THE UNIVERSITY OF QUEENSLAND

ECG Application Featuring Data Transmission by Bluetooth

by
Daniel Marr

Submitted to
The Department of Electrical and
Computer Systems Engineering,
The University of Queensland

For the degree of
Bachelor of Engineering
In the division of Computer Systems Engineering
October 2001

8/5 Bristol St.,
West End
Qld, 4101
Tel. (07) 38461903

Thursday, 18 October 2001

Head of School
School of IT and Electrical Engineering
University of Queensland
St Lucia, Q 4072

Dear Professor Kaplan,

In accordance with the requirements of the degree of Bachelor of Engineering in the division of Computer Systems Engineering, I present the following thesis entitled "ECG Application Featuring Data Transmission By Bluetooth". This work was performed under the supervision of Dr. Adam Postula.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text, footnotes and citations, and has not been previously submitted for a degree at the University of Queensland nor any other institution.

Yours sincerely,

Daniel Marr

Acknowledgements

There are many people who contributed to the completion of this document and associated research.

I wish to express thanks to my family, who have supported me through university. Without their support, this education would not have been possible.

I have also received support from my partner Joanne, who has handled herself well amidst difficulties and good times alike. Joanne also made a good patient for testing the ECG circuit on.

Technically, also, many thanks must be given. My code was based on Naveenan Vasudeevan's Bluetooth code, and Naveenan was often willing to explain different parts of the protocol. Thanks also to David Lin, the PC software member of the team.

I would like to extend thanks to the Electronics workshop staff who were very helpful to students wishing to design circuits. Having dealt mainly with software throughout my degree, Keith Bell was very helpful in explaining some of the finer points of building circuits.

Finally supervisor Adam Postula who provided the drive and imagination for this work.

Abstract

Currently in Australia heart disease poses a major problem. It has a large financial and emotional impact on our society. If the information contained in an electrocardiogram (ECG) could become smarter, and more prolific, the weight of this problem could be reduced. It is the aim of this thesis to provide an ECG sensor capable of data transmission by Bluetooth to a PC.

The ECG wave is filtered of noise and amplified to a value able to be read by an Analog to Digital Converter. The data is fetched by a microprocessor and sent to a Bluetooth module via a UART. Conceptually, the sensor is a slave, and will be requested to send data by the PC.

A more mobile ECG sensor was achieved, with the Bluetooth communication bettering previous phone dial-up sensors. This product was only developed to the prototype level, as adhering to stringent medical requirements would stretch the design time given.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Heart disease accounts for $3.7 billion (or %12 per annum) of the total direct costs to the Australian health system. More than 40 million prescriptions for cardiovascular drugs were dispensed in the community in 1997, and in 1998 prescriptions for cardiovascular cost the government 1.18 billion dollars. With an ageing population this situation is not likely to get any better. Alternative solutions must be found.

People are dying, and governments struggle to deal with a population demanding more from the health-care system. Unfortunately the current process of monitoring and feedback is not perfect, and a more streamlined process would see improvement in the health of Australians.

1.1 Statement of Thesis Problem

It is the aim of this thesis to develop a prototype of an electrocardiogram (ECG) sensor, capable of transmitting data via Bluetooth to a PC. It is argued on two fronts that this development will make ECG data more: prolific, easy to obtain, and effective. The two areas of improvement will be in the hospitals, where worker burden will be reduced, and for patients, who will, through access to a more user-friendly device will have more analytical data to help their cause.

1.1.1 Hospital Work-load Reduction

Today the hospital system of data collection is very antiquated. Hospitals have not embraced technology due to tight budgets but this lack of investment causes massive time to be spent on trivial tasks each day. Visiting and conducting a study on the Princess Alexander Hospital’s procedures, this author learned of inefficient *pen and paper* methodologies consuming hours of employees’ work.

The sensor proposed in this document would be autonomous, requiring little configuration as per Bluetooth convention. The data is stored in a file format, reducing space, increasing order, and limiting human error. Further work would see the sensor actively *seek out* the most suitable physician in the case of an emergency using the Bluetooth Service Discovery Protocol (SDP), allowing more specialised care. Software can aid in the administering of Drug prescriptions, which is where a large amount of the expense associated with cardiovascular disease stems from. Most importantly, fewer wires would mean less hazards and less administration to be performed.

1.1.2 Patient Improvement

It is reasoned that the patient will benefit from the device, which will make data more accessible by cost and ease of use. Using a familiar computing environment, the patient can have on-hand ECG data. The storing of data on file, in conjunction with communication media like the internet, provides many options. For instance, in rural Australia where a trip to the doctor may involve air travel and several days, patients could send their data for analysis via the internet. Changing one's physician would not be so difficult, as the computer file could be sent from the old doctor to the new one. Further work here would see simultaneous voice and data transmission in the case of an emergency, using Bluetooth's SCO voice link.

1.2 Organisation

Chapter 2 gives the background on ECG necessary to read the rest of the document. History and properties of the ECG wave are discussed.

Chapter 3 discusses related literature pertaining to the design and implementation of an ECG sensor, as well as the Bluetooth protocol.

Chapter 4 is entitled Hardware Implementation and Design, and describes decisions made and implementation in this field.

Chapter 5 discusses the software decisions and the implementation of software; software consisting of Bluetooth software and sensor polling software.

Chapter 6 describes the results obtained, and the effectiveness and limitations of the sensor.

Chapter 7 consists of a discussion for future work, and direction.

Chapter 8 scribes the conclusions, and completes the document.

Included in the Appendices is a hardware schematic of the sensor and software code listing. External to this document are theses related to this device.

A thesis written by Naveenan Vasudeevan explains in more detail the architecture and implementation of the Bluetooth Protocol, and the thesis by David Lin explains the implementation of the PC software and database.

1.3 Introduction to Materials

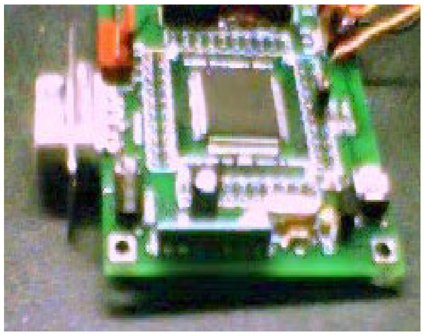


Figure 1. 68HC12 development board

Figure 1 is a picture of the development board used for this prototype. This microprocessor was chosen because of its relatively large program memory size, its co-operation with the GNU CC development environment, and its incorporation of a reliable Analog to Digital Converter (ADC). The chip is programmed using Background Debug Mode, which offers interrogation of registers while code is executing as well as programming capabilities.

An overview of the system proposed is included in Figure 2. Data acquisition is performed by the microprocessor, and data is transferred using a Bluetooth connection.

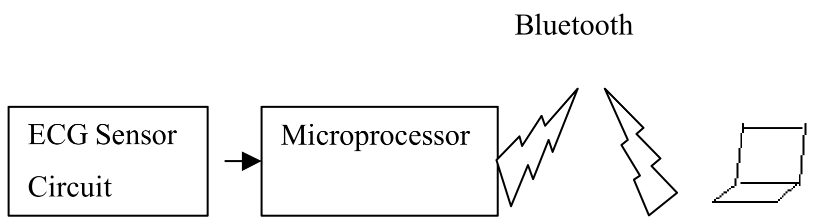


Figure 2. Overview of ECG monitoring system

The property measured by the microprocessor is an Electrocardiogram wave. This wave is a representation of the potential difference between two points in the body.

Chapter 2
ECG Background

William Einthoven developed the first electrocardiogram in 1903 using a crude galvanometer. Technology has advanced ECG measurement, but the principle remains the same. The electrocardiogram is the wave representation of the potential difference caused by heart activity. A grasp of the electrocardiogram has to be gained for two reasons: 1. An understanding of the wave forms the basis for the design of the electronic circuit to measure it; and 2. An understanding allows the concept of what an ECG is, and how its deviation enables analysis of health.

The potential difference is created by the flux of ions in cells, a typical cell with Sodium and Potassium ions is shown in
Figure **3**.

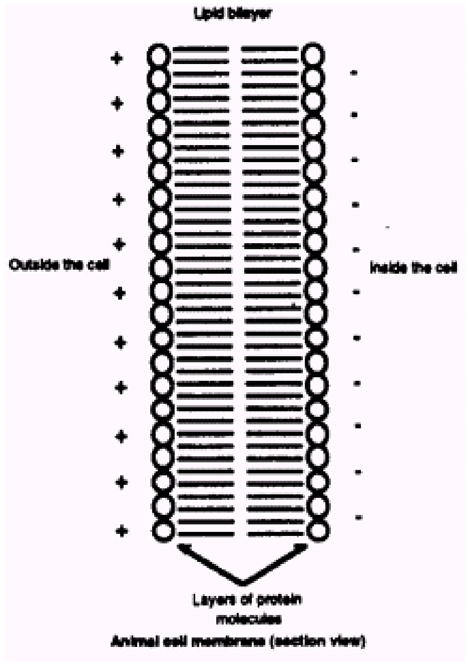

Lipid bilayer
Outside the cell
Inside the cell
Layers of protein molecules
Animal cell membrane (section view)

Figure 3. Positively charged sodium ions outside Potassium ions

When the heart pumps, the cell wall offers greater permeability and an excess of sodium is able to flow inside the cell. When the sodium flows into the cell there is no longer a negative potential with respect to the outside. This is known as depolarisation. Eventually after the excitation passes, the cell repolarises, returning the potential to a negative one.

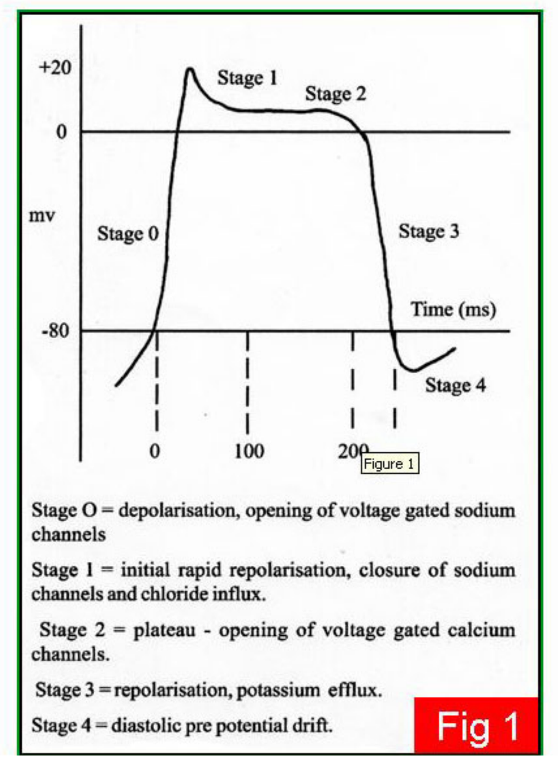


Figure 4. Electrical impulse of the heart

Figure 4, extracted from [9], shows the potential signals and the stages. This does not look like the signal from an ECG sensor. The reason is that the electrical signal will spread to different parts of the body in different ways. This is why the number of leads on an ECG sensor is important. Different forms of disease can be diagnosed from different leads. The most common lead is lead II, which is the lead implemented in this sensor. A typical output of Lead II can be seen below in Figure 5. Lead II is defined as the lead between the right and left arms.

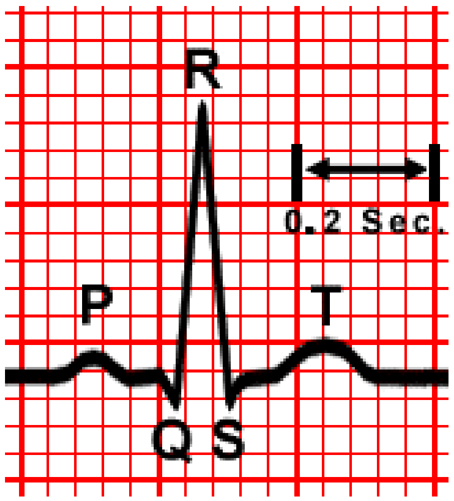


Figure 5. Typical Lead II wave

The standard for diagnostic ECG is twelve leads, however in the case of more portable, easy to use ECG one lead (usually Lead II) can be used. Lead II can diagnose the more common diseases like arrhythmias.

Cells are originally polarised such that the potential inside each cell is negative with respect to the outside. Depolarisation occurs first, making the outside of the cell negative with respect to the inside. This imbalance causes an ionic current to flow, and the Left Arm to register a positive with respect to the right arm. This is known as the *P-wave.* The depolarisation then passes to the atrioventricular node. It is still relatively negative to the left ventricle causing a current and positive voltage from the left to right arms. This is the *R-wave.* The T wave is a representation of the repolarisation of the depolarised cells. Physicians can diagnose a person's health by reading these waves.

From a signal processing perspective, the potential difference between left and right arms is typically 1-3mV and the frequency of the ECG signal lies between .02 and 150Hz. The quality of an ECG sensor could be categorised by its bandwidth. Usually sensors designed for the end of the market that this device was designed for only measure frequencies of up to 30Hz. If specifications for ECG emergency care were to be met, the frequency measured should extend to 1kHz. The problem with larger spectrums is that a proportional amount of noise must be battled with the frequencies that are measured.

Chapter 3
Literature Review

ECG sensor design is traditionally completed in the field of biomedical engineering. Investigation of Biomedical Engineering represents a good starting point for a person embarking on research of ECG.

Joseph J. Carr, *Introduction to Biomedical Equipment Technology,* 1998 Chapter 8
This book written for biomedical engineering students explains many facets of the ECG signal and its recording. An explanation of the wave is followed by a description of the noise that hampers measurement, and ways of overcoming the problems associate with noise. The operation of real ECG recorders also gives a guide to the performance requirements of the PC software with regard to tracking.

John G. Webster, *Design of Microcomputer-Based Medical Instrumentation,* 1981
This author is a self-confessed guru on the subject of ECG. A lecturer at various universities over his career, Webster has also published many times in IEEE Transactions on Biomedical Engineering. Webster was responsible for many of the enhancements that have been made on ECG sensor design. In this book he relates ECG sensor design into the realm of the computer world. Most other books describing ECG only do so with the assumption that the signal will be immediately displayed on a CRO or paper. Various ECG recorders are described in this text.

Bluetooth SIG, *Specification of the Bluetooth System,* February 2001
This specification provides a dissection of the various layers of the Bluetooth protocol. The specification is one thousand and eighty-four pages in length, but reading them does not give a total picture of Bluetooth. The specification was used as a reference for the many values of the Bluetooth op-codes, and the specific commands and what they achieve. The baseband layers explained in this specification provide a small, but incomplete understanding of Bluetooth.

J. Bray, C.F. Sturman, *Bluetooth Connect Without Cables,* New Jersey, 2000

While the aforementioned specification failed to provide much of an insight into what Bluetooth actually is, this book did not. At the beginning of the book lies a good introduction to the needs that led to Bluetooth, and Bluetooth's market positioning with respect to other wireless protocols. The further one progresses into the book, the more technically in-depth the book gets. Middle chapters explain the different layers of the protocol, the reason for their existence, and the implementation of them. Toward the end of the book actual implementation recommendations are made. That is: what horsepower a microprocessor will require for Bluetooth, segmentation of layers (Baseband and LM on one chip and RFCOMM HCI on another), and the recommendation for a Real Time Operating System.

Ericsson, *ROK101007 Preliminary Release*

This document gave an outline of the actual Bluetooth module used for this thesis. A full schematic provided information on necessary power supply pins, as well as flow control signals on the module. The Appendix A Getting Started section formed the foundations for the written Bluetooth code. The sequence of programming the module was described in detail and provided a successful platform for our coding. Appendix C provided a description of the frequency spectrum as well as an explanation of the confusing little-endian communication protocol.

Chapter 4
Hardware Design and Implementation

4.1 Sensor Decisions

A decision was made on whether to use a commercially available sensor in this design. An ECG sensor was provided by biomedical engineering company Micromedical to be used in the design. An evaluation of incorporation of the sensor into the design resulted in the decision not to use the sensor.

The Micromedical sensor transmitted data via a UART at 115,200 kilobits per second (kbps). This data rate is extremely high, and processing-wise it was too much data requiring incoming and outgoing handling. The default speed of communication with the Bluetooth module is set at 57,600 baud.

It was decided that interfacing a sensor through a UART to give data to be sent through a UART was not heading in the direction set by Bluetooth of reduction of wires. Design of the ECG sensor allowed more control over the device.

4.2 Sensor Overview

An ECG sensor was designed to measure the waveform described in Chapter Two. Two major impedances were encountered in the design of the ECG sensor. One: the signal being measured was quite small, and many other signals such as noise were larger and stronger; and two: the signal varied into positive and negative quantities at different times. This being a problem because the microcontroller's A/D converter could only read voltages from zero to five volts, and the power supply of the microcontroller only allowed positive voltages to be delivered.

The main stages of the final design are depicted in Figure 6. The signal is picked up from the patient, before undergoing signal processing to remove noise, then being amplified and put in the correct range of the ATD converter.

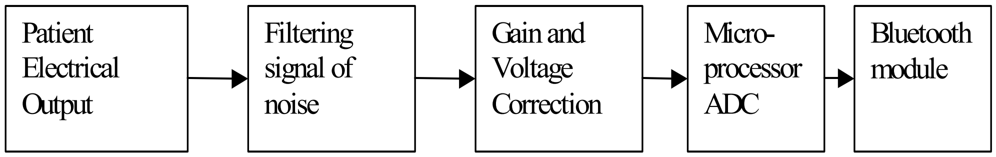

Patient Electrical Output
Filtering signal of noise
Gain and Voltage Correction
Micro-processor ADC
Bluetooth module

Figure 6. Overview of Sensor

4.3 Signal Source

The ECG is a representation of the potential difference between the right, and left arms in the case of Lead II. The best way to measure this potential would be to amplify the potential difference between the left arm and right arm. The use of an op-amp appeals in this situation. However further consideration reveals it is not so suitable. The previously alluded to noise would saturate the op-amp upon amplification, and the signal would be lost. The medical industry uses an "instrumentation amplifier" in situations like these. Instrumentation amplifiers have the property of passing common mode signals (like 50Hz noise from electricity) as a small percentage of the differential of the signal. The common-mode rejection ratio (CMRR) is the ratio of amplification of the signal divided by amplification of the common mode input. A high common-mode value is recommended in this application.

4.3.1 Instrumentation Amplifier

The instrumentation amplifier chosen for implementation was the INA114 from Burr-Brown. This particular amplifier has a CMRR of 115dB, which is considered high and meets the American Advancement of Medical Instrumentation's (AAMI) specification on electrocardiography. A CMRR of 100dB means the common mode signal will represent %1 of the output signal.

The signal should be amplified as close to the source as possible to alleviate loss of representation. However the gain of the signal should not be too high to prevent DC offsets and noise saturating the amplifier. Gain in this amplifier is added and is of the magnitude $1+\frac{50K}{R_G}$, where R_G is the value of the resistance between pins one and eight of the amplifier. The value of R_G in this case was chosen to be 51Ω, making the signal's gain close to one thousand.

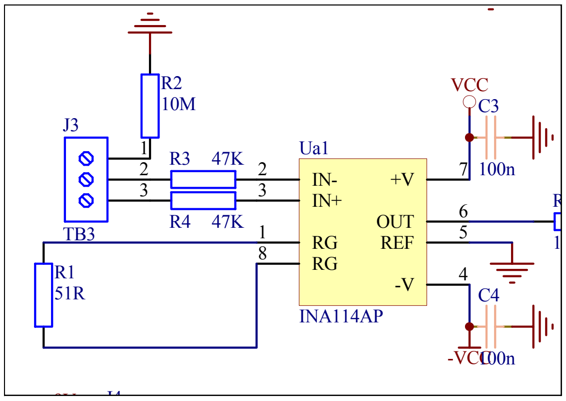


Figure 7. Instrumentation Amplifier and signal acquisition

The instrumentation amplifier stage of the circuit is shown in Figure 7. The output of the instrumentation amplifier includes: the ECG wave, and (attenuated) 50Hz inducted electricity from the patient, magnetic induction from the separation of the two wires, baseline drift at a frequency of less than .02Hz, and a multitude of light frequencies.

4.4 Filtering

To remove the unwanted frequencies in this case requires three filters. A low-pass filter is implemented to remove baseline wander of a patient. A notch or band-reject filter is used to reduce 50Hz noise. Finally, a high-pass filter is used to remove frequencies higher than 100Hz. It should be noted that a band-pass filter was not used because the pass-band was large, and it is recommended cascaded high-pass low-pass filters be used in this case.

Three options were considered for the implementation of the filtering stage of the device. A detailed explanation of the three types can be found in [7]: application note 779, from National Semiconductor. A brief description of the types is given here, with the relevance to the circuit.

4.5 Filter Principles

Most filters are built on the fundamental property of a capacitor: that it will behave as a short circuit at high frequencies and an open circuit at low ones. The “order” of a filter describes the severity of the attenuation that a signal of frequency out of the pass-band will receive. The order of a filter can be evaluated usually by counting the number of contributing capacitors. The slope representing the rate of attenuation per decade of frequency is often referred to as roll-off, or quality (Q) factor.

The selection of a filter usually involves trade-offs between several factors. A High roll-off value rejecting unwanted frequencies would be ideal, but this property will cause the pass-band’s signal to become distorted. Differences can be recognised in Figure 8, which shows a comparison of behaviour of Butterworth and Chebychev filters.

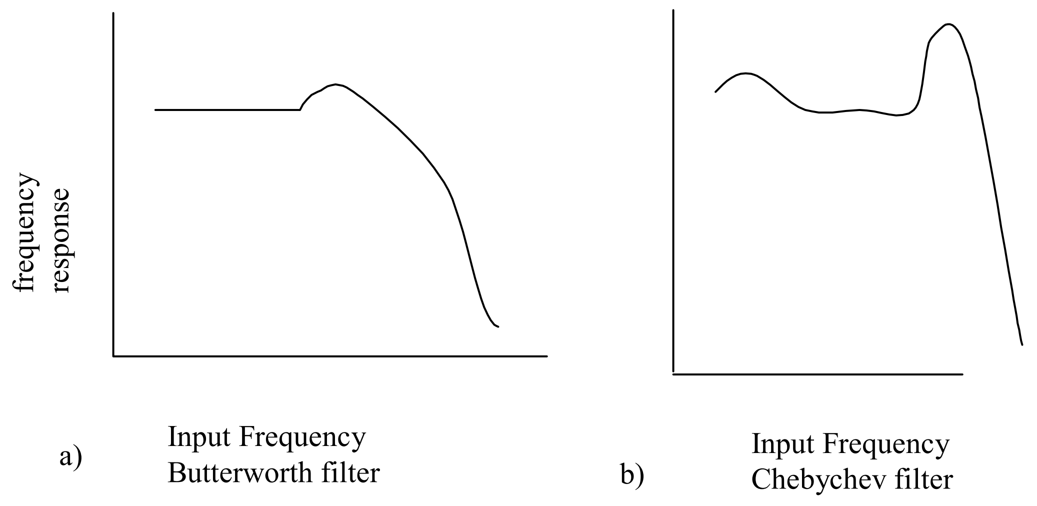


Figure 8. Flat pass-band vs high roll-off

4.5.1 Realisation of Filters

Switched Capacitor Filters –

Switched capacitor filters operate by switching a capacitor on and off to mimic a high order filter. The advantage of a switched-capacitor filter is very accurate frequency cut-off specification. An external clock sets the cut-off frequency, which means the frequency is not dependent on resistor-capacitor components, which are rarely within %5 range of the nominal value. This leads to the disadvantage where the clock must be generated either by a pre-made clock or by an oscillator (dependent on resistor-capacitor values).

An evaluation of the use of switched-capacitor filter in the circuits showed they were not effective in this design. The filters themselves generate aliases that must be filtered, and are less convenient than other types; with clocks having to be generated at difficult frequencies. A single chip can provide an 8th-order low-pass filter. The low-pass filter does not require such a steep roll-off, as little noise exists in the region immediately following the signal's frequency. High-pass and notch filters implemented by switched capacitors can only deliver a second-order filter for every chip. This is low and comparable to other types of filters.

Active Filters –

Active filters are distinguished by the inclusion in their design of an op-amp or active device. Active filters have a lower noise output than the switched-capacitor type filter. Also, due to the inclusion of an amplifier, gain can be added to the signal. For this reason the high-pass filter was implemented in this way.

Passive Filters -

Passive filters consist purely of resistors, capacitors, or inductors. An *n*th order filter is formed by cascading *n* R-C-L filters. The advantage of this filter is cost, offsetting the extra dissipation of power, and high susceptibility to component value variation.

4.6 Implementation

The rate of attenuation requires careful consideration of how accurate the pass-band must be and how steep the roll-off must be. In this example, it is most important that the pass-band be accurate, as this is representing a small ECG signal. It is considered that noise will be a common-mode input, and will already be attenuated by the instrumentation amplifier. For the case of the notch filter, a high roll-off is required to allow maximum throughput of the ECG signal. It should also be noted that the cut-off frequency calculated in equations is based on the signal being –3dB in amplitude to the original signal. Therefore the frequency calculated should take this into account.

4.6.1.1 High-Pass Filter

According to the AAMI (Association for the Advancement of Medical Instrumentation), there may exist an electrode offset of 300mV to 500mV at the input to the instrumentation amplifier. Being a DC voltage, this can be removed by a high-pass filter. Patient movement can also cause unwanted signal to be added. So the high-pass filter should be designed to cut frequencies off below 2Hz. In the circuit below, the cut-off frequency can be calculated by $f_c \approx \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$. For simplicity it is assumed

$C_1=C_2=1\mu F$, but the ratio of resistors R_1 to R_2 determines the Q of the circuit. In this circuit $Q=\frac{1}{2}\sqrt{\frac{R_1}{R_2}}$, so if R_1 was twice the value of R_2, Q would have a value of .707, and the pass-band would be maximally flat. The final constraint on the resistors is that their value should fall between 10KΩ and 270KΩ due to imperfections of the op-amp. Here the equation gives the cut-off frequency value of $\text{fc}=\frac{1}{\sqrt{56K\times120K\times1uF\times1uF}}\approx2\text{Hz}$.

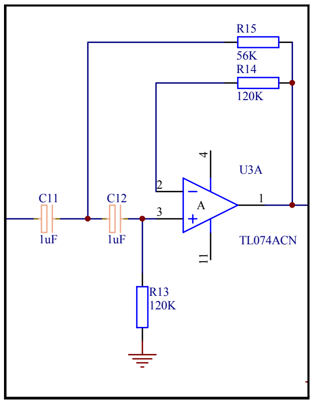


Figure 9. 40dB/decade high-pass filter

Figure 9 shows the high-pass filter implemented in this design.

4.6.1.2 Low-Pass Filter

The low-pass filter (Figure 10) was implemented as cascaded RC, or passive filters. At high frequencies, the op-amp, whose output is limited to its slew rate or maximum frequency of output, may not be able to cope with the high frequency of the signal. For this reason it was chosen to implement this filter as cascaded RC filters, before isolating the filter from the rest of the circuit by a voltage follower. The cut-off frequency is calculated by $fc=\frac{1}{2\Pi RC}$. At the cut-off frequency of the first filter, the attenuation will be 20dB/decade($10\times fc$). At the cut-off frequency of the second filter, the attenuation will be 40dB/decade thereafter. If the two cut-off frequencies are equal, then the slope will be 40dB/decade from the common cut-off frequency. The cut-off frequency calculated in this design was 158 Hz, to ensure the 150Hz signal was not attenuated. The second stage of the amplifier presents a load to the first stage, for this reason the second stage's impedance should be higher than that of the first stage.

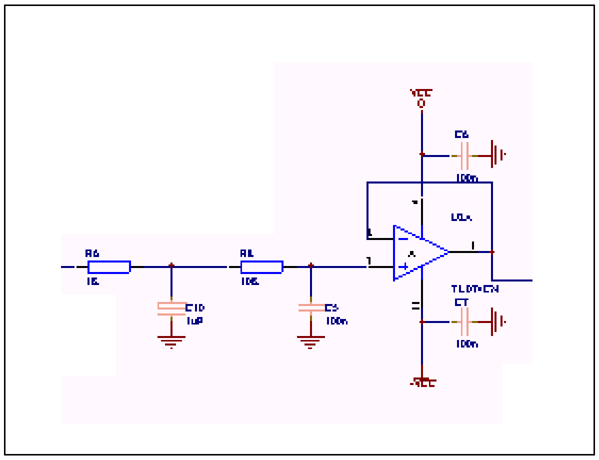


Figure 10. Low-pass filter with voltage-follower (150Hz)

4.6.1.3 Band-reject (Notch) Filter

At this stage, the signal still contains considerable 50Hz noise from mains electricity. A 50Hz notch or band-reject filter will attenuate signals at 50Hz, but not higher than, or lower than 50Hz. An application of a notch filter utilising an op-amp is shown below in Figure 11. Here resistors and capacitors applied at the inverting input form a frequency–selective feedback network. For low frequencies, the impedance of the capacitors is very high, so there is little feedback and maximum output. At *fr*, the output is maximised by the appropriate values of resistors and capacitors, decreasing output. Finally as frequencies pass *fr*, and become higher, the reactance of the capacitors decreases and the circuit behaviour approaches that of a voltage-follower.

The rejected frequency can be calculated by $fr = \frac{1}{2\Pi\sqrt{R_1 R_4 C_1 C_2}}$, and the value of Q is given by the expression $Q = .5\sqrt{\frac{R_4}{R_1}}$. The capacitors are given a value of 33nano(10^{-9})Farads, R_1 was set to 10KΩ, and R_4 was given a value of 1MΩ. That makes the Q of the circuit equal to: $.5\sqrt{\frac{1M}{10K}} = 5$. This value is small but made so deliberately to ensure a maximally flat pass-band. The depth of the notch is determined by the voltage at the non-inverting input of the op-amp, or the proportion of the resistors R_2 and R_3.

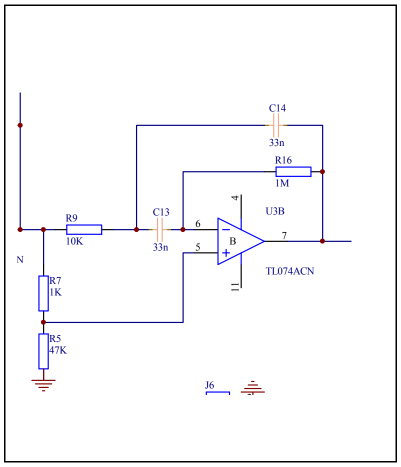


Figure 11. 50Hz notch filter

4.7 Signal Rectification

4.7.1 Gain

After the unwanted frequencies are filtered out, the gain can be added. From calibration and testing it was deemed that this amplification should occur by a factor of around seven. The signal is amplified and inverted by an op-amp, using the inverting input as its signal path. This is done because the next stage, the summing amplifier, can only be implemented on the inverting input, and will invert the signal. The formula for the gain is the familiar formula $G = \frac{-R_{feedback}}{R_{input}}$. Values chosen from the resistor series to satisfy this equation were 6K2 and 43K respectively. A graphic description of the circuit can be viewed below in Figure 12.

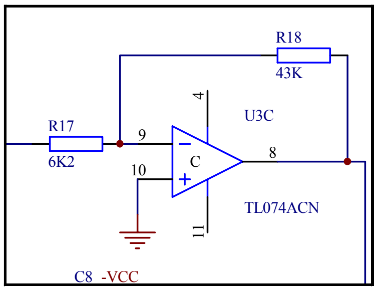


Figure 12. Signal amplification (gain of 7)

4.7.2 Voltage Range

Finally the signal must be brought into the range of zero to five volts to enable the analog to digital converter to perform analysis on it. The summing amplifier is depicted in Figure 13. Using Kirchoff's law, it is seen that the output voltage is the sum of the negative amplification of the input voltages. Here the gain is one, as all resistors are set to the same value, so the output is the addition of the signal and offset voltage inverted. In practice, the signal is around four volts peak-to-peak, so the best way to make the signal reside only in the positive domain is to add one or two 1.5VAA batteries with the positive terminals attached to ground.

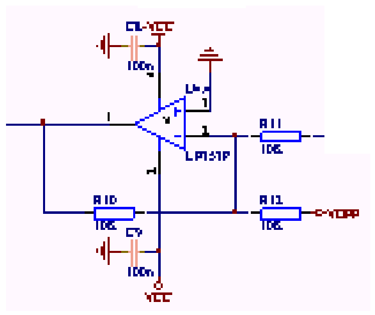


Figure 13. Summing Amplifier

4.8 Noise Reduction

Noise reduction was achieved by the incorporation of design methodologies in the final printed circuit board (PCB).

Decoupling capacitors were added close to power supply inputs. This had the effect of reducing noise by grounding voltages containing a frequency.

Large capacitors were kept away from the instrumentation amplifier and other important stages. Large capacitors have the effect of generating a considerable amount of noise.

A ground plane was implemented on the bottom layer of the board. This shielded the circuit by providing a barrier between noise and the circuit.

Twisting the lead wires reduced electromagnetic induction of noise into the circuit. Experimentally this provided a large reduction of noise.

Chapter 5
Software Design and Implementation

5.1 Software Overview

A listing of software used in this prototype is included in this document as Appendix B. The software for the microprocessor was required to perform and co-ordinate many tasks. Software regulated data acquisition of the ECG signal via the analog to digital converter. Communication with the Bluetooth module using its Serial Communication Interface was required. Further, that communication had to take place under the rules and protocols of the Bluetooth module. Data was packed in the structure of Asynchronous Connection-Less (ACL) data packets and sent via a Bluetooth link to the PC. Also, it had to handle establishment and maintenance of a Bluetooth connection. Finally the software was required to handle the interaction of the above two tasks, at times prioritising which tasks were more important.

5.2 Development Environment

The development environment used in this project was the GNU Compiler Collection. The compilation process consists of making an object file from the C file, before linking and putting data in appropriate memory space. GCC can be configured to generate code for a number of machines.

Throughout the course of research, GCC was found to be the only reasonably priced (free) compiler for machines of greater than 8kB of program memory.

As mentioned before the GCC linker *ld* is invoked to place data in appropriate memory. A linker-script is the method of communication with the linker. Commands are written in AT&T's Link Editor Command Language syntax. The linker script, *m68hc12elf.x*, configures the location of RAM and program memory, as well as loading sections of code into the right region.

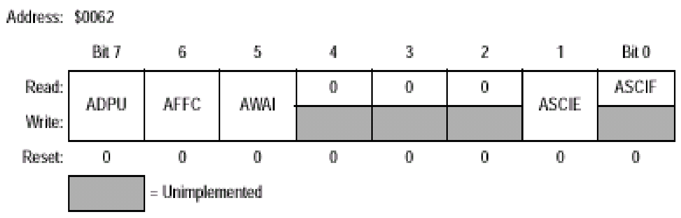
The linker script in this case was designed to allow initialisation file *install1.s* to be loaded before the HCI and L2CAP functions were loaded. The *install1.s* file initialised analog to digital conversion, and serial communication. The HCI functions perform communication with the Bluetooth module. L2CAP is a higher layer protocol that performs multiplexing.

Finally an interrupt vector table contained in *vectors.s* was loaded into memory. The linker also links in a file called *crt1.o* (C run-time), which sets the entry point of the program to point to the main function of the code, as well as other necessary house-keeping procedures.

5.3 Analog to Digital Converter

5.3.1 Initialisation

Before sampling of the ATD converter can occur, several control bits must be configured to enable, and categorise its behaviour. Bit 7 of the ATD control register 2, shown below and taken from [10], powers up the ATD converter. Setting bit 6, allow fast flag clear, means the ATD converter will take the next sample as soon as the result register has been read. Setting bit 1 enables the ATD to interrupt upon a sample being finished.

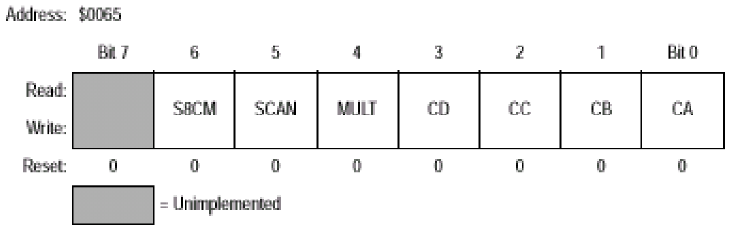


Address: $0062

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADPU	AFFC	AWAI	0	0	0	ASCIE	ASCIF
Write:				▨	▨	▨		▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14. Analog to Digital Control Register (ATDCTL2)



Address: $0065

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	▨	S8CM	SCAN	MULT	CD	CC	CB	CA
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 15. ATDCTL5: more configuration

Figure 15 shows ATD control register 5. Here the S8CM bit selects whether there are four or eight conversions per sample. The SCAN bit selects whether the ATD converter will start another sample before the user has read the register or wait, the MULT bit selects whether the conversions will be performed on one or multiple channels, and the CD-CA bits select the channel or channel group to perform conversion on.

Sampling occurs at a clocked rate formed by dividing the master clock by a certain value. Upon completion of sampling, the eight bit values of the conversion are put in memory locations ADR0H-ADR7H.

5.3.2 Behaviour

Configuring the control bits is done in assembly, and included in the file *install1.s*. Assembly language allows tight code to be written and easy modification of memory.

Firstly an appraisal of the behaviour expected of the Analog to Digital converter is given. The first demand of the sampling rate comes from the signal itself. Nyquist's theorem states that a signal must be measured at a minimum of twice its maximum frequency to be a true representation of it. The highest frequency is equal to the cut-off frequency of low-pass filter, which was 150Hz. This represents the minimum sampling, and also data transfer rate to provide an accurate signal. Another demand comes from the data transfer rate of the serial communications interface of the microprocessor. With the overhead of Bluetooth, and the serial protocol itself, the maximum data throughput is 5kHz. The rate of display on the PC screen offers another limitation, with this function being notoriously slow. This coupled with the possibility of interrupt latency, caused the sampling rate to be set to the minimum value.

The ADC can perform sampling at ten-bit, or eight-bit accuracy. The minimum prescribed accuracy of an ATD as decreed by the AAMI is twenty-two bits. However, that is because the signal needs to be found in the sea of noise. Many medical applications do not filter the signals before sampling as this introduces risk of distortion. Also data is transferred eight bits at a time and two eight-bit readings would provide a better representation of the signal than a more accurate signal at a lower rate.

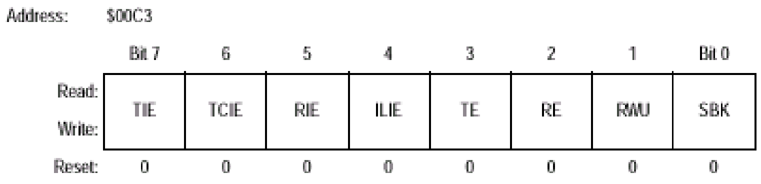
The ADC interrupt was not enabled as this may cause latency from the perspective of the SCI interrupt. Single channel conversion with eight conversions per sample was implemented.

5.4 Serial Communications Interface

5.4.1 Initialisation

SCI communication also requires initialisation. The speed at which the communication will take place must be written to the baud register. The default setting of the Bluetooth module is 57,600 baud. Putting a special value into the baud rate control register will ensure communications at the appropriate speed. That value can be calculated by the equation $BAUD = \frac{f_{ck}}{16 \times BRR}$, where f_{ck} is the frequency of the clock, in and BRR is the value written to the register.

The figure below shows the Serial Communications Control Register. Here the receiver and transmitter (bits two and three) are enabled. Bits six and five enable the transmitter or receiver interrupt respectively when set.



Address: $00C3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

Figure 16. UART setting register

5.4.2 Operation

The SCI was configured for 8 data bits, no parity, and 1 start bit. Using the equation for BRR, the value of nine was loaded into the baud rate register. The receiver interrupt was enabled but the transmitter empty interrupt was not. It was important for the receiver interrupt and the transmitter interrupt to be mutually exclusive with regard to being enabled. This was because the SCI was not full duplex, and sent and received data may clash, causing garbled messages.

5.5 Bluetooth Code

This document's scope pertains only to the embedding of Bluetooth code, and not an explanation of Bluetooth itself. The Bluetooth section of code was based on fellow thesis student Naveenan Vasudeevan's code. For an overview of the Bluetooth protocol itself, see Naveenan Vasudeevan's thesis entitled Bluetooth Development.

5.6 Software Interaction

5.6.1 Initialisation

Starting a Bluetooth connection requires commands to be sent to the Bluetooth module. A good guide to doing this is contained in Ericsson, *ROK101007 Bluetooth Documentation,* http://www.ericsson.se [4]. The module must be reset, have its buffer size and Bluetooth Address read. The module is then set to scan mode, which is looking for a connection. After each command is sent to the module receipt of a command complete event can be expected. After a connection is established, a connection complete event will occur which will provide information about the connection identifiers and the device that connected.

The sensor is originally defined as a slave in the master-slave allocation. However only a master can send data packets. Therefore a master-slave switch must be performed before the slave can send data packets.

With the connection established, the sensor is free to transmit data to the PC. The Bluetooth packet header is sent first, before the transmitter interrupt is enabled. This allows the CPU to be free to perform other tasks. When the transmitter data register is empty, the CPU will interrupt and the current value of the ADC will be loaded in for transmission. The rate at which the transmission occurs is therefore regulated by the transmitter interrupt. This method has a drawback in that transmission can not be guaranteed at a certain rate. But this method ensures maximum data throughput. If time-regulation is considered to regulate the transfer, a conservative estimate of the transfer rate must be made, and problems in transmission would break this protocol.

Software must continue to check for connection requests or other requests of the sensor. If further connection requests occur, they will be denied using the reason that limited resources are available.

The sensor will continue to transmit its data until it receives a disconnection complete event from the Bluetooth module. Hereafter the software will loop around to the beginning and perform the same operation again.

It was an aim of this thesis to embed Bluetooth code on a microprocessor. Embedding code means ensuring it fits on, and considers the embedded environment. For this reason, explanations of higher layers of the Bluetooth protocol are not given in this document. L2CAP and higher layers could be considered to be machine independent, as they handle purely software interactions with lower layers. From this thesis' perspective, higher layers fit on the 32KB of memory provided by the 68HC12, and that is all that is important. A detailed explanation is provided in Naveenan Vasudeevan's thesis.

Chapter 6
Results

6.1 ECG Hardware

The design of the ECG sensor circuit was successful. This can be seen by
Figure 17, which shows an ECG signal acquired by the sensor on a Cathode Ray Oscilloscope. Noise was reduced through implementation of a ground plane, and reduction of electromagnetics by twisting lead wire. Filtering techniques attenuated unwanted noise to highlight the electrocardiogram signal. It can be seen from
Figure 17 that the signal was rectified into realms of ADC, with only positive voltages being delivered.

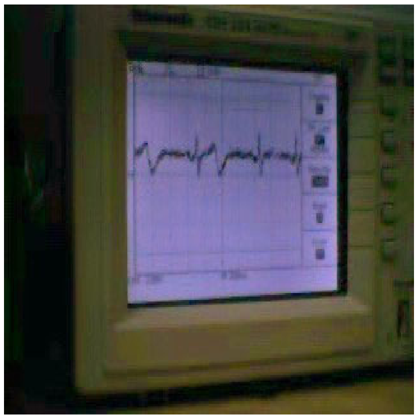


Figure 17. CRO output of the ECG sensor

Figure 18 shows the final PCB for the sensor circuit. It can be seen that the PCB is small, around 3 square inches.

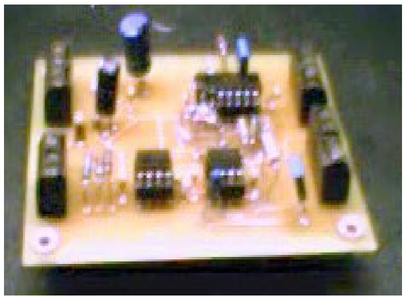


Figure 18. Assembled ECG printed circuit board

6.1.1 Budget

Table 1 provides a listing of the costs associated with the final ECG sensor circuit design. A notable omission from this list is the ROK101007 Bluetooth module. Presently costing hundreds of dollars, the price of this module has been forecast to reduce to five US dollars in the long term. It is not within the scope of this thesis to forecast market fluctuations, only to implement the design.

Table 1. ECG sensor cost

Component	**Cost ($AU)**
68HC912B32	$36.00
TL074 Op-Amp	$3.00
INA114	$7.00
LF353P	$3.00
PCB and miscellaneous	$10.00
Total:	$70.00

6.1.1.1 Power Cost

Table 2 provides a listing of the power consumption of the module. It can be deduced from this table that power consumption was minimised.

Table 2. Power Cost

Component	**Consumption mA**
68HC912B32	5mA (Wait mode)
TL074 Op-Amp	5.6mA
INA114	2.2mA
LF353P	3.6mA
ROK101007	26mA
Total:	40mA

6.2 Software

From the perspective of software, written and adapted code was successfully embedded on the 68HC12 microprocessor. The code functioned well in an embedded environment. The code to sample the ADC in conjunction with a sufficient data transfer rate provided a good representation of the signal.

Chapter 7
Future Considerations

With a successful prototype completed, foundations for exciting improvements and refinements have been laid.

A major priority for future work should be to bring the sensor design in line with specifications. The two main standards for ECG sensor design are the AAMI standard on electrocardiography, and UL544 standards. The reason these standards were not consulted nor implemented was their price-tag. The AAMI specifications cost US $95, while the UL544 standards cost US $640. This price could not be justified for a one-off project.

Some advancement would see the device able to be deployed in critical situations. A higher resolution ADC would provide more accurate results. However a different microprocessor would be needed to achieve the higher throughput required. The incorporation of USB communication capabilities would enhance the device. Currently data transfer is limited to the 57 kilobaud rate of communication with the Bluetooth module. If USB were incorporated, the data transfer rate would only be limited by the communication speed of Bluetooth, or 721 kilobaud.

This would allow more information to be sent by the sending of multiple leads' data, not just Lead II. Physicians could diagnose patients with more tools of diagnosis. The higher data rate would also allow the transfer of more precise measurements.

Many additions could occur with the maturity of Bluetooth protocol.
As mentioned in the introduction section of this report, Service Discovery Protocol could be utilised to allow the patient to find the best available physician. This hinges on the successful implementation of a piconet, which has not yet been achieved. Also

described in the Introduction was the notion of including simultaneous voice transmissions with the data transmission.

Both of these ideas require the existing Bluetooth protocol to be built upon. Currently data is limited to one PC or the maximum distance capability of Bluetooth. A limit is imposed on the data by the database storing it.

TCP/IP and UDP/IP layers would need to be built on top of the L2CAP layer of the Bluetooth protocol. Currently running TCP/IP over Bluetooth is difficult as DNS resolution, and ad hoc networking provide major problems.
If TCP and UDP could be implemented it would allow the data to travel even further, and become even more prolific.

Chapter 8
Conclusion

Throughout the course of this thesis a method has been shown to reduce the problem of heart disease in Australia.

- Background of the ECG was given and its implication to a person's health was also given.
- Theory and implementation behind an ECG sensor circuit and measurement of the ECG was given in the background section.
- Data acquisition and transmission of the sensor by the microprocessor was described.
- The embedding and interaction of Bluetooth with the sensor code was detailed.

Throughout the success of the above points, it can be realised that the two objectives of reducing heart problems have been met by this thesis:

1. Work-load reduction in hospitals
2. Proliferation of ECG data by enabling its use at home.

The prototype consists of reduced cables and reduced configuration. The autonomous sensor communicates with a PC and data is stored in a computer file format. Pen and paper methods, along with the time associated with them, can now be reduced in the hospital environment.

The ECG sensor proposed is home-enabled. Becoming appropriate for the home meant the device would have to be low-cost. This requirement was met by the sensor being cheaper than other comparable ECG sensors. The final design yielded a small sensor that would be convenient for a patient to carry. As for the hospital case, the patient benefited from a reduction of configuration given by the incorporation of Bluetooth.

This thesis concludes with the set objectives being satisfied.

References

[1] Cygnus Software, *GNUPro Compiler Tools, Using GNU CC, The C Preprocessor* 1999 http://www.cygwin.com

[2] Biomedical Instrumentation and Design

[3] F. Hughes, *Op-Amp Handbook,* Prentice Hall, N.J., 1993

[4] Ericsson, *ROK101007 Bluetooth Documentation,* http://www.ericsson.se

[5] Bluetooth SIG, *Specification of the Bluetooth System,* http://www.bluetooth.com

[6] Sterian, *GCC 68HC11 Reference Manual*, Grand Valley University, October 2000

[7] Kerry Lacanette, *A Basic Introduction to Filters – Active, Passive, and Switched-Capacitor,* National Semiconductor Application note 779, April 1991.

[8] Webster John G., *Design of Microcomputer –Based Medical Instrumentation*, Prentice-Hall, New Jersey 1981.

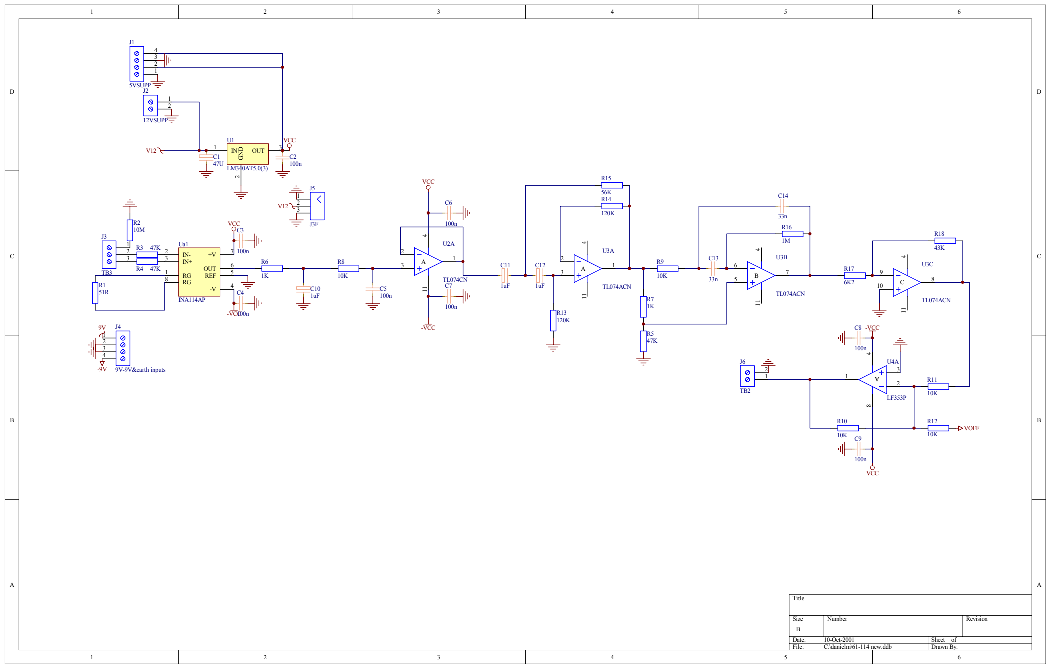
[9] N. V. Thakor, John G. Webster, "Ground-free ECG Recording with Two Electrodes," *IEEE Transactions on Biomedical Engineering,* Vol 27, Issue 12, December 1980.

[10] Dr Juliette Lee**,** *ECG Monitoring in Theatre*, Issue 11, Article 5

[11] Motorola, *MC68HC912B32 Advance Information,* Revision 3.0, http://mcu.mot-sps.com

Appendix A

Hardware Schematic



Appendix B

Firmware Code Listing

```
/*
 * TITLE:          hci.h
 *
 * TOPIC:          Bluetooth HCI layer for 68HC12
 *
 * METHOD:  Create connection before sending ECG data from Bluetooth device.
 *
 * AUTHOR:  Daniel Marr modification of Naveenan Vasudeevan.
 *
 * DATE:           19th of October
 */
#include <stdio.h>
#include <stdarg.h>
#include        "HC12.H"
#include        "nav.h"

#define         BUFFER_SIZE     100
#define         DATA_SIZE  BUFFER_SIZE-4

u8 bufferincount = 0;
u8 outcount = 0;
u8 rx_buff[BUFFER_SIZE];  /* 256 byte buffer for event */
u8 pkt_arrived = 0;                 /* Semaphore to control access to ISR  */
u8 conn_handle[2];                  /* ACL connection handle */
u8 flag;                            /* Lets receive function know the type */
u8 sample;
int variable = 0;


/*
 * Function prototypes
 */
void sci_interrupt (void) __attribute__((interrupt));
int sci_send(unsigned char);

/*
 * sci_interrupt -
 *
 * If a receiver interrupt load the data into the buffer checking for
 * final packet.  If transmitter interrupt, send the value of the
 * ATD converter.
 */
```

```
void __attribute__((interrupt))
sci_interrupt (void)
{
                    unsigned char inchar;
                    int i;

 /*
  * Receiver Interrupt. Load header values into variables,
                    * data up to the number received in the second byte.  Signal
                    * Packet received at this stage.  Event packets handled only
                    * at this stage.  ACL packets will ruin connection.
  */
 if (SC0SR1 & 0x20) {
                     SC0CR2 &= 0x7F;
                     inchar = SC0DRL;

                     /*
                      * HCI Event Packet. UART 0x04. Make sure previous request
handled.
                      */

                     if ((inchar == 0x04) && (pkt_arrived == 0)) {
                       while (!((volatile) SC0SR1 & 0x20));

                       /*
                        * Get Event Code
                        */
                       rx_buff[0] = SC0DRL;

                       /*
                        * Get parameter Length
                        */
                       while (!((volatile)SC0SR1 & 0x20));

                       /*
                        * Parameter length
                        */
                       rx_buff[1] = SC0DRL;

                       /*
                        * Put parameters in buffer
                        */
                       for (i = 0; i < rx_buff[1]; i ++) {
                        while (!((volatile) SC0SR1 & 0x20));
                               rx_buff[i+2] = SC0DRL;
                       }

                       /*
```

```
                                * Send processing required message.
                                */
                               pkt_arrived = 1;
                            }
                          }
                          /*
                           * As Receiver interrupt was not triggered,
                           * must be a transmitter interrupt.  Send value
                           * in ATD register.
                           */
                          else if (SC0SR1 & 0x80) {
                               variable += ADR0H;
                               variable += ADR1H;
                               variable += ADR2H;
                               variable += ADR3H;
                               variable += ADR4H;
                               variable += ADR5H;
                               variable += ADR6H;
                               variable += ADR7H;

                               variable /= 8;
                               sample = (char) variable;

                               SC0DRL = sample;
                               outcount++;
                          }
}

/*
 * main -
 *
 * Establish connection before entering infinite loop of seeing if processing is
 * required, then enabling the transmitter interrupt.
 */
unsigned char sci_char;
int main(void)
{
                          unsigned char pos, tosend;
  int i;
  while(1) {

                          /*
   * Must reset Bluetooth Module
                           */
                          sci_send(0x01);
                          sci_send(RESET);
                          sci_send(HCI_BB * 4);
                          sci_send(0x01);
```

```
wait();

/*
 * read_bd_addr -
 */
sci_send(0x01);
sci_send(READ_BD_ADDR);
sci_send(HCI_IP * 4);
sci_send(0x00);
wait();

/*
 * set event filter
 */
sci_send(0x01);
sci_send(SET_EVENT_FILTER);
sci_send(HCI_BB * 4);
sci_send(3);
sci_send(0x02);
sci_send(0x00);
sci_send(0x02);
wait();

/*
 * write connection accept timeout
 */
sci_send(0x01);
sci_send(WRITE_CONNECTION_ACCEPT_TIMEOUT);
sci_send(HCI_BB * 4);
sci_send(2);
sci_send(0x00);
sci_send(0x30);
wait();

/*
 * write scan enable
 */
sci_send(0x01);
sci_send(WRITE_SCAN_ENABLE);
sci_send(HCI_BB * 4);
sci_send(1);
sci_send(0x03);
wait();

/*
 * Wait for connection complete event, before retrieving data.
 */
wait();
```

```
conn_handle[0] = rx_buff[3];  //0) event 1) numparams 2) comm complete 3)
conn_handle[0]
            conn_handle[1] = rx_buff[4];
            for (i = 0; i < 6; i++) {
                bd_addr[i] = rx_buff[i+5];
            }

            /*
             * Request master-slave role switch
             */
            sci_send(0x01);
            sci_send(SWITCH_ROLE);
            sci_send(HCI_LP * 4);
            sci_send(7);
            for (i = 0; i < 6; i++) {
                sci_send(bd_addr[i]);
            }
            sci_send(0x00);

            /*
             * While no disconnection request received
             */
            while (rx_buff[0] != 0x05) {
                /*
                 * If processing required, do so.  Turn transmitter interrupt back
            on
                 * as it will have been turned off on receiver interrupt.
                 */
                if (pkt_arrived == 1) {
                        msg_handler();
                        if ((outcount > 0) && (outcount < DATA_SIZE)) {
                                SC0CR2 |= 0x80;
                        }
                        pkt_arrived = 0;
                }

                /*
                 * If maximum data sent send next packet, by header
                 *
                 */
                if ((outcount == 0) || (outcount == DATA_SIZE)) {
                        outcount = 0;
                        SC0CR2 &= 0x7F;
                        sci_send(0x02);
                        sci_send(conn_handle[0]);
                        sci_send(conn_handle[1] | 0x20);
                        sci_send(BUFFER_SIZE);
                        sci_send(0);
```

```
                                        sci_send(DATA_SIZE);
                                        sci_send(0);
                                        sci_send(0x01);
                                        sci_send(0x00);
                                        SC0CR2 |= 0x7F;
                                    }
                                }
                                }

 return 5;
}

/*
 * Function to send data on UART by polling.
 */
int sci_send(u8 data) {
                                while (((volatile) SC0SR1 & 0x80) != 0x80);
                                SC0DRL = data;
                                return 0;
}

int wait(void) {
                                while ((volatile) pkt_arrived == 0);
                                pkt_arrived = 0;
                                return 0;
}

/*
 * Reject further incoming connection requests.  Send rejection, BD_ADDR,
 * due to limited resources 0x0D.
 */
void msg_handler(void) {
                                int i;
                                if (rx_buff[0] == 0x04) {
                                    sci_send(0x01);
                                    sci_send(REJECT_CONNECTION_REQUEST);
                                    sci_send(HCI_LC*4);
                                    for (i = 0; i < 6; i++) {
                                        sci_send(rx_buff[i+2]);
                                    }
                                    sci_send(0x0D);
                                }
}
```

```
; .install1 clears COPCTL, sets up A/D
; and sets up SCI communication

        .sect   .install1
```

```
; Clear COPCTL register

      CLR    0x16
      LDS    #0xC00

; Set up for A/D conversion
; go through adctl0-5

      LDAA   #0xC0          ; Initialize ATD
      STAA   0x62                 ; Fast flag clear, ADPU

      LDAA   #0x60          ; Enable Scan Mode
      STAA   0x65


; SCI ENABLE
      LDAA    #9          ; 9600 BAUD
      STAA    0xC1

      LDAA    #0x2C    ; TIE NOT ON CURRENTLY, RIE, TE, RE
      STAA    0xC3
```

```
; vectors.s
; vector jump table
      .sect   text
      .globl  _start
      .globl  sci_interrupt
      .sect   .text

def:   rti

;----------------- Vector Section -----------

      .sect   .vectors
      .word   def                 ; ffc0
      .word   def         ; ffc2
      .word   def         ; ffc4
      .word   def         ; ffc6
      .word   def         ; ffc8
      .word   def         ; ffca Reserved
      .word   def         ; ffcc
      .word   def         ; ffce, Key Wakeup H, KWIEH[7:0]
      .word   def         ; ffd0, BDLCKey Wakeup J, KWIEJ[7:0]
      .word   def         ; ffd2, ATD, ADIE
      .word   def         ; ffd4, reserved

      ;...SCI 0.........................................
      .word   sci_interrupt   ; ffd6, SCIO
```

```
;...SPI...........................................
.word   def         ; ffd8,  SPI

.word   def         ; ffda, PACTL
.word   def         ; ffdc, PACTL OVERFLOW
.word   def         ; ffde, Timer Overflow, TOI

;...Timer channel 7 - 0...........................
.word   def         ; ffe0
.word   def         ; ffe2
.word   def         ; ffe4
.word   def         ; ffe6
.word   def         ; ffe8
            .word   def             ; ffea
            .word   def             ; ffec
            .word   def             ; ffee

;...Real Time Interrupt...........................
.word   def         ; fff0 (RTII)

;...Interrupt Request.............................
.word   def         ; fff2 (IRQ)

.word   def         ; fff4 (XIRQ)

            .word   def             ; fff6 (SWI)

            .word   def             ; fff8 (ILL)

            .word   def             ; fffa (COP Failure)

            .word   def             ; fffc (COP Clock monitor)

;...Reset.........................................
.word   _start      ; fffe
```

```
; Linker Script arranges data in appropriate place.
/* Linker script for 68HC12 executable (PROM).  */
OUTPUT_FORMAT("elf32-m68hc12", "elf32-m68hc12",
                  "elf32-m68hc12")
OUTPUT_ARCH(m68hc12)
ENTRY(_start)
MEMORY
{
  page0 (rwx) : ORIGIN = 0x800, LENGTH = 0x100
  text  (rx)  : ORIGIN = 0x08000, LENGTH = 0x8000
```

```
 data        : ORIGIN = 0x900, LENGTH = 0x300
}
.text  :
  {
    /* Put startup code at beginning so that _start keeps same address.  */
    /* Startup code.  */
    *(.install0)        /* Section should setup the stack pointer.  */
    *(.install1)        /* Place holder for applications.  */
     *(.text)
    *(.fini)
     _etext = .;
    PROVIDE (etext = .);
  }  > text
.data    : AT (__data_image)
  {
    __data_section_start = .;
    PROVIDE (__data_section_start = .);
    *(.sdata)
    *(.data)
    *(.data1)
    *(.gnu.linkonce.d*)
    CONSTRUCTORS
    _edata  =  .;
    PROVIDE (edata = .);
  }  > data
  __data_section_size = SIZEOF(.data);
  PROVIDE (__data_section_size = SIZEOF(.data));
PROVIDE (_vectors_addr = DEFINED (vectors_addr) ? vectors_addr : 0xFFC0);
  .vectors DEFINED (vectors_addr) ? vectors_addr : 0xFFC0 :
  {
    *(.vectors)
  } > text
```