McMaster University
DigitalCommons@McMaster

EE 4BI6 Electrical Engineering Biomedical Capstones

Department of Electrical and Computer Engineering

4-9-2010

Design of a Non-Invasive Blood Pressure Measurement System with Automated Communication of Information for Remote Health Monitoring

Rohinton Richard
McMaster University

Follow this and additional works at: http://digitalcommons.mcmaster.ca/ee4bi6

Part of the Biomedical Commons

Recommended Citation
Richard, Rohinton, "Design of a Non-Invasive Blood Pressure Measurement System with Automated Communication of Information for Remote Health Monitoring" (2010). *EE 4BI6 Electrical Engineering Biomedical Capstones*. Paper 31.
http://digitalcommons.mcmaster.ca/ee4bi6/31

This Capstone is brought to you for free and open access by the Department of Electrical and Computer Engineering at DigitalCommons@McMaster. It has been accepted for inclusion in EE 4BI6 Electrical Engineering Biomedical Capstones by an authorized administrator of DigitalCommons@McMaster. For more information, please contact scom@mcmaster.ca.

Design of a Non-Invasive Blood Pressure Measurement System with Automated Communication of Information for Remote Health Monitoring

by

Rohinton Richard

Electrical and Biomedical Engineering Design Project (4BI6)

Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, Canada

Design of a Non-Invasive Blood Pressure Measurement System with Automated Communication of Information for Remote Health Monitoring

by

Rohinton Richard

Electrical and Biomedical Engineering
Faculty Advisor: Prof. Jeremic

Electrical and Biomedical Engineering Project Report
submitted in partial fulfillment of the degree of
Bachelor of Engineering

McMaster University
Hamilton, Ontario, Canada
April 9, 2010

Copyright © April 2010 by Rohinton Richard

Abstract

Many individuals that suffer from hypertension have to take their blood pressure readings daily to ensure that they maintain their blood pressure level within acceptable limits. Patients currently have to write down their blood pressure manually every time it is taken and have to manually send the information to their healthcare provider (either electronically, or by bringing a log book to the next scheduled check-up). The objective of this project was to design a non-invasive blood pressure measurement system that would automatically communicate necessary information to healthcare providers for remote health monitoring. The apparatus developed in this project is a user-friendly "one-click" operational blood pressure monitoring system that automatically takes a user's blood pressure measurement, transfers the data to a computer, stamps it with the date and time, stores it in a database, analyzes and graphs the data, and finally emails the information to the healthcare provider who is in charge of monitoring the patient. The theory behind the system, hardware acquisition, software design, and the experimental results are presented.

Key words: blood pressure monitor, remote health monitoring, digital sphygmomanometer, BP Transmit, automated communication of information

Acknowledgements

The author would like to thank Dr. Aleksander Jeremic and Dr. Thomas E. Doyle of the Electrical and Computer Engineering Department at McMaster University for their encouragement and guidance over the course of this project. The author would also like to thank Mr. Brandon Katz, Product Manager at A&D Medical in San Jose, CA for his company's support and donation of the UA-767PC blood pressure monitor to this project.

Contents

Abstract **ii**

Acknowledgements **iii**

Contents **iv**

List of Figures **vi**

1 Introduction **1**
1.1 Background 1
1.2 Objective 1
1.3 General Approach to the Problem 2
1.4 Scope of the Project 2

2 Literature Review **3**
2.1 History of Blood Pressure Technology 3
2.2 Current Blood Pressure Statistics 4

3 Problem and Methodology of Solution **5**
3.1 Statement of Problem 5
3.2 Methodology of Solution 5

4 Experimental and Design Procedures **7**
4.1 Graphical User Interface Design 7
4.2 Program Setup Design 9
4.3 Device Communication Design 13
4.4 Data Analysis Design 15
4.5 Information Transfer Design 17
4.6 System Safety Measures 18

5 Results and Discussion **19**
5.1 Graphical Results 19
5.2 Discussion 23

6 Conclusion **25**
6.1 Concluding Statements 25
6.2 Recommendations for Future Work 25

Appendix A: Algorithms **26**
A.1 Algorithm for Finding Available COM Ports 27
A.2 Algorithm for Synchronizing Device Clock 28
A.3 Algorithm for SetClock Subroutine 29
A.4 Algorithm for HextoSend Subroutine 31
A.5 Algorithm for CheckSum Subroutine 32
A.6 Algorithm for Parsing Blood Pressure Data 33
A.7 Algorithm for Sending Data for Analysis 34
A.8 Algorithm for Age Calculation 35
A.9 Algorithm for Sending Email 36
A.10 Algorithm for Checking Internet Connectivity 37

Appendix B: Components **38**
B.1 Components List 39
B.2 Transmission Protocol 40

References **41**

Vitae **42**

List of Figures

4.1 Introduction form 9
4.2 Destination location form 10
4.3 Patient information form 12
4.4 Installation complete form 13
4.5 Graph of a 7-day trend 17

5.1 Graph of a 14-day trend 20
5.2 Graph of a 30-day trend 20
5.3 Patient information worksheet 22
5.4 BP Transmit email message 23

Chapter 1

Introduction

1.1 Background

The technology and significance of blood pressure measurements have come a long way from their humble origins back in 1733 when Rev. Stephen Hales first measured mean blood pressure in an unanesthetized horse using a long, glass tube [1]. Today, blood pressure is one of the most important physiological measurements in clinical medicine, and also one of the most unreliable [2]. Healthcare professionals are constantly seeking more accurate, efficient, and convenient ways to measure it [3]. Relatively accurate digital sphygmomanometers and blood pressure monitors currently exist on the market, but remote health monitoring is a growing field. Similar products that have been researched or are on the market have only brought raw blood pressure data into a computer, but have not automated the process of sending information to a healthcare provider. Research is being conducted at healthcare centres and universities, including McMaster University, to improve remote health monitoring, but currently there is no known blood pressure monitor on the market that automatically emails blood pressure information to a healthcare provider, such as a family doctor.

1.2 Objective

Many individuals that suffer from hypertension have to take their blood pressure readings daily to ensure that they keep their blood pressure level within acceptable limits, with the help of exercise, a healthy diet, and prescription medication [4]. The objective of this project was to design a non-invasive blood pressure measurement system that would automatically communicate necessary information to healthcare providers for remote

health monitoring. Instead of having a patient with hypertension write down their blood pressure manually every time it is taken and having to manually send the information to the healthcare provider (either electronically, or by bringing a log book during the next scheduled check-up), this project automates that process so that the patient's daily testing will be less of a hassle, and the healthcare provider will receive the necessary information quickly and conveniently. This project will hopefully contribute valuable technology to the growing remote health monitoring field.

1.3 General Approach to the Problem

An overview of the general methodology is provided in this section. Specific details are provided in the "Experimental and Design Procedures" section of the report. The first step in this project involved obtaining a digital sphygmomanometer with a computer interface. Such products exist on the market and are relatively affordable to purchase. Software was then developed to interact with the device via its computer interface. The coding was written in Visual Basic .NET due to the fact that Microsoft Excel 2007 was used as a database. The application was designed to be a one-click program; that is, only one mouse click is needed to obtain the blood pressure measurement and perform any of the other operations that were required. The software then processes the data to obtain information of interest, and creates trends of the blood pressure measurements over specified numbers of days. The software then connects to Yahoo's simple mail transfer protocol server to initiate an email send. The recent blood pressure measurement along with the trends are emailed to the patient's healthcare provider.

1.4 Scope of the Project

The scope of this project covers individuals aged twenty and older with hypertension who have upper arm circumferences between approximately 24 cm and 36 cm. This includes an estimated 1 billion adults worldwide [5].

Chapter 2

Literature Review

2.1 History of Blood Pressure Technology

The first recorded instance of blood pressure being taken occurred in 1733 when Rev. Stephen Hales measured mean blood pressure in an unanesthetized horse using a long, glass tube [1]. Until 1855, however, no non-invasive methods for measuring blood pressure were developed. In 1855, Vierordt of Tubingen theorized that an indirect, non-invasive technique could be used by measuring the counter pressure which is needed to cause the pulsation in an artery to cease [6]. He thus ended up developing the first sphygmograph. This device was further improved upon by individuals, such as Etienne Jules Marey, and R.E. Dudgeon. The first accurate estimation of blood pressure was made in 1856 by the surgeon Faivre, who connected an artery to a mercury manometer during an operation [6]. It was Samuel von Basch, however, who decided not to obtain blood pressure measurements from an arterial puncture, but decided rather to obtain a direct registration of the blood pressure by a column of fluid. He used an inflatable rubber bag which was filled with water and tightly connected to the neck of a manometer bulb that was filled with mercury [7]. Thus, the first sphygmomanometer was developed. The most common method of recording blood pressure measurement today – the auscultatory method – is due largely to the work of the Russian surgeon, N.C. Korotkoff, who reported that by placing a stethoscope over the brachial artery at the cuboidal fossa, tapping sounds could be heard as the cuff was deflated [7]. The technique is widely used today and is the method used for obtaining blood pressure measurements by the digital monitor used in this project.

2.2 Current Blood Pressure Statistics

The present statistics regarding individuals with high blood pressure are shocking. About 31.3% of all American adults aged twenty and over have high blood pressure (hypertension) and/or are taking blood pressure-lowering medication [8]. In sheer numbers, that is about 95 million people in the United States alone! An estimated 1 billion adults worldwide have hypertension [5]. In 2005, high blood pressure was noted as a primary or contributing cause of death for 319,000 Americans [8], and it is estimated that about 90% of middle-aged adults will develop hypertension in the remainder of their lifetime [8]. As a result, many individuals have to take their blood pressure readings daily to ensure that they keep their blood pressure level within acceptable limits, with the help of exercise, a healthy diet, and prescription medication [4]. Engineering principles will have to be applied to find a way to ameliorate this process.

Chapter 3

Problem and Methodology of Solution

3.1 Statement of Problem

Many individuals that suffer from hypertension have to take their blood pressure readings daily to ensure that they keep their blood pressure level within acceptable limits. Patients currently have to write down their blood pressure manually every time it is taken and have to manually send the information to the healthcare provider (either electronically, or by bringing a log book to the next scheduled check-up). Research is being conducted at healthcare centres and universities, including McMaster University, to improve remote health monitoring, but currently there is no known blood pressure monitor on the market that automatically emails blood pressure information to a healthcare provider, such as a family doctor.

3.2 Methodology of Solution

To ameliorate the process of communicating patient blood pressure information with their healthcare provider, this project automatically collects, analyzes, and transfers the patient's daily blood pressure reading, so that it will be less of a hassle to the patient, and the healthcare provider will receive the necessary information quickly and conveniently. The first step in this project began by selecting an appropriate digital sphygmomanometer with a computer interface. After intensive research regarding the suitability of various blood pressure monitors for this project was conducted, it was determined that A&D Medical's UA-767PC Blood Pressure Monitor was the most appropriate and would be obtained. The device conforms to European Directive 93/42 EEC for Medical Products, the statutory EMC directive 89/336/EEC, complies with part 15 of the FCC rules, and

contains the FCC ID POOWML-C40. The UA-767PC Blood Pressure Monitor is also clinically validated according to the Association for Advancement of Medical Instrumentation (AAMI) and the British Hypertension Society (BHS). A&D Medical was contacted towards the end of October 2009, and the project proposal was sent to Brandon Katz, Product Manager at A&D Medical in San Jose. The company determined that they would be able to donate one UA-767PC device along with a universal serial bus (USB) cable to this project. The device and cable were received in the middle of November 2009. A 9-pin RS-232C serial cable was obtained later that year in December. The decision for switching from a USB interface to a 9-pin RS-232C serial connection is detailed in the discussion section of this report. Once the hardware was acquired, the software had to establish connectivity with the device. After a very long and frustrating process, connectivity was established with the device using Microsoft Visual Basic .NET and some RS-232C hex commands. Software then had to be developed to process the data and provide information to the user. The design of the graphical user interface of the application developed is detailed extensively in the "Experimental and Design Procedures" section of the report. The application is designed to be a "one-click" solution, which automatically obtains the user's blood pressure measurement and performs all necessary tasks with the data. The software then processes the blood pressure data and analyzes it in Excel, plotting trends over seven, fourteen, and thirty days. The software then emails the information to the healthcare provider using the collaboration data objects system (also detailed in the "Experimental and Design Procedures" section of the report). Another installation program to install the application was also developed for this project, and a specially-formatted Excel workbook was created as well.

Chapter 4

Experimental and Design Procedures

The apparatus developed in this project was designed to be a user-friendly "one-click" operational blood pressure monitoring system that would automatically take a user's blood pressure measurement, transfer the data to a computer, stamp it with the date and time, store it in a database, analyze and graph the data, and finally email the information to the healthcare provider who is in charge of monitoring the patient. The design procedures described in this section were used to methodically implement this system.

4.1 Graphical User Interface Design

The graphical user interfaces (GUI) of both the application and the setup program were developed in the Visual Basic .NET programming language using Microsoft Visual Basic 2008 Express Edition. The graphical user interface of the application was designed before that of the setup program. Since the system was designed to be a "one-click" system, one prominent start button was needed on the main form of the application. The start button was linked to all of the necessary operations that needed to be performed. The development of the code that performs the individual operations is described in later subsections of this report. A large textbox was also added to the form as the main display to provide the user with up-to-the-second information as the program performed its operations. In terms of the patient's actual blood pressure reading, three labels were placed on the form to inform the user of their systolic and diastolic pressure, as well as their heart rate. These labels were added to allow the patient to keep their focus on the computer monitor to view their blood pressure reading, instead of having to shift focus between the computer monitor and the display screen of the blood pressure monitor. A

progress bar was also added to provide the user with some information as to how much of the data gathering/processing/analyzing/transferring has been completed. Another less-prominent button was added to the form to link the main form to the device settings form.

The settings form displays all vital settings needed for the system to perform accurately. The COM port is the most important setting listed on the form. The program automatically finds all available COM ports on the computer and lists them in a combo box, when the settings form is loaded. This is accomplished using a for-loop that searches all the serial port names on the computer and adds the available serial ports to the combo box (*cf.* Appendix A.1, page 27). The default COM port on a Dell Inspiron 6400 laptop is COM4. The settings form also informs the user of the location of the data file, which is the default directory "C:\BP-Transmit\." The patient's name as entered by the user during installation of the application, as well as the healthcare provider's email address, is also listed. This allows the user to verify that the email address that the information is being sent to is correct, and that the healthcare provider receiving the information is receiving it under the correct patient name. Both the patient's name and healthcare provider's email address are retrieved from the Excel data file. As the settings form loads, the Excel file is opened in the background, and the necessary information is retrieved from their respective cells. One last button is also placed on the settings form to synchronize the digital blood pressure monitor's device clock with that of the computer's clock. This is particularly useful if the system has just been removed from the box or has been transferred to a different time zone, and while the date, time, and time zone can be easily adjusted on a computer, the clock on the digital blood pressure monitor can be a bit of a hassle to adjust manually.

The synchronization of the device clock was programmatically intensive to implement. As with any other communication with the UA-767PC blood pressure monitor used in this system, all data is transferred and received using RS-232C command sets as hex strings. To synchronize the clock, the device needs to first enter Communication Mode from Stand-by Mode, a "set time and date" command then needs to be sent, and finally the current time and date need to be deconstructed into byte arrays and sent to the device (*cf.* Appendices A.2 – A.5, pages 28-32). The details of the RS-

232C command sets sent are described in detail in the Device Communication Design subsection of the report.

4.2 Program Setup Design

A setup file was developed to install the BP Transmit application on a user's computer. The setup file consists of four forms within one executable file. The first form is the introduction screen of the installation program and contains text, graphics, and two buttons – one to advance to the next form and the other to exit the program (*cf.* Figure 4.1 on page 9). The text notifies the user that BP Transmit will be installed if the user continues with the setup wizard, and also informs the user of connectivity and supporting applications, such as Microsoft Excel, that are needed prior to installing BP Transmit.

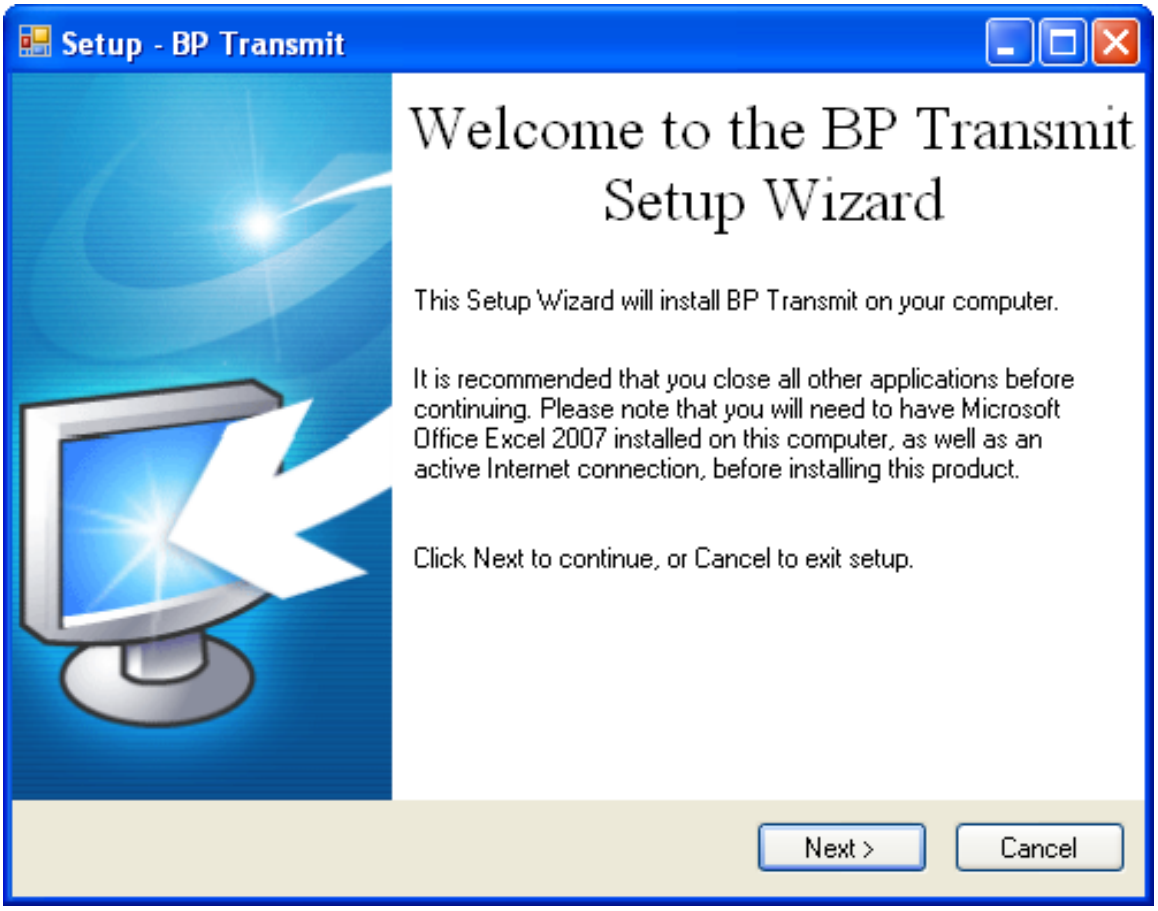


Figure 4.1: The introduction form of the setup file.

The second form notifies the user of the destination directory of the data files – that is, where the program will store the data files on the user's computer (*cf.* Figure 4.2 on page 10). The default directory is "C:\BP-Transmit." It is at this point that the user can decide to install the application, go back to the first form, or cancel the installation. There is a button on the form for each of the three options. If the user decides to proceed with the installation, the program ensures that previous patient data files do not exist in that destination directory. If previous patient data files exist, the program warns the user that continuing the installation will result in the existing data files being overwritten. If the user decides to continue, the blank Excel workbook from the installation folder, which contains sheets for the various trends that will eventually be plotted, is copied to the destination directory. During this step of the installation process, the BP Transmit application is also placed on the user's desktop.

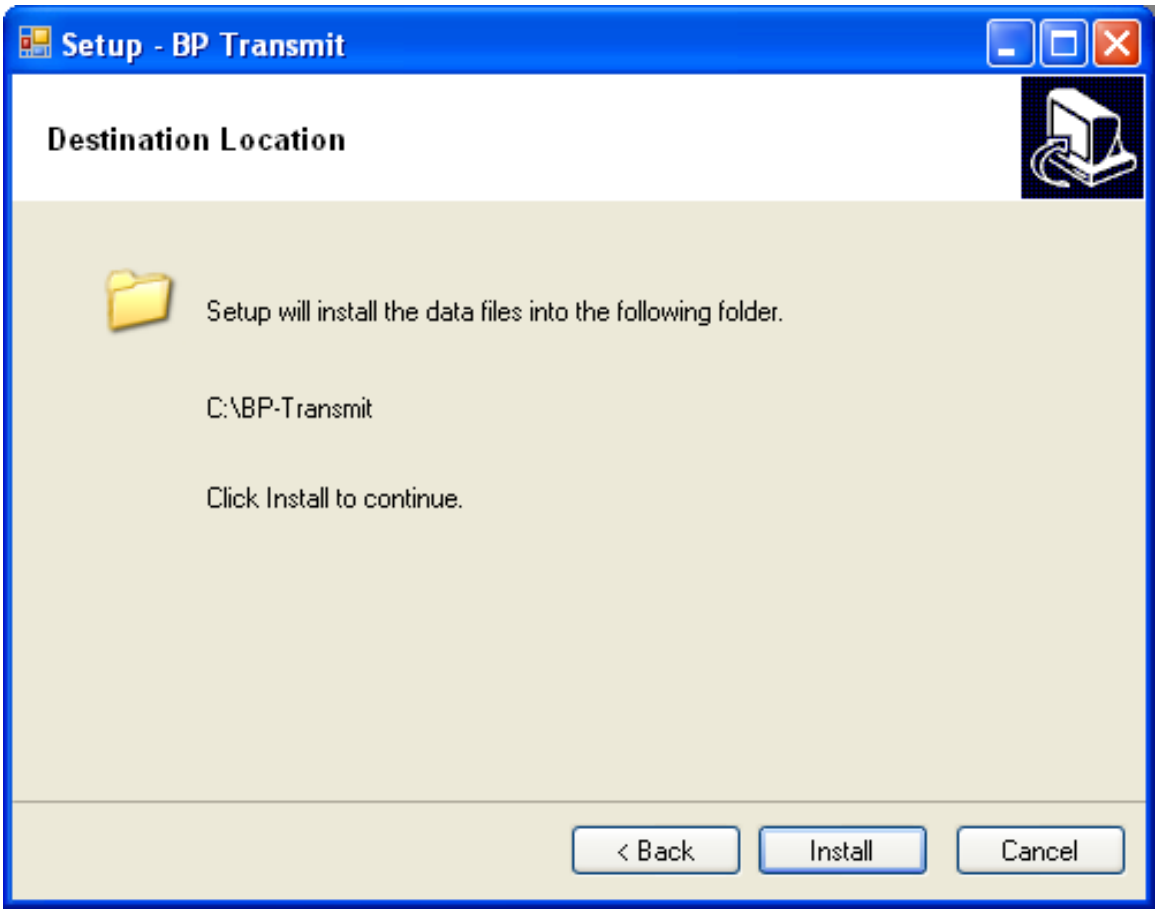


Figure 4.2: The destination location form of the setup file.

The third form is used to gather the patient information. The user has to input the patient's first name, last name, date of birth, patient identification information (if applicable), height, and weight into textboxes on the form (*cf.* Figure 4.3 on page 11). The form allows for a lot of freedom in terms of user input. The only restriction applied to the textboxes containing the first name and last name is that they should not be empty when the form is submitted. The user also has the flexibility to enter the height in either centimetres or inches, and the weight in either kilograms or pounds, as long as the units are selected by enabling the proper radio buttons and the values are not extreme. Ensuring a legitimate date of birth, however, was programmatically challenging to implement. The program had to ensure that the date was legitimate; otherwise, when it is transferred to Excel, the worksheet does not recognize the date and throws an error. Ensuring the year, month, and day categories met certain values required many conditional statements, but was not challenging in and of itself. However, ensuring that February 29 was accepted as a date of birth only on certain years that are leap years was challenging. Finally, after trying various calculations, it was determined that if a numerical year was divisible by four, it had a possibility of being a leap year, given that it was not divisible by 100. If it was divisible by 100 as well, then it had to be divisible by 400 also. Once an algorithm was developed to determine whether a particular year was a leap year, the process continued smoothly.

After the user has input all the necessary data and clicks the "Finish" button, a message box pops-up with the information the user has just entered and asks the user to confirm all the information. If the information is confirmed, it is stored in the Excel data file.

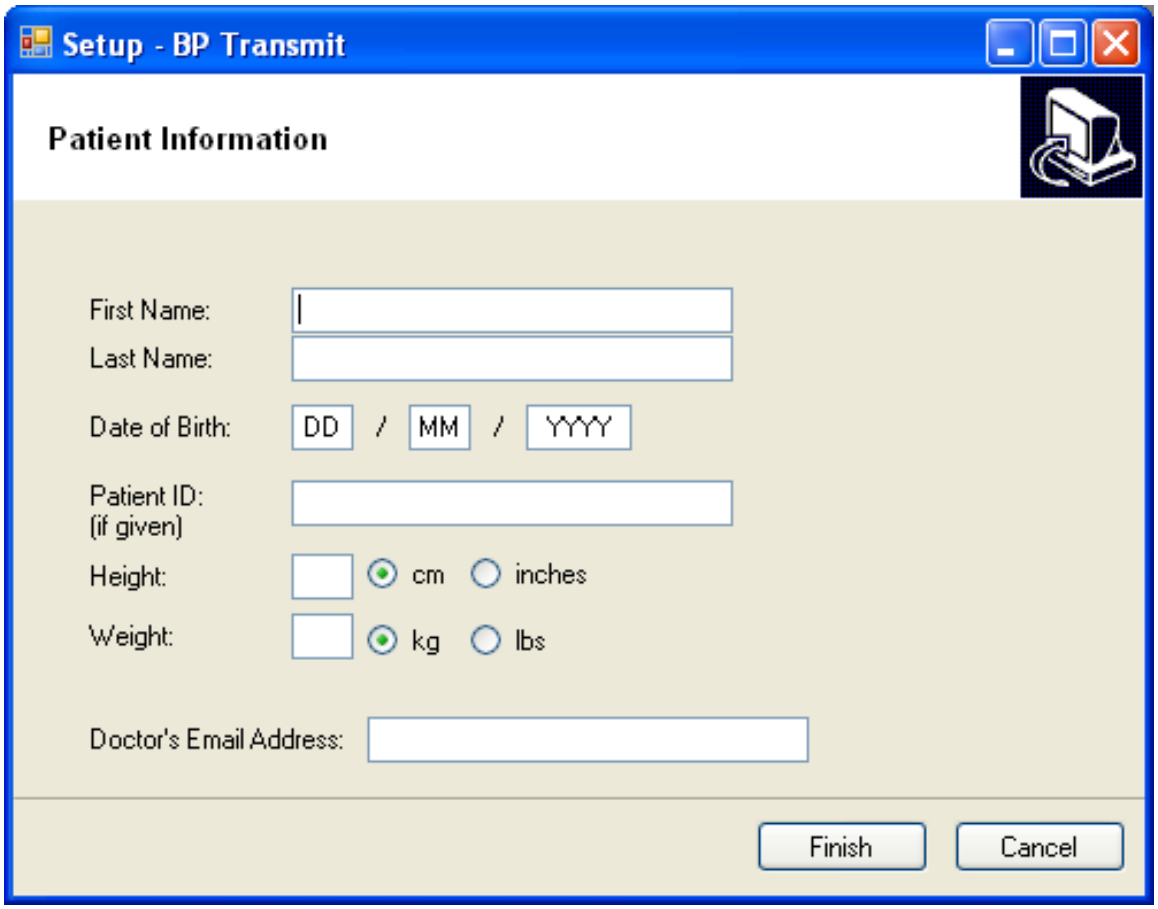


Figure 4.3: The patient information form of the setup file.

The fourth form informs the user that the installation is complete. The user has now successfully installed BP Transmit on their computer, and they may access it any time by selecting the application on their desktop. Since this is the last form of the setup file, it contains text and one button to close the setup program (*cf.* Figure 4.4 on page 13).

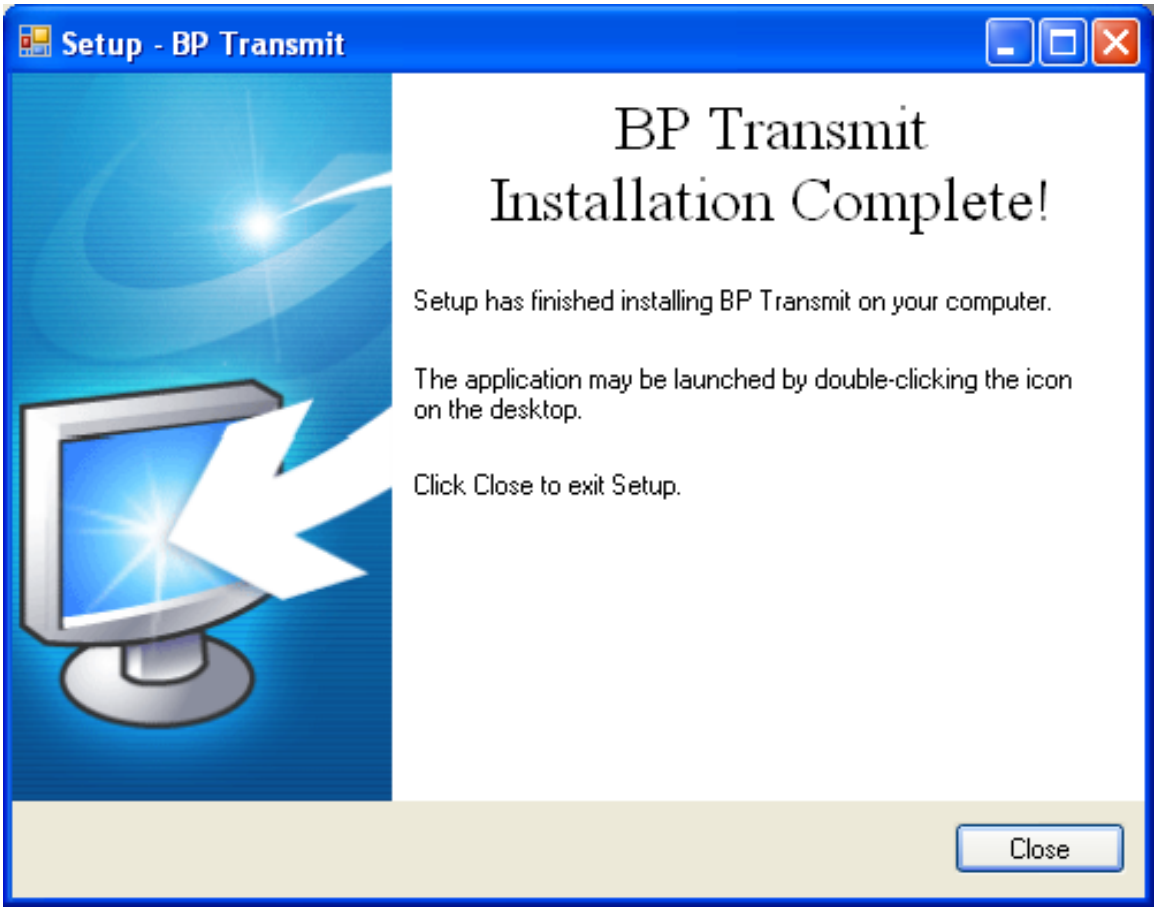


Figure 4.4: The installation complete form of the setup file.

4.3 Device Communication Design

The UA-767PC blood pressure monitor connects to a computer using a 9-pin RS-232C serial cable, which requires specific command sets to be sent as ASCII hex strings to perform any operation. In order to connect to the device using a .NET interface, a serial port component needs to be added to the form. Once the component was added, the transmission protocol parameters were coded to initiate device communication when the start button of the application is clicked (*cf.* Appendix B.2, page 40). The application then sends an arbitrary "Wake" byte to the blood pressure monitor. This sends an active-low signal to the blood pressure monitor, which then transfers the device from Stand-by Mode to Communication Mode. A small delay is then coded to ensure that the device is given enough time to transfer into Communication Mode. After the device has successfully transferred into Communication Mode, an "open" command is issued. The open

command consists of seven bytes – [02 43 50 43 30 35 3B]. The first byte [02] informs the device that the incoming message is in the form of command data and not control data, which is represented by [01]. The second byte is fixed [43] and represents the letter C (for Command) in ASCII. The next two bytes represent "PC" in ASCII and state where the data is being sent from. The two bytes after that are the command code. The bytes [30 35] represent the "open communication port" command. The last byte is the "Check Sum" byte, which is calculated by taking the sum of all the previous bytes (excluding the first byte) and retaining the least significant byte. The "Check Sum" byte is specifically calculated in the code for the synchronization of the device clock (*cf.* Appendix A.5, page 32). The entire command is sent as one byte array as follows:

```
Dim open() As Byte = {&H2, &H43, &H50, &H43, &H30, &H35, &H3B}
SerialPort1.Write(open, 0, open.Length)
```

After the device's communication port has successfully transferred to the open state, an "ACK" message is received from the blood pressure monitor. Next, another byte array needs to be sent to obtain a blood pressure reading. Most of the bytes sent are similar to that of the previous byte array, but the bytes of the command code are now [34 30], which represent the "start blood pressure measurement" command. A long pause is implemented in the code after this command is sent to give the blood pressure monitor enough time to inflate the cuff, obtain a blood pressure reading from the patient, deflate the cuff, and transfer the data to the computer. The blood pressure reading is then received as another byte array. This array then needs to be parsed to obtain the blood pressure information. Eight bytes are received from the device, but only the second, third, and fourth byte contain useful information for the purposes of this project. The parsing of the data is described in the Data Analysis Design subsection of the report.

As previously mentioned in the report, the synchronization of the device clock was programmatically intensive to implement (*cf.* Appendix A.2, page 28). Unlike open or close commands, which send a constant set of bytes to perform the operation regardless of when they are sent, the synchronization of the device clock varies based on the dynamic properties of date and time. To set the device clock, a byte array needs to be sent with a command code of [33 31]. Once that has been received by the device, the

SetClock subroutine is called (*cf.* Appendix A.3, page 29). The SetClock subroutine obtains the current date and time and divides it into the individual components of year, month, day, hour, and minute. Each component is then converted into RS-232C hex format using the subroutine HextoSend (*cf.* Appendix A.4, page 31). The hex values are then sent to the CheckSum subroutine to calculate the "Check Sum" byte for the command string. All of the data is then sent as a data byte array, which differs from a command byte array. A data byte array consists of twenty bytes as opposed to seven. The first byte [02] once again informs the device that the incoming message is in the form of command data and not control data. The second byte is fixed [44] and represents the letter D (for Data) in ASCII. The next two bytes represent "PC" in ASCII and state where the data is being sent from. The four bytes that follow represent the length of the request data being sent. In this case, ten data bytes are sent – two each for the year, month, day, hour, and minute. Another fixed byte of "0" in ASCII is then appended, followed by the ten request data bytes. Lastly, the "Check Sum" byte is calculated and attached to the end of the byte array. The following byte array is sent:

```
Dim clock() As Byte = {&H2, &H44, &H50, &H43, &H30, &H30, &H30, &H41,
                       &H30, year1, year2, &H30, month1, day1, day2, hour1, hour2,
                       minute1, minute2, sum}
```

After all necessary information has been obtained from the blood pressure monitor, the communication channel is closed by sending an "end of transmission" command to the device. This is also similar to the previous byte arrays sent, but contains the command code [30 34].

4.4 Data Analysis Design

Once the data is gathered from the blood pressure monitor, it needs to be analyzed to produce useful information. As previously mentioned, eight bytes are received from the blood pressure monitor, but only the second, third, and fourth byte contain useful information for the purposes of this project. The second byte contains the hex value of the pulse pressure, which is the difference between the systolic and diastolic pressures. The third byte contains the value of the diastolic pressure, and the fourth byte contains pulse

information. The information is obtained using typical parse methods to trim away the non-essential information and extract the essential information (*cf.* Appendix A.6, page 33). The information then needs to be input into the Excel data file to be graphed into trends over seven, fourteen, and thirty days. The variables to access the Excel application, workbook, and worksheet are defined as follows:

```
Dim Excel As New Microsoft.Office.Interop.Excel.Application
Dim WorkBook As Microsoft.Office.Interop.Excel.Workbook
Dim Sheet As Microsoft.Office.Interop.Excel.Worksheet
```

Individual sheets are then accessed and the current data is input into the appropriate cells. The fourth worksheet is the database and that is updated first with the current day, time, systolic pressure, diastolic pressure, and pulse. The first worksheet graphs a 7-day trend of the patient's blood pressure information. The program inputs the date and new blood pressure information at the top of the list and systematically transfers the existing blood pressure information to their respective cells below. The program also ensures that any existing blood pressure information that is more than seven days old is cleared. This is accomplished by measuring the difference in date between the current date and the date of the old blood pressure information using the DateDiff property. The complete code for transferring the data to the Excel workbook to plot the 7-day trend is detailed in Appendix A.7 on page 34. The Excel worksheets have already been formatted by the developer of this project prior to installation to plot the data placed in certain cells. Fully-labelled graphs with legends and trend lines are already created on the worksheets and are automatically updated as new information is placed in the data cells. The patient's body mass index (BMI) and age are also automatically calculated in the patient information worksheet after installation using in-cell formulae (*cf.* Appendix A.8, page 35). Similar coding was also implemented for the 14-day and 30-day trends in the Excel workbook. All the information is neatly detailed and organized, and the resulting trends are plotted in a visually appealing manner to supply the healthcare provider with a user-friendly document detailing the important blood pressure information of their patient (*cf.* Figure 4.5 on page 17).

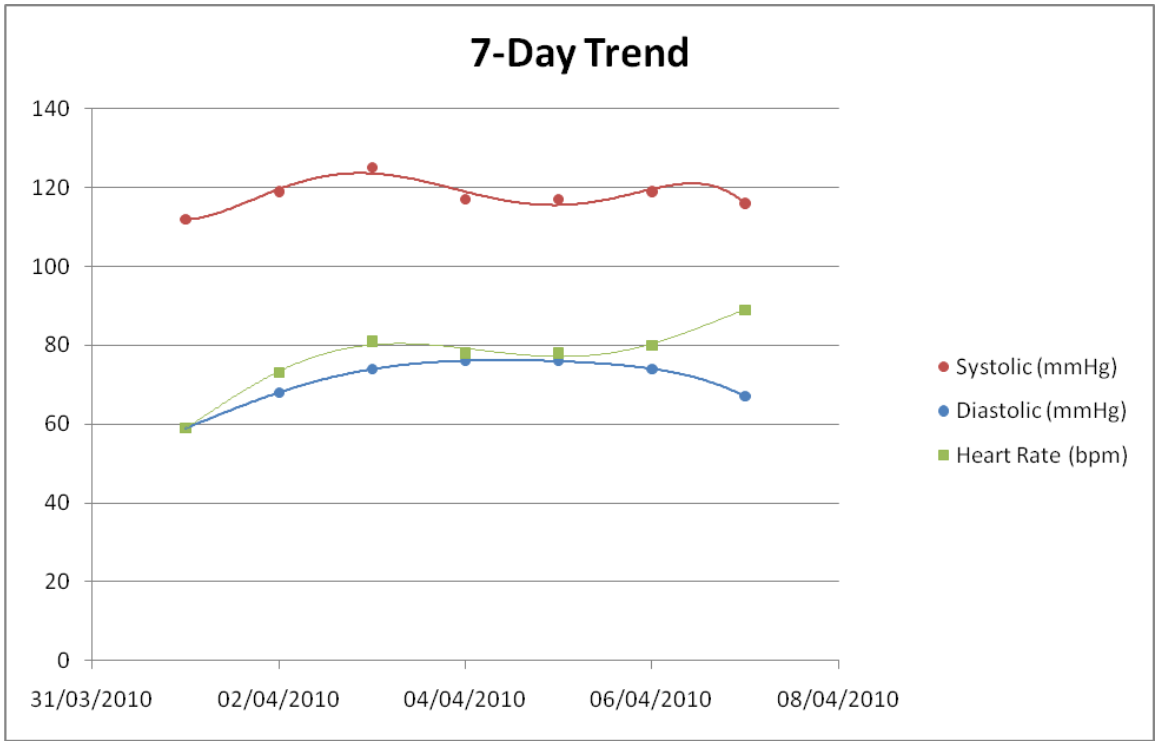


Figure 4.5: Graph of a 7-day trend containing systolic, diastolic, and heart rate values.

4.5 Information Transfer Design

After the information has been obtained, processed, and analyzed, it has to be sent to the patient's healthcare provider via email. The program first ensures that an active Internet connection is available before trying to send the information. This check is conducted by having the program try to access the Yahoo! website and check the response. If the program is able to connect to the website, the next step in the information transfer process is initiated. If the test fails however, then the user is notified of the failure, and the information is not emailed to the healthcare provider.

The next step in the transfer process is detailed in the SendMail subroutine of the application. Sending email for this project was accomplished using the collaboration data objects (CDO) system, which is available on most Windows operating systems provided that the simple mail transfer protocol (SMTP) service in Internet Information Server is installed. CDO allows the application to access email servers, such as the Yahoo! SMTP server, and send email messages. Particular settings to access the Yahoo! SMTP server are provided in the code in Appendix A.9 on page 36. A Yahoo! Mail account was

created for this application to send out email. The subject of the email contains the patient's name. The body of the email message contains the patient's systolic and diastolic pressure readings, as well as their pulse, and appends the Excel file with the trends as an attachment. This message is sent to the healthcare provider's email address as provided by the user during setup.

4.6 System Safety Measures

Numerous safety measures were implemented into the software. Almost all of the executable code has been incorporated into try-catch blocks. The application also has additional safeguards for each particular action. For example, the application checks to see whether an active Internet connection exists prior to sending email (*cf.* Appendix A.10, page 37) to prevent an error from occurring. The application also checks to see whether the directory of the data file and the Excel data file itself actually exist before taking a user's blood pressure measurement. This ensures that the user does not end up taking an unnecessary measurement that cannot be stored, and also informs the user that their file has been moved, misplaced, or deleted. There are also specific exceptions that are caught when data communication with the device fails, email fails to send, the data file is not found, or a general error has occurred. In all situations, the device connection is closed, and the patient is not exposed to any unnecessary risks from device errors. The application was rigorously tested and no errors were detected during testing. The UA-767PC Blood Pressure Monitor is also clinically validated according to the Association for Advancement of Medical Instrumentation and the British Hypertension Society for accuracy and reliability.

Chapter 5

Results and Discussion

This project resulted in the development of the BP Transmit application and its installation program. The user-friendly "one-click" operational blood pressure monitoring system developed in this project was successfully able to automatically take a user's blood pressure measurement, transfer the data to a computer, stamp it with the date and time, store it in a database, analyze and graph the data, and finally email the information to the healthcare provider who is in charge of monitoring the patient. The graphical results from patient blood pressure measurements taken with the BP Transmit system, and a discussion on project difficulties and why certain methods of data communication were chosen are described in this section.

5.1 Graphical Results

As shown in Figure 4.5 on page 17, the application produces fully-labelled graphs with legends and trend lines. The patient's data is not only plotted over seven days, but also over fourteen days (*cf.* Figure 5.1 on page 20) and thirty days (*cf.* Figure 5.2 on page 20).

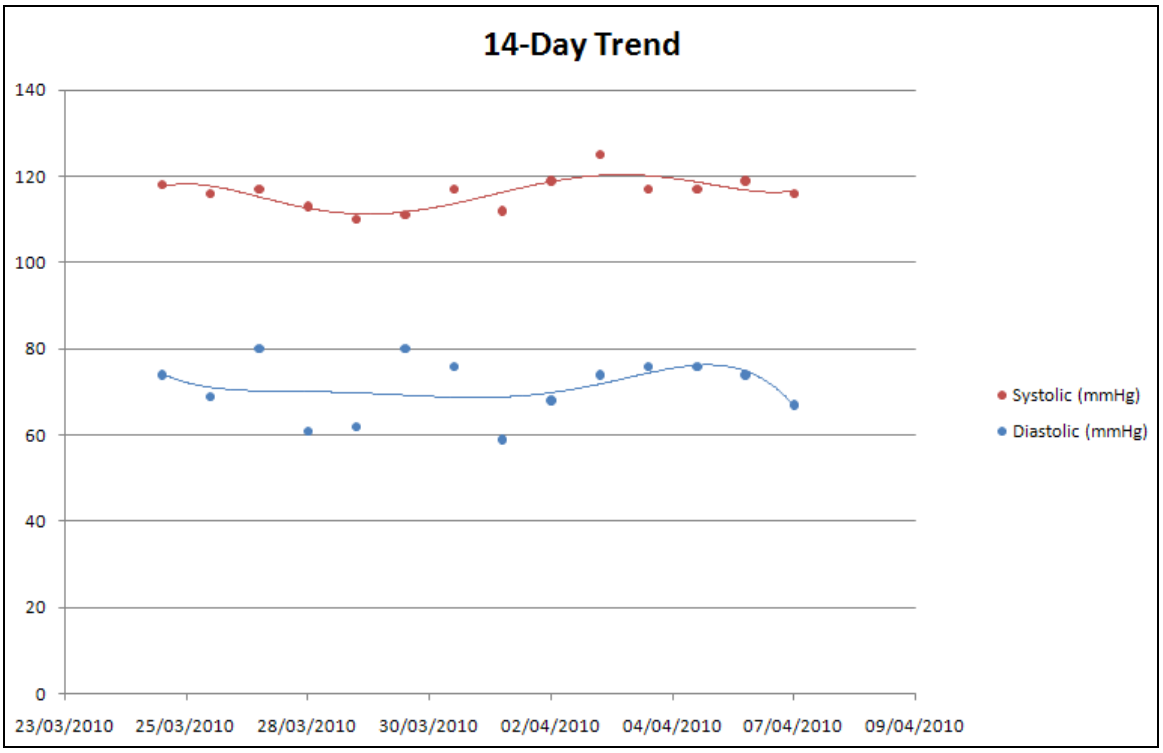


Figure 5.1: Graph of a 14-day trend containing systolic, diastolic, and heart rate values.

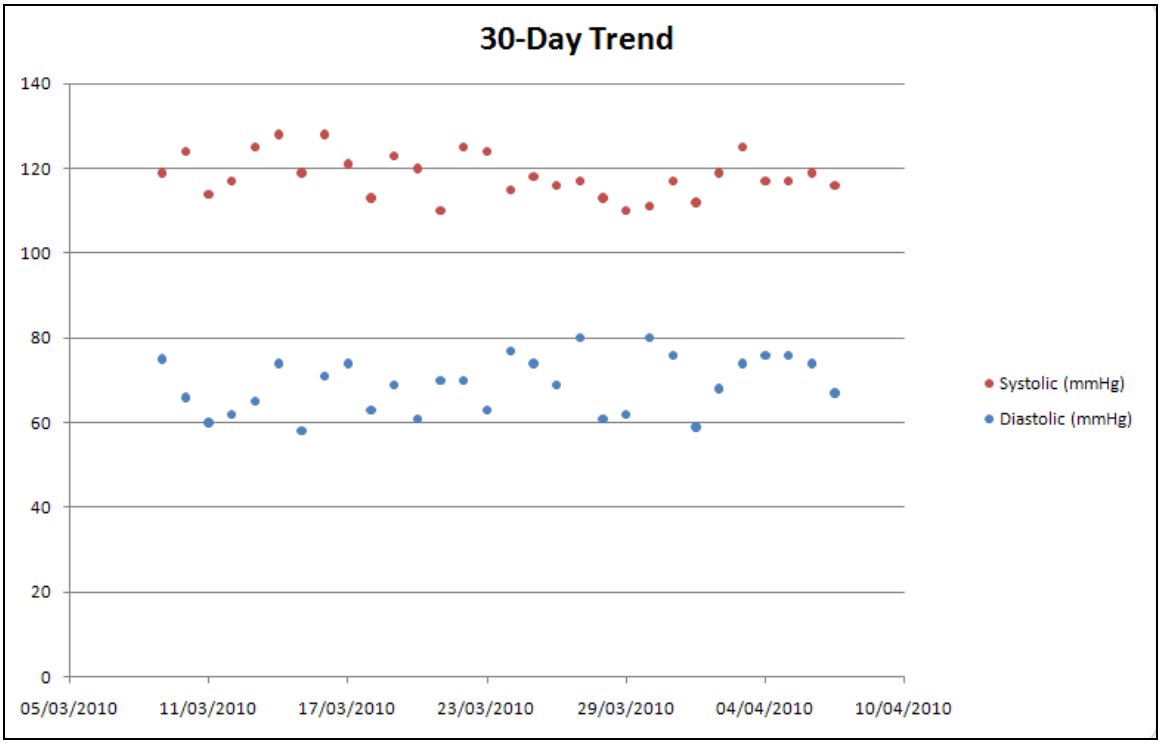
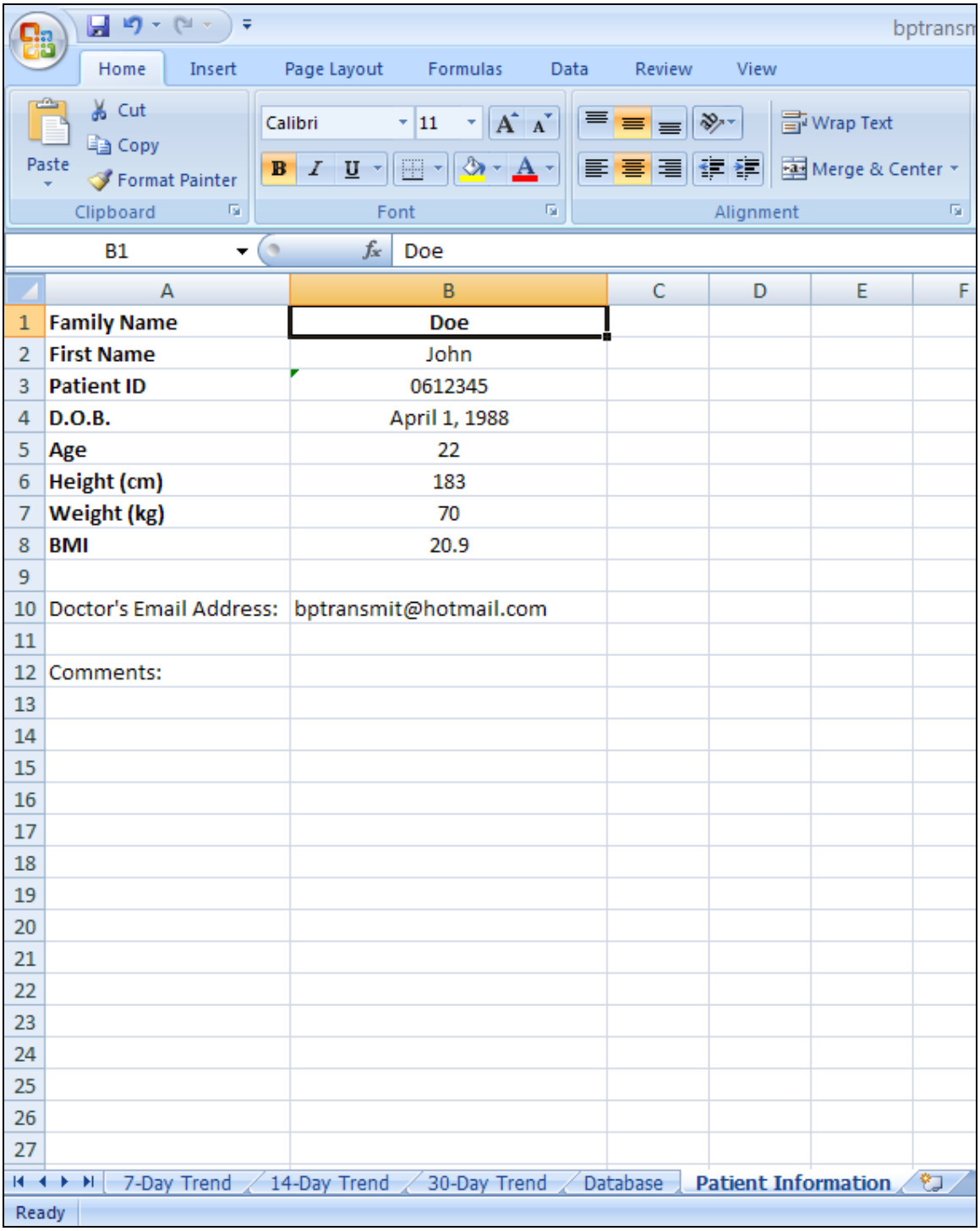


Figure 5.2: Graph of a 30-day trend containing systolic, diastolic, and heart rate values.

The trends enable the healthcare provider to check whether the patient's measurements are within reasonable limits or are showing a steady increase / decrease over time. They can be used to visibly and quantitatively measure the effectiveness of certain medication or lifestyle changes. The trends over time also allow the healthcare provider to see whether an individual's blood pressure spikes during certain seasons of the year. Plotting the data also allows for any unusual deviations from the average measurement to be quickly detected.

One of the worksheets strictly contains patient information. As the date of birth and the patient's height and weight are provided during setup, the worksheet automatically calculates the patient's age and body mass index using in-cell formulae (*cf.* Appendix A.8, page 35). The worksheet is pictured in Figure 5.3 on page 22.

The program also successfully emails out the current blood pressure reading to the healthcare provider and attaches the Excel workbook with trends to the email message. The email contains the patient's name in the subject and lists their current systolic pressure, diastolic pressure, and heart rate. A screenshot of an email message is shown in Figure 5.4 on page 23.



	A	B	C	D	E	F
1	**Family Name**	**Doe**				
2	**First Name**	John				
3	**Patient ID**	0612345				
4	**D.O.B.**	April 1, 1988				
5	**Age**	22				
6	**Height (cm)**	183				
7	**Weight (kg)**	70				
8	**BMI**	20.9				
9						
10	Doctor's Email Address:	bptransmit@hotmail.com				
11						
12	Comments:					

Figure 5.3: Patient information worksheet with calculated age and BMI.

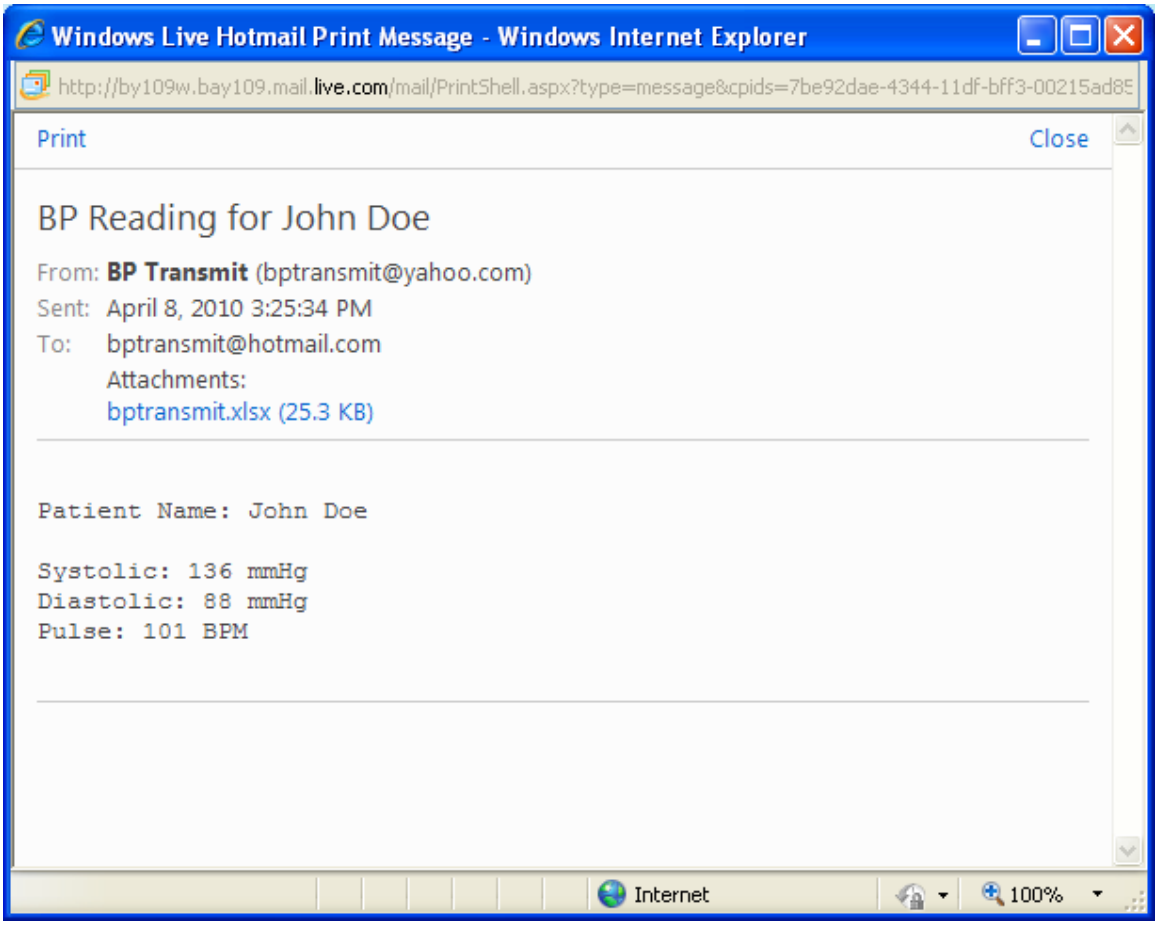


Figure 5.4: BP Transmit email message.

5.2 Discussion

As described in the methodology and design sections of this report, this project had some challenging moments, but overall the design process proceeded relatively smoothly. One aspect that was particularly frustrating was the device communication process. There is a good reason why this project uses 9-pin serial RS-232C communication rather than USB connectivity as was initially intended. The author would like to discuss the reason for choosing this method of communication by first expressing tremendous gratitude towards A&D Medical for their support in donating the UA-767PC blood pressure monitor to this project. Initially, USB connectivity was chosen to be the method of communication between the blood pressure monitor and the personal computer. As a result, A&D Medical had graciously provided a USB connectivity cable with the product. However, a

USB driver had not been provided. Without a suitable driver, the computer could not communicate with the device, and therefore could not obtain any data from the device. Developing a USB driver from scratch for a device with unknown communication methods and parameters would comprise a separate capstone project in and of itself. As a result, the company was notified of the issue, and were able to send a USB driver via email towards the end of November. This enabled the computer to recognize the device, but the computer was unable to communicate with the device driver as no commands or documentation had been provided. It seems that the device was originally made to communicate with a computer using RS-232C serial commands, and that the USB cable and device driver were trying to convert older serial technology into modern USB technology. After trying for weeks to communicate with the device driver using multiple programming languages, including C, C++, C#, and VB.NET, and achieving no progress, a choice was made to switch communication methods and a request was sent to obtain A&D Medical's 9-pin RS-232C serial cable. The cable arrived in the middle of December. Using Microsoft Visual Basic .NET and some RS-232C hex commands, data communication was established between the computer and the UA-767PC blood pressure monitor. As a result of these events, device communication was delayed and pushed back the project timeline. However, it is because of the above-mentioned circumstances that the project uses older serial connectivity as opposed to modern USB connectivity.

Once this major obstacle was dealt with, however, the rest of the project continued according to schedule. Some major programming algorithms (described in the "Experimental and Design Procedures" section of this report) took a while to develop and contributed to some of the difficulty in the project, but overall the project was successful. The final system meets all of the requirements that were set out in the objective of this capstone project. The results are clearly shown in the "Graphical Results" subsection and the system is ready to be used by patients around the world who suffer from hypertension. The application will hopefully provide patients with a hassle-free system of monitoring their blood pressure, and will boost interest in the remote health industry, so that healthcare providers in any part of the world will be able to receive necessary patient medical information quickly and conveniently.

Chapter 6

Conclusion

6.1 Concluding Statements

This project has successfully resulted in an application that automatically obtains an individual's blood pressure measurement, analyzes the data in Excel, and communicates the present information, as well as trends over time, to the patient's healthcare provider – all with the click of just one button. The application also has a setup file that was designed in this project and is therefore a fully-operational program that can be used globally as a communication tool for patients with hypertension and their healthcare providers. Hopefully, this project has contributed valuable technology to the growing remote health monitoring field.

6.2 Recommendations for Future Work

A method of having a secure, centralized database to store the patient information at a local, regional, or national level may improve the accessibility of patient data, and will eliminate the need to transfer the patient database with every blood pressure reading. Data encryption by the sending party and decryption by the receiving party can be implemented to further enhance the privacy and security of patient data.

Appendix A

Algorithms

A.1 Algorithm for Finding Available COM Ports

```
If COMBox.Text = "" Then
        For i As Integer = 0 To My.Computer.Ports.SerialPortNames.Count - 1
           COMBox.Items.Add(My.Computer.Ports.SerialPortNames(i))
        Next
        COMBox.Text = "COM4"
End If
```

A.2 Algorithm for Synchronizing Device Clock

```
SerialPort2.Open()

SerialPort2.Write("Wake")

Pause()

'Send Open Port Command
Dim open() As Byte = {&H2, &H43, &H50, &H43, &H30, &H35, &H3B}
SerialPort2.Write(open, 0, open.Length)

Pause()

'Send SetupClock Command
Dim setupclock() As Byte = {&H2, &H43, &H50, &H43, &H33, &H31, &H3A}
SerialPort2.Write(setupclock, 0, setupclock.Length)

Pause()

'Set Device Clock
SetClock()

Pause()

'Send Close Port Command
Dim close() As Byte = {&H2, &H43, &H50, &H43, &H30, &H34, &H3A}
SerialPort2.Write(close, 0, 7)

SerialPort2.Close()
```

A.3 Algorithm for SetClock Subroutine

```
Private Sub SetClock()
    Dim Now As Date = DateAndTime.Now
    Dim year As String = Hex$(Now.Year - 1900)
    Dim month As String = Hex$(Now.Month)
    Dim day As String = Hex$(Now.Day)
    Dim hour As String = Hex$(Now.Hour)
    Dim minute As String = Hex$(Now.Minute)

    Dim year1 As String = HextoSend(year(0))
    Dim year2 As String = HextoSend(year(1))

    Dim month1 As String = HextoSend(month)

    Dim day1 As String = ""
    Dim day2 As String = ""
    If day.Length = 1 Then
            day1 = "&H30"
            day2 = HextoSend(day)
    ElseIf day.Length = 2 Then
            day1 = HextoSend(day(0))
            day2 = HextoSend(day(1))
    End If

    Dim hour1 As String = ""
    Dim hour2 As String = ""
    If hour.Length = 1 Then
            hour1 = "&H30"
            hour2 = HextoSend(hour)
    ElseIf hour.Length = 2 Then
            hour1 = HextoSend(hour(0))
            hour2 = HextoSend(hour(1))
    End If

    Dim minute1 As String = ""
    Dim minute2 As String = ""
    If minute.Length = 1 Then
            minute1 = "&H30"
            minute2 = HextoSend(minute)
    ElseIf minute.Length = 2 Then
            minute1 = HextoSend(minute(0))
            minute2 = HextoSend(minute(1))
    End If
```

```
    Dim sum As String = CheckSum(year1, year2, month1, day1, day2, hour1, hour2,
                                 minute1, minute2)

    Dim clock() As Byte = {&H2, &H44, &H50, &H43, &H30, &H30, &H30, &H41,
                           &H30, year1, year2, &H30, month1, day1, day2, hour1,
                           hour2, minute1, minute2, sum}

    SerialPort2.Write(clock, 0, clock.Length)

End Sub
```

A.4 Algorithm for HextoSend Subroutine

```
Private Function HextoSend(ByVal hex As String) As String
If IsNumeric(hex) Then
        hex = "&H3" + hex
ElseIf (hex.Contains("A")) Then
        hex = "&H41"
ElseIf (hex.Contains("B")) Then
        hex = "&H42"
ElseIf (hex.Contains("C")) Then
        hex = "&H43"
ElseIf (hex.Contains("D")) Then
        hex = "&H44"
ElseIf (hex.Contains("E")) Then
        hex = "&H45"
ElseIf (hex.Contains("F")) Then
        hex = "&H46"
End If

Return hex
End Function
```

A.5 Algorithm for CheckSum Subroutine

```
Private Function CheckSum(ByVal v1 As String, ByVal v2 As String, ByVal v3 As
String, ByVal v4 As String, ByVal v5 As String, ByVal v6 As String, ByVal v7 As
String, ByVal v8 As String, ByVal v9 As String) As String

Dim total As Integer
total = 520 + Format$(v1) + Format$(v2) + Format$(v3) + Format$(v4) + Format$(v5) +
Format$(v6) + Format$(v7) + Format$(v8) + Format$(v9)

Dim output As String = Hex$(total)
Dim output2 As String = Microsoft.VisualBasic.Right(output, 2)
Return ("&H" + output2)
End Function
```

A.6 Algorithm for Parsing Blood Pressure Data

```
Dim modif As String = DataRec.Text
Dim modif2 As String = "", temp As String = "", info As String = ""
Dim sysdiahex As String = "", diahex As String = "", pulsehex As String = ""
Dim sysdia As String = "", systolic As String = "", diastolic As String = "", pulse As
    String = ""

'Parse hex command string received
If modif.Length = 16 Then
        modif2 = Microsoft.VisualBasic.Right(modif, modif.Length - 8)
        info = Microsoft.VisualBasic.Left(modif2, modif2.Length - 2)

        pulsehex = Microsoft.VisualBasic.Right(info, 2)
        pulse = HextoNum(pulsehex)

        temp = Microsoft.VisualBasic.Left(info, 4)
        sysdiahex = Microsoft.VisualBasic.Left(info, 2)
        sysdia = HextoNum(sysdiahex)

        diahex = Microsoft.VisualBasic.Right(temp, 2)
        diastolic = HextoNum(diahex)

        If IsNumeric(diastolic) And IsNumeric(sysdia) Then
           systolic = Convert.ToString(Convert.ToInt32(diastolic) +
                Convert.ToInt32(sysdia))
        Else
           MessageBox.Show("Data communication with the blood pressure monitor
                failed." + vbNewLine + "Please ensure that the device is properly
                connected to the computer and run the application again.", "Error")
           Me.Close()
           Return
        End If
Else
        MessageBox.Show("Data communication with the blood pressure monitor
             failed." + vbNewLine + "Please ensure that the device is properly connected
             to the computer and run the application again.", "Error")
        Me.Close()
        Return
End If
```

A.7 Algorithm for Sending Data for Analysis

```
WorkBook = Excel.Workbooks.Open("C:\BP-Transmit\bptransmit.xlsx")

'7-Day Trend
Sheet = WorkBook.Worksheets(1)
Dim todaytemp As String = "", today2 As String = "", systemp As String = ""
Dim diatemp As String = "", pulsetemp As String = ""
Dim systolic2 As String = systolic, diastolic2 As String = diastolic, pulse2 As String =
    pulse

i = 2
today2 = today
Dim count As Date = Date.Parse(today).ToString
Dim initial As Date = Sheet.Cells(i, 1).Value

While initial <> #12:00:00 AM# And DateDiff("d", initial, count) < 7
        todaytemp = Sheet.Cells(i, 1).Value
        Sheet.Cells(i, 1) = today2
        today2 = todaytemp
        systemp = Sheet.Cells(i, 2).Value
        Sheet.Cells(i, 2) = systolic2
        systolic2 = systemp
        diatemp = Sheet.Cells(i, 3).Value
        Sheet.Cells(i, 3) = diastolic2
        diastolic2 = diatemp
        pulsetemp = Sheet.Cells(i, 4).Value
        Sheet.Cells(i, 4) = pulse2
        pulse2 = pulsetemp
        i = i + 1
        initial = Sheet.Cells(i, 1).Value
End While

Sheet.Cells(i, 1) = today2
Sheet.Cells(i, 2) = systolic2
Sheet.Cells(i, 3) = diastolic2
Sheet.Cells(i, 4) = pulse2

i = i + 1
While i <= 40 And Sheet.Cells(i, 2).Value <> 0
        Sheet.Cells(i, 1).ClearContents()
        Sheet.Cells(i, 2) = ""
        Sheet.Cells(i, 3) = ""
        Sheet.Cells(i, 4) = ""
        i = i + 1
End While
```

A.8 Algorithm for Age Calculation

The following algorithm is implemented in Microsoft Excel to calculate the age of a patient given their date of birth:

```
=IF(MONTH(TODAY())>MONTH(B4),YEAR(TODAY())-
YEAR(B4),IF(AND(MONTH(TODAY())=MONTH(B4),DAY(TODAY())>=DAY(B4))
,YEAR(TODAY())-YEAR(B4),(YEAR(TODAY())-YEAR(B4))-1))
```

where cell B4 refers to the patient's date of birth.

A.9 Algorithm for Sending Email

```
Dim objMessage As Object = CreateObject("CDO.Message")

With objMessage
    .Subject = "BP Reading for " + name
    .From = """BP Transmit""bptransmit@yahoo.com"
    .To = Doctor
    .AddAttachment("C:\BP-Transmit\bptransmit.xlsx")
    .TextBody = "Patient Name: " + name + vbNewLine + vbNewLine + "Systolic: " +
                systolic + " mmHg" + vbNewLine + "Diastolic: " + diastolic + "
                mmHg" + vbNewLine + "Pulse: " + pulse + " BPM"
End With

Dim Flds As Object = objMessage.Configuration.Fields

With Flds
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpusessl") = True
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate") = 1
    .Item("http://schemas.microsoft.com/cdo/configuration/sendusername") =
        "bptransmit@yahoo.com"
    .Item("http://schemas.microsoft.com/cdo/configuration/sendpassword") = "*******"
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpserver") =
        "smtp.mail.yahoo.com"
    .Item("http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpserverport") = 465
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpconnectiontimeout") =
        60
    .Update()
End With

objMessage.Send()
```

A.10 Algorithm for Checking Internet Connectivity

```
Private Function CheckInternet() As Boolean
    Dim url As New System.Uri("http://www.yahoo.com/")
    Dim web As System.Net.WebRequest
    web = System.Net.WebRequest.Create(url)
    Dim response As System.Net.WebResponse
    Try
      response = web.GetResponse()
      response.Close()
      web = Nothing
      Return True
    Catch ex As Exception
      web = Nothing
      Return False
    End Try
End Function
```

Appendix B

Components

B.1 Components List

Hardware:

One (1) A&D Medical - UA767-PC Digital Blood Pressure Monitor

One (1) A&D Medical - Cuff For Digital Blood Pressure Monitors / 9.4" - 14.2"

One (1) RS-232C 9-pin serial interface cable

Optional: One (1) USB to Serial Converter

Software:

Microsoft Office Excel 2007

Connectivity:

Active Internet connection needed

B.2 Transmission Protocol

Method:	Asynchronous transmission, bi-directional
Baud rate:	9600 bps
Data bit:	8 bit
Parity:	none
Start bit:	1 bit
Stop bit:	2 bits
Code:	ASCII

References

[1] L.A. Geddes, *Handbook of Blood Pressure Measurement*, Clifton, NJ: Humana Press, 1991.

[2] D.L. Clement, Ed., *Blood Pressure Variability*, Lancaster, England: MTP Press Limited, 1979.

[3] L.A. Geddes, The Direct and Indirect Measurement of Blood Pressure, Chicago: Year Book Medical Publishers, Inc. 1970.

[4] Hugo Kesteloot and Jozef V. Joossens, Eds., *Epidemiology of Arterial Blood Pressure*, The Hague, The Netherlands: Martinus Nijhoff Publishers, 1980.

[5] Patricia M. Kearney, Megan Whelton, Kristi Reynolds, Paul Muntner, Paul K. Whelton, and Jiang He, "Global burden of hypertension: analysis of worldwide data," *The Lancet*, 2005, Vol. 365, Issue 9455, pp. 217-223.

[6] W. Meyer-Sabellek, M. Anlauf, R. Gotzen, and L. Steinfeld, *Blood Pressure Measurements*, New York: Springer-Verlag, 1990.

[7] John M.R. Bruner, *Handbook of Blood Pressure Monitoring*, Littleton, MA: PSG Publishing Company, Inc. 1978.

[8] "Health, United States, 2008," *National Center for Health Statistics*, Hyattsville, MD: 2009. Available at: http://www.cdc.gov/nchs/data/hus/hus08.pdf

Vita

Mr. Rohinton Richard is currently a senior undergraduate student of Electrical and Biomedical Engineering at McMaster University in Hamilton, Ontario, Canada. He was awarded the McMaster President's Award and the University (Senate) Scholarships for outstanding academic excellence. He is also a member of the Golden Key International Honour Society and was named to the Dean's Honour List. His leadership roles include being Vice-President (Finance) of the Bioengineering at McMaster Society and the Treasurer of McMaster Campus for Christ.