CHALMERS

eHealth Laboratory

Andrés I. Moreno

Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden, 2009

EX008/2009

Abstract

A hand on experience is important when learning technology oriented subjects. More students demand challenging tasks at the time of learning. This was the case during spring 2008, in which student surveys demonstrated the need of a laboratory for the eHealth course of the Masters program in Biomedical Engineering at Chalmers Institute of Technology.

This thesis work explores the technology, design, development and results of two health informatics experiments proposed for the eHealth course. The two experiments are a Blood Pressure (BP) monitoring application and an Electrocardiogram (ECG) monitoring application. The former application is a graphical interface for the BP monitor A&D UA-767 Plus, which is a BP measuring device that transmits measurement data via bluetooth. The graphical interface for this application was developed in LabVIEW. The latter application consists of an ECG simulator server created in LabVIEW and a Rich Internet Application (RIA) client created with Adobe Flex. The graphical interface for this application was specifically configured to allow for remote viewing of collected ECG data. A full experiment guide section with the source code is included in the work so that any person interested is capable to recreate and play with the experiments.

The discussion includes the final evaluation of the experiments, design considerations and limitations. The final evaluation discusses the relevance of the experiments into the eHealth or other similar courses. The design consideration reviews the requirements and the selection of tools and frameworks used to develop the applications. The limitations explain the graphical interface simplicity and controlled network environment scenarios used because of learning purposes. Future trends suggest expansion of the experiments to real life applications in more complex scenarios with user friendly interfaces. In conclusion, the experiments reached their functionality rendering proof that the selected tools and frameworks were appropriate for working in a laboratory environment. Furthermore individuals can obtain valuable knowledge about Information and Communication Technologies (ICT) and its growing potential in modern medicine.

Table of Contents

Acknowledgements

First I would like to express my gratitude to the people wish gave their support and confidence to make this work possible. Professor Bo Håkansson and PhD Student Anna Gund. Thank you both for the opportunity and the guidance during the development of this document. Jaakko Kerola from National Instruments, thank you for answering all my questions regarding LabVIEW, even the tricky ones. Hamid Taghavi your collaboration during the elaboration of the ECG experiment was essential. Erika Anden your cool designs put a special touch in my presentation. Classmates in Communication and Biomedical Engineering who with critical thinking and suggestions made my work better. To the S2 and Student Center staff thanks for always being there ready to help out all students.

To my roommate and one of my best friends Eduardo Epifanio you have been like a brother to me, thank you for your reliable friendship and all the good advices and laughs. To all my friends from Chalmers and Sweden, without you this experience of being abroad would have never been the same. To all my friends in Panama, whose thoughts helped along the way.

To the Afterworks, that really made my Fridays and helped keep some money in my pockets. To the Thai food stand in Kungsportplatssen the best peanut sauce I have ever tasted. To all my favorite moments and places in Sweden I will kept them forever.

To my family Mom, Dad you have been my role models, I am the man I am thanks to you. Bro, Sis you gave me the good word and support every time I needed. To my lovely friend/roommate/wife/my everything, Estela, thank you for always believing in me as a person, a professional, and a man, I always felt your love through the up and downs.

Acronyms and Abbreviations

AP	Access Point
BP	Blood Pressure
DCS	Data Communication Specification
ECG	Electrocardiogram
GPIB	General Purpose Interface Bus
ICT	Information and Communication Technologies
IDE	Integrated Developing Environment
LMP	Link Manager Protocol
L2CAP	Logical Link Control and Adaptation Protocol
MAP	Mean Arterial Pressure
MXML	Magic eXtensible Markup Language
OOP	Object Oriented Programming
PC	Personal Computer
PDU	Protocol Data Unit
RIA	Rich Internet Application
SDP	Service Discovery Protocol
SPP	Serial Port Profile
SWF	Shock Wave File
UI	User Interface
VI	Virtual Instrument
VISA	Virtual Instrument Software Architecture
VME	VERSA Module Europe
VXI	VME eXtensions for Instruments

1. Introduction

This document titled eHealth Laboratory, will describe the planning, base technology, design and deployment of two laboratory experiments intended for an eHealth course in the Biomedical Engineering Masters Program at Chalmers University of Technology. The target audience of this material is any student, professional or technician that is interested in applying Information and Communication Technologies (ICT) in the medical field for learning purposes. The document does not intent to describe any real life application or solve any specific problem, but it offers a rewarding learning experience.

1.1 eHealth and Health Informatics

The term eHealth is a recent addition to healthcare terminology; referring to electronic processes and ICT. This is a wide field which embraces many services like electronic medical records and telemedicine among others and at some point is discussed to overlap with health informatics [1]. Nevertheless, eHealth refers more than technology. Gunther Eysenbach gives us this broad definition "*eHealth is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology"*[2]. A good definition of health informatics is the one used by the School of Health Information at the University of Texas that reads *"Health Informatics combines the computer sciences and latest communication technologies with the biological and cognitive sciences to better address today's healthcare problems through better data management, analysis, and transmission"* [3]. The continuous advances in ICT within health informatics are expected to have a high impact in problem resolution in medical sciences through out the upcoming years. The experiments described in this document are technical examples which fall under the health informatics category.

1.2 The Idea of a Laboratory

During the spring semester of 2008 the idea of a laboratory emerged during the eHealth course representative's meeting. A main topic during this meeting was that the students were demanding practical concepts in the course work. Later on, during the course evaluation the survey showed that students asked for more challenging experiences, and as consequence the idea of a master thesis for creating this lab came about. This Thesis was proposed by Bo Håkansson, professor of biomedical engineering and examiner of this work, Anna Gund, PhD student in charge of the course eHealth, and Andres

Moreno, master student in communication engineering in the healthcare informatics track and author of this document.
After some brain storming, the decision was made to design two laboratory experiments in which the students were required to mix their knowledge from the Biomedical Instrumentation course with ICT. The experiments chosen were a Blood Pressure (BP) monitoring and Electrocardiogram (ECG) monitoring application. Furthermore, equipment a (BP monitor), software (LabVIEW) available in the Signal & Systems department and free licensed third party framework was used during the development. This equipment, software and framework are described in later sections of this document.

1.3 Related Work

Before developing the thesis some research was made about similar applications in order to measure the feasibility of the experiments. Looking up for related work with BP monitoring, *"Integration of Bluetooth- Enabled Sensors into an eHealth Application"* by Agustín García [4] used the same BP monitor for home healthcare and monitoring. This gave proof and confidence that the integration of the device and known framework (.Net) was possible. The ECG monitoring offered several possibilities for solution in the client side. A virtual bio-instrumentation book of LabVIEW [5] gives a good base of how LabVIEW is integrated with web clients such as Java Applets and Active X controls. The former is good reference for implementing new web client technologies like Adobe Flex, which is described in later sections of this document.

1.4 The Thesis Work

In the next chapters the technology, design, deployment, discussion, future and conclusions of this work are described. Chapter 2 presents the ICT background and developing tools needed for the laboratory experiments. Chapter 3 presents the design overview of the experiments and gives an idea of how the applications work. This is done by first giving a short description of the medical situation followed by the explanation of the application stages. Chapter 4 gives a detailed technical guide for developing the experiments; interested individuals can recreate the experiments without any major complications. Chapter 5 discusses some issues about introducing the laboratory into the learning environment and some design considerations. Chapter 6 comments future tendencies of the experiments. Finally a short conclusion of the experience is given in Chapter 7.

2. Technology Overview

This chapter reviews a few key Information and Communication Technology (ICT) concepts, programming and hardware tools necessary to develop and review the experiments in a proper manner.

2.1 Software Tools

2.1.1 Core Developing with LabVIEW

LabVIEW (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) is a visual programming language developed by National Instruments [6]. It is one of many developing programs in the market like C++ or Java. The main difference between LabVIEW and other platforms is that is not text based. Instead, it uses graphical programming commonly known as *"G"* programming [7] for developing different functional blocks in a block diagram. As other programming languages, it has the possibility of data types, objects, looping, sequence structures and error handling. The programs developed with LabVIEW are called Virtual Instruments (VIs) that is the basic unit in a block diagram. The advantage of VIs is that they can easily share information with other software or networked applications [8]. Furthermore, LabVIEW does not require vast experience in programming to get started. LabVIEW is the requested core language for developing both applications in this thesis work and Chalmers University of Technology has a very good license agreement. More information about LabVIEW can be found in [9].

2.1.2 Client Application with Flex

Flex is a set of technologies released by Adobe for developing Rich Internet Applications (RIAs). RIAs are web applications capable of running in any web browser with the same functionality of desktop applications [10]. Similar to Flash, Flex creates Shock Wave Files (SWF) rendered by a Flash Player. The programming language is directed to programmers rather than web designers, consequently it requires some knowledge of Object Oriented Programming (OOP). The development of Flex is based in a framework which uses reusable and extendable User Interfaces (UI) components within several functional services. The two core languages in Flex are Magic eXtensible Markup Language (MXML) and ActionScript. The first one is an XML based language and it is for layout and element display. The second is an ECMAScript compliant OOP language that gives logic to the application. Figure 1 illustrates the workflow of flex from the developer to the client. The client compiles the two languages in a SWF file and uploads it into a host server for the application. When the client connects to the server, it downloads the SWF and runs it in his computer. This action considerably diminishes computation load in the server. Furthermore, ActionScript allows interactivity, data handling, among other services. More information about Action

Script can be found on [12]. Flex is the programming language used to develop the client side of one of the applications.

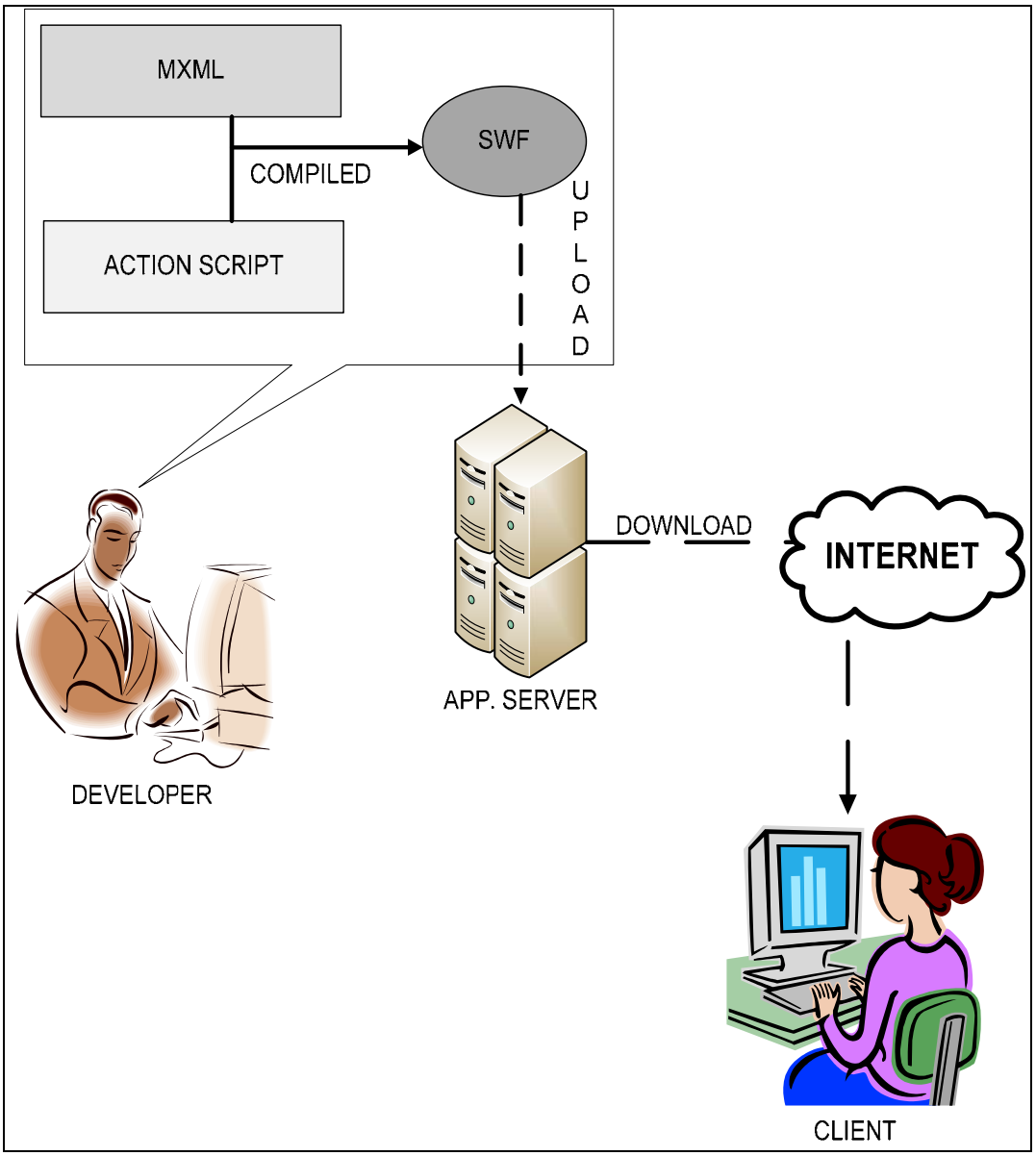

Figure 1: Flex flow diagram from developer to client from [11]

2.2 Hardware Tools

2.2.1 Blood Pressure Monitor

One of the experiments developed in this thesis work consists of Blood Pressure (BP) monitoring. For this task the A&D UA-767 PlusBT was used, a BP monitor with Bluetooth Class 1 integrated for measurement communications. To learn more about the technical specification of this device see [13]. The instruction manual can be found in [14].

2.3 Important Information and Communication Technologies Concepts

2.3.1 Bluetooth Serial Port Profile

Bluetooth is a wireless technology that provides the capability of a short range communication. It uses the 2.4 GHz band that is globally available for unlicensed low-power uses. Bluetooth is intended to provide support to endless number of wireless applications among different profiles which include data, audio, graphics video etc. [15]. The use of Bluetooth throughout this work is narrowed down to cable replacement in the A&D UA-767 PlusBT using the Serial Port Profile (SPP). The SPP sets up two virtual serial ports to allow two Bluetooth enabled devices to connect and transmit data. The SPP defines two main roles for working, device A the initiator and device B the acceptor. As seen in Figure 2, two applications that are at the top of models communicate over emulation of a serial port. The Bluetooth protocols that participate in this communication are:

- RFCOMM Protocol: It is the cable replacement protocol in Bluetooth
- Service Discovery Protocol (SDP): It is responsible to get the devices information and services from other devices.
- Link Manager Protocol: It is responsible for setting up the link between Bluetooth de devices and ongoing link management.
- Logical Link control and adaptation protocol (L2CAP): Adapts the upper layer protocols to the base band layer.
- Base band: Establishment of connections in the Bluetooth architecture.

More information regarding SPP and Bluetooth protocols can be found in [16] and [17]

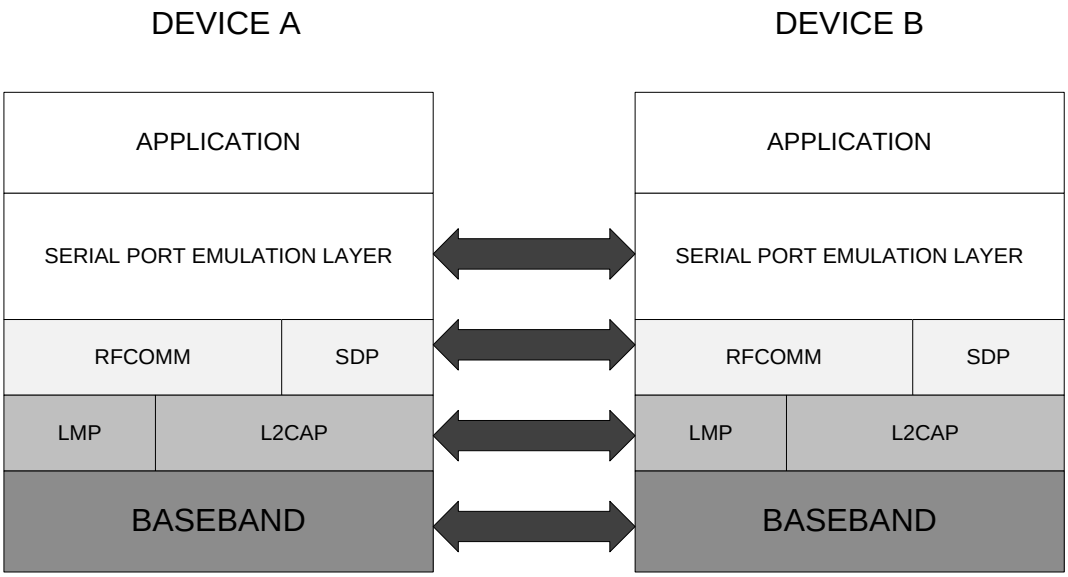

Figure 2: Bluetooth SPP protocol model from [16].

2.3.2 TCP/ IP internet server

An application server for one of the applications will be developed in the basis of Transmission Control Protocol/Internet Protocol (TCP/IP) internetworking. TCP/IP refers to the Internet Protocol suite named by its two standards Transmission Control Protocol (TCP) and Internet Protocol (IP). This is the base technology for the global internet and it is used for communication across any interconnection of networks [18]. It can be said that TCP/IP communications involve three significant parts: applications, computers and networks. File transfer and electronic mail are examples of applications. Several applications, communications and data exchange can run simultaneously in the same computer. The communication tasks of the TCP/IP protocol are divided in the following independent layers [19]:

- **Physical layer:** This layer takes all the physical interface of any device for transmitting data. This varies depending on the medium wired (Ethernet, coaxial cable etc) or wireless (Wifi, Wimax, Bluetooth etc).

- **Network access layer:** This layer takes care of the data exchange between end system and networks attached to it. Addressing issues take place in this layer and the sending device have to provide the address of the recipient in order that the network can deliver information accurately.

- **Internet layer:** The previous layer takes care of data exchange and addressing among systems contained in the same network. When the exchange and addressing is performed among different networks, this layer takes care of the routing function.

- **Transport layer:** This layer takes care of mechanism to assure the reliability of the data exchange. The TCP is the most common protocol used to provide this reliability.

- **Application layer:** This layer contains the process to support different user applications.

An interaction of the layers between two devices is illustrated in Figure 3. In the figure, the application in device A generates a data packet. Subsequently, each layer adds a header with relevant information. The relevant information contained in the headers is as follows:

- TCP header: Destination port, sequence number and checksum.
- IP header: Destination
- Network header: Destination sub network address, facilities requests.

The data packets is sent through a network and properly routed to device B. In device B the process of removing headers is performed until the application gets device A data and data exchange occurs virtually at each layer. The generic name given for a block of data exchanged at any layer is Protocol Data Unit (PDU) [21].

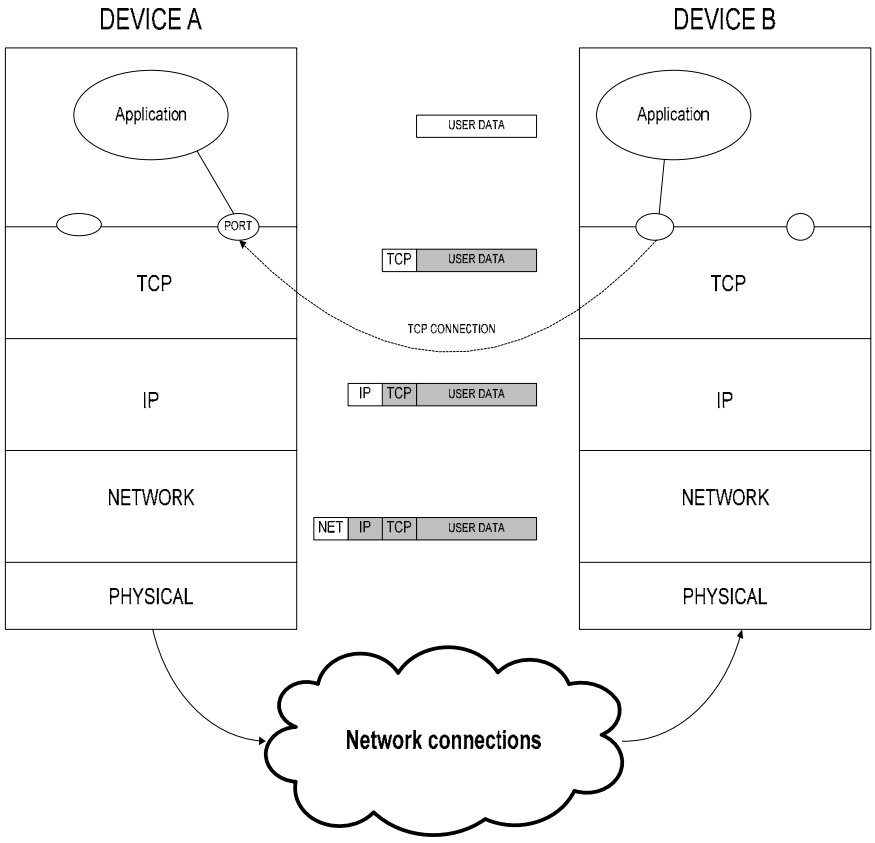

Figure 3: TCP/IP layers interaction from [20].

3. Experiments Overview

This chapter overviews the experiments design in a functional way. The idea is to give the reader a sense of the medical concept and how the applications were designed. Recommendations of which topics the students should review for a successful reproduction of the experience are provided also.

3.1 System classification

Before going into details, it is necessary to understand the kinds of applications that have been designed. Table 1 describes different kinds of networking system scenarios. The applications designed fit into Remote Monitoring perform because no major interaction occurs between the server and the client than authentication messages.

Table 1: Functional categories of networking scenarios reference [22].

Networking Scenario	Description
Remote Monitoring	A process can be monitored from another location in the network. In this process the client cannot give any feedback to the server.
Remote Control	The same as remote monitoring plus possibility of giving feedback to the server which affects the outcome.
Collaboration	Multiple users from remote locations use a client to communicate and share information with the server and each other.
Distributed Computing	Software process that runs on more than one computer to accomplish a task.

3.2 Blood Pressure Monitoring Experiment

3.2.1 What is Blood Pressure?

Blood Pressure (BP) is the force produced by circulating blood to the blood vessels and is one of the most important vital signs. It varies between systolic and diastolic pressures generated both at the beginning and the end of the cardiac cycle. A normal measure is around 110 systolic and 70 diastolic millimeters of mercury (mmHg) [23]. Furthermore, an important variable is obtained by both pressures mentioned defined medically as Mean Arterial Pressure (MAP). The MAP is the notional average blood pressure in an individual [24].

3.2.2 Design Overview

The first experiment is to construct a BP monitoring system conformed by the BP monitor A&D UA 767 PlusBT as a client and a Personal Computer (PC). The system has three stages as exposed in Figure 4:

1. ***BP measurements:*** The BP monitor user measures the systolic, diastolic, Pulse/min and MAP

2. ***Measurements communications:*** The Action of sending the measurements to a PC hosting an application by using Bluetooth Serial Port Profile (SPP).

3. ***LabVIEW application:***
    a. Serial communication: A serial communication is established with the BP monitor. This process extracts the recorded measurements and write commands according to BP monitor Data Communication Specification (DCS).
    b. Data processing: A data process based in the DCS is responsible for the graphical display.
    c. Warning system: A color warning system based in the MAP displaying lower and upper thresholds.
    d. Stacking and storage of measurements: The specifications of the BP monitor allows to store up to 40 unloaded measurements [13]. This feature provides the possibility of future use of measurements.

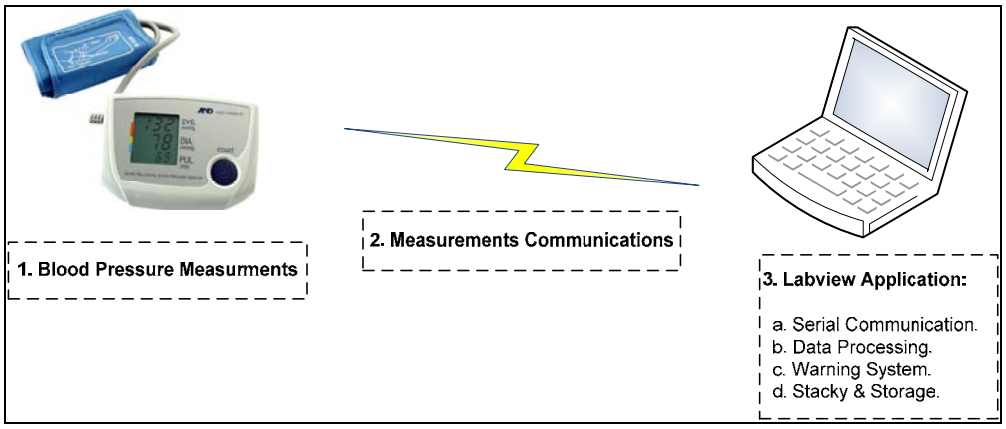

Figure 4: Experiment 1 overview

3.2.3 The Student Role

The students recreating this experiment must have basic knowledge of LabVIEW; although they are provided with an experiment guide in order to speed up the learning process. The detailed guide is available in the next section. Furthermore, the students will have to have knowledge in the following topics:

- The basics of Bluetooth communication focused in the SPP.
- The communication protocol provided by the manufacturer.
- The LabVIEW VIs mentioned in the experiment guide.

3.3 ECG Monitoring Experiment

3.3.1 What is an ECG?

An Electrocardiogram (ECG) is a recording of the electrical activity of the heart. A normal ECG signal of a single heart beat is shown Figure 5. The P wave is the atrial depolarization; the QRS complex is the ventricular depolarization and the T wave is the ventricular repolarization [25].

Figure 5: A typical ECG signal

3.3.2 Design Overview

The second experiment is to construct an ECG monitoring system composed by an ECG simulation contained in a LabVIEW server and a Flex client contained in a web browser for visualization. Ideally the experiment should be performed by two PCs and a network device (switch, router, etc) for a more real life experience. For simplicity, the experiment can be deployed in one PC as a localhost. The system has three stages as exposed in Figure 6:

1. ***LabVIEW ECG simulation server:***
    a. Simulation of an ECG signal recording: A recording of an ECG signal is simulated and repeated in a loop in order to generate an ongoing signal.
    b. Calculation of the Heart Rate: Although the simulation is repeated continuously, a calculation of the heart rate variation is possible by randomly changing the sampling frequency of the signal. As a result, it is possible to calculate different heart rates at different periods of the experiment.
    c. Creation of a data packet: Once the heart rate is calculated a data packet is created that contains the ECG amplitude and heart rate information.
    d. Publication of Packet by a Transmission Control Protocol/Internet Protocol (TCP/IP) server: After a packet is created, it is made available to the client application by a TCP/IP server waiting constantly in standby mode for an inbound connection.

2. ***Network connectivity:*** This stage represents the environment of interconnection. The use of a switch or router will suffice to simulate a controlled environment of a network. When simplicity is required it is possible to develop the whole experience in the one computer as a "localhost".

3. ***Flex TCP socket Client:***
   a. Socket connection to TCP/IP server: The client connects to the standby server by a socket connection in a specific port.
   b. Data Packet reading: Once the connection is made to the server, a reading operation of a fixed amount of bytes is executed in order to retrieve an accurate packet.
   c. Data Packet processing: After the packet is received; operations are made over it in order to retrieve the ECG amplitude and heart rate information.
   d. Data Graphical Representation: The representation of the data obtained in the packet the ECG amplitude as curved vector chart and the heart rate in a numerical display.

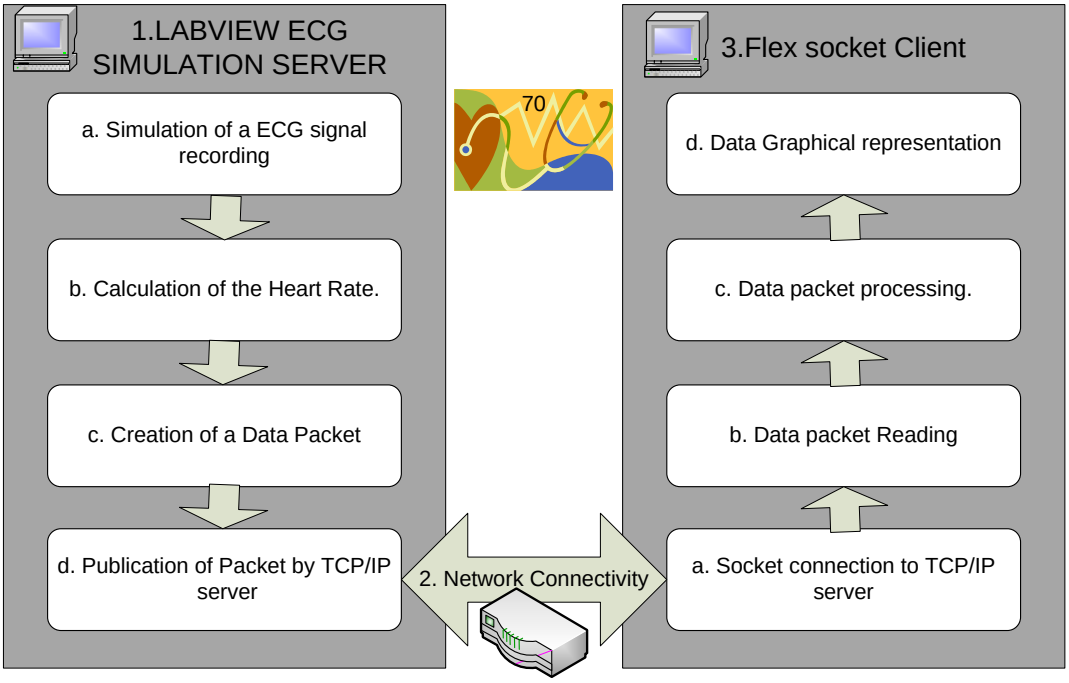

Figure 6: Experiment 2 overview

3.3.3 The Student Role:

The students developing the experiment need to have a basic knowledge of LabVIEW to develop the server application. The client is made for the students to have a taste of Object Oriented Programming (OOP). Partial knowledge of Flex is recommended for an improved experience but is not essential. A guide to speed up the learning process is included. Furthermore, the complete Flex code with extensive comments can be found in the appendix section. It is recommended that the students acquire knowledge in the following topics:

- The basics of TCP/IP internet protocols.
- The LabVIEW VIs mentioned in the experiment guide.
- Review the Flex MXML buttons and text controls [26] if the intention is to modify the client.
- Review the Flex ActionScript socket connections [27] for better understanding of the socket class designed.

4. Experiments User Guide

This chapter overviews the experiments technical design. The idea is to give the reader a detailed explanation of how the applications were designed in order to reproduce the experiments.

4.1 Blood Pressure monitoring

4.1.1 Bluetooth Communication

In this stage the A&D Blood Pressure (BP) monitor connects to the Access Point (AP) hosting the application via Bluetooth Serial Port Profile (SPP). The Bluetooth SPP creates a serial port for incoming communication from the BP monitor. In order to recreate this experiment it is necessary to have the Data Communications Specification (DCS) of the BP monitor. The DCS is confidential and only available after signing a special release form; although some fragments of this document will be referenced to understand the procedures. Some parts in the LabVIEW code are blurred due confidentiality reasons. In order to create this virtual channel between the device and the machine these steps are needed:

- Ensure that the Bluetooth device in the AP is currently working.
- Perform a measurement with the BP monitor; the BP monitor will try to establish communication with the nearest AP as in Figure 7.
- To establish communication a password will be requested. This password is available in the DCS of the BP as shown in Figure 8.
- The installation is finalized and communication is enabled between the BP monitor and the AP as in Figure 9.
- Now that the communication is initiated it is necessary to create the virtual serial port. For this is necessary to add an incoming com port in the COM PORTS section of the Bluetooth device in the AP.

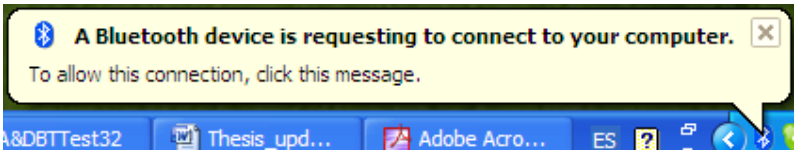

Figure 7: BP requesting communication

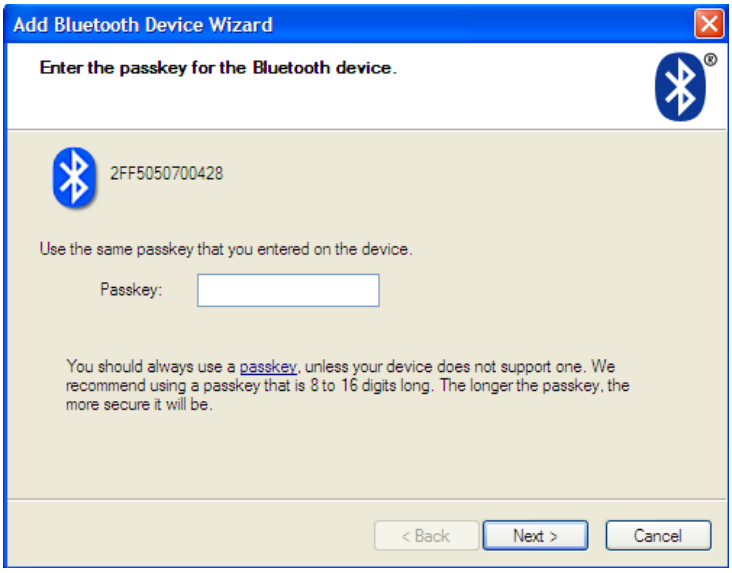

Figure 8: Password request

Figure 9: The communication successfully established

4.1.2 Serial Communication

Once the serial port is created, the serial communication stage in the application is constantly listening for incoming measurements bytes from the BP monitor at the specified port. If a valid message is received (70 bytes message), the application sends back an acknowledgement message to the BP monitor. This message communicates to the BP monitor to disregard the measurements from its database. With the help of the Virtual Instrument Software Architecture (VISA) palette of LabVIEW the serial communications are simplified. This library of functions simplifies communicating with General Purpose Interface Bus (GPIB), serial, VME eXtensions for Instrumentations (VXI), and computer-based instruments [28]. More detailed information of VISA can be found in [29]. To accomplish this stage the following steps are needed:

Construction of the Block Diagram: First go to LabVIEW block diagram and insert the VIs from the different Palettes interconnecting them as shown in Figure 10 and 11 and creating the controls referenced. The VIs are:

- VISA palette: VISA configures serial port, read, break, write and close VI's.
- Structures palette: Case structure VI.
- Comparison palette: Empty string/path VI.
- Boolean palette: Not VI.

VISA configuration: The configuration of the port session is made to establish read and write operation. It is necessary to define parameters like enable termination character, timeout, port name, baud rate, data bits, parity, stop bits and flow control to the *VISA configures serial port VI*.

Reading measurements from port: Once the desired port is configured, (the same one created during the Bluetooth communication stage) it is necessary to read the measurements from the port if they are available. Taking as reference the DCS of the BP monitor; the whole message received from the BP monitor should be 70 bytes. From this message, 60 are from the header and 10 are measurement data. Finally, a constant of 70 bytes is determined in the *VISA read VI* byte count parameter.

Sending acknowledgement back to the BP: The acknowledgement process begins as soon as a 70 byte message is received. *The comparison VI and the Boolean VI* sets the condition to writing an acknowledgement message. If the message received is 70 bytes, the condition is set to true as in figure 10. Afterwards, the *break VI* makes a pause of 250ms and a message in Hexadecimal is sent by the *write VI* to state data acceptance. All of the previously described process was made according to the DCS. If there is no 70 byte message the state is set to false as in Figure 11 and no further action is taken.

Closing the connection: After all reading and acknowledgement actions have been performed, the connection is closed by the *VISA close VI*. At this point the application

only has the ability to receive messages in a non-readable hexadecimal form and acknowledge them back. The resulting front panel is visualized in Figure 12.

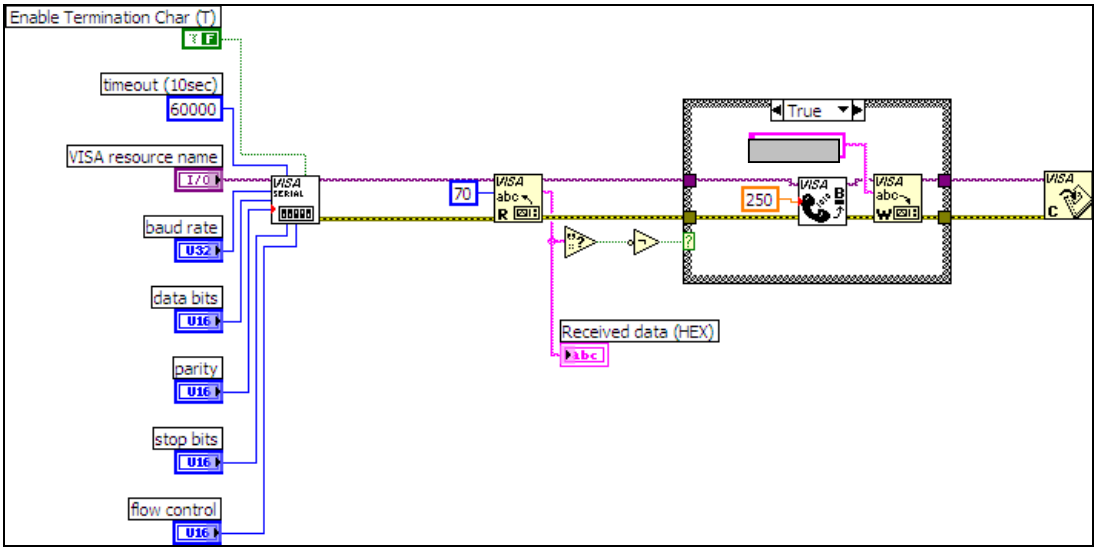

Figure 10: Interconnection showing "true" case structure.

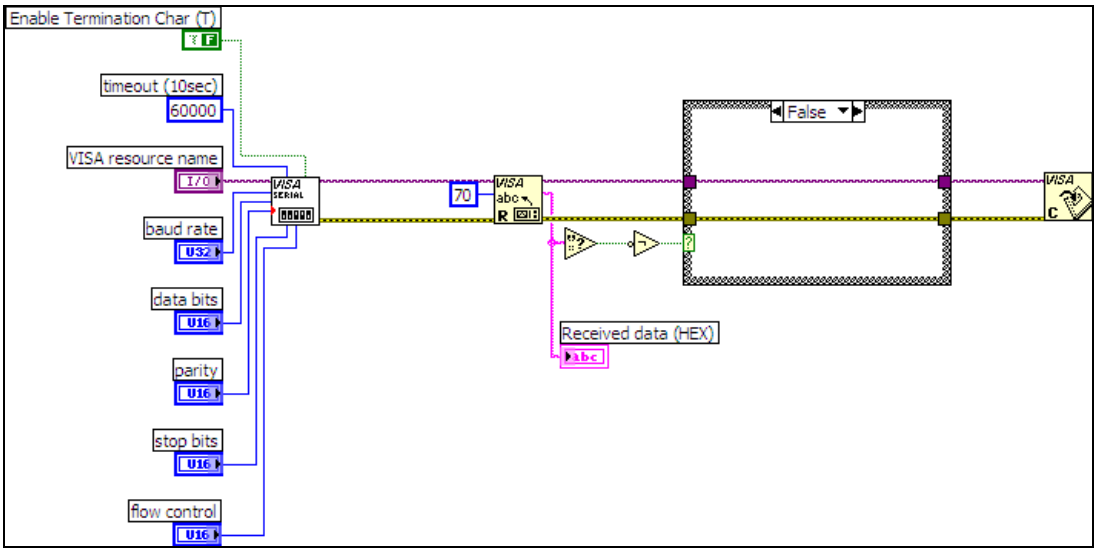

Figure 11: Interconnection showing the "false" case structure.

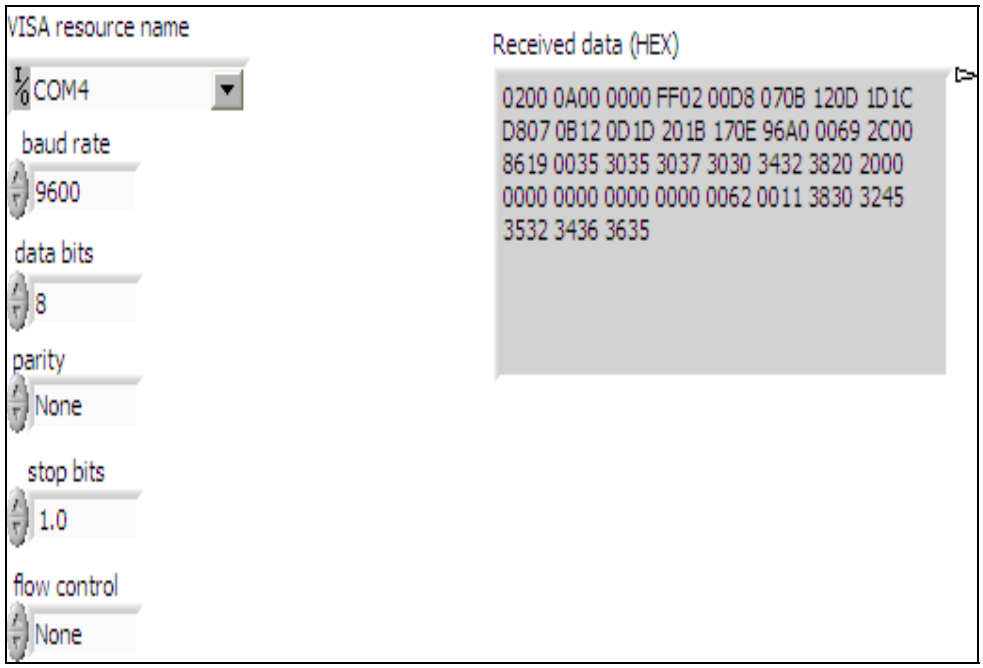

Figure 12: View of the front panel after completing a serial communication process

4.1.3 Data Processing

After the valid message is received, the data needs to be processed in order to extract relevant information. At this point the bytes are divided and transformed from a hexadecimal form to a readable form. In some cases, they are manipulated with arithmetical operations to present the relevant fields. In order to achieve this stage, the following steps are needed:

Construction of the block diagram: Right click over *Received data (Hex)* control in the block diagram and then go to *create* and then click *local variable*. This will create a variable reference inside the block diagram to receive the hexadecimal data. Insert the following VIs and interconnect them as shown in Figure 13 creating controls, indicators and constants:

- Conversion Palette: String to byte array VI and string to number VI.
- Array Palette: Index array VI.
- String Palette: String subset VI.
- Data manipulation Palette: Join numbers VI
- Structures Palette: Formula node and case structure VIs.
- Numeric Palette: Add VI.

Data processing techniques: The techniques used for data processing mix different ways of processing data. The parameters expected at the end of the operations are year, month, day, hour, minute, battery voltage, serial number, systolic pressure, diastolic pressure, Mean Arterial Pressure (MAP) and pulse/min. The data processing techniques are:

- *Indexed array data:* The technique used to obtain the year, month, day, hour, minute, and battery voltage consists in using the *string to byte array VI*. This VI arranges the hexadecimal received data in an array of bytes represented with ASCII values. With the *index array VI* it is easier to take portions of information and present it. Information of which byte represents a specific parameter can be obtained from the DCS. The year parameter is represented by two bytes and it is necessary to use the *join numbers VI* to obtain the parameter. A formula needs to be applied to the byte as specified in the DCS in order to obtain battery voltage.

- *Subset string data:* To obtain serial number, systolic pressure, diastolic pressure, MAP and pulse/min the technique used consisted in the *string subset VI*. This VI extracts the portions of received data based on the DCS. In some cases the data (systolic, diastolic, Pulse/min, MAP) needs to be transformed in numeric representation by the *string to number VI*. For the case of the systolic pulse is necessary to make an arithmetic operation with the *add VI* the operation is stated in the DCS. At this point the application is capable of taking the data obtained 4.1.2 and displaying it in a readable form as shown in Figure 14.

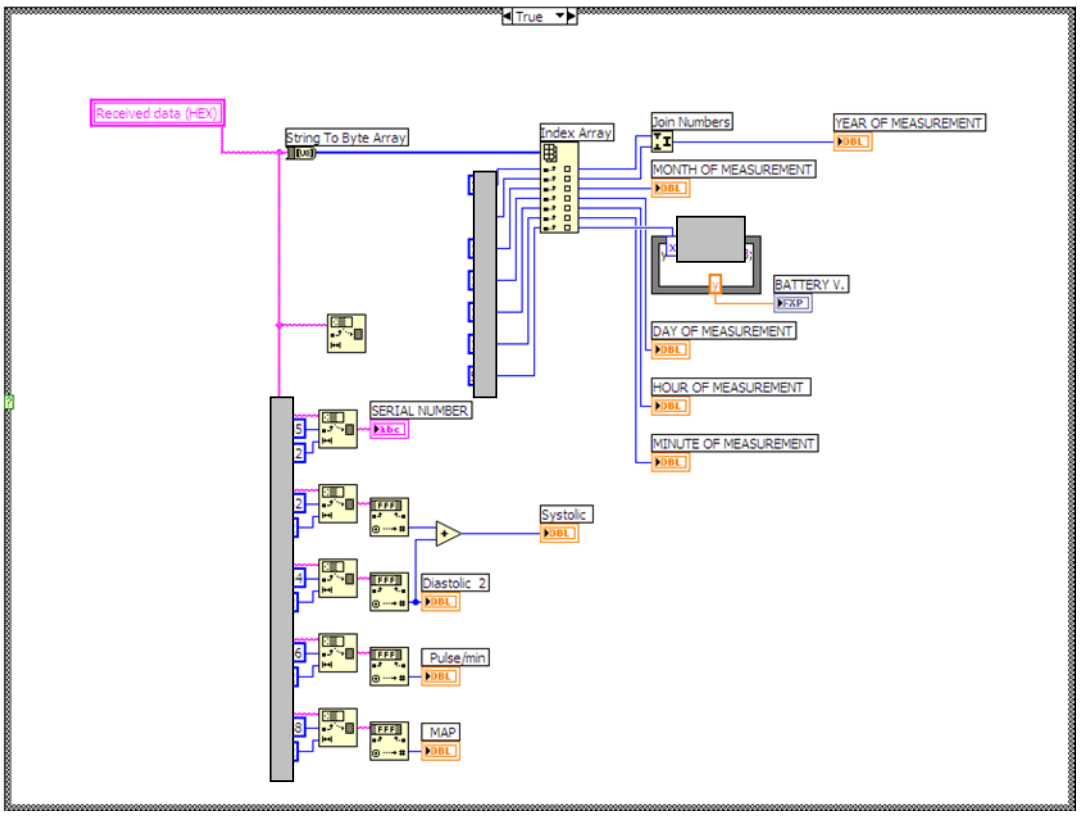

Figure 13: Data processing block diagram

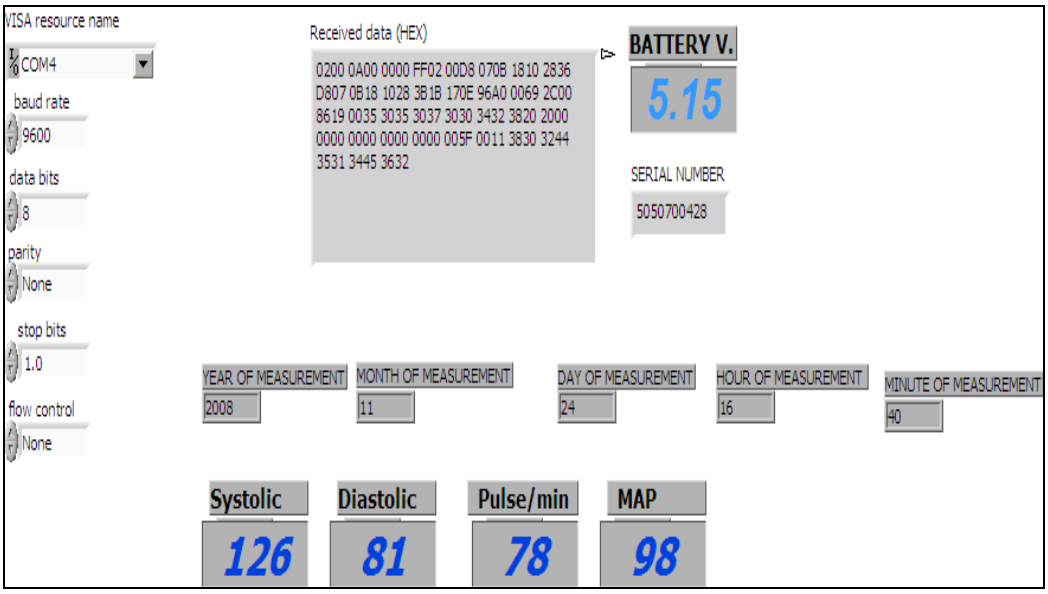

Figure 14: Capture of the front panel after data processing stage.

4.1.4 Warning System

Now that the data has been processed and displayed, a warning system for the MAP measurement is established. The color of the numeric text will change as a warning according to the thresholds established. In order to accomplish the following steps are taken.

Creation of a reference cluster control in the front panel: In order to arrange the data in a layout that will contain the information; the data will be arranged in a cluster. From the front panel a cluster will be created which will contain the year, month, day, hour, minute, systolic, diastolic, pulse/min and MAP parameters. The cluster should look similar to Figure 15.

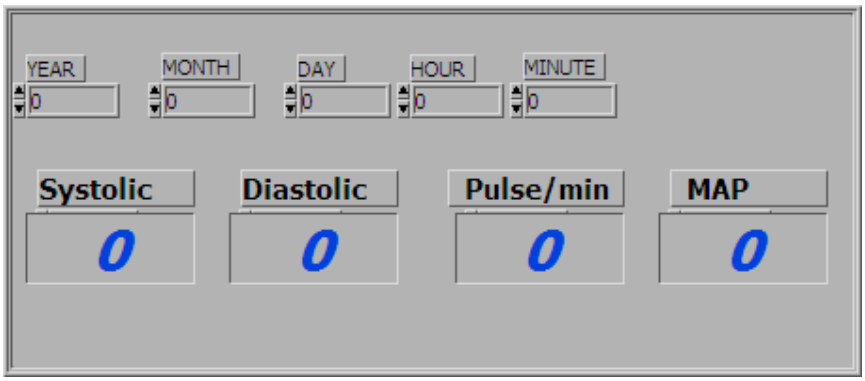

Figure 15: Cluster of measurement parameters, appearance changes according customization.

Creation of the block diagram: Insert the following VIs and arrange them as shown in Figure 16:

- Array palette: Threshold 1d array VI, index array function VI and array constant.
- Numeric palette: Round to nearest integer VI.
- Dialog and user interface palette: Color box constant.
- Cluster palette: Bundle by name VI.
- Application control pane palette: Property node VI.

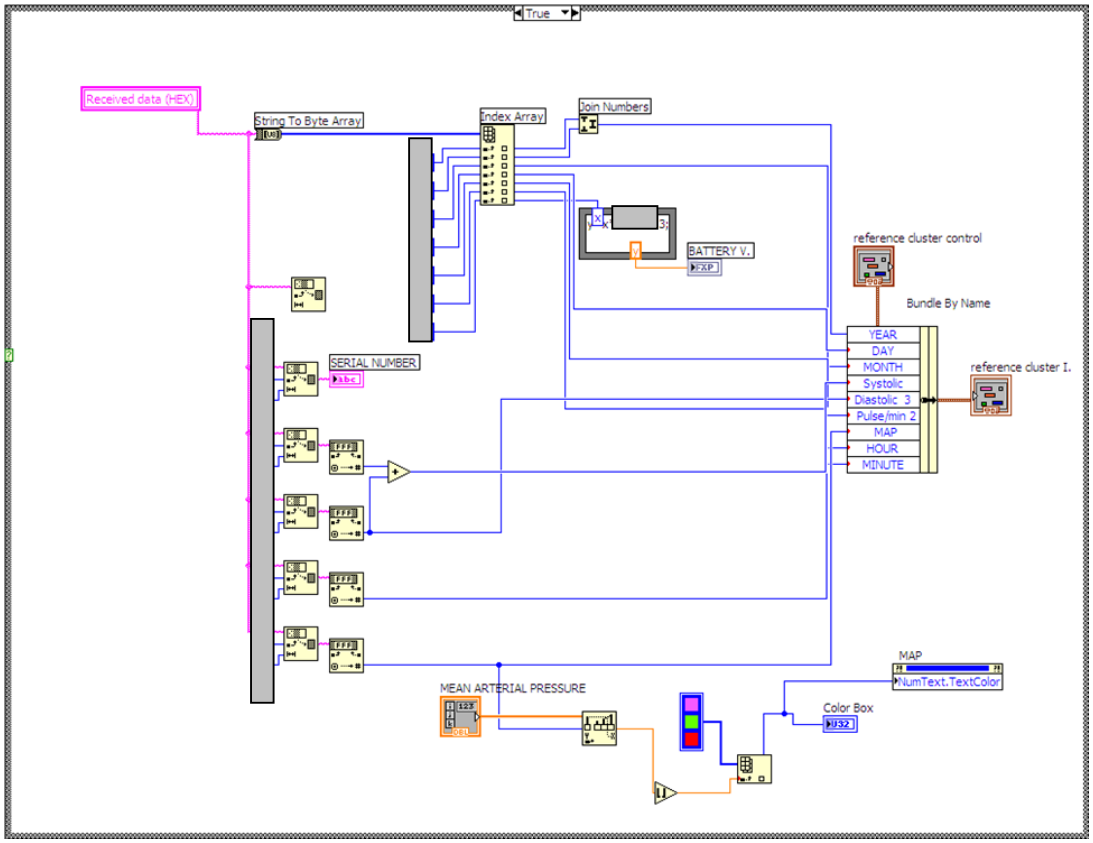

Figure 16: Updated block diagram showing only the data processing stage.

Bundling parameters into a cluster: With the reference cluster control created, it is simple to create an extra indicator by doing a copy paste operation and then changing the properties to an indicator. By the *bundle by name VI* all the parameters are bundled to a cluster as seen in Figure 17.

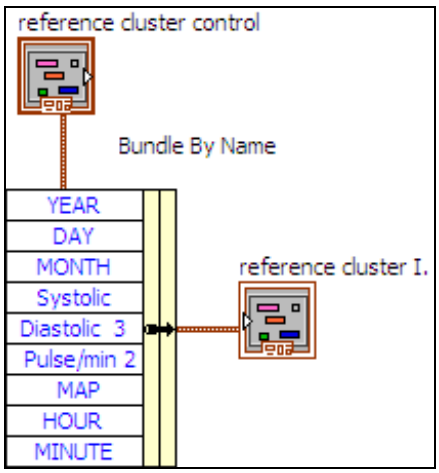

Figure 17: a cluster is filled by simply connecting the parameters by name to the VI

MAP warning display: The MAP parameter obtained from the measurements is introduced as a threshold for the *threshold 1D array VI.* It is then matched to the nearest threshold associated with the MAP control reference as seen in Figure 18. The resultant index value is rounded with the *round to nearest integer VI***;** afterwards the *index array VI* makes the proper color selection to be displayed**.** Finally the *property node VI* is linked to the MAP parameter contained in the cluster indicator, changing the color of the numeric text according to the threshold. The block diagram is shown in Figure 19. At this point the application in addition of section 4.1.3 can display a color alarm based on the MAP threshold. The final result is shown in Figure 20.

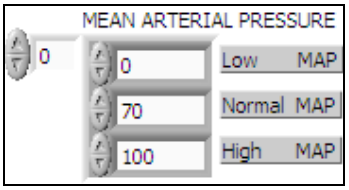

Figure 18: Threshold reference for the threshold 1D array VI.

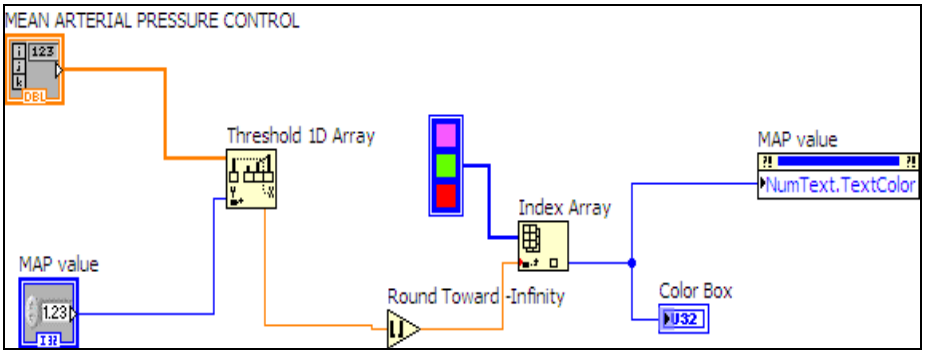

Figure 19: Color selection process for the warning system.

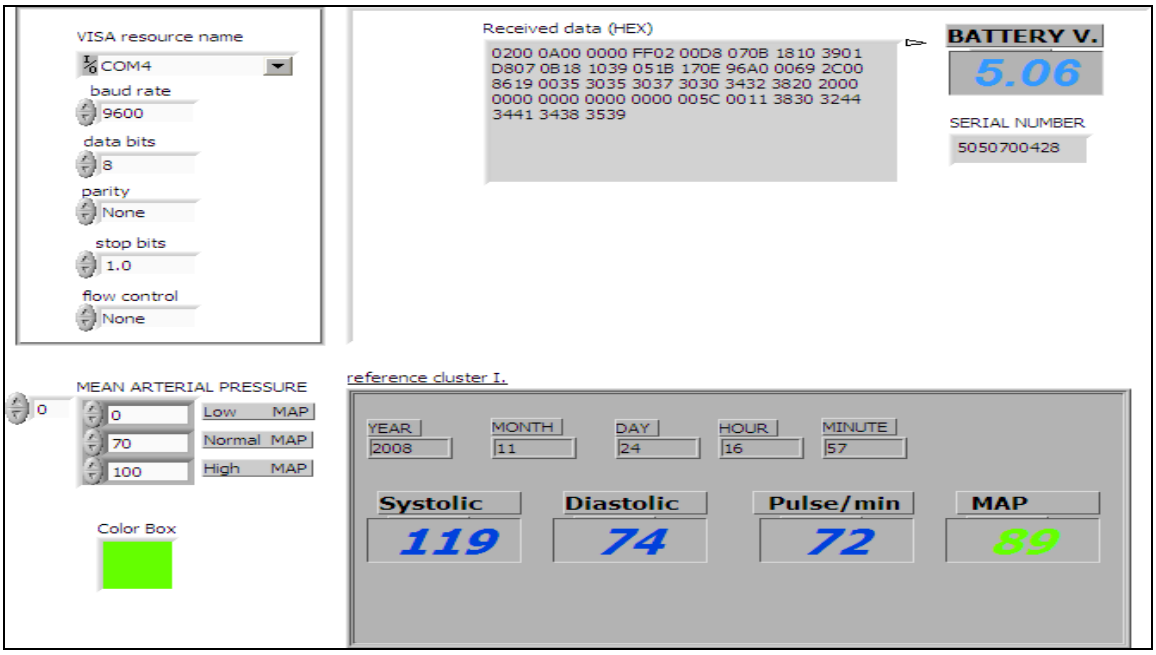

Figure 20: Capture of the front panel with warning system.

4.1.5 Measurements Stacking and Storing

In previous sections the application is only capable of processing one measurement received from the BP monitor. In this section functionality is added and the application is capable of stacking and storing multiple measurements in each session. This is useful when the user needs to take several measurements during one session, or if the BP monitor needs to upload stored measurements. In order to achieve this, the following steps are to be taken:

Creation of an array for stacking measurements: To create an array to stack measurement, the reference cluster indicator created in section 4.1.4 is taken and introduced in an array in the front panel. The result is shown in Figure 21.

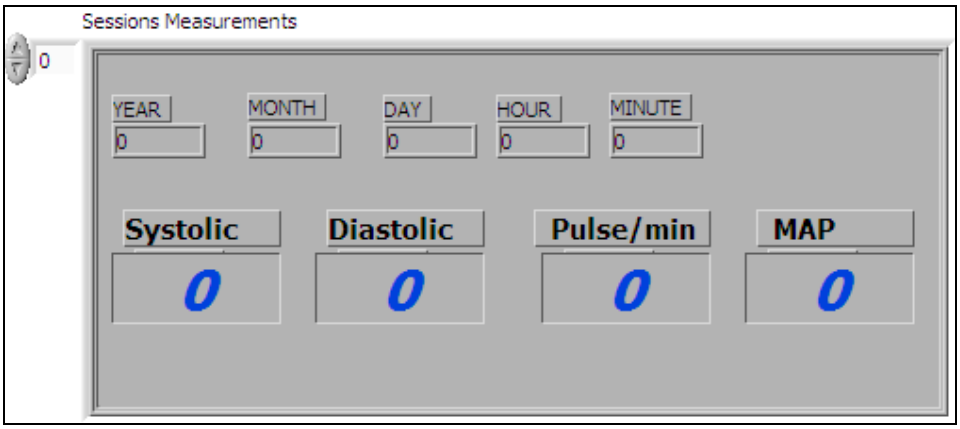

Figure 21: Array created for stacking session measurements.

Creation of the block diagram: Insert the following VIs in the block diagram and arrange them as in Figure 22:

- Array palette: into array function VI and build an array VI.
- Numeric palette: Add VI.
- Structures palette: Feedback node and while loop VI.
- Application control pane panel: Property node VI.
- File I/O palette: Read and write to spreadsheet VI.

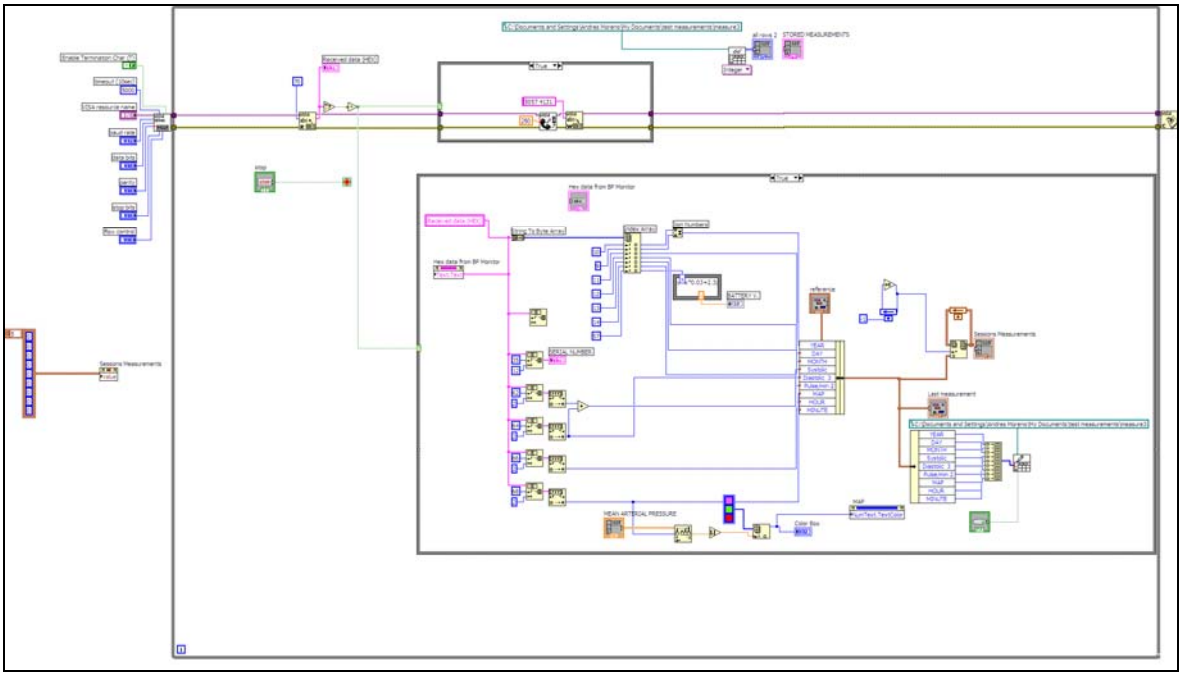

Figure 22: Block diagram with stacking and storing

Making the application ready for multiple measurements: For handling multiple measurements, a while loop contains the whole application but the serial port configuration and the closing connection of section.

Stacking measurements: For stacking the measurements each cluster lecture is stacked into an array of clusters. The *insert into array VI* stacks the cluster in the in the position given by the index generated by the *feedback node VI and the add VI*. A feedback node feed back the array in the next measurement to keep the stacked information. The operation in the block diagram is in Figure 23.

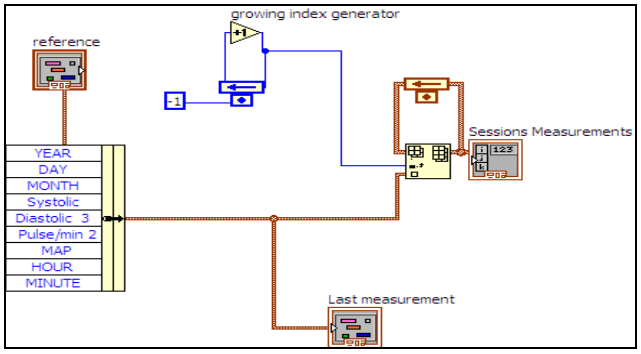

Figure 23: measurement stacking operation

Storing measurements: To store the measurements it is needed to store them in a spreadsheet file. With *unbundle by name and build array VIs*, the measurements are arranged in an array. The *write to spreadsheet VI* stores all the measurements in a spreadsheet file. Finally the retrieval of stored measurements is made with *read from spreadsheet file VI*. The block diagram of this operation is shown in Figure 24 and 25. At this point the application is complete and capable of stacking multiple measurements by session and to store measurement data in a spreadsheet.

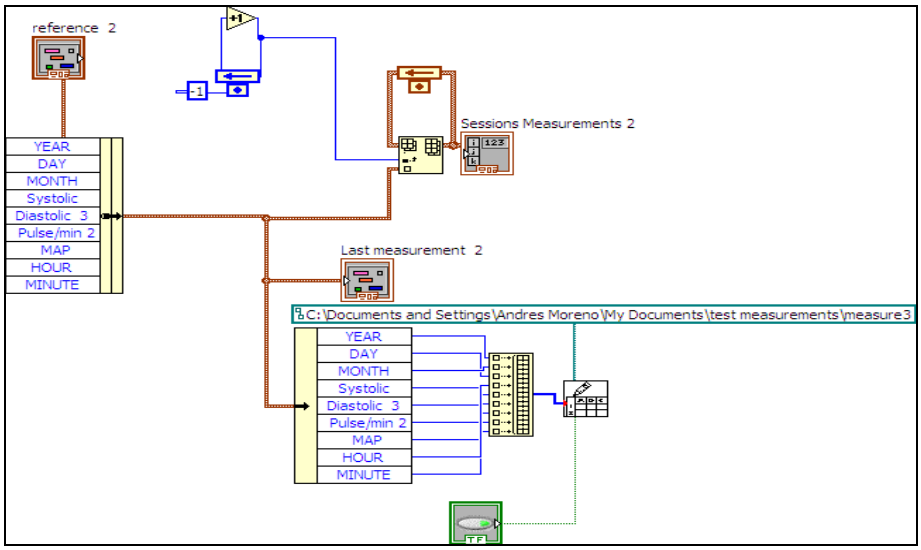

Figure 24: Summary of measurement store operation.

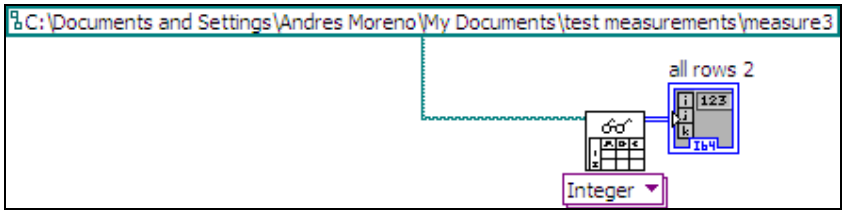

Figure 25: Read operation, this needs to be located inside while loop for updating.

4.1.6 LabVIEW BP monitoring Graphical interface

After completing the previous stages the final result is shown in Figure 26. The proposed interface has the following controls and displays:

1. ***VISA configuration panel:*** Is in charge of setting the connections parameters like COM port, baud rate, data bits, parity, stop bits and flow control.
2. ***MAP control:*** Is in charge of setting the thresholds for the MAP measurement warning display.
3. ***Hexa data display:*** Displays the raw hexadecimal data received by the device in every measurement.
4. ***Battery voltage display:*** Shows the current battery voltage of the BP monitor.
5. ***Last and current session measurements displays:*** Displays the last and all the session measurements received from the BP monitor.
6. ***Store measurements control and display:*** The control sets the action to overwrite or append measurements; it displays the stored measurements.

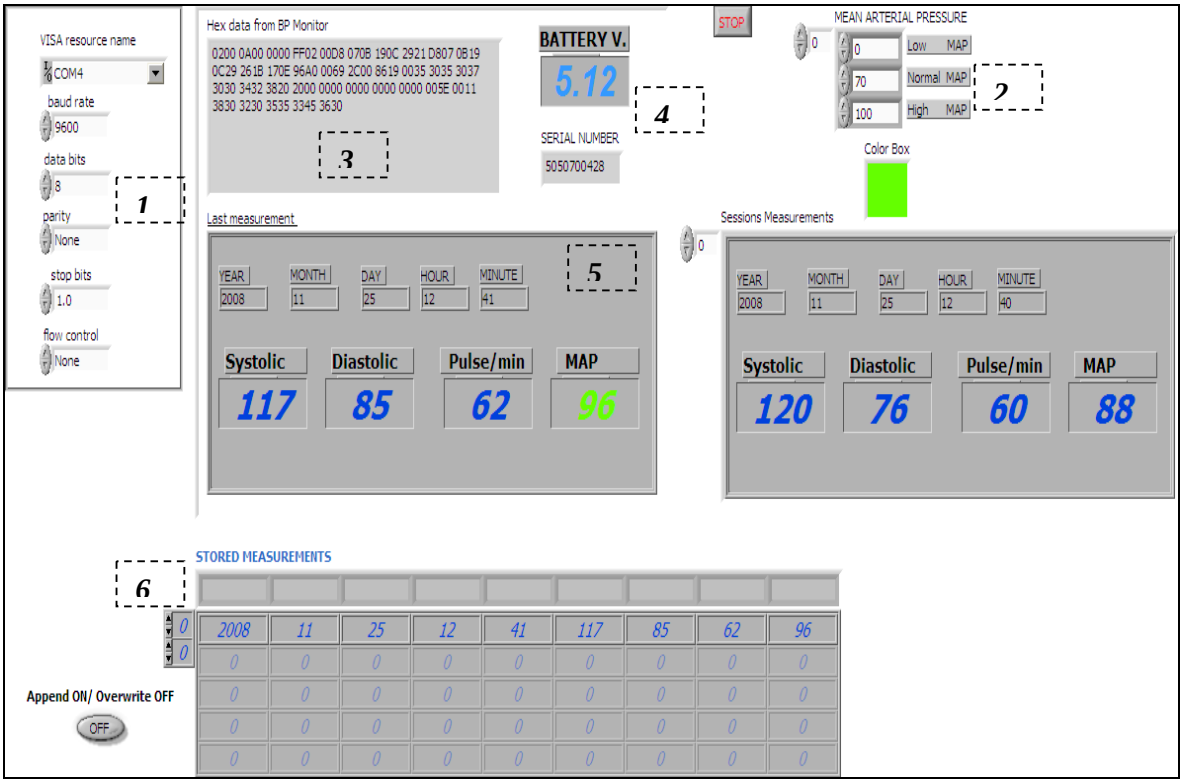

Figure 26: Final application result, the layout depends on the customization of the front panel.

4.2 ECG Monitoring

4.2.1 LabVIEW ECG Simulation

This stage simulates an Electrocardiogram (ECG) signal to publish. The signal generated is based on a recording from a laboratory performed in the Biomedical Instrumentation course. A simulation gives students the advantage of working with the laboratory experiments in different locations.

Construction of the block diagram: Insert the following VIs and interconnect them as shown in Figure 27 creating their controls and indicators.

- Structures palette: Feedback node, case structure and while loop VIs.
- Input express VI palette: Simulate arbitrary signal express VI.
- The String functions palette: Format into string, string length and concatenate strings VIs.
- Signal Manipulation Express VI palette: Convert from dynamic data express VI.
- Array palette: Insert into array, array size, index array and delete from array VIs.
- Signal Operation palette: Peak detector VI
- Comparison palette: Less? and equal ? VIs.
- Numeric Functions palette: Subtract, divide, increment and round to nearest VIs.
- Conversion VIs and Functions: To long integer VI.

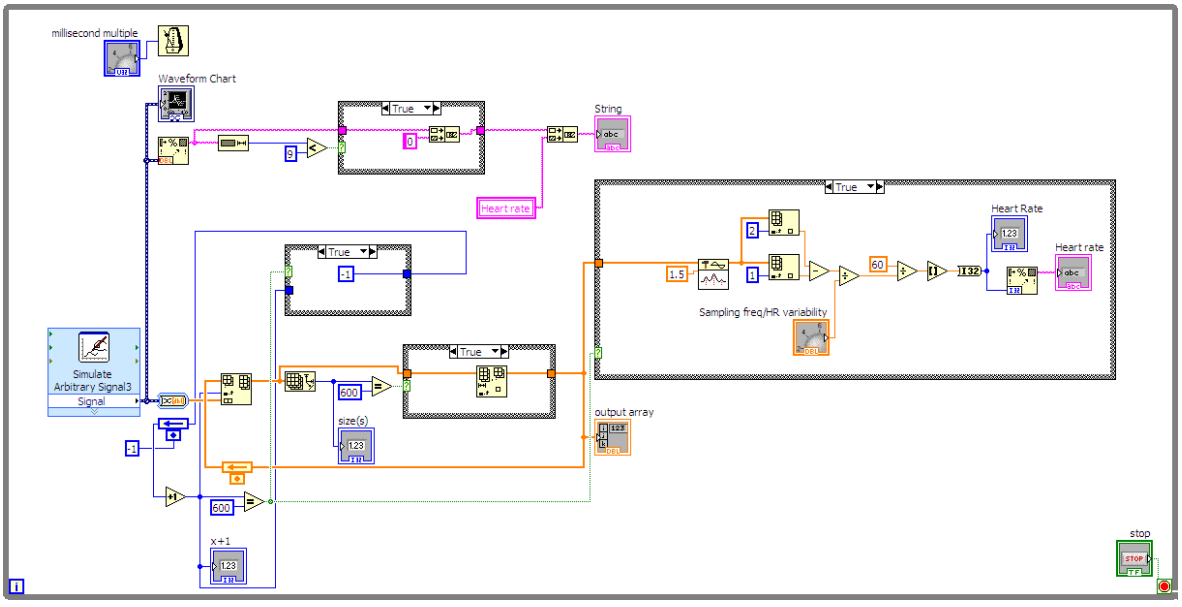

Figure 27: Labview ECG simulation block diagram

Creation of ECG Signal: The pre recorded signal is simulated in each iteration by the *simulate arbitrary signal VI* .Figure 28 shows details about the VI and signal.

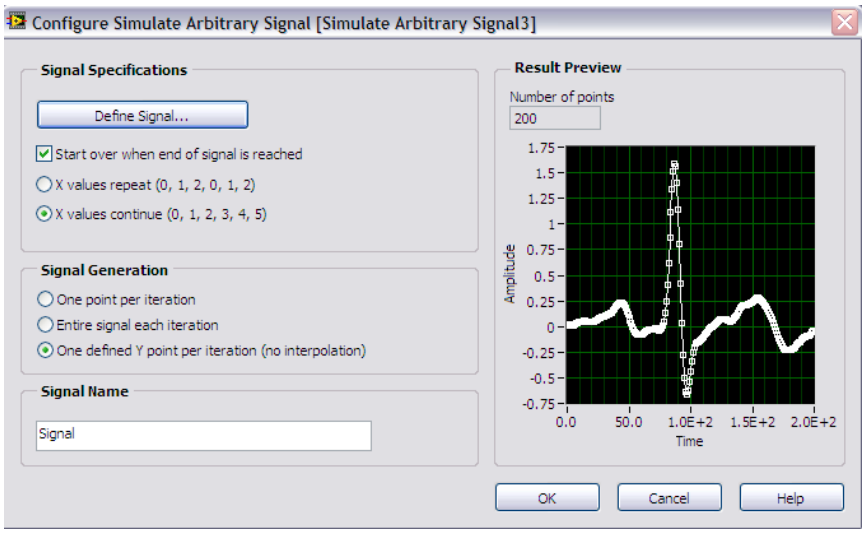

Figure 28: ECG signal details

Heart Rate Extraction: The simulated signal is analyzed during each interaction to extract the heart rate. Since the same signal is repeated continuously, it is necessary to change the estimated sampling frequency to obtain a heart rate variability effect. With the *arrays VI*, case structures, comparison and numeric functions the application first collect 600 samples. Afterwards, the *peak detector VI* detects the RS of the QRS complex of the 600 samples signal. The indexes of these RS are subtracted and divided by the variable and arbitrary sampling frequency. After dividing these by the 60 seconds in a minute the estimated heart rate is obtained. Finally this value is rounded and formatted into a string for further use. Figure 29 shows the section of the block diagram responsible of this task.

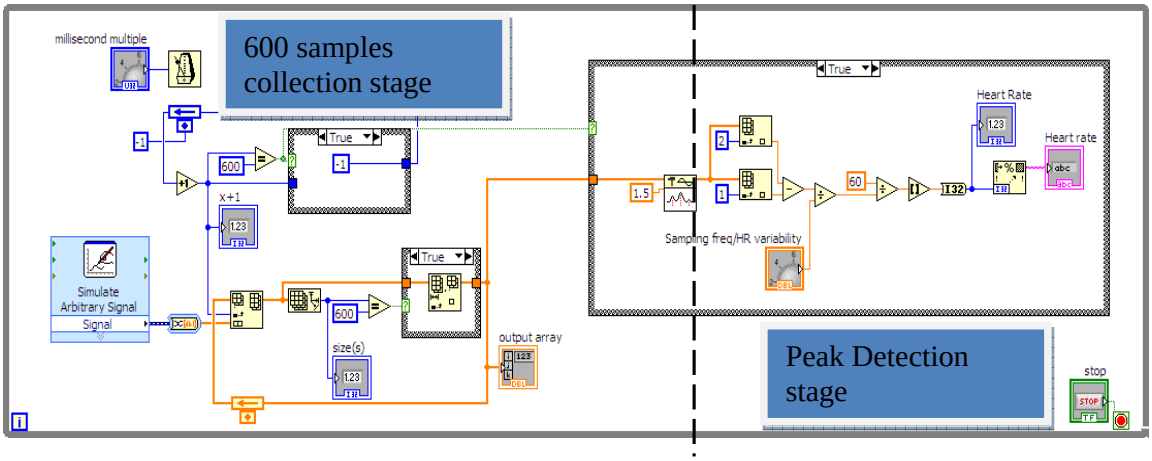

Figure 29: Heart Rate Extraction block diagram.

Creation of the Data Package: Once the heart rate variable is obtained it is merged with the amplitude values to create a data package for publishing. Due amplitudes can be either positive or negative it is necessary to make the length of the packet even before being published. The maximum length is set to 9; in case that the amplitude value does not present this condition a zero will be concatenated to the packet. Afterwards, the value of the heart rate is also concatenated and the string is ready to be published. Figure 30 shows the corresponding section of the block responsible of this operation.

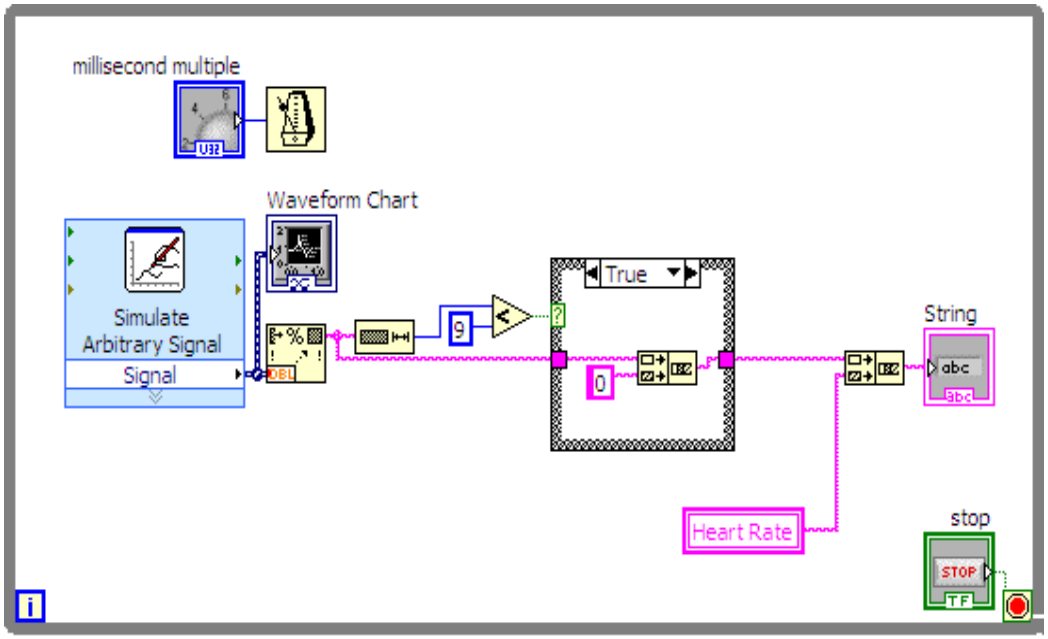

Figure 30: Data package creation block diagram.

4.2.2 LabVIEW TCP Publishing System

At this section there is a Transfer Control Protocol (TCP) server listening constantly for incoming connections from a client. Once the connection is detected the server sends the resulting string of the previous section to the client. If the client closes the session the server will get in stand-by mode until a new connection is made.

Construction of the block diagram: Insert the following VIs and interconnect them as shown in Figure 31 creating their controls and indicators.

- TCP VI and Functions: TCP listen, TCP write and TCP close VIs.
- Timing VIs and Functions: Wait until next ms multiple functions.
- Cluster palette: Unbundle by name function.
- Boolean Functions palette: OR function.
- Dialog & User Interfaces VIs and Functions: Clear errors VI.

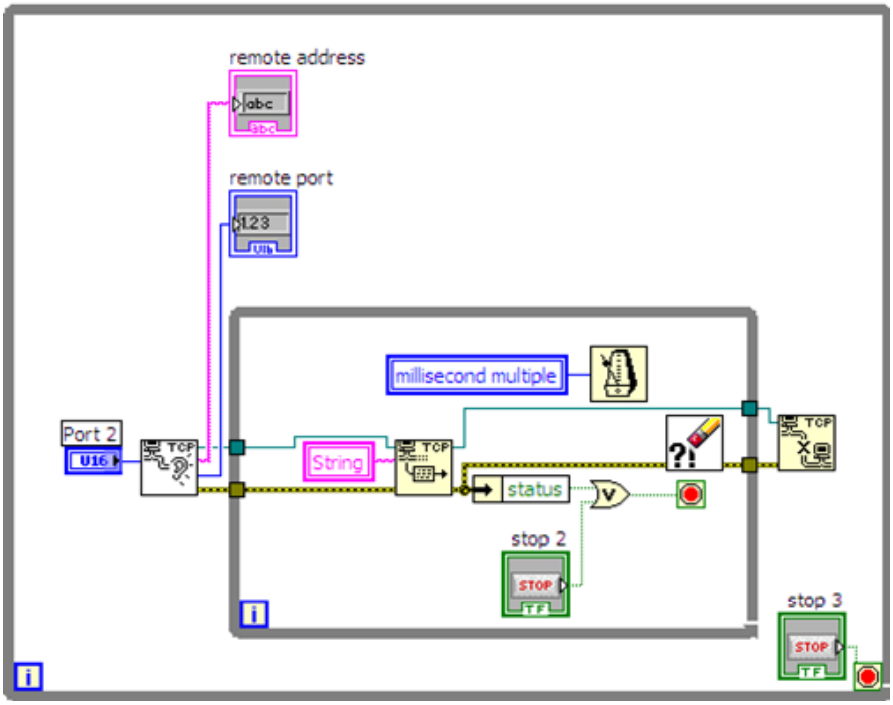

Figure 31: LabVIEW TCP publishing system block diagram.

TCP Server: This server is contained in two main loops. The outer loop will take care of the stand by functionality of the system. The inner loop is in charge of the write operation and will publish the resultant string if available. Both TCP server write function and data package loop are synchronized with the same iteration delay control. This way the timing of the data is reliable. The *unbundled by name VI* detects if there is any error in the connection (connection closed by client) and stops the writing function. The *clear error VI* clears any minor errors that need to be ignored for continuing the execution of the server.

4.2.3 LabVIEW ECG Simulation Server

By merging the blocks of sections 4.2.1 and 4.2.3 the final Simulation server is obtained as seen in Figure 32. The proposed graphical interface is shown in Figure 33. The interface has the following controls and displays:

1. ***Delay Control:*** Controls the speed of the simulation signal that is running.
2. ***Heart Rate Variability control:*** Controls the change of the heart rate parameter.
3. ***Heart Rate display:*** Displays the current heart rate.
4. ***Samples display:*** Displays the amount of samples collected for the hearth rate calculation.
5. ***Published string display:*** Displays the string to be published by the TCP server.
6. ***ECG amplitudes display:*** Plots the ECG amplitudes in a chart.
7. ***TCP port control:*** Controls in which TCP port the data will be published.
8. ***Remote address:*** The client's address.
9. ***Remote port:*** Port where the client is connected.

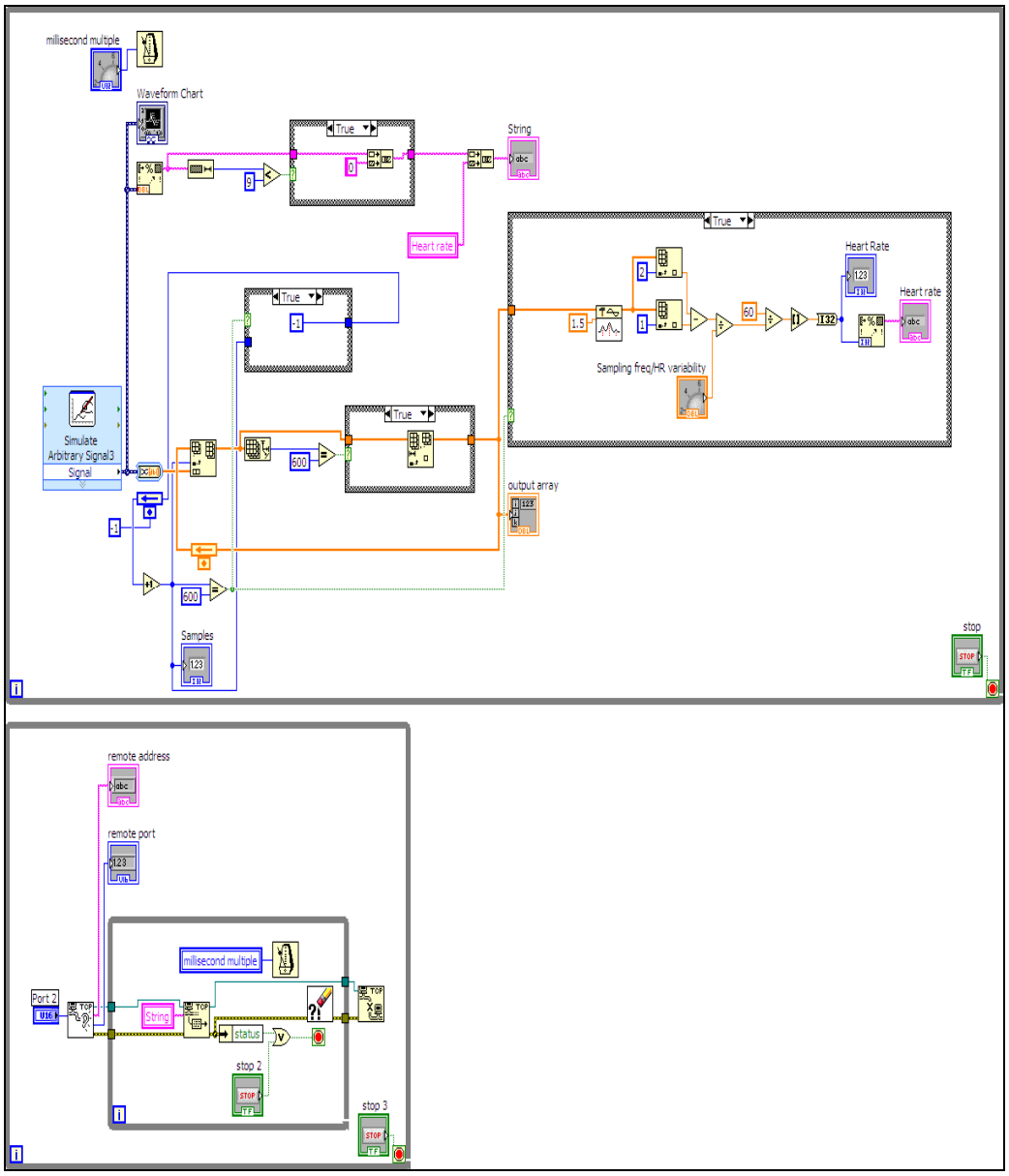

Figure 32: Simulation Server block diagram.

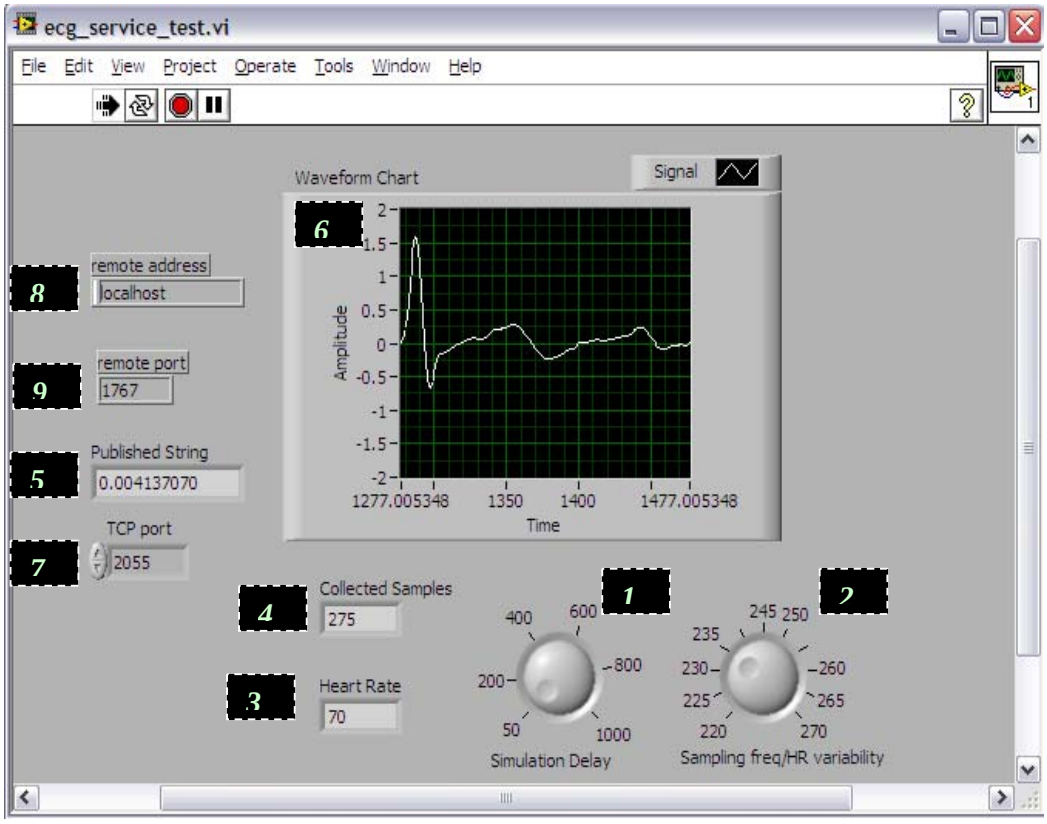

Figure 33: Simulation Server Front Panel

4.2.4 Flex Socket Client

A web based client with TCP sockets is created with the Flex Builder. The client can initiate and end connections to different ECG servers that publish data in different sockets. The real time data is plotted in a graph and also shown in a text box. When using this application, it is necessary to take into consideration the network policies in which the client and the server are contained. It is recommended to execute this experiment in a controlled scenario. To accomplish this, the following steps are taken:

Creation of a New Flex Builder Project: The first step is to create a new project in the flex builder. Go to file/New/Flex in the configuration window to create a new project, Figure 34. In this window the proper name and the option of web application are selected as in Figure 35. No application server is chosen in order to deploy the result as a standalone application in the web browser. After clicking finish, the environment is ready to start developing a flex application as shown in Figure 36.

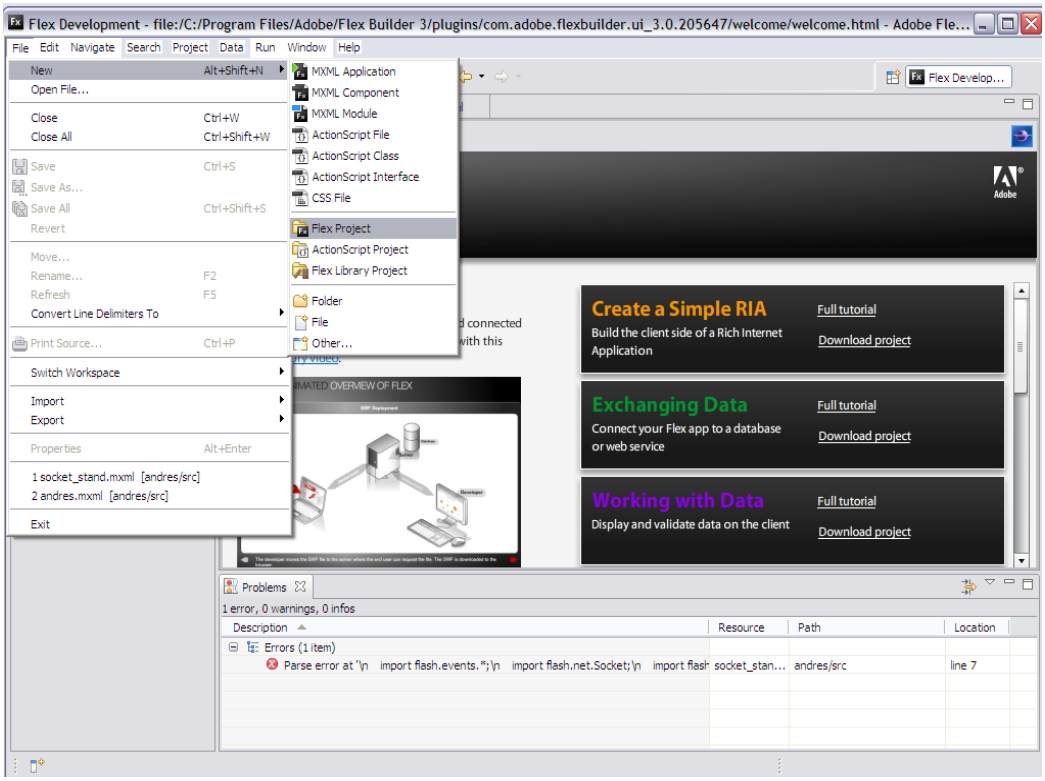

Figure 34: Creation of new flex project.

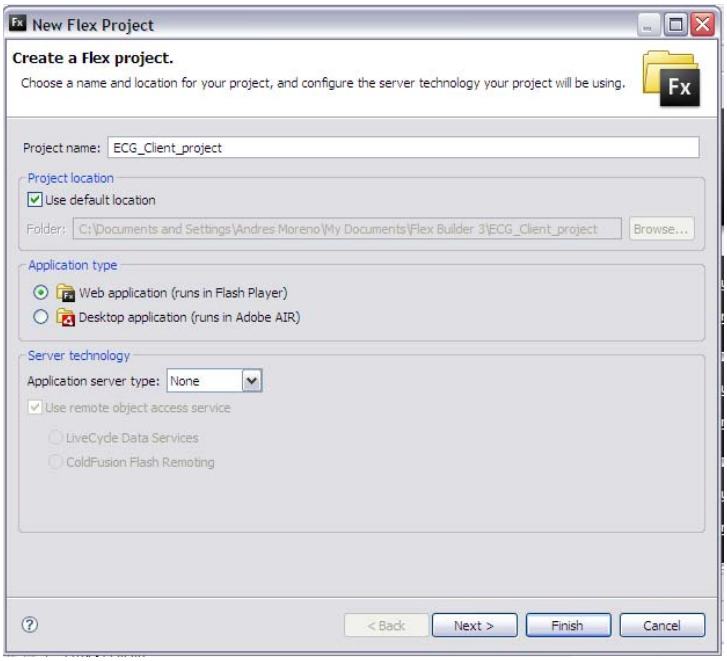

Figure 35: Flex project configuration

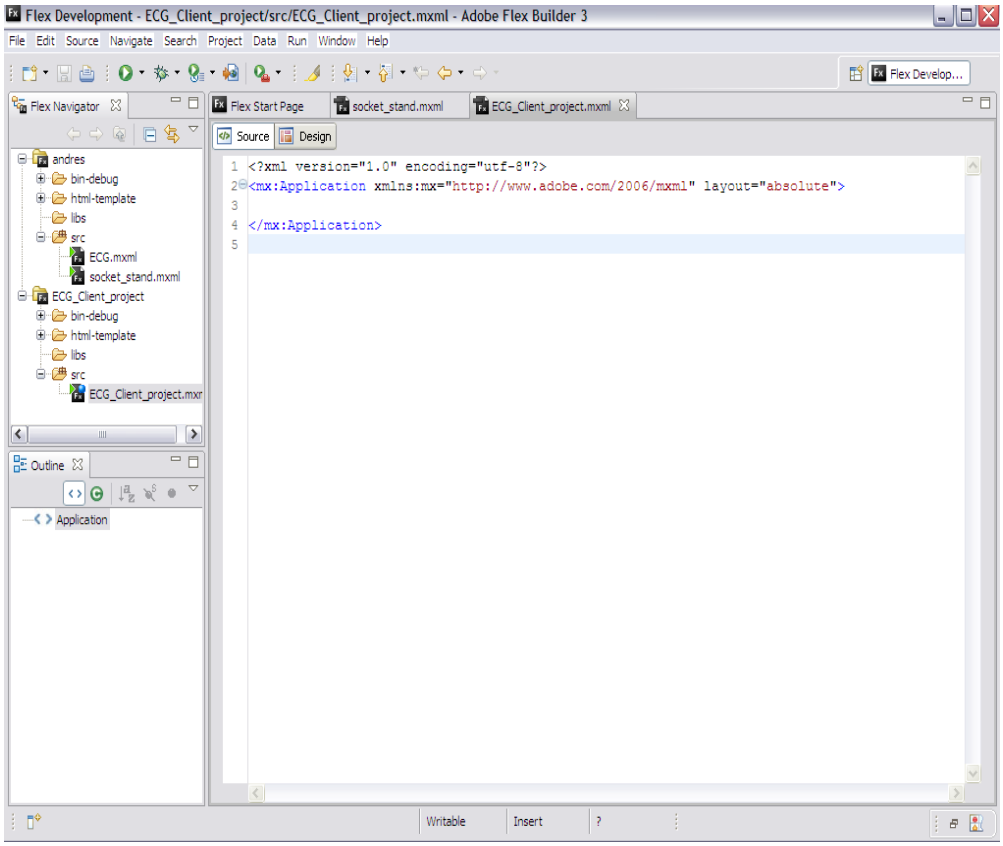

Figure 36: Flex project working area

Developing the Flex code: Flex coding requires knowledge of ActionScript and MXML. The code is divided in two sections: the ActionScript and the MXML graphical layout code. Appendix A contains the source code properly commented. The first part of the code introduces behavior (ActionScript) to the application. The second one (MXML) models the graphical interface.

Flex client application: Copy the code described in Appendix A to the working area of the flex project and run it; the final graphical interface can be visualized in Figure 37. The application is easy to use and has the following controls and displays:

1. ***Server Name control:*** The name or address of the host containing the LabVIEW ECG Simulation Server.
2. ***Port control:*** The TCP port in which the server data is available.
3. ***Start and Stop Buttons controls:*** To begin and stop reading data from the server.
4. ***Server Messages display:*** Displays the data read from the port in the server.
5. ***Heart Rate display:*** Displays the heart rate data extracted from the server messages.
6. ***Chart area display:*** Plots the amplitudes values obtained from the server messages.

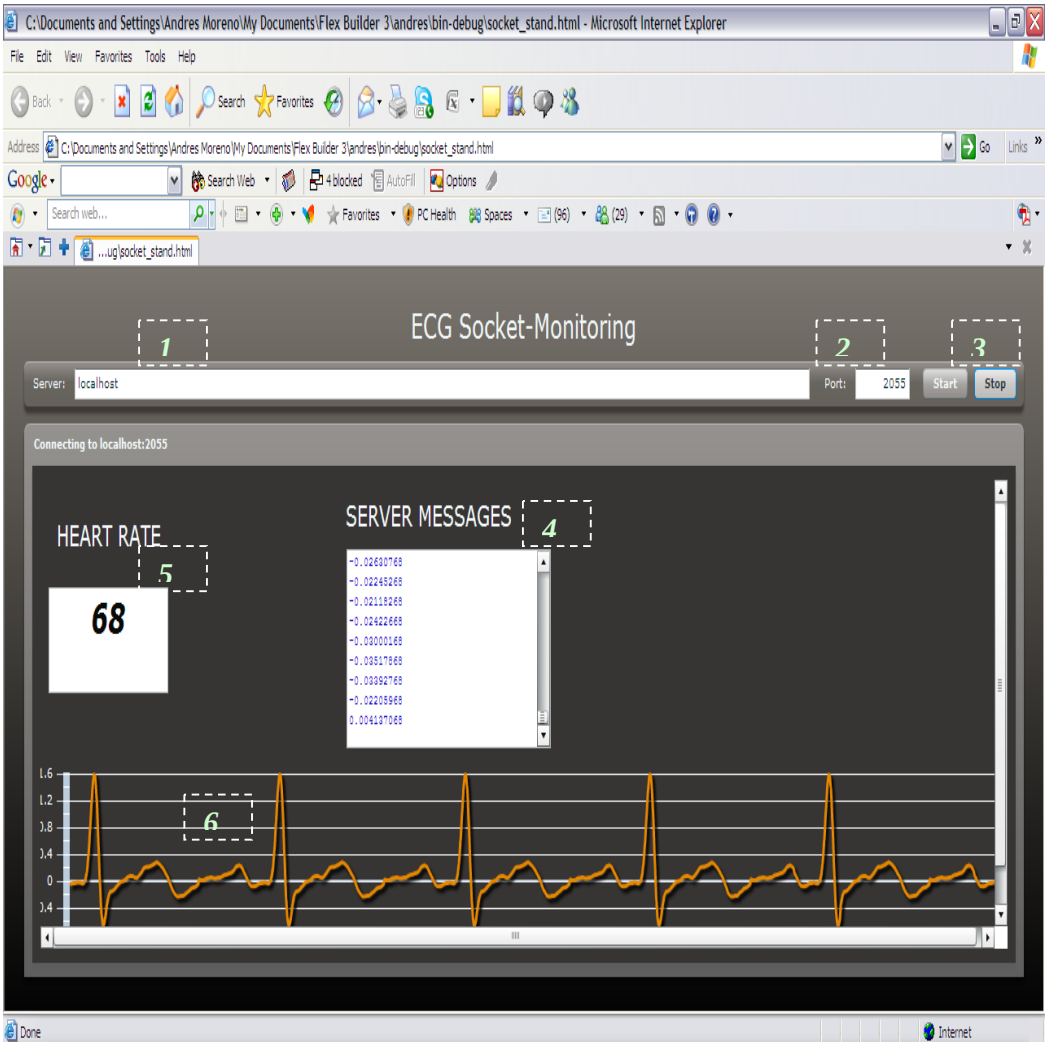

Figure 37: ECG client interface connected to Local host server in port 2055

5. Discussion

5.1 Experiments Evaluation

The purpose of this work was to design a laboratory for the eHealth in the Biomedical Engineering Masters Program. Furthermore, this course is part of the Health Care Informatics specialization in the Communication Engineering Masters Program. During this document's final stage of evaluation and through its final presentation with other professors, the inclusion of this laboratory in the eHealth course is still under consideration. The use of these experiments is meant for use in laboratories in biomedical instrumentation courses or as future reference for works in other medical system projects.

5.2 Design Considerations

When these experiments were developed some requirements were established before hand. The initial requirements are:

- Use of LabVIEW as core programming language when developing the experiments. LabVIEW is in increasing inclusion in the biomedical engineering department in courses like Biomedical Instrumentation.
- Use of A&D UA-767 PlusBT, a Blood Pressure (BP) monitor in one of the experiments.
- Use the concept of the Electrocardiogram (ECG) for a monitoring application.
- Use a real time web based client for the ECG monitoring application.

Complications begun when choosing a proper tool for designing the web based client for the ECG. The author's inclination towards Flex came after extensive research of available frameworks in the market like Java, Ajax, Microsoft Silverlight etc. Flex stands out among the others for the following reasons:

- Clear and easy to follow tutorial documentation. The author reviewed the documentation tutorial for all of the previous mentioned frameworks. For learning purposes. Flex has a quicker learning time and it is recommended for these experiments.
- Friendly and freeware Integrated Developing Environment (IDE) Flex Builder 3 Student Edition.

5.3 Limitations

An important limitation can be observed in the graphical interface of the applications. The experiments are oriented to learning purposes; there are no complex visual designs or animations.

Furthermore, some limitations were observed when designing the ECG experiment. Due to the Remote Monitoring scenario of the application, a controlled testing scheme was needed to achieve success. In one case, the networking environment was developed as a localhost application. Both server and client were working in the same computer. In other cases, a simple network device (switch, router) was used for the network interconnection.

6. Future Work

The applications developed in the experiments are pieces of big puzzles in more complex scenarios. Although including large databases or complex network scenarios are out of the scope of this thesis work; real life applications requires taking into consideration these and other several human factors. An example of work related to real life applications can be seen in *"Design of an Internet-Based Disease Management System for Chronicle Heart Failure"* by Anna Gund [30]. In this work, the same Blood Pressure (BP) monitor and internet technology were used to create a disease management system. Students are encouraged to overcome the limitations found in this document and develop state of the art applications for real medical problems. Some suggestions for future projects are:

- Expand to more complex network scenarios which include remote control.
- Develop more sophisticated user interfaces.
- Use live data instead of simulations.
- Incorporate multiple vital signs.

7. Conclusion

The purpose of this thesis was to design a laboratory for the eHealth course. Although the experiments developed are yet to be approved for use in the proposed course; other important points were proven during the development of the applications. LabVIEW as a language program has been proven to be useful in the developing of the experiments. At the moment the author of this document begun the development of the applications he had no previous knowledge in LabVIEW. This proofs that it is possible for any student to reproduce the experiments with guidance. There was some struggle at the moment of working with Flex due to Object Oriented Programming (OOP); although the learning curve is lower compared to more complex languages like Java or .Net. Flex stands out as a good alternative for students searching for less complex alternatives to develop web and desktop applications. Besides all this, all the applications worked successfully accomplishing the functionalities for what they were designed. Students who decide to take on this challenge will get a great deal of experience in how medical concepts and Information and Communication Technologies (ICT) can work together to take medical solutions to a next level.

References

[1] eHealth. 2009 [cited 2009 Jan 19]. Available from: http://en.wikipedia.org/wiki/EHealth

[2] Eysenbach G. What is eHealth?. J Med Internet Res [serial online] 2001 [cited 2009 Mar 27]; 3(2). Available from URL: http://www.jmir.org/2001/2/e20

[3] Johnson TR. What is Health Informatics?. [Online] 2006 [cited 2009 Jan 19]. Available from: http://www.shis.uth.tmc.edu/about-us/what-is-health-informatics

[4] Garcia A. Integration of Bluetooth- enabled Sensors into an E-health application. Master Thesis, School of Engineering, University College of Borås; 2008

[5] Olansen JB, Rosow Eric. Virtual Bio-Instrumentation: Biomedical, Clinical, and Healthcare Applications in labview. Upper Saddle River (NJ): Prentice Hall, Inc.; 2002. p.513-526

[6] LabVIEW. [Online] 2009 [cited 2009 Jan 26]. Available from: http://en.wikipedia.org/wiki/LabVIEW

[7] National Instruments Corporation. Introduction to G Programming. [Online] 2009 [cited 2009 Mar 27]. Available from: http://zone.ni.com/devzone/cda/tut/p/id/7668

[8]Olansen JB, Rosow Eric. Virtual Bio-Instrumentation: Biomedical, Clinical, and Healthcare Applications in labview. Upper Saddle River (NJ): Prentice Hall, Inc.; 2002. p.490

[9] National Instruments Corporation. How Can I Learn LabVIEW?. [Online] 2009 [cited 2009 Jan 26]. Available from: http://www.ni.com/academic/lv_training/how_learn_lv.htm

[10] Rapoza J. RIA War Is Brewing. 2008[cited 2009 Jan 26]. Available from: http://etech.eweek.com/content/application_development/ria_war_is_brewing.html

[11] Adobe Systems Incorporated. Get Oriented To Flex. [Online]. 2007 [cited 2009 Jan 26]. Available from: http://learn.adobe.com/wiki/display/Flex/Get+oriented+to+Flex

[12] Adobe Systems Incorporated. Programming Action Script 3.0/Introduction to Action Script 3.0 . [Online]. 2008 [cited 2009 Jan 26]. Available from: http://livedocs.adobe.com/flex/3/html/01_Introduction_2.html#122319

[13]. UA 767 PlusBT. 2009[cited 2009 Jan 26]. Available from: http://www.telemedicine.jp/product_ua767pbt.html

[14]. Instruction Manual. 2009[cited 2009 Jan 26]. Available from: http://www.aandd.jp/products/manual/manual_medical.html

[15] Stallings W. Wireless Communications and Networks. 2nd ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2005. p 464-465

[16] Bluetooth SIG. Serial Port Profile (SPP). [Online]. 2009 [cited 2009 Jan 26]. Available from: http://www.bluetooth.com/Bluetooth/Technology/Works/SPP.htm

[17] Stallings W. Wireless Communications and Networks. 2nd ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2005. p 466

[18] Comer DE. Internetworking with TCP/IP Volume1: Principles, Protocols, and Architecture. 5th ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2006. p 2

[19] Stallings W. Wireless Communications and Networks. 2nd ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2005. p 71-72

[20] Stallings W. Wireless Communications and Networks. 2nd ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2005. p 73-74

[21] Stallings W. Wireless Communications and Networks. 2nd ed. Upper Saddle River (NJ): Pearson Education, Inc.; 2005. p 75

[22] Olansen JB, Rosow Eric. Virtual Bio-Instrumentation: Biomedical, Clinical, and Healthcare Applications in labview. Upper Saddle River (NJ): Prentice Hall, Inc.; 2002. p.492

[23] Tortora GJ, Derrickson B. Introduction to the Human Body: The essentials of anatomy and physiology. 7th ed. Danvers (MA): John Wileys & Sons, Inc.; 2007. p 391

[24] Mean Arterial Pressure. 2009 [cited 2009 Jan 26]. Available from: http://en.wikipedia.org/wiki/Mean_arterial_pressure

[25] Tortora GJ, Derrickson B. Introduction to the Human Body: The essentials of anatomy and physiology. 7th ed. Danvers (MA): John Wileys & Sons, Inc.; 2007. p 374

[26] Adobe Systems Incorporated. User Interfaces. [Online]. 2008 [cited 2009 Jan 26]. Available from: http://livedocs.adobe.com/flex/3/html/controls_01.html

[27] Adobe Systems Incorporated. Programming Action Script 3.0/Flash Player Apis/ Networking and Communications . [Online]. 2008 [cited 2009 Jan 26]. Available from: http://livedocs.adobe.com/flex/3/html/17_Networking_and_communications_5.html#124745

[28] Gregory E. VISA. [Online] 2004 [cited 2009 Jan 26]. Available from: http://cnx.org/content/m12288/latest/

[29] National Instruments Corporation. LabVIEW 8.2 help/ VISA VIs and Functions. [Online] 2006 [cited 2009 Jan 26]. Available from: http://zone.ni.com/reference/en-XX/help/371361B-01/lvinstio/visa_lib_ref_func_descr/

[30] Gund A. Design of an Internet-Based Disease Management System for Chronicle Heart Failure. Licentiate Thesis, Dept. of Signals and Systems, Chalmers University of Technology; 2008

Appendix A: ECG Monitoring Flex Client Source Code

As follows the source code of the Flex application with proper comments; the comments start with a "//" or "/*":

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
backgroundGradientAlphas="[1.0, 1.0]" backgroundGradientColors="[#7B766F,
#040404]" viewSourceURL="srcview/index.html">
```

I. ACTION SCRIPT CODING :

```
  <mx:Script>
        // Import the classes needed for script functionality
                <![CDATA[
  import flash.events.*;
  import flash.net.Socket;
  import flash.system.Security;
  import flash.utils.ByteArray;
  import flash.utils.setTimeout;
  import flash.utils.IDataInput;
  import flash.display.Sprite;
  import mx.collections.ArrayCollection;
  import mx.controls.TextArea;
  import mx.core.UIComponent;
// Creation of Private variables for using in the standalone script
  private var serverURL:String;
  private var port:int;
  private var socket:Socket;
  private var ta:TextArea;
  [Bindable] private var myRandomData:ArrayCollection;
  [Bindable] private var RandomData:ArrayCollection;



 // fucntion incorporated as standalone

 public function connect():void
 {
        // Link the variables of the graphical interface to the script
        serverURL = serverName.text;
        port=int(portNumber.text);
        ta = output;

        /* Creation of a new Socket object for TCP connection
        and assignation of  event listeners.*/

        socket = new Socket();
```

```
    socket.addEventListener(Event.CONNECT, connectHandler);
    socket.addEventListener(Event.CLOSE, closeHandler);
    socket.addEventListener(ErrorEvent.ERROR, errorHandler);
    socket.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);
    socket.addEventListener(ProgressEvent.SOCKET_DATA, dataHandler);

    // Load policy file from remote server.
    Security.loadPolicyFile("http://" + serverURL + "/crossdomain.xml");

    // Connecting to remote Labview ECG simulation server.
    try {
        msg("Trying to connect to " + serverURL + ":" + port + "\n");
        socket.connect(serverURL, port);
    } catch (error:Error) {
        /*
            If the client is unable to connect
            to the server, an error message is displayed
            and the connection is closed.
        */
        msg(error.message + "\n");
        socket.close();
    }
    console.title = "Connecting to " + serverName.text + ":" + portNumber.text;
    console.enabled = true;

    //Array to collect a 1000 samples  from the server
    var myArray:Array= new Array(1000);
    myRandomData = new ArrayCollection(myArray);

    }
    //Function to disconnect to the remote ECG
   public function disconnect():void
    {
            try{
    socket.close();
     msg("connection closed...\n");
                }
                catch(error:Error)
                {
                }
     }

  // This method is called if the socket finds an ioError event.

public function ioErrorHandler(event:IOErrorEvent):void {
    msg("Unable to connect: socket error.\n");
}


// This method display the connection message.
```

```
        private function connectHandler(event:Event):void {
            if (socket.connected) {
                msg("connected...\n");
            } else {
                msg("unable to connect\n");
            }
        }

        /* This method is used when the connection is closed by
           the server.*/
        private function closeHandler(event:Event):void {
            msg("connection closed by the server...\n");
        }
        //  This method is used if the socket got an error.

        private function errorHandler(event:ErrorEvent):void {
            msg(event.text + "\n");
        }

        // This method is used to handle the data from the server.

            private function dataHandler(event:ProgressEvent):void {
            var n:int = socket.bytesAvailable;

        /* the variable len is = to 11 due the message package is 9 (hr1) for
        amplitude and 2 (hr2) for the heart rate ult is the last index of the array
        pri is first index */

            var len:uint=11;
            var ult:int=999;
            var pri:int=0;
            var hr1:Number=9;
            var hr2:Number=2;
         /* the operation will be performed while there is a valid package message of 11
           characters and read them as UTF bytes. The portion of the amplitude and heart
rate
           are extracted and displayed in their respetive graphical display.
         */
            while (--n >= 11) {


                    try{

                var str3:String=socket.readUTFBytes(len);
                var hr:String=str3.substr(hr1,hr2);
                var myECG:Number = Number(str3);

         /* the data array is updated by removing the first element and adding one
         at the end */
                 myRandomData.removeItemAt(pri);
```

```
        myRandomData.addItemAt(myECG,ult);
        heart_rate.text=hr;
//Display the incomming values in the text area


                msg(str3+"\n");

    }

    // Catch undesirable delayed data errors
     catch(error:Error)
     {

     }


  }
}


/*
  Add the data  to the TextArea.
  After adding text, the setScroll() method controls
  the scrolling of the TextArea.
 */
private function msg(value:String):void {
    ta.text += value;
    ta.dispatchEvent(new Event(Event.CHANGE));
    setTimeout(setScroll, 100);
}

/*
  Scroll the TextArea to its maximum vertical scroll
  and the TextArea always shows the last line returned
  from the ECG server.
 */
public function setScroll():void {
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}

            ]]>
   </mx:Script>
```

II. MXML CODING

```
 <mx:SeriesInterpolate id="eff"/>
   <mx:Label id="title" text="ECG Socket-Monitoring" fontSize="24" fontStyle="bold"
color="#F1F7F8"/>
   <mx:ApplicationControlBar width="100%">
      <mx:Label text="Server:"  color="#F0F6F7"/>
      <mx:TextInput id="serverName" width="100%" text="localhost" />
      <mx:Spacer />
      <mx:Label text="Port:"  color="#F1F7F8"/>
      <mx:TextInput id="portNumber" text="2055" textAlign="right" maxChars="5"
restrict="0-9" />
      <mx:Spacer />
      <mx:Button label="Start" click="connect();"  color="#F2F7F8"/>
      <mx:Button label="Stop" click="disconnect()"  color="#F2F8F9"/>
   </mx:ApplicationControlBar>

   <mx:Spacer />
  <mx:Panel id="console" enabled="false" width="100%" height="100%"
paddingTop="10" paddingBottom="10" paddingLeft="10" paddingRight="10"
cornerRadius="7" backgroundColor="#373434" color="#F4F9FA">
        <mx:Canvas width="100%" height="100%">
              <mx:Label text="HEART RATE" width="168" fontSize="20"
textAlign="center" y="24"/>
              <mx:TextArea id="heart_rate"  width="147" color="#080909"
fontSize="30" fontWeight="bold" fontStyle="italic" textAlign="center" height="75"
x="10" y="76"/>
              <mx:Label x="367" y="10" text="SERVER MESSAGES" width="221"
textAlign="center" fontSize="20"/>
              <mx:TextArea id="output" editable="false" width="251" height="141"
fontFamily="Courier New"  color="#1D19CF" fontSize="10" x="376" y="49"/>
              <mx:LineChart id="linechart1" height="150" width="100%"
dataProvider="{myRandomData}" color="#EEF5F7" x="-10" y="198">
                    <mx:series>
                          <mx:LineSeries displayName="Random"
showDataEffect="{eff}" form="curve"/>
                    </mx:series>
              </mx:LineChart>
        </mx:Canvas>
   </mx:Panel>
</mx:Application>
```