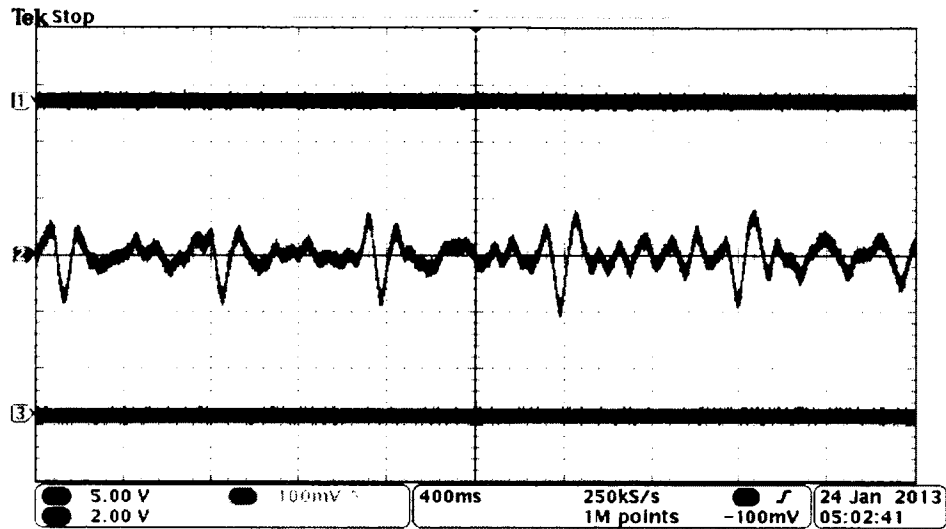


Figure 3.1: Example Human Pulse Patency Waveform

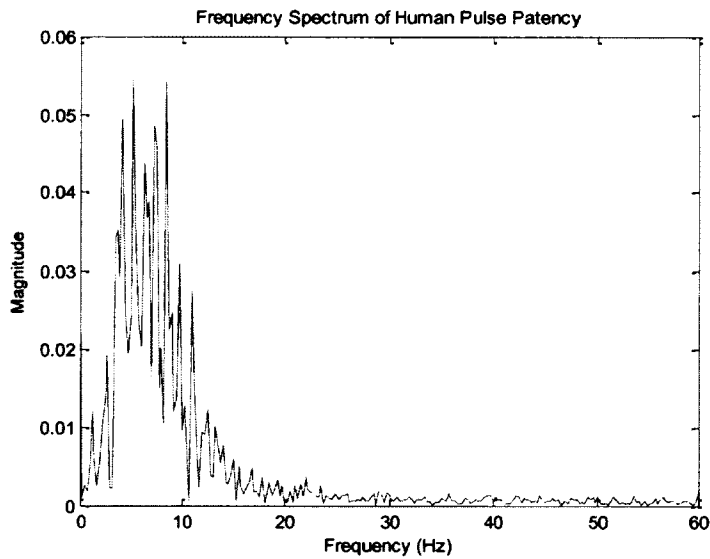


Figure 3.2: Frequency Spectrum of Human Pulse Patency Waveform

Seen in Figure 3.1 is one of the many pulse patency waveforms produced during human testing and its normalized frequency spectrum (Figure 3.2). The results for the frequency spectrum are what are expected. Since this is the human model, there is expected to be noise present. This noise is due to numerous factors.

Of those reasons, the most prominent reason is the transient response of all the circuit elements when the input waveform is a unit step function (used to model pulse patency). Due to the 4th order low pass filter with frequency cut off of 10Hz, there is a significant spectral drop off at 10 Hz, which is evident in Figure 3.2.

As seen in Figure 3.1, the pulse waveform has an approximate period of 0.8 seconds, or 1.25Hz. When a 1.25Hz heart rate, or 75BPM, is simulated, the output of the Simulink diagram is shown in Figure 3.3 along with its frequency spectrum, seen in Figure 3.4.

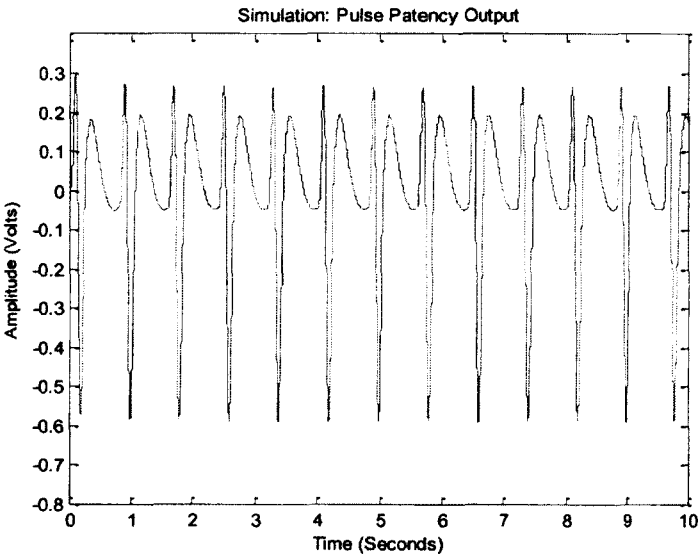


Figure 3.3: Simulink Simulated 75BPM Heart Rate

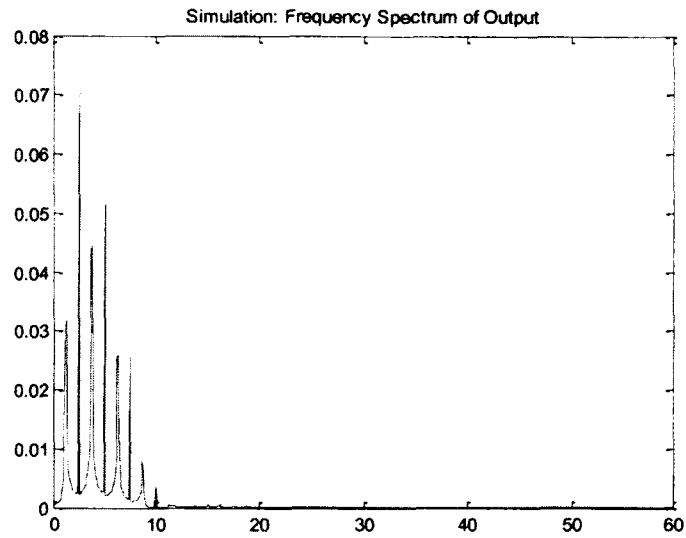


Figure 3.4: Frequency Spectrum of Simulated Waveform

The physical model simulation is also show in Figure 3.5 along with its respective frequency spectrum (Figure 3.6).

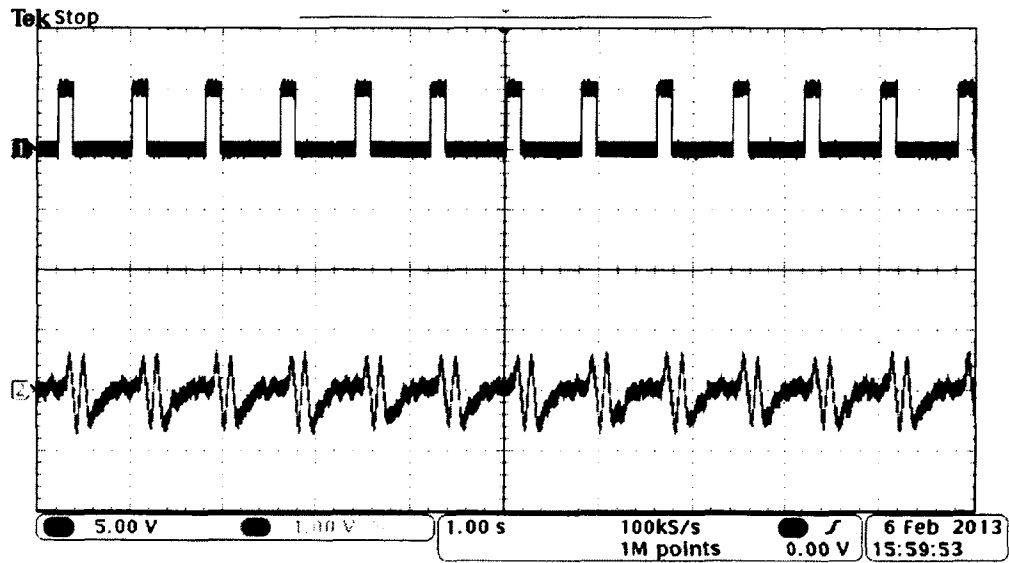


Figure 3.5: Physical Model Simulation of a 75BPM Heart Rate

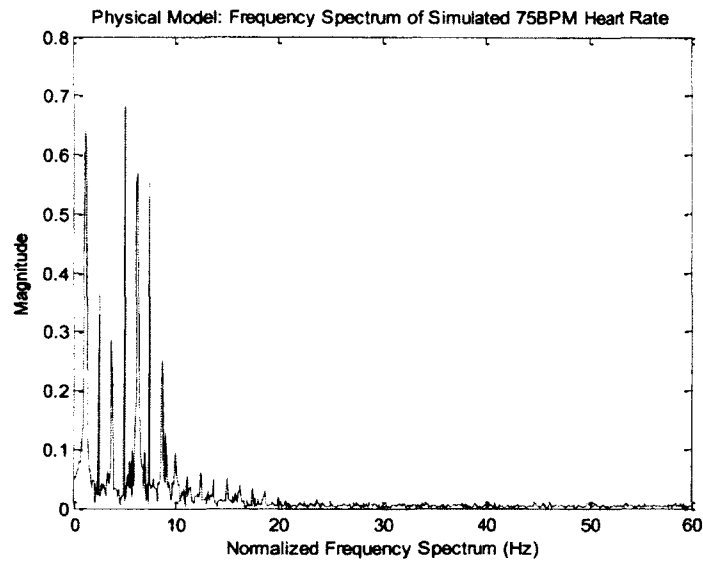


Figure 3.6: Frequency Spectrum of 75BPM Physical Model waveform

As seen through comparison of Figure 3.5 and Figure 3.3 to Figure 3.1, the waveforms produced have very similar characteristics. The initial response by the accelerometer is in the positive direction to indicate the expansion of the artery to allow blood flow. The upward pulse is immediately followed by a negative pulse indicating the contraction of the artery as the blood has passed. Finally, the waveform returns to its steady state value. All of these figures representing the human model, the simulation output, and the physical model output have these signal properties.

Through the comparison of Figures 3.6 and Figure 3.4 to Figure 3.2, the frequency spectrum of each produced waveform is almost identical. Figure 3.6 and Figure 3.2 compare closest due to their noise content within the 10 to 20Hz noise. This noise is very minor compared to the other frequency content and is most likely

correlated with the sudden impulse response of the artery expansion and contraction. Figure 3.4 also has minor noise present between 10 and 20Hz, but most of the frequency content is located lower than 10Hz.

Due to the inability to raise or lower the test subjects heart rate across the entire bandwidth of interest (60BPM – 300BPM), the frequency content of the physical model and the simulation output are shown at each integer number frequency in the bandwidth of interest (1Hz to 5Hz). As seen in Figures 3.7 to Figure 3.11, the frequency content of each are very similar and provide results indicating that the simulation and physical model produce accurate results relating to human pulse patency.

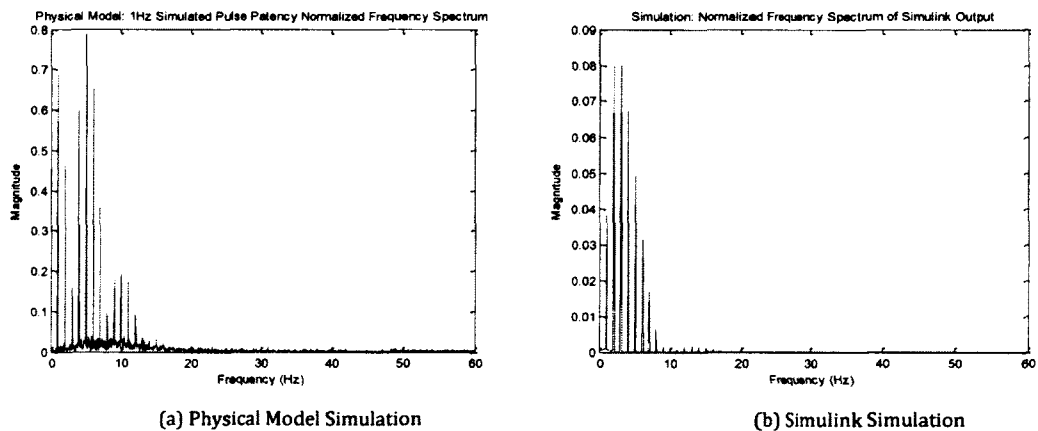


Figure 3.7: Physical Model (a) versus Simulink Simulation (b) Normalized Frequency Spectrum for 1Hz (60BPM) Pulse Patency Measurement

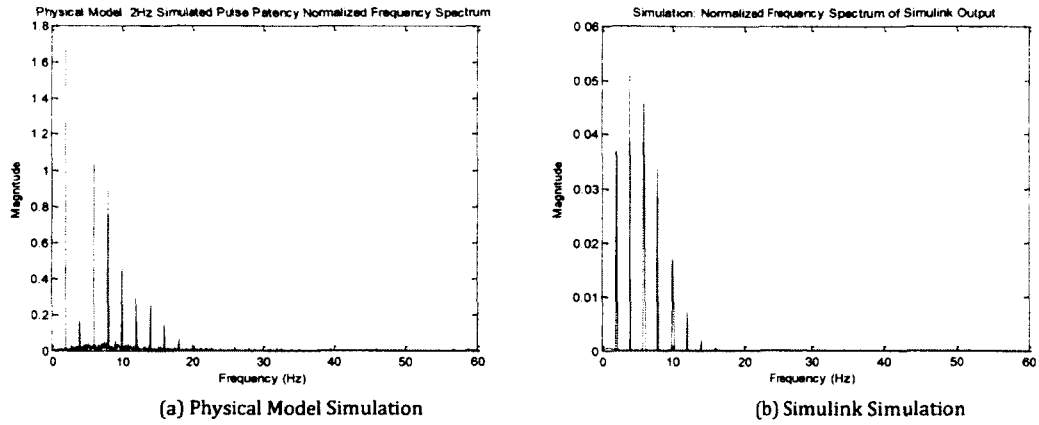


Figure 3.8: Physical Model (a) versus Simulink Simulation (b) Normalized Frequency Spectrum for 2Hz (120BPM) Pulse Patency Measurement

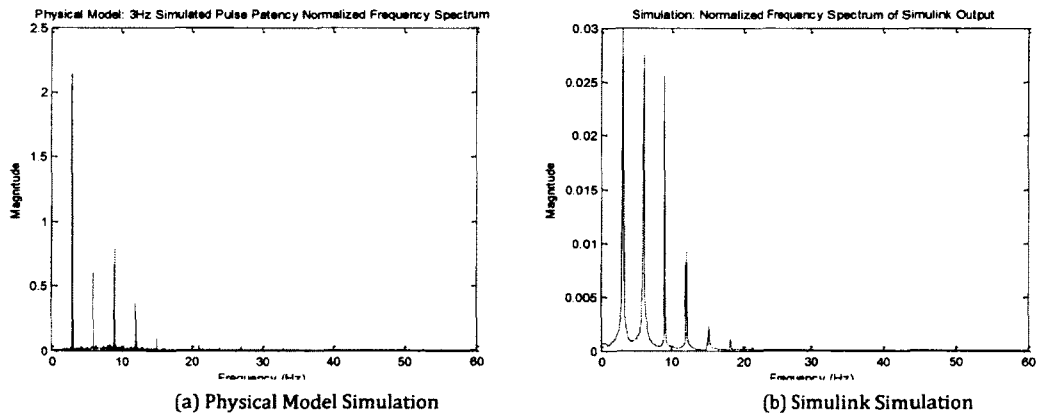


Figure 3.9: Physical Model (a) versus Simulink Simulation (b) Normalized Frequency Spectrum for 3Hz (180BPM) Pulse Patency Measurement

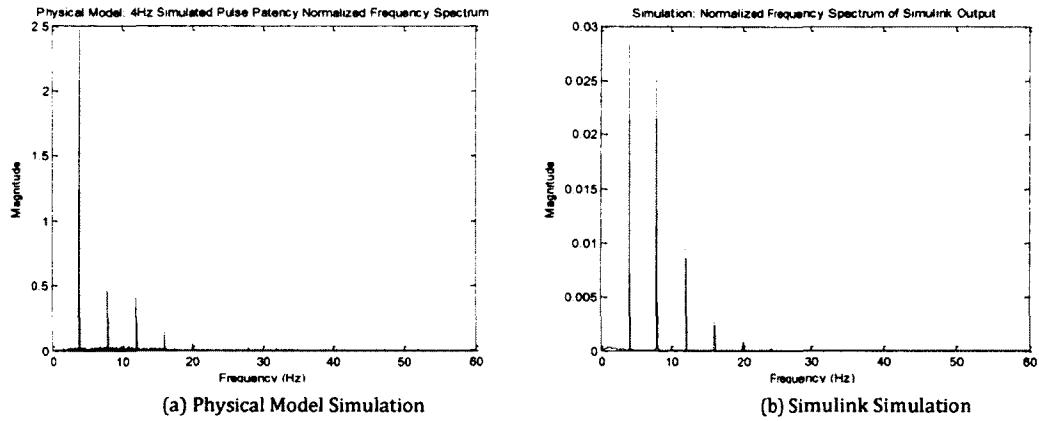


Figure 3.10: Physical Model (a) versus Simulink Simulation (b) Normalized Frequency Spectrum for 4Hz (240BPM) Pulse Patency Measurement

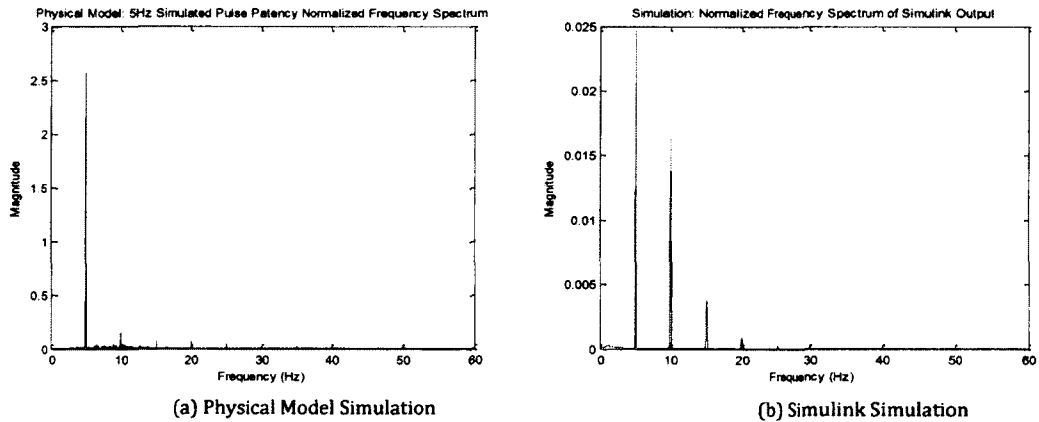


Figure 3.11: Physical Model (a) versus Simulink Simulation (b) Normalized Frequency Spectrum for 5Hz (300BPM) Pulse Patency Measurement

Figure 3.7 through Figure 3.11 show the correlation between the physical model and the simulation model frequency results. Simple observation would show frequency data, in the simulation graphs, out to approximately 15Hz and, sometimes, out to 20Hz. As mentioned before, this is due to the sudden response of

the system to the impulse force which does not influence the peak detect algorithm. The sampling frequency can be adjusted based off of the formula presented in Chapter 2, however, since the frequency at the peak is a low frequency, the results aren't affected by the lower sampling frequency.

Algorithm Verification: Simulation

Before the pulse patency sensor can be mounted on the physical model or even the human model, the algorithm can be simulated in MATLAB using Simulink. Since the MATLAB algorithm is identical to that of the Arduino, or microprocessor, algorithm, if it works in simulation, then ideally, it works during physical model and human testing.

Through simulation, a control heart rate can be established for a certain time along with the tested sampling frequency. The algorithm was verified at the calculated 1kHz sampling frequency. Using a simple calculation (Equation 3.1), the exact number of pulses can be calculated. Having MATLAB count the number of pulses, a percent error can be detected between the expected results and the results shown by the algorithm.

$$\text{No. of Peaks} = \text{Heart Rate (in Frequency)} * \text{Length of Time (in Seconds)}$$

Equation 3.1: Expected Number of Heart Beats

It is important to note, for algorithm verification, the moving average array is initiated by the tester. This allows for accurate verification of the peak detect code

itself. The identical code can then be copied and a moving average initiation portion added to the code to detect the first “X” amount of peaks, or detect all the peaks within a “X” amount of time. By looking at the difference in results, it is evident how many pulses it took to verify all of the peaks. The moving average elements were inputted based off of the simulations’ peak-to-peak voltage output (Figure 3.12). In the case of Figure 3.12, the moving average elements were initiated to 1.05V.

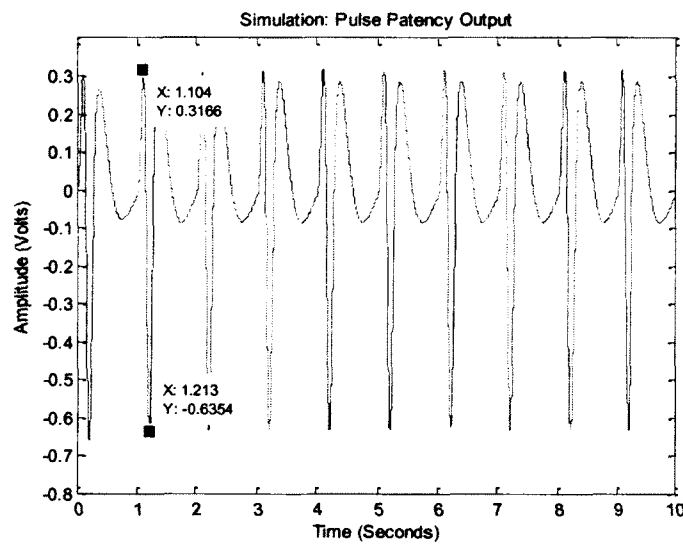


Figure 3.12: Simulated 1Hz Pulse Patency with Peak-to-Peak Voltage

The results, after running it through the MATLAB algorithm display several important factors. Figure 3.13 displays the moving average array throughout the entirety of the sampling. The results show an initial value of 1.05V and a final pulse detection count of 60. Since the test was run for 60 seconds, this yields 100% accuracy for pulse detection.

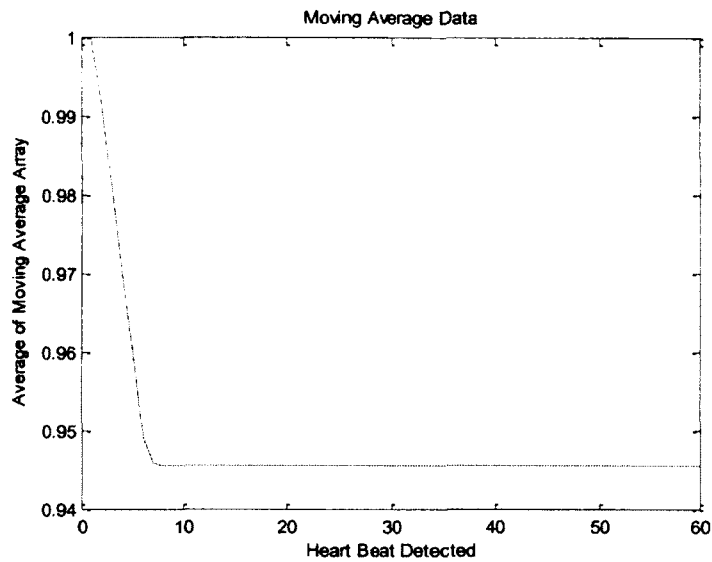


Figure 3.13: Moving Average Array Average across all Simulated Pulse Patency waveforms

Figure 3.14 shows the derivative between each data point. These are important plots due to granting the tester the ability to determine the cause of error. Since this is simulation, and the results are “ideal”, one can adjust the allowable slope to detect peaks. Since the slope calculation for a 10Hz wave yielded a maximum slope of 2, the derivative was left at 2. If the slope was lowered, one would notice a more accurate moving average array average, but would be much more susceptible to noise.

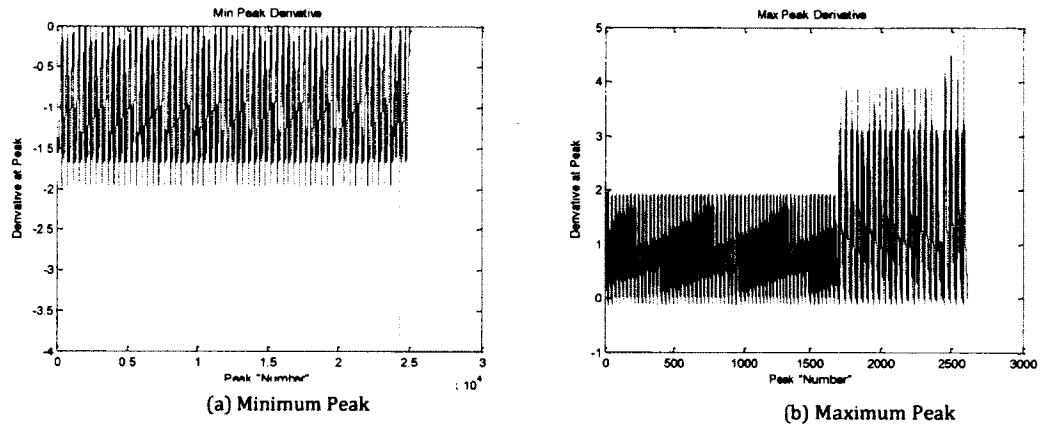


Figure 3.14: Minimum Peak (a) and Maximum Peak (b) Derivative

The self-initiating moving average array average is also plotted across the entirety of the signal. Figure 3.15 shows the average of the moving average algorithm.

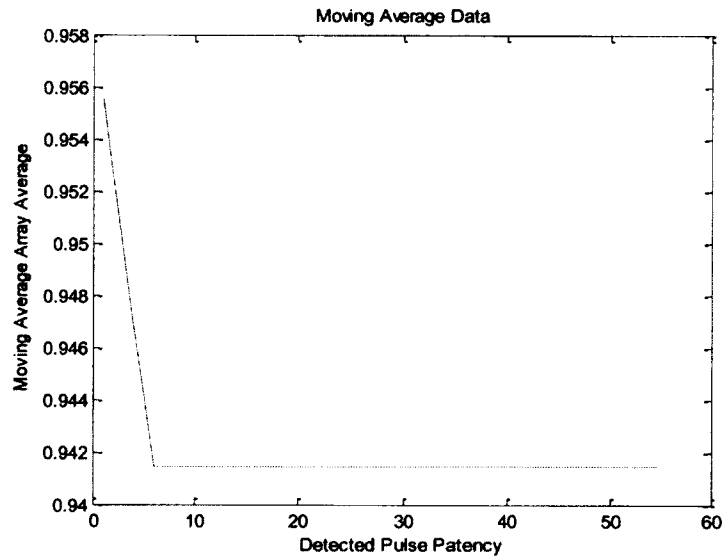


Figure 3.15: Self-Initiating Moving Average Array per Detected Pulse Patency Beat

Table 3.1 shows the results conducted to verify the code. This includes results from implementing the self-initiating moving average algorithm based off of the first 5 seconds of the testing. *Peak Actual Input* refers to tester initiation of moving average algorithm and *Peak Actual Self* refers to self-initiation of moving average array. The results produce perfect accuracy for peak detects.

| Frequency (Hz) | Run Time (Seconds) | Peaks Expected | Peak Actual Input | Peak Actual Self |
|----------------|--------------------|----------------|-------------------|------------------|
| 1 | 60 | 60 | 60 | 45 |
| 2 | 60 | 120 | 120 | 110 |
| 3 | 60 | 180 | 180 | 165 |
| 4 | 60 | 240 | 240 | 220 |
| 5 | 60 | 300 | 300 | 275 |

Table 3.1: Results from Algorithm Verification

The results presented in Table 3.1 yields near perfect results. When verifying the algorithm at 3Hz, the only results with a percent error, since the period of 3Hz is a series of repeating 3's, the resolution can only be so high for Simulink. Therefore, over 60 seconds, there were two extra heart beats. As seen throughout the results, the self-initiating moving average algorithm is always a constant value behind the inputted moving average array code (Equation 3.1). Since the algorithm was set to

detect the number of peaks in 5 seconds to initiate the moving average array, the time would equal 5.

Algorithm Performance: Physical Model

Verification of the algorithm with the physical model presented its own set of challenges. As discussed in Chapter 2, due to the valve closing with the physical model during lower frequencies, there was a second “spike” or “beat” that was produced (Figure 3.16). The output of the physical model was still analyzed using the coding algorithm.

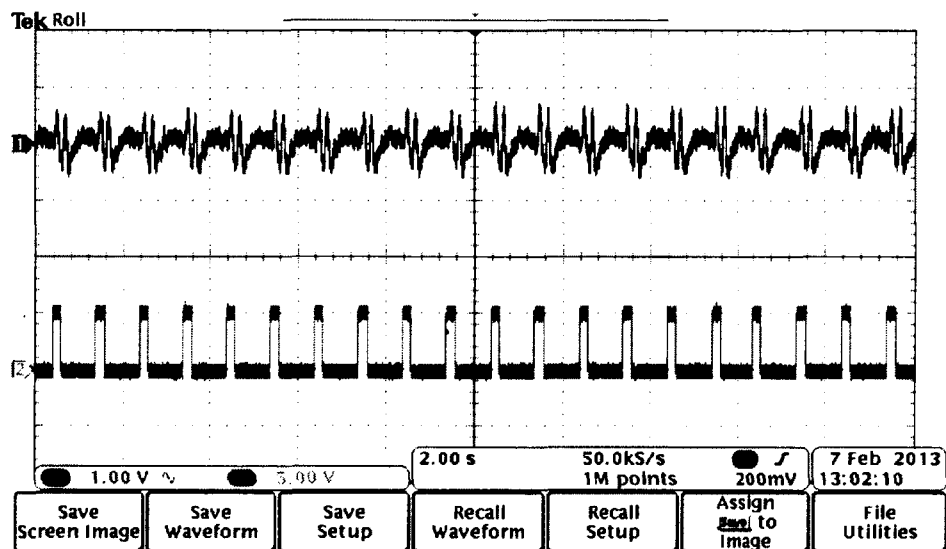


Figure 3.16: 1Hz Pulse Patency (top) Waveform Generated by Physical Model (bottom)

The results are presented in two fold. First, the results were analyzed using the algorithm on MATLAB. This allowed for accuracy detection on a real system. Second, the algorithm was analyzed using the Arduino R3 Uno. Using the Arduino,

however, there was no definitive way of checking if the algorithm on the Arduino was actually detecting the peaks. An LED was used on an output pin of the Arduino and the voltage across the LED monitored. This was displayed on the screen of the oscilloscope. Therefore, the results for the Arduino are presented in qualitative figures, rather than quantitative data.

Physical Model: MATLAB

In this section, the results are presented regarding the algorithm performance when tested using MATLAB. These results aren't taken in from an ideal source anymore, so the results aren't perfect, like they were in the last section. The results are presented in a similar fashion as the last section. The moving average array was both initiated by the tester, which yields the most accurate insight into how many peaks were detected, and then the results are presented with the self-initiating moving average array. A more visual version of these results can be seen in the next section where the physical model is tested with an Arduino.

For example, in the 1Hz pulse patency waveform, seen in Figure 3.16, the expected number of beats is 20 beats for the given amount of time of 20 seconds. The results from the MATLAB algorithm with the user initiated moving average algorithm count 38 peaks (Figure 3.17). This is a large percent error from 20, but, the problem of double peaks has been identified, so the expected answer is actually 40 (percent error is 5%).

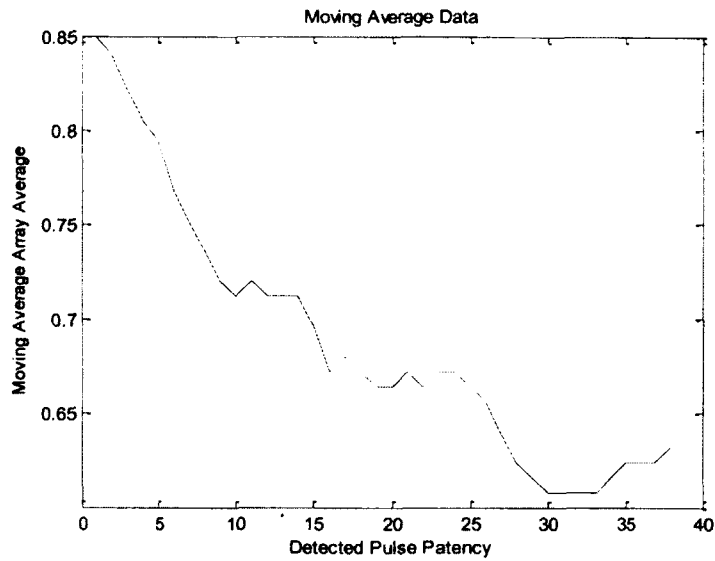


Figure 3.17: Moving Average Array Average at each Detected Peak

Table 3.2 is the tabulated results regarding the peak detects with MATLAB. Since it has been proved that the algorithm has 100% accuracy in Table 3.1, it is expected that these results are accurate. In cases such as the example above presented in the table, it important to recall the double peaks when considering the percent error on some of the results.

| Frequency (Hz) | Run Time (Seconds) | Peaks Expected | Peak Actual Input | Peak Actual Self |
|----------------|--------------------|----------------|-------------------|------------------|
| 1 | 20 | 20 | 38 | 23 |
| 2 | 20 | 40 | 43 | 29 |
| 3 | 20 | 60 | 58 | 43 |
| 4 | 20 | 80 | 85 | 59 |
| 5 | 20 | 100 | 120 | 65 |

Table 3.2: Physical Model Output Analyzed by Algorithm

Table 3.3 shows the percent error between the expected peaks and the amount detected in Table 3.2. It is observed that the self-initiated moving average array is significantly more accurate than that of the inputted moving average array. This is due to the removal of human error. It is also important to note that the percent error calculation for the self-initiating peaks column is slightly altered compared to its counterpart in the Peaks Actual Input column. This is because the expected peaks for the self-initiated moving average array column is a certain amount of peaks less than that (can be calculated using Equation 3.1). So, the “expected” value is less than that of its counterpart.

| Frequency (Hz) | Percent Error | | |
|----------------|----------------|--------------------|-------------------|
| | Peaks Expected | Peaks Actual Input | Peaks Actual Self |
| 1 | 20 | 90.00% | 5.00% |
| 2 | 40 | 87.50% | 12.50% |
| 3 | 60 | 16.67% | 6.67% |
| 4 | 80 | 62.50% | 17.50% |
| 5 | 100 | 20.00% | 13.33% |

Table 3.3: Percent Error of Results in Table 3.2

Physical Model: Arduino R3 UNO

The results presented in this section are similar to the last section. The algorithm was implemented into the Arduino microprocessor and the output signal from the physical model was analyzed with it. Since there is no way of getting accurate quantitative answers using the Arduino, an LED was used as the output. This allowed for visual monitoring of peak detect. These results produced by the

Arduino are visual representation of the data that is produced in Table 3.2 and Table 3.3. The Arduino code that was used was a self-initiating moving average array algorithm.

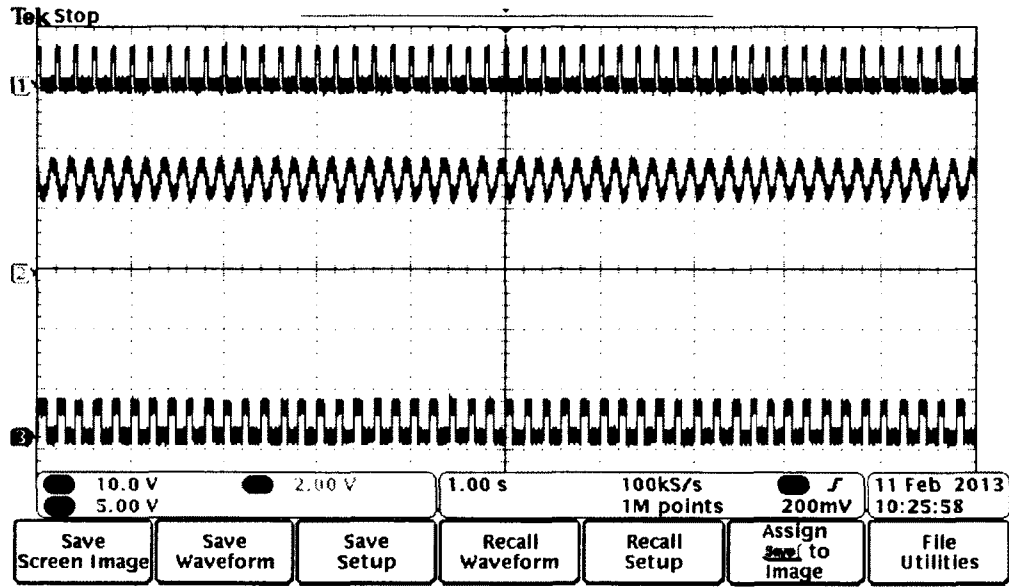


Figure 3.18: 5Hz Pulse Patency Physical Model Results from Arduino Processing (top trace: function generator; middle trace: pulse patency sensor output; lower trace LED voltage)

Figure 3.18 shows the results seen on a digital oscilloscope from a 5Hz simulated waveform produced by the Physical Model. The top signal (marked “1”) is the pulse waveform produced by the function generator. The middle signal (marked “2”) is the pulse patency waveform produced by the pulse patency sensor. The lowest signal (marked “3”) is the voltage across the LED output from the Arduino. The order of the signals on the Figure applies to all follow on Figures (Figures 3.19 through Figure 3.22).

As seen by the results in Figure 3.18, the results are perfect when comparing signal “2” to signal “3”. This indicates perfect detection of pulse patency signals. However, the lower the frequency of pulse patency, the more the “double peaks” showed up in the Physical Model output. Figure 3.19 is a 4Hz pulse patency output; Figure 3.20 is a 3 Hz pulse patency output; Figure 3.21 is a 2Hz pulse patency output; Figure 3.22 is a 1Hz pulse patency output.

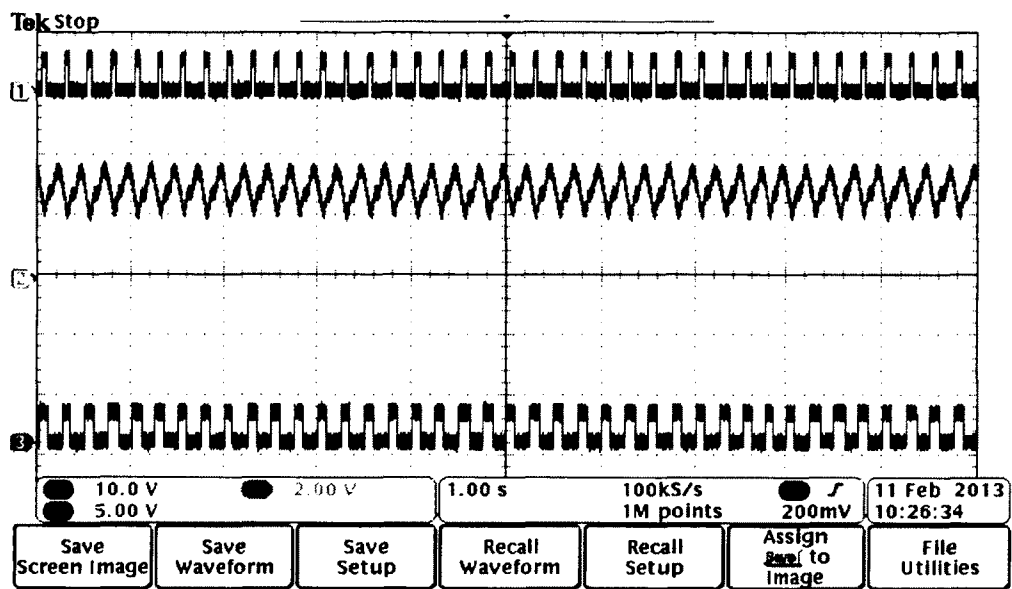


Figure 3.19: 4Hz Pulse Patency Physical Model Results from Arduino Processing (top trace: function generator; middle trace: pulse patency sensor output; lower trace LED voltage)

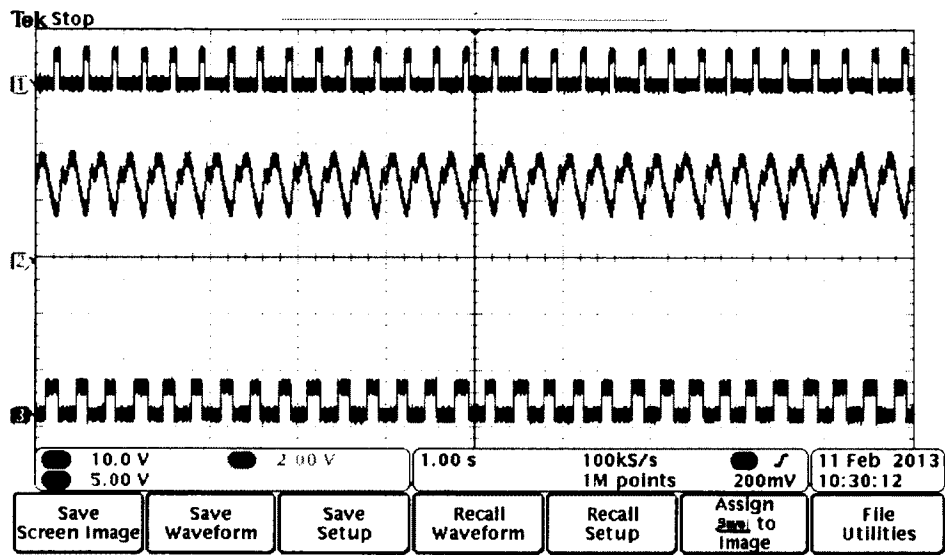


Figure 3.20: 3Hz Pulse Patency Physical Model Results from Arduino Processing (top trace: function generator; middle trace: pulse patency sensor output; lower trace LED voltage)

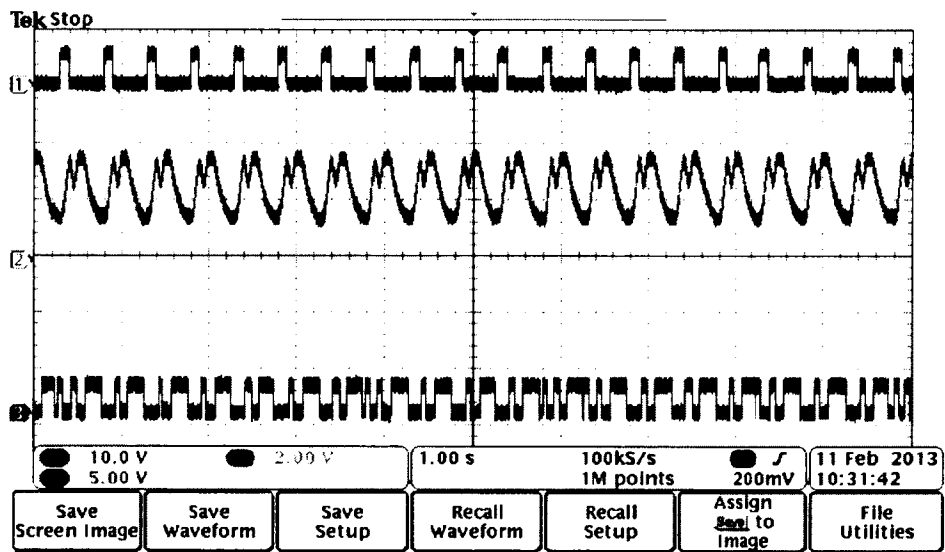


Figure 3.21: 2Hz Pulse Patency Physical Model Results from Arduino Processing (top trace: function generator; middle trace: pulse patency sensor output; lower trace LED voltage)

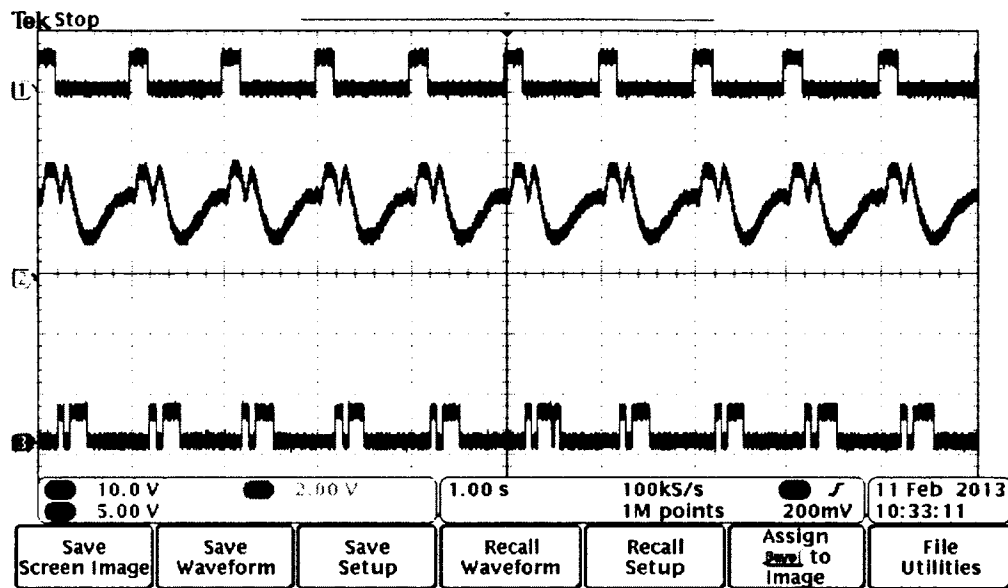


Figure 3.22: 1Hz Pulse Patency Physical Model Results from Arduino Processing (top trace: function generator; middle trace: pulse patency sensor output; lower trace LED voltage)

As seen by Figure 3.18 through Figure 3.22, the lower the frequency, the higher the inaccuracy in the Physical Model due to the double peaks. This is just a limitation of the Physical Model and the microprocessor (further discussed in the following Chapter). The results seen in the above figures are constant with what is presented in Table 3.2 and Table 3.3.

Human Model: Microprocessor and MATLAB

The Human Model testing was conducted over several iterations. Due to the limitations of the technology, the heart rate needed to be increased to increase the outward force produced by the blood flow. In order to do this, the subjects were

asked to perform physical exercise for a minute. The subjects were then seated and the accelerometer placed along the STA at the temple. The output waveform of one of the iterations is shown in Figure 3.23.

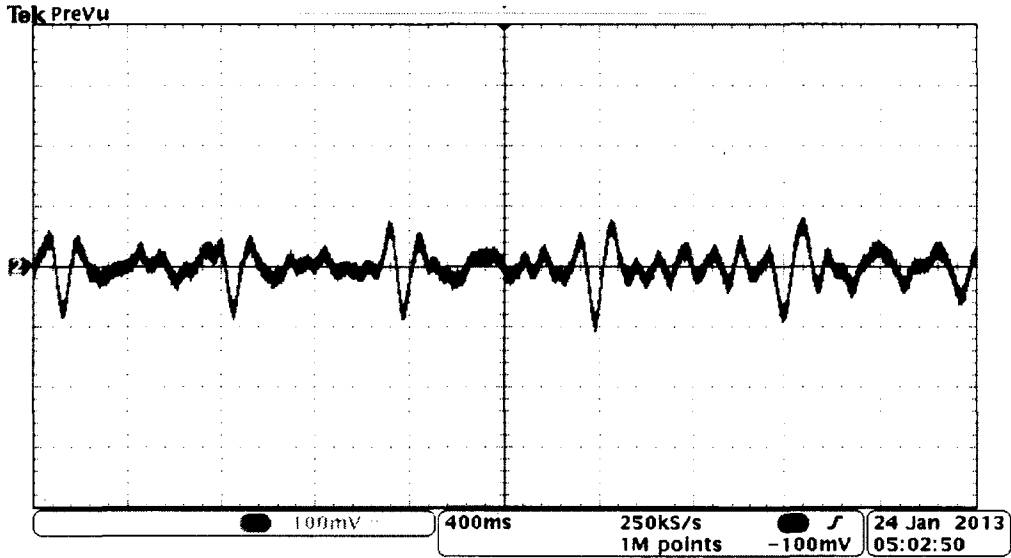


Figure 3.23: Snapshot of Pulse Patency Waveform from Human Subject

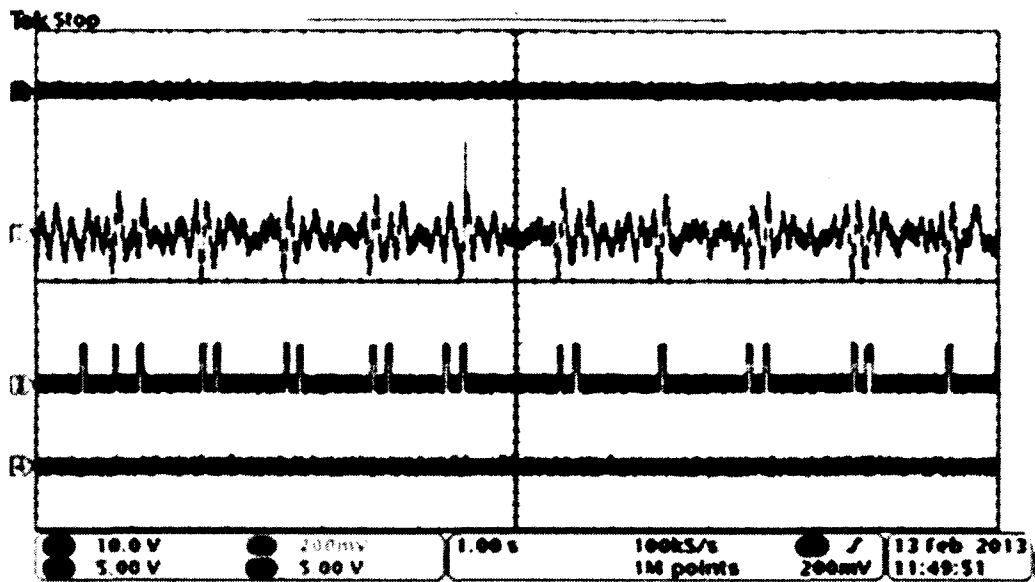


Figure 3.24: Arduino Response to Accelerometer Output from Human Subject (top trace: function generator; second trace: pulse patency sensor output; third trace LED voltage; fourth trace: alarm LED)

Figure 3.24 shows the response of the Arduino (waveform 3) to the input pulse waveform taken from a human subject (waveform 2). At each pulse patency pulse, the Arduino responds by turning on an LED to show detection of the pulse. However, as seen in the figure, due to ringing of the system, the Arduino detects two peaks, similar to what was seen in the lower frequencies of the Physical Model results. An attempt to increase the “time filter” was implemented, but the time difference between current peak and the last peak can be no greater than 0.2 seconds due to the period of 5Hz, which is part of the heart rate bandwidth.

The human simulation trials were processed with identical code in MATLAB. This allowed for a quantitative accuracy analysis of the Algorithm and the Human

Model. The data was taken over a 20 second period. The heart rate of each waveform, similar to that in Figure 3.23, was around 83BPM or 1.4Hz. However, depending on the actual physical exertion of the subject, the heart rates varied. The waveform was plotted and the actual peaks counted by the tester. The results were inserted in column "Peaks Expected". Just the self-initiating moving average array algorithm was tested. Those results are shown in Table 3.4.

| Trial | Run Time | Peaks Expected | Peak Actual Self |
|-------|----------|----------------|------------------|
| 1 | 20 | 25 | 40 |
| 2 | 20 | 30 | 46 |
| 3 | 20 | 40 | 63 |
| 4 | 20 | 30 | 57 |
| 5 | 20 | 30 | 44 |

Table 3.4: Five Trials of Human Model Results

By observation of Table 3.4, it is evident that there is significant difference between the peaks detected and the peaks expected. By observation of Figure 3.24, the algorithm often detects two peaks due to the ringing of the accelerometer,. Note that the algorithm detects the peaks at the correct time; however, the ringing causes it to detect two peaks often instead of just one. Figure 3.25 through Figure 3.29 are the moving average array average plots for all trials (1 through 5).

Taking the double detection of single peaks into account, along with the fact that it takes 5 seconds (or 20% of total test time) for the moving average array to initialize the expected amount of peaks is:

$$\text{Peaks Expected} = (\text{Counted Peaks} * 80\%) * 2$$

Equation 3.2: Expected Peaks for Self-Initiating Algorithm

The percent error for Table 3.4 is presented in Table 3.5 with Equation 3.2 into consideration.

| Trial | Percent Error | | |
|-------|------------------------|-------------------------|-------------------|
| | Initial Peaks Expected | Computed Peaks Expected | Peaks Actual Self |
| 1 | 25 | 40 | 0% |
| 2 | 30 | 48 | 4.2% |
| 3 | 40 | 64 | 1.6% |
| 4 | 30 | 48 | 18.7% |
| 5 | 30 | 48 | 8.3% |

Table 3.5: Percent Error of Results from Table 3.4

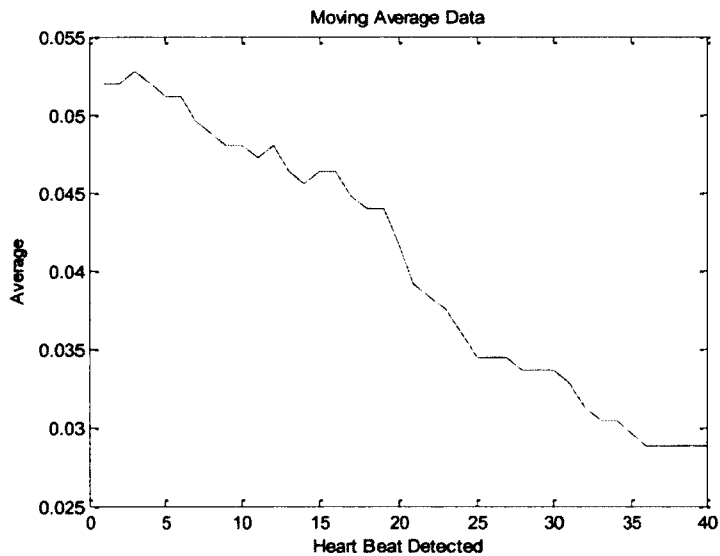


Figure 3.25: Moving Average for Trial 1

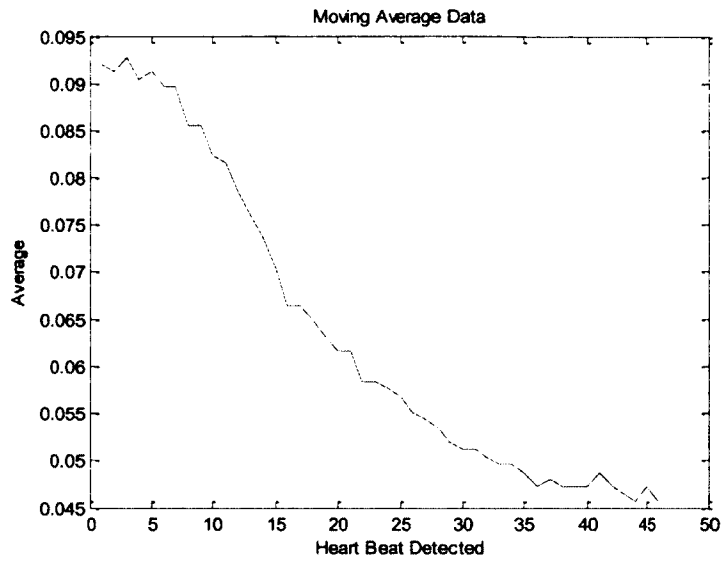


Figure 3.26: Moving Average for Trial 2

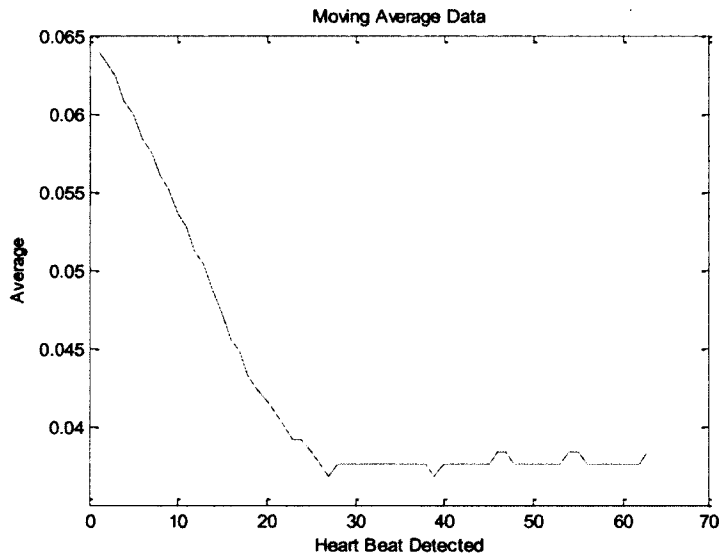


Figure 3.27: Moving Average for Trial 3

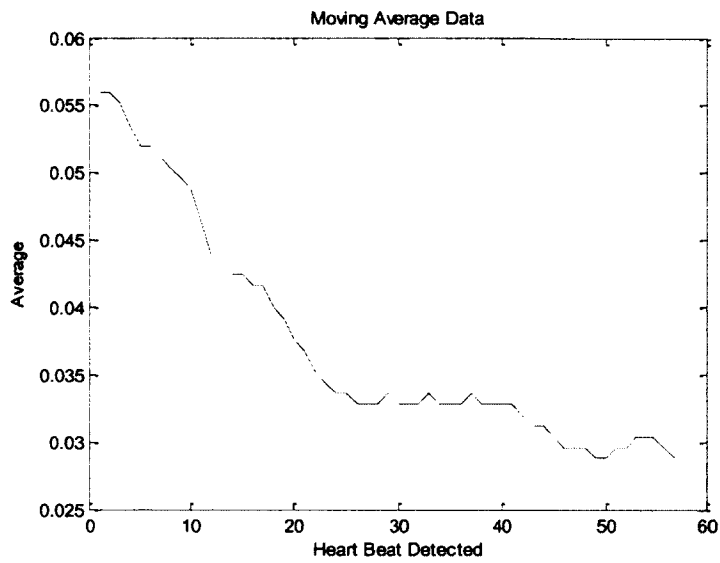


Figure 3.28: Moving Average for Trial 4

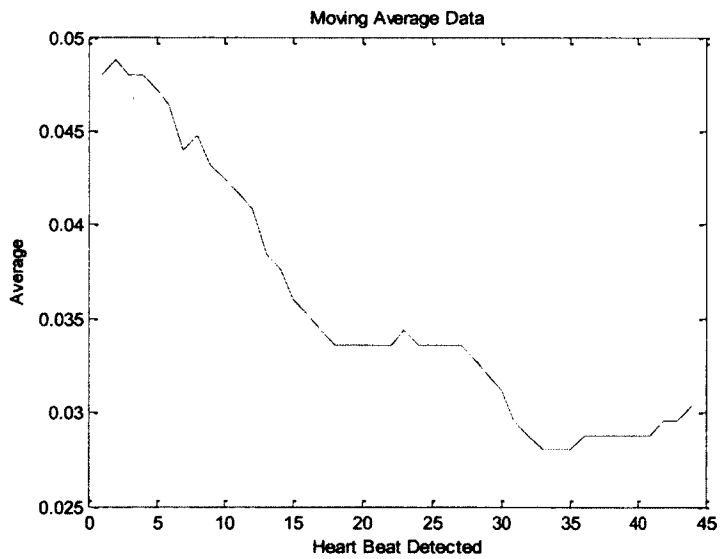


Figure 3.29: Moving Average for Trial 5

Seen in the figures above, the general trend of the moving average array can be modeled using a very basic exponential graph. The initialization of the moving

average array along with the variance variable causes the array to decay until the average finds a “steady state” value. Due to the complexities of ranging pulse patency forces acting upon the accelerometer, it is difficult to get a constant value, which is why a more general decaying exponential plot can be used to model the moving average array.

Simulation: G-Profile Simulation

In order to fully explore the entire algorithm and to diminish redundancy, there will be multiple aspects of the algorithm that will be presented to provide insight into the algorithm at a constant heart rate frequency. All of the simulations were done using the Simulink simulation at a heart rate of 120BPM, or 2Hz, and was analyzed using the self-initiating moving average array algorithm. The simulation was run for 60 seconds; therefore, the expected number of peaks is 110. As presented above, the accuracy detection for the simulation is 100% accurate. Therefore, discrepancies in results presented in this section will be due to the explored aspects of the algorithm. The aspects that will be explored are: moving average array size variation and variation on simulated G-induced reduction in pulse patency. Ideally, the system should be able to detect all peaks, and set an alarm after a specific amount of time has expired, a recommended 4 seconds. Figure 3.31 shows the microprocessor detecting all peaks produced by the physical model, and engaging an alarm, the bottom signal, after an approximate 4 seconds after the

detection of the last peak. This represents the activation of an auto-recovery mechanism and/or voice interrogation.

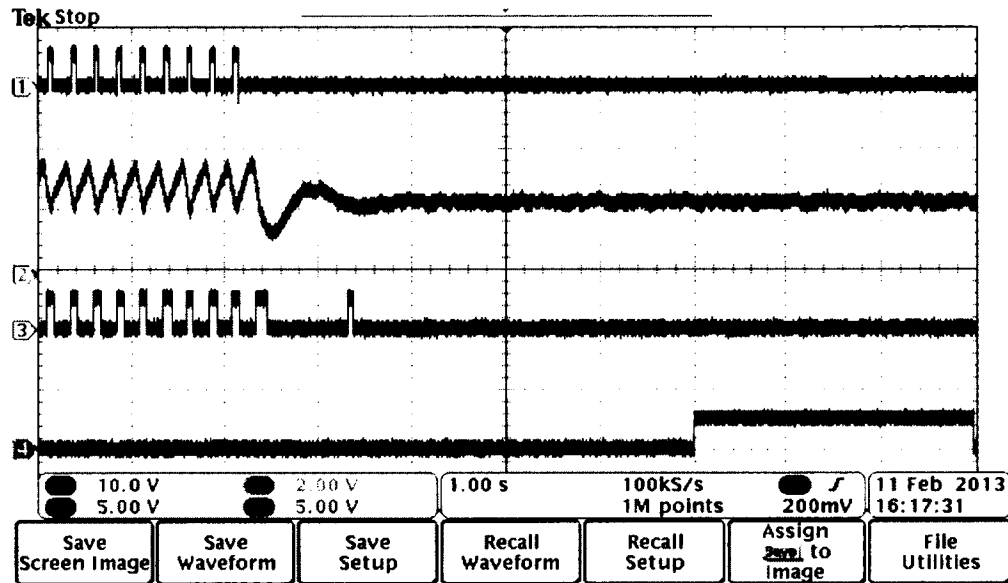


Figure 3.30: Alarm Engaging after Set Time since Last Pulse Detect (top trace: function generator; second trace: pulse patency sensor output; third trace LED voltage; fourth trace: alarm LED)

Variation in Moving Average Array Size

Without question, a physiological system that monitors consciousness in a pilot needs to respond quickly, but must maintain a certain level of accuracy. However, one can't have both. A larger moving average array size allows for the most accurate detection of pulse waveforms. Simply, the more subjects that are part of a study, the more representative of the whole the study becomes. However, a change in pulse patency, with a large array, would require more time to respond

than a smaller array size. Therefore, experimentation needs to be done, once this system is implemented in real life, to best suit the needs of the pilot and GLOC detection system. The following results will only give insight into the effect of array size.

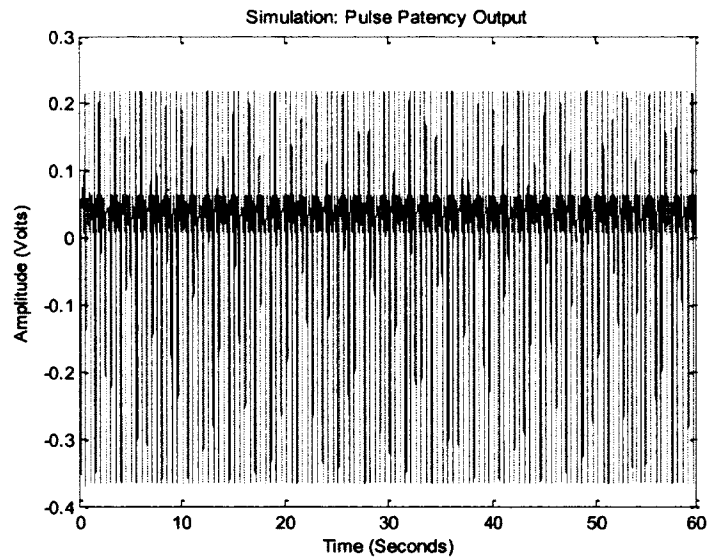


Figure 3.31: Simulated 120BPM (2Hz) Pulse Patency Waveform

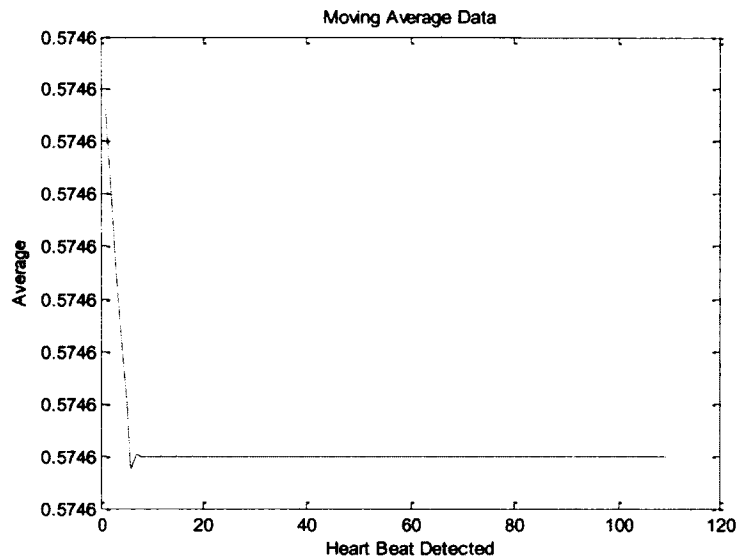


Figure 3.32: Initial Moving Average Data for Simulated Pulse Patency Waveform

Figure 3.31 and Figure 3.32 show the simulated 120BPM waveform along with the analysis from the algorithm. As seen, although the decay isn't as drastic, the model of exponential decay with the moving average array is similar to that from the results in the section *Human Model: Microprocessor and MATLAB* (Page 123). The first window that will be applied is the linear decay window. The window starts at 25 seconds and stops at 35 seconds. The final scalar value from the window is 0.5; which shows a 50% drop in pulse patency force in 10 seconds (Figure 3.33).

For future reference, "elements" refers to the size of the moving average array size. For all of the results and testing up to this point, the size of the moving average array has been 5 elements.

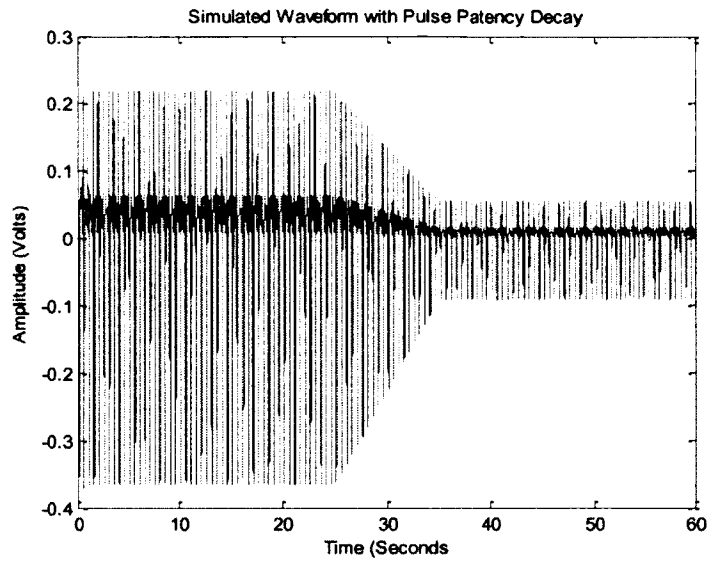


Figure 3.33: Simulated Pulse Patency with Linear Decay

Figure 3.34 shows the moving average array response to the change in pulse patency and Figure 3.35 is an enhanced view showing the decay of average values. As seen in the comparison, the 3 element array responds quicker than the rest of the arrays. The 7 element array is 4 heart beats slower than the 3 element array, which is 2 seconds worth of heart beats. In a restricted amount of time (i.e. 5-7 seconds of reserve oxygen), a difference of 2 seconds is nearly 30% of the total system reaction time.

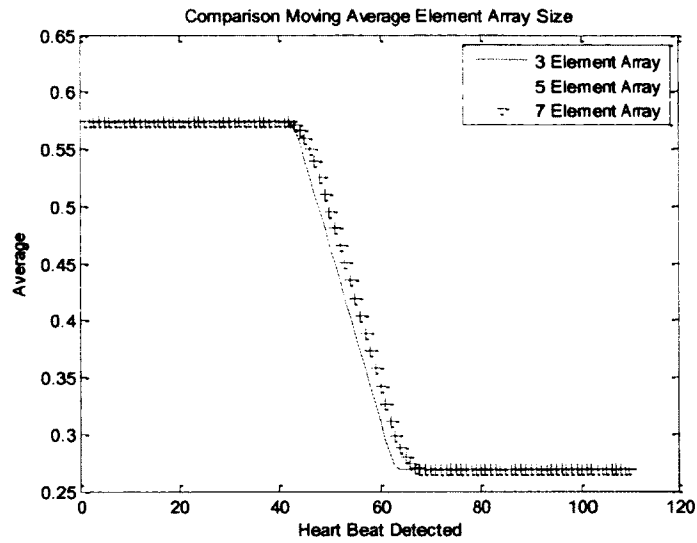


Figure 3.34: Moving Average Array Response to Pulse Patency Loss

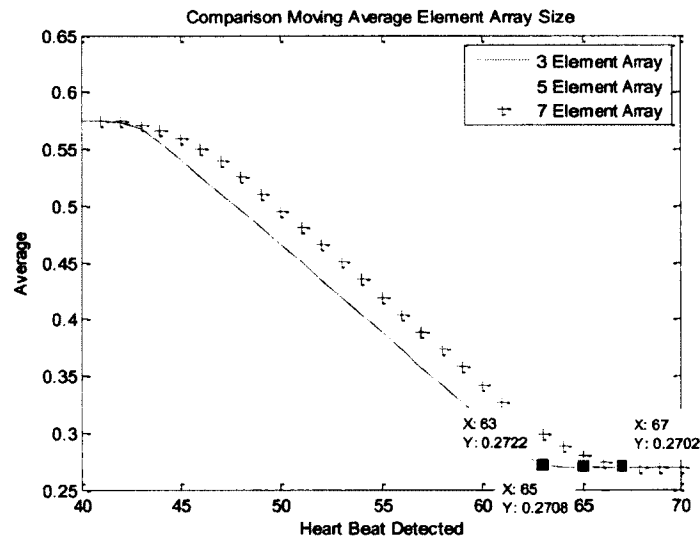


Figure 3.35: Enhanced view of Moving Average Array Response

Figure 3.36 shows another scenario where pulse patency drops off even quicker. In Figure 3.36, the linear decay starts at 30 seconds and decays by 50% up to 35 seconds. As seen by Figure 3.36 and Figure 3.37, the more elements present in

the array, the slower the response time. This may cause the array to detect a loss of pulse patency, which, such as in this case, it will be a false positive.

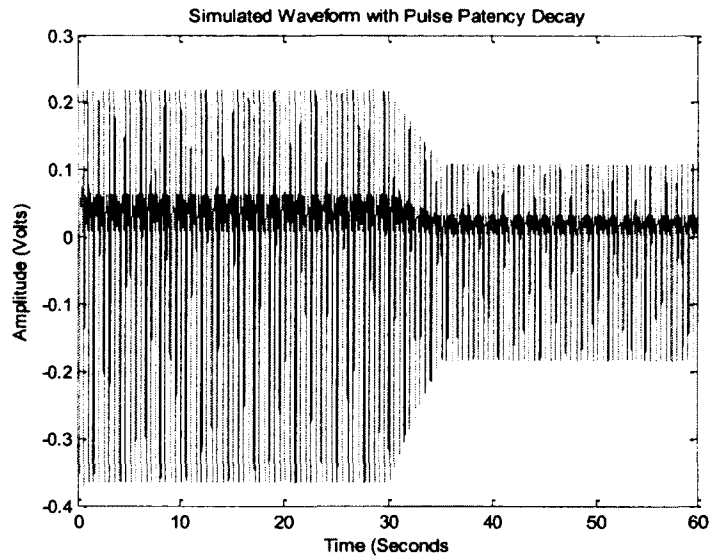


Figure 3.36: Simulated Loss of Pulse Patency

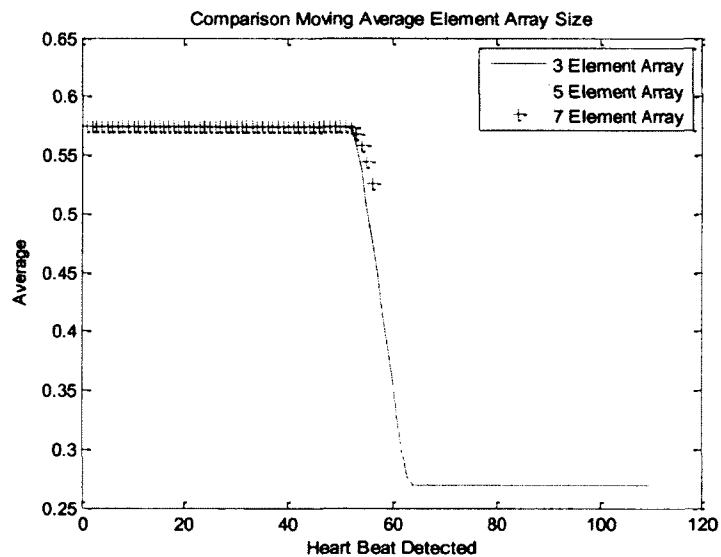


Figure 3.37: Moving Average Response to Pulse Patency Loss

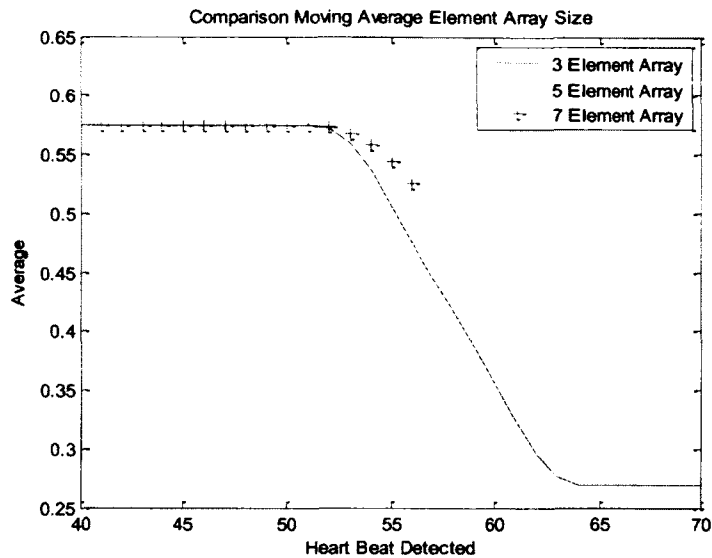


Figure 3.38: Enhanced view of Pulse Patency Loss

A similar experiment was done with pulse patency decay modeled as an exponential. This method of modeling is less expected than linear decay due to the fact that systolic blood pressure decays linear with the onset of G forces. However, since the onset of G forces on the human body may not be linear, it is useful to still see the effects of array length in regards to exponential decay.

The time constant for the exponential was defined as a quarter of difference between the start and stop times. The first test was conducted with a start time of 20 seconds and a stop of 35 seconds (Figure 3.39). Figure 3.40 shows the moving average response with different size array elements.

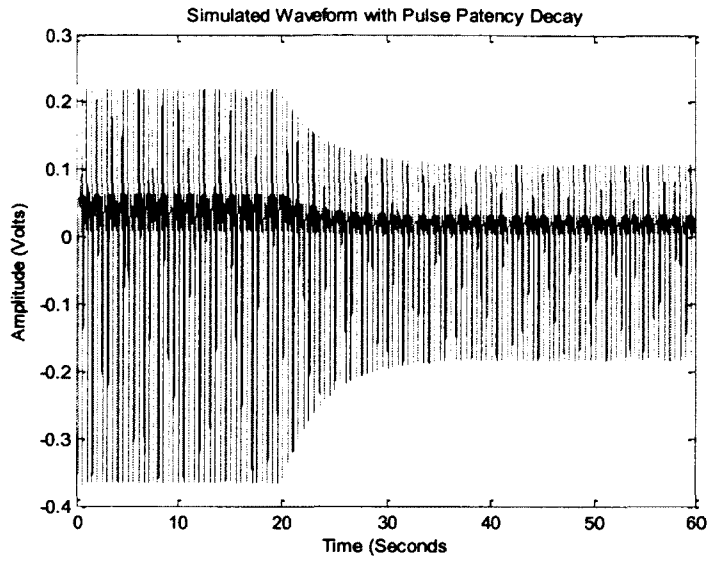


Figure 3.39: Simulation of Exponential Pulse Patency Delay

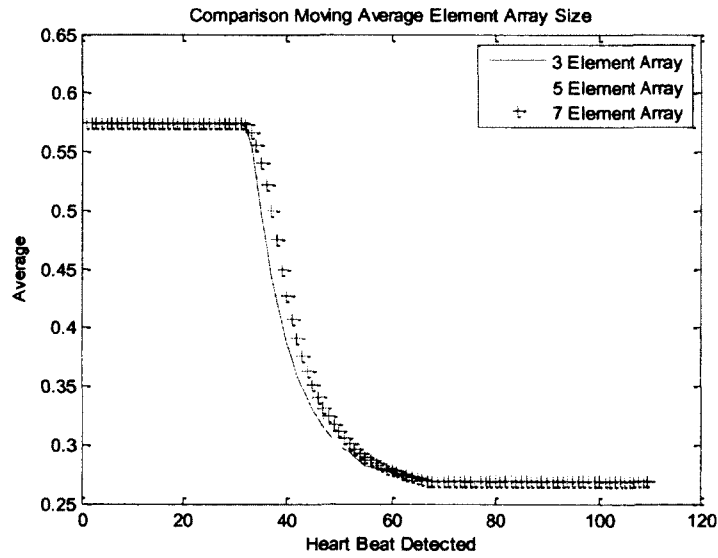


Figure 3.40: Moving Average Array Response

Similarly, the same thing occurs as what is seen in Figure 3.38 where the decay is too sudden for the algorithm to detect. There are two main methods of fixing

it. The first of those reasons being the variance variable. A larger variance variable will allow for detection of larger changes, but will consequently make the system more susceptible to noise. The second method would be to re-initiate the moving average algorithm. By reinitiating the algorithm, the pulse patency will always be detectable. However, the downfall of re-initiating the moving average array would be that it takes time to perform this task. For example, a 5 element moving average array would take a maximum of 5 seconds to initiate (assuming the slowest heart rate expected of 60BPM). During the 5 seconds, the pulse patency could stop in the pilot at forehead level and the pilot could already be in a state of GLOC.

Physical Model: Controlled Noise Analysis

When trying to utilize the accelerometer for pulse patency detection, noise is a major factor, particularly due to motion artifact. Motion artifact can't be eliminated from the accelerometer output, but its effects can be minimized. Motion artifact isn't the only cause of unwanted signals present in the pulse patency waveform. Radiating frequencies from close by electronics easily couple into wiring and affect circuitry.

The elimination of noise and adaptation to pulse patency decay are not only due to slope differences between data points, but are also due to the variance variable. The variance variable is the variable that determines if the following pulse meets the peak-to-peak voltage criteria compared to that of the moving average array average. For testing, the variance variable was set to 85%. Assuming a 1 volt

peak-to-peak average, this allows the follow on peak to be as large as 1.15 volts and as low as .85 volts peak-to-peak.

Having this margin of allowable signals allows for the reduction of intrinsic noises. Since the accelerometer is highly sensitive to motion, this reduces the number of false positives. Figure 3.41 shows the output waveform of the human and the detection of the pulse patency signal.

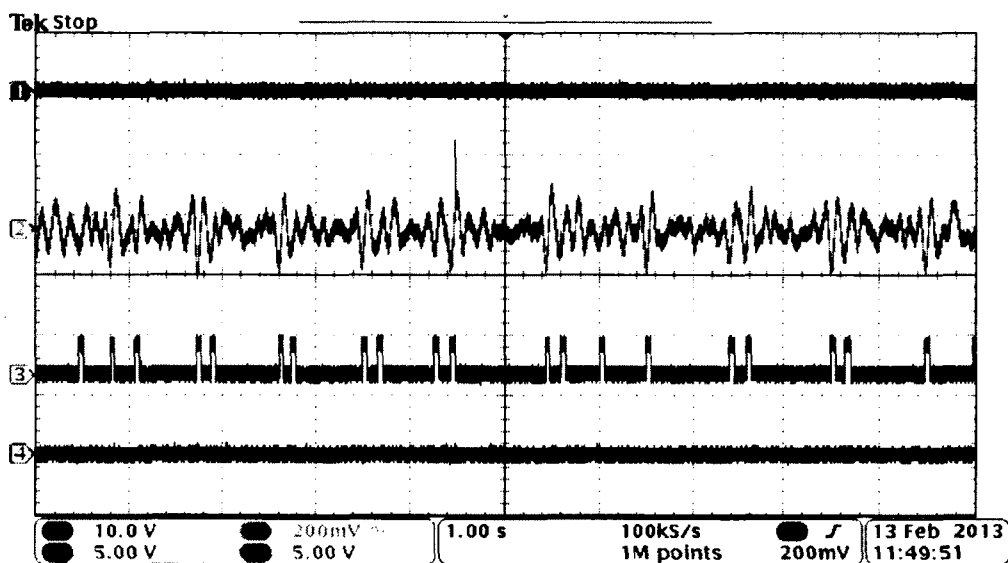


Figure 3.41: Human Model Pulse Patency and Detection of Peaks (top trace: unused; second trace: pulse patency sensor output; third trace LED voltage; fourth trace: alarm LED)

In Figure 3.41, it is seen that there are only two false detections of pulse patency signals (pulse 1 and pulse 14 if counted from left-to-right). This is due to the application of the variance variable. Through all the noise (ringing) and the different

amplitudes of the pulse patency waveform, the variance variable provided for accurate detection of the signals.

Using the Physical Model, noise was added to the system. As seen by the Physical Model diagram in Chapter 2, the second air compressor can be used to simulate noise. A 3Hz pulse was sent through the system which created adequate noise within the system (Figure 3.42). The top signal was the pulse waveform to simulate a 60BPM heart rate. The middle signal was the response by the accelerometer and the lower signal was the square wave used to generate noise through the latex tubing.

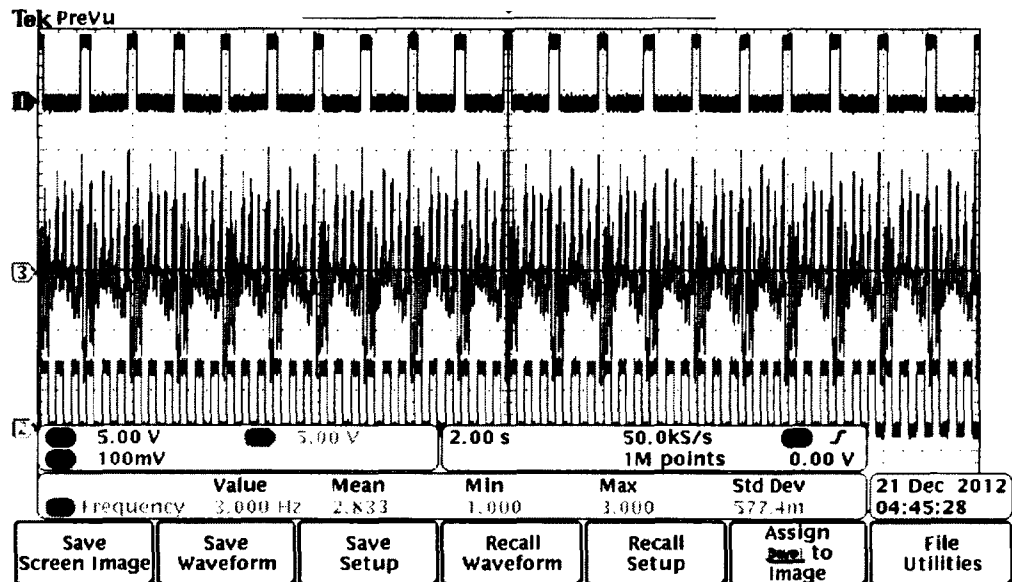


Figure 3.42: Pulse Patency Signal with Added Noise (top trace: function generator; second trace: pulse patency sensor output; third trace: added noise to system)

When run through the algorithm, the total pulses that were detected for the 1Hz pulse for 20 seconds (expected would be 20) was 19. To show the importance of the variance variable, if the variance was dropped to 60%, the detected waveforms increased to 26 (an increase of 37%).

Pulse Oximeter

Pulse oximetry testing was done using the OEMIII Evaluation Board along with the evaluation software. Using the evaluation board with USB connection, the results could be done using a computer. The output of the OEMIII module with 8000R reflectance pulse oximeter is I2C formatted data. Although the Arduino is fully capable of reading serial data, the plethysmograph (PPG) data would not be available for viewing. Since the PPG is of interest, the evaluation software proved to be sufficient in the needed testing.

The testing that was done consisted of location analysis at three prominent areas of the head. The first is the recommended location for reflectance pulse oximetry, the middle of the forehead. The other two regions were along the STA on the temple and the post-auricular region. After the readings were done, tests were done involving the effect of motion artifact on the device along with response to oxygen deprivation. Oxygen deprivation testing was done with the subject holding his/her breathe for 1 minute. The resulting changes in PPG and oxygen level are shown.

Pulse Oximeter Locations

The three locations that were documented were: the forehead, the STA, and the post-auricular region. Since PPG is defined as a measurement of the change of volume, it is expected that plot be larger for the STA (Figure 3.44). The forehead (Figure 3.43) and the post-auricular (Figure 3.45) plots are smaller due to the measurement of volume changes through capillaries. Therefore, it would be expected to be a smaller volume change.

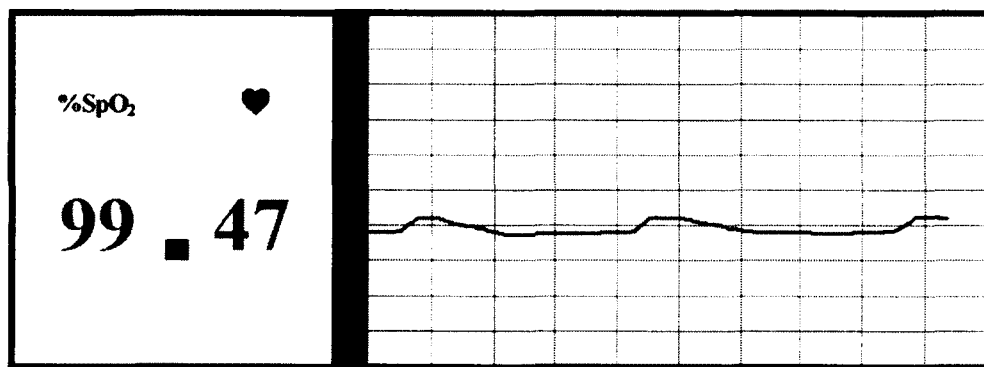


Figure 3.43: Forehead Positioned Blood Oxygenation Response

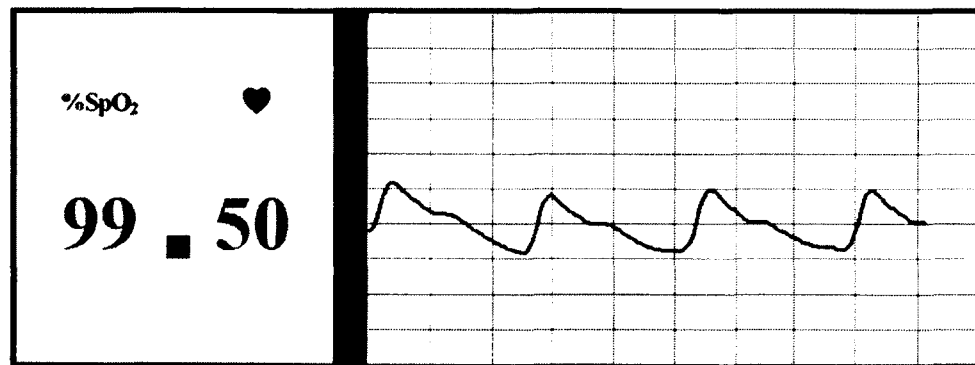


Figure 3.44: STA Positioned Blood Oxygenation Response

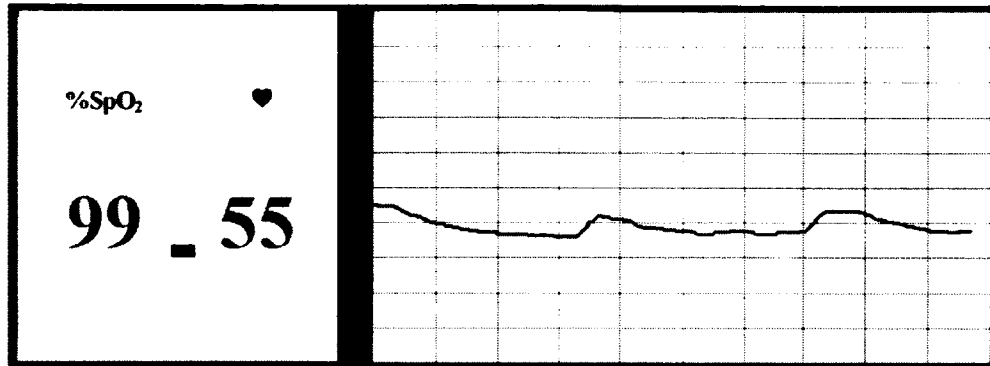


Figure 3.45: Post-auricular Positioned Blood Oxygenation Response

Since it is expected for there to be head motion in a cockpit, it is crucial to visually understand the effect of motion artifact on the PPG. Figure 3.46 through Figure 3.48 shows the effect of side-to-side head movement with PPG response.

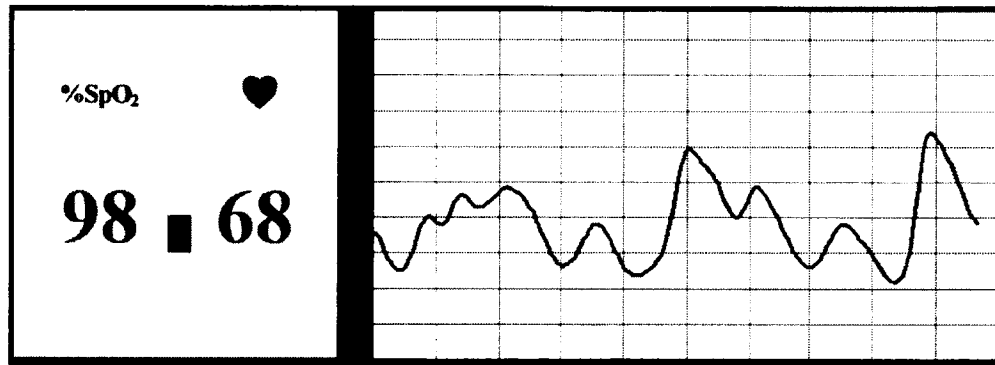


Figure 3.46: Forehead PPG Response to Head Movement

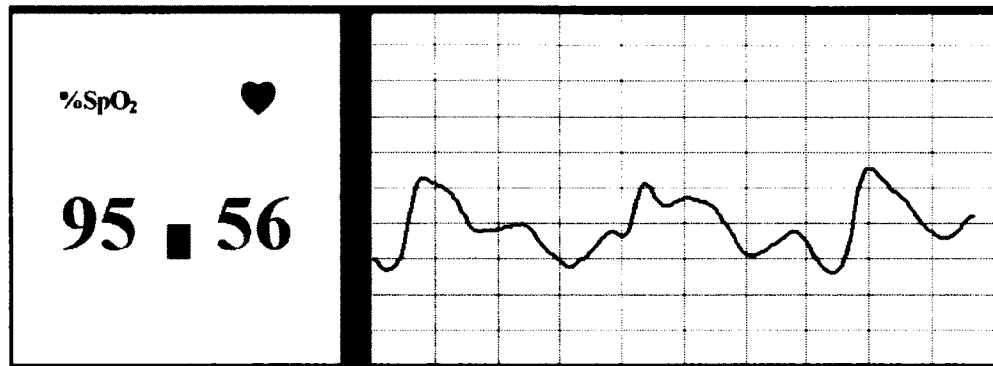


Figure 3.47: STA PPG Response to Head Movement

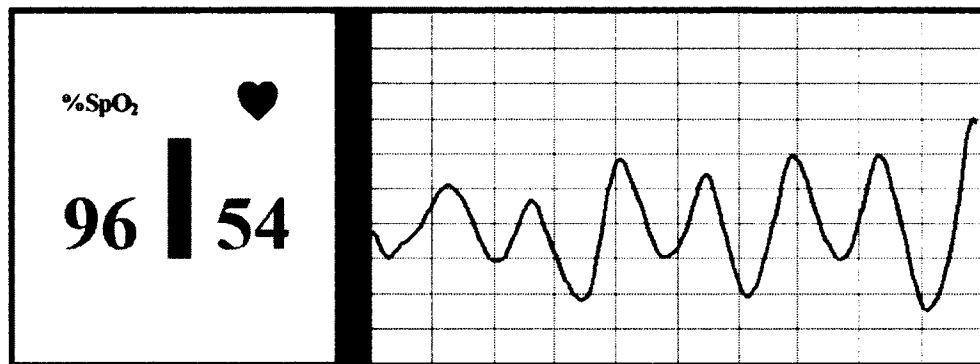


Figure 3.48: Post-Auricular PPG Response to Head Movement

As seen in the above figures, the forehead and STA PPG waveforms are still distinguishable among the noise, where the post-auricular PPG response is not. Figure 3.49 through Figure 3.51 show the effect of PPG waveform and oxygenation with 1 minute of oxygen deprivation. This test was conducted by asking the test subject to withhold his/her breathe for a minute.

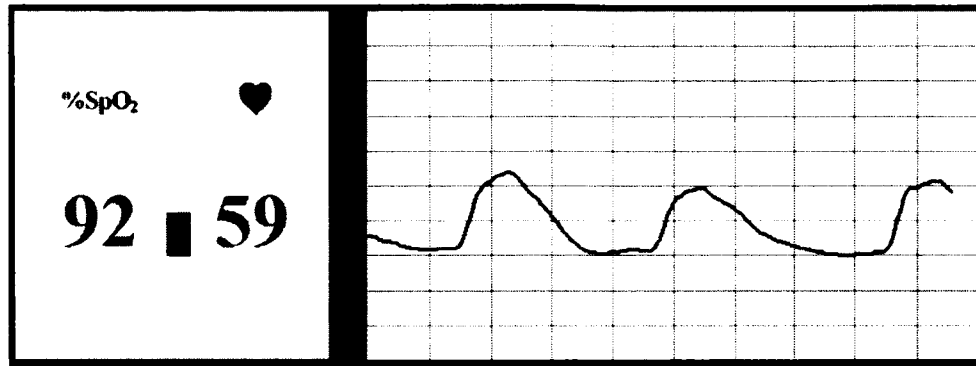


Figure 3.49: Forehead Response to Oxygen Deprivation

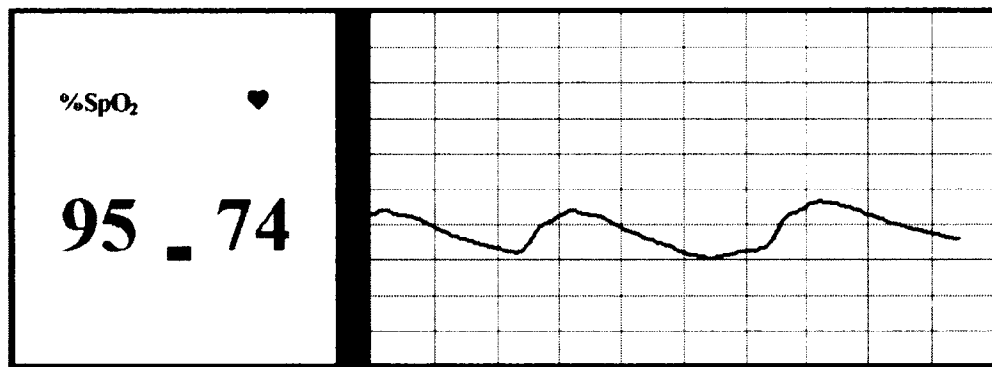


Figure 3.50: STA Response to Oxygen Deprivation

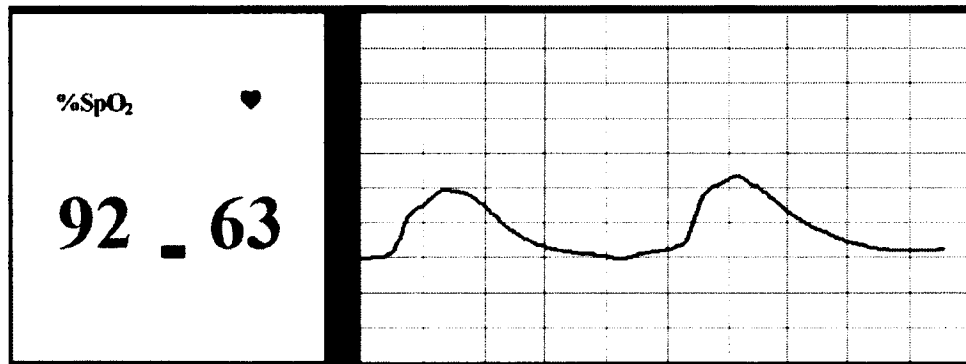


Figure 3.51: Post-Auricular Response to Oxygen Deprivation

This test is important for several reasons. Head-level oxygen fluctuation in the cockpit is expected due to the production of oxygen through engine bleed air coming through the oxygen mask. Seen in Figure 3.49, the blood oxygen level drops nearly 8%, which, depending on the threshold decided upon to declare GLOC conditions, may be close to that GLOC threshold. However, using PPG data is imperative to determining GLOC conditions. Mechanically measuring PPG data is what the pulse patency sensor measurement is. Together, these two can detect GLOC. Just one standalone system or sensor would not provide accurate results.

Pulse Patency and PPG Correlation

It is inferred that pulse patency and PPG data are in phase with one another. The PPG data, as seen in the previous section, was collected using an optical sensor. The optical sensor that was used was the blood oximeter. Since the PPG is the measurement of volume of change in an organ, or part of the body, the PPG taken on the STA would evidently be the change in blood volume present in the artery.

As discussed in earlier chapters, pulse patency is the measurement of force produced by the expansion and contraction of the elastic arterial wall. The arterial wall expands as the force of blood expands the arterial wall and, conversely, the arterial wall contracts as blood continues on its path through the artery.

It can be concluded that pulse patency is the mechanical measurement of PPG. Sensing pulse patency through the use of accelerometers, piezos, or resonating

antennae's is a mechanical sensor technique used to measure the change in volume, or PPG, in an artery. Therefore, it can be concluded that PPG data and pulse patency waveform should be in phase. The only thing that would delay the pulse patency waveform from the PPG would be the analog signal condition front end.

CHAPTER 4:

Discussion and Future Work

Overview

Up until this point, this thesis has been an exploration in a feasibility study inspired by the Loss of Consciousness Monitoring System (LOCOMS) system. The LOCOMS system was developed based on theoretical ways of detecting GLOC in pilots. Those ways included eye blink rate, head slumping, strength or absence of grip on stick, head level arterial pulsations, eye level blood pressure, arterial oxygen saturation percentage, spectral shift in electroencephalogram (EEG), presence and quality of anti-G straining maneuver (AGSM) and pilot response to voice synthesized interrogations. Due to numerous reasons (refer to Chapter 1), the choices that remained were head level arterial pulsations, eye level blood pressure, and pilot response to voice synthesized interrogations.

Many studies have been done on two of those areas of interest (head level arterial pulsations and eye level blood pressure). Methods of monitoring head level arterial pulsations have been done through plethysmography (PPG) in nose

mounted devices, resonating antennae's, and piezo-benders to monitor the expansion and contraction of arterial walls, or pulse patency. All of their results proved to have positive correlation with GLOC, but lacked resolution at high G-forces. The technologies were also highly susceptible to noise; a concern in the scope of this thesis.

Improvements in blood oxygenation monitoring techniques have improved vastly, but are also susceptible to the same problems of motion artifact and noise. In an attempt to gain some qualitative data for future recommendations, an exploration into blood oxygenation was conducted during this thesis.

The results attained from the modeling and testing of this thesis show that the development of a GLOC detection sensor is within reach. The results and review of the system are presented in this chapter. This chapter also concludes with recommendations for developing the optimum sensors and system for detecting GLOC under noisy conditions and high levels of G-strain.

Pulse Patency Sensor

It was hypothesized that an accelerometer would be able to detect pulse patency along the superficial temporal artery (STA) on the forehead of a pilot. The ability to detect pulse patency with an accelerometer would solve the problem of high-G resolution of as compared to the other technologies. By using an accelerometer, the sensor would simply be placed along the STA. There would be no

requirement of minimum force needed to bend a piezo [11] or resonating antennae [5] if the accelerometer were to be used.

The results from this study proved that pulse patency detection along the STA is possible with an accelerometer. However, as with every technology, there are limitations. First, the accelerometer is highly sensitive to motion. Simple head motion would cause a very low frequency (almost at DC) offset to the signal. Since the gain of the system was so high, this would cause the signal to go to the rails of the ADC. However, similarly to a high frequency noise riding on a low frequency signal, detection of the pulse patency signal is still possible assuming that the head motion or motion artifact was a very low frequency (less than 1Hz). The gain of the system needed to be high due to the minimum force produced by the expansion of the arterial wall and the resolution error from the 10bit ADC converter on the Arduino board.

In terms of noise reduction, the pulse patency analog front end proved to eliminate noise outside of the frequency range of interest and produce a clean signal for the ADC and microprocessor to analyze. The primary concern for noise was the influence of noise through rapid head movement and the high frequency noise produced by the vibration of the aircraft during flight. The only noise issue that was intrinsic to the system was the concept of ringing from the accelerometer. The transient response of the accelerometer with a step input caused ringing that could be seen in the results. In order to increase the speed of response, one would have to move the poles of the system (those created by the low pass filter and high pass

filter) further away from the origin causing a shift in the band pass frequencies.

Therefore, advanced control theory concepts such as adaptive control would have to be implemented that would filter out all signals excluding the fundamental frequencies.

The G-selection attribute to the accelerometer did not prove useful in filtering out some of the noise. The higher G-selection (6G's) only provided a lower voltage depiction of the lower G-selection (1.5G's) waveform. This would only require more gain needed to produce a recognizable signal, which looked similar to the lower G-selection waveform.

Blood Oxygenation

The blood oxygenation study gave insight into the detection of blood oxygenation at various locations along the head of the individual and the responses. Studies show that there is approximately a 10%-15% drop in spacial oxygenation (SpO₂) when an episode of GLOC occurs. Therefore, it is not necessarily important to measure every beat or every PPG signal with 100% accuracy. The results are all relative and are based off of a proportional drop.

The results (Chapter 3) show significant differences between the forehead, post-auricular, and STA mounted reflectance pulse oximeter locations. The post-auricular region is highly susceptible to motion artifact in terms of simple head movement and doesn't provide any better insight into blood oxygenation or blood

PPG than the forehead location or STA location. Therefore, it is inadvisable to place the pulse oximeter at this location.

Since the majority of blood oxygenation sensors are designed for forehead use, they tend to work better at this location. Placing the reflectance pulse oximeter on the STA provides insight into blood flow (or PPG) as it reaches eye-level. As the STA approaches the forehead, the artery becomes smaller and eventually breaks apart into capillaries that supply blood to the forehead. Since the reflectance pulse oximeter works by intercepting the signals off of the reflection of red and IR light from the forehead, it is imperative that the medium in-between remain motionless as the sample is taken. With a large artery in the way, it is understandable why the Air Force got noisy results when it was placed on the STA and better results along the forehead [9].

However, placement of the reflectance pulse oximeter on the STA along the temple, where the heart beat can barely be felt by the touch of a finger is the best location. The results of the investigation on sensor placement showed that this is the area that will provide the best PPG data along with accurate SpO₂ values. At this location, certainly there are causes of motion artifact. Similar to the pulse patency sensor, a clench of the jaw will cause motion artifact. But with the eight beat average ability of the Nonin OEMIII 8000R, that value will just be perceived as an outlier and won't be used for processing. Conversely, at this location the pulse oximeter is not susceptible to forehead muscle movement (a problem for forehead placement) and side-to-side head movement (a problem with post-auricular placement).

Microprocessor Algorithm

Development of an accurate algorithm was a large portion of the research conducted during the thesis. The intent of the algorithm was to be able to *accurately* detect peaks, eliminate false peaks (such as noise), detect loss of patency, resample, intelligently process the data and determine of a state of GLOC. The main concepts behind the algorithm include variance variables, moving average algorithm and array size, peak detect methods, and time restraints.

The purpose of the variance variable was to evaluate all future pulse waveforms based off of current pulse patency data. As seen throughout the results the variance variable, along with the moving average algorithm, were able to detect up to a 100% accuracy during simulation and an average of 93% accurate on the human model. The limitation of this, however, is the allowable G-profile of the pilot. A high onset of G, for example, would go outside the range of the variance variable and subsequent peaks would not be detected. For a situation like this, there are two options.

The first option is to increase the variance variable. An increase in the variance variable would allow for more liberal G-profile. Subsequently, this would cause for a larger chance that noise would affect the moving average array. The second option is to continuously resample and re-initiate the moving average algorithm. This would insure that the moving average algorithm always be up-to-date with current pulse patency values. Although this approach has its benefits, its

cons are significant. If the array is re-initiated during a state of GLOC, the array would be filled with noise amplitude values and would not realize that it is not detecting pulse patency values causing a catastrophic fault in the system. This, of course, can be fixed with setting a minimum allowable threshold value. Assuming that the noise amplitude is lower than the deemed pulse patency threshold cut off value, this would be a possible solution. This could only be determined with actual experimentation values of pilots pulse patency values during flight or in a centrifuge chamber.

Concerning the moving average array, the array length would also have to be adjusted according to expected G-profile and through centrifuge studies. The moving average array size experiments and reaction to pulse patency decay are shown in Chapter 3. These results are crucial. It is recommended that five values be used for the moving average array. Five elements in the moving average array provide proper response to simulated G-profiles along with maintaining accurate averaging results given the presence of noise.

The peaks detect methods discussed in Chapter 2 worked to a high degree of accuracy. By simply going point to point and finding the largest point is an ineffective due to its susceptibility to noise. With the average of data points, high frequency noise gets eliminated along with ADC error. Ten-bit ADC's are fairly inexpensive compared to higher 24bit ADC and using the derivative peak detect method, a physical cost can be spared.

The time restraints category is really a two part discussion. The first part requires filtering of the signal. The time filtering of the signal are setting dedicated time windows for allowable peaks and filtering out the higher frequency peaks. The time window also allowed for accurate detection of the correct peaks and not the false peaks produced by noise.

The second time restraint refers to the 5-7 seconds that the pilot has of reserve oxygen before GLOC starts. This doesn't seem like a significant amount of time, but in reality, it is. With capable systems, this allows for secondary systems to also analyze data, and alarms to be set. The alarms were set at four seconds allowing for the data to continuously be resampled. During this time, blood oxygenation can also be checked to determine if a sudden decrease in PPG or blood oxygenation is depicted by the reflectance pulse oximeter. Once again, however, these time restraints are highly theoretical and need to be calibrated according to centrifuge results.

As an improvement, the Arduino R3 UNO is not suitable for such data analysis. The algorithm worked perfectly in the MATLAB environment, and worked well for the Arduino during the Physical Model portion of this study. However, the microprocessor is simply incapable of handling all the data. It was calculated that a sampling frequency of 1kHz is needed to effectively process the data. During testing, the Arduino was barely getting a sampling rate of 1 kHz, without any I/O commands other than reading the analog signal. The *digitalWrite* command used to output a signal to the LED is a process that takes numerous milli-seconds to do, causing a

faulty sampling rate. It is recommended that a different microprocessor be used to process the data that is capable of higher sampling rates and executing processes in the micro-second timeframe.

As far as the sampling rate goes, an adaptive sampling rate algorithm should be used designed. This algorithm should vary the sampling rate based off of the heart rate of the individual. The oversampling of the signal will cause the algorithm to detect peaks along rising edges of waveforms due simply to the low signal frequency. This is counterproductive to what the algorithm should be doing, which is accurately detecting each actual peak.

Having an adaptive smart technology is ideal for such a scenario. Such a technology should be calibrated according to each pilot. By "signaturizing" each helmet, the helmet can be calibrated accurately. Obviously, such a helmet would require extra efforts, however, such efforts wouldn't equate to the cost of one fighter jet or human life. By calibrating each helmet to its owner, better insight will be given into the pilots G-profile under controlled conditions (i.e. centrifuge) and during flight.

Future Work

The limitations and benefits of this technology were discussed for each individual portion of such a sensor. As for the system as a whole, there are certain future work considerations that need to be addressed to further the continuation of

such a study. First, it is important to recognize that such a sensor shows the *ability* to act as a multi-sensor GLOC detection system. It has been shown, over and over, that blood oxygenation and pulse patency drop off as GLOC takes over. Therefore, the integration of these two technologies is the future of GLOC detection.

The primary recommendation for future work, other than the improvements mentioned in previous sections of this chapter, is to incorporate such a sensor into a package and finalize the time restraints through testing. If the human response to “slamming” on breaks to prevent an accident is in the milli-second range, then the response to a voice interrogation won’t take long. Initiating voice interrogations too early would cause distrust and annoyance in the system. It is almost better to do it at the last second, or even too late, than to do it early. At least, if it’s a second to late (i.e. at 7 seconds after pulse patency seizes), GLOC has been confirmed and auto-pilot or an auto-recovery system can be taken over.

The order of sensor priority should be accurately established. According to the testing, the pulse patency sensor should have priority over all sensors. It takes time for blood oxygenation to drop off given a GLOC state. It took 60 seconds for the subjects’ blood oxygenation to drop by 8% in the blood oxygenation experimentation. Therefore, pulse patency is the quickest response. The pulse patency sensor could be consistently verified by the PPG data taken from the reflectance pulse oximeter.

After “x” amount of seconds (a recommended 4 seconds), the algorithm should resample the blood oxygenation while it is continuously looking for a pulse

patency signal. If a loss of blood oxygenation is detected, voice interrogation may begin. If there is no loss of blood oxygenation, the moving average array should be re-initiated and the sampling for pulse patency waveforms should continue. This would provide for the most accurate GLOC detection.

Finally, it is recommended that consideration be given in an accelerometer array to detect pulse patency. Common mode signals can be used to remove the effect head motion or other forms of motion artifact. An array of four accelerometers can be constructed and placed on the STA to detect pulse patency.

Overall, it is feasible that such a technology be designed to detect pulse patency and blood oxygenation. Further research and development should be done in noise reduction in the pulse patency sensor. Calibration of the algorithm needs to be done with centrifuge results. This will provide for the most accurate determination of array length and variance variable values.

LIST OF REFERENCES

- [1] U. Balldin, "Acceleration Effects on Fighter Pilots," *Medical Aspects of Harsh Environments*, pp. 1014-1027, 2002.
- [2] M. D. Parkinson, "G-Awareness for Aircrew," Department of the Air Force, Washington, D.C., 2010.
- [3] J. Cammarota, "Integrated Systems for Detecting and Managing Acceleration-Induced Loss of Consciousness," Naval Air Development Center, Warminster, 1991.
- [4] L. Tripp, J. Warm, G. Matthews and P. Chiu, "+Gz Acceleration Loss of Consciousness: Time Course of Performance Deficits with Repeated Experience," *Human Factors*, pp. 109-120, 2006.
- [5] W. Albery and R. Van Patten, "Development of Gravity Induced Loss of Consciousness (GLOC) Monitoring System," *IEEE*, pp. 831-837, 1990.
- [6] N. Lewis, "The EEG as an Indicator of G-Induced Loss of Consciousness (GLOC)," in *59th Annual Scientific Meeting of the Aerospace Medical Association*, New Orleans, 1988.
- [7] P. Werchan, J. Schadt, J. Fanton and M. Laughlin, "Total and Regional Cerebral Blood Flow During Recovery from G-LOC," *Aviation, Space, & Environmental Medicine*, pp. 751-758, 1996.
- [8] L. Tripp and W. Albery, "Development of an Oxygen Mask Integrated Arterial Oxygen Saturation (SaO₂) Monitoring System for Pilot Protection in Advanced Fighter Aircraft," Harry G. Armstrong Aerospace Medical Research Laboratory, Wright-Patterson AFB, 1987.
- [9] R. Melker, M. Banner, B. Fuehrlein, G. Worley and N. Euliano, "Methods and Devices for Countering Gravity Induced Loss of Consciousness and Novel Pulse Oximeter Probes". U.S. Patent US 2008/0058621 A1, 6 March 2008.
- [10] D. Mawhinney and T. Kresky, "Superficial Temporal Artery Monitoring," David Sarnoff Research Center, Princeton, 1986.
- [11] J. LaCourse and K. Sivaprasad, "A System to Measure a Pilot's Temporal Pulse Pressuring During Acceleration," *Aviation, Space, and Environmental Medicine*, pp. 356-362, 1991.
- [12] B. Shender, J. Cammarota, H. Ryoo, E. Forster and H. L., "Almost Loss of Consciousness: a Factor in Spatial Disorientation?," in *Spatial Disorientation in Military Vehicles: Causes*,

Consequences, and Cures, La Coruna, Spain, 2002.

- [13] R. Simmons, J. Chandler and D. Horning, "A Forehead-Mounted Measure of O₂ Saturation: A Potential for in Cockpit Hypoxia Early Detection and Warning," Naval Aerospace Medical Research Laboratory, Pensacola, 2010.
- [14] PBS.org, "Orville and Wilbur Wright," [Online]. Available: <http://www.pbs.org/kcet/chasingthesun/innovators/owwright.html>.
- [15] PBS.org, "Wright Flyer," [Online]. Available: <http://www.pbs.org/kcet/chasingthesun/planes/wrightfly.html>.
- [16] L. Warsitz and G. Brooks, *The First Jet Pilot*, UK: Pen and Sword, 2009.
- [17] Department of the Army, "Chapter 4: Gravitational Forces," in *Aeromedical Training for Flight Personnel*, Washington, D.C., Department of the Army, 2000.
- [18] R. Burton, "G-Induced Loss of Consciousness - definition, history, current status.," *Aviat Space Environ Med*, 1988.
- [19] L. D. Tripp, J. S. Warm, G. Matthews, P. Y. Chiu and B. Bracken, "On Tracking the Course of Cerebral Oxygen Saturation and Pilot Performance During Gravity-Induced Loss of Consciousness," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 51, no. 6, pp. 775-784, 2009.
- [20] A. D. Woodrow and J. T. Webb, "Handbook of Aerospace and Operational Physiology," USAF, San Antonio, 2010.
- [21] J. Cammarota and L. Hrebien, "Using Arterial Pulse Wave Delay to Assess +Gz Protection," in *Northeast Bioengineering Conference*, Warminster, 1987.
- [22] J. E. Hall and A. C. Guyton, *Guyton and Hall Textbook of Medical Physiology*, Philadelphia: Saunders, 2010.
- [23] J. E. Dickau, "Automatic Pilot Controls System (ACS) for GLOC and ILOC, with Aircraft Following Systems". Patent US 2003/0034902 A1, 20 Feb 2003.
- [24] SRICO, "www.srico.com," SRICO, 2005. [Online]. Available: www.srico.com. [Accessed 12 2012].
- [25] William Collins Sons & Co., *Collins English Dictionary - Complete & Unabridged 10th Edition*, HarperCollins Publishers, 2009.

- [26] S. Thiagarajah, D. Girnar and K. Park, "Blood Pressure Monitoring Using the Superficial Temporal Artery and a Doppler Ultrasonic Flow Detector," *Anesthesia and Analgesia*, vol. 58, no. 6, pp. 526-527, 1979.
- [27] G. Florence, R. Bonnier, L. Riondet, D. Plagnes, D. Lagarde, P. Van Beers, A. Serra, X. Etienne and D. Tran, "Cerebral Cortical Blood Flow During Loss of Consciousness Induced by Gravitational Stress," *Neuroscience Letters*, vol. 305, pp. 99-102, 2001.
- [28] T. Myers, W. D. Glanze and K. Anderson, *Mosby's Medical Dictionary*, Mosby, 2008.
- [29] J. G. Webster, *Medical Instrumentation Application and Design*, Marblehead: Wiley, 2009.
- [30] R. A. McKinley, L. D. Tripp, S. D. Bolia and M. R. Roark, "Computer Modeling of Acceleration Effects on Cerebral Oxygen Saturation," *Aviation, Space, and Environmental Medicine*, August 2005.
- [31] National Institute of Health, "Medline Plus: Pulse," 23 January 2012. [Online]. [Accessed 3 1 2013].
- [32] National Institute of Health, "NINDS Cerebral Hypoxia Information Page," 18 October 2010. [Online]. Available: <http://www.ninds.nih.gov/disorders/anoxia/anoxia.htm>.
- [33] W. Hennigan, "Fatal Problems Plague the U.S. Costliest Fighter Jet," *Los Angeles Times*, 19 December 2011.

APPENDIX A:

WAVEFORM GENERATOR CODE

```
close all;
clc;
clear all;

load 'Waveform_Values.mat';
%%Sampling Frequency 150Hz
%%For 60BPM (1hz) Period = .6267 - .02668 = .6

%Simulated Heartbeat (in BPM)
heart_beat = 180;

heart_freq = heart_beat/60;
heart_period = 1/heart_freq;

% Gz =
%Simulated G Pulled

%%Sample Length (in Seconds)
s_length = 60;

wavelength = max(new_time) - min(new_time);

%%Calculate the amount of iterations needed for the requested time
length
iterations = round(s_length / heart_period);

%%SIMULATED G MANUEVER%%
%%Initial Conditions

g_final = 2; %Final G condition

%%G scale is the ratio of expected patency between initial G condition
and
%%final G condition. (i.e. assume patency at 2G is 1/2 of that at 1G,
%%therefore g_scale_amp will equal .5.
g_scale_amp = .5;
```

```

%Input in Seconds
g_time_start = 20; %When G Manuever is conducted
g_time_stop = 30; %When G Manuever is finished

%Creating the waveform
for a = 1:iterations
    for i = 1:99
        sim_signal(i+(a-1)*99) = new_signal(i);
        if heart_beat >= 60
            sim_time(i+(a-1)*99) = (new_time(i)*heart_period) + (a-
1)*heart_period;
        else
            sim_time(i+(a-1)*99) = new_time(i) + (a-1)*heart_period;
        end
    end
end

%%%*****Upsample and Downsample to get fs*****
fs = 1000;

up_down = roundn((s_length*fs/(s_length*heart_freq*99)),-2);
[up,down] = rat(up_down); %Gets the rational function

%%%*****Upsampling (Interpolating)*****
sim_signal_up(1:(length(sim_signal)*up)) = 0;
sim_signal_up(1:up:length(sim_signal_up)) =
sim_signal(1:length(sim_signal));

%Create a time array associated with the interpolated signal
for y = 1:length(sim_signal_up)
    sim_time_up(y) = (y-1)*(s_length/length(sim_signal_up));
end

%Declare pass band frequency and stop band frequency
f_pass_up = (length(sim_signal)/s_length)/2 - 10; %Freq in Hz
rad_pass = 2*pi*f_pass_up/(length(sim_signal_up)/s_length); %Normalized
freq in rad

f_stop_up = (length(sim_signal)/s_length)/2; %Freq in Hz
rad_stop = 2*pi*f_stop_up/(length(sim_signal_up)/s_length); %Normalized
freq in rad

%Determines the interpolated sampling frequency
fs_high = length(sim_signal_up)/s_length;

%Design a filter based off of varying input criteria
d_up =
fdesign.lowpass('Fp,Fst,Ap,Ast',f_pass_up,f_stop_up,1,60,fs_high)
df_up = design(d_up,'butter');
% fvttool(df_up)

```

```

%%Filter the signal
sim_signal_up_filt = filter(df_up,sim_signal_up);

%%Add a gain to the signal
sim_signal_up_filt = sim_signal_up_filt * up;

****
%%Downsampling (Decimating)%%
****
f_pass_down = fs/2 - 5;          %%Sampling Frequency/2 - df (5)
f_stop_down = fs/2;             %%Sampling Frequency/2 = Nyquist rate

%%fs_low = 1000; %%Sets low frequency

%%Design a filter to decimate the data
d_down =
fdesign.lowpass('Fp,Fst,Ap,Ast',f_pass_down,f_stop_down,1,60,fs)
df_down = design(d_down,'butter');
% fvtool(df_down)

%Filter the signal
sim_signal_down_filt = filter(df_down, sim_signal_up_filt);

sim_down_length = length(sim_signal_down_filt); %%Finds the total
length of the signal

%%Zero pad the signal to make it divisible by decimation factor
for down_n = 1:down
    if rem(length(sim_signal_down_filt),down) ~= 0
        sim_signal_down_filt(sim_down_length+down_n) = 0;
        sim_time_up(sim_down_length + down_n) =
sim_time_up(sim_down_length) + down_n*(sim_time_up(2) -
sim_time_up(1));
    elseif rem(length(sim_signal_down_filt),down) == 0
        break
    end
end

sim_down_length = length(sim_signal_down_filt); %%Find length of signal
again

%%Decimates the signal based off of "down" variable
sim_signal_down(1:sim_down_length/down) =
sim_signal_down_filt(1:down:sim_down_length);
sim_time_down(1:sim_down_length/down) =
sim_time_up(1:down:sim_down_length);

%%Renames the variable
output = -1*sim_signal_down;
time = sim_time_down;

```

```

for z = 1:fs
    sig_length = length(output);
    if rem(sig_length, fs) == 0
        break
    else
        if rem(sig_length, fs) ~= 0
            output(sig_length+1) = 0;
            last_time = max(time);
            time(sig_length+1) = last_time + z/fs;
        end
    end
end
end

g_time_start = round((g_time_start/s_length)*length(time));
g_time_stop = round((g_time_stop/s_length)*length(time));

%%Create the Line Window Function
g_line_window(1:g_time_start) = 1;

for z = 1:(g_time_stop - g_time_start)
    g_line_window(g_time_start + z) = ...
        g_line_window(g_time_start+ z - 1) - (1 - g_scale_amp) / ...
        (g_time_stop - g_time_start);
end

g_line_window(g_time_stop:length(time)) = g_scale_amp;

%%Create the Exponential Window Function
%Exponential decay 0<g_decay<1

g_decay = 4/(g_time_stop - g_time_start);

g_exp_window(1:g_time_start) = 1;

for l = 1:(g_time_stop - g_time_start)
    g_exp_window(g_time_start + l) = (1-g_scale_amp)*exp(-
1*g_decay*l)...
        + g_scale_amp;
end

g_exp_window(g_time_stop:length(time)) = g_scale_amp;

%%Applying the window function

% output = output .* g_line_window;
% output = output .* g_exp_window;

```



```

% figure;
% plot(sim_time, sim_signal);
% title(sprintf('Simulated Patency %d BPM', heart_beat));
% xlabel('Time(seconds)');
% ylabel('Amplitude(Volts)');
% & xlim([0 1]);

figure;
plot(1:length(g_line_window), g_line_window)
title('Linear Window Function');
xlabel('Sample Number');
ylabel('Scaling Factor');
ylim([0 1.5]);

figure;
plot(1:length(g_exp_window), g_exp_window)
title('Exponential Window Function');
xlabel('Sample Number');
ylabel('Scaling Factor');
ylim([0 1.5]);

figure;
m = 0:length(output)-1;
n = length(output);
freq = (1/(time(2) - time(1))).*m/n;
mag = abs(fft(output)/length(output));
plot(freq, mag);
title('Frequency Spectrum of Simulated Waveform');
ylabel('Normalized Magnitude');
xlabel('Frequency (Hz)');
xlim([0 (1/(time(2)-time(1)))/2]);

figure;
plot(time, output);
xlabel('Time(S)');
ylabel('Amplitude(V)');
title(sprintf('Simulated Signal %d BPM', heart_beat));
ylim ([-.6 .6]);
xlim([0 s_length]);

% figure;
% stem(time, output);
% title('Discrete: 300 Beats Per Minute: Simulation');
% xlabel('Time(S)');
% ylabel('Amplitude(V)');
% title(sprintf('Simulated Signal %d BPM', heart_beat));
% ylim ([-.6 .6]);

```

APPENDIX B:

LINEAR AND EXPONENTIAL DECAY WINDOW CODE

```
s_length = 60;           %input time of simulation

g_time_start = 20;       %input time to start window
g_time_stop = 35;       %input time to stop decay

g_scale_amp = .5;       %scalar of window after "manuever"

g_time_start = round((g_time_start/s_length)*length(time));
g_time_stop = round((g_time_stop/s_length)*length(time));

**Create the Line Window Function
g_line_window(1:g_time_start) = 1;

for z = 1:(g_time_stop - g_time_start)
    g_line_window(g_time_start + z) = ...
        g_line_window(g_time_start+ z - 1) - (1 - g_scale_amp) / ...
        (g_time_stop - g_time_start);
end

g_line_window(g_time_stop:length(time)) = g_scale_amp;

g_line_window = g_line_window(1:length(time));

**Create the Exponential Window Function
**Exponential decay 0<g_decay<1

g_decay = 4/(g_time_stop - g_time_start);

g_exp_window(1:g_time_start) = 1;

for l = 1:(g_time_stop - g_time_start)
    g_exp_window(g_time_start + l) = (1-g_scale_amp)*exp(-
1*g_decay*l)...
    + g_scale_amp;
end

g_exp_window(g_time_stop:length(time)) = g_scale_amp;
g_exp_window = g_exp_window(1:length(time));
```

```
    %Applying the window function

s_output = output .* (g_line_window)';
s_output = output .* g_exp_window';

figure;
plot(time,output);
title('Simulated Waveform with Pulse Latency Decay');
xlabel('Time (Seconds)');
ylabel('Amplitude (Volts)');
xlim([0 s_length]);
```

APPENDIX C:

MATLAB ALGORITHM CODE: USER INPUT MOVING AVERAGE ARRAY

```
clc;
close all;

%% Pulse detect

pulse = 0;
comp_min = 0; comp_max = 0;
min_time = 0; max_time = 0;

x_s = 0; x_t = 0;

max_time_prev = 0;

low_threshold = .1;
high_threshold = 2.3;
t_window = .25;

max_time_prev = -.5

threshold_trigger = .01;    %%Trigger for the alarm
time_index = 1;

variance = .8;

%%Set up Moving Average
comp_diff = 0;    %%Moving Average variable
mov_avg_array = [low_threshold low_threshold low_threshold
low_threshold low_threshold];
elements = 5;
mov_index = 0;    %%Index to place in array during comparison

%%Set index numbers for array import
fs = round(1/(time(2) - time(1)));

f_x = 1;
f_n = 1;

i=1;    %%Mov_avg_data index
```

```

k=1;          %Peak_Detect_Array index
d=1;
rd=1;        %Calculate the first couple pulses amplitude
detect_max = threshold_trigger; % Set detect max as the lowest
threshold

maxd = 1;mind = 1; % Index for derivative arrays

sig_length = length(output);
last_time = time(sig_length);

mov_avg_data = 0;

%Zero Pad Array
z = 1;
for z = 1:fs
    if rem(length(output), fs) == 0
        break
    else
        if rem(length(output), fs) ~= 0
            output(sig_length+z) = 0;
            time(sig_length+z) = last_time + z*.0001;
        end
    end
end
end

z = 0;

sample_iteration = length(output)/fs;

for x = 1:(sample_iteration);

    %Import the Arrays
    if f_x <= (length(time) - fs)
        for f = f_x:(f_x+fs-1)
            x_s(f_n) = output(f);
            x_t(f_n) = time(f);
            f_n = f_n+1;
        end
    else
        break
    end

    f_x = f_x+fs; %Increase the import index by a full sample
size

    f_n = 1; %Reset index count

    for n = 1:(length(x_s)-1)
        %Detect the minimum peak by finding the derivative
        max_derivative = ( x_s(n+1) - x_s(n))/(x_t(n+1) - x_t(n));
        min_derivative = (x_s(n+1) - x_s(n))/(x_t(n+1) - x_t(n));

        current_time = x_t(n);

```

```

current_value = x_s(n);

if x_t(n) - max_time_prev > .1
    if max_derivative > -1
        if max_derivative < 2
            if x_s(n) > comp_max
                comp_max = x_s(n+1);
                max_time = x_t(n+1);

                comp_min = x_s(n+1);
                min_time = x_t(n+1);

                max_der(maxd) = max_derivative;
                maxd = maxd+1;
            end
        end
    end
end

end

end

end

%%Detect the maximum peak by finding the derivative
if min_derivative < 1
    if min_derivative > -2
        if x_s(n+1) < comp_min
            comp_min = x_s(n+1);
            min_time = x_t(n+1);

            min_der(mind) = min_derivative;
            mind = mind + 1;
        end
    end
end

end

end

%%Calculate difference in min and max amplitude/time
comp_diff = comp_max - comp_min;
comp_time = min_time - max_time;

%%Detect the difference of the first 5 iterations
%%Import into an array to determine the threshold
if x <= 5
    if comp_time > 0
        if comp_time < t_window
            if min_time - max_time_prev > .0
                if comp_diff > threshold_trigger
                    peak_detect_array(k) = comp_diff;
                    k = k+1;

                    switch_var = comp_max;
                    comp_max = comp_min;
                    comp_min = switch_var;
                end
            end
        end
    end
end

end

end

end
end

```

```

**Process further pulses based off of imported array
if x >= 1
    %Load up Moving Average Array
    if rd < 2
        %Zero pad the array so that it is divisible by the
amount
        %of elements in the moving average array
        for rem_n = 1:length(peak_detect_array)
            if rem(length(peak_detect_array),elements) ~=0
                peak_detect_array(length(peak_detect_array) +
rem_n) = 0;
            elseif rem(length(peak_detect_array),elements) ==
0
                break
            end
        end
    end
    %Determine the (amount of) elements peaks in the
array and
    %place it into the moving average array
    for dr =
1:length(peak_detect_array)/elements:length(peak_detect_array)
        for r = 1:(length(peak_detect_array)/elements)
            if peak_detect_array(r) >= detect_max
                detect_max = peak_detect_array(r);
            end
        end
        mov_avg_array(d) = detect_max;
        d = d+1;
        detect_max = 0;
    end
    rd = rd+1;
end

%Calculate the moving average
mov_avg = (mov_avg_array(1) + mov_avg_array(2) + ...
    mov_avg_array(3) + mov_avg_array(4) + ...
    mov_avg_array(5))/elements;

**Detect if signal is a pulse
if comp_time > 0
    if comp_time < t_window
        if min_time - max_time_prev > 0

            %Set thresholds based off of moving average
            low_threshold = mov_avg * variance;
            high_threshold = mov_avg * (1+(1-variance));

            if comp_diff > low_threshold
                if comp_diff < high_threshold

                    %Increase Pulse
                    pulse = pulse + 1;
                end
            end
        end
    end
end

```

```

pulse_time(time_index) = min_time;
time_index = time_index+1;

%Shift down the moving average numbers
%Put the new number in the mov. avg.
array
mov_avg_array(4-y);

for y = 0:3
    mov_avg_array(5-y) =

end
mov_avg_array(1) = comp_diff;

%Set last max time
max_time_prev = min_time;

%Rotate variables
switch_var = comp_max;
comp_max = comp_min;
comp_min = switch_var;

%Add this to the data array to plot
mov_avg_data(i) = mov_avg;
i = i+1;

end
end
end
end
end
max_time_prev = 0;
end
end
end

pulse

figure;
plot(1:length(mov_avg_data),mov_avg_data);
title('Moving Average Data');

%%Derivative Plots
figure;
plot(1:length(max_der), max_der);
title('Maximum Derivative');

figure;
plot(1:length(min_der), min_der);
title('Minimum Derivative');

```


APPENDIX D:

MATLAB ALGORITHM CODE: SELF-INITIATING MOVING AVERAGE ARRAY

```
clc;
close all;

%% Pulse detect

pulse = 0;
comp_min = 0; comp_max = 0;
min_time = 0; max_time = 0;

x_s = 0; x_t = 0;

max_time_prev = 0;

low_threshold = .1;
high_threshold = 1.3;
t_window = .25;

threshold_trigger = .01;    %%Trigger for the alarm

variance = .85;
variance_high = .5;

%%Set up Moving Average
comp_diff = 0;                %%Moving Average variable
* mov_avg_array = [low_threshold low_threshold low_threshold];
elements = 5;
mov_index = 0;                %%Index to place in array during comparison

%%Set index numbers for array import
fs = round(1/(time(2) - time(1)));

f_x = 1;
f_n = 1;

i=1;                          %%Mov_avg_data index
k=1;                          %%Peak_Detect_Array index
d=1;
rd=1;                          %%Calculate the first couple pulses amplitude
```

```

detect_max = threshold_trigger; %Set detect_max as the lowest
threshold

maxd = 1;mind = 1; %Index for derivative arrays

sig_length = length(output);
last_time = time(sig_length);

mov_avg_data = 0;

peak_detect_array(1:5) = 0;

%%Zero Pad Array
z = 1;
for z = 1:fs
    if rem(length(output), fs) == 0
        break
    else
        if rem(length(output), fs) ~= 0
            output(sig_length+z) = 0;
            time(sig_length+z) = last_time + z*.0001;
        end
    end
end
end

z = 0;

sample_iteration = length(output)/fs;

for x = 1:(sample_iteration);

    %Import the Arrays
    if f_x <= (length(time) - fs)
        for f = f_x:(f_x+fs-1)
            x_s(f_n) = output(f);
            x_t(f_n) = time(f);
            f_n = f_n+1;
        end
    else
        break
    end

    f_x = f_x+fs; %Increase the import index by a full sample
size

    f_n = 1; %Reset index count

    for n = 1:(length(x_s)-1)
        %Detect the minimum peak by finding the derivative
        max_derivative = ( x_s(n+1) - x_s(n))/(x_t(n+1) - x_t(n));
        min_derivative = (x_s(n+1) - x_s(n))/(x_t(n+1) - x_t(n));

        if max_derivative > 0
            if max_derivative < 4

```

```

        if x_s(n) > comp_max
            comp_max = x_s(n+1);
            max_time = x_t(n+1);

            comp_min = x_s(n+1);
            min_time = x_t(n+1);

            max_der(maxd) = max_derivative;
            maxd = maxd+1;
        end
    end
end

%%Detect the maximum peak by finding the derivative
if min_derivative < 0
    if min_derivative > -4
        if x_s(n+1) < comp_min
            comp_min = x_s(n+1);
            min_time = x_t(n+1);

            min_der(mind) = min_derivative;
            mind = mind + 1;
        end
    end
end

%%Calculate difference in min and max amplitude/time
comp_diff = comp_max - comp_min;
comp_time = min_time - max_time;

%%Detect the difference of the first 5 iterations
%%Import into an array to determine the threshold
if x <= 5
    if comp_time > 0
        if comp_time < t_window
            if min_time - max_time_prev > .0
                if comp_diff > threshold_trigger
                    for bs = 1:5
                        if comp_diff > peak_detect_array(bs)
                            peak_detect_array(bs) = comp_diff;
                            switch_var = comp_max;
                            comp_max = comp_min;
                            comp_min = switch_var;
                            break;
                        end
                    end
                end
            end
        end
    end
end

end
end
end

%%Process further pulses based off of imported array
if x > 5
    %%Load up Moving Average Array

```

```

if rd < 2
    %Zero pad the array so that it is divisible by the
amount
    %of elements in the moving average array
    for rem_n = 1:length(peak_detect_array)
        if rem(length(peak_detect_array),elements) ~=0
            peak_detect_array(length(peak_detect_array) +
rem_n) = 0;
        elseif rem(length(peak_detect_array),elements) == 0
            break
        end
    end
end

%Determine the (amount of) elements peaks in the array
and
%place it into the moving average array
for dr =
1:length(peak_detect_array)/elements:length(peak_detect_array)
    for r = 1:(length(peak_detect_array)/elements)
        if peak_detect_array(r) >= detect_max
            detect_max = peak_detect_array(r);
        end
    end
    mov_avg_array(d) = detect_max;
    d = d+1;
    detect_max = 0;
end

rd = rd+1;
end

%%Calculate the moving average
mov_avg = (mov_avg_array(1) + mov_avg_array(2) + ...
    mov_avg_array(3) + mov_avg_array(4) + ...
    mov_avg_array(5))/elements;

**Detect if signal is a pulse
if comp_time > 0
    if comp_time < t_window
        if min_time - max_time_prev > 0

            %%Set thresholds based off of moving average
            low_threshold = mov_avg * variance;
            high_threshold = mov_avg * (1+(1-variance));

            if comp_diff > low_threshold
                if comp_diff < high_threshold

                    %Increase Pulse
                    pulse = pulse + 1;

                    %Shift down the moving average numbers
                    %Put the new number in the mov. avg.
array
                    for y = 0:3

```

```

mov_avg_array(4-y);

mov_avg_array(5-y) =
end
mov_avg_array(1) = comp_diff;

%Set last max time
max_time_prev = max_time;

%Rotate variables
switch_var = comp_max;
comp_max = comp_min;
comp_min = switch_var;

%Add this to the data array to plot
mov_avg_data(i) = mov_avg;
i = i+1;

end
end
end
end
end
max_time_prev = 0;
end
end
end
end

pulse

figure;
plot(1:length(mov_avg_data),mov_avg_data);
title('Moving Average Data');
xlabel('Heart Beat Detected');
ylabel('Average');

%%Derivative Plots
% figure;
% plot(1:length(max_der), max_der);
% title('Maximum Derivative');
%
% figure;
% plot(1:length(min_der), min_der);
% title('Minimum Derivative');

```

APPENDIX E:

ARDUINO ALGORITHM CODE

```
/*Peak Detect*/

const int input = A0;    //Input pin for reading
const int ledPin = 13;  //Output pin for onboard LED pin
const int emergency = 12;

double time1 = 0;    //Previous data sample time (x-value)
double time2 = 0;    //Current data sample time (x-value)
double diffTime = 0; //Variable to compute difference in time

double val1 = 0;    //First data point value (y-value)
double val2 = 0;    //Second data point value (y-value)
double val3 = 0;    //Current data point value (y-value)
double valAverage = 0; //Average of second and third data point
double diffVal = 0; //Variable to compute the difference in values

double slope = 0;

double peakMin = 0; //Stores current minimum peak
double peakMax = 0; //Stores current maximum peak
```

```

double diffPeak = 0; //Calculates the different in peak values
int peakSwitch = 0;

double lastMax = 0; //Stores last max TIME value
double lastMin = 0; //Stores last min TIME value
double timeDiff = 0; //time difference in peak times;

double lastBeat = 0; //Stores last heart beat
double lightTime = 0;

int avgInitiate = 0; //index
int initialPeaks = 100; //number of peaks to detect initially and calculate the
moving average algorithm
const int index = 5; //numbers of values in moving average array
double movingAverage[index]; //initiates the array
double average = 0; //what the average is of the array
double minDiff = 0; //minimum accepted peak for moving average
double variance = .98; //variance from average
double eTime = 0; //emergency light comes on

void setup() {
  pinMode(ledPin, OUTPUT); //sets the LED pin as the OUTPUT pin
  pinMode(emergency, OUTPUT);

  Serial.begin(9600);

  for (int start = 0; start < index; start++){

```

```

    movingAverage[start] = 0;    //initiates the entire moving average values to 0
  }
}

void loop(){
  time1 = time2;

  val1 = val2;
  val2 = val3;
  val3 = analogRead(input);
  time2 = millis();
  diffTime = time2 - time1;
  val3 = val3*5/1023;

  //Serial.println(val3);

  valAverage = (val2+val3)/2;

  if ((time2 - lightTime) >= 50){
    digitalWrite(ledPin, LOW);
  }
  if ((lastBeat - eTime) > 2000){
    digitalWrite(emergency, LOW);
  }
  //Resets the max peak value if there hasnt been a peak in longer than a second
  if ((time2 - lastMax)>=1000){

```



```

    peakMax = 0;
}

//Simulates autopilot if there is no pulse for 5 seconds
if ((time2 - lastBeat)> 1000){
    digitalWrite(emergency, HIGH);
    eTime = time2;
}

//Serial.println(diffTime);

slope = (valAverage - val1)/diffTime;

if ((slope >= 0)&&(slope < 1)){ //checks for maximum peak
    if (val2 > peakMax){ //compares the value of the current peak to last peak
        peakMax = val2;
        peakMin = peakMax;

        //Serial.println("MAX");

        lastMax = time2;
        lastMin = time2;

        lastBeat = time2;
    }
}

```

```

if ((slope <= 0)&&(slope > -1)){ //check for minimum peak
  if (val2 < peakMin){
    peakMin = val2;
    lastMin = time2;

    //Serial.println("MIN");

    diffPeak = peakMax - peakMin;
    timeDiff = lastMin - lastMax;
  }
}

//Serial.println(diffPeak);
//Moving Average Initition

if (avgInitiate <= initialPeaks){
  if (diffPeak >= minDiff){
    if (diffPeak > movingAverage[0]){
      movingAverage[0] = diffPeak;
    }
    if (diffPeak > movingAverage[1]){
      movingAverage[1] = diffPeak;
    }
    if (diffPeak > movingAverage[2]){
      movingAverage[2] = diffPeak;
    }
    if (diffPeak > movingAverage[3]){

```

```

    movingAverage[3] = diffPeak;
}
if (diffPeak > movingAverage[4]){
    movingAverage[4] = diffPeak;
}
avgInitiate++;
}
}

average = calculateAverage(movingAverage[0], movingAverage[1],
movingAverage[2], movingAverage[3], movingAverage[4]);

//Serial.println(average);
if (avgInitiate > initialPeaks){
    if (diffPeak >= (average*variance)){
        //Serial.println("PULSE!!!");
        peakSwitch = peakMax;
        peakMax = peakMin;
        peakMin = peakMax;
        lightTime = millis();
        lastBeat = lightTime;
        digitalWrite(ledPin, HIGH);
        Serial.println(lastBeat);

        for (int a = 0; a < index; a++){
            movingAverage[4-a] = movingAverage[3-a];
        }
    }
}

```

```
    movingAverage[0] = diffPeak;  
  }  
}  
}
```

```
double calculateAverage(double v, double w, double x, double y, double z){  
    double sum = v + w + x + y + z;  
    return (sum/index);  
}
```