University of Cincinnati

Date: 8/26/2015

I, Nirjhar Bhattacharjee , hereby submit this original work as part of the requirements for the degree of Master of Science in Electrical Engineering.

It is entitled:

A Real-Time Data Acquisition and Digital Signal Processing System for Neuromonitoring

Student's name: **Nirjhar Bhattacharjee**

This work and its defense approved by:

Committee chair: Chong Ahn, Ph.D.

Committee member: Chunyan Li, Ph.D.

Committee member: Jed Hartings, Ph.D.

Committee member: Philip Wilsey, Ph.D.

19496

A Real-Time Data Acquisition and Digital Signal Processing System for Neuromonitoring

A thesis submitted to the

Division of Research and Advanced Studies of

the University of Cincinnati

in partial fulfillment of the requirements for the degree of

Master of Science

in the School of Electronics and Computing Systems of

the College of Engineering and Applied Science

2015

By

Nirjhar Bhattacharjee

B.Tech, National Institute of Technology Silchar, 2009

Committee Chair: Dr. Chong H. Ahn

Abstract

This work reports a versatile real-time data acquisition, signal processing and sensor calibration system using NI (National Instruments) LabVIEW. The goal of the developed system is to simultaneously monitor different physiological parameters in an animal's brain during traumatic brain injury (TBI). The system is developed in conjunction with the smart catheter sensor array which is an integrated platform of multiple microsensors in a single microcatheter device. The sensors simultaneously records key physiological parameters from an animal brain which includes temperature, partial pressure of Oxygen, Glucose, Lactate, Cerebral Blood Flow (CBF) and neural activity. The system after data acquisition and processing displays the real-time data in a user interface in a clear and concise way. A sensor calibration software is also developed using LabVIEW which provides calibration solution for the Smart Catheter sensors.

The developed system consists of a front end motherboard circuit interface which encodes the real-time analog signal to serial bits and transmits it to a backend system for data-logging and signal processing. The back-end system consists of a NI cRIO (Compact RIO) hardware with serial I/O ports. The cRIO system consists of an inbuilt FPGA (Field Programmable Gate Array) chip and a Real-Time (RT) controller which are used for data processing and transmission to a Laptop computer with user interface. The program development was done using LabVIEW to create a high fidelity and fast response system. The developed system was tested with simulated signals to validate the system for long term monitoring application. Finally the Smart Catheter sensors were calibrated with the calibration function and in-vivo recording was done from a rat's brain. The results of the experiments have been reported in this thesis.

Acknowledgements

For all the opportunities and intellectual developments, I should thank many of my guardians, mentors and friends, without whom, my present state of existence would have been entirely different. These have been the people in my life who provided me constant support and nurture to help me reach the current stage.

Firstly, I would like to thank Dr. Chong Ahn, my advisor and guide, who helped me acquire the practical knowledge, not only in engineering but also important for life without which it would have been impossible to take the next step into the world outside academia. His encouragement and support played the most crucial role in helping me overcome many difficulties in my research. I will always be indebted to Dr. Chunyan Li for helping me with the animal test experiments and an immense source of stimulating ideas. I am also thankful to Dr. Pei-Ming Wu whose experience with hardware systems as well as programming helped me in dealing with complicated yet highly interesting challenges. I am thankful to Zhizhen, who has been a senior, a mentor and an even better friend who taught and helped me hone my skills in microfabrication and provided support and advice in the darkest hours.

I would like to thank my parents for being my pillars and providing me all the necessary support, emotionally as well as financially and igniting in me the love for science and mathematics. I thankful to Kasturi, the love of my life to be a source of my strength and boldly face many hardships. I would like to Binoy sir in my alma mater NIT Silchar, who was my first guru and made me realize the beauty of electrical engineering. I am thankful to my friends Pradeepta, Anukana, Sumit, Dhanashree and Shalini without whom life as a graduate student would have been utterly dreary.

Finally, I would like to express my gratitude the Department of Defense (DoD) for making this work possible with their grant (#W81XWH-10-1-0977) and helping me earn an MS degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The cerebral cortex comprises of the external part of the brain that covers the cerebrum and the cerebellum (1.5 mm to 5 mm thick). It is the layer of the brain known as the gray matter. This is further divided into four lobes for classification, frontal, parietal, temporal and occipital. The most complex mental activities like speech, memory, problem solving, planning, awareness, and the execution of complex responses in the world are controlled by the cerebral cortex. Hence monitoring the cerebral cortex for various disorders becomes extremely important for proper diagnosis and care.

For the purpose of developing neuromonitoring platforms, basic knowledge about the disorders to be analyzed is necessary. The research work in this thesis comprises of developing a real-time neuromonitoring system for *in vivo* monitoring in animal brain for understanding traumatic brain injury (TBI) and its consequences.

1.1 Background

TBI is caused due to external force or high acceleration injuring the brain causing serious trauma. It is considered as a highly serious threat to health and wellbeing in higher income countries. In addition to an immediate injury to the brain tissues, TBI can inflict further damage hours and even days after the event which are classified as secondary injury. Changes in pressure and blood flow in the brain, contribute further to initial injury induced damage.

The Glasgow Coma Scale (GCS) is used for quantifying severity of TBI in patients. Using GCS, an individual's consciousness can be graded on a scale of 3-15. With the severity of TBI increasing down the scale [4]. An injury caused to the brain can be very different than any other injury. The repercussions of a brain injury can be a totally altered personality which affects the patient as well as others. The healing process of brain injuries is very different which may involve a functional recovery, the mechanism of which is not clearly understood. The type, severity as well as the consequence of the TBIs can be very different. Symptoms may be observed immediately or may not be present for days or even weeks after the injury [6]. Hartings et al [7] observed spreading depolarization (SpD) waves travelling on the cortex as an effect of TBI. Data on concurrent changes due to TBI in intracranial pressure, tissue oxygenation, and fever are not conclusive enough [8]. To understand the cause and effect of TBI or any other neurological disorder, it is important that to continuously monitor key physiological parameters of the brain. This motivates the work of developing a real-time multimodal monitoring system which is desired for neurocritical care.

1.2 Real-Time Monitoring and Data Acquisition Systems

Data acquisition and monitoring systems have been reported by many groups [1, 2]. Both Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Array (FPGA) based systems have been developed. FPGA based systems have an immense advantage over ASIC systems. Being reconfigurable and reprogrammable, FPGA systems can be easily upgraded to include additional functions. Gomez et.al, [9] reported an integrated data acquisition and signal processing platform. A hardware system with six channels, along with data processing software was developed. The advantage of the reported system was in its flexibility and scalability for

implementing new algorithms. Furthermore, Wilson et.al, [10] reported a data acquisition system for neuromonitoring application. A multimodal data acquisition system was developed using a C++ programming language. LabVIEW based systems have been developed and widely used for automotive applications [11]. Also available are commercial monitoring systems like the CNS Multimodal Monitor [12] which, though highly sophisticated in functionality lack the flexibility to be customized to implement complex algorithms and control functions. The advantage of LabVIEW based system compared to the others is in the simplicity in programming and scalability of the system without using any additional hardware [13]. Being a graphical interface for high level programming, it is easier to implement complex logical operations in LabVIEW compared to other programming languages. Moreover, pairing up the system National Instruments (NI) Compact Reconfigurable Input Output (cRIO) system which has inbuilt FPGA and a real-time (RT) controller enhances the computational capability of the whole system. The FPGA chip and the real-time controller can be programmed using LabVIEW. The system carries out multiple tasks for data acquisition and signal processing. Using the real-time controller in the cRIO has an advantage of a true real-time operation with minimum jitter [14]. The system was also enabled with real-time data acquisition and Digital Signal Processing (DSP) capability for multiple channels acquiring data at different sampling rates. In addition to these, a sensor calibration toolbox was also developed and incorporated in the system which is not available in other commercial systems.

1.3. The Smart Catheter Tube Integrated with Multiple Sensors

For simultaneously monitoring multiple parameters in the brain, biomedical and physical sensors have to be implanted for the duration of recording. Individual sensors can be used for each of the

parameters which are commercially available. This, however would involve craniotomy of a large area of the skull and would inflict significant damage to the brain. A smart solution to this issue was developed by Li et.al [3] where, a BioMEMS technology based microsensors (or sensor array) were developed as an integrated platform.

The smart catheter tube sensor array was fabricated on a flexible polyimide substrate using BioMEMS micro-fabrication process of depositing and patterning metal wires, enzyme layers, and insulation or passivation layers to form sensing structures which are fabricated on a flexible polyimide strip. After the completion of the fabrication process, the strip integrated with sensors is spirally rolled to form a lab-on-a-tube (LOT) with a drain channel for flowing Cerebrospinal fluid (CSF) out of the brain.

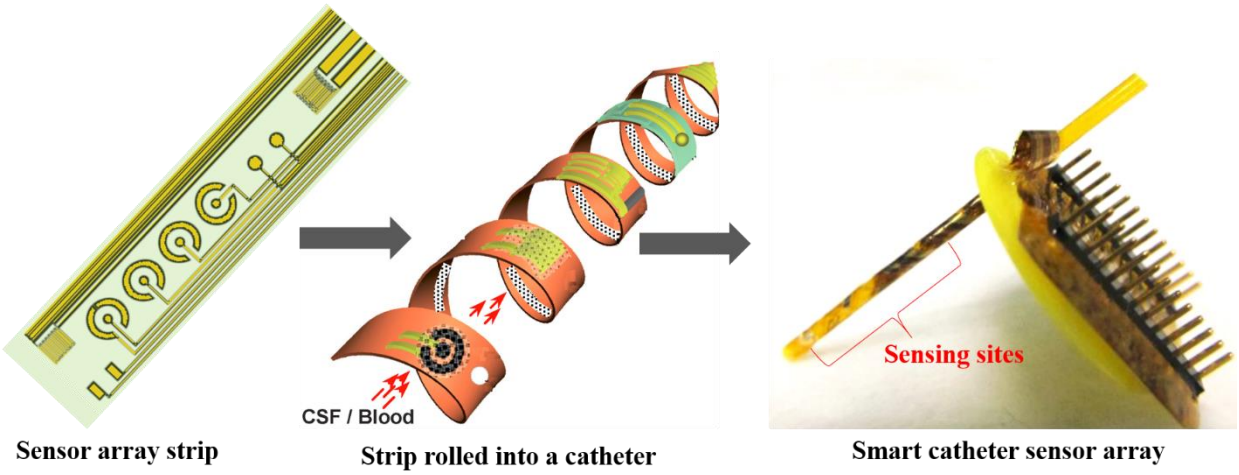


Figure 1.1 Microfabrication steps of the lab-on-a-tube based smart catheter with multiple sensors.

The Figure 1.1 shows the LOT based smart catheter. The sensors in the array simultaneously monitor key physical and electrochemical parameters as well as neuronal electrical activities. This is extremely useful as no additional device is needed for monitoring multiple parameters reducing injury to the brain due to implants.

1.4 Objective of Research

The primary goal of this work is to develop a real-time neuromonitoring and data acquisition system for the smart catheter sensor array. The Smart Catheter sensor is an integrated sensor array for real-time monitoring of Temperature, pO_2, Glucose, Lactate, CBF and ECoG. The sensor array and the interfacing circuit were developed at Feinstein Institute for Medical Research at New York. NI cRIO hardware along with NI 9870 serial I/O (Input/Output) ports were used as the data acquisition hardware. The RT controller in the cRIO has an embedded operating system. This provides highly reliable real-time data processing with minimum jitter. LabVIEW can be easily paired with the NI hardware for program deployment and hence was used as the development language.

The developed system has the capability to be scaled up for recording as many channels as needed. The most critical issue for a monitoring and data-logging system is loss of data during transmission. As the first goal, the work in this thesis addresses this issue by implementing codes for high-fidelity data transmission and parallel processing using FPGA. Another critical issue for digital signal processing systems are latency and transients in response. The second goal of this work addresses this issue and an efficient solution has been proposed to solve the problem. As the third goal, to provide an integrated solution for sensor calibration, a calibration toolbox needs to be developed in LabVIEW and incorporated in the system. The software will plot a calibration curve and provide the slope and intercept which can be used in the real-time signal from the sensors. The data-logging capability was provided to store the data in an external hard-drive. The data stored in text files can be used for off-line analysis using software such as LabChart Reader or MATLAB.

1.5. Outline of the Thesis

Chapter 1 begins with comparison of some data acquisition and monitoring systems in literature and introduces the MEMS based smart catheter sensor array. Finally, this chapter ends with the goals for this research work.

Chapter 2 talks about data acquisition and the algorithms developed and implemented for parallel data byte combination using FPGA.

Chapter 3 discusses the algorithms implemented in the RT controller for arithmetic and DSP operations. The chapter also discusses the Sensor Calibration function developed for running in the host computer.

Chapter 4 discusses the *in vitro* and *in vivo* experimental results obtained using the developed Multimodal Monitor.

Chapter 5 summarizes the thesis work and discusses the possible future work for implementation using the existing system.

Chapter 2

Data Acquisition Using FPGA

2.1. Introduction

The Smart-Catheter sensor array has multiple sensors acquiring data from the animal's brain simultaneously. Each sensor due to the nature of the parameter/analyte being measured has varied levels of signal amplitude. To cater to these, the signal needs to be conditioned to the right specifications by processing using both hardware and software interfaces. The hardware architecture of the data acquisition system designed consisted of a front-end mother board for A/D (analog to digital) conversion and a back-end National Instruments (NI). Signal acquired by the MEMS sensors is converted into serial data which is transmitted via optical cables in a lossless manner to the back-end NI system where the processing takes place. The NI system communicates with the host-computer which runs the code for the user to interactively run the data acquisition system.

2.2. The Front-end Mother Board and the back-end NI System

The front-end motherboard developed by the team in the Feinstein Institute in New York has a pressure sensor interface circuit, a flow and temperature sensor interface circuit and a headstage A/D convertor. All the components are encased in aluminium cases to shield against electromagnetic noise from the surroundings. The digital power supply consists of switching units which encodes the digital data from the sensors to be transmitted serially. The optical transmitter transmits the serial data to the back-end circuit board. The advantage of using optical communication is that the data transmission is fast, lossless and noise free. This puts it as a more

preferable medium over coaxial cables for data transfer. Figure 2.1 illustrates a picture and the schematic diagram of the circuit interface with the back-end system.

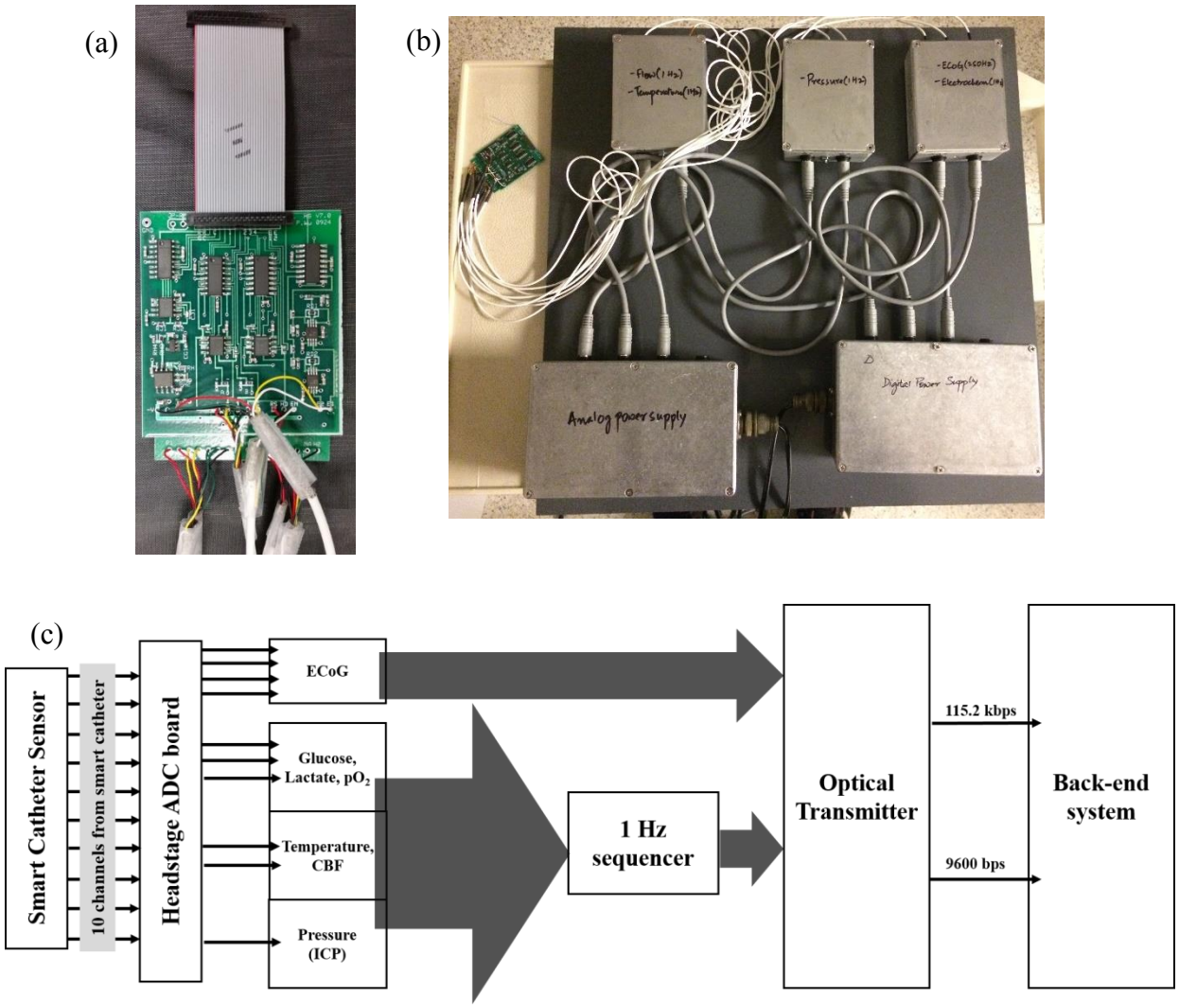


Figure 2.1. (a) Headstage A/D converter circuit board. (b) The front-end mother board for A/D conversion and transmission of serial bytes and (c) Schematic of the front-end motherboard.

The A/D convertor in the headstage board converts the analog signal from the MEMS sensors to digital serial bytes. The signal from pressure sensor, the electrochemical sensors and the flow and temperature sensors are further sampled to 1 Hz serial data. The signal from ECoG channels are transmitted at 250Hz. For the NI cRIO system to acquire the signals, it has to be transmitted as

RS232 protocol signals. This task along with routing the data bits to the correct channel is handled by the back-end circuit board shown in figure 2.2.

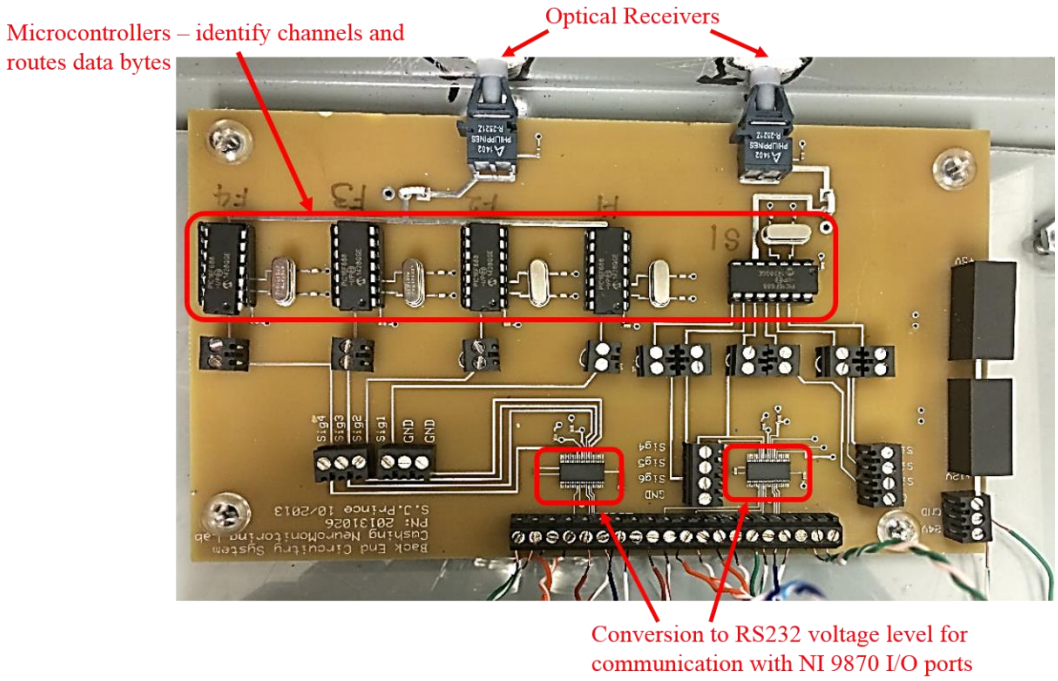
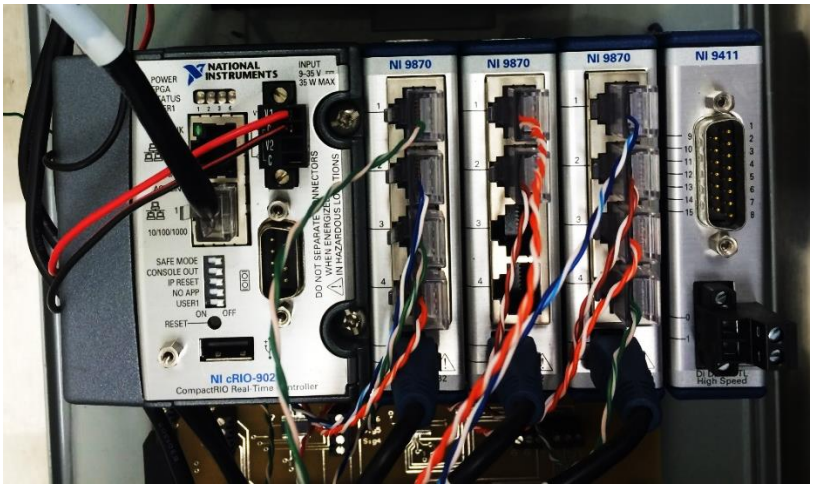


Figure 2.2. The back-end circuit board for conversion of digital serial signal to RS232 protocol signals.

2.3. The NI Compact RIO system

Data acquisition from the interface circuit and signal processing is done by a NI cRIO-9024 system. The cRIO is a data acquisition and control system. The NI cRIO 9024 has an in-built real-time controller with an 800 MHz processor which provides a highly deterministic platform for real-time computations. The cRIO-9024 is attached to an NI 9113 chassis for connecting I/O ports for acquiring data from sensors. A Xilinx Virtex-5 FPGA chip with a clock rate of 40 Hz is housed in the chassis which can be used for fixed-point data processing operations. The ability of the NI system to acquire and process data simultaneously from multiple channels parallely makes it particularly useful for our application as a real-time monitor/analyzer system.

(a)



(b)

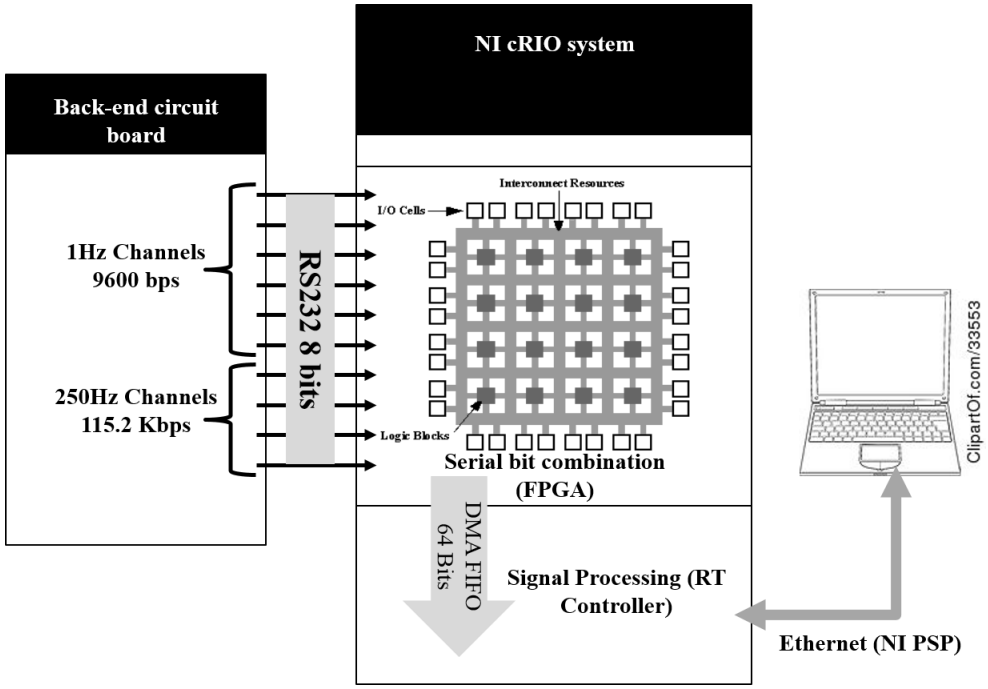


Figure 2.3 (a) NI cRIO 9024 along with NI 9113 chassis and NI 9870 serial port modules. (b) Data bits transmission from the back-end circuit board to the host computer with user interface.

The serial port I/O modules NI 9870 was mounted for acquisition of serial data from the back-end circuit board. The NI 9870 modules consist of four slots with 64 byte data buffer provided in each of the ports powered by an 8 - 28 V DC power supply. The biggest advantage of the NI cRIO system from a developer's point over conventional FPGA and real-time controllers is the simplicity in programming. Knowledge of LabVIEW programming is sufficient for developing and deploying applications in the FPGA and RT controllers in the system. Figure 2.3 shows an

illustration of the cRIO-9024 along with the NI9870 serial port modules and the scheme of transmission of serial data bits in the system

2.4. Serial Data Communication

Data transfer between digital systems can be done using either parallel or serial communication. In parallel communication, 8 bits are transmitted at a time with a dedicated routes for every channel. On the other hand, serial communication can be done with only two channels for duplex systems and a single channel for a simplex system. In our case, the system architecture has been designed as a combination of both the forms of communication. The data from the front-end motherboard is transmitted serially via two optical cables to the back-end circuit board, each dedicated to a different baud-rate. Six electrochemical and physical sensors were transmitted at 9600 bps whereas four ECoG channels were transmitted at 115.2 kbps. The back-end circuit board has microcontrollers which identifies and routes each byte to the correct channel. The data bytes are then transmitted in a parallely to the cRIO-9024 for signal processing.

2.5. FPGA Program Architecture

FPGA (Generation I) Program

Communication between FPGA and the RT controller (Host) can be established either by using programmatic front panel communication (Read/Write Control function in the host) or using first in first out (FIFO) structures with direct memory access (DMA) capability. The front panel communication method being low throughput, the DMA FIFO method was adopted, even though the rate of data transfer is slower. DMA FIFOs are large buffer structures which are mastered by the FPGA for data transfer to the real-time controller. These transfers are lossless as long as the

FIFO doesn't get filled up. To acquire the data by the FPGA, a "Method Node" with "Read Byte" method was used. A Method Node invokes an input/output method on the NI 9870 serial I/O port.

The data byte was then transmitted along with the channel number to the real-time controller for processing using DMA FIFO. Figure 2.4 shows, the method for data acquisition using

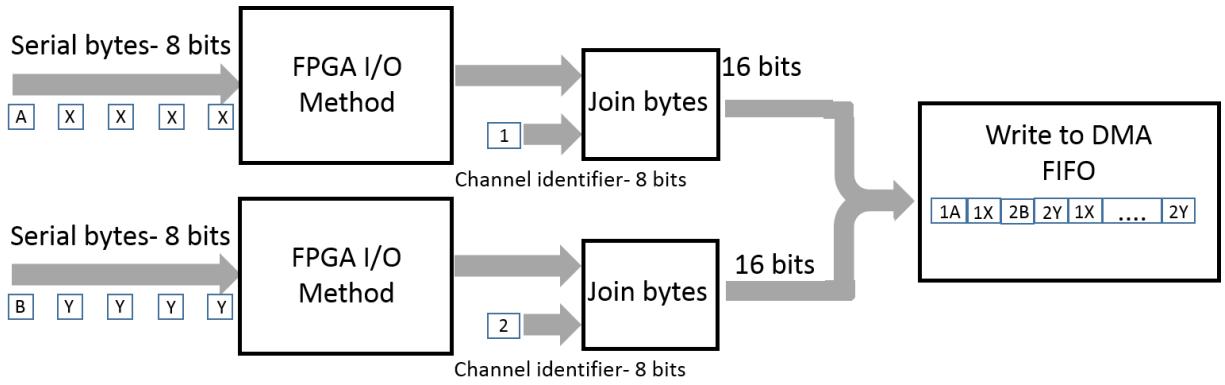


Figure 2.4. The schematic of FPGA program for data acquisition.

FPGA. The baud-rate for data acquisition for each of the ports can be set in the properties of NI 9870 modules. In our case, data was acquired from six channels at 9600 bps and from four channels at 115.2 kbps.

The data from the slow channels and the fast ECoG channels were handled differently. The slow channels are transmitted serially as ASCII (American Standard Code for Information Interchange) codes of numeric values whereas, the ECoG data are transmitted as ASCII codes of the corresponding hexadecimal (hex) representation, each byte representing an ASCII code. To convert the data byte stream to the corresponding numerical value, in both the cases the string is converted to a string array with channel identifier (delimiter) as the tab. The array functions as a FIFO and picks up the first element in string array in every loop iteration. Each data point is represented as a 20 bit integer in hexadecimal form which LabVIEW interprets as 32 bit integer. In unsigned notation, 20 bit integers have values in the range of 0 to 1048575, which in hex

translates from 00000 to FFFFF. For signed integers the range shifts from -524288 to 524287. So any unsigned integer, x above 524288 in signed notation is (x - 1048576). The subVI (function) for conversion from hex string to the corresponding signed integer is as illustrated in Figure 2.5.

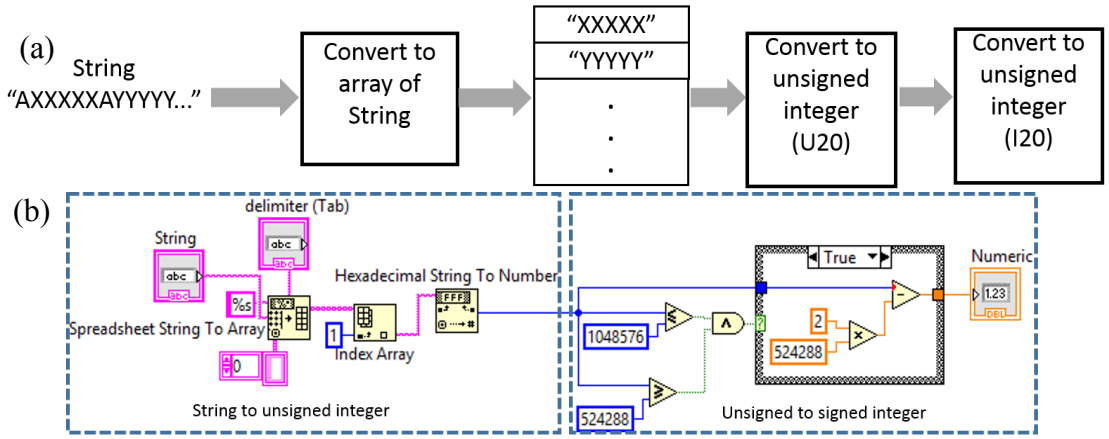


Figure 2.5. (a) Schematic of conversion of ASCII string to corresponding unsigned 32 bit integer which is converted to signed 32 bit integer and (b) LabVIEW program for implementation of the logic.

The first part of the program converts the ASCII string of hex code to the corresponding U20 (unsigned 20 bit integer). The second part of the code converts the U20 integer to I20 (signed 20 bit integer). The results of the conversion are provided in Table 2.1 for five sets of data.

The algorithm for conversion of the U20 – I20 integer is as follows:

1. If U20 < = 2^{20} AND U20 >= 2^{19}, I20 = U20 – (2^{20})

2. Else, I20 = U20

Table 2.1. Hex string to signed integer conversion

Hex String (ASCII)	Unsigned Integer	Signed Integer
80000	524288	-524288
84000	540672	-507904
A4000	671744	-376832
04000	16384	16384
FFFFF	1048575	-1

FPGA (Generation II) Program

The FPGA program architecture mentioned in the previous section suffered from a major drawback of data-loss. The serial bytes especially from the fast channel suffered huge losses during conversion to a string of data. To solve this issue, a completely new architecture for the FPGA program was developed for the faster baud-rate channels. In the new method, instead of transmitting serial bytes to the RT controller, the bytes were combined to form data-points right in the FPGA and transmitted as 64-bit signed integers. Another modification done was using dedicated DMA FIFOs for the 115.2 kbps and 9600 bps channels.

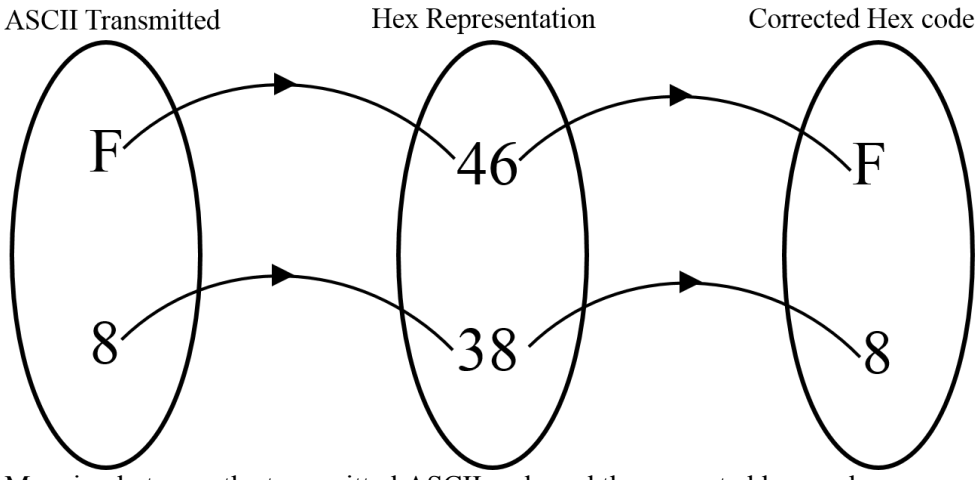


Figure 2.6. Mapping between the transmitted ASCII code and the corrected hex-code.

For the purpose of digital data transmission, each data-point was represented as a set of five hexadecimal (hex) codes. Each of the data byte was transmitted serially as a 4-bit ASCII representation of the hex code e.g, the hex number F is transmitted as an ASCII code of F, which in decimal form represents 70 (46 in hex). The figure 2.6 illustrates the mapping between the set of transmitted and the final corrected hex code used for processing. To represent the data bytes, we need codes from 0-9 and A-F in hex form. Each of the data packet was received as 8-bit ASCII code. For ASCII codes 0-9 the corresponding hex codes are 30-39 and hex codes representing A-

F are 41-46. The algorithm for the corrected Hex representation is as follows with all the numbers in hexadecimal representation:

1. If, Input >= 39 AND Input <= 20, corrected code = Input -30

2. Else if, Input >=46 AND Input <= 41, corrected code = Input -37

Table 2.2. Mapping between ASCII codes and corrected hex number.

input (ASCII)	*input (Hex)*	*input (Decimal)*	*corrected code (Decimal)*	*corrected code (Hex)*
0	30	48	0	0
1	31	49	1	1
2	32	50	2	2
. . .	. . .	. . .	. . .	. . .
9	39	57	9	9
A	41	65	10	A
B	42	66	11	B
. . .	. . .	. . .	. . .	. . .
F	46	70	15	F

A LabVIEW code was developed to convert the ASCII bytes to the corresponding hex representation (Appendix A).

An algorithm for the purpose of combining the serial bytes and conversion to decimal from hex was implemented in LabVIEW FPGA module. The code for the combination consisted of a circular buffer array which stores the hex values of the data bytes in a predefined array. Whenever, the channel identifier is read, a case-structure containing the code for combination turns to "True" and the next five data bytes are combined inside the case-structure. The data points are then converted from unsigned to signed 20 bit integers and converted to the required decimal points. The following algorithm can be used to combine the bites for transmission in FPGA:

1. Initialize an array of size 6n (n = integer).

2. Input hex data byte elements in the array.

3. Scan, the array for the channel identifier (20, 21, 22 or 23 in hex)

4. If the channel identifier is found, combine the next five bytes in the array

5. Convert unsigned integer in step 4 to signed 32 bit representation of 20 bit integer using U20 – I20 conversion algorithm mentioned earlier

6. Transmit the data to the RT controller using DMA FIFO.

A direct memory access (DMA) FIFO is then used to transmit the 64-bit data points directly to the RT controller for digital signal processing (DSP) operations (Appendix 1).

2.6. Conclusion

This chapter discussed the data acquisition done using FPGA. An FPGA has the ability to perform true parallel operations and hence multiple channels with different baud-rates could very efficiently acquire and process data from the I/O ports. The FPGA chip was by compiling a LabVIEW code which uses FPGA I/O method function for data acquisition. Two different architectures of FPGA codes were discussed. The former, due to the limitation of data loss, while combining serial bytes was avoided for faster 115.2 kbps channels. The FPGA generation II program was developed to combine the serial data in the FPGA before transmission to RT controller using a dedicated DMA FIFO which eliminated data loss during transmission. The algorithms for decrypting the correct hex codes and combination of serial bytes was presented.

Chapter 3

Real-Time Controller and Host Computer Operations

3.1 Introduction

A general purpose operating system running on a computer has the disadvantage of running multiple processes on time sharing basis. A developer using such kind of operating systems has limited or no control on setting priorities for a task e.g., a computer running a data acquisition program can prioritize running an antivirus scan at the same time, reducing the efficiency of the former program. A real-time operating system on the other hand, provides the developer with flexibility to set priority for a processes. This would solve the problem of data-loss due to other unrelated processes running parallely. The NI cRIO used in this work consists of an 800 MHz processor with 512 DDR2 RAM and 4GB of non-volatile memory, running a real-time operating system. This enables processing of data with minimum jitter from multiple channels acquiring data with different baud-rates.

The data from FPGA is transmitted serially to the RT controller using DMA FIFOs as discussed in the last chapter. Two different DMA FIFOs were used for the slower 9600 bps and 115.2 kbps channels respectively. The data points from the two different DMA FIFOs are handled differently as will be elaborated in this chapter. The data points from the RT controller after processing are transmitted to the host computer using Network Published Shared variables via Ethernet. The host computer handles the Calibration Function and the multimodal monitoring operations which constitute the user interface of the system.

3.2. Slow Channel Operations

The data bytes after transmission were read from the DMA FIFO using an Invoke Method function. Data extracted from the DMA FIFO is in the form of ASCII code of serial bytes. To form meaningful data points, the serial bytes have to be joined together in the RT controller. The data bytes associated with a particular channel was identified using a case structure. They were then combined to form a string with the first character being the channel identifier represented by ASCII codes of alphabets. Figure 3.1 shows the sequence of data flow from FPGA to RT controller for acquiring and combining data bytes.

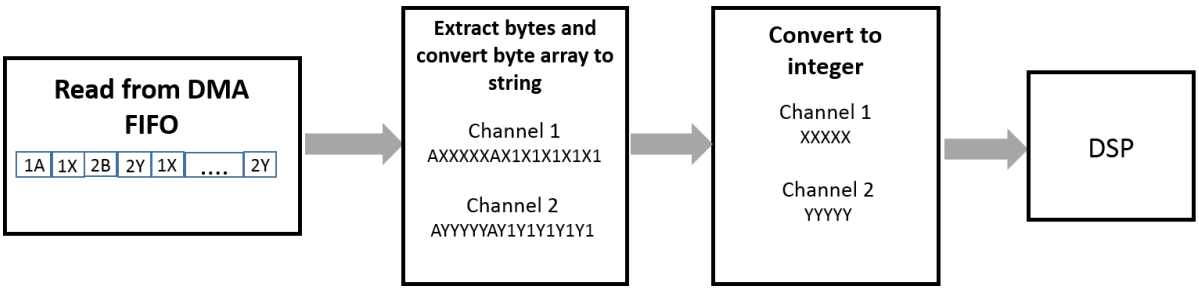


Figure 3.1. The schematic of RT VI for converting data bytes to string for processing.

The data string is in the form of AXXXXX where, A is the delimiter and Xs are numeric values. Each channel is identified using different alphabets as delimiters e.g, the channel 1 with A, channel 2 with B, etc. A simple LabVIEW code was developed which identifies the delimiter for a channel and extracts the numeric data. The code was used as a subVI which could be called for all the channels. Figure 3.2 shows the logical flow for the conversion of ASCII data string to numeric values is similar to the first part of the code shown in figure 2.5 (section 2.5). This architecture of the code was developed in the previous work and is maintained for the slower baud-rate channels.

The data communication between the real-time controller in the cRIO and the laptop with the user interface was established using Network Published Shared Variables via Ethernet. Hosted

in the RT controller, shared variables use TCP/IP underneath as the lowest level protocol for data transmission. All the slow channels are combined into an array using the "Build Array" function and transmitted using a single shared variable as shown in Figure 3.2 a.

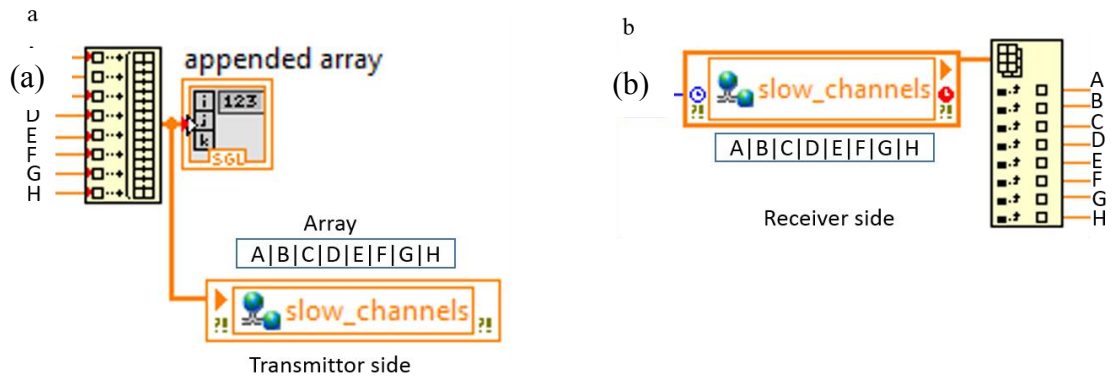


Figure 3.2. (a) Transmission of 9600 bps data channels from the RT controller to host computer using a shared variable and (b) Extracting each channel-data from the array.

The extraction of data points from the array in the host computer is done using "Index Array" function as shown in figure 3.2 b. This function returns the indexed values of the array as output. A constant value of "-1" is plugged in to the "millisecond timeout" node of the shared variable at the receiver side to enable infinite timeout. The shared variable at the receiver side will wait for an indefinite time till a data packet arrives. This immensely reduces issues of transmission data-loss due to delays in the system. The data from the slow channels are used in the calibration function as well as the multimodal monitor function for displaying the real-time data in the user interface as well as data-logging.

3.3. Real Time Digital Signal Processing of ECoG channels

The 64-bit data points are transmitted from FPGA to the RT controller using a dedicated DMA FIFO for the faster 115.2 kbps channels. Subsequently, the data from each channel is written to RT FIFO structures. RT FIFO structures are provided by LabVIEW to transmit data efficiently between VIs running in the RT controller. This communication method from DMA FIFO to the

DSP function in the RT program is fairly different from the 9600 bps channels. This very effectively eliminates the data loss problem within the RT controller for the faster ECoG channels.

The most important job handled by the real-time controller for ECoG signals is handling the DSP function for the channels. For the purpose of digital signal processing, a bandpass IIR (Infinite Impulse Response) filter was designed using LabVIEW's "Classical Filter Design Toolkit". This toolkit provides a graphical interface for designing the filter by adjusting its frequency response. This provides the user with the order of the filter which can be used in the standard digital filter VIs. An IIR filter can be expressed in the form of the equation [13],

$$y_i = \frac{1}{a_0}\left(\Sigma_{j=0}^{N_b-1} b_j x_{i-j} - \Sigma_{k=1}^{N_a-1} a_k y_{i-k}\right) \quad (1)$$

where, b_j and a_k are the sets of forward and reverse coefficients, N_b and N_a are the number of forward and reverse coefficients, x and y are inputs and outputs respectively. Forward and reverse coefficients are computed for designing an IIR filter. The transfer function for an IIR filter can be written as,

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + + b_{N_b-1} z^{-(N_b-1)}}{1 + a_1 z^{-1} + \cdots + a_{N_a-1} z^{-(N_a-1)}} \quad (2)$$

The order of the filter is directly proportional to N_a, the reverse coefficient length. Though the frequency response becomes sharper as the order increases, the stability of the filter reduces. Thus, for a certain cutoff frequency, optimal parameters of a filter needs to be computed.

LabVIEW provides digital filter functions, which can optimally compute the forward and reverse coefficients from the online data. But, the optimization process is very slow and may take several minutes for the IIR filter to reach a steady state. To solve this issue, the filter can be fed a

finite training data set to be initialized. A buffer array was created which will store the data points required for computing the IIR filter's forward and reverse coefficients. Once the coefficients are calculated, a case structure containing the array will turn to false and the VI will be initialized to filter the subsequent data points.

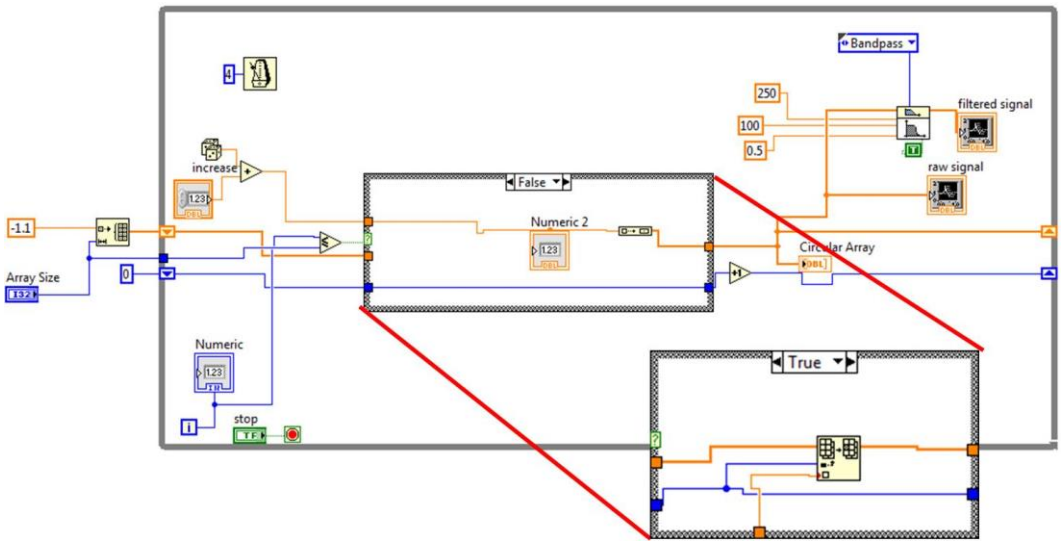


Figure 3.3. A sample code for the optimized digital filter function for bandpass filtering of data.

Figure 3.3 shows the LabVIEW code used for implementing the digital IIR filter. When the case-structure turns to "FALSE", the data points pass directly to the digital filter without further computation of IIR filter coefficients. When a signal with a DC offset is passed through a band-pass or high-pass filter, ideally the filter should attenuate DC frequency (i.e 0 Hz) and the resultant signal should be centered. This principle was used to verify the output waveform from the digital filter. An offset was created by adding a constant number to the raw signal and passed through the filter. The filter very effectively removed the DC component and centered the signal on 0. The optimal code for the filter was able to get the signal to a steady state within few seconds. Figure 3.4 shows the raw signal with a DC offset of 10 units and the filtered signal with the offset attenuate.

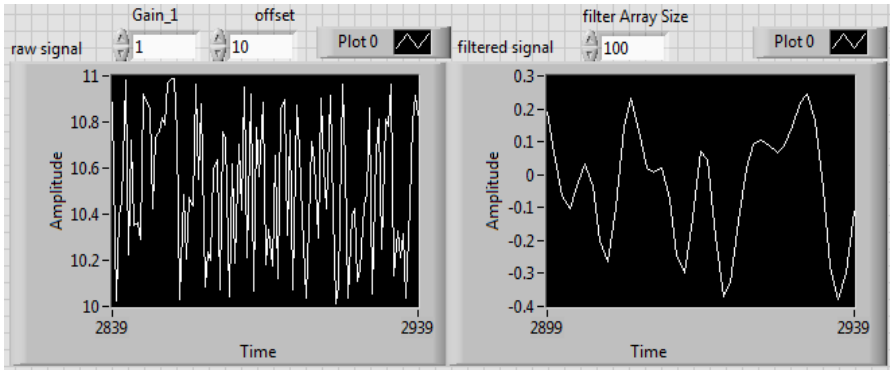


Fig 3.4. The front panel of simulated IIR filter displaying random signal and the resultant filtered signal with the offset removed (Amplitude in arbitrary units).

3.4. Calibration Function

Calibration is an extremely important aspect of accurate data recording by sensors. The functioning of sensors are carried out in a fairly linear range. The data recorded from sensors are compared with standard known values to form a calibration curve. The calibration curve can be represented by the standard straight line equation as,

$$y = mx + c \qquad (1)$$

where, x is the recorded data, y is the standard known value, m is the slope and c is the intercept of the calibration curve. The Calibration Function was developed to record multiple data points by the user and use liner regression to compute the slope and intercept of the calibration curve. The interface consists of a real-time signal display, a steady-state analyzer display and calibration curve. Figure 3.5 shows the flowchart for using the calibration function. The calibration function takes the mean of a user defined number of sampling points every time a point is recorded and plots it w.r.t the standard value of the parameter.

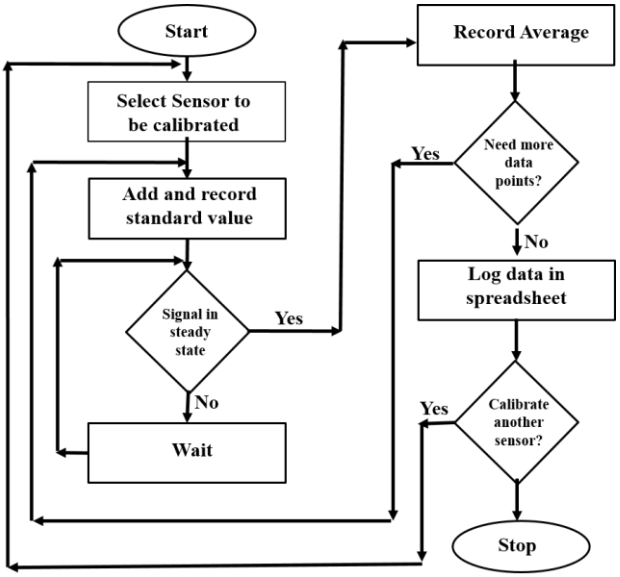


Fig 3.5. Flowchart of the calibration algorithm using a multi-point calibration method.

Fig 3.6 shows the user interface for the calibration function using Temperature Sensor signals.

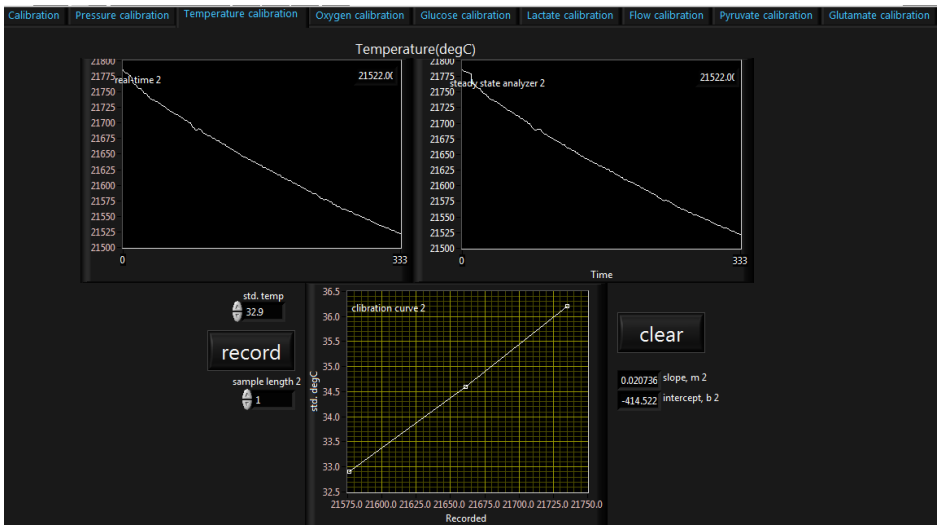


Fig 3.6. The front Panel of the Calibration Function using Temperature sensor signal.

The recorded data from the sensors is plotted on a waveform graph as the x-axis value. The standard value of the parameter is entered by the user and plotted w.r.t the y-axis. Multiple points are recorded by pressing the “record” button. The program computes the slope and intercept of the curve which is used for calibration of the sensor.

3.5. Conclusion

This chapter discussed the data transmission using Network Published Shared Variables. The shared variables use NI-PSP for communication via Ethernet and contains TCP/IP as the underlying lower level protocol. Setting the timeout for the shared variables to receive data as infinite, highly efficient data transmission between the RT controller and the host computer was achieved. To minimize the number of shared variables and hence memory overload in the RT controller, multiple channels were combined to form an array before transmission. The faster 115.2 kbps channels used RT FIFOs for data communication within the RT controller for high fidelity. Finally, successful implementation of digital filter using a method to train the IIR filter function was discussed. The learning time and hence the transient response of the IIR filter function was improved immensely by implementing a buffer array which stores a finite length training data and feeds it to the VI. Finally, the Calibration Function developed for calibration of the Smart Catheter sensors was discussed in details.

Chapter 4

Experimental Results and Discussions

4.1. Introduction

The software development for data acquisition and real-time monitoring was done in the University of Cincinnati, parallely with the development of the front end interface circuit at Feinstein Institute for Medical Research, NY. The systems were mounted on a height adjustable cart for easy portability as shown in Figure 4.1. The system was tested and verified at every stage of development using simulated signals.

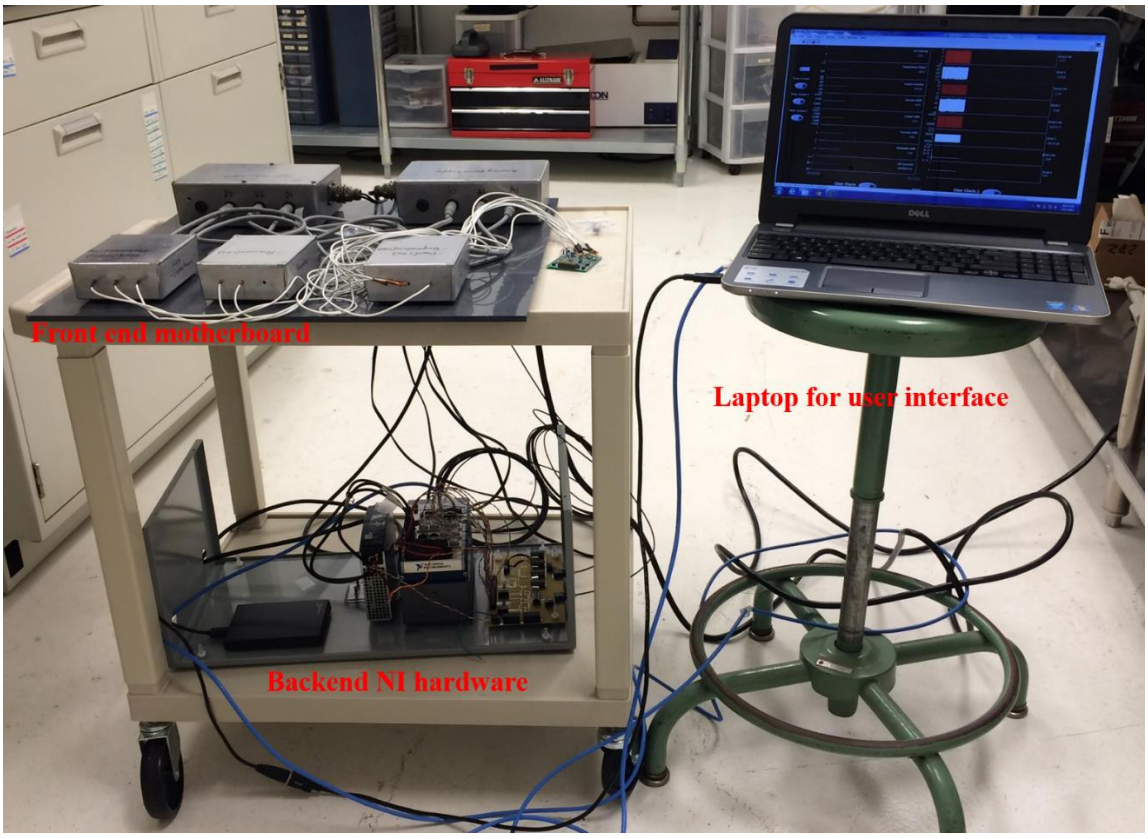


Fig 4.1. The front-end motherboard and the backend system mounted on a height adjustable cart.

Finally, the system was tested in the field and used for real-time data acquisition and monitoring from a rat's brain to analyze the effects of TBI simultaneously on multiple parameters. This chapter focusses on the experiments performed for validation of all the functions developed in the system. The experimental results for both online and offline analyses will be evaluated and discussed.

4.2. Validation of Digital Filter

A random signal with a sampling rate of 250 Hz was generated using LabVIEW to test the functionality of the digital IIR filter. The generated signal was passed through a bandpass digital IIR filter developed as discussed in Chapter 3. The data from the raw signal as well as the filtered signal were logged in to a spreadsheet file. To validate the digital filter function, power spectrum density (PSD) of the signals were plotted and analyzed. PSD is defined as the Fourier Transform of autocorrelation of a signal. LabVIEW contains a library function which accepts signals in the form of waveform data type and computes the power spectrum. The code for computation and plotting PSD on a waveform graph is as shown in Figure 4.2.

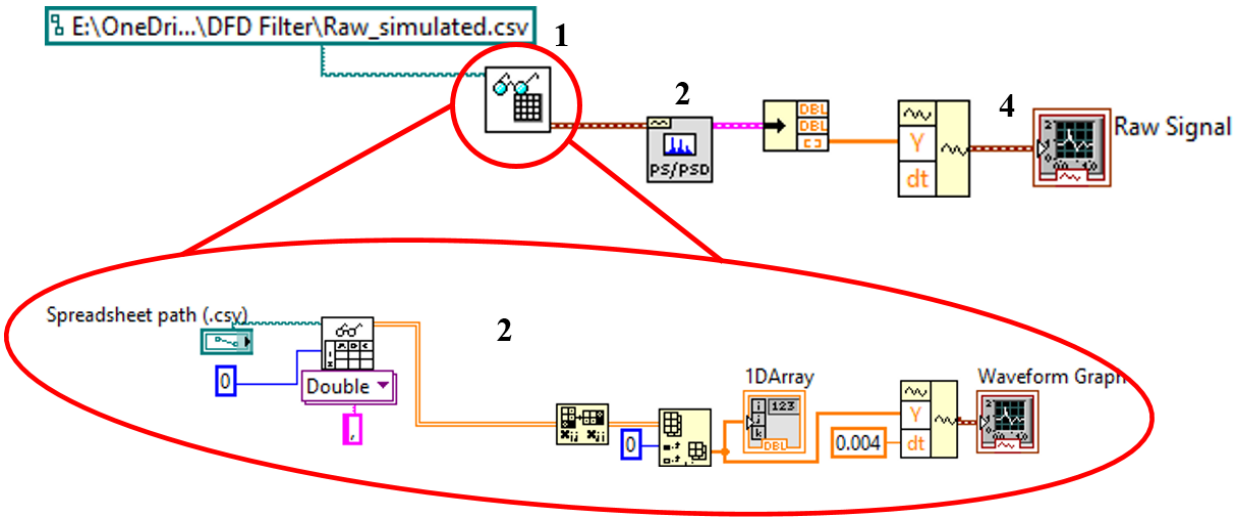
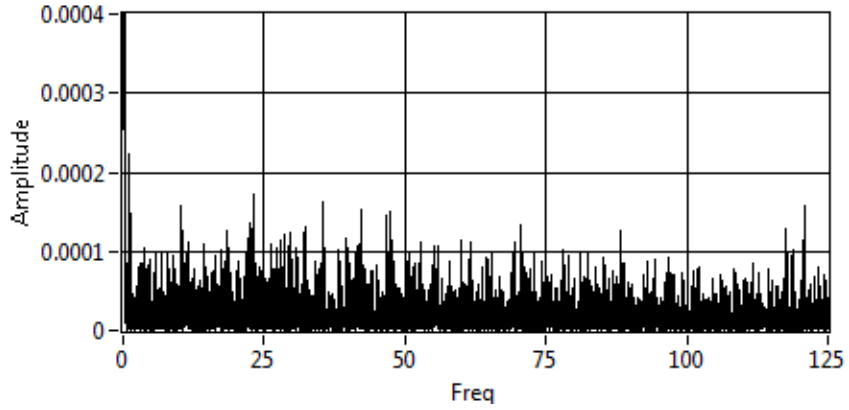


Figure 4.2. LabVIEW code for computing PSD of a signal from a spreadsheet file and plotting the data.

(a)



(b)

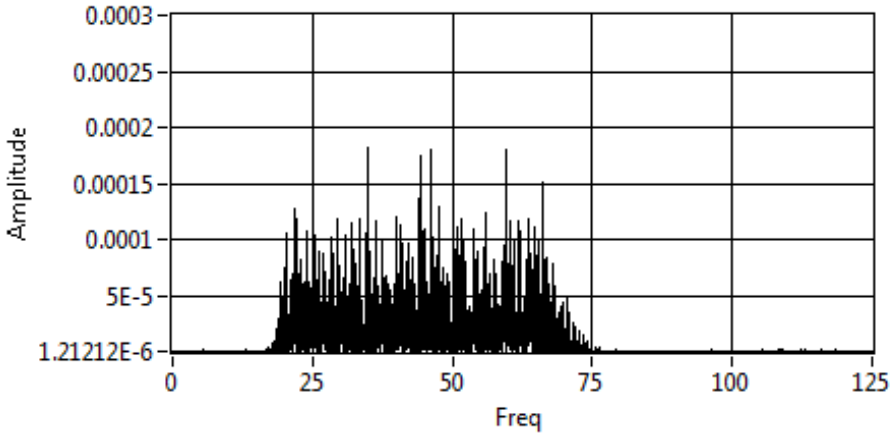


Figure 4.3. (a) The PSD of unfiltered and (b) the filtered (20-70 Hz cutoff, arbitrarily) simulated random Gaussian signal.

The code performs the following functions:

1. Read data from spreadsheet file.

2. Convert the array of double data type to waveform data-type - subVI.

3. FFT power spectrum VI computes the FFT spectrum from the waveform datatype.

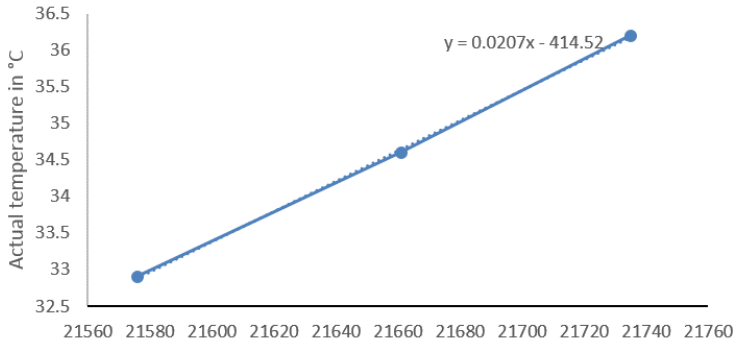
4. Plot the FFT power spectrum in a waveform graph.

The FFT spectrum of the signals were plotted in waveform graphs for offline analysis. Figure 4.3 shows the result of the experiment. The figure demonstrates a random signal with gain unity and offset of 10 units. We clearly observe attenuation of frequencies below 0.5 Hz and above 50 Hz as desired.

4.3. Validation of Calibration Function

Validation of calibration function was done using the temperature sensor due to simplicity of operation. Water in a 50 mL beaker was heated for 5-10 minutes. A commercial thermos-couple and the Smart-Cather Sensor Array were inserted in the beaker close to each other and temperatures readings were recorded as the water cooled down. The "Std. temperature" (y-axis) was noted and entered in the calibration function. The recorded data (x-axis) was plotted w.r.t the standard temperature (from thermocouple). The slope and intercept of the calibration curve was noted and used for calibration of the real-time data. Figure 4.3 (a) shows the obtained calibration curve for the temperature sensor along with the equation of a straight line. The values of the slope and intercept were entered in the "settings" tab of the multimodal monitor. The temperature of water in a beaker cooling down is showed the real-time values in °C as shown in Figure 4.3 (b).

(a)



(b)

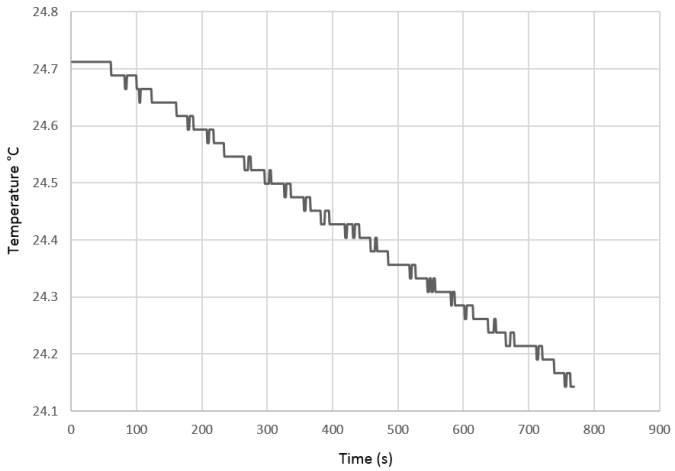


Figure 4.4. (a) The calibration curve with the equation of a straight line for temperature sensor. (b) Recorded data of temperature sensor for 769 seconds measuring temperature of water in a beaker.

4.4. Testing of the System using Signal Generator

The monitor analyzer system was tested with ten channels running simultaneously (6 slow channels ,baud rate 9600 bps and 4 fast channels, baud rate 115000 bps). Signals were generated using microcontrollers and transmitted using UART (Universal Asynchronous Receiver Transmitter) protocol. Table 4.1, shows the scheme of the channels transmitted.

Table 4.1. Scheme of channels transmitted.

Channels	Baud-rate (bps)	Sampling Rate (Hz)
ICP, Temperature, Oxygen, Glucose, Lactate, CBF	9600	1
ECoG1, ECoG2, ECoG3, ECoG4	115200	250

The FPGA chip after receiving signals parallaly processes the data points, and transmits them to the RT controller via DMA FIFOs. The controller further transmits the data points to the host computer running the user interface program using Network Published Shared Variables. Figure

Figure 4.5. Output waveforms from dummy signal sources with 6 slow channels and 4 fast channels and two channels with digital filtering enabled.

4.4 shows the output waveforms displayed in the user interface created for data monitoring and logging. All the channels were transmitted without any data loss as was verified from the real-time display and the recorded data. The blue circle in Figure 4.5 shows the temperature compensation switches, and the red circle shows the switch used to clear the channels.

4.5. Real-Time Monitoring

The hardware components including the front-end motherboard and the NI cRIO system was mounted on a height adjustable cart for portability. The MEMS smart catheter sensor array was implanted on a rat's brain and connected to the front-end circuit board. This experiment was performed at the Feinstein Institute for Medical Research, NY. The sensors were calibrated using the developed Calibration Function. The slope and intercept values were entered in the "settings" tab of the Multimodal Monitor. Also, threshold values for all the sensors can be set by the user in the settings tab. The Figure 4.6 shows an instance of real-time monitoring as a snapshot of the Multimodal Monitor screen.

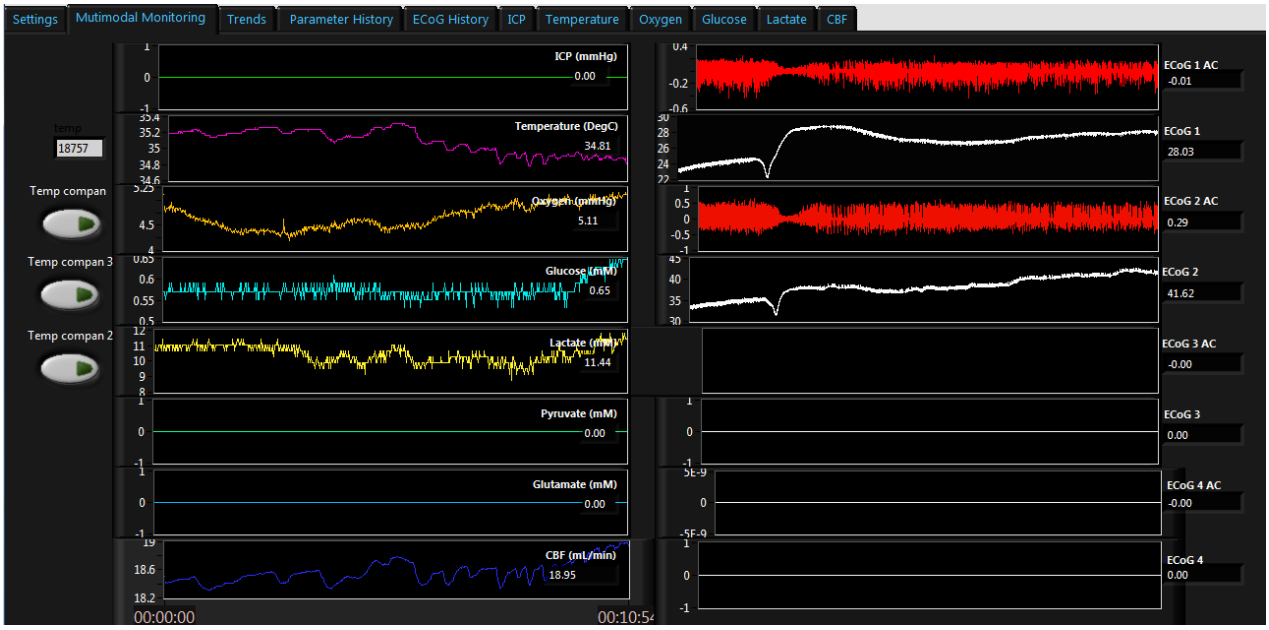


Figure 4.6. Screenshot of the multimodal monitor system for an instance of recording from a rat's brain.

The signals from the rat's brain were recorded using the Smart Catheter monitor system and logged into an external hard-drive for offline analysis. The effect of TBI is observed as a SpD wave which travels on the cortex from the surface to deeper layers. The neuronal signal recorded is passed through the bandpass filter which evidently attenuates the DC offset and clear signs of suppression of neural activity is observed. Figure 4.7 below shows the real-time ECoG signals over a 15 minutes window.

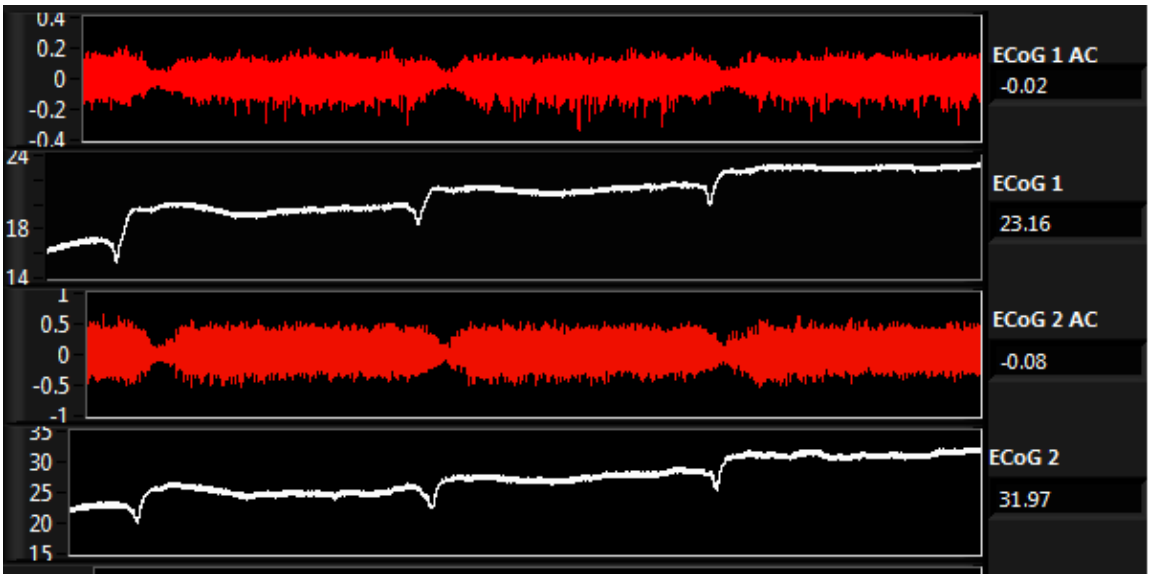


Figure 4.7. ECoG signal neural activity for 15 minutes from the smart microcatheter-tube integrated with multiple microsensors. The white signals are raw ECoG recordings with offset and red signals are bandpass filtered.

Along with the ECoG signals, other key parameters viz Temperature, pO_2 (partial pressure of Oxygen), Glucose, lactate and CBF (Cerebral Blood Flow) were also simultaneously recorded by the system. Figure 4.8 displays, trends in the temperature, Oxygen, Glucose and Lactate parameters recorded before the first SpD occurrence.

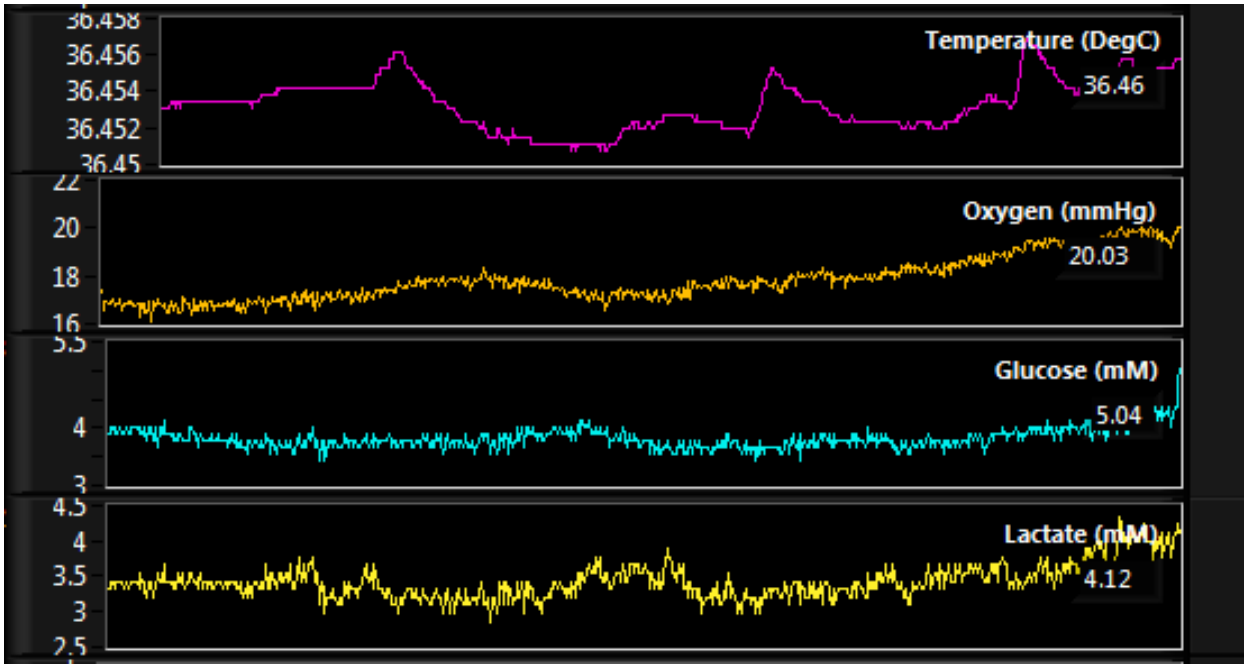


Figure 4.8. Trends observed in Temperature, pO_2, Glucose and lactate sensors monitored during the recording of SpD.

4.6. Conclusion

This chapter discussed the results using the Smart Catheter Monitor involving both *in vivo* and *in vitro* experiments. The calibration function was validated using temperature sensor and the calibration curve was obtained. The slope and intercept of the calibration curve was used to calibrate the sensors. The real-time neuromonitoring results showing the screenshot of the multimodal monitor was presented. In vivo recording of neuronal signals was done using the developed system after the Smart Catheter sensor was implanted in a rat's brain. A LabVIEW code was developed for plotting of the offline data from all the sensors. The real-time data recorded for multiple channels has been displayed demonstrating the occurrence of SpD wave due to TBI.

Chapter 5

Conclusion

5.1 Summary

In this thesis, the development of a real-time data acquisition and digital signal processing system was reported. The analog signals from sensors are converted to digital in the frontend motherboard and transmitted as serial bytes to the backend system where an NI cRIO-9024 hardware system which contains an in-built FPGA and RT controller was used for data processing. Programming for system development was done using LabVIEW for all the modules which allowed implementation of highly complex algorithms in the system. The computational capability of all the units viz, FPGA, RT controller and host computer were used for highly efficient data processing and transmission. Algorithms for combining serial data bytes before transmission was developed and compiled to be implemented in FPGA. A method of training IIR filter function in RT controller for faster response was also developed. The signal after real-time processing in the RT controller was sent to the host computer with very high efficiency using Network Published Shared Variables. The host computer handles the job of sensor calibration and real-time monitoring of all the parameters.

The Smart Catheter sensors were calibrated using the Calibration Function. The slope and intercept of the obtained calibration curves as well as the threshold values for the sensors can be controlled in the “Settings” tab of the multimodal monitor program. The system developed was tested both using signal generators and with the sensor array. After confirming the successful data acquisition and processing using signal generators, the system was used for animal test

experiment at the University of Cincinnati. The Smart Catheter sensor was implanted on a rat's brain and connected to the developed monitor/analyzer system for real-time neuromonitoring. Snippets of the results of the animal test experiment have been reported in this work. TBI caused SpD waves to travel in the animal's neocortex which was successfully recorded using ECoG sensors in the Smart Catheter. Along with ECoG signals at a sampling rate of 250 Hz, other key parameters at a sampling rate of 1 Hz were simultaneously recorded, which included, Temperature, pO_2, Glucose, Lactate and CBF. Data logging capability is also provided in the Multimodal Monitoring program which saves the data files in an external hard-drive along with a time-stamp for offline analysis.

5.2 Possibilities for Future Work

The system developed had complete data acquisition and sensor calibration capabilities. The NI hardware used can function as a complete automation solution for applications requiring precise measurement and control operations. As such, the possibilities of applying the system to various applications have limitless possibilities.

TBI is followed by brain oedema and increase of intracranial pressure. To reduce the pressure, medical professionals follow a procedure of removing CSF (Cerebrospinal Fluid) from the intracranial regions. In this regard, a system for the controlled release of CSF can be developed using the NI cRIO system. Furthermore, controlled drug delivery capability using microchannels can be developed and programmed using the system. Also, since the system can generate and output arbitrary voltage waveforms using LabVIEW, a highly effective neural recording and stimulation system can be developed using MEMS based electrodes coupled with the NI cRIO system.

References

[1] M. Abdallah, O. Elkeelany, “A Survey on Data Acquisition Systems”, International Conference on Computing, Engineering and Information, 2009.

[2] W.P. Chan, M. Je, “A Review of CMOS Multimodal Neuromonitoring Sensors and Systems”, 13th International Symposium on Integrated Circuits, 2011.

[3] C. Li, P. Wu, Z. Wu, N. Bhattacharjee, J. A. Hartings, R. K. Narayan, C. H. Ahn, “Multifunctional smart lab-on-a-tube (LOT) probe for monitoring traumatic brain injury (TBI)”, Proceedings of 60th IEEE International Electron Devices Meeting, IEDM 2014; San Francisco; United States; 15 December 2014 through 17 December 2014.

[4] Maas AI, Stocchetti N, Bullock R. “Moderate and severe traumatic brain injury in adults”, Lancet Neurol 2008; 7: 728-41.

[5] Melissa D. Carter, "Post-Traumatic Stress Disorder (PTSD) Following Traumatic Brain Injury", http://www.adlergiersch.com/post-traumatic-stress-disorder-ptsd-following-traumatic-brain-injury/ .

[6] J. A. Hartings et.al.,” Spreading depolarisations and outcome after traumatic brain injury: a prospective observational study”, The Lancet Neurology, Volume 10, No. 12, p1058–1064, December 2011.

[7] J. A. Hartings, T. Watanabe, M. R. Bullock, D. O. Okonkwo, M. Fabricius, J. Woitzik,J. P. Dreier, A. Puccio, L. A. Shutter, C. Pahl and A.J. Strong, "Spreading depolarizations have prolonged direct current shifts and are associated with poor outcome in brain trauma", Brain 2011:134, pp 1529-1540.

[8] N. Stocchetti, "Traumatic brain injury: prognostic implications of cortical electrical disturbances", The Lancet Neurology, Volume 10, No.12, December 2011, p1037–1039.

[9] H. Gomez, J. Camacho, B. Yelicich, L. Moraes, A. Biestro and C. Puppo, "Development of a multimodal monitoring platform for medical research", proceedings of Engineering in Medicine and Biology Society (EMBC), 2010, p2358 – 2361.

[10] J. A. Wilson, L.A. Shutter and J. A. Hartings, "COSBID-M3: A Platform for Multimodal Monitoring, Data Collection, and Research in Neurocritical Care", Acta Neurochirurgica Supplementum, Vol. 115, 2013, pp 67-74.

[11] A. C. Mitra and N. Banerjee, "A LabVIEW-Based Data Acquisition System in a Quarter Car Test Rig to Optimize Vehicle Suspension System", Intelligent Computing and Applications, Advances in Intelligent Systems and Computing 343, 2015, pp 593-601.

[12] CNS Multimodal Monitor, Morberg Research, Inc., Ambler, PA, 2011.

[13] B. R. Ramaswamy, "An integrated real-time system for multimodal monitoring of Traumatic Brain Injury (TBI)." Electronic Thesis or Dissertation. University of Cincinnati, 2012. OhioLINK Electronic Theses and Dissertations Center.

[14] National Instruments, "What is a Real-Time Operating System (RTOS)", Nov 22, 2013, http://www.ni.com/white-paper/3938/en/.

[15] National Instruments, "IIR Filters", June 2010, http://zone.ni.com/reference/en-XX/help/371361G-01/lvanlsconcepts/iir_filters/.

[16] Maas AI, Stocchetti N, Bullock R. “Moderate and severe traumatic brain injury in adults”, Lancet Neurol 2008; 7: 728-41.

[17] Nino Stocchetti, “Traumatic Brain Injury: Prognostic Implication and Cortical Electrical Disturbances”, www.thelancet.com/neurology, published online Nov 3, 2011.

[18] Jed A. Hartings, Michael L. Rolli, X.X. May Lu, Frank C. Tortella; “Delayed Secondary phase peri-infarct depolarizations after focal cerebral ischemia: relation to infarct growth and neuroprotection”, The Journal of Neuroscience, Dec 2003.

[19] Jens P Dreier; “The role of spreading depression, spreading depolarization and spreading ischemia in neurological diseases”, Nature Medicine Volume: 17, Pages: 439–447 Year published:(2011) Published online 07 April 2011.

[20] Matsushima K, Hogan MJ, Hakim AM, “Cortical spreading depression protects against subsequent focal cerebral ischemia in rats”, J Cerebral Blood Flow Metab 16:221–226, 1996.

[21] Jason Hinzman. Theresa Currier Thomas, Jason J, Bureister, Jorge E.Quintero, Peter Huettl, Francois Pomerleau, Greg A. Gerhardt, Jonathan Lifshitz, “Diffuse brain injury elevates tonic glutamate levels and potassium-evoked glutamate release in discrete brain regions at two days post-injury: an enzyme based microelectrode study”, 27: 889-899, Journal of Neurotrauma, May 2010.

[22] Robert Sapolsky (2005). "Biology and Human Behavior: The Neurological Origins of Individuality, 2nd edition". The Teaching Company, Pages 19 - 20.

[23] Brian S. Meldrum, “Glutamate as a neurotransmitter in the brain: review of physiology and pathology”, International Symposium on Glutamate, Oct 12-14 1998.

[24] Sarah E Hopwood, Mark C Parkin, Elizabeth L Bezzina, Martyn G Boutelle, Anthony J Strong; “Transient changes in cortical glucose and lactate levels with peri-infarct depolarizations, studied with rapid-sampling microdialysis”, 25: 391-401, Journal of Cerebral Blood Flow & Metabolism, 2005.

[25] Strong AJ, Harland SP, Mel drum BS, Whittington DJ; “The use of in vivo fluorescence image sequences to indicate the occurrence and propagation of transient focal depolarizations in cerebral ischemia”. J Cereb Blood Flow Metab 16:367–77, 1996.

Appendix A – LabVIEW Codes

Introduction

The development of the data acquisition system was done in LabVIEW environment. FPGA was used for parallel data acquisition and pre-processing for multiple channels with different baud-rates. The data is then sent to RT controller for digital signal processing which then transmits the data to the host computer using Shared Variables. In this section, important codes developed to implement the algorithms for data processing mentioned in the main thesis chapters will be presented.

Signal Acquisition

The RS-232 serial signal from the back-end circuit board is directly received using the FPGA unit of the NI cRIO-9024. The FPGA VI for signal acquisition part consists of two parts: slower 9600 bps channels and faster 115200 bps signals. The signal acquisition is done using FPGA I/O method nodes. For the slow channels, method nodes send the serial data bytes to DMA FIFOs to

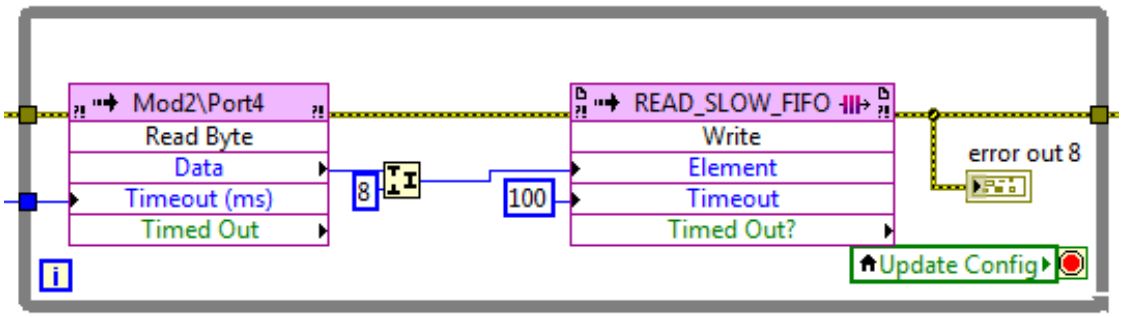


Figure A.1. Signal acquisition in 9600 bps slower channels.

be transmitted to the Real-Time controller without any processing. This is illustrated in the figure A.1.

For the faster 115200 bps channels, the serial data bytes were combined in the FPGA VI to form packets of 64 bits before transmitting to the real-time controller. This was done to prevent data loss during transmission in the DMA FIFOs. To combine the data, a code was developed which uses a circular buffer of 24 elements (each data packet is of 6 bytes). Each serial byte is transmitted as ASCII character the corresponding hexadecimal number. A sub VI as illustrated in figure A.2. was developed to convert the hexadecimal byte to the corresponding decimal U8 (Unsigned 8 bit) integer.

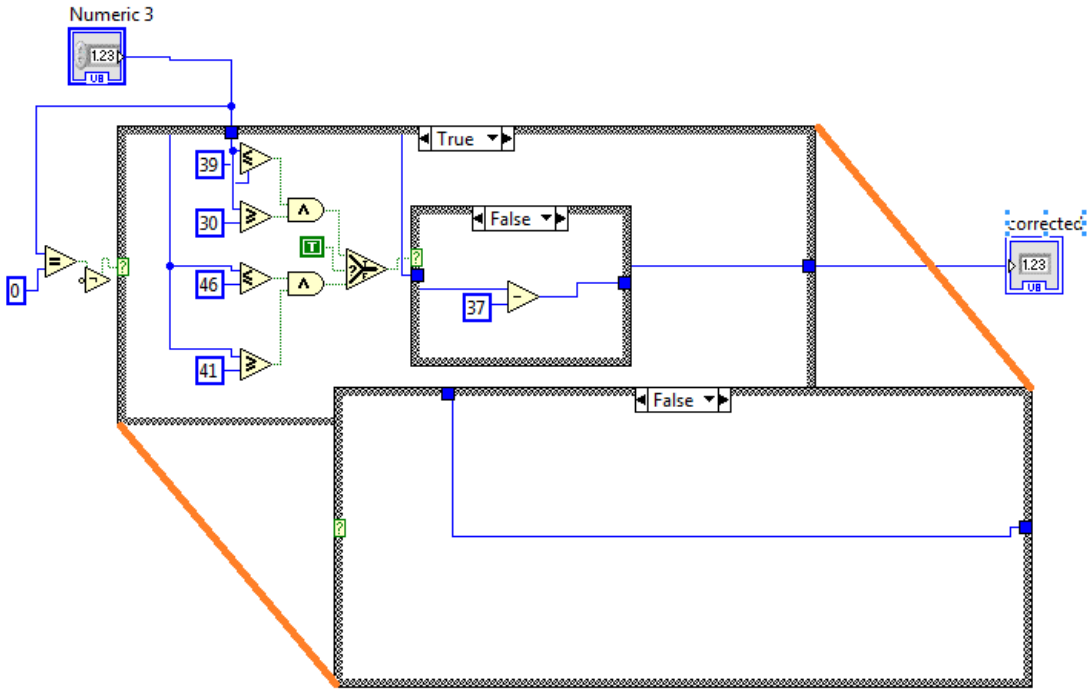


Figure A.2. Code for converting ASCII code of hex to 2'complement U8 decimal.

Next, comparison for ASCII code for the channel identifier was made. If the first element after the circular buffer is populated in the channel identifier AND the pointer is at 23 (the array locations are from 0 to 23), the case structure turns to "TRUE" and the next five bytes are combined to form a data packet (the first byte is the channel identifier).

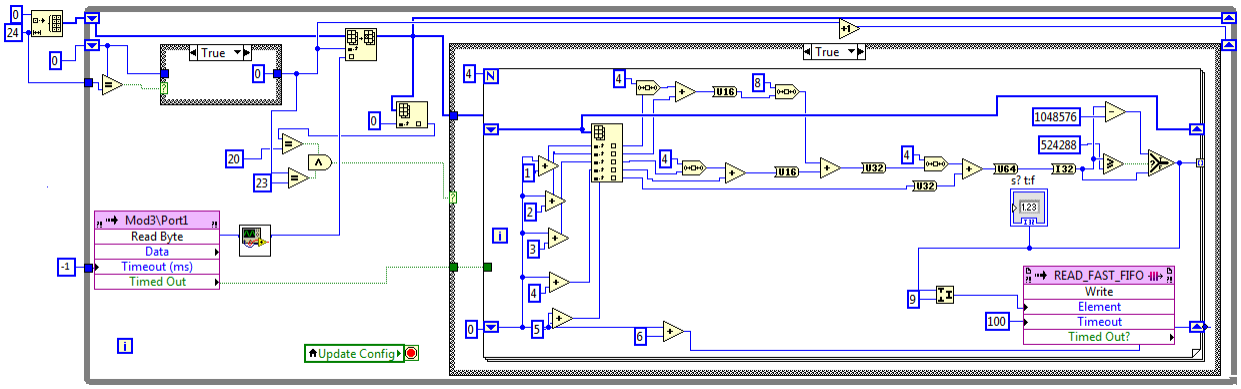


Figure A.3. Signal acquisition and data packet formation using serial data bytes in FPGA VI for faster 115.2 kbps signals.

The for loop runs four iterations to convert all the data bytes in the buffer array, five at a time to form data packets which are then converted to 2's complement I64 (64 bit signed integer) each. The I64 integer is sent to the real-time controller using DMA FIFO. Fig A.3. Illustrates the code for signal acquisition and transmission to the real-time controller. The "-1" constant plugged in to the "Timeout (ms)" makes the FPGA I/O node wait indefinitely till it receives a signal in the physical NI 9870 port assigned to it.

Real-time processing

The real-time VI (RT VI) consists of the following parts:

1. Reading the data from FPGA
2. Alarms for threshold
3. Data processing
4. Data transmission to host computer.

Calling FPGA VI by Reference

The reference of the FPGA VI is passed on to the RT VI for calling it as a function. This is done using the "Open FPGA VI Reference Function". This function uses the reference of the FPGA VI bitfile after compilation. This function can be configured very easily and run the FPGA VI from

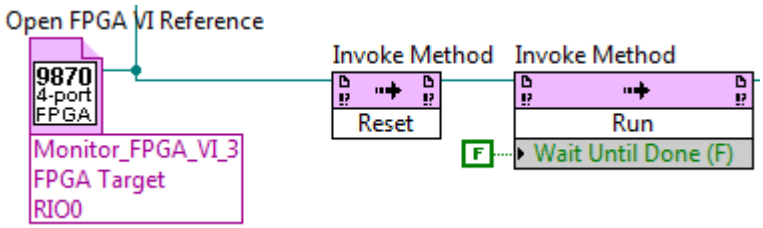


Figure A.4. Calling FPGA VI by reference to be run from the RT VI.

the caller, making it easier for the user to use the system. Figure A.4 shows the implementation of the function and passing the FPGA VI reference to "Invoke Method" function to run the FPGA VI from RT VI.

Reading data from FPGA

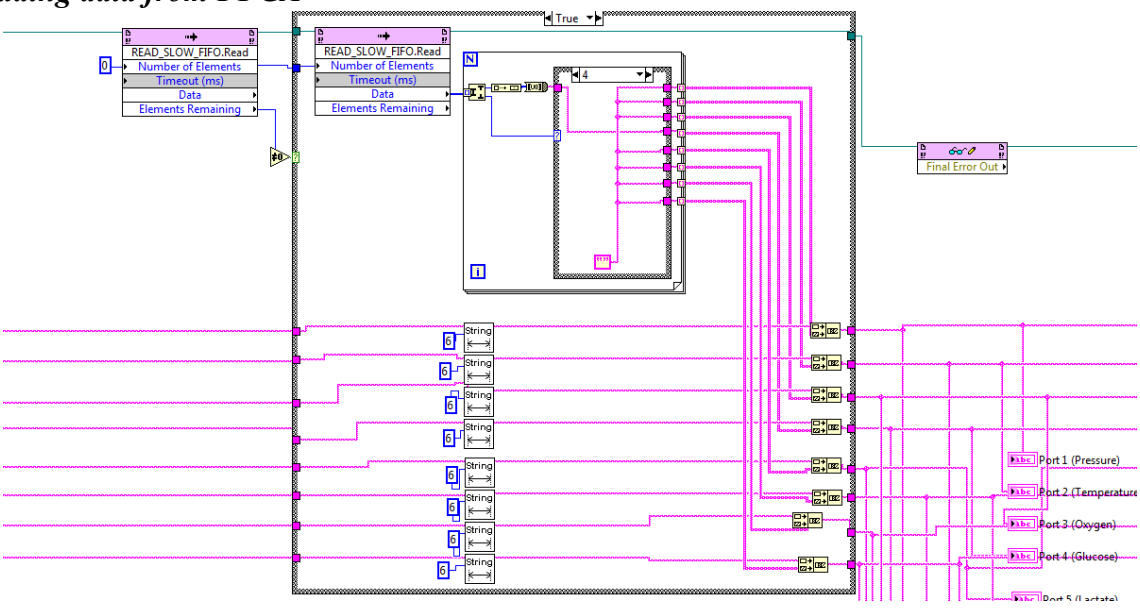


Figure A.5. Reading serial data from DMA FIFO for slower channels and combining them to from strings.

The data transmitted from the FPGA using DMA FIFOs was received in the RT VI. Two different FIFOs were used for transmission of slow and fast signals. For the slow channel, the figure A.5 illustrates the code for receiving the data using "Invoke Method" to read the FIFO.

The data was transmitted as a stream of serial data bytes from the FPGA VI, 8 bits at a time. The bytes were converted to strings of hexadecima numbers with channel identifiers (A, B, C …) in the beginning of each data string. Each character of the string was represented as the ASCII

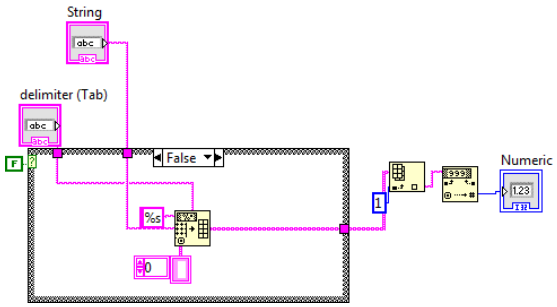


Figure A.6. Hex string to decimal extract subVI.

of the correspong number. A sub VI was created which identified the channel identifier and converted the data string to the corresponding 2's complement decimal number. The figure A.6. shows the block diagram of the sub VI.

For the faster channels a different code was implanted. The data bytes in these channels were already combined in the FPGA VI as explained earlier. They were transmitted to the RT VI as 64 bit signed integers (I64) using RT FIFOs. Figure A.7 below illustrates the code for writing the data to RT FIFOs for the fast ECoG channels.

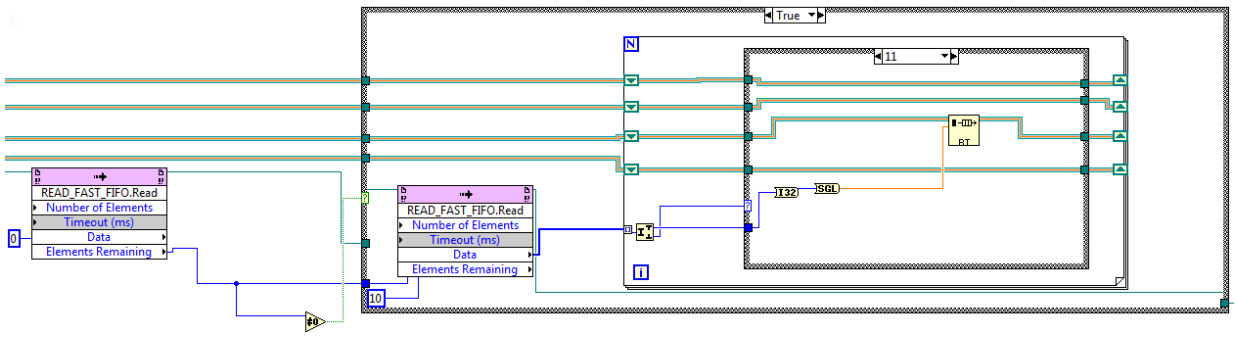


Figure A.7. Writing data from FPGA to RT FIFOs.

The data was then read from the RT FIFOs and passed through band pass digital filter which will be explained in the section on data processing.

Alarm for threshold

The alarms for the parameters were implemented using another set of subVIs which used Boolean inputs and outputs which persisted for 0.5 seconds. The code for temperature alarm is as shown in figure A.8 below as an example.

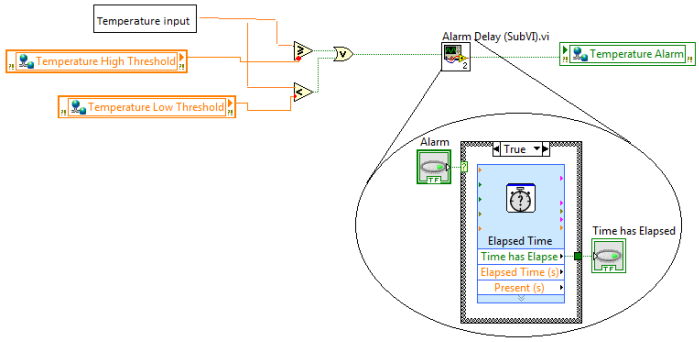


Figure A.8. Code for implementation of alarm for temperature.

Data Processing

The data from the RT FIFOs are read using the "read RT FIFO VI". The signal from the ECoG sensors is then band-pass filtered to confine the signal to 0.5 Hz to 100 Hz which is the desired band. An array of predefined size is initialized before outside the while loop. Once the code runs, the data points are replaced in the array to create a training data set for the digital IIR filter to compute the forward and reverse coefficients and initialize itself. After the training of the digital filter is over it uses the coefficients computed for the subsequent runs. The figure A.9 below shows the code for the digital filter implemented. To test the implementation of the digital filter, an offset can be created in the raw data and the output can be observed. For bandpass and high-pass filters, the DC component should be removed from the signal which implies the output signal will be centered.

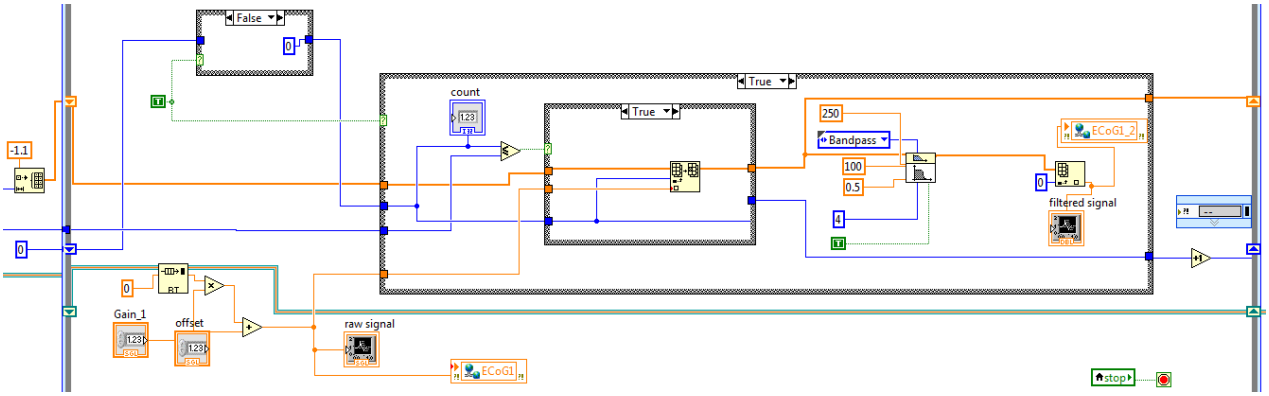


Figure A.9. Reading data from RT FIFO and digital IIR filter implementation with an intelligent code for training the filter.

Data transmission to host computer

Data is transmitted from the RT target to the host computer using network published shared variables. The shared variables are hosted on the RT target and stored in labview libraries with extension *.lvlib. Any data type can be written into a shared variable and transmitted from RT VI to the host computer and vice versa.

TPC VI- Host Computer

For memory considerations, the host VI was separated into two parts.

1. Calibration function
2. Multimodal monitor system

The slow channels which includes the physical and electrochemical sensors are transmitted to the host computer though a single shared variable to reduce the load on the shared variable engine. The faster ECoG channels have dedicated shared variable for each of them. This configuration was found to work optimally with zero data loss. Figure A.10 illustrates the data transmission between RT target and host computer.

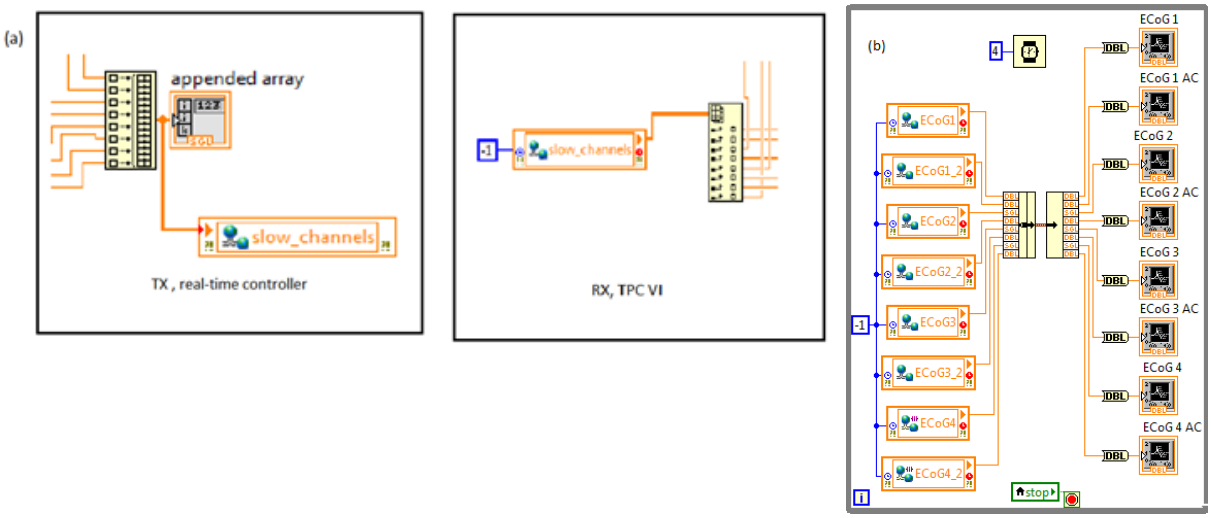


Figure A.10. (a) Transmission and receiving of slow channels using a single shared variable, (b) Enabling indefinite "ms timeout" in the ECoG channels using timeout function.

Calibration function

For the purpose of calibration, a multi-point approach was adopted. The average of a few data points corresponding to each unit of the standard analyte is recorded. The values are plotted with the standard analyte in the ordinate and the measured value in the abscissa. If x is the recorded

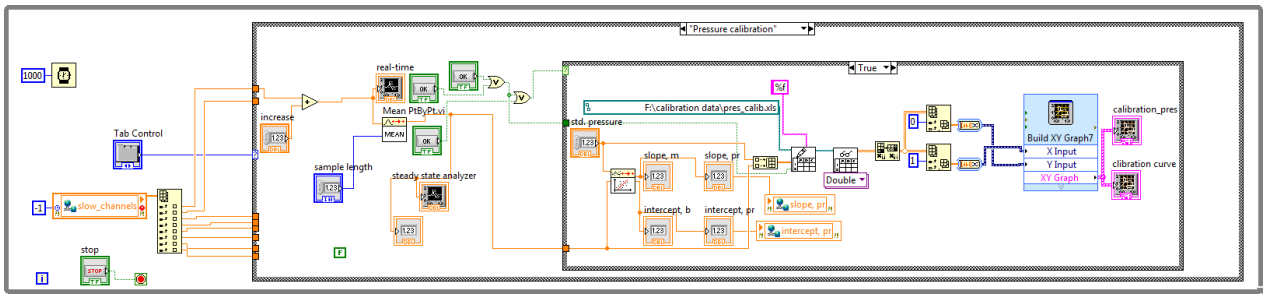


Figure A.11. Code for pressure sensor calibration and simulated signal with calibration.

signal and y being the standard concentration, then the straight line equation y = mx + b is followed to calibrate the sensor data where m and b are the slope and intercept of the sensor as computed using the calibration curve. The code for the calibration function is as illustrated in the figure A.11 below for pressure sensor as an example.

The data once recorded is logged into an excel file along with the standard unit value.

Following are the steps for operating the calibration function:

1. Run the calibration program.
2. Press “Record” to record the first data point in the calibration curve.
3. Increase the standard unit value of the analyte to be added.
4. Add the analyte and wait for the signal to stabilize looking at the “steady state analyzer”.
5. Once in steady state press the “Record” button again. This will record the second point corresponding to the second unit.
6. Keep on continuing 4 and 5 for as many data points you need.
7. The slope and intercept of the calibration curve will be computed by liner curve fitting.

The slope and intercept values should be noted from the calibration window and input them in the “Settings” tab of the multimodal monitor program.

Multimodal monitor VI

The “Settings” tab of the Multimodal Monitor program includes the settings for threshold values and the slope and intercept for the sensors. The intercept and slope of the calibration curve are incorporated into the signal from slower channels to calibrate the signals to standard concentration or unit values. Figure A.12 below illustrates the implementation of the calibration curve parameters using arithmetic operation in the LabVIEW code.

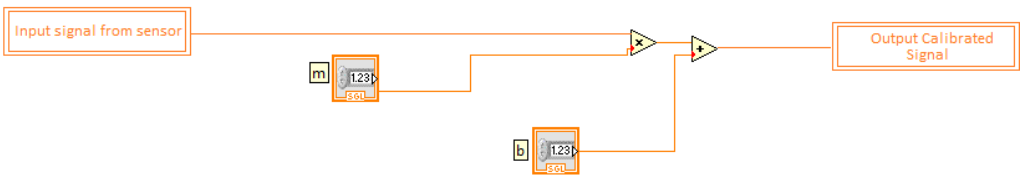


Figure A.12. Output calibrated signal using computed slope and intercept values of calibration curve.

The signal from the sensors are displayed in the multimodal monitor tab using waveform charts.

Data Logging and History

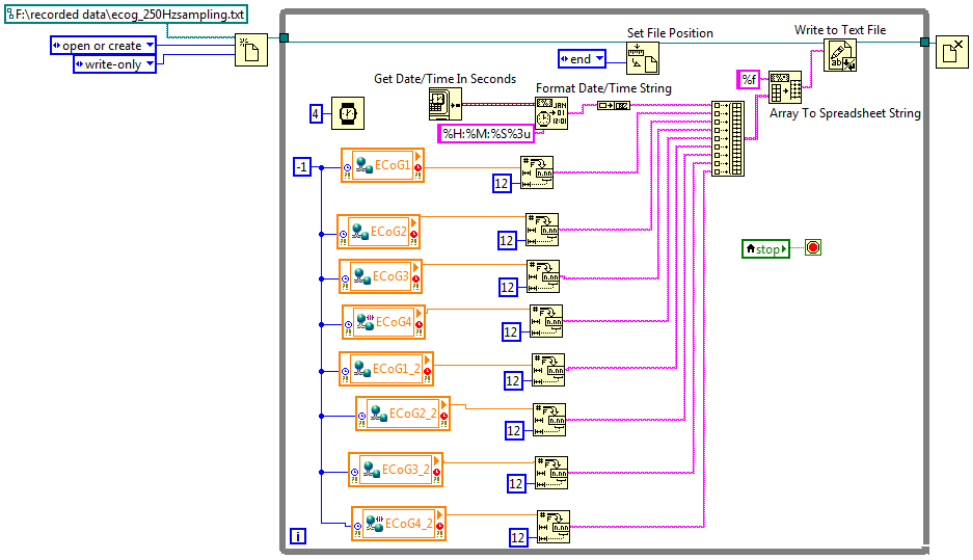


Figure A.13. Sample code used for data logging to an arbitrary URL.

Data from all the sensors are logged into spreadsheet files along with the time stamp. The assembled system along with the circuit components and the cRIO also includes an external hard drive. The data is logged into the hard drive and can be used for offline analysis. Figure A.13 illustrates the code. A similar approach is used for the slower 9600 bps channels as well. The ECoG channels were down-sampled to 1Hz and plotted along with the other sensor parameters in waveform charts. The history tabs displayed the trends of the parameters throughout the duration of the recording.

Plotting Offline Long-Term Data

The datasets recorded from *in vivo* experiments are generally huge, especially for faster ECoG channels. For files so big, specialized programs need to be developed to select and plot certain subsets of the data. A simple code for reading subsets of data as shown in A.14 was developed. The code performs the following operations:

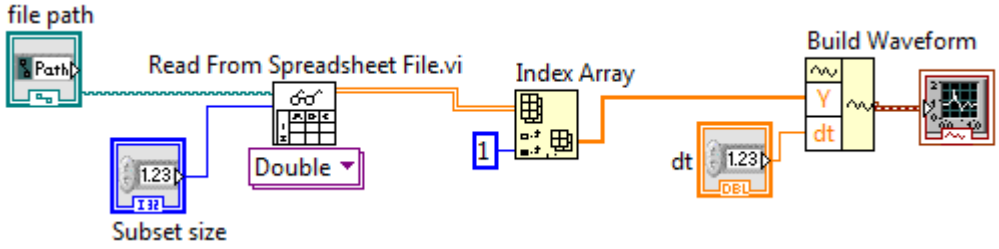
1. Reads from a spreadsheet file.

2. Selects the data array to be read from the file and forms a new subarray.

3. Using the time interval (in seconds) between each data point (“dt”), generates a waveform.

4. Plots the data in a waveform graph.

(a)



(b)

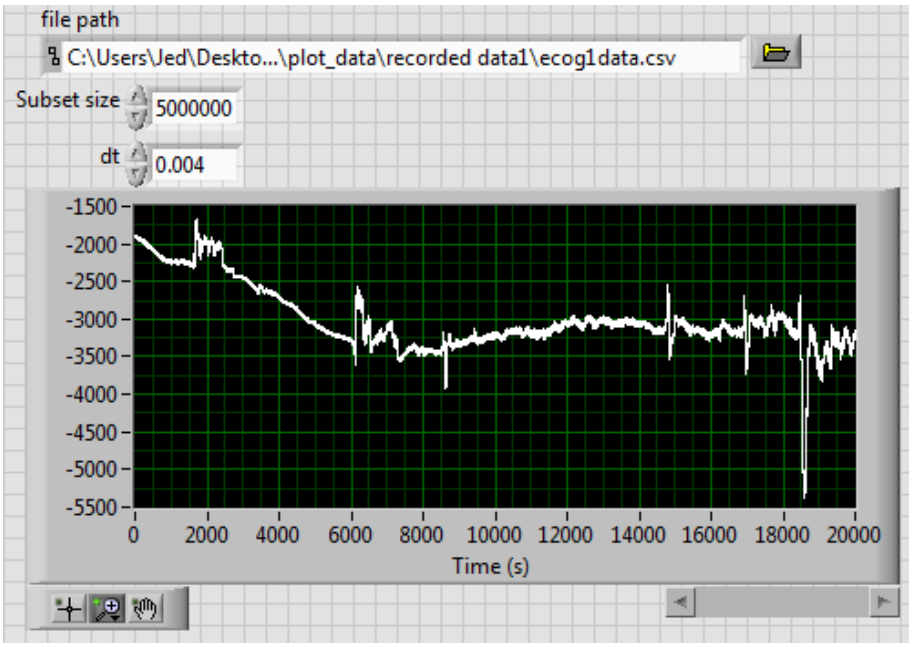


Figure A.14 a. The LabVIEW code and b. the front panel of the offline plotting function with ECoG signal recorded for 20000 seconds.

Appendix B – User Guide

Introduction

This document is meant to be used as the user guide for running the Smart Cathter Monitor program. The following steps should be regarded by the user as the general instructions for running all the codes:

1. To run all LabVIEW codes use the "➡" button at the top left corner of the window.
2. To stop all LabVIEW codes use the "◉" button third from the top left corner of the window.

Updatating IP Address of the cRIO using NI MAX

NI MAX is the measurement and automation explorer. This software can access the installed NI software and device drivers from the development computer. Once the external NI hardware is connected to the host computer, it can be accessed using NI MAX. Following are the steps for updating IP address of the cRIO 9024:

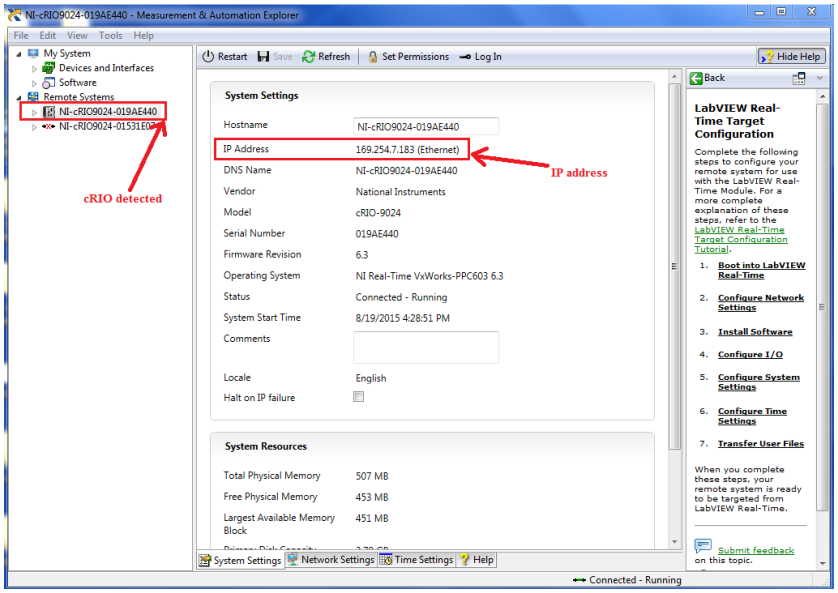


Figure B.1. NI MAX window for detection of the cRIO connected.

1. Connect the cRIO to the host computer using Ethernet cable.

2. Open NI MAX from the “Start” menu.

3. Expand “Remote Systems” >> click on “cRIO...”

4. The NI MAX window displaying the cRIO device detected is as shown in figure B.1.

5. Note down the IP address of the cRIO device.

The labview Project

Figure B.2 below illustrates the labview project used for the smart catheter monitor program.

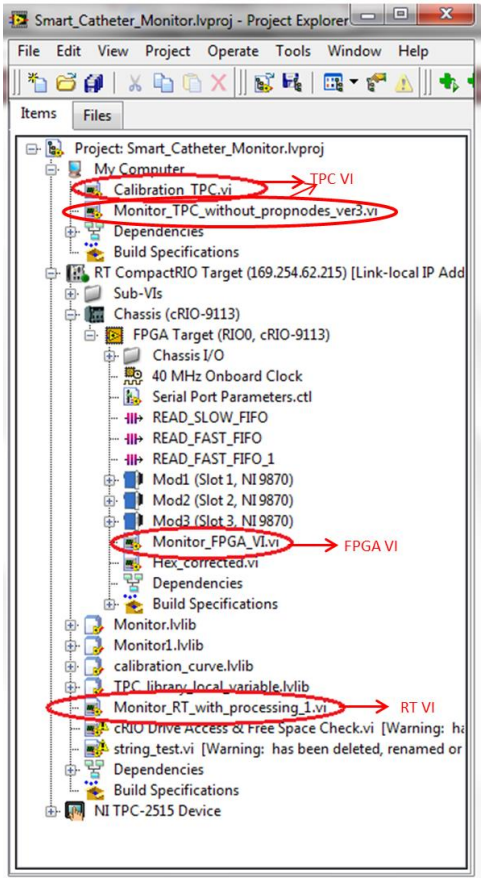


Figure B.2. The Smart Catheter Monitor LabVIEW project.

Compiling the FPGA VI

The FPGA VI needs to be compiled to generate the G-code which is written into the FPGA VI. The FPGA compile worker starts working to execute the compilation process. The compilation is done only **ONCE** whenever changes are made to the FPGA VI. Following are the steps compiling the FPGA VI:

1. Open the LabVIEW project.

2. Change the IP address of the cRIO by right-click "RT CompactRIO Target" >> select "Properties" >> select "General" >> enter the IP address on the right.

2. Expand RT CompactRIO target >> Chassis (cRIO-9113) >> FPGA Target >> open Monitor_FPGA_VI.vi

3. Run the "Monitor_FPGA_VI.vi" which opens the pop-up window in B.3

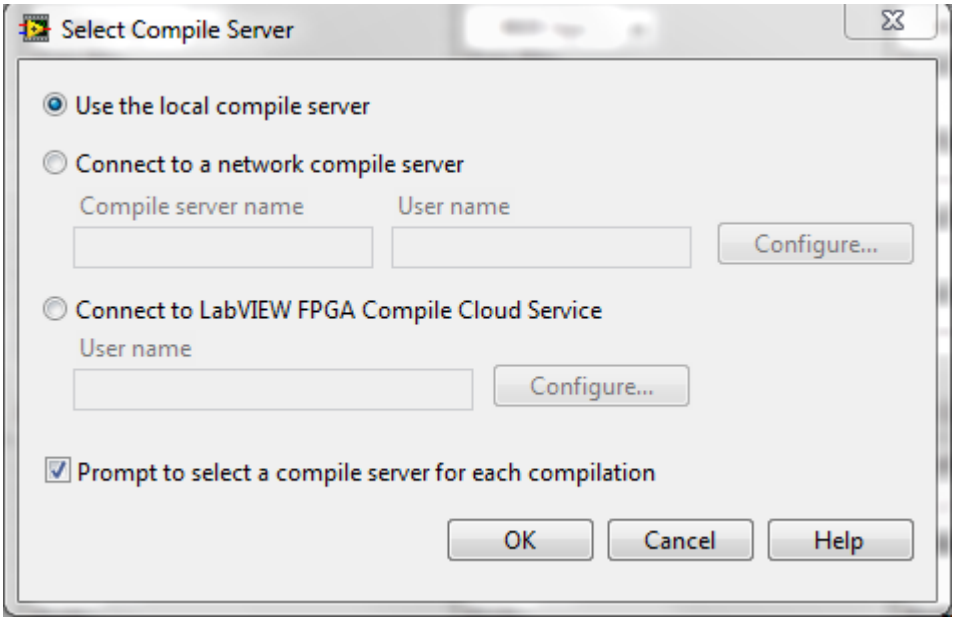


Figure B.3. Compile Server selection for FPGA VI.

4. Choose "Use the local compile server" and click "OK"

5. The compilation process is initiated which can be closed once it completes.

Instructions for running the program:

>> Open the Labview Project Smart_Cathter_Monitor.lvproj" inside the Smart Catheter Monitor folder.

1. Expand the RT CompactRIO Target.
2. Expand the Chassis > FPGA Target.
3. The FPGA code is placed in the FPGA target which deals with serial data acquisition.
4. The RT code is hosted in the RT CompactRIO target which deals with real-time data processing after acquisition.
5. The TPC code – Calibration and Monitoring VIs are hosted in the host computer.

Running the Software

1. Run the RT VI, "Monitor_RT_with_processing_1.vi".
2. Run the Calibration code "Calibration_TPC" depending on the requirement for calibration.
3. Run the monitoring VI "Monitor_TPC_without_propnodes_ver3.VI" to log the data to the external hard drive.

Calibration Function

1. Run RT VI to start data acquisition and processing

2. Run the calibration function

The figure B.4 below shows the front panel of the calibration function and an example program using simulated signals to form a calibration curve.

1. Run RT VI.
2. Run the Calibration VI.
3. Enter the standard unit of the analyte in the std.<“”> window.
4. Record the first point by pressing the record button.
5. Repeat steps 1 and 2 for desired number of data points.

The window consists of “real-time” and “steady-state” data waveform charts. The steady-state chart is formed by averaging a finite number of data points as defined by the user in the “sample length” window. The data points are also stored in the “F:\calibration data” folder in the external hard drive which the user can access.

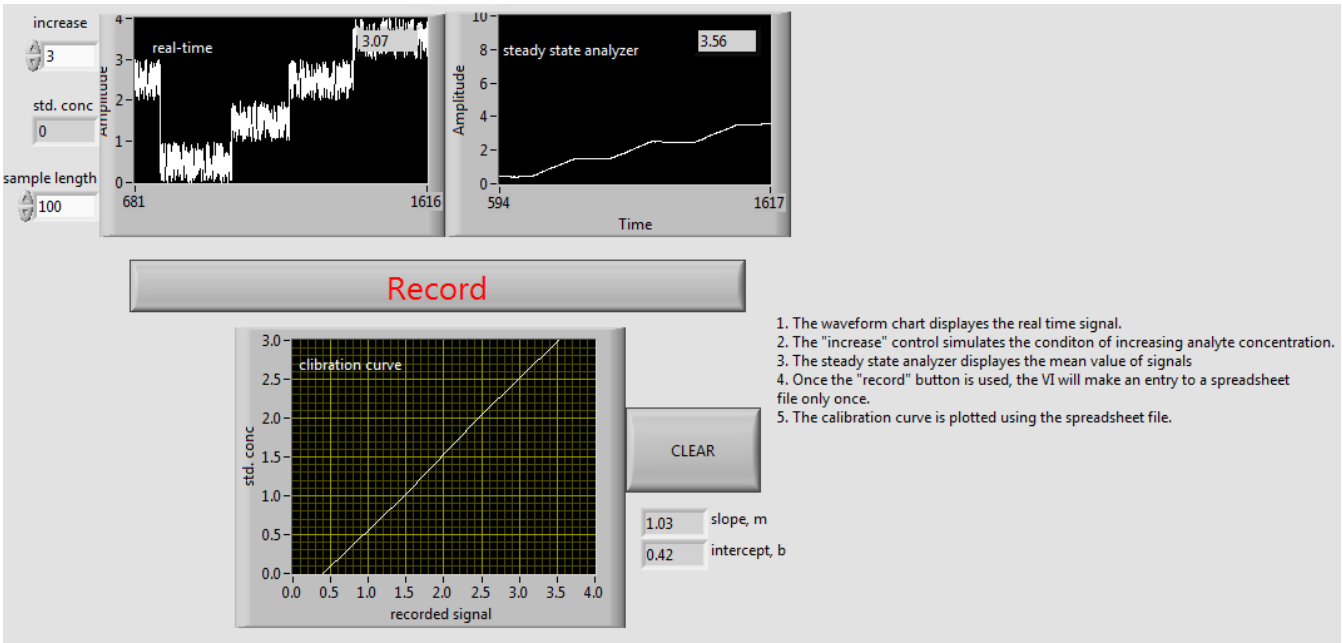


Figure B.4. Testing a version of calibration code with random signals

TPC VI- Host Computer

Settings tab

This tab has been provided for the user to enter the threshold and calibration settings. Figure B.5 below shows a screen shot of the "Settings" tab.

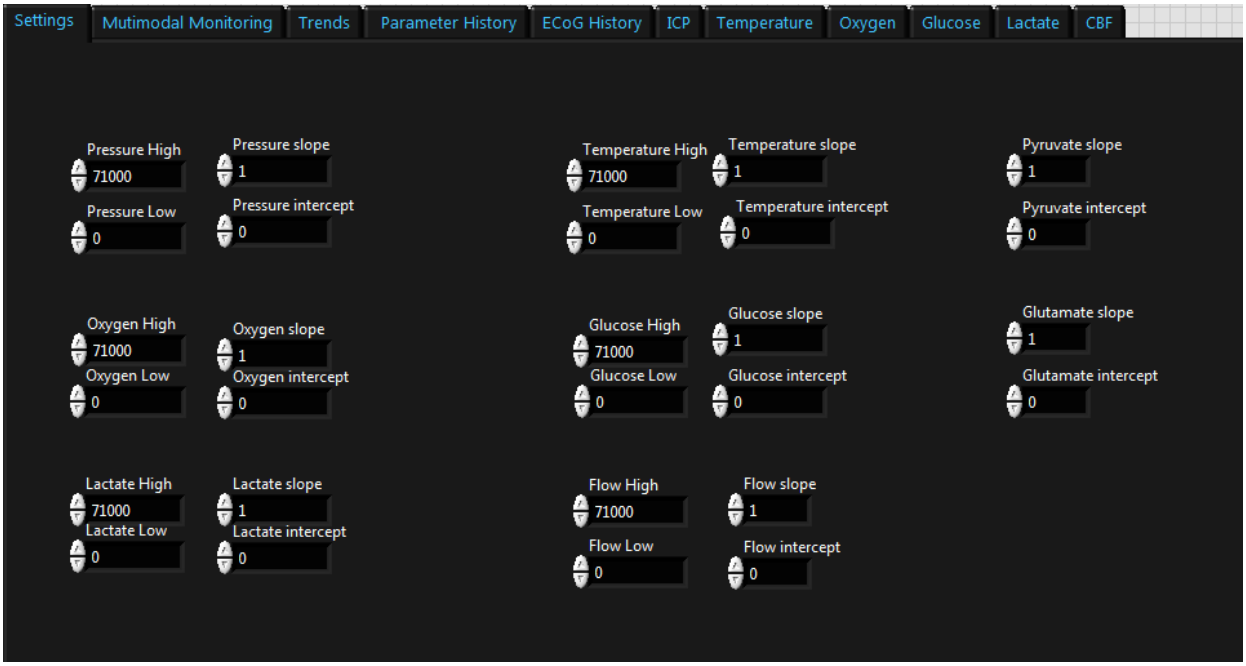


Figure B.5. Settings tab in the TPC VI to adjust the calibration and threshold settings for the channels.

Steps for settings tab:

1. Run the calibration code and get the slope and intercept values for the sensor.

2. Enter the threshold and calibration settings for the channels **before running the TPC VI**.

Multimodal monitoring tab

The “multimodal monitoring” tab displays the data from all the channels simultaneously in waveform charts. The tab also has buttons available to enable and disable temperature compensation for Oxygen, Glucose and Lactate channels. This VI does the data logging function. Figure B.4 below shows the “multimodal monitoring” tab with all the channels running simultaneously acquiring data from a dummy sensor.

Figure B.4. Multimodal monitoring window with the channels acquiring data from a dummy sensor.

Steps to run the Multimodal monitor tab:

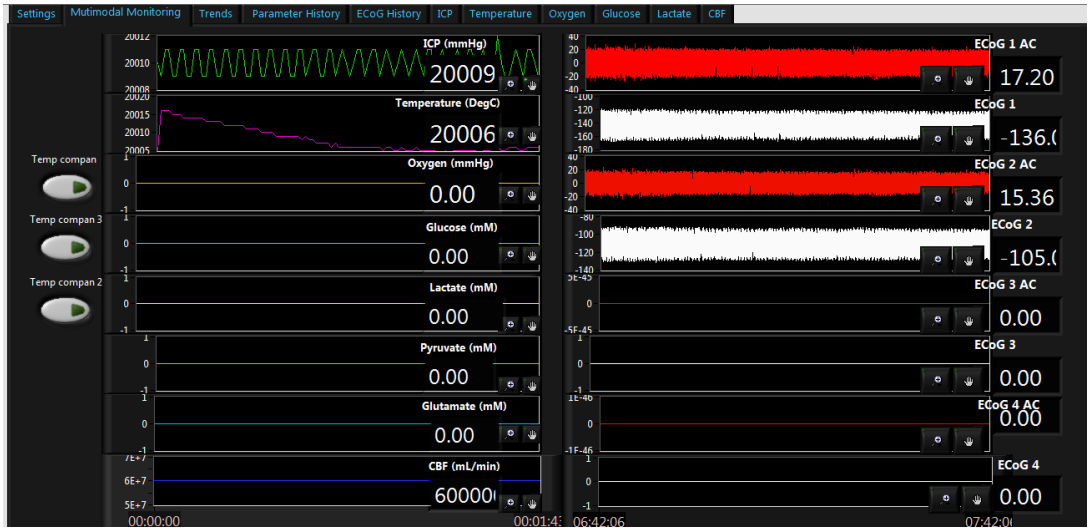


Figure B.6. Multimodal monitor front panel with real time display of simulated signals.

1. Run the RT VI
2. Run the calibration code.
3. Stop the calibration code when the desired calibration curve is obtained.
4. Enter the calibration and threshold values in the “Settings” tab of the TPC VI.
5. Run the TPC VI.