**Work Smarter, Not Harder**

The University of Notre Dame Department of Electrical Engineering

Senior Design Final Report

Brian Celeste

Claire Gaffney

Thomas Kalil

Matthew Lin

Table of Contents

1. **Introduction**

1.1 Problem………………………………………………………………………………4

1.2 Solution……………………………………………………………………………....5

1.3 Expectations vs. Reality……………………………………………………………6

1. **Detailed System Requirements**

2.1 Overall System Block Diagram……………………………………………………7

2.2 Subsystem Block Diagram…………………………………………………………7

2.3 Hardware Component Requirements Descriptions……………………………..8

1. **Detailed Project Description**

3.1 System Theory of Operation…………………………………………………..…11

3.2 System Block Diagram………………………………………………………….12

3.3 Detailed Design/Operation of LEDs and Photodiode Hardware……………13

3.4 Detailed Design/Operation of MATLAB Data Processing…………………….17

3.5 Detailed Design/Operation of RSL10 Communication with AFE4490……….24

3.6 Detailed Design/Operation of RSL10 Communication with Accelerometer...33

3.7 Detailed Design/Operation of RSL10 Communication with Flash Storage... 33

1. **System Integration Testing**…………………………………………………………45
2. **User’s Manual/Installation manual**

5.1 How to install and setup the product…………………………………………….47

5.2 How to use the product…………………………………………………………...47

5.3 How the user can tell if the product is working……………………………...…48

5.4 How the user can troubleshoot the product………………………………….…48

1. **To-Market Design Changes**……………………………………………………...…48
2. **Conclusion**………………………………………………………………………….…49
3. **Appendices** (Attached/Found on Website).........................................................50
   1. Complete Hardware Schematics
   2. Complete Software Listings
   3. Parts List and Links to Component Datasheets, Research Links

**1 Introduction**

1.1 Problem

While several commercial wearable devices can measure heart rate, there is nothing on the market to monitor other aspects of an athlete’s health during a workout. Recently, there has been a trend of data driven fitness, through apps and wearables. Photoplethysmography, which measures changes in blood volume by the absorption and reflection of red and infrared light in hemoglobin, could prove useful for a high performance athlete, for example. Exposure to important data, such as blood oxygenation and water content, could assist an athlete and help maximize their workout and measure results.

This project expanded upon a previous project, which used photoplethysmography to monitor heart rate and blood oxygenation in order to detect the symptoms of cardiomyopathy. Cardiomyopathy, also known as enlarged heart syndrome, is a condition caused by the thickening of the heart walls, which constricts blood flow and can ultimately lead to unexpected cardiac arrest. There were a few problems associated with the previous design, including the inability to determine if movement of the board is negatively affecting recorded data, the inability to measure more than two wavelengths at the same time, the necessity of physical on-board memory, and the fact that the board is too large to be easily and conveniently worn during exercise.

1.2 Solution

To address the market need and the problems surrounding the previous design, the new design retained the functionalities of last year’s project and expanded them to incorporate the new applications.

To account for possible noise generated by movements of either the device or the body (both highly likely during exercise), an accelerometer was added. If the noise from this movement is significant, the accelerometer data is used to perform the necessary error-correction during data processing to attain more accurate final results.

In order to test for a wider variety of molecules, an additional wavelength for detecting water content in tissue was added to the design. Because of the three wavelength measurements (oxygenated hemoglobin, deoxygenated hemoglobin, and water content), an additional AFE4490 chip was required to account for the added LED. Further, a green LED was added because it retains fewer motion artifacts, and is thus a more accurate measurement.

To improve data collection and post-processing, a flash storage device was added to the new design so that an athlete could record data during a workout for an extended period of time and then transmit the data via Bluetooth to the computer for post-processing.

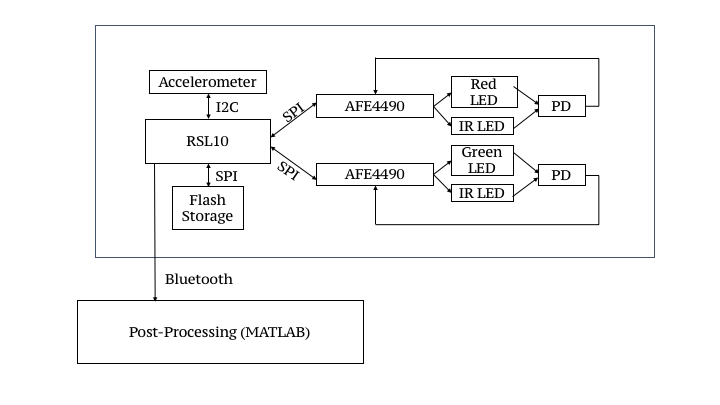
Wearability of the device was improved by making the design smaller and more compact. In addition, a two-sided board and careful design consideration made the board as user friendly as possible. The LEDs, on the bottom of the board, can sit comfortably on the wrist for data collection, the buttons for starting data collection and Bluetooth transmission are on the top of the board, so the athlete can press them when he or she is ready.

1.3 Expectations vs. Reality

The project was successful in adding all the desired new features to the board while reducing board size. While hardware issues prevented the RSL10 from being properly coded, the board serves as a proof of concept that a small device can contain all the necessary functionality. For post-processing, time constraints prevented the inclusion of an option to stream data to MATLAB immediately as it is collected, as well as the development of a smartphone app that would also be capable of connecting to the board and displaying the collected data.

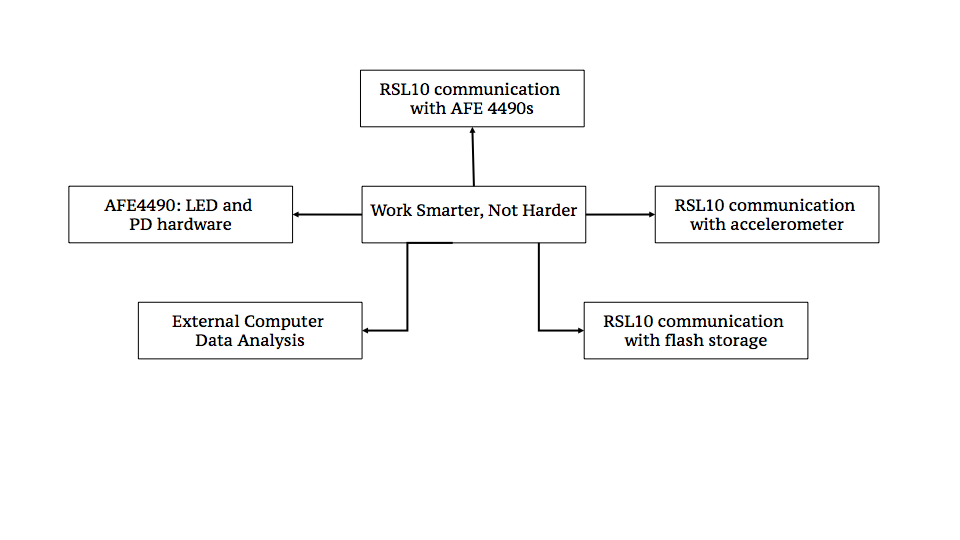
**2 Detailed System Requirements**

2.1 Overall System Block Diagram



**Figure 1**. Overall System Block Diagram

2.2 Subsystem Block Diagram



**Figure 2** . Subsystem Block Diagram

2.3 Hardware Component Requirements Descriptions

*Microcontroller*

We are using the RSL10 microcontroller as our main programmable component and the core of our board design. This microcontroller has been selected due to its onboard Bluetooth transmission capabilities to avoid the need for an external Bluetooth transmitter component in addition to a microcontroller (a PIC32 would have required this). It and its DIO pins can be supplied with a maximum of 3.63V, with the chip itself being recommended to have a supply of 1.25-3.3V. It can be configured to operate on a 48MHz external oscillator or a 32kHz external oscillator. This chip has 16 DIO pins, which can be used to configure SPI, I2C, PCM and UART interfaces through code. We allow the AFEs to share pins for SPI data in, data out, and clock, and have them use separate chip select pins (3+1+1=5 pins). The SPI memory shares the same clock, then uses its own pins for chip select, data in and data out (5+3=8 pins). The I2C accelerometer shares clock and uses its own data pin (8+1=9 pins). 7 other pins are utilized for additional interrupt access and button functionality, as well as the external programmer/debugger pins. The component has a 6mm x 6mm footprint.

*AFE4490*

The AFE4490 is an integrated analog front-end for pulse oximeters. It consists of a low-noise receiver channel with a 22-bit analog-to-digital converter, an LED transmit section, and diagnostics for LED and photodiode fault detection. Thus, the AFE4490 provides a suitable functionality for medical pulse oximetry applications. The AFE4490 communicates with an external microcontroller via SPI communication and is able to do so with the RSL10. The LED drive current is programmable to 50, 75, 100, 150, and 200 mA. The continuous input current to any pin is 7 mA. The device is small at 6.00 mm x 6.00 mm. The Rx analog supply pin requires a supply voltage of 2.0 V to 3.6 V. The transmit control supply pin (Tx) requires a supply voltage of 3.0 V to 5.25 V.

*LEDs/photodiode system*

The TEMD5080X01 is a silicon PIN photodiode suitable for visible and infrared radiation, which is necessary for the capturing of the reflected or transmitted light from the 4 diodes of various wavelengths. The package is small at 5.00 mm x 4.24 mm. Each AFE4490 can handle 2 LEDs, and these LEDs should be connected in antiparallel, as the default configuration for the AFE4490s is that the LEDs alternate between on and off due to voltage switching pins. The wavelengths used are 525 nm, 631 nm, 940 nm, and 970 nm, which respectively are suitable absorption wavelengths for the concentration calculations for deoxygenated hemoglobin (gives fewer motion artifacts), deoxygenated hemoglobin (more motion artifacts but better contrast between it and oxygenated hemoglobin), oxygenated hemoglobin, and water. The reflected or transmitted light is the signal used for the pulse oximetry analysis. In order to capture the signal to use as data, a photodiode captures the light and converts it to voltage. The 525 nm green LED (Vishay VLMTG1300) operates at a maximum of 20mA for a range of 2.8-3.6V forward voltage, and it has a small footprint of 2.3mm x .8mm. The 631 nm red LED (Vishay VLMS1300) operates at a maximum of 30mA but is typically operated at 20mA with a forward voltage of up to 2.4V. Its footprint size is the same as the VLMTG1300 at 2.3mm x .8mm. The 940 nm IR LED (Vishay VSMB1940X01) operates at a maximum of 100mA and a typical forward voltage of 1.35V, with a 2.6mm x 1.45mm footprint. The 970 nm IR LED (Marubeni SMT970) operates at a maximum forward current of 100mA and a typical forward voltage of 1.3V, with a 4.5 mm x 2.6 mm footprint.

*Power system*

Based on the voltage requirements of the subsystems above, the power system outputs a VDD of 3.3 V to the entire board. To do this, a Li-Ion battery of about 3.7-4 V is regulated by the TPS61201, a low input voltage synchronous boost converter which takes the battery as its input and outputs a constant 3.3 voltage.

*Accelerometer*

The accelerometer we’re using is the KX126-1063, which can communicate to the RSL10 through either SPI or I2C protocol. It has a small footprint of 2mm by 2mm. The I2C clock can be up to 3.4MHz and the SPI clock can be up to 10MHz. Due to the SPI configurations already being used up by the AFEs and memory, we opted to use the I2C protocol, which is also beneficial as it allows the accelerometer to use fewer RSL10 DIO pins. The accepted supply voltage (VDD) range is 1.71 - 3.6 V, with the IO pad voltage range being 1.71 - VDD. There are high resolution, low power, and standby modes, which respectively have current consumptions of 145uA, 10uA, and .9uA.

*Flash memory*

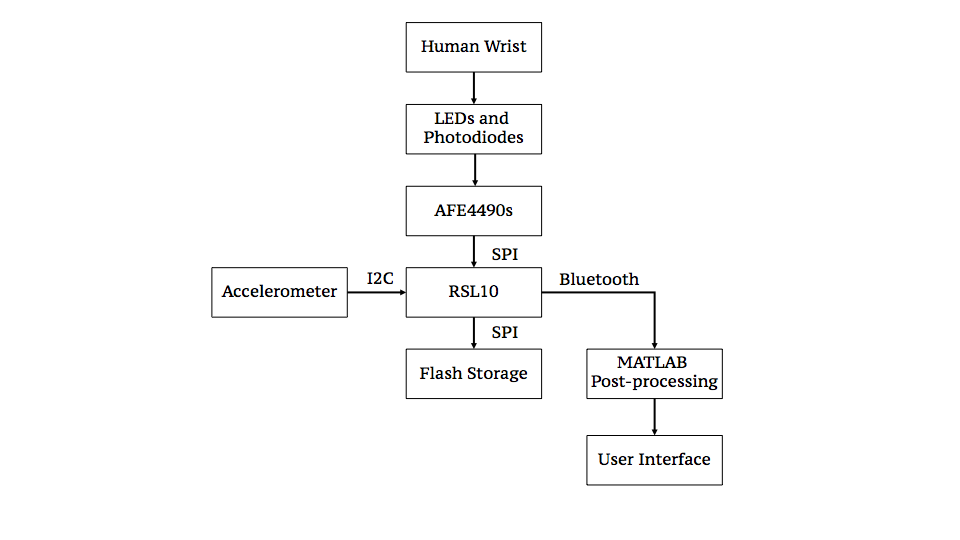
The flash memory chip we’re using is the Winbond W25N01GVZEIG TR, which can only communicate with the RSL10 through SPI protocol and which operates at 2.7-3.6V voltage supply. Current consumption is as low as 25mA active and 10uA for standby. Clock speeds of up to 104MHz are allowed. Its footprint is 8 mm x 6 mm, and it has a 128MB capacity. It allows over 100,000 erase/program cycles, and supports over 10 years of data retention. The continuous data transfer rate is 50MB/sec.

**3 Detailed Project Description**

3.1 System Theory of Operation

From a high level view, the system operates by the blood’s absorption and reflection of light. The RSL10 uses SPI communication to control the two AFE4490 digital-to-analog converters and I2C communication to control the accelerometer. After the RSL10 communicates with the AFE4490s, each controls the flashing rate of two LEDs, connected in antiparallel: one set is red (631 nm) and infrared (970 nm), the other green (525 nm) and infrared (940 nm). The LEDs light shines on the human body, and oxygenated hemoglobin absorbs the red and green light, water in tissue absorbs the 970 nm infrared light, and deoxygenated hemoglobin absorbs the 940 nm infrared light. Light not absorbed by the blood is reflected back to a photodiode, which converts the signal to an analog current and sends the value to the AFE4490 digital-to-analog converter. The AFE4490 converts the signal to digital and sends it to the RSL10 using SPI communication. The accelerometer detects when the acceleration exceeds the allowable value (that which would interfere with the data from the photodiode), which is then paired with the readings from the AFE4490s. The RSL10 then sends the data to flash storage using SPI communication. When prompted by the user (by the push of the Bluetooth button), the data from the flash storage is sent to a computer via Bluetooth. A MATLAB program analyzes the data to present the user with the molecular concentrations of oxygenated/deoxygenated hemoglobin and water, heart rate, tissue oxygenation, and tissue hydration.

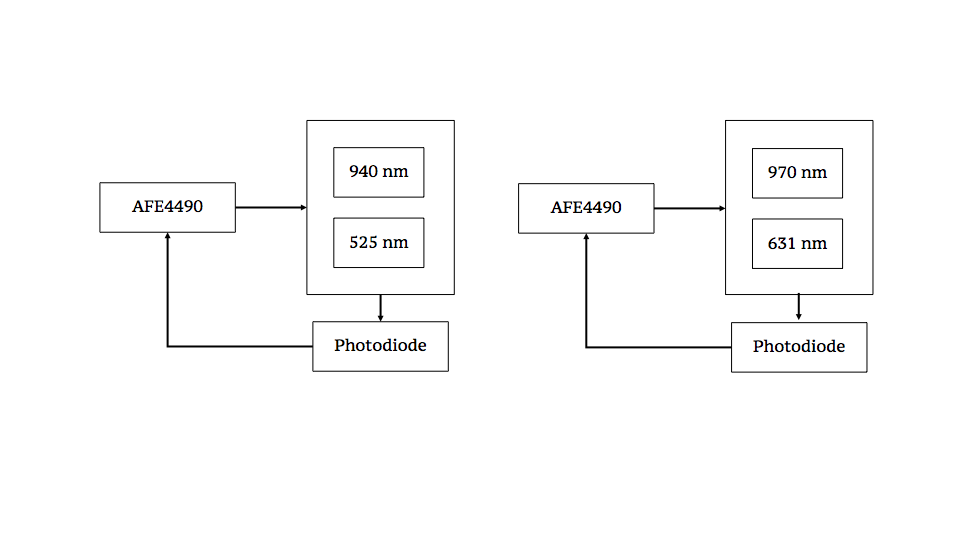
3.2 System Block Diagram



**Figure 3**. System Block Diagram

3.3 Detailed Design/Operation of LEDs and Photodiode Hardware

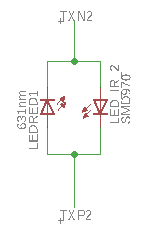
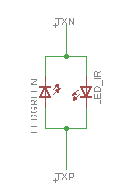
*System Flow Chart*



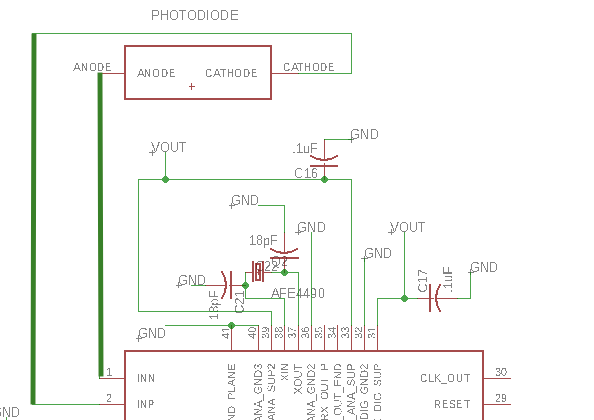
**Figure 4.** AFE4490, LEDs, and Photodiode Flow Chart

*Schematic*

The dual light emitting diodes, red and infrared (and green and infrared for the other AFE) are placed on a small board 7 millimeters apart. The TI AFE4490 digital-to-analog converter sends the flash rate to the LEDs. The LEDs are connected to the AFE4490 at pins 14 (TXN) and 15 (TXP). The 631 nm red LED and 970 nm infrared LED send light to the human body (the wrist), and the oxygenated blood and water in the tissue absorb the red and infrared wavelengths, respectively. The 525 nm green LED and 940 nm infrared LED also send light, and the oxygenated blood and deoxygenated blood absorb the green and infrared wavelengths, respectively. Light is reflected from the body and then captured by the photodiode. The photodiode captures the reflected light and converts it into current. This signal is then sent back to the AFE4490 digital-to-analog converter. The anode of each photodiode is connected to its corresponding AFE4490 at pin 1 (INN), and the cathode is connected at pin 2 (INP).



**Figure 5.**  LED Schematic



**Figure 6** Photodiode Schematic (connections highlighted by thicker green lines)

*Function and Connection to Other Subsystems*

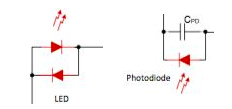
The external hardware consists of two LEDs and one photodiode for each AFE4490. The 970 nm LED (infrared) is connected in antiparallel with the 631 nm LED (red). For the second set of LEDs, the 940 nm LED (infrared) has the part number is connected in antiparallel with the 525 nm LED (green). The LEDs correspond to the wavelengths needed for deoxygenated blood, oxygenated blood, and water to absorb light. They are all surface mount LEDs, which allows for relatively easy soldering on the small board.

The photodiode exists separately from the light emitting diodes. The photodiode is placed 7 millimeters from the LEDs on the back of the board, capturing the reflected light. The photodiode part number is TEMD5080X01. This silicon PIN photodiode is sensitive from 350 to 1100 nm, so by using one photodiode for each AFE4490, both wavelengths can be captured. The photodiode is a surface mount part, making it easy to use solder paste on a small board.

Externally, each LED and photodiode system is connected directly to its corresponding AFE4490. The AFE4490 controls the flashing rates of the LEDs. The blood absorbs some light, and some light reflects back to the photodiode. The photodiode then captures the signal and sends a current to the AFE4490, which then converts the signal from analog to digital for the microcontroller to process. The analog LED drive, pin 14, controls the 940 nm infrared LED in the first AFE4490, and the 970 nm infrared LED in the second AFE4490. The analog LED drive pin 15, controls the 525 nm green LED in the first AFE4490, and the 631 nm red LED in the second AFE4490. Finally, the photodiode anode and cathode send the signal to the receiver input pins 1 and 2 respectively for each AFE4490.

*Subsystem testing*

An evaluation board mounted with the TI AFE4490 was used for familiarization and testing. This incorporated the dual red/infrared LED and photodiode for accurate collection of data. The dual red/infrared LED and photodiode are connected in an antiparallel orientation. The following diagrams from the AFE4490 datasheet show the dual LED and photodiode system:

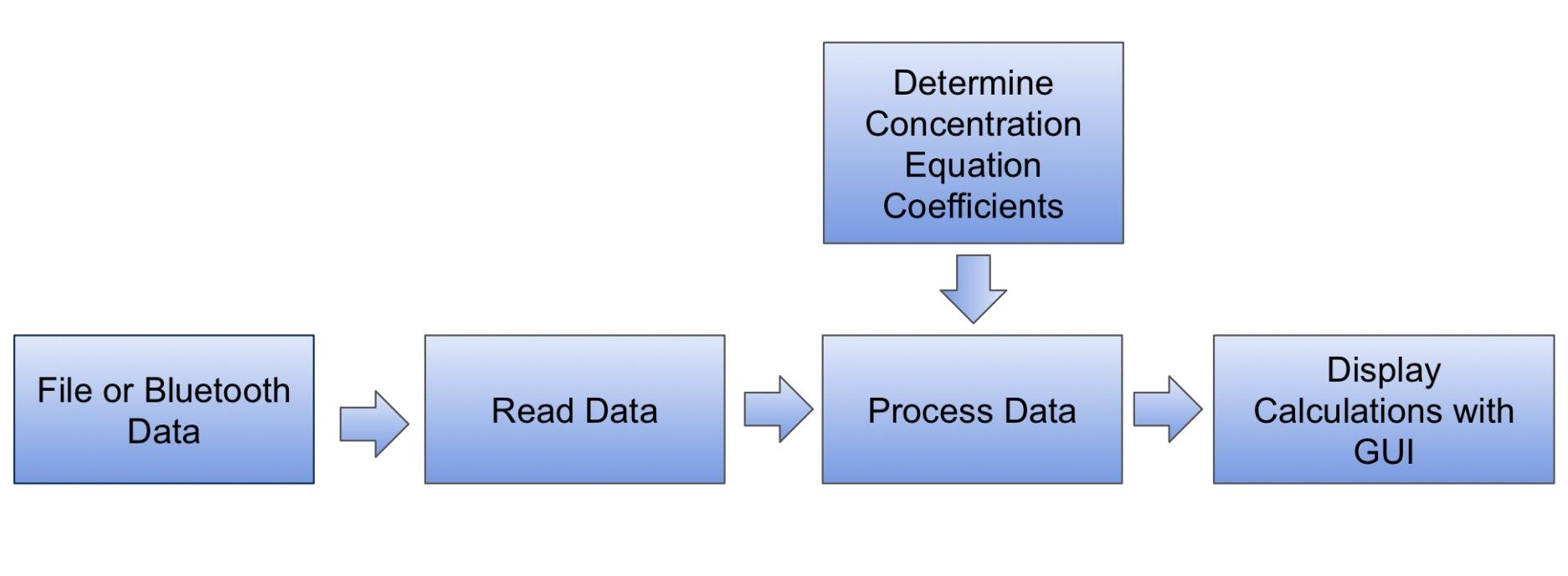


**Figure 7** . AFE4490 Datasheet LEDs and Photodiode

The evaluation board came with software enabling the manipulation of certain operating conditions and monitoring the data being collected in real time. The TI setup included an oximetry clip commonly found in hospitals for use on a patient’s finger. The AFE4490 controls the flashing rate of the LEDs. Light shines through the finger, and the finger absorbs some of the red and infrared light. Using reflection geometry, the photodiode, on the same side of the finger as the LEDs, captures the uninterrupted photons, which generate a current to be read as a voltage by the AFE4490.

The evaluation board provided a guideline for design of the actual device. The choice of green as well as red LEDs was made to accommodate movement (because green light does not carry as much motion artifacts), a probable scenario given this device’s application for high performance athletes. However, even though the value of the LEDs is different, the principle is the same.

3.4 Detailed Design/Operation of MATLAB Data Processing

**Figure 8**. Flow Chart of MATLAB Data Processing.

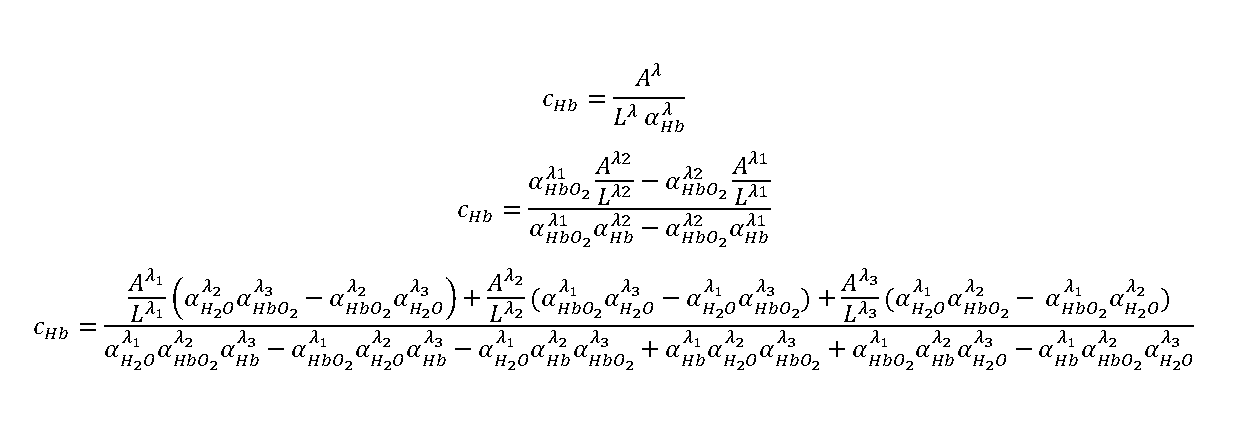
The processing and display of the collected data is a crucial aspect of our product. The main goal is to take the data collected from the photodiode and present the user with derived useful information, in particular the molecular concentrations of oxygenated/deoxygenated hemoglobin and water, heart rate, tissue oxygenation, and tissue hydration. This was achieved through the use of MATLAB, and includes several program components: the determination of concentration equation coefficients, the collection of photodiode data from a file or through Bluetooth, calculations based on input data (molecule concentrations, heart rate, oxygenation, and hydration), and the display of the calculations in a user-friendly GUI.

We chose to use MATLAB for our data processing for a couple reasons. First, the previous year’s project that we were continuing had used MATLAB for all of its data processing, which included features such as file reading and concentration calculations for two LEDs. This provided a solid foundation for the development of our own programs, as our data processing had similar goals to their data processing. Furthermore, we were already very comfortable using the MATLAB environment to create programs for mathematical computations and simulations. As such, it seemed to be an obvious choice for us to continue using MATLAB for the processing and display of the photodiode data.

Before any actual data processing can begin, a preliminary determination of the coefficients needed for the concentration equations is necessary. This is important as these coefficients depend on the LED wavelengths being used, the number of LEDs being used, and the specific molecules chosen for the concentration calculations. We wrote two programs, one for the calculation of coefficients when using two LEDs (for oxygenated and deoxygenated hemoglobin), and one for the calculation of coefficients when using three LEDs (for oxygenated hemoglobin, deoxygenated hemoglobin, and water), named calc2coeff.m and calc3coeff.m respectively. The programs function by taking the known molar extinction coefficients of each molecule at the the specified wavelengths and calculating the modified coefficients for each A/L wavelength-dependent term in the concentration equations for each molecule (see the concentration equations figure below). The calculation of these modified coefficients is significantly more complex for three wavelengths than for two, which is the main reason we decided to make a separate program for each case. Once these coefficients are determined by the respective program, they can be included in the actual concentration calculations.

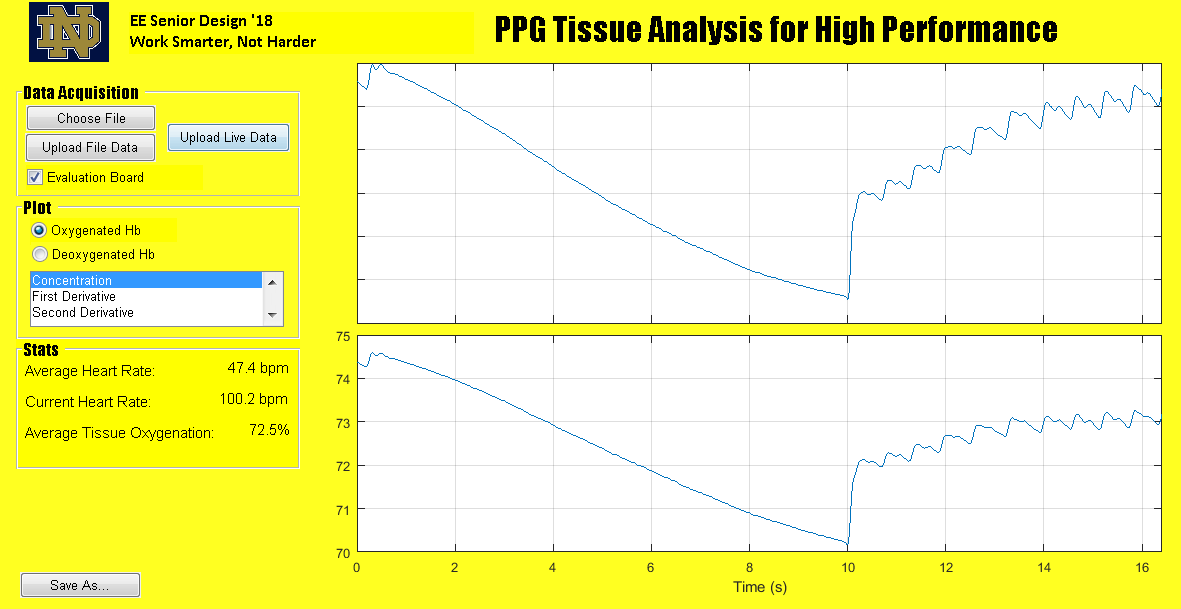
In order to calculate things such as the molecular concentrations, we of course require some sort of relevant input data. In our case, that would be the voltage across the photodiode, which is directly related to the intensity of light detected. The data is therefore made up of a column of voltage readings per LED wavelength with each row being a sample time—two columns if using two LEDs, three columns if using three LEDs, and four columns if using four LEDs. We account for two forms of obtaining input data: obtaining data through existing Excel files, and obtaining data live from the RSL10 through a Bluetooth connection. The data from excel files can be uploaded through the programs ReadCalc2.m, ReadCalc2eval.m, and ReadCalc3.m. ReadCalc2.m and ReadCalc2eval.m are used for reading data obtained from 2 LED wavelengths, and ReadCalc3.m is used for reading data obtained from 3 LED wavelengths. The ReadCalc2eval.m program is a variation on ReadCalc2.m that is designed to obtain data obtained from the AFE Evaluation Board and attached finger pulse oximeter, which use different LED wavelengths than our own board. Having these two forms of uploading data allows us to be able to process and display both saved and live data, a useful feature for a health data product such as ours.

Now that the concentration equation coefficients are determined and there is input data available, we can process the data and calculate wanted information. For molecular concentrations, we have the function programs calc2concs.m and calc3concs.m. As their names would suggest, calc2concs.m calculates concentrations from two LED wavelengths (namely oxygenated and deoxygenated hemoglobin), and calc3concs.m calculates concentrations from three LED wavelengths (namely oxygenated/deoxygenated hemoglobin and water). Concentrations are given in the units of moles/liter (mol/L, or simply M). The two programs obtain their A/L coefficients from the calc2coeff.m and calc3coeff.m, respectively. They also include estimations of the incident light intensities for each wavelength (in volts) and average path lengths for each wavelength (in centimeters). These functions are called in the aforementioned ReadCalc2/2eval/3.m programs. The first and second derivatives of the concentrations are also calculated in the derivatives.m function program, which is slightly altered version of the derivative.m program from the previous year’s project. The figure below shows the equations for finding the concentration of deoxygenated hemoglobin using one, two, or three LED wavelengths, respectively. The one wavelength equation is derived from the the modified Beer-Lambert Law (Kocsis), which can be altered to include more wavelengths by using linear regression. The two wavelength concentration equation was previously calculated in the Kocsis paper and was used in the previous year’s project, but we had to derive the three wavelength equation ourselves. For each wavelength, A is the natural logarithm of the incident light intensity over the the detected light intensity, L is the average path length, and α is the molecule’s molar extinction coefficient. It is easy to see that calculating molecular concentrations increases greatly in complexity with each wavelength added.

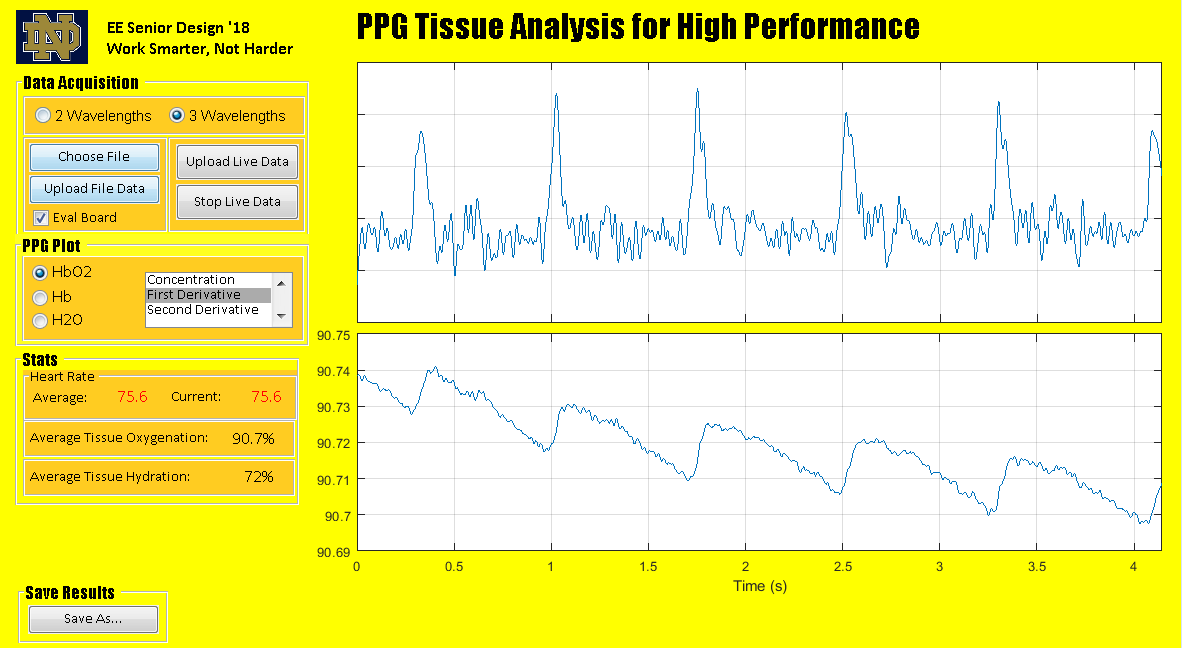
**Figure 9**. Example concentration equations for one, two, and three wavelengths.

As for other calculations, the heart rate is calculated through the heartrate\_section1.m and heartrate\_section1eval.m function programs. These work by finding the peaks in the photodiode voltage data and, taking into account the sample frequency of the board or evaluation board, estimating the heart rate by calculating the number of peaks (beats) per minute. The programs required no significant changes or additions from the previous year’s code. Tissue oxygenation, a new program feature for our product, is calculated by taking the calculated concentration of oxygenated hemoglobin and dividing it by the total concentration of hemoglobin (oxygenated plus deoxygenated). This calculation was done in the ReadCalc2/2eval/3.m programs after the concentration calculations. Finally, tissue hydration, another new program feature, is calculated by comparing the calculated water concentration with the known number of moles of water per liter of water (considered as 100% concentration).

Now that the data is processed into meaningful information, it can be displayed to the user. We decided to do this through an interactive graphical user interface (GUI). We made two different versions of the GUI. One built solely for use with file data of two wavelengths (testgui.m and testgui.fig), and is very helpful for the simple processing of saved evaluation board data. It has a button for choosing a data file, a button for uploading the data from the chosen file, a checkbox for using the evaluation board, two graphs, a panel with chosen calculated stats, and a button to save the processed data. In one graph, it displays a plot of the concentration, concentration first derivative, or concentration second derivative for either oxygenated or deoxygenated hemoglobin. In the other graph, it plots the calculated tissue oxygenation versus time. On the left of the graphs is also displayed the following information in the Stats panel: the average heart rate, the current heart rate (heart rate of the last four seconds), and the average tissue oxygenation

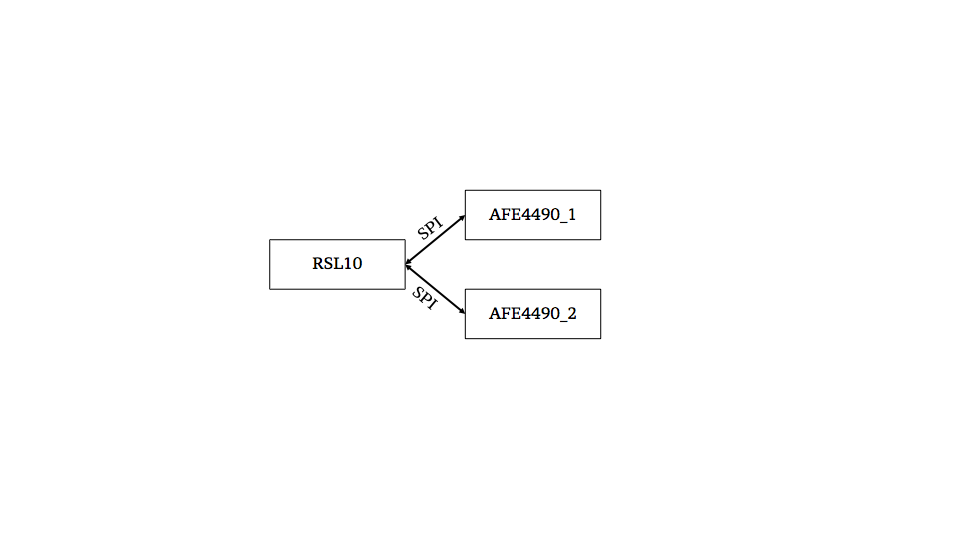
. **Figure 10**. testgui GUI displaying example processed evaluation board data.

The other GUI (SD\_GUIfinal.m and SD\_GUIfinal.fig) is a more detailed version of testgui, for use with two or three wavelengths. It displays the similar information, with the addition of the option to plot the concentration/first derivative/second derivative of water in the first graph, and the average tissue hydration in the Stats panel. The majority of the testing of our data processing involved the extensive use of the evaluation board, and comparing our calculations to the literature and to the calculations from the previous year’s project. As the math is sound, we are confident in the results.

**Figure 11**. SD\_GUIfinal GUI with example processed data.

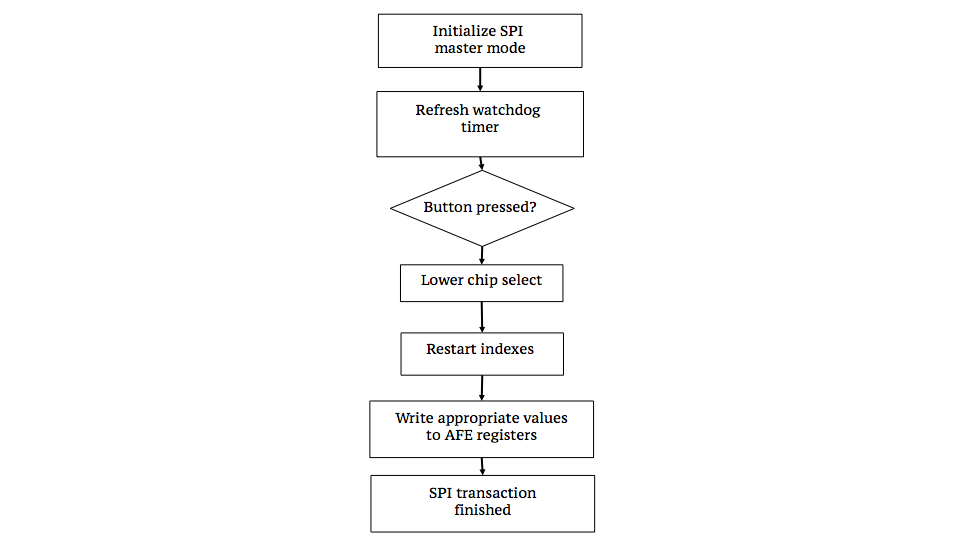
3.5 Detailed Design/Operation of RSL10 Communication with AFE4490

*System Flow Chart*



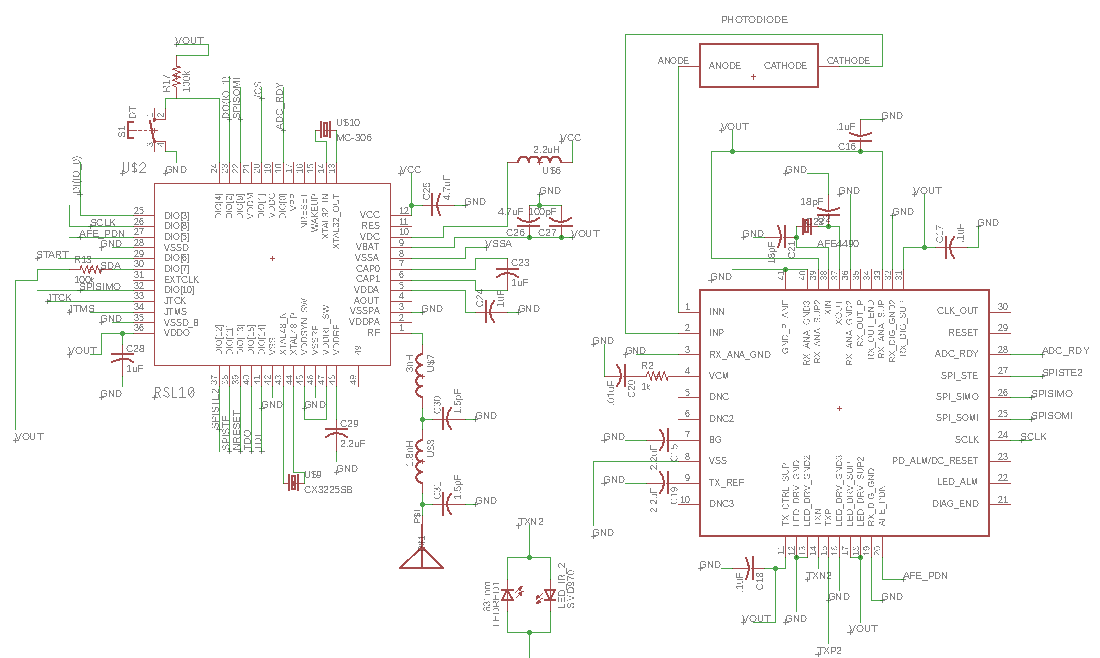
**Figure 12 .** RSL10 and AFE4490s Flow Chart

*Software Flow Chart*

**

**Figure 13** . RSL10 to AFE Software Flow Chart

*Schematic*



**Figure 14.** Schematic of RSL10 (left) and AFE4490 (right). The pin configuration is the same for the second AFE4490.

*Function and Connection to Other Subsystems*

The RSL10 is the master to each AFE4490 digital to analog converter. The RSL10 dictates the clock rate of the AFE4490, which then controls the LED flashing rate. Once the RSL10 (the master) communicates with the AFE4490 (the slave), the LEDs and photodiode subsystem send a signal back to the AFE4490, which converts the signal to analog and sends it to the microcontroller.

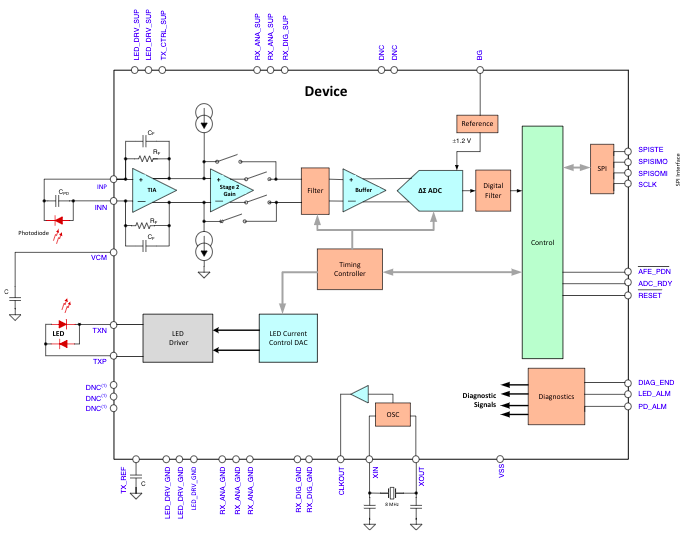
The specific pin configuration are shown on the schematic above. The SPI connections between the RSL10 and AFE4490 include:

* the SPI clock at pin 26 (SCLK - DIO[8]) on the RSL10 to pin 24 of each AFE4490
* chip select at pins 38 (SPISTE - DIO[11]) and 37 (SPISTE2 - DIO[12]) on the RSL10 to pin 27 of each AFE4490
* SPISIMO at pin 32 on the RSL10 to pin 26 of each AFE4490
* SPISOMI at pin 22 on the RSL10 to pin 25 of each AFE4490

The RSL10 sends a clock signal to the AFE4490s via SPI communication, which is enabled by the chip select pins. On the AFE the SPI serial in master out (SPISIMO) receives signals from the microcontroller. The SPI serial out master in (SPISOMI) sends the data received from the LEDs and photodiode subsystem to the RSL10.

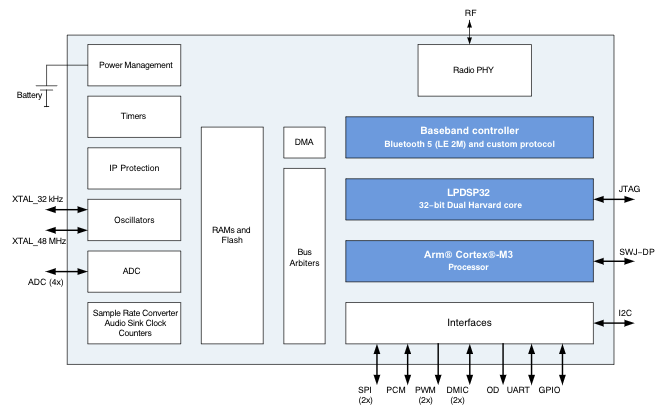
The AFE power down input (AFE\_PDN), set as active low, at pin 20, connects to pin 27 (DIO[5]) of the RSL10. The ADC\_RDY at pin 28 of the AFE4490 is an output signal that indicates if the analog-to-digital conversion is complete. The output is the input to pin 18 (DIO[0]) of the RSL10.

The functional block diagram for the AFE4490 is shown below:



**Figure 15** . AFE Internal Block Diagram

The functional block diagram for the RSL10 chip is shown below:

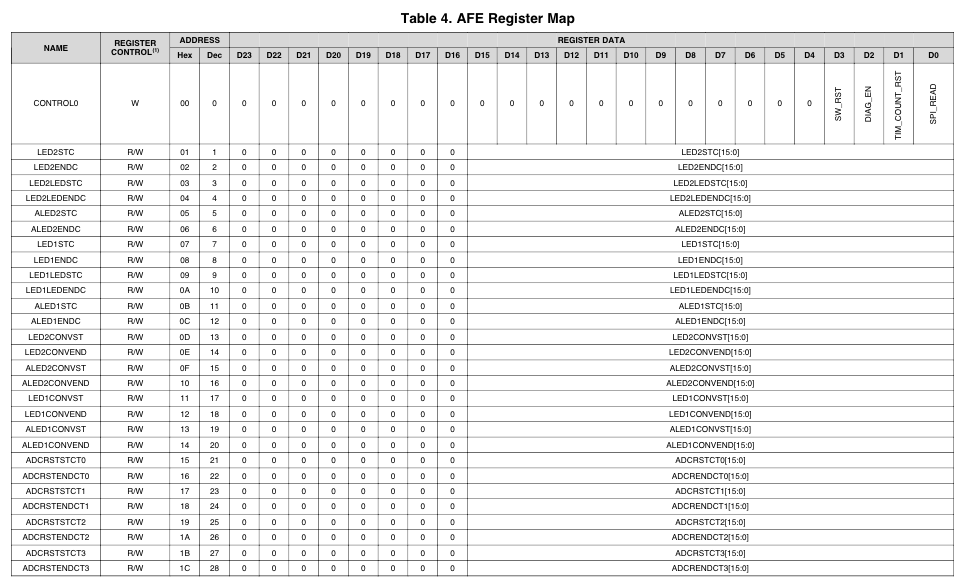


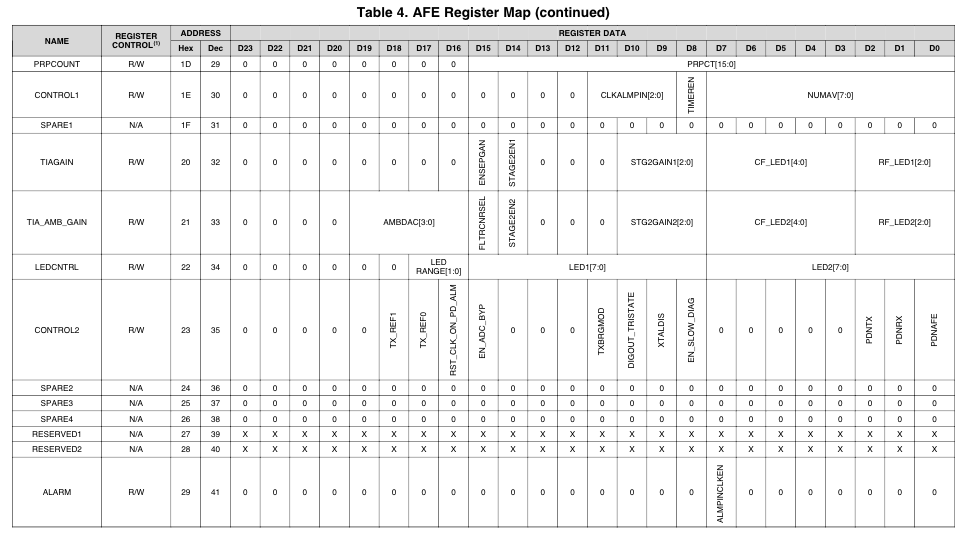
**Figure 16.** RSL10 Internal Block Diagram

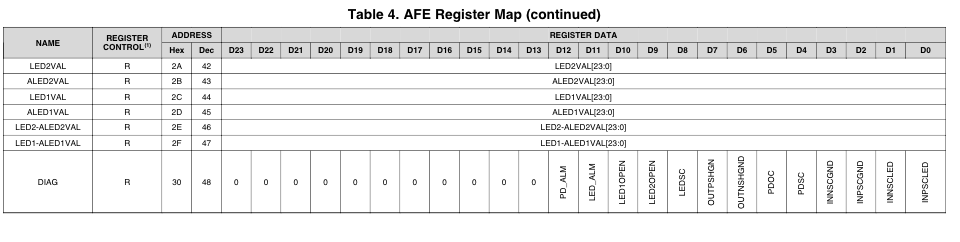
*Function and Connection to Other Subsystems (Software)*

The digital subsystems communicate via serial peripheral interface (SPI) bus. The RSL10 and AFE4490s exchange data over master-out-serial-in (MOSI) and master-in-serial-out (MISO) connections, with the RSL10 as master and each AFE4490 as the slave. The SPI is first initialized by masking all interrupts, configuring the RSL10 as master, configuring the type of data transfer (read/write) configuring the SPI DIOs (digital input/outputs), then enabling interrupts and stop masking interrupts. After the SPI is initialized, the watchdog timer is reset. This automatically generates a system reset if the main program neglects to periodically service it (this is often used to automatically reset a device that hangs because of a software or hardware fault. The program waits until the data collection button is pressed by the user. When this happens, the chip select is lowered to give time for the slave (the AFE4490) to prepare. The transmit buffer index is restarted, and the series of write instructions set up the extensive timing module on the AFE4490, controlling the LED flashing, photodiode sampling sequence, and analog-to-digital conversion, as well as the specifications for the gain stages, control registers (delineating various operational modes and reference voltages), and the current values for the LEDs to manipulate brightness.

Below is a register map of the AFE4490:

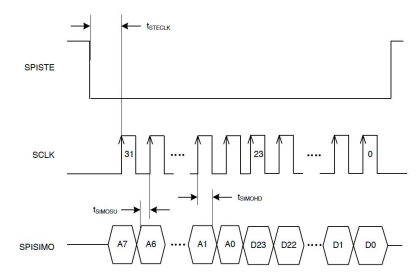




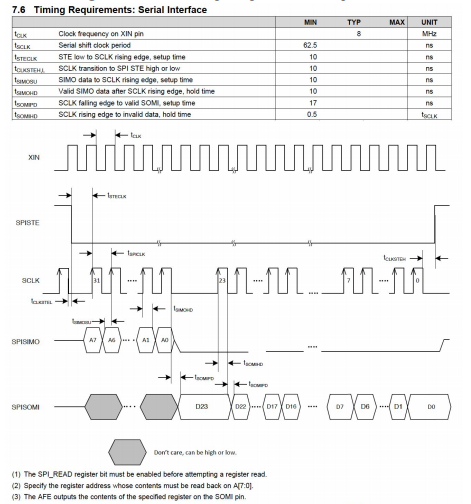


**Figure 17** . AFE Register Maps

After all of the registers above are initialized, the final bit of the Control 0 register is switched from 0 to 1 in order to put the AFE into a continuous read mode during which it outputs data to the RSL10. The instruction and data flow for the SPI communication between the RSL10 and the AFE are shown below:



**Figure 18** SPI Timing Diagram, Write Operation



**Figure 19** SPI Timing Diagram, Read Operation

*Subsystem Testing*

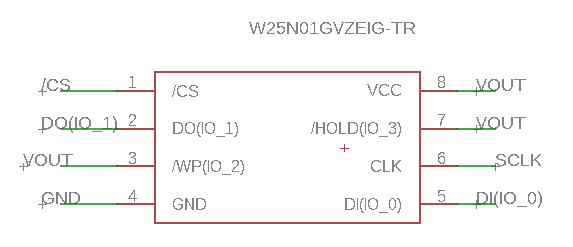
To build and test the RSL10 code for this subsystem, many of the functions were adapted for the purposes of this design project. The RSL10 development board, in particular, was helpful in building and testing the code for this project. The firmware manual provided a convenient software layer to build the applications, and included function listings and usage examples to help understand the RSL10 firmware and its parts. In particular, the sample programs SPI Master Mode and SPI Slave mode, which demonstrated SPI port usage, showed the proper SPI configurations between two peer Evaluation and Development boards in different roles, and the use of SPI (TX, RX) and GPIO interrupts. These programs were instrumental in the design and implementation of the final code for programming the AFEs via SPI. Due to hardware constraints, we were unable to test the program, but it successfully compiled and built, and based on the information and examples that were provided as part of the RSL10 development board, as well as the successful execution of many of the example programs, we believe that the programs would have executed.

3.6 Detailed Design/Operation of RSL10 Communication with Accelerometer

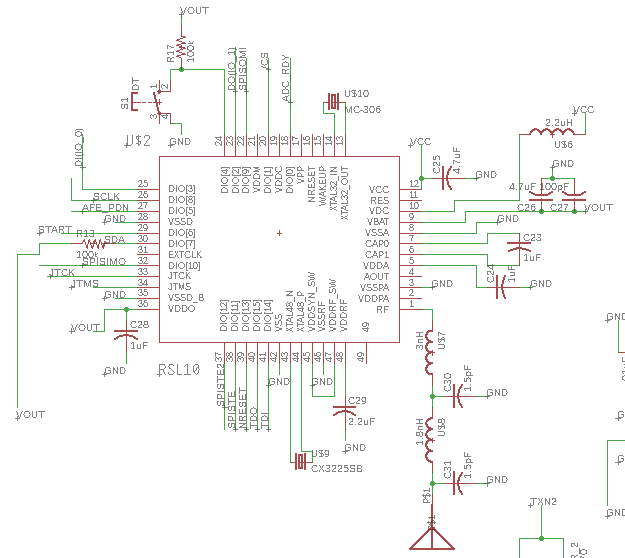
The RSL10 communicates with the accelerometer using SPI. After every LED cycle, the RSL10 will issue three commands to address 0x00 on the accelerometer: 0x08, which will prompt the accelerometer to return the lower byte of its two-byte measurement of the movement along the x-axis; 0x0A, which does the same for the y-axis; and 0x0C, which does the same for the z-axis. Once this data has been read the RSL10 determines the total motion of the accelerometer, and if it finds it to be moving it will set a flag that will be stored with that set of photodiode data.

3.7 Detailed Design/Operation of RSL10 Communication with Flash Storage

*Schematic*

**

**Figure 20**: Flash memory (Winbond W25N01GVZEIG TR) schematic



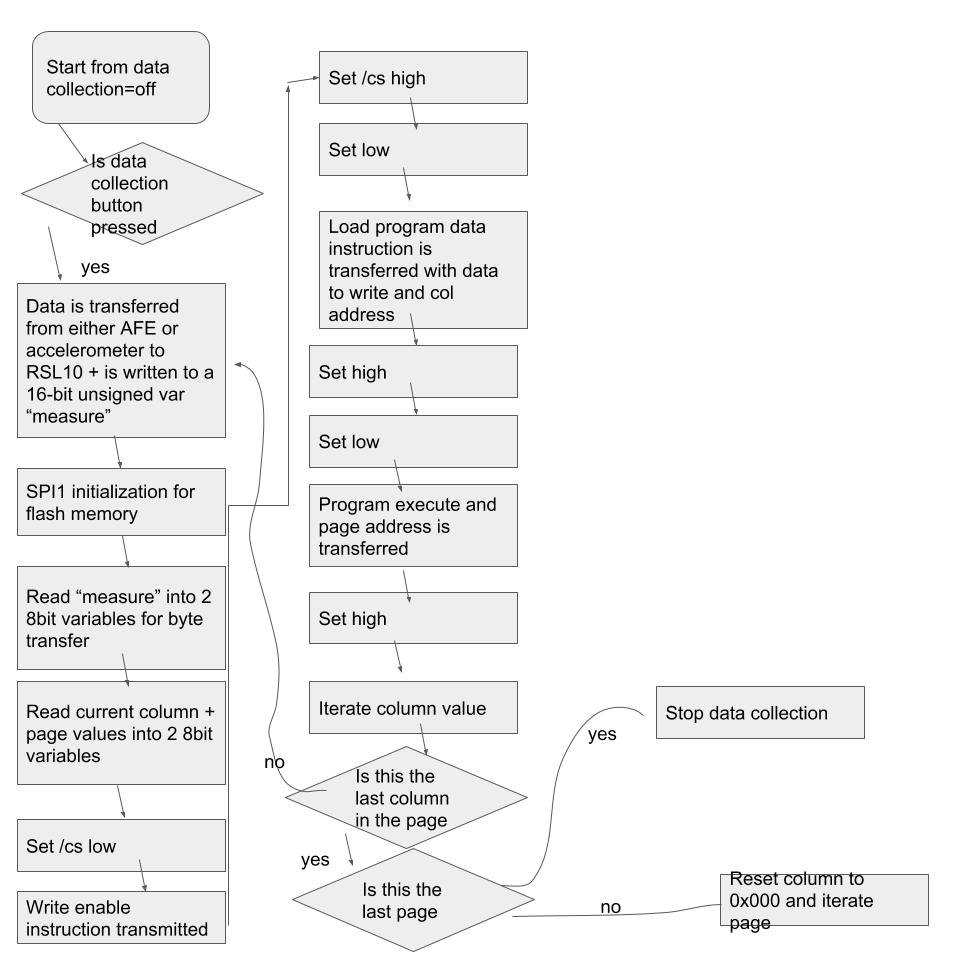
**Figure 21**: Memory component connects to RSL10 microcontroller

In terms of hardware, we chose to use the Winbond W25N01GVZEIG TR, a 128MB flash memory component which is capable of interfacing with a microcontroller through SPI protocol. This component was chosen to replace the SD card component from the previous iteration of this project. This was done for several reasons. Interfacing with the SD card required the usage of legacy software and libraries in order to be able to write files to it, such as the .txt file written in the previous iteration of the project. This dependency is a major downside in writing code to interface with the SD card component, as these libraries are not supported and they are difficult to find and get to work properly. With flash memory, data can be written directly with no need for a legacy OS or file handling. Additionally, the SD card slot takes up more space on a board than a flash memory component does. The board should ideally be as small as possible such that the entire design can fit on an applicable section of skin and take usable measurements. The human wrist is a good example of such an application; the board can be worn on a wristband or armband such that the LEDs and photodiode are face down on the skin’s surface while the button interface is on the side facing up. For comparison, an SD card slot’s footprint is 11mm x 15mm, while our selected flash memory chip has a footprint of 8mm x 6mm, almost a 71% size reduction. These benefits led to choosing flash memory for our storage needs. Preliminary calculations based on sampling rate and bits/sample led to a guess of 49.5 MB written per hour, and 198 MB written per 4 hours. A 4 hour usage case was chosen as a conservative estimate of workout time.

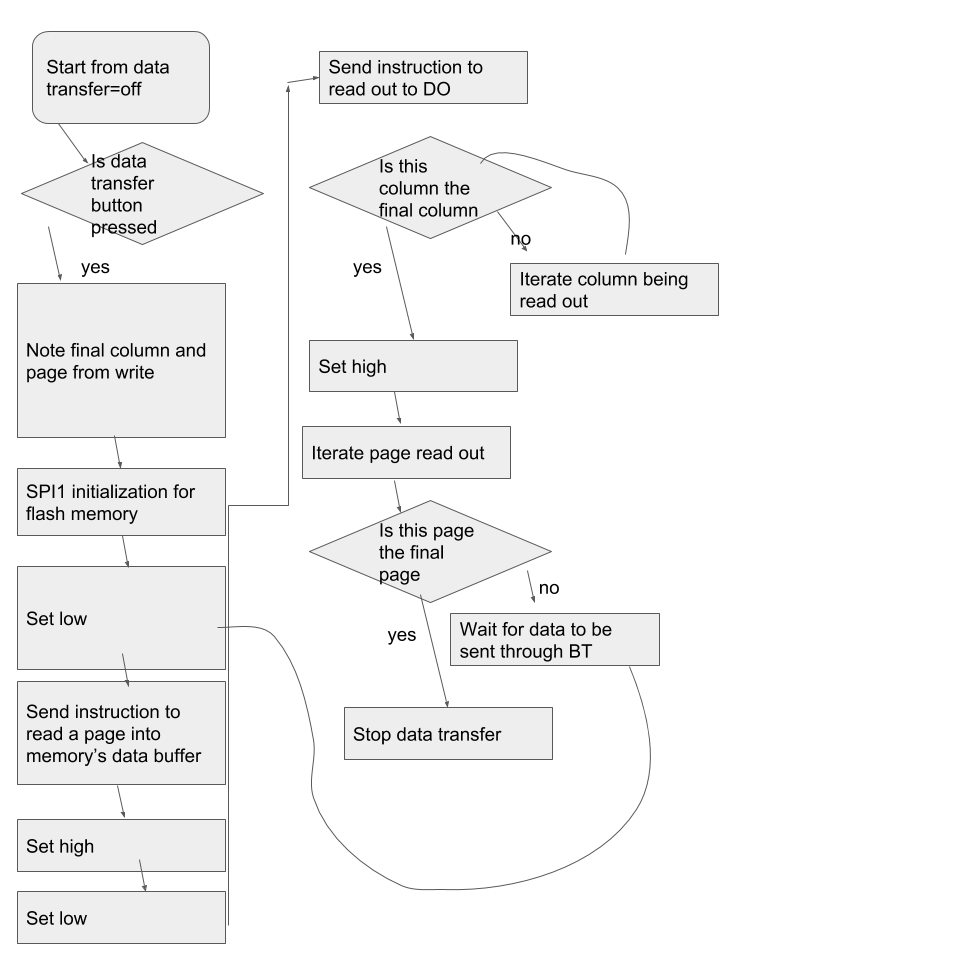
This meant that for 4 hours, we would want 198MB of available flash memory to continuously write to. As our requirements dictated that the flash memory be SPI (the AFEs are limited to SPI interface, and the RSL only allows for 2 SPI configurations - we found an I2C interface accelerometer, taking up the only I2C configuration, so we needed to find an SPI flash memory component), this was a limiting factor, and it was difficult to find surface mount flash memory chips with sufficiently large storage capacity. In the end, the 1Gb=128MB was the largest capacity flash memory chip we could find meeting all our other criteria, and this is theoretically still good for measurements during ~2.6h of continuous exercise. The way of thinking about writing to storage evolved the more familiar we became with the flash memory structure, and our preliminary calculations no longer apply.

In terms of circuitry, the storage component does not have many requirements. The SPI interface-relevant pins are connected to the RSL10 microcontroller, these being /CS (chip select), DO(IO\_1) (data out), DI(IO\_0) (data in), and CLK(SPI clock). /CS must be able to track the supply voltage, so an internal pull-up resistor to the corresponding RSL10 DIO pin is set through RSL10 registers. We ran out of DIO pins on the RSL10 so we weren’t able to tie the write protect and hold pins to RSL10 pins to be configured through code, instead opting to tie write protect and hold to VDD, as this allows for us to actually write to the storage, disabling write protect and hold functionality. The GND (ground) pin is tied to overall ground for our board, and the supply voltage is tied to the output of our 3.3V DC to DC converter. These supply the component with power. The SPI pins allow the RSL10 microcontroller to interface with our flash memory component. The SPI interface will run off of a clock generated through a RSL10 DIO. /CS is set active when low, and inactive when high, and these signals are also supplied through an RSL10 DIO. Instructions are written to the flash memory through the data in pin when chip select is low. If an instruction is written to this component which calls for data to be input to the master, that data will be transferred through the data out pin until chip select is pulled high.

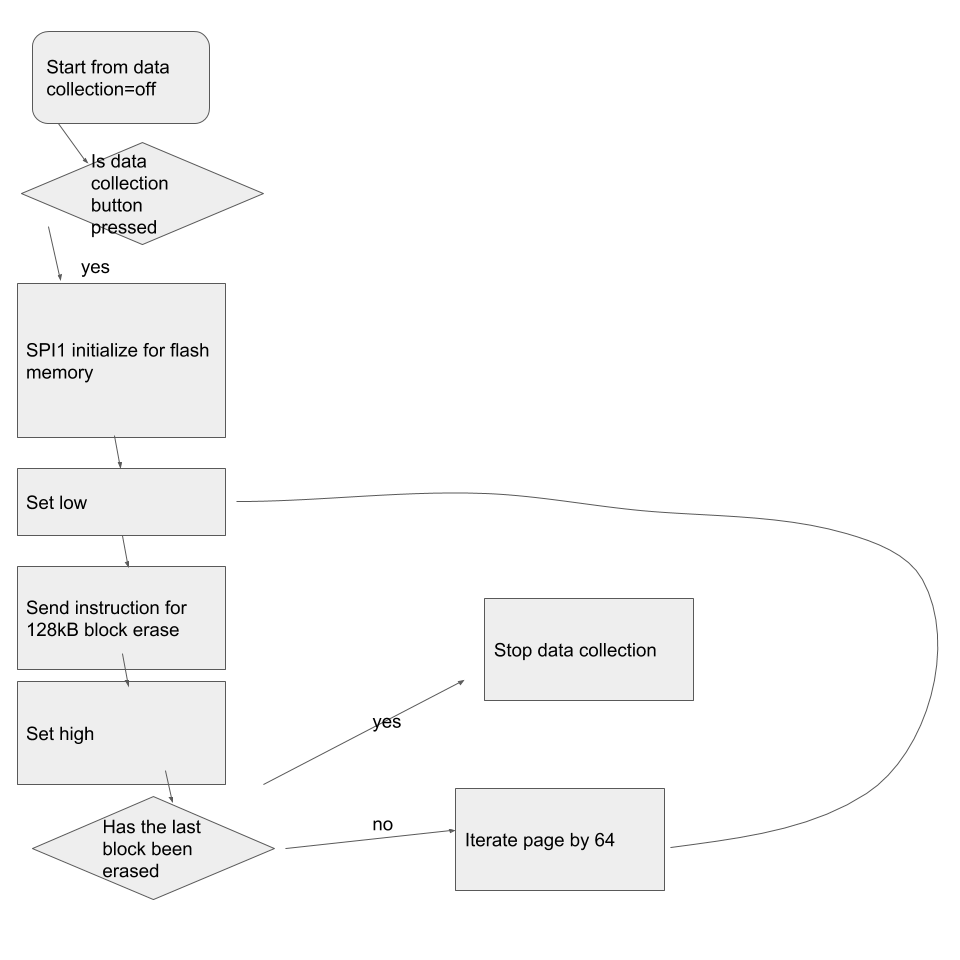
*Software Flow Chart*



**Figure 22**: flow for “write” function



**Figure 23.**  flow for “read” function

**

**Figure 24**: flow for “erase” function

There are 3 main software functions associated with the flash memory: write the data to the flash memory, read the data out from the flash memory, and erase the flash memory. The 3 flow structures give a broad overview of the functions, and all “instructions” sent to the flash memory through SPI are described in detail in the Winbond W25N01GVZEIG TR data sheet, section 8. Additionally, in my code for these functions you can see the exact method I used to send instructions and in which exact order I send them.

Erase runs immediately after the data collection button is pressed in the “data collection off” device state. SPI1 is initialized for the flash memory, and the instruction is sent for a 128kB block erase, the block being defined by following the instruction with a page address (all instructions are preceded by setting /CS low, and finished by setting /CS high). After the block has been cleared, a check is run to see if the block just cleared is the last block in memory. If it isn’t, the page number is iterated by 64 (1 block being 64 pages) and the instruction is sent again. If it is, data collection begins.

The data collection state of the device involves cycling through taking measurements from the AFEs and measurements from the accelerometer. Each measurement is followed by a write of that measurement to memory. This write occurs by initializing SPI1 for the flash memory, then manipulating the measurement such that there are no decimal values (multiply the small measurements by 10000, cast to int, cast to unsigned short). After MATLAB readout in the data transfer phase, these inflated values can easily be changed to the original decimal form by multiplying the column values by the inverse scalar (1/10000). The measurements, page and column values are all 16-bit unsigned shorts and are broken into 8-bits (1 byte) for SPI data transfer. The first instruction sent is a write enable, which allows us to write. This bit is automatically reset after writing to memory. The next instruction is to load program data, which means that any data we specify in the instruction to the flash memory chip will be read into the flash memory’s data buffer. This sets up our last instruction, which is the program execute instruction. A page address is specified and the data buffer is written to the desired memory address. We then iterate the column value by 2 bytes (as 2 bytes from the “measurement” value are written.) We check if it is the last column in the page, and if it is, if we are on the last page in memory. If it is the last page, then we have written to every location in memory and we stop the data collection. If it is not the last page, we reset the column value to 0 and iterate the page. To understand the structure of memory in this particular chip, I recommend looking at section 5 of the flash memory datasheet.

Reading is initiated by pressing the data transfer button from a non-data-transfering state. The final written value’s column and page are saved so that we know when to stop reading out data. SPI1 is initialized for flash memory. The instruction is sent to read a page into the data buffer (“page data read”), with the page being specified in the instruction. This sets up the next instruction, “read”, which reads the specified page data out via DO to the RSL10 column by column (byte by byte). If the last written memory address is not in this page, the loop of columns will finish and /CS will be set high. If the final written memory address is in this page, when this address is reached, the “read” instruction will be terminated and data transfer will stop. In the RSL10, the data sent from the flash memory will be stored in a receive buffer. An array of bytes the size of a page is created for all the data to be stored in. Once one page is read out, the program will wait for the array to be successfully transmitted via Bluetooth before the next page is read into the array.

*Function and Connection to Other Subsystems*

There are several functions of this subsystem, but they all center around the necessity of ample flash memory to serve as storage so that the data can eventually be transferred to another machine to undergo post-processing.

One function of this subsystem is to allow for the AFE4490s and accelerometer to ultimately write the vital output data to storage. The AFE4490s and accelerometer cycle in terms of interfacing with the RSL10, such that only one output value is being read into the RSL10 at a time. That output enters the RSL10 through the receive buffer. In order to avoid dealing with decimal point values less than one (due to writing unsigned bytes to flash memory directly), these output values will undergo some alterations to get them in a recoverable format. The receive buffer data is read to a float, which is immediately multiplied by 10,000. Values output by the AFE4490s are around the tenth place (10^-1), so this method will typically guarantee 4 significant figures that will be saved. The float is cast to an int to remove the decimal points, and this int is cast to a 16-bit unsigned short “measure” for sending in 8-bit chunks to the flash memory component. This subsystem is connected to accelerometer and AFE4490 data transfer subsystems in that it is called after each write from these components to the RSL10, to clear the buffer and ensure that all the data just written to a variable in the RSL10 is written to a more permanent memory location.

Another function of this subsystem is to read out the memory to the RSL10 during the data transfer device state. This is important so that the data we stored in memory can be transferred out of the flash memory chip to an external device with data analysis capabilities and the data can actually serve its purpose in providing the user with relevant metrics. In this respect this subsystem is connected to the Bluetooth data transfer subsystem, which handles the transfer of data from the RSL10 to a connected Bluetooth device. It is also connected to the MATLAB data processing subsystem, as the RSL10 will write the data it receives from this subsystem through directly to MATLAB to generate a matrix from the raw data, which can then be used for analysis.

The final function of this subsystem is to erase the flash memory chip so that new data can be written to the chip for later analysis. This function is important so that the device can be used multiple times for different workout sessions; otherwise the flash memory would fill up after only a couple sessions and become unusable.

*Subsystem Testing*

We were not able to test the functionality of the flash memory component, as we received the part late into the semester and did not have a board on which to test the component until around this same late point. Going forward from this point, this can be tested with the RSL10 evaluation board. A test board can be designed very easily, with a 8-WSON footprint for the flash memory chip connected to test pins, such that the component can be easily hooked up to the RSL10 evaluation board SPI pins and power through multiple jumper wires. Additional test pins can be included for easy logic analyzer application. We can load sample code onto the RSL10 evaluation board that writes to a specific location in memory, and then reads from that location. Once this is accomplished, more intricate sample code can be written to test our understanding of the memory structure to ensure we can write to and call from the correct addresses. Write, read, and erase instructions can all be tested. These can be verified through the use of a logic analyzer hooked up to the free test pins. The logic analyzer software can be loaded, and the program can be run. We will specifically look for the instruction and information bytes we intend to send, and troubleshooting can proceed as the situation calls for until basic write, read and erase instructions are achieved. Achieving this level of functionality is the core of this subsystem, and the basics can then be applied to the overall device code structure for tighter integration.

**4 System Integration Testing**

Initial testing began with last year’s board, which contained the foundational functionality of this year’s design. However, despite multiple attempts to fix potential software and hardware issues, the board did not function properly.

The final board, which included all of the new components and design, including the accelerometer, RSL10, two AFE4490s paired with the new LED values, and the flash storage device, as well as the buttons for starting and finishing data collection and transfer over Bluetooth, arrived later than desirable and resulted in abbreviated system integration testing. In order to integrate the various subsystems prior to the board’s arrival, we utilized the RSL10 and AFE4490 development boards. The I2C communication from the RSL10 was confirmed using the development board and a logic analyzer. Likewise, the Bluetooth connection was first confirmed using to MATLAB was confirmed using the development board and the MATLAB program. The AFE4490 development board was used to obtain data to be processed in the MATLAB GUI. The SPI interface code, which controls the flash storage and the AFE4490s, was successfully configured and built.

The interface between hardware and software, programming the RSL10, proved problematic because we could not successfully connect to the RSL10. After confirming the hardware connections and voltages throughout the board, we discovered that the reset pin was connected to JTAG, instead of the RSL10. We attempted to re-connect it with wire, and sever the incorrect trace. This did not fix the problem of being able to program the RSL10 successfully, and because of time constraints, we were not able to successfully connect the hardware to the software.

If this problem had been alleviated in time, the system integration testing would include downloading the RSL10 code onto the device and testing various components with the logic analyzer. First, the RSL10 SPI communication to the AFE4490 would need to be confirmed via a logic analyzer. We would need to test and ensure that the LEDs were flashing at the correct rate (so as to ensure that their light does not interfere) using a power meter. Communication back from the AFE4490s to the microcontroller would need to be tested. To test this SPI communication, a logic analyzer could be hooked up to the CLK, SPISTE (SPI enable), SOMI, and SIMO pins of the RSL10. The SIMO connection was used to write values to the registers within the AFE, controlling the timer, gain, LED current, etc. The SOMI responded to the SIMO connection with the converted digital values (from the photodiode) contained in the register being read.

Next, the SPI communication between the flash storage and the RSL10 would be confirmed, again using the logic analyzer. Then, the I2C communication between the accelerometer and the RSL10 would be confirmed. Finally, the transmission of data via Bluetooth to the computer would be tested. The values from the flash storage would need to (approximately) match data obtained from the AFE4490 development board. The accelerometer data would need to be confirmed with the readings. Finally, the MATLAB post-processing program would need to successfully process the incoming data from the board.

**5 User’s Manual/Installation Manual**

5.1 How to Install and Setup the Product

The board requires only a battery to be ready for use. The GUI requires the user to have MATLAB installed, including the Instrument Control Toolbox. Additionally, the user’s computer needs either a built-in bluetooth adapter or an external bluetooth dongle attached.

5.2 How to Use the Product

To collect oxygenation and hydration data using the board, the user should hold the board to the wrist using a wristband and press the black button to put the board into data collection mode. Once exercise is done, the black button can be pressed again to stop data collection. To send the data to MATLAB, the user should press the blue button to cause the board to begin searching for a computer. Note that the blue button can also be pressed while in data collection mode, which will also end data collection. The user can then pair the board with MATLAB, at which point the board will automatically upload all its collected data into a file which MATLAB will use to create graphs of the user’s hemoglobin and water levels, as well as oxygenation, versus time.

5.3 How the User Can Tell if the Product is Working

The user can confirm that the board is in data collection mode by checking the LEDs on the back of the board. The red and green LEDs emit visible light and should be seen blinking on and off. The user can confirm that the board is successfully broadcasting bluetooth by using a computer or smartphone to see if the board is discoverable as a bluetooth device.

5.4 How the User Can Troubleshoot the Product

Due to limited input/output pins on the RSL10, there is no on-board troubleshooting method for diagnosing hardware problems. However, it is still possible to determine the cause of a failed pairing by connecting a bluetooth test app to the board. If the user sends the board the command 0x1111 and receives data back, the board is working and there is a problem with either the computer running MATLAB or MATLAB itself.

**6 To-Market Design Changes**

Due to budget and time limitations, the product of this senior design project differs from the product that would be taken to market. There are a few modifications and enhancements that would be necessary before selling this product commercially.

Though the board was much smaller than last year’s prototype, and could, feasibly, fit comfortably on the wrist, realistically, it would need to be slightly smaller. This could be done by simply rearranging the board layout to save more physical space.

Additionally, an iPhone/Android app that could collect and process the data from the device via Bluetooth in real time would prove significantly more useful for athletes on the go to get information about their body during a workout (i.e., they would not require a constant connection to a computer running MATLAB to be able to see data, or wait to process a saved data file).

Another change that could be made would be to choose the optimal wavelengths for hemoglobin and water concentration calculations. There have been a few studies in which these optimal wavelengths have been calculated, and choosing these could help increase the quality of our product’s input data, and accuracy of the processed data.

**7 Conclusion**

By expanding upon last year’s ‘Heart of the Matter’ project, which utilized photoplethysmography to monitor heart rate and blood oxygenation in order to detect the symptoms of cardiomyopathy, we developed a sophisticated and robust high performance activity minor. Our device retained the functionality of last year’s project (measuring oxygenated and deoxygenated hemoglobin), but added key features that would appeal to a high performance athlete. It added an additional wavelength to measure water content in tissue, which describes hydration levels. An accelerometer was added to account for motion, and a green LED was incorporated in order to make the device more adaptable to error. The device also contains flash storage, for extended workouts, and integrated bluetooth capabilities for easy data transfer to a computer. These features enable the device to serve as a precise assessment of health during strenuous physical activity, and monitor and optimize performance.

Throughout the design process, the ‘Work Smarter, Not Harder’ team polished old skills and acquired a plethora of new skills. We developed a concept and tailored it toward a specific market demand and technological need. We pored through various research sources to learn about which components would be most applicable for our project. We then used their datasheets to design a board and put together software. After a significant amount of work, the team managed to develop a robust (though imperfect) product. With additional time or an improved board development timeline, the team feels confident that they would have succeeded in addressing the remaining issues and creating a fully-functional product.

**8 Appendices**

These appendices can be found on our website:

http://seniordesign.ee.nd.edu/2018/Design%20Teams/heart2/

**Appendix A:** Hardware Schematics

**Appendix B**: Parts List and Links to Component Datasheets, Research Links

**Appendix C**: Complete Software Listings