10-19-2015

Multiple detector wireless FNIRS sensor to detect hemodynamic activity

Sivaram Karra

Follow this and additional works at: http://rdw.rowan.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

http://rdw.rowan.edu/etd/510

This Thesis is brought to you for free and open access by Rowan Digital Works. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Rowan Digital Works. For more information, please contact LibraryTheses@rowan.edu.
MULTIPLE DETECTOR WIRELESS FNIRS SENSOR TO DETECT HEMODYNAMIC ACTIVITY

by

Sivaram Karra

A Thesis

Submitted to the
Department of Electrical and Computer Engineering
College of Engineering
In partial fulfillment of the requirement
For the degree of
Master of Science in Electrical and Computer Engineering
at
Rowan University
August 19, 2015

Thesis Chair: Linda Head, Ph.D.
Acknowledgments

I would like to express my gratitude to my advisor and committee chair, Associate Professor Linda Head of the Department of Electrical and Computer Engineering at Rowan University for the continuous support that she has provided to me to complete my research and thesis. Her guidance and scientific approach have helped me in many ways to complete this task and fulfill my dream of becoming a postgraduate.

I sincerely thank all the staff of Henry M. Rowan College of Engineering for their help and motivational support provided me throughout the Masters education.
Abstract

Sivaram Karra
MULTIPLE DETECTOR WIRELESS FNIRS SENSOR TO DETECT HEMODYNAMIC ACTIVITY
2014/15
Linda M. Head, Ph.D.
Master of Science in Electrical and Computer Engineering

An established, non-invasive and flexible technique, "functional Near Infra-Red Spectroscopy" (fNIRS), has opened new opportunities for the study of brain functionality. This powerful method uses the optical properties of a tissue at multiple near infra-red wavelengths to dynamically determine the levels of both oxygenated hemoglobin (HbO$_2$) and deoxygenated hemoglobin (Hbr). Traditional techniques such as functional Magnetic Resonance Imaging (fMRI), Positron Emission Tomography (PET), Electroencephalography (EEG) and Event Related brain Potentials (ERP) have increased our understanding of brain functionality, however, the basic limitations of these technologies such as high cost of equipment, difficulty of operation and levels of radiation make access limited for researchers exploring brain function in real-time environments. The purpose of this work is to demonstrate the enhancement of the existing single photodiode wireless fNIRS sensor to two photodiode wireless fNIRS sensor for brain imaging using Bluetooth technology. This enhanced design allows arbitrary positioning of sensors that are able to detect HbO$_2$ and Hbr response at multiple positions in the cerebral cortex.
# Table of Contents

Abstract ................................................................................................................................. iv
List of Figures ........................................................................................................................ vii
List of Tables .......................................................................................................................... ix
Chapter 1: Introduction ....................................................................................................... 1
Chapter 2: Background ....................................................................................................... 6
  2.1 Near Infrared Spectroscopy (NIRS) ............................................................... 7
  2.2 Types of Techniques in fNIR .......................................................................... 9
  2.3 Traditional Imaging Techniques ...................................................................... 9
  2.4 fNIRS Light Path .............................................................................................. 11
  2.5 Modified Beer Lambert Law (MBLL) .......................................................... 12
  2.6 Applications of fNIRS .................................................................................. 14
Chapter 3: Approach ....................................................................................................... 16
  3.1 Sensor Node Block Diagram .......................................................................... 16
  3.2 Reduction of Circuitry ...................................................................................... 17
  3.3 Major Components in the Circuit .................................................................. 18
    3.3.1 Flash USB microcontroller C8051F342-GM ..................................... 18
    3.3.2 Bluetooth radio LMX9838SB ............................................................. 19
    3.3.3 NIR LED ......................................................................................... 20
    3.3.4 Photodiode OPT101 ......................................................................... 21
  3.4 Programmable gain amplifier (PGA) ............................................................... 22
    3.4.1 Programmable gain amplifier PGA112 ............................................ 23
    3.4.2 SPI commands for PGA112 ............................................................. 25
    3.4.3 Scope of PGA116 ........................................................................... 27
    3.4.4 Common features of PGA112 and PGA116 .................................. 28
    3.4.5 Comparison between PGA112 and PGA116 .................................. 29
  3.5 Updating Firmware .............................................................................................. 30
    3.5.1 Coding for two photodiode wireless sensors ..................................... 31
    3.5.2 Requirements for the code .................................................................. 31
    3.5.3 Flow chart for the code ...................................................................... 32
Table of Contents (continued)

3.6 Matlab GUI ..................................................................................................................33
  3.6.1 Communication between MATLAB and sensor ....................................................34
  3.6.2 Graphical controls .................................................................................................34
Chapter 4: Results ..............................................................................................................37
  4.1 Programming the Wireless fNIRS Device ...............................................................38
  4.2 Software Testing .......................................................................................................39
  4.3 Running the fNIRS Sensor .......................................................................................40
  4.4 Discussion ..................................................................................................................44
Chapter 5: Conclusions and Recommendations for Future Work .............................45
  5.1 Accomplishments .....................................................................................................46
  5.2 Future Scope .............................................................................................................46
References .............................................................................................................................48
Appendix A: List of Acronyms ..........................................................................................52
Appendix B: Firmware Code ...............................................................................................54
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1 University of Tokyo’s fNIRS device for Schizophrenia</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2 Wireless fNIRS device with one photodiode</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.3 Wireless fNIRS device after packaging</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.1 Near-infrared absorption characteristics of $HbO_2$ and $Hbr$</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.2 fNIRS light path with respect to separation</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3.1 Block diagram for two photodiode wireless fNIRS sensor</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.2 Pin Connections for C8051F342-GM Microcontroller</td>
<td>19</td>
</tr>
<tr>
<td>Figure 3.3 LMX9838 Bluetooth circuit’s pin connections</td>
<td>20</td>
</tr>
<tr>
<td>Figure 3.4 Epitex NIR LED L4x730-4X805-4X850-40Q96-I top view</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3.5 Texas Instrument’s Photodiode op-amp OPT101</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3.6 Photodiode op-amp OPT101 pin connections</td>
<td>22</td>
</tr>
<tr>
<td>Figure 3.7 Texas Instrument’s PGA112 block diagram</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3.8 Pin connections of PGA112</td>
<td>25</td>
</tr>
<tr>
<td>Figure 3.9 Pin diagram for PGA116</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.10 Annotated wireless fNIRS sensor MATLAB GUI exerciser screenshot</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.11 USB Bluetooth radio for non-Bluetooth machines</td>
<td>36</td>
</tr>
<tr>
<td>Figure 4.1 HP 7 inch Windows 8.1 tablet</td>
<td>38</td>
</tr>
<tr>
<td>Figure 4.2 Setup for programming wireless fNIRS device</td>
<td>38</td>
</tr>
<tr>
<td>Figure 4.3 Flash USB type B male socket connections</td>
<td>39</td>
</tr>
<tr>
<td>Figure 4.4 Screenshot of Silicon labs emulator while debugging</td>
<td>40</td>
</tr>
<tr>
<td>Figure 4.5 Powered wireless fNIRS device indicating red LED</td>
<td>41</td>
</tr>
<tr>
<td>Figure 4.6 Screen shot of adding wireless fNIRS device to system</td>
<td>41</td>
</tr>
</tbody>
</table>
List of Figures (continued)

Figure 4.7  Screen shot of serial port device *COM* port ............................................. 42

Figure 4.8  Screen shot of Matlab *GUI* to connect the wireless *fNIRS* device ............. 43

Figure 4.9 Screen shot of working of two photodiode detector plot .............................. 44
List of Tables

Table | Page
---|---
Table 2.1 Classification of Infrared band | 6
Table 2.2 Comparison of Imaging Techniques | 10
Table 3.1 SPI commands to select channels in PGA112 | 26
Table 3.2 Gain settings for PGA112 | 26
Table 3.3 Channel Selection for PGA116 | 28
Table 3.4 Comparison between PGA112 and PGA116 | 29
Table 3.5 SPI commands used in PGA112 | 30
Table 3.6 Process Flow for Firmware | 33
Chapter 1

Introduction

Human brain activity can be measured through hemodynamic responses that are associated with neuron behavior [1] [2]. Regular metabolic activities performed by tissues and organs require oxygen which is supplied by Red Blood Cells (RBC). RBC carry the dissolved oxygen molecules with the help of a binding protein called Hemoglobin (Hb) which constitutes one third of the weight of the RBC and has two derivatives; Oxygenated Hemoglobin (HbO$_2$) and deoxygenated hemoglobin (Hbr) [3]. HbO$_2$ is carried to body tissues which are in need of oxygen through the arteries while deoxygenated blood is sent to lungs and heart through the veins. HbO$_2$ and Hbr exhibit specific optical properties at wavelengths in the Near Infrared spectrum ranging between 700nm and 1000nm known as the optical window [4]. In this range biological tissues have high scattering characteristics that are used along with light and energy absorption by HbO$_2$ and Hbr to make NIRS (Near-Infrared Spectroscopy) a distinctive neuroimaging technology. By using the Beer Lambert Law physiological changes can be determined by measuring changes in reflected light and energy.

According to Kocsis et al., “The light absorption of biological tissues is relatively low in the NIR window range (650-900nm); therefore NIR light can be applied to gain physiological information about deeper tissue layers in a noninvasive manner” [5].

The goal of this project is to develop a multi-detector, wearable, wireless functional NIRS (fNIRS) sensor that is flexible enough to measure the brain’s hemodynamic data in the prefrontal cortex in infants and adults at multiple brain sites and with minimal inconvenience to the subject. Many other existing fNIRS devices are
designed to perform in a manner which restricts sensor placement and subject movement [6]. The sensor presented here allows arbitrary placement of source and detector as well as relative freedom of movement for the subject.

Utilizing an NIR source with a varying number of detectors that can be individually placed on the pre-frontal cortex will enhance the ability of researchers to examine brain functionality. Previously designed fNIRS devices are customized according to the tasks and physiology by soldering the photodiodes in a specific pattern so that they can’t be rearranged for different experiments making reusability for other experiments challenging [7].

Figure 1.1 University of Tokyo’s fNIRS device for Schizophrenia [8]
As an example, the University of Tokyo’s wearable NIRS sensor system shown in Figure 1.1 is designed for the study of Schizophrenia, a chronic and severe brain disorder. Schizophrenia accounts for 2.3% of all worldwide diseases and unerring treatment for such mental disorders has not yet been found [8]. Figure 1.1A illustrates source and detector placement on the human scalp; 1.1B shows a commercially available fNIRS machine used in clinical laboratories; 1.1C and 1.1D show 3x11 probe settings having source and detectors arranged at regular distance points to detect the reflected signal from the pre-frontal cortex. During a verbal fluency task (VFT) measurement on the pre-frontal cortex from the schizophrenia study showed the presence of different characteristic waveform patterns [8]. Activation patterns of schizophrenia were compared with healthy controls and the results were published in 1994 [9]. There is the need for identification of the segments of the brain which are correlated with disorders such as schizophrenia, however, the source and detector placements prioritize a single, average physiology rather than allowing placement for individual subject and/or site locations.

As an initial step towards mobility, arbitrary placement and wireless implementation, a NIRS sensor system was designed which consists of a network of identical general-purpose wireless nodes, which includes the control system electronics. This system, shown in Figure 1.2 allows researchers to place each sensor according to the experimental design instead of redesigning and fabricating an entire sensor array.
The circuit shown in Figure 1.2 has one *NIR LED* and one photodiode and is capable of taking readings at 730nm and 850nm wavelengths [10]. The size of the printed circuit board is 2.5cm x 4cm. The circuit is enclosed in a plastic casing as shown in Figure 1.3 to protect the circuitry and its components from static charges which may cause failure in sensitive electrical components. This also protects the subject from direct contact with circuit components.

*Figure 1.2 Wireless fNIRS device with one photodiode [10]*

*Figure 1.3 Wireless fNIRS device after packaging [10]*
The present work focuses on development of a multiple photodiode version of the sensor in Figure 1.2 that can obtain and transmit data acquired from multiple locations. The data is transmitted to a wireless device such as a laptop or tablet that has Matlab software installed. As a proof of concept, a single photodiode is used where output data plots are cycled for every 10 output points between two photodiode channels, one populated, one open. This same principle and software solution will work for multiple photodiode detectors by increasing the number of photodiodes and amplification channels.
Chapter 2

Background

Sir William Herschel, an astronomer, was the first to discover Infrared (IR) light [11]. The IR name was given because of its lower frequency with respect to the visible spectrum. He conducted various temperature based tests on all colors of the visible spectrum starting from Violet to Red and observed that temperature increases significantly when it reaches visible red light [11]. He investigated the spectrum beyond visible red, noting that he found radiating heat waves which eventually came to be known as the Infrared band [11].

The Infrared band was further classified into three regions named Near Infrared (NIR), Mid Infrared and Far Infrared. Their assigned approximate wavelength ranges are indicated in Table 2.1.

Table 2.1

*Classification of Infrared band* [12]

<table>
<thead>
<tr>
<th>Type of Infrared</th>
<th>Wavelength Range</th>
<th>Usage for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near-Infrared</td>
<td>700nm - 5000nm</td>
<td>Bio-Medical, Pharmaceutical and food processing industries [13].</td>
</tr>
<tr>
<td>Mid-Infrared</td>
<td>5000nm - 40,000nm</td>
<td>Defense, space communications, Light Detection and Ranging (LIDAR) [13][14].</td>
</tr>
<tr>
<td>Far-Infrared</td>
<td>40,000 - 350,000nm</td>
<td>Astronomy, studying cosmic energy and galaxies [15].</td>
</tr>
</tbody>
</table>
2.1 Near Infrared Spectroscopy (NIRS)

Though Sir William Herschel discovered IR radiation in 1800 [16] it took more than a decade to make use of it on a major scale. The seeds planted by William Herschel in the discovery of NIR led to more applications in various fields. For instance, NIRS is used as a tool for quality testing of food products [17]. The first international conference on NIRS was held in Norwich, U.K. in 1987 and the slogan resulting from the conference was “high time for the giant to wake up” [18].

Later, with the advancement in electronic and optical components, NIR spectroscopy expanded into numerous other fields such as neuroimaging, physiology and sports medicine. The use of NIRS for neuroimaging, where brain activity is measured through hemodynamic responses, is known as functional Near Infrared Spectroscopy (fNIRS) [2].

The NIR range, between 700nm to 900nm is considered to be the spectral window where photon absorption is low in tissue. Human blood contains hemoglobin of two different oxygenation states: oxyhemoglobin (HbO₂) and deoxyhemoglobin (Hbr). These two different states of hemoglobin exhibit different absorption spectra that are normally represented in terms of molar extinction coefficients which define how strongly each species absorbs light at a given wavelength per mole concentration. In the NIR optical windows range most of the biological tissues exhibit transparency to light due to their water content. HbO₂ and Hbr chromophores do absorb some amount of light in this optical window [19]. Absorption characteristics for HbO₂, Hbr and water in the optical window are as shown in Figure 2.1.
When NIR light is introduced to the prefrontal cortex through the tissue, light particles are transmitted in the form of photons. Because of the tissue’s optical properties most photons will pass through the prefrontal cortex and be scattered isotropically. Due to the isotropic scattering pattern, some photons will be reflected back through the tissue where they can be detected. This process results in the ability to monitor absorption of the NIR light in the prefrontal cortex as will be further described below [19].

![Figure 2.1 Near-infrared absorption characteristics of $HbO_2$ and $Hbr$ [19]](image)

The brain constitutes only 2% of body weight but consumes 20% of the total energy consumed by the body [20], this is due to the neuronal activity performed with the help of glucose and hemoglobin coming through blood. When more energy is required by the brain, neurons in a particular region fire causing the local capillaries to expand transporting more blood to the brain increasing the amount of oxygenated hemoglobin [20]. The more complex the task, more energy is required by the neurons to handle the
tasks. This eventually increases the amount of blood flow to brain allowing investigation of brain functionality using *NIR* absorption characteristics.

### 2.2 Types of Techniques in *fNIR*

There are three different types of *fNIR* techniques. They are as follows:

- Time-Resolved Systems (*TRS*)
- Frequency-Domain Systems (*FDS*)
- Continuous Wave Spectroscopy Systems (*CWSS*)

They have their own advantages and disadvantages with respect to cost, convenience and reliability. Of the above mentioned three techniques *CWSS* is widely used because of its inexpensive hardware and safer approaches by using *LEDs* instead of laser technology [19]. In contrast *TRS* and *FDS* provide information in terms of phase and amplitude of the incident light, which is more precise than *CWSS* but *CWSS* is adequate for the required level of imaging [19].

### 2.3 Traditional Imaging Techniques

Traditional imaging techniques such as functional Magnetic Resonance Imaging (*fMRI*), Positron Emission Tomography (*PET*), Electroencephalography (*EEG*), Computed Tomography (*CT scan*) have been used to increase our understanding of brain functionality. However, the basic limitations with these technologies such as cost of equipment, operational complexity and radiation danger represent limiting conditions to researchers’ ability to explore brain functionality. Some of these techniques are shown with their Advantages, Disadvantages and Applications in Table 2.2
Table 2.2

Comparison of Imaging Techniques

<table>
<thead>
<tr>
<th>IMAGING TECHNIQUE</th>
<th>ADVANTAGES</th>
<th>DISADVANTAGES</th>
<th>APPLICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Rays</td>
<td>• Images will come in less than an hour.</td>
<td>• Organs are sensitive to radiation.</td>
<td>• Mostly used for bone structure analysis and during fractures</td>
</tr>
<tr>
<td></td>
<td>• Painless and no special preparation needed.</td>
<td>• Not advisable during pregnancy.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computed Tomography- (CT)</td>
<td>• CT scan provides detailed images of blood vessels and internal organs.</td>
<td>• Costlier (Starts at 1 lakh dollars).</td>
<td>• To check the blood vessels, bones, and inner parts.</td>
</tr>
<tr>
<td></td>
<td>• Very quick in giving images (minutes).</td>
<td>• Contrast gel is injected into vein before the scan.</td>
<td>• More helpful in the events of accidents to determine the damaged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Person should not move.</td>
<td>parts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional Magnetic Resonance Imaging- fMRI</td>
<td>• Gives activation maps of brain which are involved in the activation process.</td>
<td>• Working principle is purely based on blood flow and difficult to interpret.</td>
<td>• Neurosurgical planning.</td>
</tr>
<tr>
<td></td>
<td>• It doesn’t uses radiation.</td>
<td>• More Costlier</td>
<td>• Diagnosis and evaluation of functional brain disorders</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positron Emission Tomography- PET</td>
<td>• Shows the blood flow, oxygen and glucose metabolism in the working brain.</td>
<td>• Radioactivity decays rapidly, so short tasks can only be performed.</td>
<td>• Monitoring brain activity while performing different tasks.</td>
</tr>
</tbody>
</table>

[21] [22] [23] [24]
Table 2.2 (continued)

<table>
<thead>
<tr>
<th>IMAGING Technique</th>
<th>ADVANTAGES</th>
<th>DISADVANTAGES</th>
<th>APPLICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fNIRS</strong></td>
<td>• Portable, inexpensive and able to take readings even the subject is in motion.</td>
<td>• Low spatial resolution. • limited to brain surface</td>
<td>• Measuring brain dynamics, cognitive workload during extended memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*fNIRS’s* low cost, portability and safety make it an attractive alternative to traditional techniques. *fNIRS* uses the optical properties of light in tissue at multiple near infrared wavelengths to dynamically determine the changes in volume and concentration of oxygenated hemoglobin (*HbO₂*) and deoxygenated hemoglobin (*Hbr*).

### 2.4 *fNIRS* Light Path

The paper written by Scott C Bunce et al. in 2006 [19] discussed the *fNIRS* systems and their optical properties involved in measuring the hemodynamic activity of the brain. When the NIR light is projected on the prefrontal cortex, it travels through the skin and hits the inner tissues and scatters in all directions; the light that is detected at the detector is the reflected light from the tissue. The more distance maintained between the
source and detector the more depth is probed by the detector as shown in Figure 2.2, although more than 2.5 cm to 3 cm of separation between NIR source and photodiode detector causes excessive attenuation of the reflected signal.

![Figure 2.2 fNIRS light path with respect to separation](image)

**Figure 2.2** fNIRS light path with respect to separation [27]

### 2.5 Modified Beer Lambert Law (MBLL)

The Modified Beer Lambert Law (MBLL) is the basis of continuous wave NIRS. Change in the light attenuation is proportional to the change in the concentrations of tissue chromophores $HbO_2$ and $Hbr$ [5]. The reflected signals at two or more wavelengths are measured and concentration changes are calculated by assuming scattering loss is constant and absorption of the tissue changes are homogenous [5]. “The traditional approach of continuous wave NIRS utilizes one or more light source-detectors, the
illuminated tissue is considered as optically homogenous and applies the modified Beer-Lambert law "page N92 [5]. The Beer Lambert Law states:

\[ A = \ln \frac{I_{\text{inc}}}{I_{\text{det}}} = L \mu_a + G \]  

Eq. 1

Where A is the attenuation

\( I_{\text{inc}} \) is the Incident light Intensity

\( I_{\text{det}} \) is the detected light intensity

L is the total mean path length of detected photons

\( \mu_a \) is the absorption coefficient of the tissue

G is geometry-dependent factor representing Intensity loss

The differential form of MBLL holds with G held constant, then absorption changes homogenously in the illuminated tissue volume:

\[ \Delta A = \ln \frac{I_{\text{inc}}}{I_{\text{det}}} = L \Delta \mu_a \]  

Eq. 2

The attenuation change can be calculated from the detected intensity values of two different states of the tissue, and is proportional to the change of absorption. Then the resultant will be the weighted sum of the change in the concentration of tissue chromophores, mainly \( HbO_2 \) and \( Hbr \):

\[ \Delta \mu_a = \left( \alpha_{HBO_2} \Delta C_{HBO_2} + \alpha_{Hbr} \Delta C_{Hbr} \right) \]  

Eq. 3
Where \( \alpha \) denotes the specific absorption coefficients of chromophores. Let the wavelengths of measurements be \( \lambda_1 \) and \( \lambda_2 \). The changes in the concentrations can be estimated by:

\[
\Delta C_{HbO_2} = \frac{\alpha_{HbO_2} \lambda_1 \Delta A_{\lambda_2} - \alpha_{HbO_2} \lambda_2 \Delta A_{\lambda_1}}{\alpha_{HbO_2} \lambda_1 \alpha_{HbO_2} \lambda_2 - \alpha_{HbO_2} \lambda_2 \alpha_{HbO_2} \lambda_1} \quad \text{Eq.4}
\]

\[
\Delta C_{Hbr} = \frac{\alpha_{HbO_2} \lambda_1 \Delta A_{\lambda_2} - \alpha_{HbO_2} \lambda_2 \Delta A_{\lambda_1}}{\alpha_{HbO_2} \lambda_1 \alpha_{HbO_2} \lambda_2 - \alpha_{HbO_2} \lambda_2 \alpha_{HbO_2} \lambda_1} \quad \text{Eq.5}
\]

When the changes of attenuation are measured at more than two wavelengths, concentration changes can be determined by linear least-squares fitting [28].

2.6 Applications of fNIRS

Initially, NIRS was predominantly used in Agriculture and soil analysis for assessing soil fertility and nutrients and in the pharmaceutical industry to measure the product’s purity [29]. NIRS was first used in pulse-oximetry by Takuo Aoyagi in 1972 [30]. In 1977, Frans Jobsis reported that a high degree of transparency is observed in brain tissue in the NIR range (700nm - 1000nm) using trans-illumination spectroscopy [31]. With the discovery of the functional activation of the human cerebral cortex, NIRS spectroscopic research became more widespread. Over the last decade there have been an increasing number of publications on NIRS applications and many clinical instruments have entered the market place.

These advances have made fNIRS an appropriate technique for cortical brain imaging using changes in \( HbO_2 \) and \( Hbr \) to monitor cortical activation, while a subject performs cognitive tasks. fNIRS is also used in sports medicine where athletes muscle
hemodynamic responses are monitored during exercise. Bhambhani [32] mentioned that there are systematic changes in the muscle and cerebral hemodynamic responses which are coincident with respiratory alterations occurring during incremental exercise. There is heavy demand for portable, inexpensive diagnostic and patient monitoring devices which indicates that this technology is in transition from clinical research to advanced development of clinical and consumer applications.
Chapter 3

Approach

The motivation for this work is to enhance the capability of an existing single node wireless fNIRS sensor to serve multiple nodes at different time instants without increasing physical complexity and allowing the multiple sensor nodes to transmit data wirelessly to devices like laptops and tablets. In order to fulfill these requirements the existing circuits’ component characteristics and parameters are very important. Each individual component of the existing device is taken into consideration with respect to its limitations and ranges while trying to develop a new design for a multi-node wireless fNIRS sensor. To validate the concept of multi-node operation, a two photodiode design is developed as a proof-of-concept of the multi-node design. First, the block diagram for this two photodiode design is developed to provide a blue print for the process flow.

3.1 Sensor Node Block Diagram

The following block diagram, shown in Figure 3.1, represents the connections of a two detector node wireless fNIRS sensor. All the components of the device utilize the common battery power with a voltage of 3.7 volts and 210 mA current. In addition, the device can be charged through a micro USB connector. Firmware is downloaded onto the device using the micro USB connector and cable. The NIR LED requires high current so an LED driver circuit is required to increase the power which is implemented by using a buck-boost converter. A programmable gain amplifier provides variable gain along with multiple channels for the photodiodes. A bi-colored LED is used, indicating whether the device is connected to the system or not. On a successful establishment of connection, red color turns into green. The Bluetooth radio is connected to the micro-controller to
transmit the sensor data in the form of packets to the tablet or personal computer using the standard *IEEE 802.15.1* protocol.

![Diagram of two photodiode wireless fNIRS sensor](image)

*Figure 3.1* Block diagram for two photodiode wireless fNIRS sensor

### 3.2 Reduction of Circuitry

Instead of using an entire circuit for each sensor node, the same component resources are shared for multiple photodiodes. The firmware is written to run the device continuously by taking the output from the photodiodes cyclically. The existing control circuit can support a maximum of two photodiodes only but by upgrading the programmable amplifier with a multi-channeled version, a maximum of 10 photodiodes
can be connected to this circuit. In addition, updating firmware is straightforward because the same two photodiode sensors’ firmware can be used by including the other photodiodes’ channels in the loop of the program thereby increasing the efficiency of the device and cutting down the additional circuitry cost.

3.3 Major Components in the Circuit

In order to write the firmware for the multi-node wireless fNIRS sensor, the existing component specifications for the single-node sensor are studied and analyzed to determine the compatibility for the multi-node sensor. The components are described in the following sections.

3.3.1 Flash USB microcontroller C8051F342-GM. The Silicon Lab’s C8051F342-GM microcontroller is the heart of the sensor node responsible for acquiring, processing and transmitting the NIRS data [10]. It controls and drives all the other components in the circuit. The clock cycle for this microcontroller is set for 1 millisecond. In Figure 3.2 the various pin connections associated with the microcontroller are shown.
3.3.2 Bluetooth radio LMX9838SB. The National Semiconductor LMX9838SB Bluetooth serial port module has 2.4 GHz radio, antenna, crystal and can support up to 7 Bluetooth devices as slaves in master mode [33]. It belongs to power class 2 emitting a maximum output power of 2.5 mW and range of up to 10 meters [33]. The pin connections associated with LMX9838SB in the existing single node sensor are shown in the Figure 3.3.
The pins TXD, RXD, CTSN and RTSN of the C8051F342 microcontroller are connected to TXD, RXD, RTS and CTS pins of the LMX9838 Bluetooth module, which facilitates smooth flow of data bits from micro controller to Bluetooth module.

3.3.3 NIR LED. The NIR LED used is L4x730-4X805-4X850-40Q96-I, having three different NIR wavelengths at 730nm, 805nm, 850nm. For the experiment purpose only 730nm, 850nm were used in the circuit. It has 8 pins with a heat sink pedestal sealed on a flat glass can as shown in Figure 3.4. Pulse forward current ranges between 200mA and 500mA while forward voltage ranges between 9.0V to 6.5V relative to 730nm, 850 nm [34].

Figure 3.3 LMX9838 Bluetooth circuit’s pin connections [10]
3.3.4 Photodiode OPT101. The OPT101 is a monolithic photodiode and single supply transimpedance amplifier with $1\,\Omega$ internal feedback resistor and supply voltage ranging from 2.7V minimum to 36V maximum providing output voltage with respect to change in intensity of light [35]. This is an 8 pin surface mountable IC.

Figure 3.5 Texas Instrument’s Photodiode op-amp OPT101 [36]
The parallel combination of the 324 kΩ resistor and 330 pF capacitor is helpful for low light sensitivity applications. OPT101 is powered by +3.3V supply using V+ pin. The high impedance CH1 of the PGA112 (described below in section 3.4) is driven by VDIODE of OPT101. The MOSFET DMN5L06VAK, shown below in Figure 3.6 is used to eliminate the quiescent current when the photodiode is not in use [36] [10].

![Figure 3.6 Photodiode op-amp OPT101 pin connections [10]](image)

### 3.4 Programmable Gain Amplifier (PGA)

A PGA is an electronic amplifier whose gain can be controlled by external digital signals such as a Serial Peripheral Interface (SPI). In the sensor control circuit the PGA is
used to maximize the resolution of the Analog to Digital Converter (ADC) after the amplification of the fixed gain output of the OPT101 trans-impedance amplifier.

3.4.1 Programmable gain amplifier PGA112. The Texas Instruments PGA112 shown in Figure 3.7, provides binary gains of 1, 2, 4, 8, 16, 32, 64, and 128, and substantially increases the dynamic range of the 10-bit ADC in the microcontroller with two input channels. These two channels are connected with the outputs of OPT101 photodiode.

Figure 3.7 Texas Instrument’s PGA112 block diagram [37]
In the existing circuit, the PGA112 has two inputs with CH1 connected to the output of the trans-impedance amplifier and CH0 connected to the interval voltage reference in the microcontroller. The output gain is set by the microcontroller via the SPI. The three pin SPI interface allows communication with the master device to change gain and channels. The CH1 pin is driven by the VDIODE signal from the OPT101. The CH0/VCAL pin is driven by the VREF output of the C8051F342 microcontroller and is used to perform a gain calibration of the combined PGA112 and C8051F342 ADC signal chain. The VREF pin is tied to GND since the output offset is handled by the OPT101 VBIAS signal [10]. Therefore, CH0 (previously unused) is the choice for a second photodiode connection.

The C8051F342 SPI interface is connected to the PGA112’s SCLK, DIO, and CS pins and is used to select the input channel, set the gain, and enter and leave shutdown mode as shown in Figure 3.8. Consequently the switching action for the channels can be performed by using SPI of the microcontroller.
3.4.2 SPI commands for PGA112. Using the Serial Peripheral Interface, channels can be selected by using the required commands. Depending on the channel number a unique command is selected to assign control to that channel. The following two tables provide the commands for channel selection and gain selection.
Table 3.1

*SPI commands to select channels in PGA112 [37]*

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>SPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>0</td>
<td>0</td>
<td>CH1</td>
<td>CH0</td>
<td>Write</td>
</tr>
</tbody>
</table>

Table 3.2

*Gain settings for PGA112 [37]*

<table>
<thead>
<tr>
<th>G3</th>
<th>G2</th>
<th>G1</th>
<th>G0</th>
<th>Binary Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>128</td>
</tr>
</tbody>
</table>

To select the *CH0* 0x00 is used whereas 0x01 is used to select *CH1* with gain 0 using *SPI* as shown in Table 3.1. So, each time it is desired to switch the input channels these command should be sent to the PGA112 through *SPI*. To change the gain, G3 G2 G1 G0’s hexadecimal values must be sent into the PGA112.
3.4.3 Scope of PGA116. In the expanded version of the fNIRS sensor, PGA116 will replace the existing PGA112. The PGA116 offers 10 analog inputs and a four pin SPI interface with daisy-chain capability. The functionality of this device is same as the PGA112 but with the addition of 8 more channels altogether giving 10 channels as shown in Figure 3.9. At any given point of time only one channel can be used to perform gain operations. The same concept as used for the PGA112 can be applied to this device, which will further extend the number of OPT101 photodiode detectors.

![Pin diagram for PGA116](image)

*Figure 3.9 Pin diagram for PGA116 [37]*

Therefore, the proposed PGA116 can be used in the place of PGA112 by appending the new firmware with command lines for all the 10 channels as shown in Table 3.3. The PGA112 and PGA116 have common features as mentioned below.
Depending on the number of channels either PGA112 or PGA116 is selected for applications.

Table 3.3

*Channel Selection for PGA116* [37]

<table>
<thead>
<tr>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>PGA116</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CH0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CH1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CH2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>CH3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>CH4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>CH5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>CH6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CH7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CH8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CH9</td>
</tr>
</tbody>
</table>

### 3.4.4 Common features of PGA112 and PGA116.

The PGA112 and PGA116 have common features as mentioned below. Depending on the number of channels either PGA112 or PGA116 is selected for applications.

- They have maximum offset value of 100µV/°C and typical of 25 µV/°C
- Gain switching time is 200 ns
- Both of them supports *SPI* Interface
- Maximum gain error for G > 32 is 0.3%, G ≤ 32 is 0.1%
- Low noise of 12nV/√Hz
- Zero drift with typical of 0.35 µV/°C and maximum of 1.2 µV/°C

### 3.4.5 Comparison between PGA112 and PGA116.

To program the Programmable Gain Amplifiers, *SPI* pins *SCLK, DIO, CS* are used which are common for both PGA112 and PGA116. Selection data made by the microcontroller is sent through the *DIO* pin to the *PGA*.

Table 3.4

*Comparison between PGA112 and PGA116 [37]*

<table>
<thead>
<tr>
<th>Feature</th>
<th>PGA112</th>
<th>PGA116</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Layout</td>
<td>MSOP-10 Pin</td>
<td>TSSOP-20 Pin</td>
</tr>
<tr>
<td>Common pins</td>
<td>*AV&lt;sub&gt;DD&lt;/sub&gt;, <em>DV&lt;sub&gt;DD&lt;/sub&gt;, GND, VOUT, CS</em></td>
<td>*AV&lt;sub&gt;DD&lt;/sub&gt;, <em>DV&lt;sub&gt;DD&lt;/sub&gt;, GND, VOUT, CS</em></td>
</tr>
<tr>
<td>SPI common Pins</td>
<td>SCLK, DIO, CSN</td>
<td>SCLK, DIO, CSN</td>
</tr>
<tr>
<td>Channel Pins</td>
<td>CH0, CH1</td>
<td>CH0, CH1, CH2, CH3, CH4, CH5, CH6, CH7, CH8, CH9</td>
</tr>
<tr>
<td>H/W Enable Pin</td>
<td>No</td>
<td>Pin 13 enables device on Logic Low</td>
</tr>
</tbody>
</table>

Channel and gain selection is programmed in the microcontroller which will communicate to the *PGA* through this *DIO* pin. By writing hexadecimal values into *SPIODAT* the channel and gain of the *PGA* can be set.
Table 3.5

*SPI commands used in PGA112*

<table>
<thead>
<tr>
<th>$SPIODAT =$ 0x2A</th>
<th>$MSB$</th>
<th>$SPI$ command to write into $PGA$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SPIODAT = (0x00 &lt;&lt; 4)+0x01$</td>
<td>$LSB$</td>
<td>$SPI$ command to select (gain)+ channel</td>
</tr>
</tbody>
</table>

For PGA112 and PGA116 the gain range is common, from 0x00 to 0x07. For PGA112 channels can be 0x00 or 0x01 as it has 2 channels, whereas PGA116 has 10 channels their range is from 0x00 to 0x09 in the last 8 bits of $LSB$. When the PGA116 is being used a maximum of 10 photodiodes can be connected. Each of the photodiodes’ output is connected to these 10 channels, the output of these 10 channels is switched by loading the $SPIODAT$ command every time as mentioned above. The hardware connections for both the PGA112 and PGA116 remain the same except the photodiode channels.

### 3.5 Updating Firmware

The new firmware was developed to enable the existing device to work for two photodiode detectors. For this application the existing device’s firmware is used as the base and blocks of code were added to make it capable of operating multiple photodiodes. In the existing device firmware, Special Function Registers ($SFR$’s) are initialized then followed by configuring Input and Output ($I/O$) port pins. After initializing the configuration, the program enters an infinite loop which waits for device parameter values to be entered by user through Matlab GUI using $get\_key()$. Each of these parameters are uniquely identified by using alphabets as passing key to route them to their respective blocks of code. Timer 2 is enabled after the device is configured and
generates an interrupt every millisecond. Then, based on the device settings, sensor data is updated to the Matlab GUI by using the LMX9838SB UART [10].

3.5.1 Coding for two photodiode wireless sensors. To meet the goal of switching the photodiodes dynamically while the device is in operation, ISR’s are the best way to handle the dynamic changes. Read ambient (READAMB) is the first state in the interrupt loop to send the readings from device followed by SET and READ states of 730 nm and 850 nm NIR LED wavelengths. Timer 2 in C8051F342 micro controller vectors itself as interrupt 5. By using this interrupt, when READAMB is asserted an IF condition is written to check the present channel. After 5000 clock cycles (5 seconds), execution control clears SPIF and SPIODAT is loaded with a hexadecimal value of gain and channel by writing NSSMD0 =0 (slave mode) to the PGA112. (See Appendix B for further documentation associated with the code.)

Execution flow returns and the appropriate wavelength source, 730nm or 850nm, is set. After completion of 5 seconds of data acquisition the channel is again switched. This process is repeated continuously. More information is provided in the documentation of code in Appendix B. In the process of writing the code, the existing firmware for the microcontroller C8051F342 is modified only at ISR of timer2 interrupt5 and global variables for count operations and switching channels are declared.

3.5.2 Requirements for the code. The firmware for the two-photodiode wireless fNIRS was written in ANSI C standard, enabling the potential for developing sophisticated on-board NIR measurement. This code is designed to meet the following requirements.

- To initialize and configure the registers and ports
• Enabling *NIR LED* to drive with its rated power requirements

• Providing the transceiver functionality in terms of exchanging data using Bluetooth

• Converting analog information to digital at PGA112

• Setting the channels and gain of PGA112

• Switching channels dynamically between *CH0* and *CH1* for every few cycles

• *USB* port debugging.

### 3.5.3 Flow chart for the code.

Based on the above requirements, firmware for the two-photodiode wireless sensor is modified to increase the scope for adding multiple sensors. The process flow for the firmware is explained stepwise below in Table 3.6.
Table 3.6

Process Flow for Firmware

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1:</td>
<td>Declare all SFR’s Global Constant, variables and functions</td>
</tr>
<tr>
<td>Step 2:</td>
<td>Initialize device Special Function Registers</td>
</tr>
<tr>
<td>Step 3:</td>
<td>Configure all the I/O ports</td>
</tr>
<tr>
<td>Step 4:</td>
<td>Loop: Update the device settings according to the values specified by user</td>
</tr>
<tr>
<td>Step 5:</td>
<td>Control is taken by ISR. Inside the ISR for every 5000 milliseconds ch0 and ch1 of PGA112 are switched</td>
</tr>
<tr>
<td>Step 6:</td>
<td>Sensed data is loaded into buffer to send to UART. Again the process is looped from the previous step.</td>
</tr>
<tr>
<td>Step 7:</td>
<td>Once user clicks stop button on Matlab GUI, control is looped out from ISR and repeats from step 4.</td>
</tr>
</tbody>
</table>

3.6 Matlab GUI

Data from the fNIRS device is displayed on a Graphical User Interface (GUI), a Matlab functionality that allows development of customized controls with push buttons
and data boxes with specified names, which provides easy understanding for users. This GUI works on all Matlab versions starting from the 2009 version. The GUI was modified to work with a 7 inch portable windows 8.1 based tablet using Matlab 2013a, allowing researchers to avoid the requirement of interfacing with a laptop or desktop computer thereby providing increased mobility in taking data from the fNIRS sensors.

3.6.1 Communication between MATLAB and sensor. One of the crucial logic aspects in developing this software is bringing both the device and GUI application to a common channel. Using Bluetooth, wireless technology communication between these two devices is achieved. Matlab’s GUI is designed such that it can access the Bluetooth stack of the tablet using COM port numbers assigned by the tablet device manager. This number changes from device to device so flexibility in entering the COM port number in the GUI is the gateway to connect to the tablet’s Bluetooth.

3.6.2 Graphical controls. MATLAB was selected as the programming environment to develop the exerciser because it is easy to use, extensible, and popular with researchers. This Matlab GUI plots the last 200 samples recorded (730 nm, 850 nm, and ambient), and exports sample data to the MATLAB workspace for analysis. It communicates with the sensor via an Instrument Control Toolbox serial object associated with the virtual RS-233 serial port created by the Microsoft Bluetooth stack on the PC. The exerciser shown in Figure 3.10 shows the usage of Matlab GUI controls to interface with the fNIRS device.
Computers or laptops with no Bluetooth functionalities can use externally connectable Bluetooth radio to USB port as shown in Figure 3.11.
In order to instantiate a virtual serial port, these external Bluetooth radio’s must be paired with LMX9838SB. Once they are paired Microsoft Bluetooth stack assigns a serial COM port. This COM port number is used at the Matlab GUI COM# field to establish a link between the exerciser and the paired NIRS sensor [10]
Chapter 4

Results

A wireless fNIRS device with multi photodiode detector is programmed and tested. To create more mobility and portability, a Windows 8.1 O.S based tablet is used to run the Matlab and plot the sensors data. This is much smaller and easier to operate. Its specifications are as follows:

- Operating system: Windows 8.1 with Bing 32
- Processor: Intel® Atom® Z3735G, 1.8 GHz, 2 MB cache
- Display: 7” diagonal HD (1280 x 800 resolution)
- Memory: 1 GB DDR3L SDRAM
- Internal storage: 32 GB eMMC
- Wireless: 802.11b/g/n
- Battery: 3000 mAh Li-ion polymer
- Battery life: Up to 8 hours
- Ports: 1 micro-B USB 2.0; 1 audio jack (3.5 mm)
- Expansion slots: 1 microSD
- Sensors: Accelerometer
- Dimensions: 4.36 x 0.39 x 7.59 in
- Weight: 0.8 lb

Tablets are more user friendly than a laptop so this Windows tablet has become our choice to run the Matlab 2013a. The Intel atom processor has 4 cores which makes the execution speed faster. Figure 4.1 shows the HP Windows based 8.1 tablet [39].
4.1 Programming the Wireless fNIRS Device

The fNIRS device was powered-up and programmed with the firmware using the set-up shown in Figure 4.2. The USB emulator / debugger, from the device is connected to the USB of the computer which provides power (+5V), ground, C2 Data (C2D), and C2clock/Reset (C2CK/RST) lines. Figure 4.3 shows the prototype circuit used to program / debug the C8051F342.
The perfboard shown in the Figure 4.3 also has a jumper between C2CK/RST and ground which can be used to manually reset the connected fNIRS sensor. Even though the debugger can provide all of the power needed to operate the wireless sensor, the LTC3558 Linear USB Battery Charger with Buck and Buck-Boost Regulator requires that a battery be connected before the charger [40].

![Image](image.png)

*Figure 4.3 Flash USB type B male socket connections*

### 4.2 Software Testing

The firmware for this device was written in embedded C language. Silicon laboratories’ debugger/emulator was used to compile the code. To program the fNIRS device a micro USB type B connector was constructed on a perfboard connecting a 10 pin female plug for interface with the USB debugger. Initially, the 4th pin of micro USB was not connected which made the device unrecognizable by the debugger.

After reviewing the documentation for the micro USB, it was clear that the 4th pin acts as sensor/detector to plug in the device. This prevented the Silicon Labs debugger software from acknowledging the device connection. After correcting this issue Silicon
Labs software was able to detect the device. Figure 4.4 shows the screen shot of the Silicon Labs window when device is ready to connect.

![Screenshot of Silicon labs emulator while debugging](image)

**Figure 4.4** Screenshot of Silicon labs emulator while debugging

The fNIRS device firmware was compiled. Errors related to init.c source file occurred because of not declaring the include command for the init.c file. Soon after declaring init.c source file in the code, firmware was successfully compiled and downloaded onto the device.

### 4.3 Running the fNIRS Sensor

The procedure to operate the wireless fNIRS device is described step by step below:

- Power the device using 3.7 volt battery. The red indicator light is illuminated upon powering the device also indicating that the Bluetooth radio is available to the surrounding wireless devices, shown in Figure 4.5.
Go to ‘Add a Device’ in the computer control Panel and look for device named ‘Serial Port Device” as shown in Figure 4.6, then click on ‘Pair’ which will ask for pairing code. The pairing code for this Bluetooth module by default is ‘0000’. Successful pairing will allow the tablet/laptop to install the Bluetooth’s drivers.

Figure 4.5 Powered wireless fNIRS device indicating red LED

Figure 4.6 Screen shot of adding wireless fNIRS device to system
• The system will assign a \textit{COM} port to the device which can be checked by looking at the properties of the serial port device once it is added to the system. This com number is important in establishing connection with the device. The Serial Port Device Properties screen is shown in Figure 4.7.

![Serial Port Device Properties](image)

\textit{Figure 4.7} Screen shot of serial port device \textit{COM} port

• Open nirsgui.m file in MATLAB to enter the com port number and click connect. This will turn the device \textit{LED} to green indicating that the device is ready to take readings. Figure 4.8 shows the screen shot of Matlab \textit{GUI} to connect the device.
Figure 4.8 Screen shot of Matlab GUI to connect the wireless fNIRS device

- Enter the desired parameter values and click the write button to program the parameter values into the fNIRS device and click the start button to get the sensor reading plots output to the GUI window. The GUI window with data is shown in Figure 4.9.

- By clicking the clear button the GUI plot window can be cleared. Pressing the disconnect button terminates the link between device and system, with the indication of the red LED.
4.4 Discussion

Figure 4.9 show the output from the fNIRS sensor with the new firmware installed on the existing hardware which shows only one photodiode connected to PGA112. The second channel on the PGA112 is not connected to a photodiode. So, sensor data is plotted for the first photodiode and during second photodiode period data is not sensed.

Thus, it is demonstrated that the hardware device is switching between photodiode channels. By modifying the sensor circuit and adding an additional photodiode at the second channel, data from two photodiodes can be displayed by the Matlab GUI. Additional modifications and the use of the PGA116 would allow up to 10 photodiode channels to be connected.
Chapter 5

Conclusion and Recommendations for Future Work

This project has successfully completed the proof of concept for a multi-detector version of the original wireless, wearable fNIRS device. By taking advantage of the 2 available amplification channels of the PGA112, the author has shown that it is possible for the sensor to consecutively access multiple detectors responding to a single NIR LED stimulus source. By redesigning the circuit to use the PGA116, amplification circuit with 10 channels, the fNIRS device could flexibly accommodate from 1 to a maximum of 10 detectors.

The wireless fNIRS device developed is easily configurable. It can aid researchers and clinicians in their investigation of brain functionality with its low cost and low complexity. Extension of this sensor to multi-photodiode detectors will allow analysis of the data more thoroughly and accurately. Auto cycling the photodiodes’ output at regular intervals yields a visual output of each sensor reading which allows concentration on points of interest.

Moreover, the combination of Bluetooth technology and a portable tablet device for data recording allows device operations with minimal operator interaction. It also allows the data acquisition process to proceed unperturbed while the clinical subject performs tasks such as driving, walking and doing exercises.

Possible Applications:

- Monitoring cognitive workload in air traffic controllers
- Measuring brain dynamics during extended working memory
- Monitoring the development of expertise in piloting the unmanned vehicles
• Prediction of chemical composition of organic matter

5.1 Accomplishments

• Learned about fNIRS technology, its applications and devices
• Designed and implemented a multiple photodiode wireless fNIRS device
• Upgraded the working operations of this device by implementing MATLAB in HP 7 inch windows tablet
• Demonstrated the principles of operation of the multiple photodiode / multiple detector fNIRS device

5.2 Future Scope

fNIRS is an addition to the existing neuroscientific methods available to assess neural mechanisms of the brain. Future applications/improvements include:

• Conducting functional activation studies of different neurophysical disorders, developmental syndromes, attention deficiency hyperactivity
• Expanding the types of tests (tasks) performed to analyze hemodynamic activity
• Extending the region of concentration to entire brain not just the prefrontal cortex
• Making the device more accurate by nullifying the interference signals using digital signal processors and filters
• Wireless fNIRS has the scope to solve the problem of scanning while seizures are in progress which is a limiting feature in fMRI;
- Can be used in BCI (Brain Computer Interface) in restoration of movement disorders and paralysis. Combination of wireless fNIRS and BCI might provide a significant step towards neurorehabilitation
References


[40] Linear Technology, “LTC3558 - Linear USB Battery Charger with Buck and Buck-Boost Regulators,” 2008.
### Appendix A

#### List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CS</td>
<td>Chip Select</td>
</tr>
<tr>
<td>CTSN</td>
<td>Clear To Send</td>
</tr>
<tr>
<td>DIO</td>
<td>Data Input and Output</td>
</tr>
<tr>
<td>EEG</td>
<td>Electroencephalography</td>
</tr>
<tr>
<td>fMRI</td>
<td>functional Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>fNIRS</td>
<td>functional Near Infrared Spectroscopy</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input Output</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>ISR</td>
<td>Interrupt Service Routine</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Matrix Laboratory</td>
</tr>
<tr>
<td>MBLLL</td>
<td>Modified Beer Lambert Law</td>
</tr>
<tr>
<td>MSOP</td>
<td>Mini Small Outline Package</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PET</td>
<td>Positron Emission Tomography</td>
</tr>
<tr>
<td>PGA</td>
<td>Programmable Gain Amplifier</td>
</tr>
<tr>
<td>RTSN</td>
<td>Request To Send</td>
</tr>
<tr>
<td>RXD</td>
<td>Receive</td>
</tr>
<tr>
<td><strong>SCLK</strong></td>
<td>Serial Clock</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------</td>
</tr>
<tr>
<td><strong>TSSOP</strong></td>
<td>Thin Shrink Small Outline Package</td>
</tr>
<tr>
<td><strong>TXD</strong></td>
<td>Transmit</td>
</tr>
<tr>
<td><strong>UART</strong></td>
<td>Universal Asynchronous Receiver and Transmitter</td>
</tr>
<tr>
<td><strong>USB</strong></td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td><strong>VFT</strong></td>
<td>Verbal Fluency Task</td>
</tr>
</tbody>
</table>
Appendix B

Firmware Code

// NIRS-0.c
// David Rauth
// Wireless f/NIRS Sensor
// Bio Sensors Lab at Rowan University
// This program creates a UART loop-back through the LMX9838SB Bluetooth module.
// Version 0.1: 13-Mar-11
// -Program created
// Sivaram Karra
// Modified the NIRS-0.c for Multi-Photodiode detectors
// APRIL 14, 2015
// INCLUDES

#include <c8051f340.h>    // SFR declarations
#include <stdio.h>
#include <stdlib.h>
#include <init.c>

#define AVERAGE 1             // Number of samples to average (10)
#define IDLECYCLES 500        // ISRs between updates (100)
#define SETDELAY 1             // ISRs between state switch (2)
#define IADJ2LIMIT 50         // IADJ2 duty cycle limit

#define M_IDLE 0               // Waiting for Bluetooth link
#define M_SINGLE 1             // Single-shot sampling mode
#define M_BAT 2                // Measure battery
#define M_AUTO 3               // Continuous sampling mode

// State machine DEFINITIONS

#define S_SETAMB 0             // Setup for ambient light
#define S_READAMB 1             // Sample ambient light
```c
#define S_SETLED730  2                   // Setup for 730 nm LED
#define S_READLED730  3                   // Sample 730 nm LED
#define S_SETLED805  4                   // Setup for 805 nm LED
#define S_READLED805  5                   // Sample 805 nm LED
#define S_SETLED850  6                   // Setup for 850 nm LED
#define S_READLED850  7                   // Sample 850 nm LED
#define S_IDLE        8                   // Idle
// SFR and I/O DEFINITIONS
#define VOUT       0x01                // AMX0P P1.1
#define VBAT       0x02                // AMX0P P1.2
#define VTEMP      0x1E                // AMX0P Temp sensor
sfr16 ADC0       = 0xBD;               // ADC0 result
sbit SCLK = P0^0;               // SPI clock (P0.0)
sbit MISO = P0^1;               // SPI MISO (P0.1)
sbit MOSI = P0^2;               // SPI MOSI (P0.2)
sbit HPWR = P0^3;               // Battery charge USB current (P1.5)
sbit TX = P0^4;               // UART TX (P0.4)
sbit RX = P0^5;               // UART RX (P0.5)
sbit CS = P0^5;               // SPI chip select (active low) (P0.6)
sbit VILED = P1^0;               // PWM output for LED current (P1.0)
sbit CHRG = P1^3;               // Charge indicator (active low) (P1.3)
sbit LINK = P1^4;               // Bluetooth GPIO PG6 - Link (P1.4)
sbit MODE = P1^5;               // Regulator mode
sbit SUSP = P1^6;               // Suspend battery charging (P1.6)
sbit VBIAS = P1^7;               // PWM output for sensor offset (P1.7)
sbit SHDNN = P2^0;               // LED shutdown (active low) (P2.0)
sbit LED730 = P2^1;               // NIR 730 nm LED enable (P2.1)
sbit LEDGRN = P2^2;               // Status LED GREEN (P2.2)
sbit LEDRED = P2^3;               // Status LED RED (P2.3)
sbit LED805 = P2^4;               // NIR 805 nm LED enable (P2.4)
sbit LED850 = P2^5;               // NIR 850 nm LED enable (P2.5)
sbit RTS = P2^6;               // UART RTS (1 = RX not ready) (P2.6)
sbit CTS = P2^7;               // UART CTS (1 = TX not ready) (P2.7)
```

```c
// Function PROTOTYPES
extern void Init_Device (void);
void Config_Device (void);
void itoa3(int n, char s[]);
```

```c
// Global VARIABLES
static unsigned int LED730Duty = 0;       // 730 nm LED duty cycle (0 %)
```
static unsigned int LED805Duty = 0;  // 805 nm LED duty cycle (0 %)
static unsigned int LED850Duty = 0;  // 850 nm LED duty cycle (0 %)
static unsigned int gain = 0;        // PGA gain (unity)
static unsigned int bias = 0;        // Photodiode bias
static unsigned int samples = AVERAGE;  // Samples per point
static unsigned int idleCycles = IDLECYCLES;  // ISRs between updates
static unsigned int setDelay = SETDELAY;  // State switch delay
static unsigned int mode = M_SINGLE;   // Current mode
static unsigned int state = S_SETAMB;  // Current state

// appended for multiple Diodes
int ch0=0;
int ch1=0;

int cycles;

// MAIN Routine

void main (void)
{
    unsigned char addr;
    unsigned char reg[3];
    unsigned char n;
    // Initialization
    Init_Device();       // SFR initialization
    // Configuration
    Config_Device();     // Configure I/O

    // Loop to update device settings according to user editings from Matlab GUI
    while (1)
    {
        RTS = 0;
        addr = getkey();      // gets the key value
        switch (addr)       // Transfers the control to case according to key value
        {
        case 'A':
            for (n = 0; n < 3; n++)
                reg[n] = getkey();
            if (reg[0] == '?')
            {
                // while(CTS);
                itoa3(samples, reg);
                putchar(addr);
                for (n = 0; n < 3; n++)
                {
                    // while(CTS);
                    putchar(reg[n]);
                }
while(CTS);
    putchar(\n');
}
else
{
    if (atoi(reg) >= 1 && atoi(reg) <= 100)
        samples = atoi(reg);
}
break;

case 'B':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(setDelay, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar(\n');
    }
    else
    {
        if (atoi(reg) >= 1 && atoi(reg) <= 10)
            setDelay = atoi(reg);
    }
break;

case 'C':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(idleCycles, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar(\n');
    }
else
{
    if (atoi(reg) >= 10 && atoi(reg) <= 999)
        idleCycles = atoi(reg);
}
break;

case 'D':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(LED730Duty, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar('n');
    }
    else
    {
        if (atoi(reg) >= 0 && atoi(reg) <= IADJ2LIMIT)
            LED730Duty = atoi(reg);
    }
break;

case 'E':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(LED805Duty, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
```c
    putchar(\n');
}
else
{
    if (atoi(reg) >= 0 && atoi(reg) <= IADJ2LIMIT)
        LED805Duty = atoi(reg);
}
break;
*/

    case 'F':
        for (n = 0; n < 3; n++)
            reg[n] = getkey();
        if (reg[0] == '?')
        {
            // while(CTS);
            itoa3(LED850Duty, reg);
            putchar(addr);
            for (n = 0; n < 3; n++)
                // while(CTS);
                putchar(reg[n]);
        }
        // while(CTS);
        putchar(\n');
    else
    {
        if (atoi(reg) >= 0 && atoi(reg) <= IADJ2LIMIT)
            LED850Duty = atoi(reg);
    }
break;

    case 'G':
        for (n = 0; n < 3; n++)
            reg[n] = getkey();
        if (reg[0] == '?')
        {
            // while(CTS);
            itoa3(gain, reg);
            putchar(addr);
            for (n = 0; n < 3; n++)
                // while(CTS);
                putchar(reg[n]);
        }
```

59
// while(CTS);
putchar(\n);
}
else
{
    if (atoi(reg) >= 0 && atoi(reg) <= 7)
    {
        gain = atoi(reg);
        SPIF = 0;
        while (!NSSMD0);
        NSSMD0 = 0;
        SPI0DAT = 0x2A;  // MSB (command)
        while (!TXBMT);
        while (!SPIF);
        SPIF = 0;
        SPI0DAT = (atoi(reg) << 4) + 0x00;  // LSB (gain + channel)
        while (!TXBMT);
        while (!SPIF);
        SPIF = 0;
        NSSMD0 = 1;
    }
}
break;

case 'H':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(bias, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar(\n);
    }
    else
    {
        if (atoi(reg) >= 0 && atoi(reg) <= 255)
        {
            bias = atoi(reg);
            PCA0CPH1 = 255 - bias;
if (PCA0CPH1 == 255)
    PCA0CPM1 &= 0xBF;
else
    PCA0CPM1 |= 0x40;
}
}
break;

case 'J':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(mode, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar('n');
    }
else
    {
        if (atoi(reg) >= 0 && atoi(reg) <= 3)
            mode = atoi(reg);
    }
    break;

case 'L':
    for (n = 0; n < 3; n++)
        reg[n] = getkey();
    if (reg[0] == '?')
    {
        // while(CTS);
        itoa3(!CHRG, reg);
        putchar(addr);
        for (n = 0; n < 3; n++)
        {
            // while(CTS);
            putchar(reg[n]);
        }
        // while(CTS);
        putchar('n');
}  
        } else 
        {  
        if (atoi(reg) >= 0 & & atoi(reg) <= 1)  
            SUSP = atoi(reg);  
        }  
        break;  
    }  
    RTS = 1;  
}  
}  

void Timer2_Interrupt (void) interrupt 5  
{  
    static unsigned int stateISRcount = AVERAGE; // state ISR counter  
    static unsigned int idleISRcount = 0; // idle ISR counter  
    static unsigned long accumulator = 0;  
    static unsigned long datapoint = 0;  
    cycles = cycles++;  
    if(cycles==5000) //after 5000 milli seconds, channel is switched  
    {  
        if(ch0==0) //if PGA112’s channel 0 is not on then to ON  
        {  
            SPIF = 0;  
            while (!NSSMD0);  
            NSSMD0 = 0;  
            SPI0DAT = 0x2A; // MSB (command)  
            while (!TXBMT);  
            while (!SPIF);  
            SPIF = 0;  
            SPI0DAT = gain + 0x00; // LSB (gain + channel)  
            while (!TXBMT);  
            while (!SPIF);  
            SPIF = 0;  
            NSSMD0 = 1;  
        }  
        // In case of PGA116 we have to use 10 channels instead of 2 by putting this loop in a  
        // function and calling it.  
        ch0=1;  
        ch1=0;  
        }  
        else  

SPIF = 0;
while (!NSSMD0);
NSSMD0 = 0;
SPI0DAT = 0x2A;                      // MSB (command)
while (!TXBMT);
while (!SPIF);
SPIF = 0;
SPI0DAT = gain + 0x01;       // LSB (gain + channel)
   while (!TXBMT);
   while (!SPIF);
SPIF = 0;
NSSMD0 = 1;
//void channel_1(void);
ch1=1;
ch0=0;
}
cycles=0;
//Re-setting count variable after every 5 sec or 5000 milli-seconds

LEDGRN = LINK;
LEDRED = (~LINK | ~CHRG);

if (mode != M_IDLE)
{
   if (idleISRcount == 0)
   {
      switch (state)
      {
      case S_SETAMB:
         idleISRcount = setDelay;
         state = S_READAMB;
         if (mode == M_BAT)
            AMX0P = VBAT;
         break;

      case S_READAMB:

         AD0BUSY = 1;
         while (!AD0INT);
         AD0INT = 0;
         accumulator += ADC0;
stateISRcount--;  
if (stateISRcount == 0)  
{
    datapoint = accumulator / samples;  
    accumulator = 0;  
    stateISRcount = samples;  
    if (mode == M_BAT)  
    {
        AMX0P = VOUT;  
        printf("K%ld\n", datapoint);  
        state = S_SETAMB;  
        mode = M_IDLE;  
    }  
    else  
    {
        printf("I%ld ", datapoint);  
        state = S_SETLED730;  
    }  
}  
break;

case S_SETLED730:  
    LED730 = 1;  
    PCA0CPH0 = 255 - LED730Duty;  
    PCA0CPM0 |= 0x40;  
    idleISRcount = setDelay;  
    state = S_READLED730;  
    break;

case S_READLED730:
    AD0BUSY = 1;  
    while (!AD0INT);  
    AD0INT = 0;  
    accumulator += ADC0;  
    stateISRcount--;  
    if (stateISRcount == 0)  
    {
        datapoint = accumulator / samples;  
        printf("%ld ", datapoint);  
        accumulator = 0;  
        stateISRcount = samples;  
        PCA0CPM0 &= 0xBF;  
        LED730 = 0;  
        //state = S_SETLED805; // 805 nm LED not installed
state = S_SETLED850;
}
break;

case S_SETLED805:
    LED805 = 1;
    PCA0CPH0 = 255 - LED805Duty;
    PCA0CPM0 |= 0x40;
    idleISRcount = setDelay;
    state = S_READLED805;
    break;

case S_READLED805:
    AD0BUSY = 1;
    while (!AD0INT);
    AD0INT = 0;
    accumulator += ADC0;
    stateISRcount--;
    if (stateISRcount == 0)
    {
        datapoint = accumulator / samples;
        printf("%ld ", datapoint);
        accumulator = 0;
        stateISRcount = samples;
        PCA0CPM0 &= 0xBF;
        LED805 = 0;
        state = S_SETLED850;
    }
    break;

case S_SETLED850:
    LED850 = 1;
    PCA0CPH0 = 255 - LED850Duty;
    PCA0CPM0 |= 0x40;
    idleISRcount = setDelay;
    state = S_READLED850;
    break;

case S_READLED850:
    AD0BUSY = 1;
    while (!AD0INT);
    AD0INT = 0;
    accumulator += ADC0;
    stateISRcount--;
    if (stateISRcount == 0)
    {

datapoint = accumulator / samples;
printf("%ld \n", datapoint);
accumulator = 0;
stateISRcount = samples;
PCA0CPM0 &= 0xBF;
LED850 = 0;
idleISRcount = idleCycles;
state = S_SETAMB;

    if (mode == M_SINGLE)
        mode = M_IDLE;
    }
break;
}
}
else
    idleISRcount--;
}

TF2H = 0;                                // Reset Interrupt
}

void Config_Device()
{
    // Set outputs to default
    LEDGRN = 0;                     // Green LED off
    LEDRED = 1;                     // Red LED on
    HPWR = 0;                       // Battery charge current set to 100mA
    MODE = 0;                       // Regulator set to PWM mode
    SUSP = 0;                       // Allow battery charging
    SHDNN = 1;                       // Enable NIR LED driver and sensor
    LED730 = 0;                     // NIR 730 nm LED off
    LED805 = 0;                     // NIR 805 nm LED off
    LED850 = 0;                     // NIR 850 nm LED off
    RTS = 0;                         // Not ready to send

    // Set PGA (CH1, gain = 1)
    SPIF = 0;
    while (!NSSMD0);
    NSSMD0 = 0;
    SPI0DAT = 0x2A;                  // MSB (command)
    while (!TXBMT);
    while (!SPIF);
    SPIF = 0;
    SPI0DAT = (0x00 << 4) + 0x00;    // LSB (gain + channel)
while (!TXBMT);
while (!SPIF);
SPIF = 0;
NSSMDO = 1;

// Set PCA outputs to 0% duty cycle (ECOM0, ECOM1 = 0)
PCA0CPM0 &= 0xBF; // Set VILED to 0
PCA0CPM1 &= 0xBF; // Set VBIAS to 0

TR1 = 1;       // Start Timer1
TR2 = 1;       // Start Timer2
TI0 = 1;       // Indicate TX0 ready

}  

void itoa3(int n, char s[])
{
    // Assumes the length of s is 3
    int i, j;
    char c;
    s[0] = '0';
    s[1] = '0';
    s[2] = '0';
    
i = 0;
do
    { /* generate digits in reverse order */
        s[i++] = n % 10 + '0'; /* get next digit */
    } while ((n /= 10) > 0); /* delete it */
s[i] = '0';

    for (i = 0, j = 3-1; i<j; i++, j--)
    {
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}