

**DESIGN AND IMPLEMENTATION OF WIRELESS TELEMEDICINE
SYSTEM – ELECTROCARDIOGRAPH (ECG)**

NG ENG HUI

**A project report submitted in partial fulfillment of the
requirements for the award of the degree of
Bachelor (Hons.) of Biomedical Engineering**

**Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

APRIL 2011

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : NG ENG HUI

ID No. : 07UEB08672

Date : 13 MAY 2011

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**DESIGN AND IMPLEMENTATION OF WIRELESS TELEMEDICINE SYSTEM - ELECTROCARDIOGRAPH (ECG)**” was prepared by **NG ENG HUI** has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Biomedical Engineering (Hons.) at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Mr. Teoh Chee Hooi

Date : 13 MAY 2011

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2011, Ng Eng Hui. All right reserved.

Specially dedicated to
my beloved grandmother, mother and father,
brothers and sisters,
Mr Wei Ping, Mr Cuau, Mr Teoh,
and lastly the special and only one in my life.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr. Teoh Chee Hooi for his invaluable advice, guidance and his enormous patience throughout the development of the research. Without him I could not have completed this project.

In addition, I would also like to express my gratitude to my loving parents, family and friends who had helped and given me encouragement throughout this project. I am grateful to Mr. Cuau, developer of Freescale's TWR-MCF51MM kit who sponsored me in this project. I would like to thank Mr. Wei Ping, Freescale's senior engineer who shared his knowledge and gave valuable advice to me in this project.

Lastly, I would like to thank my only one and special person in my life for her continuous support and encouragement to me in this project.

**DESIGN AND IMPLEMENTATION OF WIRELESS TELEMEDICINE
SYSTEM – ELECTROCARDIOGRAPH (ECG)**

ABSTRACTS

Heart attack is a critical disease nowadays and it kills millions of people worldwide. Electrocardiograph is medical equipment that able to monitor the electrical activity of heart and diagnose any possible heart abnormality. The heart condition of heart disease high risk groups especially elderly need to be monitored from time to time to detect any abnormality. However, to set up the conventional medical device such as electrocardiograph is not a workable plan as it is space occupying, high cost, complex in term of operating, need schedule maintenance, rather immobile (connect with the cable), etc. Currently, there are many brands of blood pressure monitor and blood glucose tester available in the market for sale but not the electrocardiograph. There is a trend observed increasing demand for home monitoring system for healthcare purpose. As heart disease is one of the chronic disease among the elderly, there is a need to develop an electrocardiograph system which is cost efficient, easy to setup, mobile, user friendly, etc. In this project, the author concentrate on develop a wireless prototype system that able to monitor the health parameter at home. Freescale's development kit, TWR-MCF51MM used to develop the prototype system. Developed prototype system able to monitor the health status of someone in real time without present to the hospital and it has advantages such as cost efficient, cable less, easy to use, occupy lesser space, etc. Electrocardiogram will help to diagnoses any heart rhythm abnormality and a GUI will be used to display the ECG data. Wireless transmission of ECG data has been established in this project but medical server has yet to develop. Using this prototype system, subject can capture own ECG data in easy setting up, fast and cost efficient manner.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	vi
ABSTRACTS	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS / ABBREVIATIONS	xvi
LIST OF APPENDICES	xvii

CHAPTER

1	INTRODUCTION	1
	1.1 Background	1
	1.2 Aims and Objectives	2
	1.3 Problem Statement	3
	1.4 Outline of Report	4
	1.5 Challenges	5
2	LITERATURE REVIEW	6
	2.1 Telemedicine	6
	2.2 Telemedicine Concepts	7
	2.3 Current Situation of Telemedicine	8
	2.4 Heart and Heart-Electrical Activity	9
	2.5 Electrocardiograph (ECG)	12

2.6	Electrocardiograph Interpretation	13
2.7	Electrocardiograph Electrode	15
2.8	Electrocardiograph Interference Source	16
2.8.1	Noise Originated from Source	17
2.8.2	Noise Originating from Patient	18
2.8.3	Noise Originating from Electrode Contact	18
2.9	Wireless ECG system (Quick Doc ECG)	19
2.10	Wireless Technology – ZigBee	22
2.11	Current Application of Wireless ECG System	24
3	METHODOLOGY	27
3.1	Scope of Works and Work Flow	27
3.2	Work Breakdown	29
3.3	Gantt Chart	30
3.3.1	Gantt Chart Semester I	30
3.3.2	Gantt Chart Semester II	31
3.4	Architecture for Project	33
3.5	Freescale’s medical development kit (TWR-MCF51MM)	35
3.5.1	TWR-MCF51MM	35
3.5.2	TWR-SER	38
3.5.3	TWR-ELEV	38
3.5.4	MED-ECG	39
3.6	Cytron’s SKXBee (Wireless Module)	41
3.7	MAX232IN	44
3.8	Software	45
3.8.1	CodeWarrior Development Studio for Microcontrollers 6.3	45
3.8.2	Digi Maxstream’s X-CTU	46
3.8.3	MED-ECG GUI	47
3.8.4	Eltima’s RS232 Data Logger	48
4	DESIGN AND IMPLEMENTATION	49
4.1	Placement of ECG Electrodes	49

4.2	Programming of Freescale's Medical Development Kit	50
4.2.1	Programming Flow	52
4.3	Jumper Settings on Freescale's Medical Development Kit	56
4.4	MAX232 Circuitry	57
4.5	Wireless Transmissions using XBee	58
4.5.1	Flow of Data in Transmitter and Receiver	60
4.6	Prototype Development	62
5	RESULTS AND DISCUSSIONS	66
5.1	Results of the Prototype System	66
5.1.1	Transmitter Site of Prototype System	66
5.1.2	Receiver site of Prototype System	69
5.2	Analysis of Results	71
5.2.1	ECG waveform at Transmitter Site	71
5.2.2	Data Received at Receiver Site	73
5.3	The Effects of Different Placements of ECG Electrodes	74
5.4	Conversion Factor of ADC level to Voltage level	78
5.5	Noise issue	79
5.6	XBee Buffer Issue	82
6	CONCLUSION	84
6.1	Comments on the project	84
6.2	Possible Improvement and Recommendations for Future Works	85
	REFERENCES	86
	APPENDICES	90

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	The placement of the electrode on the three lead ECG	12
2.2	ECG waveform	14
2.3	Comparison between ZigBee and Bluetooth	24
3.1	Function and description of SKXBee components	42
3.2	Interfacing pin of SKXBee	43
4.1	Modification of Jumper Settings	56
4.2	Address for both SKXBee	60
5.1	Summary of Different Placements of ECG Electrode on Chest	74

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Blood circulation scheme	9
2.2	Cardiac conduction system	10
2.3	Myocardium electrical activity	10
2.4	Einthoven Triangle	11
2.5	ECG lead placement	12
2.6	Typical ECG signal	13
2.7	Electrodes construction	15
2.8	Silver/Silver Chloride electrode	16
2.9	Receiver connected to the PDA.	21
2.10	Quick Doc ECG System	22
2.11	ZigBee Protocol Stack	23
2.12	Code Blue infrastructure	25
2.13	MOLEC infrastructure	26
3.1	Scope of project	27
3.2	Work Flow of the project.	28
3.3	Work Breakdown	29
3.4	Gantt Chart Semester I	31
3.5	Gantt Chart Semester II	32
3.6	Architecture of Receiver Site	33

3.7	Architecture of Transmitter Site	34
3.8	TWR-MCF51MM	35
3.9	TWR-MCF51MM Block Diagram	36
3.10	TWR-SER	38
3.11	Feature of TWR-ELEV	39
3.12	MED-ECG	39
3.13	MED-ECG block diagram	40
3.14	Fingertips Collocation on Slider Electrodes	41
3.15	SKXBee, Zigbee module	42
3.16	MAX232IN	44
3.17	RS232 to TTL convertor in MAX232IN.	45
3.18	Code Warriors 6.3	45
3.19	X_CTU program interface	46
3.20	MED-ECG GUI	47
3.21	Eltima's RS232 Data Logger	48
4.1	External Electrodes Connection	49
4.2	Silver Chloride ECG electrode	50
4.3	Flow Chart of Main Function (main.c)	52
4.4	Flow of Check USB status	53
4.5	Flow of TestApp_Task	54
4.6	Communication pipe between GUI and Prototype	55
4.7	Circuit diagram of MAX232 circuitry	57
4.8	MAX232 circuitry on breadboard	57
4.9	Data Flow from microcontroller to XBee module	58
4.10	Example Data Format of XBee (8 bits, none parity, 1 stop bits)	59

4.11	Internal Data Flow Diagram	61
4.12	ECG prototype (Transmitter site)	62
4.13	Prototype system (Receiver site)	63
4.14	Reprogram the TWR-MCF51MM	64
4.15	Program microcontroller on board using CW 6.3	64
4.16	Data Flow Diagram	65
5.1	Typical ECG waveform obtained with 9x op-amp gain	67
5.2	Movements when measuring will create noise	67
5.3	Measuring with smallest op-amp gain 2x	68
5.4	Measuring with biggest op-amp gain 17x	68
5.5	Measuring without DSC function	68
5.6	Measuring with slide electrode (unsecure, with slight finger movement)	69
5.7	Measuring with slide electrode (secure, no finger movement)	69
5.8	Monitoring data received at receiver site using X-CTU	70
5.9	Data received convert from ACHII to hex form	70
5.10	Analysis of Real Time ECG data	71
5.11	Various QRS Complex Morphologic	72
5.12	ECG waveform results analysis	72
5.13	A standard rhythm stripe	73
5.14	Cardiac Axis	76
5.15	Shape of QRS complex depends on orientation of electrode	77
5.16	Best orientation of electrodes that output closest results to typical ECG waveform (LC- red, RC- white, LA- black)	78

5.17	60 Hertz Noise	80
5.18	Value of ADC level during noise interference	81
5.19	Example of 60 Hertz noise	81

LIST OF SYMBOLS / ABBREVIATIONS

GUI	Graphical user interface
ECG	Electrocardiograph or electrocardiogram
DSC	Digital Signal Conditioning
AgCl	Silver Chloride
AC	Alternating current
DC	Direct current
MCU	Microcontroller
V_{signal}	Voltage of ECG signal
CW	Code Warriors (Program)
LC	Left chest
RC	Right chest
LA	Left abdominal

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Pin Layout Table for Freescale's Medical Development Kit	90
B	Programming Files and Functions for Freescale's Medical Development Kit	92
C	Default Jumper Settings for Freescale's Medical Development Kit	93
D	txt file captured using RS232 data logger	95
E	Notepad++ (convert ASCII to hex) and Analysis of hex data	96
F	Pinout for XBee and XBee Pro	98
G	Programming Source Code	99
H	Schematics Diagram of Freescale's Medical Development Kit	119

CHAPTER 1

INTRODUCTION

1.1 Background

According to a World Health Organization (WHO) estimate, cardiovascular disease kills almost seventeen million people around the world each year, with around twenty million people at a risk of sudden heart failure. (WHO, 2004) At present, the number of Malaysians aged 60 years and above is estimated to be 1.4 million and is projected to increase to 3.3 million in the year 2020. Between 1990 and 2020, the population of Malaysia is expected to increase from 18.4 million to 33.3 million (an increase of 80%). The percentage of the population that is 60 years and over has also increased over the years 5.2% in 1970, 5.7% in 1990 and 6.3% in the year 2000. In the year 2020, this percentage is expected to be 9.8% of the population. (Mohamed, 2000) The population of elderly will keep increasing in the future and other countries such as Japan and United State are facing issue of population ageing too. The elderly are tend to have health problem compare to the youngsters, hence an increase in the proportion of the aged group is associated with an increase in the prevalence of ill health especially with the heart disease. The physical and social changes associated with ageing are combined with the debilitating effects of multiple, acute and chronic diseases.

The average Westerner suffers a heart attack at age 66. A Malaysian gets heart attack at age 59. Worse, there are even teenage victims of heart attack (Chai, 2008). Elderly is the group of people that more susceptible to heart attack as their heart function and blood vessel is not as good as the youngsters. Heart attack can

occur everywhere especially in the home where the elderly spend most of the time. It will be a good idea to have ECG at home to record the elderly heart rhythm and if there is abnormality in the electrocardiograph, the system will alert with the family members or the medical server. If there is heart attack happened on elderly especially for the stay alone elderly, medical attention can be provided immediately to the victims and this aids to save life.

Heart attack results from blood vessel disease in the heart. Coronary heart disease (CHD), sometimes referred to as coronary artery disease (CAD), are general names for heart attack (and angina). A heart attack, or myocardial infarction, occurs when the blood supply to part of the heart muscle itself (the myocardium) is severely reduced or stopped. This occurs when one of the coronary arteries (the arteries that supply blood to the heart muscle) is blocked by an obstruction, such as a blood clot that has formed on plaque due to atherosclerosis. If the blood supply is cut off drastically or for a long time, muscle cells suffer irreversible injury and die. Disability or death can result, depending on how much heart muscle is damaged (Yayasan Jantung Malaysia, 2009). The electrocardiograph system able to detect such complication and it can save life via notifying the patient, family members or medical server even before the complication happened by detecting abnormality in electrocardiogram (wide or missing of QRS complex in electrocardiogram).

1.2 Aims and Objectives

The main objective of this project is to develop a hardware application for home monitoring system using Freescale's Medical Development Kit for MCF51MM Family, TWR-MCF51MM kit. Secondary objective is to establish wireless transmission between the data capturing site (transmitter) to a receiving site. In this report, the author will aim on reporting the details working of wireless ECG monitoring system part in the monitoring system. The author also aims to develop a ECG prototype system at the end of the project. This product will reliably measure

the electrical activity around the heart and transmit this data via wireless module to a receiver connected to a PC.

1.3 Problem Statement

As a person ages, the heart undergoes subtle physiologic changes, even in the absence of disease. The elderly represent the fastest-growing segment of the human population., which is people faced in the bad life style and not care about their healthy. There are two problems predominate in elderly patients with heart disease such as chronic coronary artery disease (CAD) and chronic congestive heart failure (CHF). (Mohamed, 2000) The elderly heart becomes less responsive to adrenaline and cannot increase the strength or rate of its contractions during exercise to the same extent it could in youth. Thus, they need a monitoring system to prevent or at least give alert during the pre and post of the complication.

The elderly who suffering from heart disease may be have a heart attack at any time and everywhere especially at home which is the place they spend most of the time. The heart problems can be diagnosed using electrocardiograph and the doctor will analyze the electrocardiogram to determine whether there is abnormality in the rhythm of heart. However, the conventional way of healthcare need the elderly patients need to go to hospital or clinic frequently to check their health conditions and it is inconvenient to them who are disabled and sick. Moreover, most elderly do not like go to hospital as they have misconception on the hospital. Elderly are the group of patient who is more susceptible with the heart disease compared to the children and adult due to the physiologic changes. If the heart problem didn't diagnosed in early stage, treatment cannot be administrated to the patient and complication might occur without any signs. Settings up of conventional electrocardiograph at home is inconvenient too as it occupy spaces and attach with multiple cable. Moreover, the operations of conventional electrocardiograph are complex, high costs and the function of machine not easily comprehend by the public who has no knowledge on the machine. In addition, elderly spend most of the time in

the home especially those who are disabled. Current type of medical devices for vital sign monitoring use very expensive components and not design for multi patients. Consequently, patients are unaffordable to buy that particular medical monitoring device. There is a need to developed a easy set up, simple, effective and low cost wireless electrocardiograph system or a comprehensive home monitoring system to suit the market's needs.

1.4 Outline of Report

This documentation introduces application of Freescale's medical development kit to capture the ECG signal and establish wireless transmission of ECG data between a receiver site and a transmitter site. Following listed outline for each chapter in this report:

- **Chapter 2** provide some of literature reviews about the project such as working principle of ECG, details about the ZigBee, current application of wireless ECG, etc.
- **Chapter 3** record down the methods and strategies implemented in this project. It also includes the introduction of Freescale's medical development kit, MAX232IN, SKXBee, software, etc implemented in this project.
- **Chapter 4** includes the development and design of the project which is mainly concentrated on receiver and transmitter site of prototype system. Programming Flow and some project related issue will be discussed in details too in this chapter.
- **Chapter 5** is the presentation of results obtained in this project and analysis towards the results obtained. Some designing issue during the project development phase will be discussed in this chapter too.
- **Chapter 6** is the last chapter and it sum up what the author done in this project. Possible improvements also discussed in this chapter.

1.5 Challenges

The main challenge in this project is programming part of the Freescale's medical development kit. As in course syllabus, programming Freescale's microcontroller is not included in any subject. Working on Freescale's medical development kit is totally new thing to the author and yet the development kit product is new (released November 2010). There are not much information about the kit and not many people has used this kit. Another challenge is the difficulties in purchasing ECG accessories such as ECG electrode and ECG cable in Malaysia. Although there are many medical supplier in Malaysia but they rarely sold the medical accessories in small quantity and often offer very high price for retail buyers. The author has no choice and chooses to purchase accessories from United State and China which offer reasonable price even after addition of shipping. Insufficient budget also constraints author from purchasing more useful components that help enhance the project. Due to insufficient time and heavy work load, the author not possible to develop hardware and software simultaneously, thus medical server is not develop yet for this project and there are improvements to implement in the future on this prototype system. Challenges and barriers have to overcome in order to succeed in this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Telemedicine

The prefix tele derives from the Greek meaning ‘far’ or ‘at a distance’ or ‘remote’. Hence the word telemedicine signifies as medicine delivered at a distance (Norris, 2002). A more accurate and informative definition of telemedicine defined as the transfer of electronic medical data from one location to another (Telemedicine Research Centre, 1999).

Telemedicine is a branch of e-health that uses communication networks or information technologies (IT) for delivery of healthcare services and medical education from one geographical location to another. It is deployed to cope with issues like uneven distribution and shortage of infrastructural and human resources (Sanjay and Victor, 2007). In other ways, telemedicine may be as simple as two health professionals discussing a case related to health care over the telephone, or as complex as using satellite technology and video-conferencing equipment to conduct a real-time consultation between medical specialists in two different countries. (Meystre, 2005)

2.2 Telemedicine Concepts

Telemedicine is practised on the basis of two concepts: real time (synchronous) and store-and-forward (asynchronous).

Real Time Telemedicine (synchronous) is referred to as two way interactive television (IATV). In another meaning, it can be as simple as telephone calls or as complex as sophisticated virtual reality (VR) robotic surgery or tele-surgery. In it providers/patients at different locations interact with each other using communication technology in the form of audiovisual and wireless or microwave signals. Apart from video-conferencing, peripheral sensing devices can also be attached to the patient to aid in interactive examination. Besides that, it can also be used for long term monitoring for home care patients. In fact, due to the high cost constraints, quality and continuity of care issues, mal-distribution of physicians in different geographic regions and scarcity of the same, remote home care of chronically ill patients and of long term care patients, is the fastest emerging use of telemedicine. Specialities for which it is used most frequent are psychiatry, internal medicine, rehabilitation, cardiology, paediatrics, obstetrics and gynaecology, neurology. (Meystre, 2005)

Store and Forward (asynchronous) technology involves acquiring medical data (images, bio-signals) and transmitting this data to a medical specialist for consultation, evaluation or other related purposes. It does not require simultaneous communication between both persons in real time. Tele-radiology and tele-dermatology is the fastest emerging branches that use such kind of services. Overall radiology, pathology and dermatology are most tending for utilizing this mechanism.

These basic telemedicine technologies as mentioned previously are utilized for providing various health care services that spawns numerous specialties and can be broadly categorized as telehome Home Health Care, telepsychiatry, teleradiology, general telemedicine, telecardiology, telemedicine consulting (teleconsultation), teledermatology, emergency telemedicine, telepathology, teledentistry, telesurgery, telediagnostic, telemonitoring, telecare and teleeducation. Among these specialties, teleconsultation is one of the most significant applications as it uses multimedia

telecommunication through networks for medical consultation. It can either use ordinary telephone, email, or video-conferencing equipments. Real-time consultations use the video-conferencing technology and permit the interaction and communication between medical experts and clients. (Meystre, 2005)

2.3 Current Situation of Telemedicine

Telemedicine is a growing field which has a high potential for improving accessibility to services, quality and continuity of care and significant savings in the overall cost of healthcare. However the use of telemedicine applications has not spread as extensively as other commonly used engineering techniques, such as medical imaging. (Shimuzu, 1999) Although telemedicine applications have proliferated in recent years, their diffusion has remained low in terms of the volume of consultations especially in Malaysia. This is not because telemedicine is less important but because supporting technologies have been traditionally costly, institutionalized and less pervasive and less capable in terms of data transfer speed and quality. They need to develop technically feasible, medically valid, reimbursable, and institutionally supported applications in order to justify the value of telemedicine and engender consistent and frequent use by medical experts or physicians. (Tanriverdi & Iacono) Fixed communication network has been used in different telemedicine setup for some years and it has shown its values, whereas wireless technologies within telemedicine have been developed only in the last few years. One of the sole decisive factors that will cause a telemedicine system successful, in urban and rural areas, is the application of modern communication technology for information exchange between a homecare patient and the medical specialists providing care.

2.4 Heart and Heart-Electrical Activity

The heart is the organ responsible for pumping blood throughout the body. It is located in the middle of the thorax, slightly offset to the left and surrounded by the lungs. The heart is composed of four chambers; two atriums and two ventricles. The right atrium receives blood returning to the heart from the whole body. That blood passes through the right ventricle and is pumped to the lungs where it is oxygenated and goes back to the heart through the left atrium, then the blood passes through the left ventricle and is pumped again to be distributed to the entire body through the arteries (Carlos, Americas & Guadalajara, 2010).

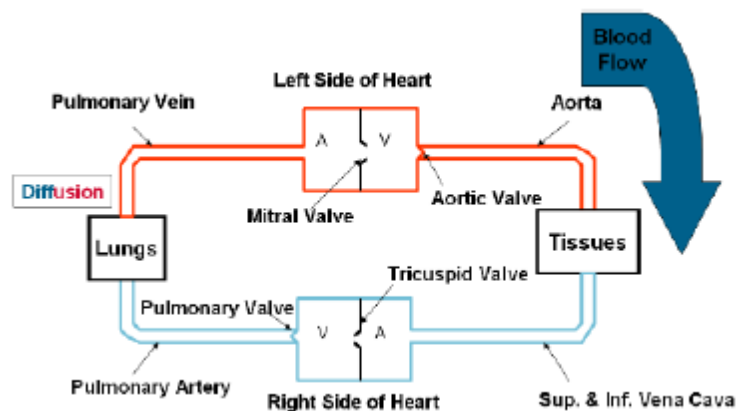


Figure 2.1: Blood circulation scheme

Electrical heart activity is based on depolarization and re-polarization of myocardial cells. The electrical impulse starts in the sinoatrial node (natural pacemaker) flowing through the atriums to reach the atrioventricular node and generating the atrium contraction. The current then flows through the His Bundle reaches the ventricles and flows through them generating the ventricular contractions. Finally, the current reaches the Purkinje fibers and re-polarization of the heart tissue occurs (Carlos, Americas & Guadalajara, 2010).

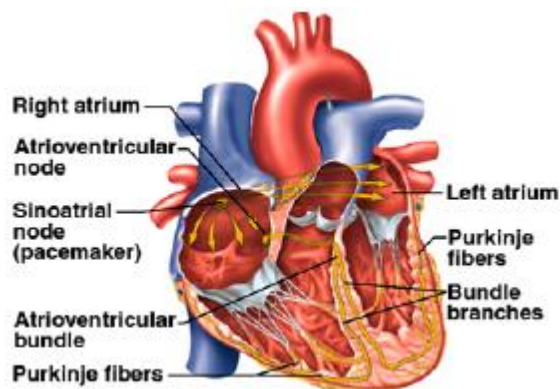


Figure 2.2: Cardiac conduction system

The electrical potentials generated by the heart can be represented as vector quantity. For understanding purposes, the heart is represented as a dipole located in the thorax with a specific polarity at a certain moment, and an inverted polarity the next moment. The potential in a specific moment is defined by the amount of charge and the separation between charges. Figure show the list of events that occur in the heart on each heart beat.

- 1) Atrium begins to depolarize
- 2) Atrium depolarizes
- 3) Ventricles begin to depolarize at apex. Atrium repolarizes
- 4) Ventricles depolarize
- 5) Ventricles begin to repolarize at apex
- 6) Ventricles repolarize

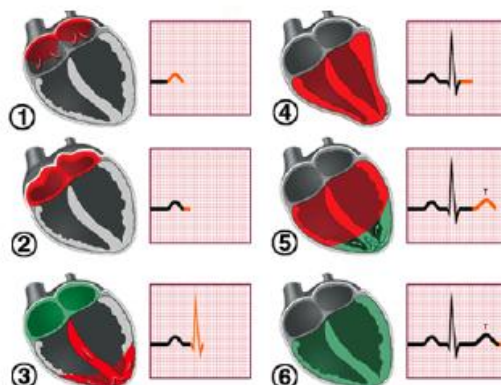


Figure 2.3: Myocardium electrical activity

Each pair of electrodes or an electrodes combination is defined as lead. There are three basic leads used for cardiology. Lead I is at 0° , lead II is at 60° , and lead III is at 120° . The three basic electrode leads make-up the frontal-plane. Electrodes are placed on the limbs; left arm (LA), right arm (RA), and left leg (LL). Those connections are due to the legs and arms being used as a “wire” to detect the bio-potentials that occur in the chest (Carlos, Americas & Guadalajara, 2010). The graphic representation of each lead is shown in Figure 2.4.

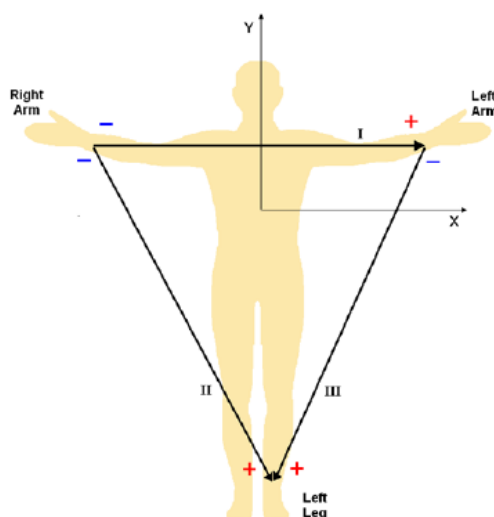


Figure 2.4: Einthoven Triangle

Einthoven's triangle is known as the "three lead" ECG, with measurements taken from three points on the body. If two leads are connected between two points on the body will forming vector between them, electrical voltage observed between the two electrodes is given by the dot product of the two vectors. Another lead connected at the body acting as ground to protect human body. (Patrick et al., 2002) Figure show the triangle that formed around the heart which refers to as the Einthoven's triangle. The top of triangle is formed by lead I, the left side is formed by lead II and at the right side is formed by lead III (Brenda, Beasley & Michael, 2003). Table 2.1 shows the placement of the electrode on the three lead ECG. The most significant among these is lead II because their ability to visualize p wave.

Table 2.1: The placement of the electrode on the three lead ECG

Leads	Positive Electrode	Negative Electrode
I	Left arm	Right arm
II	Left leg	Right arm
III	Left leg	Left arm

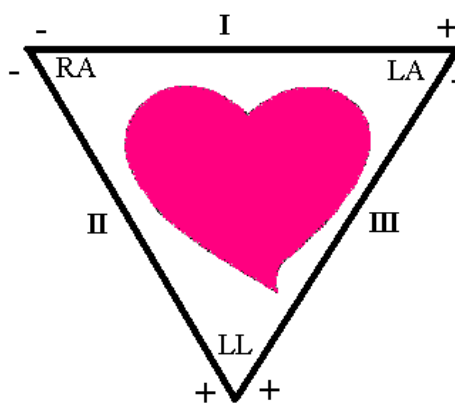


Figure 2.5: ECG lead placement

2.5 Electrocardiograph (ECG)

Electrocardiogram (ECG) is the recording of the heart's electrical activity over time via skin electrodes. The deviations in the normal electrical patterns indicate various cardiac disorders and abnormalities. Cardiac cells, in the normal state are electrically polarized. Their inner sides are negatively charged relative to their outer sides. These cardiac cells can lose their normal negativity through a process called depolarization, which is the fundamental electrical activity of the heart. This process is propagated from cell to cell, producing a wave of depolarization that can be transmitted across the entire heart. This wave of depolarization produces a flow of electric current and it can be detected by keeping the electrodes on the surface of the body (skin). Once the depolarization is complete, the cardiac cells are able to restore their normal polarity by another process named re-polarization. This process also sensed by the electrodes

(Cromwell & Weibell, 2005). In additions, ECG patient simulator is a tool that simulates or recreates an ECG.

2.6 Electrocardiograph Interpretation

The ECG records the electrical activity of the heart over time, where each heart beat is displayed as a series of electrical waves characterized by peaks and valleys. Any ECG gives two kinds of information. First, the duration of the electrical wave is crossing the heart which in turn decides whether the electrical activity is normal or slow or irregular while the second is the amount of electrical activity passing through the heart muscle which enables to find whether the parts of the heart are too large or overworked. (Saritha & Sukanya, 2008) Normally, the frequency range of an ECG signal is of 0.05C , 100 Hz and its dynamic range of 1C, 10 mV. A typical scalar electrocardiographic lead is shown in Fig., where the significant features of the waveform are the P, Q, R, S, and T waves, the duration of each wave, and certain time intervals such as the P-R, S-T, and Q-T intervals.

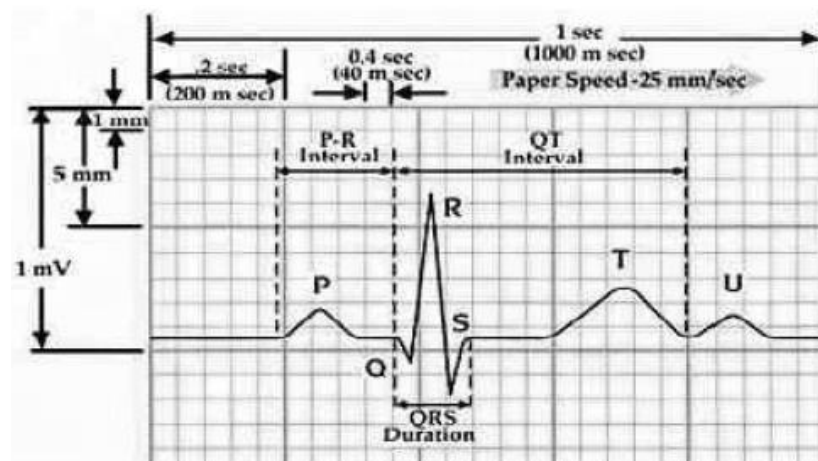


Figure 2.6: Typical ECG signal

In ECG signal, the heart muscles generate different voltages. The P wave represents the atrium contraction. QRS complex and the T wave represents the ventricles actions. Each time that this signal is present, a heart beat is generated. For

this reason it is important to develop analog and digital signal conditioning. First, it is necessary to amplify the signal and filter the noise, and then extract the QRS complex (Carlos, Americas & Guadalajara, 2010).

Noise and interference signals acquired in this type of system are caused by the electric installation. The small electrical signal from the heart generates a common-mode voltage and noise in the system. The signals from the heart are too small and it is necessary to amplify the signal and reduce the common-mode voltage on the system. Other aspects that generate noise are muscle contractions, respiration, electromagnetic interference, and electromagnetic emissions from electronic components (Carlos, Americas & Guadalajara, 2010).

In the normal sinus rhythm (normal state of the heart) the P-R interval is in the range of 0.12 to 0.2 seconds. The QRS interval is from 0.04 to 0.12 seconds. The Q-T interval is less than 0.42 seconds and the normal rate of the heart beat is from 60 to 100 beats per minute. So, from the recorded shape of the ECG, the author can conclude whether the heart activity is normal or abnormal. The electrocardiogram is a graphic recording or display of the time variant voltages produced by the myocardium during the cardiac cycle. The P, QRS and T waves reflect the rhythmic electrical depolarization and repolarization of the myocardium associated with the contractions of the atria and ventricles and very useful in diagnosing various abnormalities and conditions associated with the heart. (Saritha & Sukanya, 2008)

Table 2.2: ECG waveform

Amplitude	Durations
P-wave - 0.25 mV	P-R interval: 0.12 to 0.20 ss
R-wave - 1.60 mV	Q-T interval: 0.35 to 0.44 s
Q-wave - 25% R wave	S-T interval: 0.05 to 0.15 s
T-wave - 0.1 to 0.5 mV	P-wave interval: 0.11 s
	QRS interval: 0.09 s

The horizontal segment of ECG waveform preceding the P-wave is indicated as the baseline or the isopotential line. The P-wave represents depolarization of the atrial musculature and the QRS complex is the combined result of the repolarization of the atria and depolarization of the ventricles, which occur almost simultaneously. The T wave is the wave of ventricular repolarization. Consequently, the duration amplitude and morphology of the QRS complex is useful in diagnosing cardiac arrhythmias, conduction abnormalities, ventricular hypertrophy, myocardial infection and other disease or abnormalities (Li & Zheng, 1995).

2.7 Electrocardiograph Electrode

Electrode is not the same concept as lead. An electrode is a physical patch which connects to the patient. Meanwhile, a lead is a specific vector in which voltage is measured. ECG electrodes are used for sensing bioelectric potential (electrical activity) as caused by cardiac muscle. The electrical activity can be seen as a constant DC electric field or a constant flux of charge-carrying particles or current. The electrodes work as transducers converting ionic current flow from the body into the electron flow of the metallic wire, and consequentially ECG signal can be diagnosed after amplified and processed. A high ionic concentration gel is therefore normally used in the skin electrode interface to increase conductivity. (Aily, 2009)

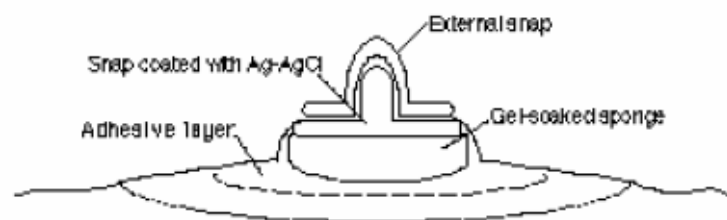


Figure 2.7: Electrodes construction

The choice of material is important as well because the small electrical charge at the skin-electrode interface vary with different electrode materials. The best currently available are gold, platinum, stainless steel, while the most common used is the silver chloride electrode. (Aily, 2009)

Another sensor that was considered was the piezoelectric sensor. Piezoelectric materials generate an electric potential when mechanically strained. During a heart beat, the pressure in the blood vessels is higher than when the heart is in its resting stage. This higher blood pressure causes a physical deformation in the skin, and thus a piezoelectric sensor can produce an electric potential during every heartbeat. The principal reason why the piezoelectric sensor is less than ideal is that it is pressure sensitive. (Aily, 2009) In order to pick up a signal the user (elderly, family members, etc) would have to press the sensor hard against the patient which could cause a permanent deformation of the piezoelectric material. Thus, the silver chloride electrode (inert, cheap, biocompatible) is used in this project rather than the piezoelectric electrode to give best performance of ECG waveform and avoid for possible complications occur if the author use the piezoelectric sensor.



Figure 2.8: Silver Chloride electrode

2.8 Electrocardiograph Interference Source

In order to design an effective wireless ECG, the author have to cater the possible interferences that might exist when the author undergo the data capturing on the patient. The interference sources can be divided into 3 distinct groups:

- 1) Noise originating from sources external to the patient.
- 2) Interference originating from the patient.
- 3) Unwanted Potentials as well as interference originating from patient-electrode contact.

2.8.1 Noise Originated from Source

The noise originating from source to patient generally divided into two type such as electrostatics charges and electromagnetic induction. When a charged body is brought up close to an uncharged one, an equal & opposite charge develops on the uncharged body. For example if an unearthed body is close to any electronic device that is connected to the mains supply voltage, the body will develop a surface charge of equal & opposite potential even though no current is flowing between the two bodies. This phenomenon is commonly known as ESD (Electrostatic Discharge). ESD has been well documented in the recent past and extends a lot further than this particular case. The process of electron transfer as a result of two objects coming into contact with each other and then separating is known as 'triboelectric charging'. As the mains potential has a frequency of 50 Hz, the induced potential will also have this frequency. Other sources of electrostatic charge include the operating table, other persons, electronic equipment. (Anwar, 2005)

An interference that occurs in the vicinity of wires carrying AC currents. Due to the generation of a magnetic field by the flow of a current, all conductors carrying mains currents are surrounded by electromagnetic fields. The South African Department of Public Works has published a document categorizing the uninterrupted power supply in South Africa. The document states that mains supply is at a frequency of 50Hz with a tolerance of 2Hz. The frequency may be anywhere in the range of 48Hz – 52Hz. The 50 Hz mains interference is a difference in potential, relative to ground, that is impressed upon any patient/subject in proximity to the wire carrying alternating (50Hz) main supply current; the patient takes on a potential that is neither that of ground, nor that of the mains, but rather, somewhere

in between. Since the mains current is fluctuating (AC), the induced voltage of the subject is also fluctuating. The effect is however minimized by the fact that the electromagnetic field generated by the live wire is to a large degree cancelled out by the neutral cable flowing adjacent to the live cable but in the opposite direction. (Anwar, 2005)

2.8.2 Noise Originating from Patient

An electromyogram (EMG) measures the electrical activity of muscles at rest and during contraction. Analysis of the (EMG Brain and Nervous System Health Center, 2005) shows that the frequency (Hz) components of both the EEG & ECG both lie within the same band. The EMG signal however is typically five times larger (up to 30mV) than that of the ECG signal. Muscular activity (especially shivering) can lead to large interference in any ECG signal since they occupy the same frequency band. (Anwar , 2005)

2.8.3 Noise Originating from Electrode Contact

ECG electrodes do not act as a passive non-invasive conductor. The placement of any metal adjacent to an electrolytic solution (gel on ECG pads combined with surface of skin) produces an electrochemical half-cell, similar to (although a lot less complex) than that of a battery, resulting in potentials on the surface of the skin. If a differential amplifier is connected to a pair of such electrodes it will amplify any difference in potentials. Ideally if the cells are identical the output will be zero. If the potentials however are not identical, any difference between the two electrodes will be amplified. Additionally, the small current produced by the offset potential may result in polarization. Polarization of the electrode will further distort any signal. (Anwar, 2005)

2.9 Wireless ECG system (Quick Doc ECG)

Patients who have survived cardiac arrest, ventricular tachycardia or other cardiac disease are at a higher risk of sudden cardiac death. Many of these patients are living at home without any kind of cardiac monitoring systems. By using a wireless and wearable monitoring system for detection of arrhythmia it is possible to alert healthcare professional to the patient's condition so that the necessary action for an emergency rescue can occur. Advanced monitoring solutions using telecommunication systems are used for remote ECG diagnosis. Such systems can be divided into two categories real time mode and store-and-forward mode. The systems available today are either based on standard ECG electrodes and a wired connection to a recording device or by pressing a recording device directly onto a patient's chest when a symptom arises (Kong, Ng & Ong, 2000). Recent developments in wearable biomedical sensors have opened up possibilities for continuous wireless ECG monitoring systems.

The wearable wireless ECG (Quick Doc ECG) was used to detect Respiratory Sinus arrhythmia (RSA) in a patient. Many current ambulatory ECG recording equipment are dependent on the patient operating them, depending on the patient's condition this many not be possible. This method is also time consuming. For a reliable monitoring system it is necessary to develop an automatic system that will monitor the patient and send alarm conditions to a central safety alarm system. Given the fact that time is of the essence during a cardiac arrest this device has the possibility of increasing the survival chances of a patient. The idea is to develop a simple smart sensor that detect critical cardiac conditions and give early alarm signals even if the patient is unconscious or unaware of cardiac arrhythmia.

The sensor transmits the ECG information to a device on the patient. The Quick Doc ECG utilizes silver electrodes which acquire the signal and sent it to the amplification and transmission stages via short shielded wires. The wires are also twisted together to reduce noise. The Quick Doc ECG then transmits the information wirelessly to a receiver then to a computer. The Quick Doc system performs post processing on the ECG signal after acquisition but with some modifications can be

made to produce real time post processed data. In the paper (Crawford, 1999), the patient wears an ECG sensor that employs smart electronic electrodes capable of wireless transmission of ECG signals to a dedicated Hand Held Device (HHD). The HHD monitors the continuously recorded ECG signal and can detect abnormal ECG activity using an automatic arrhythmia detector. Based on this the device transmits alarm conditions to remote Clinical Alarm Station (CAS).

In order to perform continuous ambulatory ECG recordings a new wireless ECG sensor has to be designed which can measure the ECG signal and transmit it continuously to the receiver in the HHD. This means that only one lead is used for the recording of the ECG signal. To accomplish this, the authors (Crawford) used a compacted “double-electrode” with no wires connected (Crawford, 1999). This electrode is equipped with a wireless transmitter and battery supply for several days of continuous usage. The ECG sensor includes two electrical contact points with conducting gel applied to the patient’s skin for obtaining the signals. These points are connected electrically to the electronic circuit that consists of an amplifier, a high pass filter with a cut off of 0.5 Hz and a low pass filter with a cut off of 250 Hz (Crawford, 1999).

The Quick Doc ECG also has a power supply and wireless transmitter thereby making it wearable and viable for continuous ECG monitoring. The power supply in the Quick Doc ECG utilizes a 9V battery and because the power consumption of the system is only 11mW. The electrode designed by the authors uses a combination of real-time mode and store-and forward mode to produce a continuous cardiac event recorder. The ECG signal is picked up by the electrode and transmitted to an RF-receiver which is connected to a standard PDA. The Hand Held Device (HHD) consists of the PDA and the RF receiver.



Figure 2.9: Receiver connected to the PDA.

The PDA then using GPRS transmits the alarm conditions to the remote CAS in the event of an abnormal ECG. The CAS in a hospital will give an alarm to the operator and will display the actual recorded ECG signal from the patient. The wireless module used in this system transmits at 434.44 MHz (Crawford, 1999). The receiver acquires the signal and converts it back into its analog form using an ADC. The PDA is connected to the receiver via a RS232 cable. The PDA contains a program in Labview which analyses the data.

The Quick Doc ECG system follows a similar model in that the ECG acquired from the electrodes is amplified, filtered and then transmitted at 900 MHz to the receiver when it is converted back into analog and then acquired by the DAQ which send it via an USB cable to a computer running Matlab. The electrode design in this paper involves placing two electrodes 3 cm apart from one another and building them into one unit with the electronic circuits and battery supply (Thakor & Webster, 1980). The electrodes are positioned directly on the patient's chest. As the design does not use any wires from the electrodes to the amplifier there is very little power noise. In the Quick Doc ECG system there are short leads that connect the electrodes to the instrumentation amplifier and the electrodes are placed on the wrists. The Quick Doc ECG system provides the user with the cardiac rate and rhythm.



Figure 2.10: Quick Doc ECG System

2.10 Wireless Technology – ZigBee

One of the emerging standards in the move toward a wireless world is an approach called ZigBee (Venkat, 2002). Pioneered by Phillips, it has since formed into an alliance of companies working together to create a wireless communication protocol. The ZigBee stack unlike Bluetooth is relatively straightforward compared to bluetooth. The main purpose of this standard is to provide its users with three main features such as low data rate, low power consumption and low cost. the ZigBee technology operates in the 2.4 GHz ISM band. The maximum data rate achievable on this technology is 250 kbps. On top of that, it caters for a range of between 10 meters to 75 meters depending on the power consumption required for a given application.

The design for ZigBee took into consideration the high power consumption of Bluetooth. For most application, the ZigBee module is capable of a battery life of 6 months to 2 years with AA batteries. This is achieved by using sleep mode functions to allow communication only when the application deems necessary. The ZigBee chip draws a few milliamps in sleep mode against 100 microamps or more for a comparable Bluetooth state. Furthermore this prevents the device from interference problems as it often won't be operating when other modules are using the 2.4 GHz band. The cost of ZigBee solution also lower compare to the Bluetooth solution.

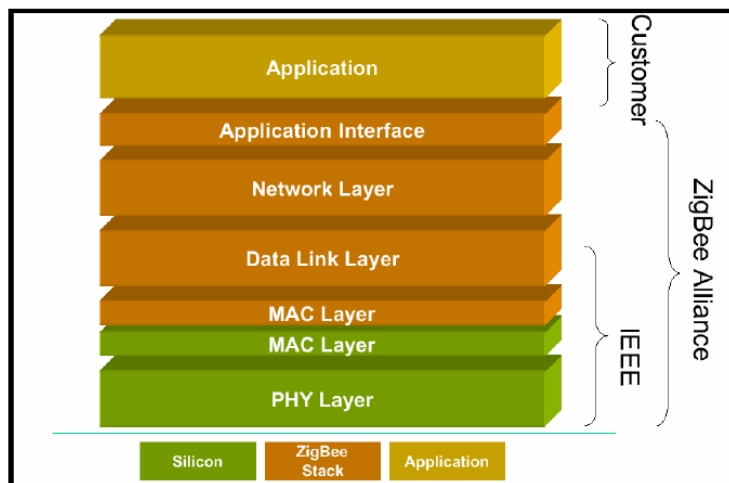


Figure 2.11: ZigBee Protocol Stack

When the author determine what wireless technologies to use in this project, the author always want to compare the available in term of range of coverage, suitability, power consumptions, data rate, operating frequency, complexity, etc. Bluetooth is one of the wireless technology that available now and it did come to our consideration when the author decide what type of wireless technologies that to be implement in this project, ZigBee chosen as the wireless technology that to be implement due to reasons below,

- a) low power
- b) wireless power supply
- c) efficient, reliable and lean communication protocols
- d) easily accessible physical measurements
- e) interaction between instruments and controllers
- f) costs and availability

The most well-suited technology for this design would be ZigBee, mainly due to its power features and relatively simple stack. Bluetooth in contrast, has certain unnecessary and rather complex features in its protocol. Table 2.3 summaries the comparison between the ZigBee and Bluetooth technology.

Table 2.3: Comparison between ZigBee and Bluetooth

ZigBee	Bluetooth
Ideally for a static network, which comprises of a multiple devices communicating with smaller packets	Provides an ideal ad hoc network between capable device for transferring audio, screen graphics, picture and file
Operates in the 2.4 GHz ISM band and maximum data rate achievable on this technology is maximum data rate of 250 kbps	Operating in the unlicensed 2.4 GHz industrial scientific and medical (ISM) band with maximum data rate of 1 Mbps
Caters for a range of between 10 meters to 100 meters (Up to 400 meters)	Caters for range of 10 meters. (Up to 100 meters)
Long battery life. 6 months to 2 years with AA batteries	Short battery life. May fully consume battery within a week if vigorous usage
IEEE 802.15.4 Standard	IEEE 802.15.1
Simpler than Bluetooth	More complicated than the ZigBee
Network topology> ad hoc, star, mesh hybrid	Network topology > ad hoc piconets
Very flexible	Medium flexibility, depend on the profile
Number of device per networks> 2 to 65000	Number of device per networks> 8
Maximize slave power requirement	Maximize ad hoc functionality

2.11 Current Application of Wireless ECG System

There are several research had been done in ECG sensor development especially that used wireless IEEE 802.15.4 as a transmission medium. The typical application scenarios of wireless ECG sensor are various. Smart home health monitoring (Thakor and Webster, 1980), for example, expects to help older people or the patients with chronic disorders to increase the chance of survival ability live. Smart ward in the hospital reduce the time of routine check-up and its real-time monitoring also allows emergency situation to be handled immediately. Moreover personal

contacts can be avoided for the nurses to reduce the possibility of infection in a contagion ward. Wireless ECG sensor also can be utilized for athletic performance monitoring, for example, tracking one's pulse and respiration rate via sensors and sending the information to a computer for later analysis.

For emergency medical care application CodeBlue, CodeBlue is designed to provide routing, naming, discovery and security for wireless medical sensors, and other devices that may be used to monitor and treat patients in a range medical setting. CodeBlue is designed to scale across a wide range of network densities, ranging from sparse clinic and hospital development deployments to very dense, ad hoc deployments at a mass casualty site. Wireless ECG sensor can be used to capture real-time vital signs from patients in a moving ambulance, relay the data to handheld computers carried by physicians for pre-hospital diagnosis. The use of ad hoc networking in CodeBlue will allow the "mesh" of connectivity to extend across an entire building or between multiple, adjacent facilities. Additional coverage, if necessary, will be possible with placement of fixed nodes in hallways, rooms, or other areas. (Nor Syahidatul et al., 2008)

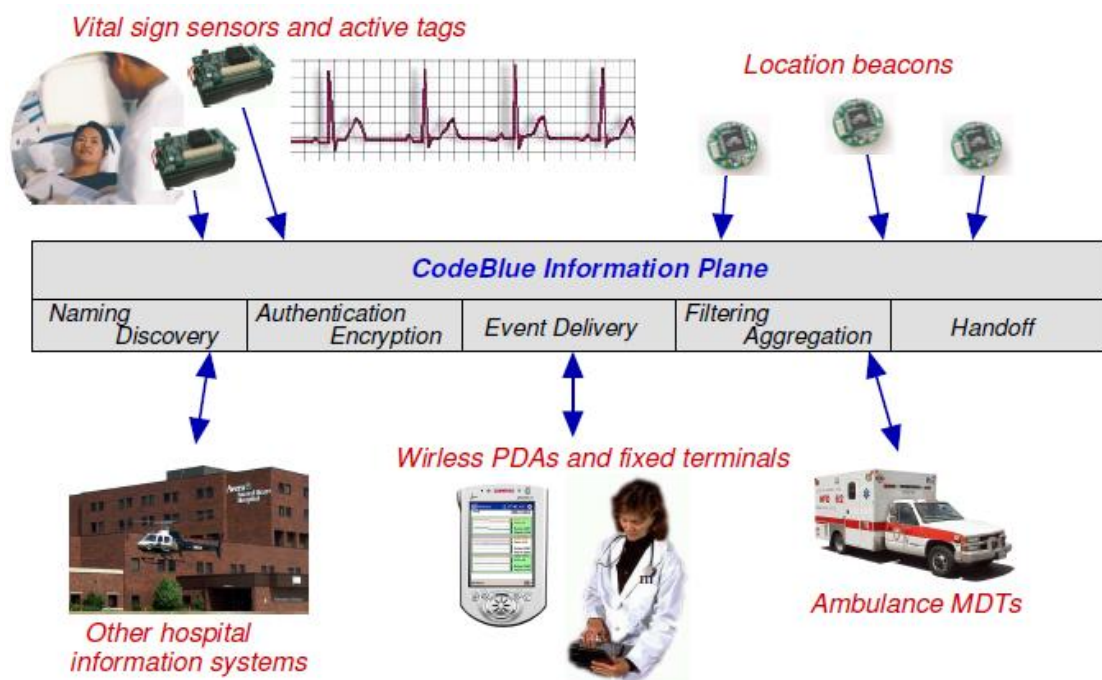


Figure 2.12: Code Blue infrastructure

Another emergency medical care existing application is MOLEC. The MOLEC Monitor is capable storing the ECG signal and embedded real-time system that captures, processes, detects, analyzes and notifies possible dangerous abnormalities to an alarm centres through the network from anywhere and at any time. The MOLEC Centre is the system part that manages the communication with all the PDA monitors and updates the MOLEC Centre's database with the new information that receives from each of them. The Alarm Center receives all the risk alarms detected into the PDA, in order to react and immediately provide proper medical assistance. (Nor Syahidatul et al., 2008)

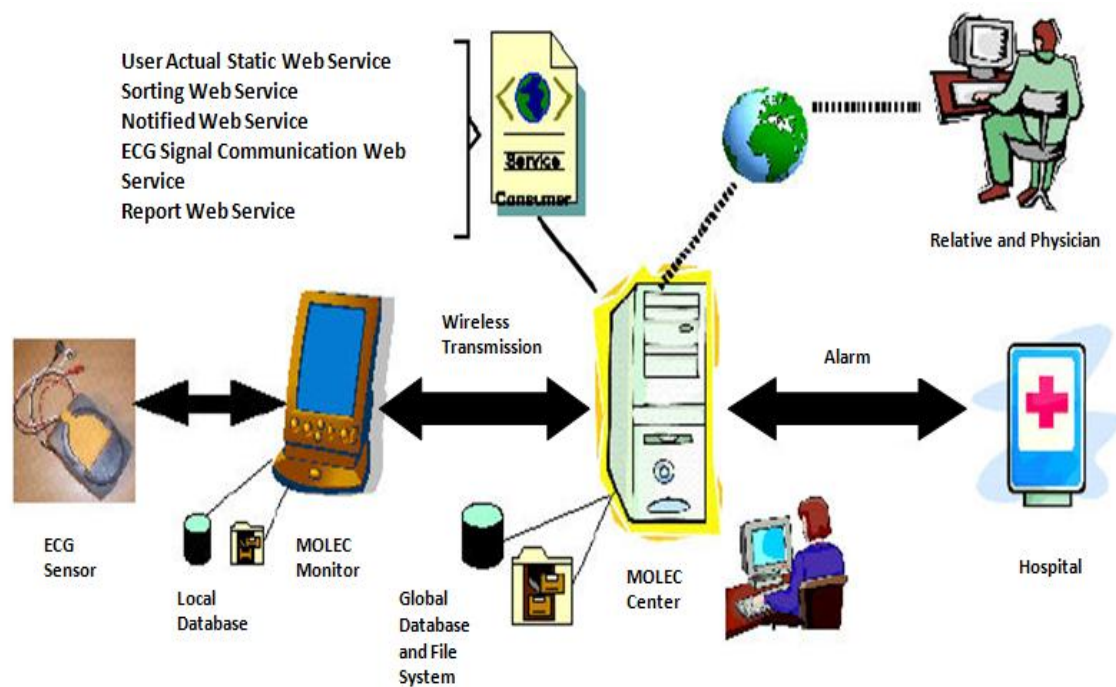


Figure 2.13: MOLEC infrastructure

CHAPTER 3

METHODOLOGY

3.1 Scope of Works and Work Flow

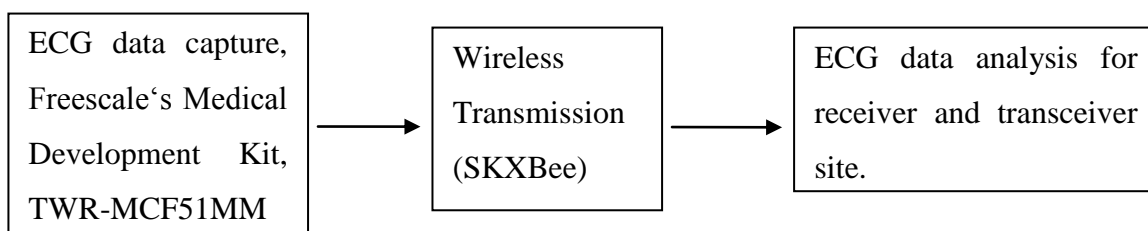


Figure 3.1: Scope of project

In order to achieve the objective of this project, the overall scope of project has been outlined as figure above. As stated in the objective, this project generally consists of two parts, which are hardware development and implementation of wireless module to the prototype board. In this report the details of work on ECG implementation will be illustrated. For the hardware development, ECG prototype board will be developed using Freescale's medical development kit, TWR-MCF51MM. Programming of TWR-MCF51MM is the most challenging task as this kit is a new product (just released on November 2010), there are not much information and support regarding the development kit. Implementation of ECG function on the development kit will be the first and most important milestone before proceed to wireless transmission of ECG data via SKXBee. After successful of implementing

ECG data capturing, wireless receiver and transceiver site of ECG data will be developed using Cytron's SKXBee. The captured ECG signal only concentrates on normal sinus rhythm strip (normal heart beat) rather than abnormal heart beat. Following is the main module, items and equipments using in this project.

- 1) Freescale's TWR-MCF51MM kit consists of 4 major components such as TWR-ELEV, TWR-MCF51MM, TWR-SER and MED-ECG.
- 2) Cytron's SKXBee module.
- 3) Shanghai Jun Kang Medical Product's ECG electrode (AgCl electrode)
- 4) Welch Allyn's AHA 3 lead ECG cable (red, white and black)
- 5) Two computer or laptop (serve as host in wireless receiver and transceiver site)
- 6) MAX232 circuitry on breadboard.

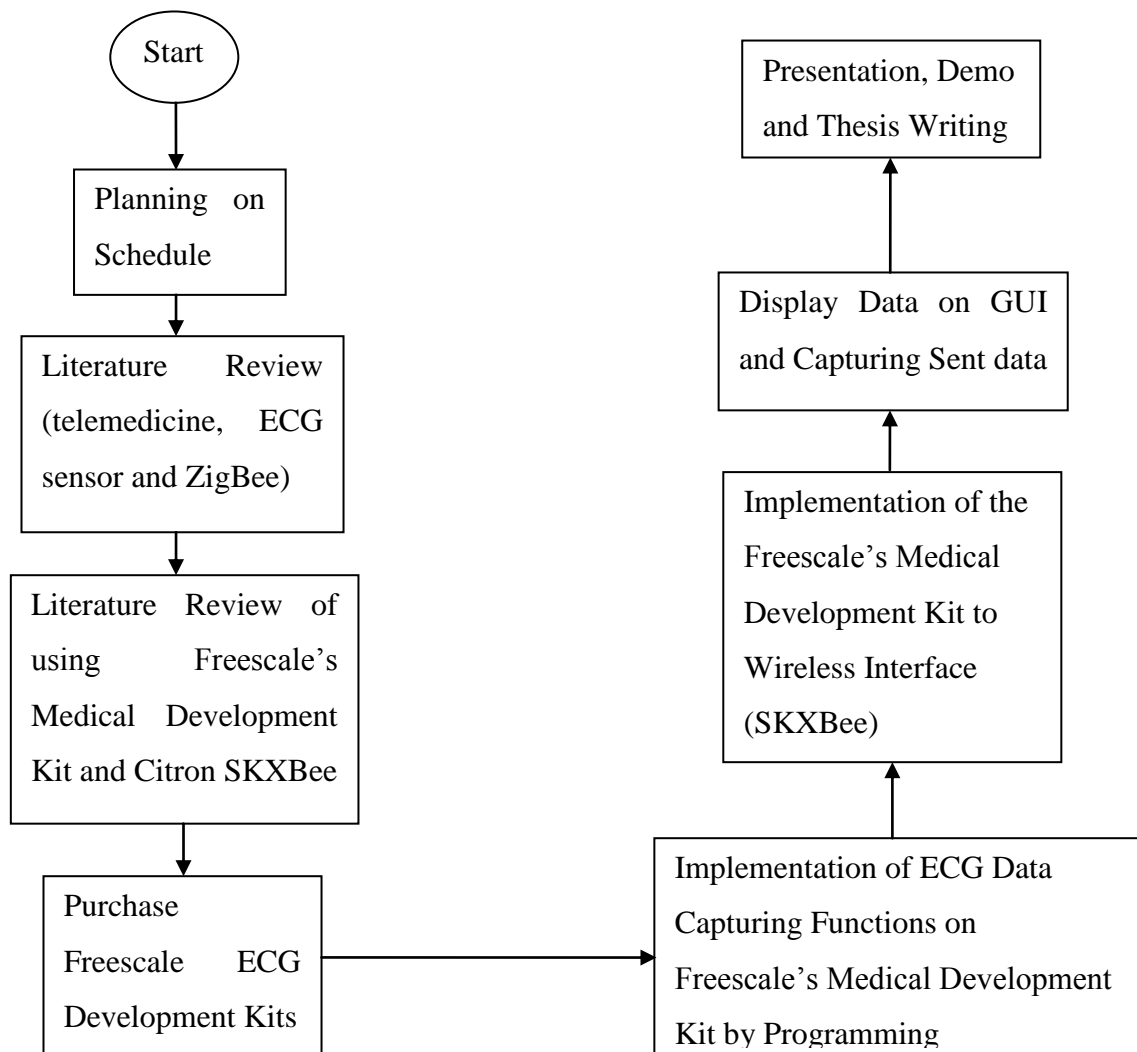


Figure 3.2: Work Flow of the project.

3.2 Work Breakdown

The main process in completing the project is basically divided as study, design and implementation. Study is the process that continuous throughout the project, from the beginning till the end. At the beginning of the project, a lot of study has been done which include study on the concepts of ECG signals, analysis signal, programming of Freescale's microcontroller, usage of SKXBee, etc. Meanwhile in the design phase, the project system will be developed according system block diagram and programming on microcontroller. In the implementation phase, the project will be implemented with the hardware and wireless approaches. The hardware approach is using the entire prepared component to build the system. Then for the wireless approach is application of two SKXBee to implement the wireless transmission from transceiver to receiver site. Then the analysis, coding and testing will be done on the overall system to ensure the output of the system is under consideration and expected. Finally the improvement is done to enhance the system performance.

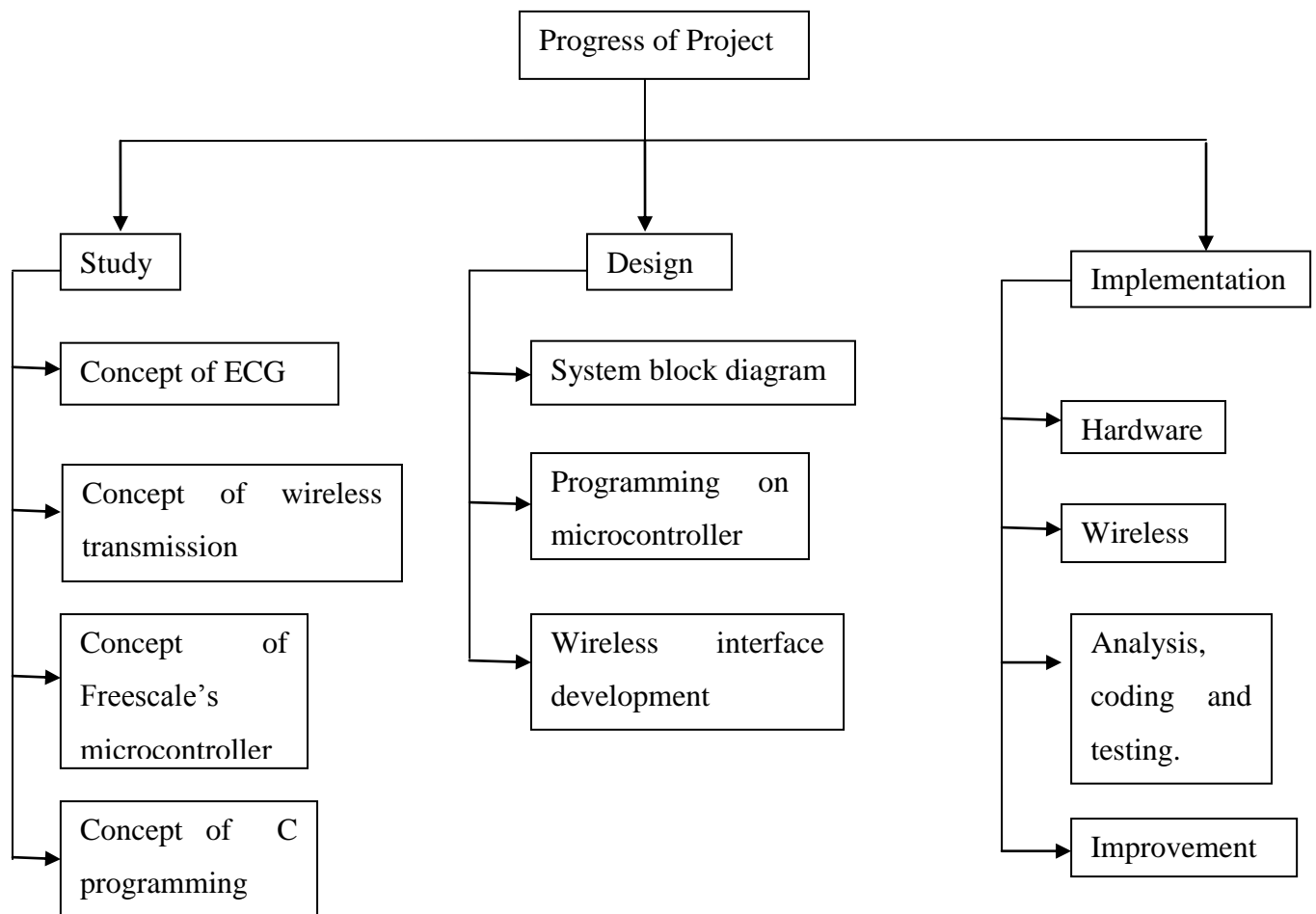


Figure 3.3: Work Breakdown

Meetings with Supervisor															
Preparing and Writing Progress Report															
Preparation for Presentation															

Figure 3.4: Gantt Chart Semester I

3.3.2 Gantt Chart Semester II

The planning of the project activities for the semester II is integration of Freescale's medical development kit into the wireless module (SKXBee). Basically it is focused on programming on Freescale's medical development kit for the ECG data capturing, secure communication network between transceiver and receiver of the ECG data, and displaying ECG data in Freescale's MED-ECG GUI. This project generally will follow the following schedule so that the ECG prototype system manages to produce in time.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Plan														
Part 1 of Project-Implementation of Medical Interface	█	█	█	█										
Part 2 of The Project-Implementation of Wireless Interface				█	█	█	█	█	█	█	█	█		
Part 3 of Project-Post Project Improvement							█	█	█	█	█	█	█	
Preparation of Thesis	█	█	█											
Writing of Thesis				█	█	█	█	█	█	█	█	█		
Checking and Improve Thesis												█	█	
Group Meetings		█		█		█		█		█		█		
Meetings With Supervisor	█	█	█	█	█	█	█	█	█	█	█	█	█	
Project Demonstration														█
Project Presentation														█

Figure 3.5: Gantt Chart Semester II

3.4 Architecture for Project

The ECG signal captured by the ECG silver chloride surface electrode then flow via Welch Allyn's AHA 3 lead ECG cable to the Freescale's medical development kit to process. Processed ECG data will be display in Freescale's MED-ECG GUI on a computer or laptop via USB interface. There is another output of ECG data from Freescale's medical development kit (specifically TWR-SER, DB9) via RS232 interface will be directed to MAX232 circuitry to undergo conversion of signal. MAX232 circuitry will convert RS232 data to TTL/CMOS (5V) to enable interfacing with the SKXBee. After signal conversion in MAX232 circuitry, the data will be send to Rx pin of SKXBee and transmit to another SKXBee (connect with another computer) which serve as receiver at the other site. Wireless transmission of ECG data at receiver site monitored using X-CTU software and saved by RS232 data logger in txt file form.

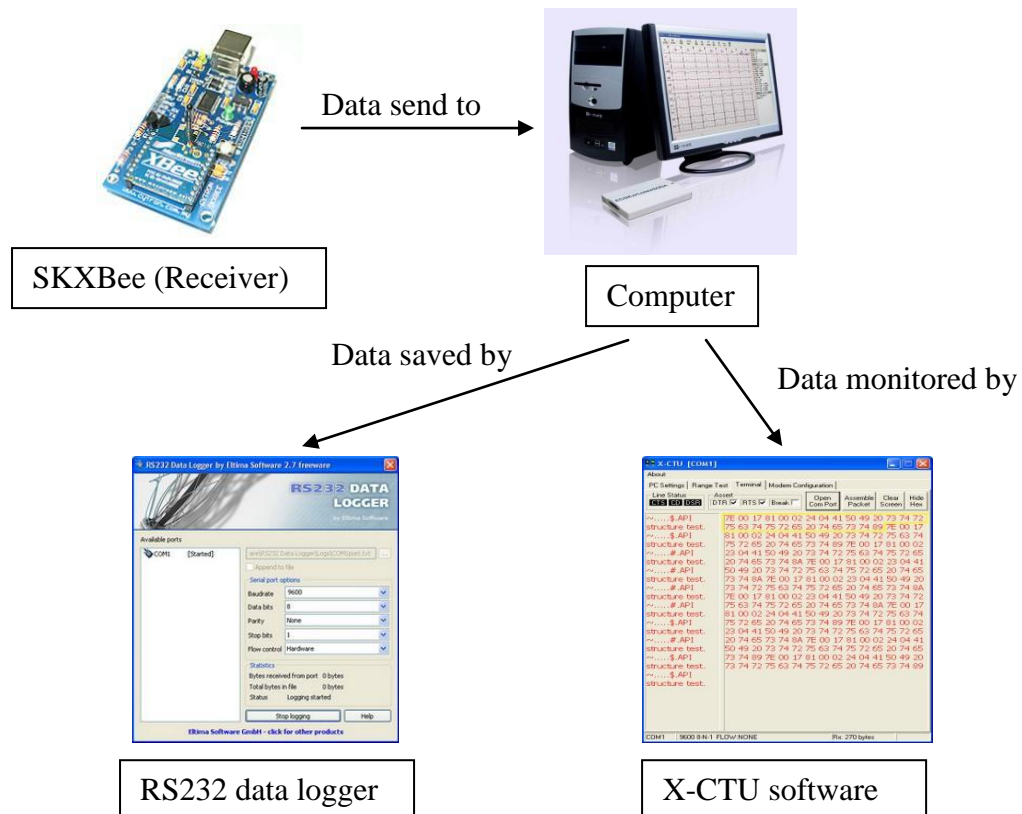


Figure 3.6: Architecture of Receiver Site

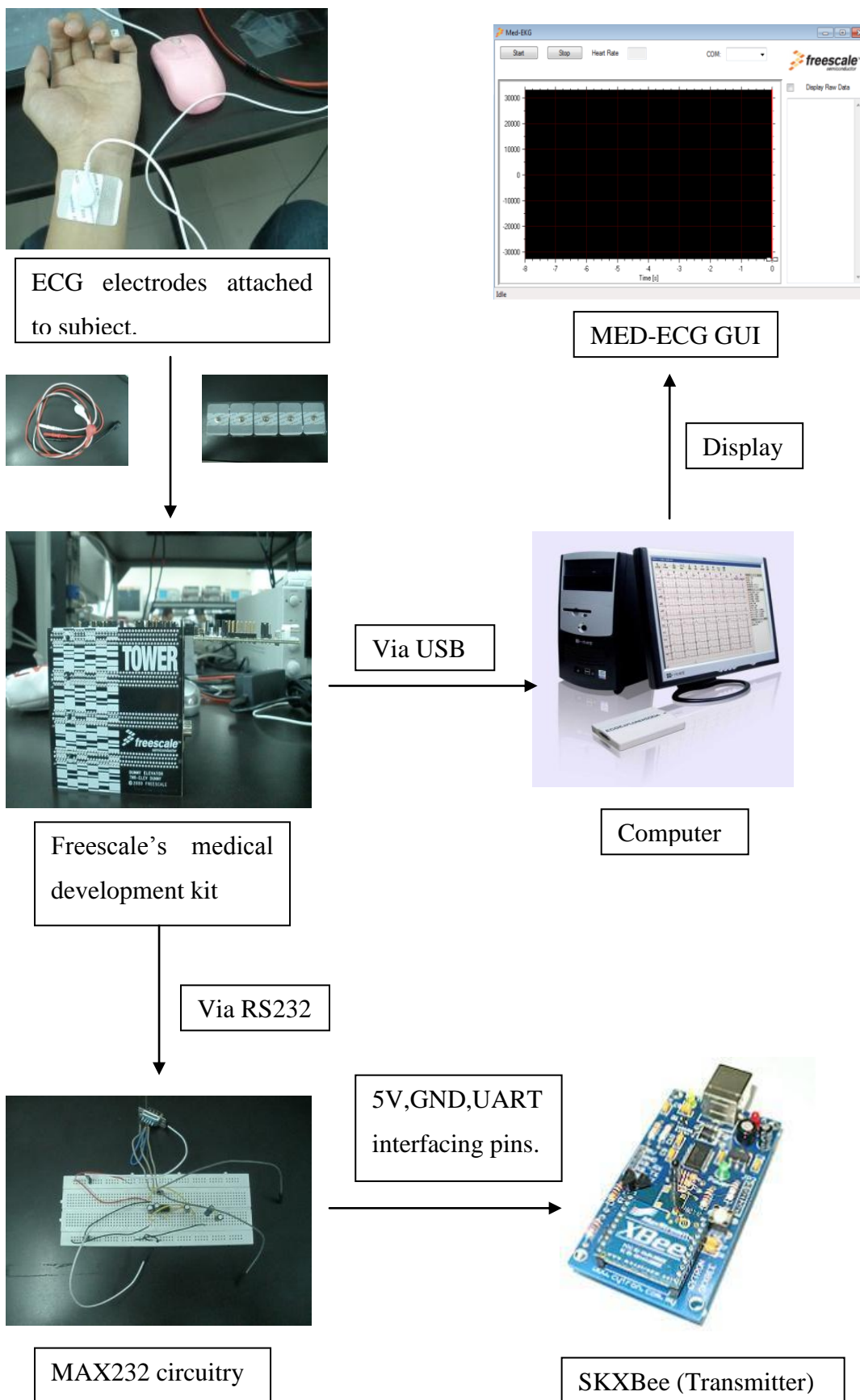


Figure 3.7: Architecture of Transmitter Site

3.5 Freescale's medical development kit (TWR-MCF51MM)

TWR-MCF51MM-KIT is a medical-oriented development tool for the MCF51MM256 microcontroller and full featured modular development platform using the Freescale Tower System that allows for quick code development and easy prototyping. (Freescale Inc., 2010) The development kit includes the following components:

- 1) TWR-MCF51MM: A standalone development board featuring the MCF51MM256VLL microcontroller.
- 2) TWR-SER: A serial connectivity board to that supports both USB and RS232.
- 3) TWR-ELEV: Two (2) elevator boards to connect the MCU to additional Tower peripheral modules.
- 4) MED-EKG: A sensor module that detects EKG data (for use with Flexis MM parts only)

3.5.1 TWR-MCF51MM



Figure 3.8: TWR-MCF51MM

TWR-MCF51MM is a low-cost evaluation, demonstration and development board that features a 32-bit MCF51MM256 microcontroller. The TWR-MCF51MM can operate stand-alone or as the main control board in a Tower System with peripheral modules. The following list summarizes the features of the MCF51MM Tower MCU boards:

- Tower compatible processor board
- Open Source BDM (OSBDM) circuit
- Analog measurement circuitry
- 4 LEDs
- DIP Switches and push buttons for user input
- Potentiometer
- MMA7361L three-axis accelerometer
- RS232 transceiver and 2x5 pin header

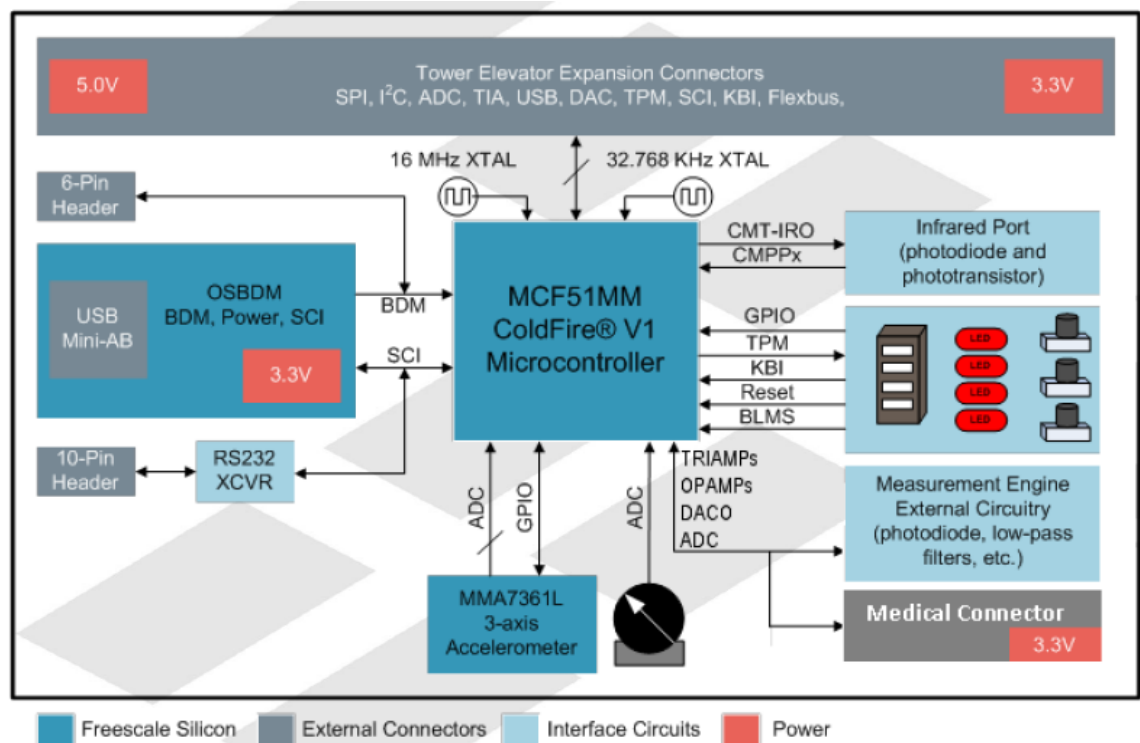


Figure 3.9: TWR-MCF51MM Block Diagram

There are two crystals provided on the board for clocking the MCF51MM256 device such as 16 MHz crystal connected to XTAL2 and EXTAL2 for system clocking and 32.768kHz crystal connected to XTAL1 and EXTAL1 for TOD usage. The TWR-MCF51MM can be powered by the Open Source BDM (OSBDM) circuit via the Mini-B USB connector when stand-alone. When assembled with the Tower System and the TWR-SER is configured to run USB device mode (J16 pin 3 and 4 connected), the Mini-B USB connector is no longer used as a power source and only used for OSBDM debugging purposes. In this case, the power will be supplied from

the Mini-B USB from the TWR-SER. Plug in the Mini-B USB connector from TWE-SER before plugging in the Mini-B USB connector from TWR-MCF51MM. A standard USB A male to Mini-B male cable used to supply power from a USB host or powered USB hub. Jumper, J11, can be used to isolate the 3.3V supply from the microcontroller. This connection can be used to measure the power usage of the MCF51MM256 microcontroller. (Freescale Inc., 2010)

The TWR-MCF51MM features a 2x10 expansion connector J27 to MED-EKG for routing the medical engine signals to external medical board so it can use the OPAMP, TRIAMP, ADC and DAC on MCF51MM to implement the requirement signal conditioning for medical applications. When the DSC MC56F8006 from the MED-EKG is enabled, MCF51MM256 can choose to read the conditioned EKG results output from the DSC via I2C transmission (pin 3 and pin 4). To enable I2C communication, user must assemble the MEG-EKG with the Tower System because the TWR-SER has the pulled up resistors circuit required for I²C transmission. (Freescale Inc., 2010)

The TWR-MCF51MM features two expansion card-edge connectors that interface to elevator boards in a Tower System: the Primary and Secondary Elevator connectors. The Primary Elevator connector, comprised of sides A and B, utilized by the TWR-MCF51MM, while the Secondary Elevator connector only makes connections to ground (GND). The user must ensure correct configuration of tower system has been constructed as misplace the board with the tower will turn no results (no activation of board). An on-board, MC9S08JM60 based OSBDM circuit provides a debug interface to the MCF51MM256. The MC9S08JM60 is a USB-enabled microcontroller with an 8-bit HC9S08 core. The OSBDM circuit provides a USB-to-debug interface that allows run-control and debugging of the MCF51MM256 target device. The USB drivers required to communicate with the OSBDM are provided in development tools such as Freescale CodeWarrior. When TWR-MCF51MM is used stand-alone, this single USB connection can also be used for power. (Freescale Inc., 2010)

3.5.2 TWR-SER



Figure 3.10: TWR-SER

The TWR-SER Serial Module provides USB, Ethernet, CAN and RS232/485 connectivity solutions for designers developing with the Freescale Tower System. This peripheral module is designed to be combined and used with other microcontroller and peripheral modules in the Tower System. (Freescale Inc., 2008) In this project, TWR-SER will route the data from TWR-MCF51MM board to SKXBEE via DB9 (RS232) after voltage conversion at MAX232 circuitry. TWR-SER also supplied the voltage power to the tower system via its USB port. The feature of TWR-SER can be summarized as follow:

- 10/100 Ethernet PHY with MII and RMII interface
- Ethernet connector with integrated magnetic and LEDs
- RS232 and RS485 transceivers and single DB9 connector
- CAN transceiver with 3-pin head

3.5.3 TWR-ELEV

The TWR-ELEV Elevator modules are the basic building block of the Freescale Tower System. Designed to connect microcontroller and peripheral modules, the Elevator modules provide the power regulation circuitry and structural integrity

needed for all configurations of an assembled Tower System. In this project, the USB wall plug will connect with the TWR-ELEV to supply the voltage to this kit. There are two elevators in the TWR-ELEV such as primary and secondary or functional and dummy. (Freescale Inc., 2008) User must ensure correct configuration of boards to the TWR-ELEV to allow voltage be supplied to respective board. Figure below demonstrate the feature TWR-ELEV.



Figure 3.11: Feature of TWR-ELEV

3.5.4 MED-ECG

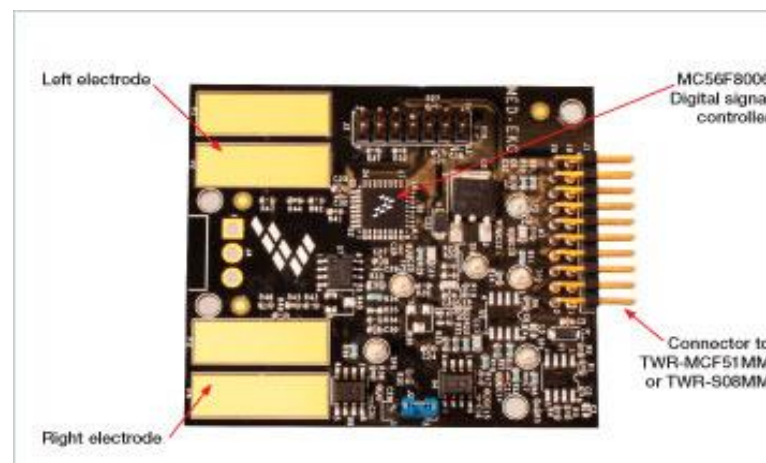


Figure 3.12: MED-ECG

The Medical EKG Module (MED-EKG) is developments board that rapid prototype electrocardiogram (EKG/ECG) application. MED-ECG is designed to work with the TWR-MCF51MM module. (Freescale Inc, 2010) Generally speaking, it is the sensor board that detects and captures the ECG signal from subject via electrodes or on board slide electrode. The following list summarizes the features of the MED-EKG board:

- Freescale Tower System compliant and small form factor.
- External connector to TWR-MCF51MM MCU modules.
- Includes the MC56F8006 DSC used for signal conditioning.
- Electrodes embedded in development board for easy EKG signal detection using fingertips.
- Open connector for any type of EKG electrodes for additional precision.
- JTAG-ONCE connector for DSC programming.
- Different jumper configurations to allow various signal conditions.
- Low power.

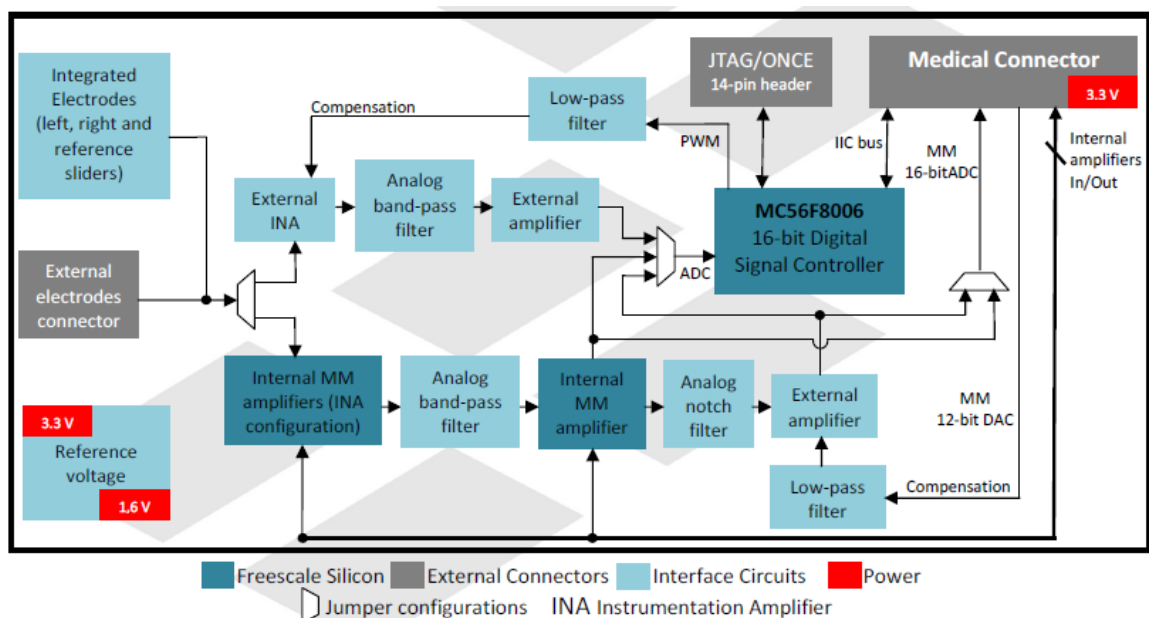


Figure 3.13: MED-ECG block diagram

The MED-EKG is powered by the default trough connector J1, but can be also powered by the JTAGONCE port. Do not apply both power sources at the same time to avoid the damage to board itself. Header J1 (2x10 pins) of MED-EKG

provides connectivity with either the TWR-MCF51MM through the medical connector. It allows the use of internal OPAMPs, TRIAMPs, ADC and DAC of Flexis MM 32-bit or 8-bit microcontrollers to implement the requirement of signal conditioning. Table attached in the appendices A shows the signal present in each pin of medical connector. (Freescale Inc., 2010)

The on-board slider electrodes on MED-ECG are used when external electrodes are not available. The user can place their fingers as is shown in Figure below. Since the contact is not as secure as the external electrodes leads, the user must avoid any small movement to reduce input noise. This method is not a good method for longer period of capturing ECG signal from subject.

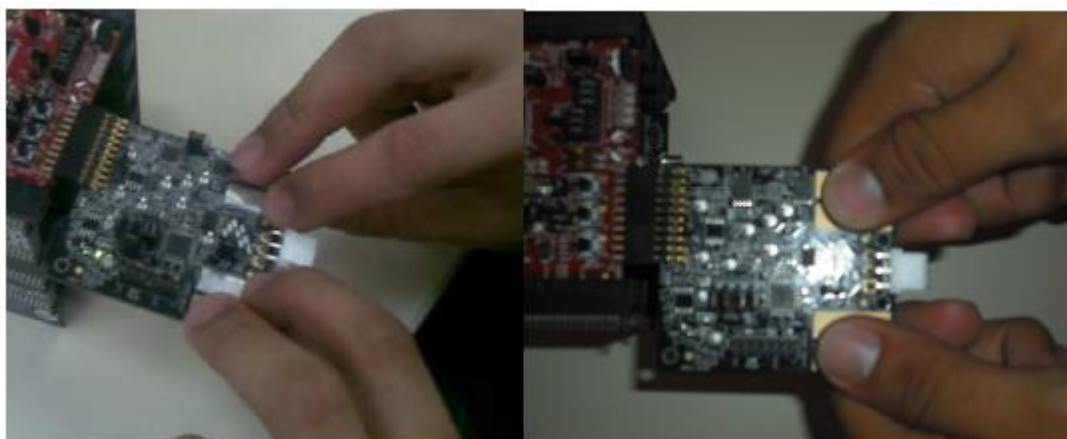


Figure 3.14: Fingertips Collocation on Slider Electrodes

3.6 Cytron's SKXBee (Wireless Module)

SKXBee has been designed for 5V TTL logic interface where no extra voltage divider is necessary. With minimum interface, it is ready to connect to microcontroller for embedded XBee development. Furthermore, on board USB to UART converter offer easy yet reliable communication to PC for functionality test and as XBee dongle. SKXBee can support both XBee and XBee PRO because they are interchangeable and pin-to-pin compatible with each other. (Cytron Technologies, 2008) On board USB to UART converter is design for easy communication with PC

for functionality test and as XBee dongle. The author will need this convertor at the receiver part of our ECG wireless transmission. 5V TTL logic interface with no extra voltage divider offer straight forward interface to microcontroller for embedded wireless development. RS232 output from Freescale's tower system underwent voltage level conversion using MAX232 (convert coltage level from RS232 to TTL, 5V) It has been designed with capabilities and features as follow:

- Support both XBee and XBee PRO modules
- USB Plug and Play UART function
- 5V powered
- 5V UART interface, ready for microcontroller interface
- Default baud rate of 9600bps
- Long Range Data Integrity
- Low power consumption
- Compact yet easy and reliable platform
- As serial port replacement (wireless)
- Point-to-point, point-to-multipoint and peer-to-peer topologies supported

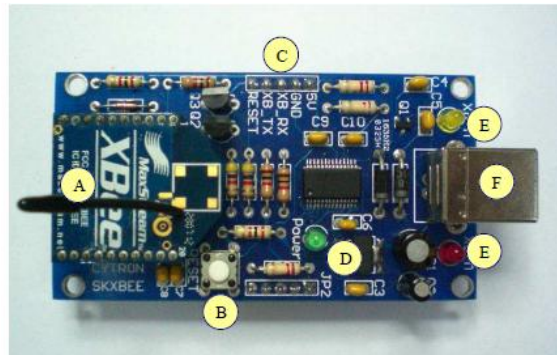


Figure 3.15: SKXBee, Zigbee module

Table 3.1: Function and description of SKXBee components

Label	Function and description
A	Connector for either XBee or XBee-Pro module. In our project, XBee module will be soldered on SKXBee.
B	Reset button for XBee module.
C	5 ways header pin for external power supply and interface to

	microcontroller. If this kit is connected to microcontroller board, it should be powered with 5V. In our project, it will be powered by MAX232 circuits (5V).
D	3.3V power indicator. This small green LED indicates the status of 3.3V from on board voltage regulator. It should be ON if either external 5V power or USB connection is connected to SKXBee.
E	These are a pair of small LED, red and yellow in color. These LEDs are connected to on board USB to UART converter. It indicates the receiver and transmitter activity. It will only work if SKXbee is connected to PC or laptop through USB cable. Red LED indicates USB's transmitter activity, while yellow LED indicate USB's receiver activity.
F	USB B type socket. If connection to PC or laptop is required, connect one end of USB cable (B type) to this socket, while the other end to PC or laptop USB port.

SKXBee has its own interfacing pin instead of USB dongle. It is very convenient to use SKXBee to interface with other microcontroller. When USB dongle is used for SKXBee, all 5 pin connection should not be connected. In this project, the author will connecting the MAX232 circuitry to the UART pin of SKXBee to send the ECG data to receiver site of SKXBee whereas the USB dongle used at the receiver site to receive data. The table below show the function of each interfacing pin of SKXBee:

Table 3.2: Interfacing pin of SKXBee

Label	Definition	Function
5V	Power Input for SKXBee	External power source for SKXBee, the typical voltage is 5V. On board 3.3V voltage regulator will regulate the voltage to 3.3V for XBee module. The power is not necessary if SKXBee is connected through USB cable.
GND	Ground or negative	Ground of power and signal.

XB_RX	XBee UART Receive signal	This is XBee module's receiver pin, it should be interfaced to 5V logic UART, no divider is necessary. This is an input pin to SKXBee. It should be connected to microcontroller's transmitter pin.
XB_TX	XBee UART Transmit signal	This is XBee module's transmitter pin; it should be interfaced to 5V logic UART. This is an output pin from SKXBee. It should be connected to microcontroller's receiver pin.
RESET	XBee Reset pin	Reset pin of XBee module. It should be connected to a push button to Gnd, or NPN transistor.

3.7 MAX232IN

The MAX232IN is a dual driver/receiver that manufactured by Texas Instrumentations includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ± 30 -V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. (Texas Instrumentation Inc., 2004) In this project, the author used MAX232IN to convert the signal of RS232 to TTL, 5V (UART) to allow interfacing with SKXBee for wireless transmission purpose.

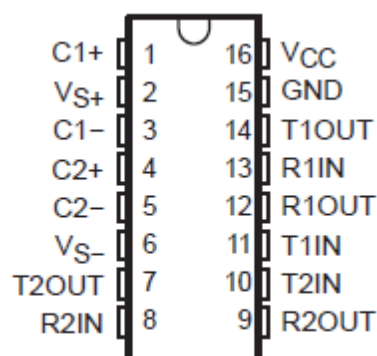


Figure 3.16: MAX232IN

There are two RS232 to TTL convertor in MAX232IN chip. Pin 11 and 10 are input UART or UART Transmit, pin 14 and 7 are output RS232 or RS232 Receive, pin 12 and 9 are output UART or UART Receive, pin 13 and 8 are input RS232 or RS232 Transmit. RS232 signal utilized voltage level from -15V until +15V whereas UART or TTL signal utilized voltage level from 0V until 5V.

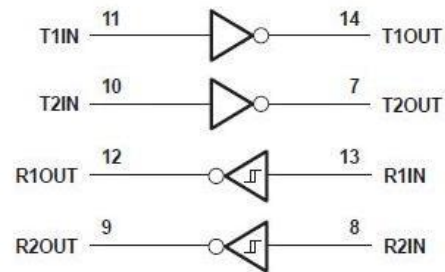


Figure 3.17: RS232 to TTL convertor in MAX232IN.

3.8 Software

3.8.1 CodeWarrior Development Studio for Microcontrollers 6.3

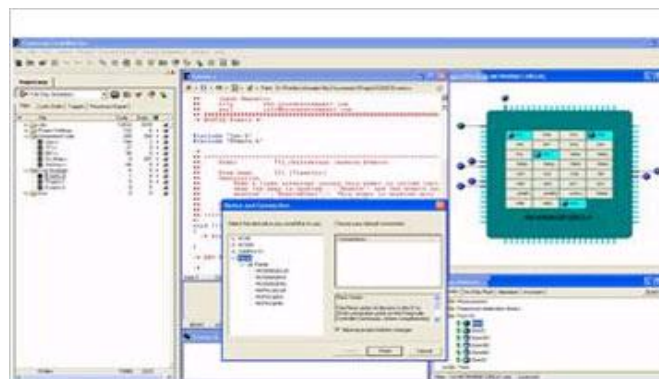


Figure 3.18: Code Warriors 6.3

Freescle's CodeWarrior Development Studio for Microcontrollers v6.3 is a single, integrated tool suite designed to fast track with RS08, HC(S)08 and ColdFire V1 members of the Freescle Controller Continuum. CodeWarrior provides optimized tools to take full advantage of the Freescle microcontroller. (Freescle Inc.,2010) In

this project, CodeWarrior Development Studio for Microcontrollers 6.3 is used to program the MCF51MM microcontroller in order to carry out the ECG signal capturing function and transmit the data. To allow burning program into the microcontroller, MCF51MM, the TWR-MCF51MM board must be connected with the computer with USB via on port USB port and run under OSBDM mode (two additional LED will light on indicate activation of OSBDM mode). The programming will be compiled and debug using Code Warrior before burning into microcontroller.

3.8.2 Digi Maxstream's X-CTU

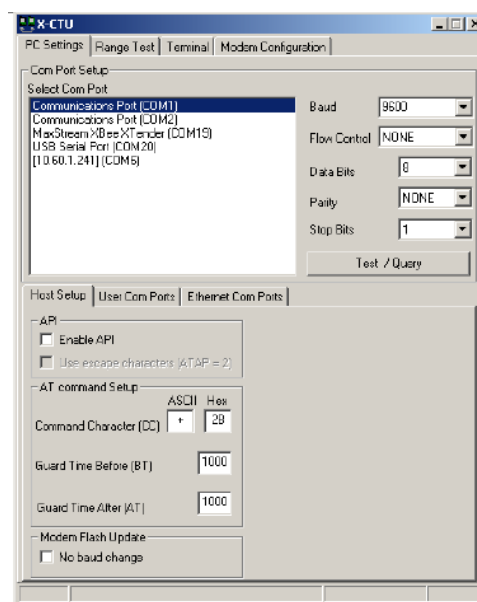


Figure 3.19: X_CTU program interface

X-CTU is a Windows-based application provided by Digi. This program was designed to interact with the firmware files found on Digi's RF products and to provide a simple-to-use graphical user interface to them. (Digi Technologies, 2008) X-CTU is important to configure the parameter such as baud rate, stop bits, parity bits, parity, destination address, etc which is very important to secure communication between receiver and transmitter site of SKXBee. This program used to program XBee module and serve as monitoring software at receiver site to monitor the data

received. After launched, user will see four tabs across the top of the program. Each of this tab has a different function. The four tabs are:

- **PC Settings:** Allows users to select the desired COM port and configure that port to fit the radios settings.
- **Range Test:** Allows users to perform a range test between two radios.
- **Terminal:** Allows access to the computers COM port with a terminal emulation program. This tab also allows the ability to access the radios' firmware using AT commands. In this project, we using this terminal tab to monitor the data received by the SKXBee.
- **Modem Configuration:** Allows the ability to program the radios' firmware settings via a graphical user interface. This tab also allows customers the ability to change firmware versions.

3.8.3 MED-ECG GUI

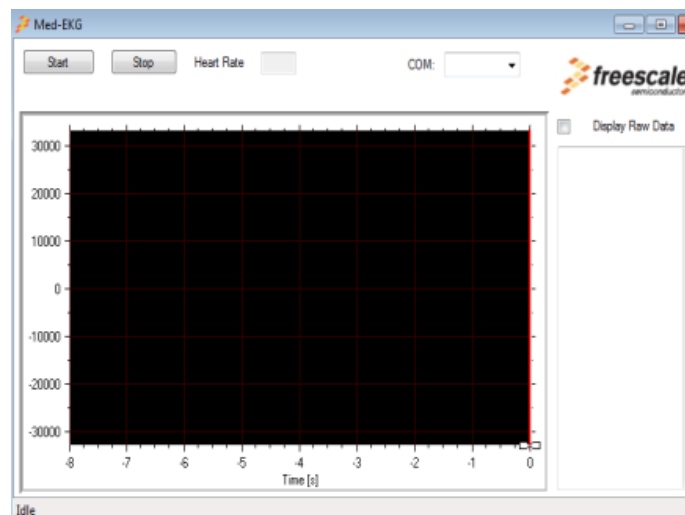


Figure 3.20: MED-ECG GUI

MED-ECG GUI is an user friendly platform that developed by Freescale Inc. to display the ECG signal captured using the Freescale's medical development kit. In this program, there are start and on button. Before install this program to the computer, Microsoft Framework 2.0 (up to Window XP) and Microsoft Framework 3.5 (up to Window 7) must be installed or the MED-ECG will malfunction. User can

display the raw ECG data by clicking the “display raw data” column. To adjust the range of time display such as 1 second, 0.5 second, etc, adjust the x axis correspondingly by pulling it to the right. After connecting the medical development kit to computer, user must choose correct serial com port to correctly output the ECG data. After the development kit connects to computer, it will be recognized as virtual com port. The choice of com port can selected at the top right site of the program. Heart rate of subject will be estimate by GUI by calculating QRS complex peak. At the bottom left site, there is a status bar which indicates the current status of MED-ECG for user reference.

3.8.4 Eltima’s RS232 Data Logger

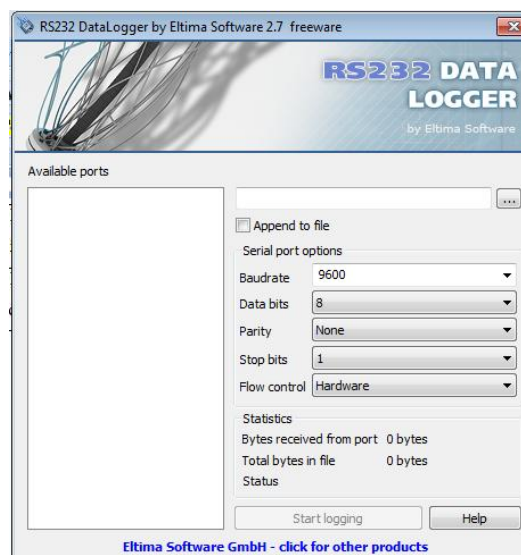


Figure 3.21: Eltima’s RS232 Data Logger

Eltima’s RS232 Data Logger is the software that able to capture the data received at specified com port and saved data in .txt file for further review. To ensure data captured at correct com port, set the compatibility settings such as baud rate, stop bits, parity, data bits, etc and number of com port correctly before data transmission start. Do not open X-CTU to monitor the com port and RS232 data logger to save the data for same com port simultaneously as there is only one action allow for one com port at one time.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Placement of ECG Electrodes

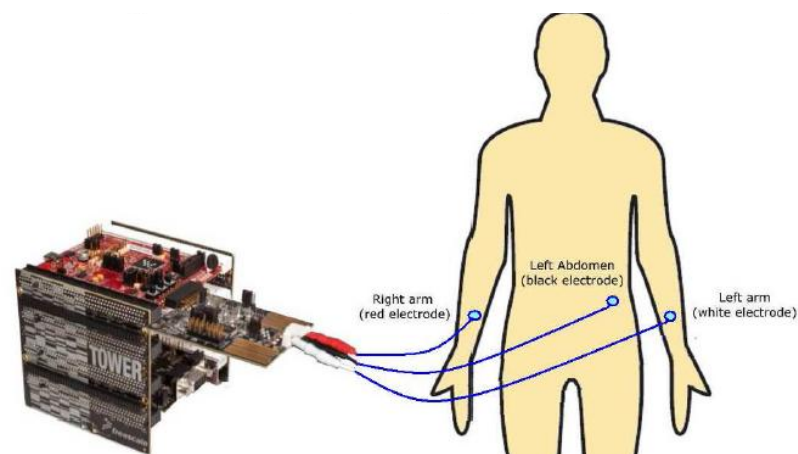


Figure 4.1: External Electrodes Connection

In this project, the author used the external electrodes connector which is labelled as J12 on the MED-EKG module. This connection yields less noise and provides the best way to prototype EKG application instead using the on-board sliders contact. The author connects a 3 lead cable (2 electrodes and 1 ground) via J12 header and use external electrodes. Red electrode will be attached to right arm, white electrode to left arm and lastly black electrode to left abdominal in this project. Welch Allyn ECG Lead Wires for Atlas Monitor, 3-lead AHA will be used with silver chloride ECG electrode to capture the ECG signal and transfer the data to Freescale's development kit for further data processing.

Placing one tab on each arm and left abdominal, results in three unipolar leads, called the “augmented limb leads.” These three leads are referred to as aVR (right arm), aVL (left arm) and aVF (left leg or left abdominal). They record a change in electric potential in the frontal plane. Although the electrodes are placed on each arm and left abdominal, they measure the electrical activity between the heart and shoulders and heart and left abdominal. Different positioning of the ECG electrodes will result in different ECG signal patterns due to different cardiac axes measured. (Francis et. Al., 2003)

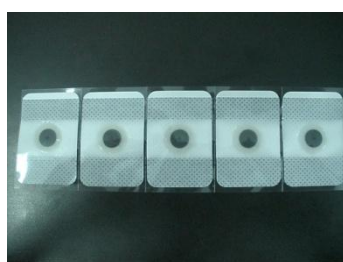


Figure 4.2: Silver Chloride ECG electrode

The same three leads that form the augmented limb leads also form the standard leads, usually designated as I, II, and III. (Francis et. al., 2003) They are all bipolar (they detect a change in electric potential between 2 points) and detect an electrical potential change in the frontal plane:

- 1) Lead I is between the right arm and left electrodes, the left arm being positive.
- 2) Lead II is between the right arm and left abdominal electrodes, the left abdominal being positive.
- 3) Lead III is between the left arm and left abdominal electrodes, the left abdominal again being positive.

4.2 Programming of Freescale’s Medical Development Kit

In this project, programming the MCF51MM is the great challenge to make everything in this project work. Below show the overall structure of the programming of Freescale’s medical development kit. The TWR-MCF51MM is

program in such a way that initiated by MED-ECG GUI from the computer by clicking start button. Take note that if the GUI didn't initiate the data capturing event, MED-ECG board will not capture any data. The MED-ECG board then capture the signal and transfer the raw data to the TWR-MCF51MM board via the medical connector header (20 pins). In the TWR-MCF51MM board, the raw ECG data will process on MCF51MM256 on-chip analog modules (TRIAMPs, OPAMPs, DAC, and ADC) along with noise filters to condition the ECG signal. The ECG signal condition includes the following stages:

- 1) Instrumentation amplification using on-chip TRIAMP1, TRIAMP2 and OPAMP1
- 2) Band pass filtering
- 3) Amplification using on-chip OPAMP2 programmable gain
- 4) Notch filtering
- 5) Amplification using an off-chip OPAMP
- 6) Automatic baseline compensation using on-chip 12-bit DAC
- 7) Digital filtering with the pre-programmed MC56F8006 Freescale DSC

After digital filtering tuning, the end results are sent from DSC via I²C protocol back to MCF51MM256. The MCF51MM256 then uses the Freescale USB stack to forward the results to the computer via peripheral module, TWR-SER. (Freescale Inc, 2010) The tower system, TWR-ELEV play an important role here to connect the TWR-MCF51MM and TWR-SER together and allow data transmission without cabling between these two board. They are then displayed on the Freescale MED-EKG graphical user interface (GUI). There is another data output at the TWR-SER which is via serial communication, RS232. The serial port will output the data that is exactly the same with the Freescale's USB port. This signal output from serial port of TWR-SER is essential to be utilised in the wireless transmission via SKXBee.

In this project, the author primarily program the TWR-MCF51MM to sequence the capturing ECG signal from MED-ECG and process the raw ECG data obtained on the TWR-MCF51MM. Then the process ECG data send back to the computer's GUI via USB cable. The author adds another output of ECG raw data via RS232 or DB9 on the TWR-SER board using SCI (Serial Communication Interface). This is to utilise the RS232 signal to interface the wireless module by converting the

signal to TTL/CMOS (5V) using MAX232 circuitry. Refer to appendices for programming file and source code of the project.

4.2.1 Programming Flow

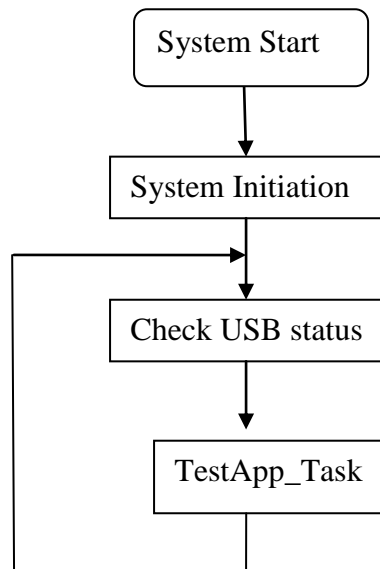


Figure 4.3: Flow Chart of Main Function (main.c)

At first, initializes the system. The ADC, TPM, GPIO,MCU, MCG, KBI, RTC, and other modules are set to system default. The USB is enabled in this process. Then the main function enters into an infinite loop. The Check_USB_Status function controls the transfer of the USB states between attached and suspend. The TestApp_Task is called in the main loop after the device succeeds to enumerate. TestApp_Task is the event of ECG signal capturing and processing. For Check USB status, it initializes the USB BDT according to the USB RAM assignment, then initializes the USB registers, and sets the USB state to ATTACHED at last. To set the USB registers, it configures the USB module (pullup resistor, USB regulator in terms of the hardware design) first, then enables the EP0 and the USB interrupt. For self-powered devices, the USB module can be enabled when MCU detects the USB bus power, and the USB device initial state is POWERED.

After the USB is enumerated and configured successfully, a communication pipe is created between the USB host and the device. All commands and data are transferred into the pipes. There are five pipes are built in the data logger system such as default pipe (control pipe), command pipe, status pipe, data out pipe and data in pipe. The command pipe is used by the host to send the commands to USB device. It is built on endpoint 1 of the USB device. The status pipe used to sends the response to the host via the status pipe after it receives and executes the command. Data out pipe is used by the host to transfer data to the device. Data in pipe sends the data to the host via the data in pipe. (Freescale Inc., 2008)

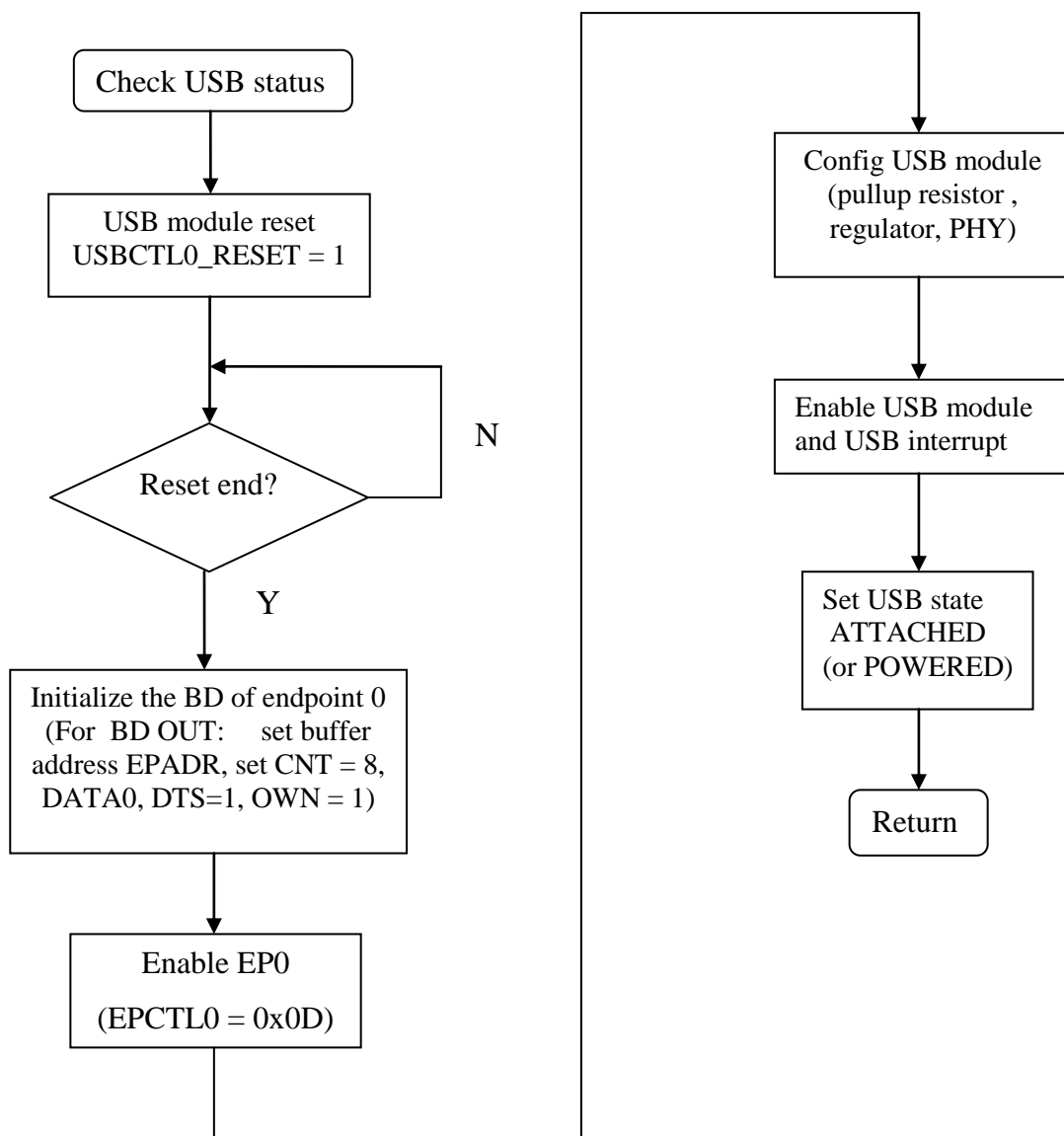


Figure 4.4: Flow of Check USB status

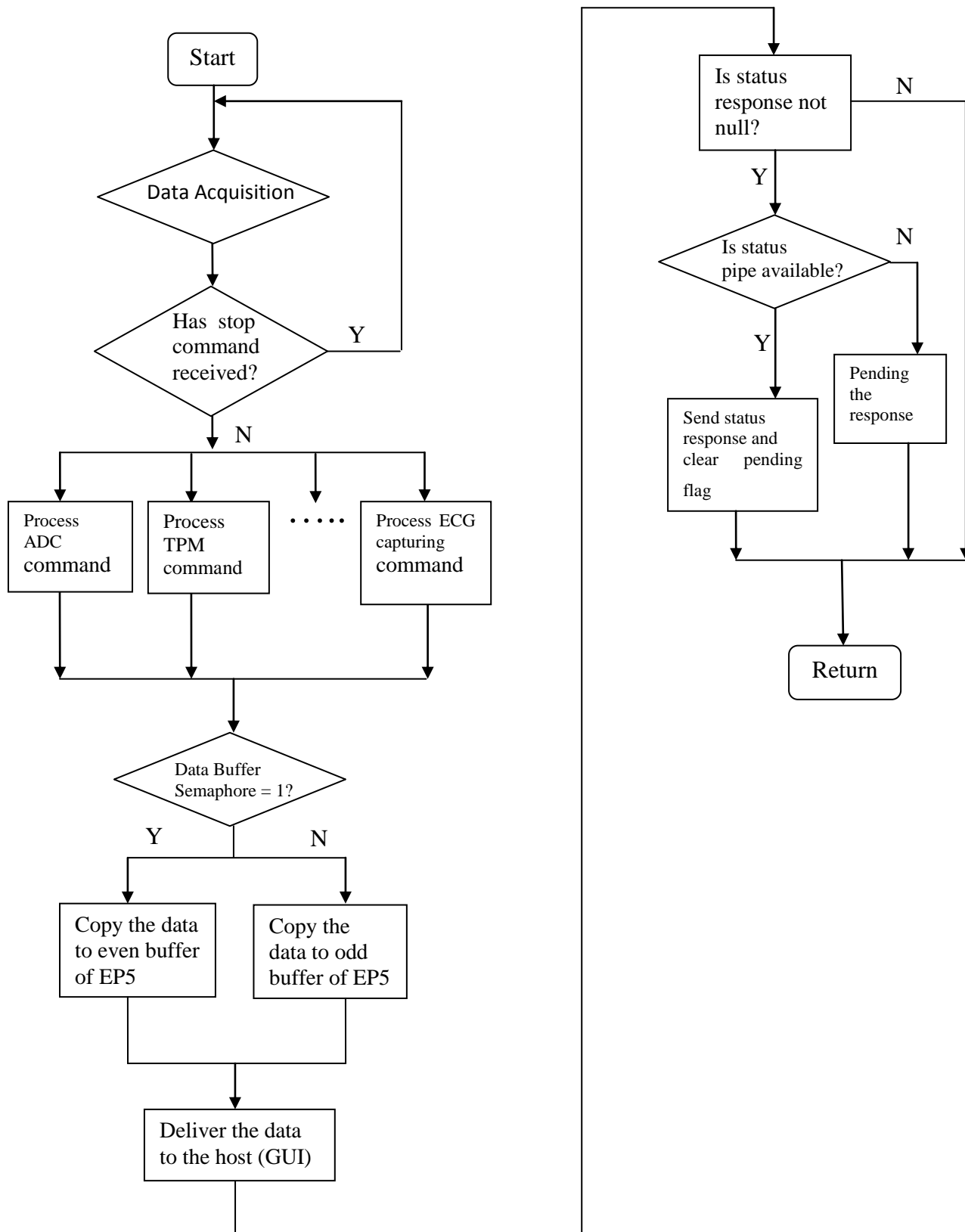


Figure 4.5: Flow of TestApp_Task

At first, the TestApp_Task checks whether the data acquisition has started. The data acquisition only initiated by clicking start on the MED-ECG GUI, give true value to ADC_CHANNEL_FEEDBACK_SIGNAL, ADC_CHANNEL_ECG_SIGNAL (via EP1) to allow ECG data acquisition. The acquired data will undergo a series of processing such as ADC (Analog to Digital Conversion), DSC (Digital Signal Conditioning), amplification (by op-amp), filtering processes, etc (Freescale Inc., 2010). While the data acquisition is working according to the status of the ADC conversion, it copies the data to the endpoint buffer of endpoint5. Semaphore is a protected variable or abstract data type that provides a simple but useful abstraction for controlling access by multiple processes to a common resource in a parallel programming environment. It is useful tool in the prevention of race conditions and deadlocks in the continuous ECG data acquisition processes. The data then delivered to the GUI to display via EP5.

Afterwards the program begins to check whether there is response data needed that needs to be transferred to the host via status pipe. If there is a response data, it is delivered. If there is no response, the program suspends and then exits from the function (back to acquisition data until user stop or no response from host). If the program (MED-ECG GUI) shut down during data acquisition, the control pipe disable and EP0 deactivate. This will results ECG data's acquisition stop instantly. There is another output of data (RS232) in this project. It is executing before the semaphore function by adding a SCI commands and send it via I²C protocol.

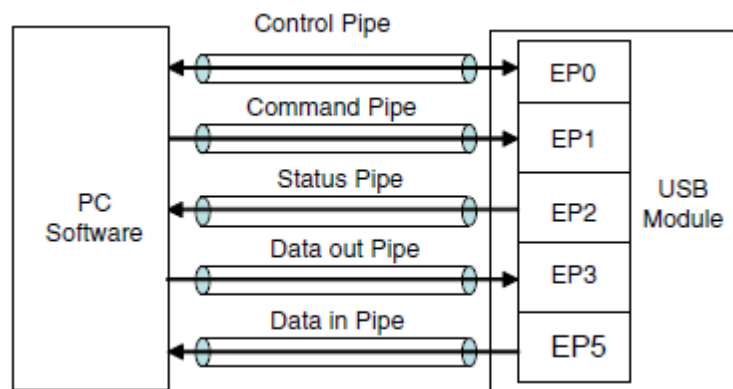


Figure 4.6: Communication pipe between GUI and Prototype

4.3 Jumper Settings on Freescale's Medical Development Kit

Settings of jumpers on Freescale's Medical Development Kit are essential step to ensure proper working of the programming code and avoid errors such as additional gain or voltage added to the data. Refer to appendices for default settings of the Freescale's medical development kit. Generally the jumpers are set to output the data via RS232, printing the ECG data to the GUI via USB port, correct op-amp gain settings, etc. Majority of the jumpers settings set at defaults but the author make some changes of certain jumper settings on the kit as following table on ensure proper functioning:

Table 4.1: Modification of Jumper Settings

Board	Modifications and Reasons
TWR-SER	J16 on TWR-SER has pins 3 and 4 connected instead of pins 1 and 2. This selects the USB port to support device class transmission which is required for printing the EKG signal to the MED-EKG GUI.
TWR-MCF51MM	<p>Remove the jumpers that connect pins 1 and 2, 9 and 10, 11 and 12, and 13 and 14 from header J18.</p> <p>Shunt 1-2 connects PTE7 with LED4, but PTE7 also enables the Power Supply of Medical Connector. If this jumper is placed, a value of about 1.2 V is present on pin PTE7 and the power-up sequence of DSC may fault. If you are not using the DSC, it is not important.</p> <p>Shunts 9-10, 11-12 and 13-14 must be disconnected in any case, because they connects the on-board accelerometer outputs to ADC inputs, and they are connected to the same channels used for ECG signals. If they are not removed, the signal will include the added noise of the accelerometer signals.</p>

4.4 MAX232 Circuitry

In this project, the author constructed MAX232 circuitry to convert the signal of RS232 to TTL, 5V to allow interface of data with SKXBee on bread board. The MAX232 chip that the author used was MAX232IN manufactured by Texas Instruments. As explained earlier, SKXBee can only be interface with TTL/CMOS signal which voltage level is 5V. Thus, the author has to construct the MAX232 circuit on the breadboard to change the RS232 signal to TTL (5V). The pins 2 (Transmit Data) and pin 3 (Receive Data) of Serial Port from TWR-SER (Freescale's module) connected to pin 14 and pin 13 of MAX232 respectively. Pins 11 (UART Transmit) and pin 12 (UART Receive) of MAX232 connected to Tx pin and Rx pin of SKXBee respectively. Pins 5 of the DB9 connected to the ground as reference voltage. Pins 7 and 8 of the DB9 short together to allow continuous data cleared to send to MAX232 circuitry.

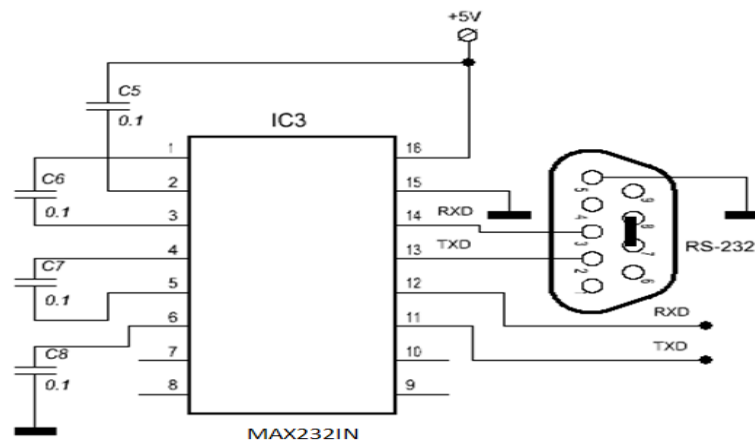


Figure 4.7: Circuit diagram of MAX232 circuitry

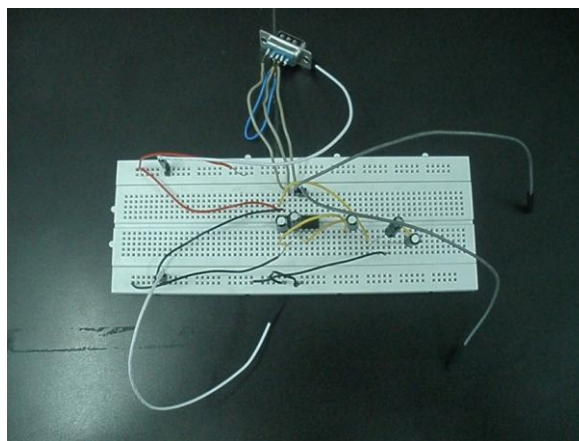


Figure 4.8: MAX232 circuitry on breadboard

4.5 Wireless Transmissions using XBee

XBee module (mount on SKXBee) served as the most important component in wireless transmission of project. The XBee modules interface to a host device through a logic-level asynchronous serial port. There are two type of operation for XBee such as transparent operation and API operation. By default, XBee Modules operate in Transparent Mode. When operating in this mode, the modules act as a serial line replacement - all UART data received through the DI pin is queued up for RF transmission. (Digi Technologies, 2008) When RF data is received, the data is sent out the DO pin. In this project, the author implements the transparent operation to transmit the ECG data captured using Freescale's development kit.

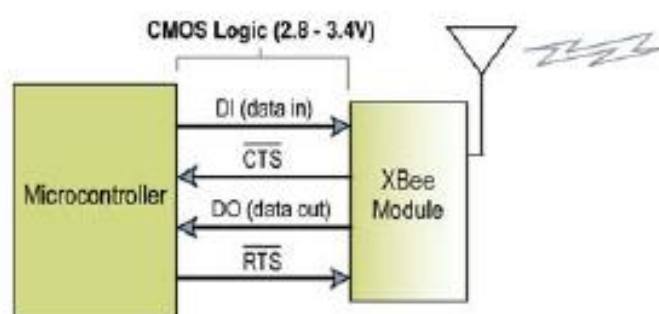


Figure 4.9: Data Flow from microcontroller to XBee module

As XBee module was solder on the SKXBee board, the author only concentrate on interfacing the pins on SKXBee (5V, ground, TX, RX , RESET) rather than on XBee module itself in this project. The RS232 output from Freescale's module will converted to 5V TTL/CMOS signal via MAX232 circuitry. The data then flows in to SKXBee via RX pin of the SKXBee. The flow of data occurred in such a way from SKXBee board to the XBee module: ECG data enters the module UART through the DI pin (pin 3) on XBee module as an asynchronous serial signal. The signal should idle high when no data is being transmitted. Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). In this project, 9600 baud rate, none parity, 1 stop bit and 8 bit used for XBee settings. The module UART performs tasks, such as timing and parity checking, that

are needed for data communications. (Digi Technologies, 2008) Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits). To ensure the communication between two XBee module, the author has to set both XBee module to have same compatible settings (9600 baud rate, none parity, 1 stop bit and 8 bits). The following figure illustrates the serial bit pattern of data passing through the module:

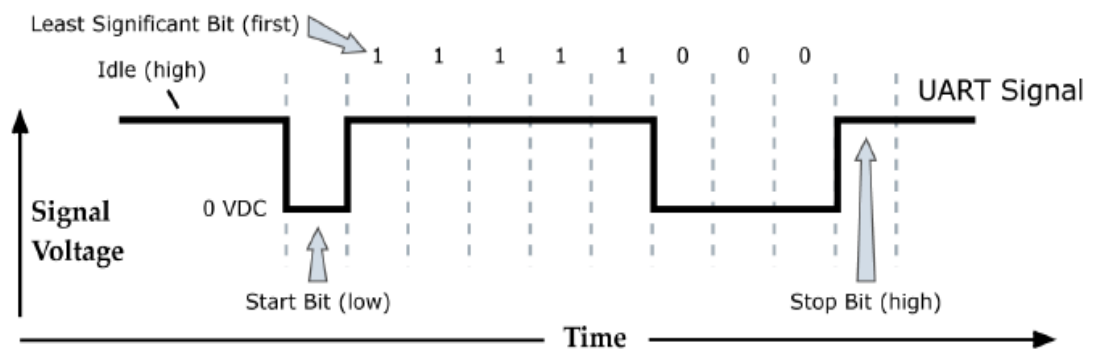


Figure 4.10: Example Data Format of XBee (8 bits, none parity, 1 stop bits)

Other than the stated compatibility settings, to ensure continuous transfer of data between both XBee module, the author has to set the correct address for both XBee module. Every XBee data packet sent over-the-air contains a Source Address and Destination Address field in its header. The RF module conforms to the 802.15.4 specification and supports both short 16-bit addresses and long 64-bit addresses. 16-bit addressing is practice in this project. To send a packet to a specific module using 16-bit addressing: Set DL (Destination Address Low) parameter to equal the MY parameter and set the DH (Destination Address High) parameter to '0'. The XBee module can be configured to use short 16-bit addresses as the Source Address by setting (MY < 0xFFFFE). (Digi Technologies, 2008) Setting the DH parameter (DH = 0) will configure the Destination Address to be a short 16-bit address (if DL < 0xFFFFE). For two modules to communicate using short addressing, the Destination Address of the transmitter module must match the MY parameter of the receiver. By default, the RF module operates in Unicast Mode. Unicast Mode is the only mode that supports retries. While in this mode, receiving modules send an ACK

(acknowledgement) of RF packet reception to the transmitter. If the transmitting module does not receive the ACK, it will re-send the packet up to three times or until the ACK is received. Table below show the address set in this project for both XBee module using X-CTU.

Table 4.2: Address for both SKXBee

Parameter	XBee (Receiver)	XBee (Transmitter)
MY (Source Address)	0x01	0x02
DH (Destination Address High)	0	0
DL (Destination Address Low)	0x02	0x01

4.5.1 Flow of Data in Transmitter and Receiver

The XBee at receiver site should receive ECG data continuous once the prototype system initiated. The receiver must recognise the beginning of the start bit, and then sample the line halfway through each of the 10 bit periods. The receiver clock runs at 16 times the baud rate, and the receiver can sample the input line once per period of this clock. (John Foster, 2001) The sequences of operations are as following:

- 1) The start bit begins.
- 2) At its next clock cycle, the receiver detects that the start bit has begun. This may be up to 1/16 of a bit period after the actual start.
- 3) After another 8 cycles, the receiver samples the line again. If the line is still at logic 0, the start bit is confirmed. Otherwise the initial transition is dismissed as noise.
- 4) After another 16 cycles the receiver samples the line. This is repeated a further time, to get the values of the eight data bits.
- 5) After another 16 cycles the receiver samples the line again, expecting to see the logic 1 level of the stop bit. If it doesn't see a logic 1 at that point, it discards the data and reports a framing error.
- 6) After one more cycle the receiver starts sampling the line at every cycle, waiting for the next start bit.

When ECG data is sent by XBee in transparent mode at transmitter site, all bytes are first converted to ASCII. (John Foster, 2001) Then the values are reported, each terminated by a carriage return. Refer to appendix for pin out of XBee module. The order is:

- 1) The channel indicator with a bit set for each enabled input or output. This is a 16-bit word, with the most significant byte sent first. In the high byte, bit 7 is not used and bits 6 to 1 are set if the data will include samples from ADC5 to ADC0 respectively. Bit 0 of the high byte is set if the data will include the state of line D8. In the low byte, bits 7 to 0 correspond to lines DIO7 to DIO0.
- 2) The DIO data (line state) for each enabled digital input or output, if there are any enabled digital inputs or outputs. This is a 16-bit quantity, with the D8 state in bit 0 of the msb and bits 7–0 in the lsb.
- 3) The ADC data for each enabled analogue input, in ascending order of line number. Each is an unsigned 16-bit quantity, with the 10-bit value in the low-order 10 bits.
- 4) The data is terminated by an extra carriage return.

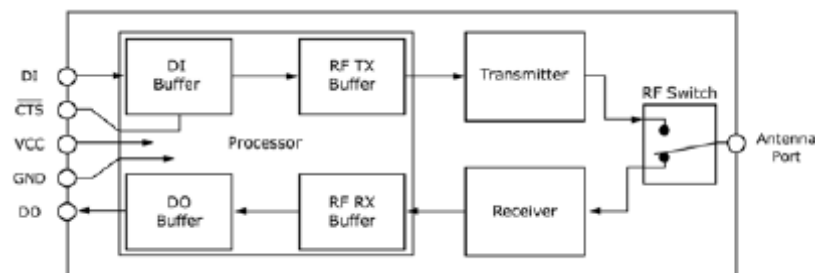


Figure 4.11: Internal Data Flow Diagram

For transmitter site, when the ECG data enters the XBee module through the DI pin (pin 3), the data is stored in the DI Buffer until it can be processed. The ECG data will be sent via transmitter. For receiver site, when ECG data is received, the data enters the DO buffer and is sent out the serial port to a host device (computer). Once the DO Buffer reaches capacity, any additional incoming RF data is lost. (Digi Technologies, 2008)

4.6 Prototype Development

In the end of this project, the author was able to develop prototype systems which comprise of receiver and transmitter site. The receiver site receives ECG data transmitted from transmitter site and monitored by X-CTU. The received data then saved by the Eltima's RS232data logger software. In the transmitter site, ECG data display in MED-ECG GUI in the computer. However the data generated by MED-ECG will not be saved automatically. To redisplay the ECG data generated in a session, user has to check "display raw data" column before initiates the ECG prototype. The MED-ECG GUI will then display data in that specified column. To redisplay the ECG results, copy the generated ECG data at MED-ECG and paste it in Microsoft Excel to construct an ECG graph. Lighting of LED on prototype board after power supplied indicates working condition. If there was no LED lighting after power supplied, it may caused by incorrect assemble of the tower system.

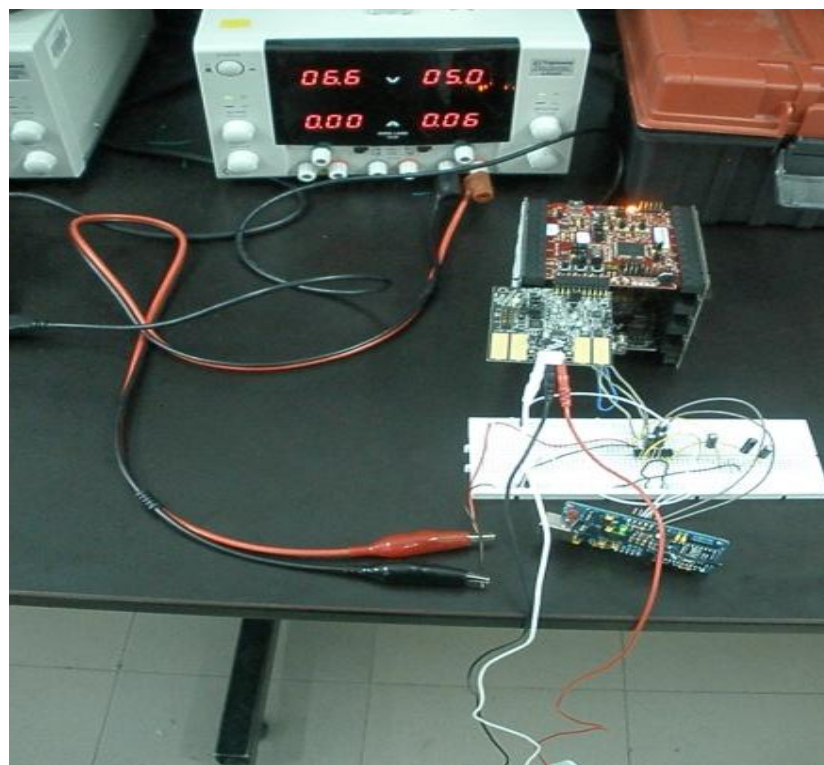


Figure 4.12: ECG prototype (Transmitter site)

As seen in the figure for transmitter site, DC power supply (red and black cable) connected to the MAX232 circuitry for 5V. The power of the Freescale's medical development kit will be supplied by computer's USB port (black USB cable connecting the TWR-SER board). 3 ECG cable such as white, black and red will be connected to left hand, left abdominal and right hand respectively from MED-ECG connector. The Freescale's medical development kit constructed with the TWR-MCF51MM board in the top slot of TWR-ELEV whereas TWR-SER will be connected at the lowest slot of the TWR-ELEV. Primary slot of the boards has to connect to the functional block of TWR-ELEV whereas the secondary slot has to connect to the dummy block. Incorrect connecting the board to TWR-ELEV will result in malfunction of system. MED-ECG connected to TWR-MCF51MM board via medical connector (20 pins). SKXBee connected with MAX232 circuitry via its 5V to source, ground to MAX232 circuitry ground, Rx and Tx pins to MAX232IN's pins 12 and 11 respectively. The ECG capturing event initiated by MED-ECG GUI.

The receiver site of the prototype system comprise of SKXBee and computer as figure shown. The SKXBee will receive data from transmitter site and display in X-CTU under terminal tab or saved using RS232 data logger. To ensure secure transfer of data, set the compatibility settings and address of both XBee module correctly as discussed in earlier section.



Figure 4.13: Prototype system (Receiver site)

The prototype can be reprogram with modified source code using Code Warrior 6.3. Compare to other microcontroller board, this prototype is more convenient in term of burning new source code into the microcontroller, MCF51MM. To do this, connect a mini USB to the TWR-SER board first to supply power to the prototype system. Then connect a mini USB cable connecting the computer with the TWR-MCF51MM board (for reprogram purpose, normally not necessary). Connect the USB cable for TWR-SER first, never connect the USB cable to TWR-MCF51MM before TWR-SER or it will posses damage to the prototype! The USB port in TWR-MCF51MM is serves as reprogramming microcontroller purpose. After connected the USB, the CW 6.3 can be initiate to begin the debug and burn tasks for microcontroller on board.



Figure 4.14: Reprogram the TWR-MCF51MM

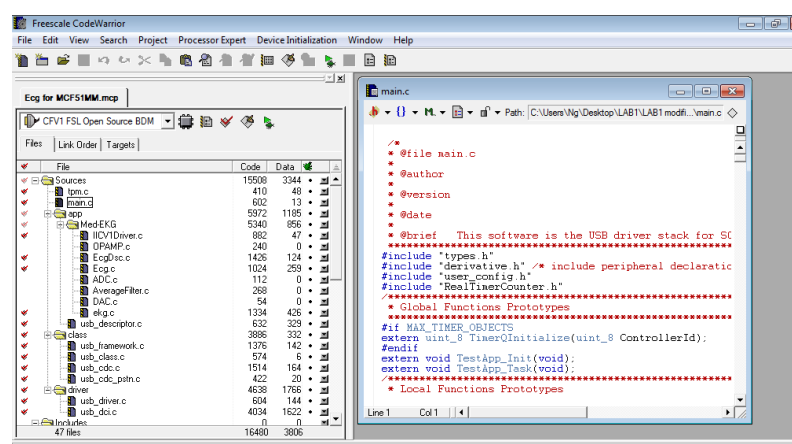


Figure 4.15: Program microcontroller on board using CW 6.3

In this project, if DSC function need to turn off, the only way is to reprogram the microcontroller to deactivate the DSC function. The author reprograms the microcontroller throughout the development phase of the project for trials. The benefit of using this development kit is the flexibility (able to reprogram) compare to other development module which unable to reprogram. When TWR-MCF51MM connects with computer, there are two LEDs light on indicating ready to reprogram as figure 4.14. The captured ECG data undergo a series of data processing through Freescale's medical development kit, MAX232 circuitry, SKXBee and computer as shown in figure 4.15.

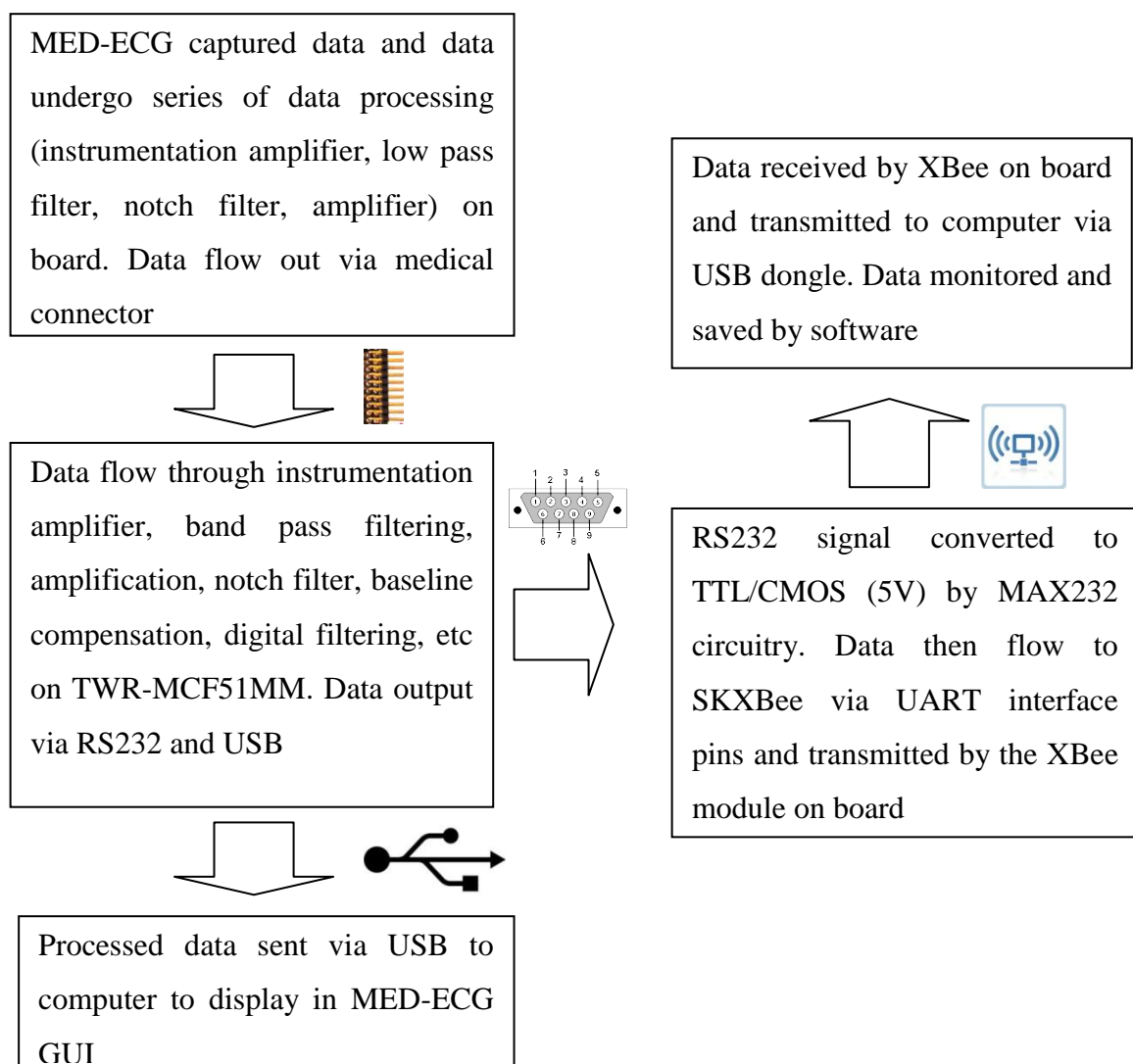


Figure 4.16: Data Flow Diagram

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Results of the Prototype System

5.1.1 Transmitter Site of Prototype System

ECG signal that obtained in the transmitter site will be display in MED-ECG GUI in a real time graph. Please note that the ECG results generated in this project is based on three external leads of ECG electrode placing on the both hands and left abdominal other than the slide electrode on MED-EC. Different placement of ECG electrode will output different ECG data due to different cardiac axis. If the ECG signal generated are not in desired range (too small or big of amplitude), following strategies can be took to obtain better ECG results:

- 1) Changing the op-amp gain of TWR-MCF51MM by press and release black button (SW2, SW4 and SW1) on-board. SW2 is increase op-amp gain, SW4 is decrease op-amp gain and lastly SW1 is reset the op-amp gain. By default, TWR-MCF51MM board has 9x op-amp gain. Each time that the gain is reduced, LED D9 blinks; if the gain is increased, LED D10 blinks. If the maximum or minimal gains are reached, the related LED will remain solid. Increase the gain if amplitude of ECG too small and decrease it when amplitude too big.
- 2) Deactivate the DSC function of the TWR-MCF51MM by modifying the source code. This will disable the DSC function and obtain raw ECG data without amplification.

- 3) Shunt J3 and J4 of the MED-ECG from 2-3 to 1-2 to increase the gain of ECG electrode from 10x to 100x. Please note that this is not recommend strategy as increasing the gain ten folds, it also increase the noise originated from electrode ten folds in the same time.

ECG data display in the following with following settings unless specified by terms:

- 1) 3 lead ECG electrodes.
- 2) Shunt 2-3 at J3 and J4 MED-ECG (10x).
- 3) With DSC function operated.
- 4) Op-amp gain of TWR-MCF51MM at 9x
- 5) ECG electrode place on both arm and left abdominal.

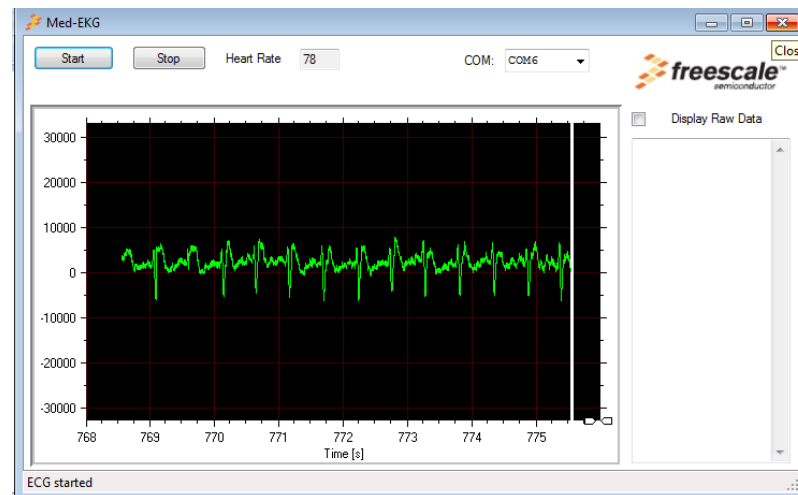


Figure 5.1: Typical ECG waveform obtained with 9x op-amp gain

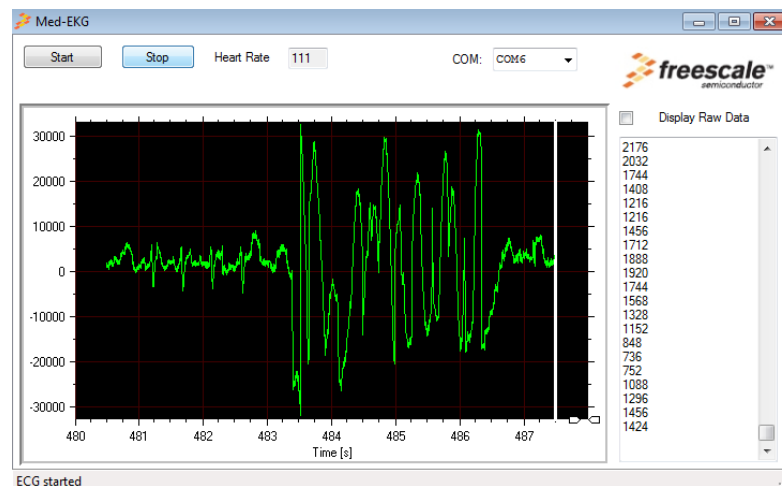


Figure 5.2: Movements when measuring will create noise

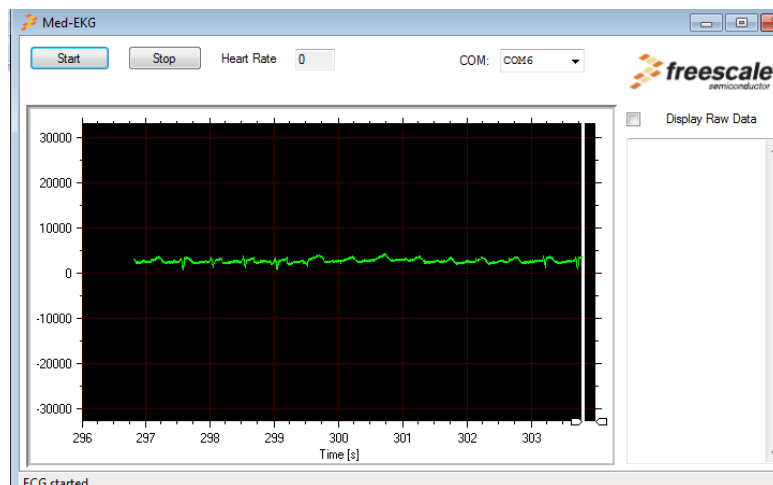


Figure 5.3: Measuring with smallest op-amp gain 2x

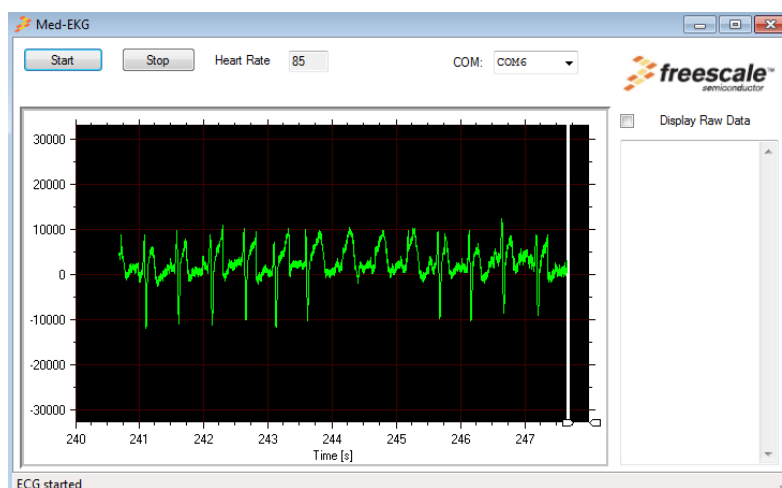


Figure 5.4: Measuring with biggest op-amp gain 17x

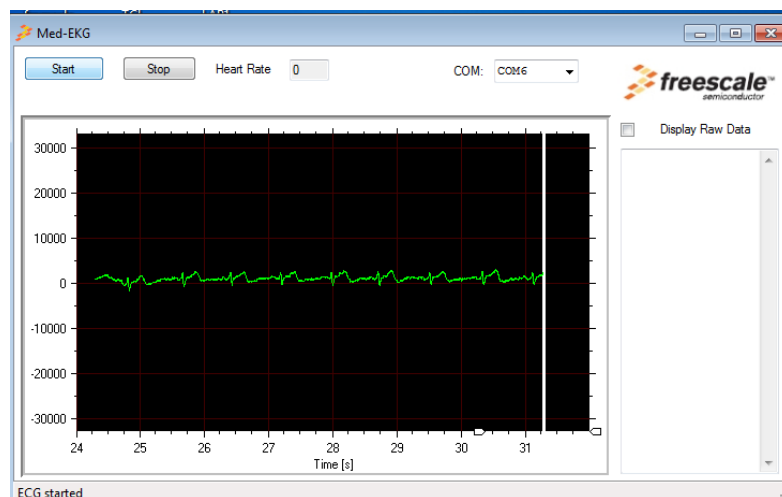


Figure 5.5: Measuring without DSC function

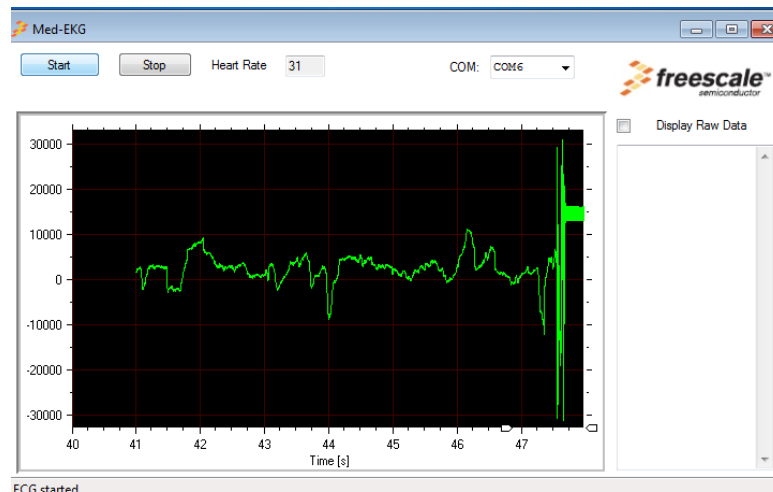


Figure 5.6: Measuring with slide electrode (unsecure, with slight finger movement)

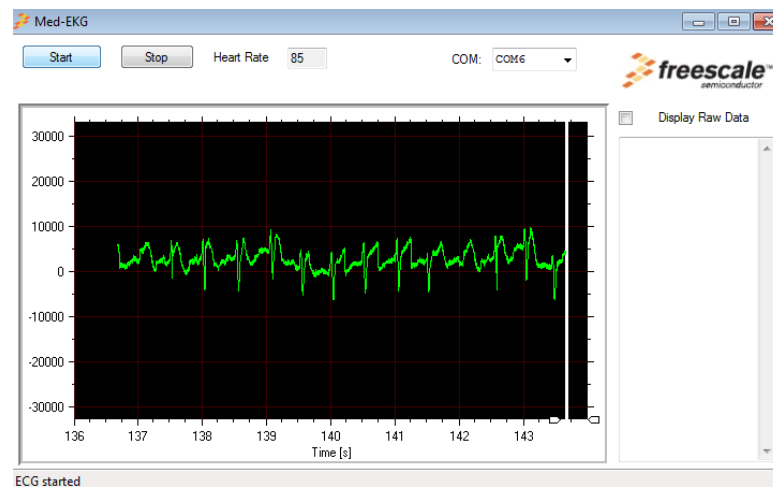


Figure 5.7: Measuring with slide electrode (secure, no finger movement)

5.1.2 Receiver site of Prototype System

At the receiver site, there will be no real time graph of ECG display as transmitter site. The author will show proof of wireless transmission of ECG data between transmitter and receiver of the prototype system. Data received in the receiver site will be monitored by X-CTU program and saved by the RS232 data logger. Only one program allows implementing at one time as one com port cannot perform different tasks at one time. For RS232 data logger, a .txt file will be created after one session

of data capturing. Please refer to the appendices for one of the sample obtained using RS232 data logger in .txt file. As explained in earlier section, the data received is in ASCII form and need convert to hex form for convenient of data analysis.



Figure 5.8: Monitoring data received at receiver site using X-CTU

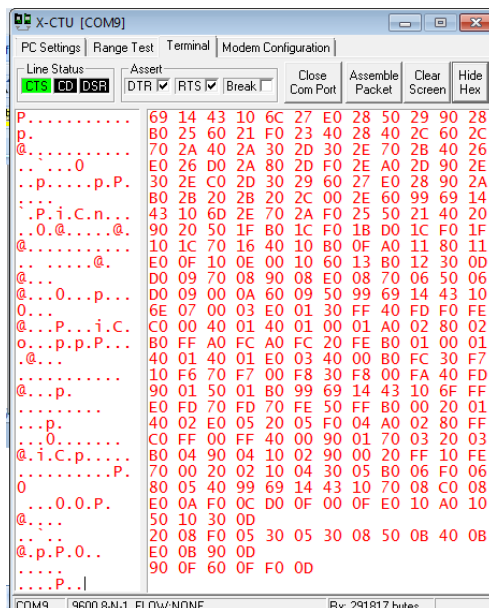


Figure 5.9: Data received convert from ACHII to hex form

5.2 Analysis of Results

5.2.1 ECG waveform at Transmitter Site

The ECG data that obtained by the author is in cyclical pattern. However, it is not the typical ECG graph that the author reviewed in earlier section of this project which has typical P, Q, R, S and T segments. Peak observed in the graph for each beat detected (one beat equal to range between two peaks). Apparently P, Q, R, S and T segments can be detected in the obtained results mixed with some noise. The peak is actually QRS complex, the P wave and T wave is detected before and after the peak respectively which mixed with some noise. The MED-ECG GUI will estimate the heart rate by calculating the number of QRS peak (negative or positive) as one QRS peak will equivalent to one heart beat. After approximately 5 seconds of data sampling, the heart rate will floating amends with the ECG waveform pattern. If there are noises exist caused by movements or power line supply, heart rate will not be accurate as the GUI unable to detect the heart beat correctly. Measuring without DSC function or low op-amp gain also causes heart rate estimation fail as there was no obvious peak or peak for GUI to detect as one heart beat.



Figure 5.10: Analysis of Real Time ECG data

Although the ECG waveform that obtained is not the typical one as shown in the literature review, this didn't mean that the ECG waveform is problematic or abnormal. In practical, it is hard to obtain typical ECG waveform due to noises originating from patient body, ECG circuitry, ECG electrode, power line supply noise, etc. At figure 5.12, power line noise observed in the graph which make the line "thick" in shape. The pattern of ECG is depends on the placements of electrode which measure heart electrical activity from various axis. In figure below, it show some morphologic of the QRS complex. For QRS complex #2, #3 and #4, they are all normal morphologic of QRS complex of different shape. (Tracy, 2003). The morphologic of QRS complex in this project is fall at morphologic #2 and #3 which are normal QRS complex according to Tracy, 2003.

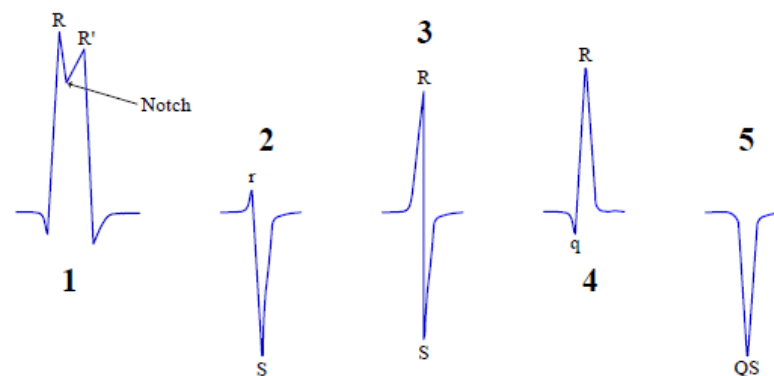


Figure 5.11: Various QRS Complex Morphologic

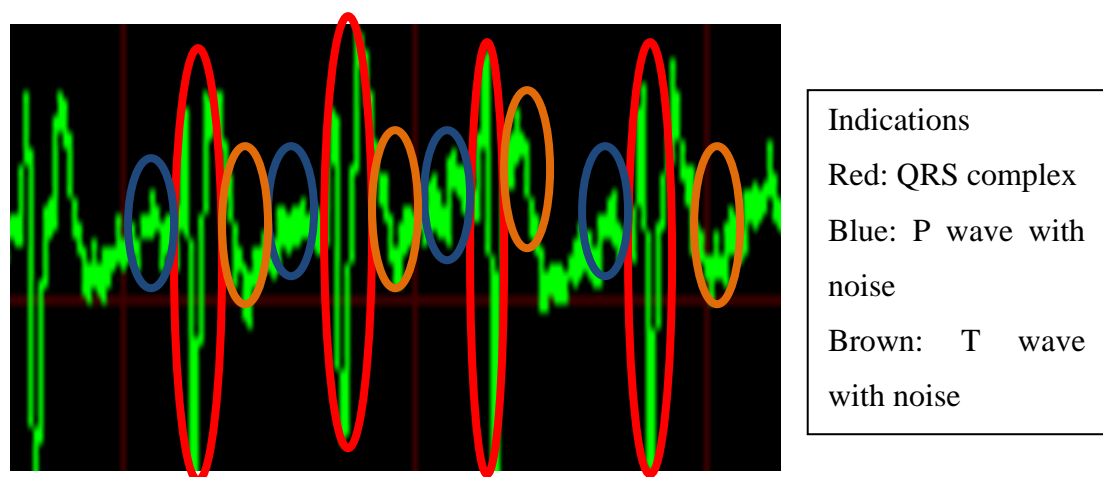


Figure 5.12: ECG waveform results analysis

The patterns of the ECG waveform generated using the Freescale's medical development kit is similar to the clinical ECG waveform. Compared to the clinical waveform (Francis et. al , 2003), the difference of ECG waveform generated with Freescale's medical development kit is the existence of noise which disrupt the P wave and T wave. The noise in the waveform can be cause by various factors and will be discussed in later section.

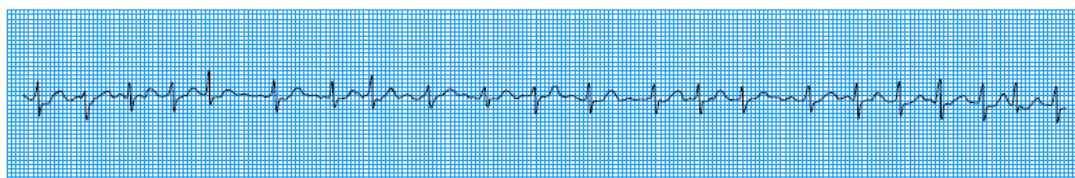


Figure 5.13: A standard rhythm stripe

5.2.2 Data Received at Receiver Site

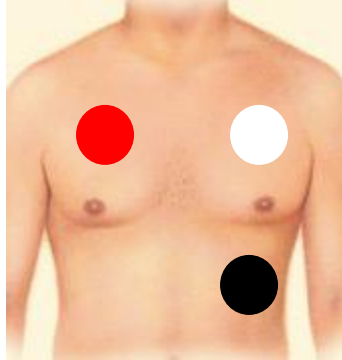
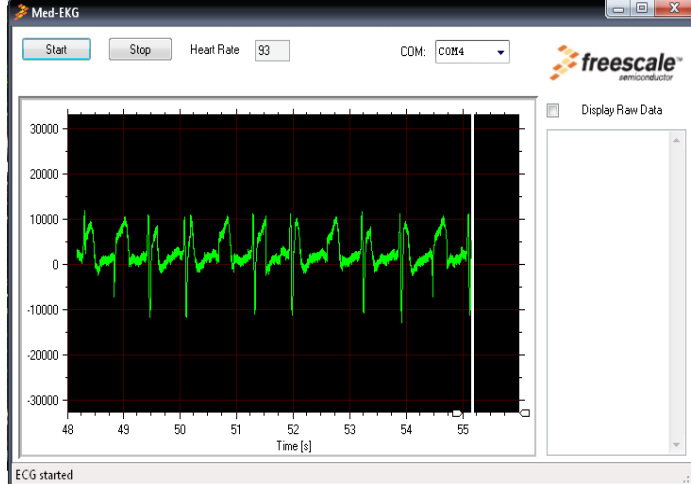
At the receiver site, the data monitored in X-CTU program under terminal tab transmitted at 9600 baud rate. The data keep flow in continuously in ASCII form. To analyze the data, the received data have to convert to hex form. The data saved using RS232 data logger converted from ASCII to hex form using Notepad++. Notepad++ is a free and powerful software which able to convert data from ASCII to hex form. Refer to appendix for the notepad++ screen shots. In the received data, there will be acknowledge of start command which initiated by the GUI. After the acknowledge code, header for ECG data will be the next code observed. The ECG capturing process then run continuously until the stop by the GUI or interrupted (power off). Thus, the codes after the header code are all the ECG data. Implementation of this prototype system need to take account of the data type and the medical server need to be able to convert the ASCII data to hex data or decimal data to allow plotting of graph. Refer to the appendix for .txt file sample saved by RS232 data logger and the analysis received data in hex form.

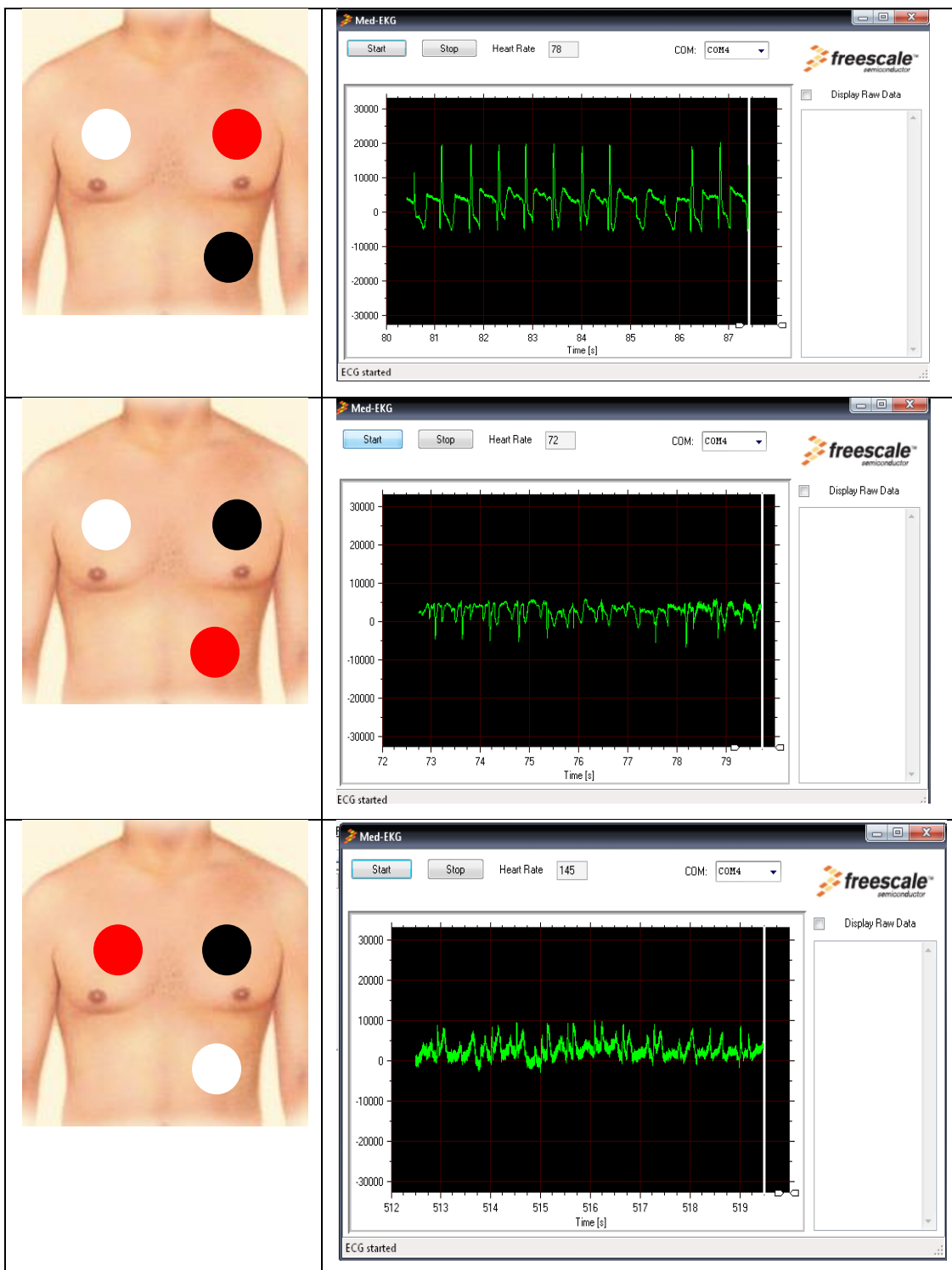
5.3 The Effects of Different Placements of ECG Electrodes

In this project, the author placed the 3 leads electrode to both arm and left abdominal and took the results as the primary results to analyze. To research more on the effects of different placement of electrode, the ECG electrode placed to the chest to obtain the ECG data and the results will be display in this section. The author found out that placing the electrodes on chest generate different pattern of ECG waveform depend on the positioning of electrodes compare to place the electrode on both arm and tend to have lesser noise (especially AC and DC noise). This is due to different placement of ECG electrodes will create different cardiac axis. Note that the black, red and white color circles in the table represent ECG electrodes that attached with the specified area. The ECG results shown in the table following with settings below:

- 1) 3 leads ECG electrodes.
- 2) Shunt 2-3 at J3 and J4 MED-ECG (10x).
- 3) With DSC function operated.
- 4) Op-amp gain of TWR-MCF51MM at 9x
- 5) ECG electrode place on chest (Red, White, Black).

Table 5.1: Summary of Different Placements of ECG Electrode on Chest

Placements of ECG Electrode	ECG Results
	



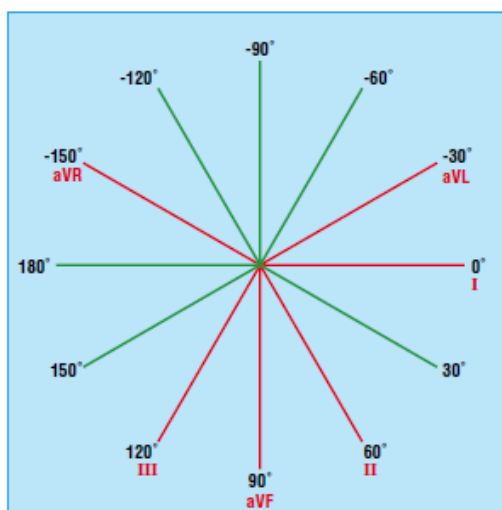
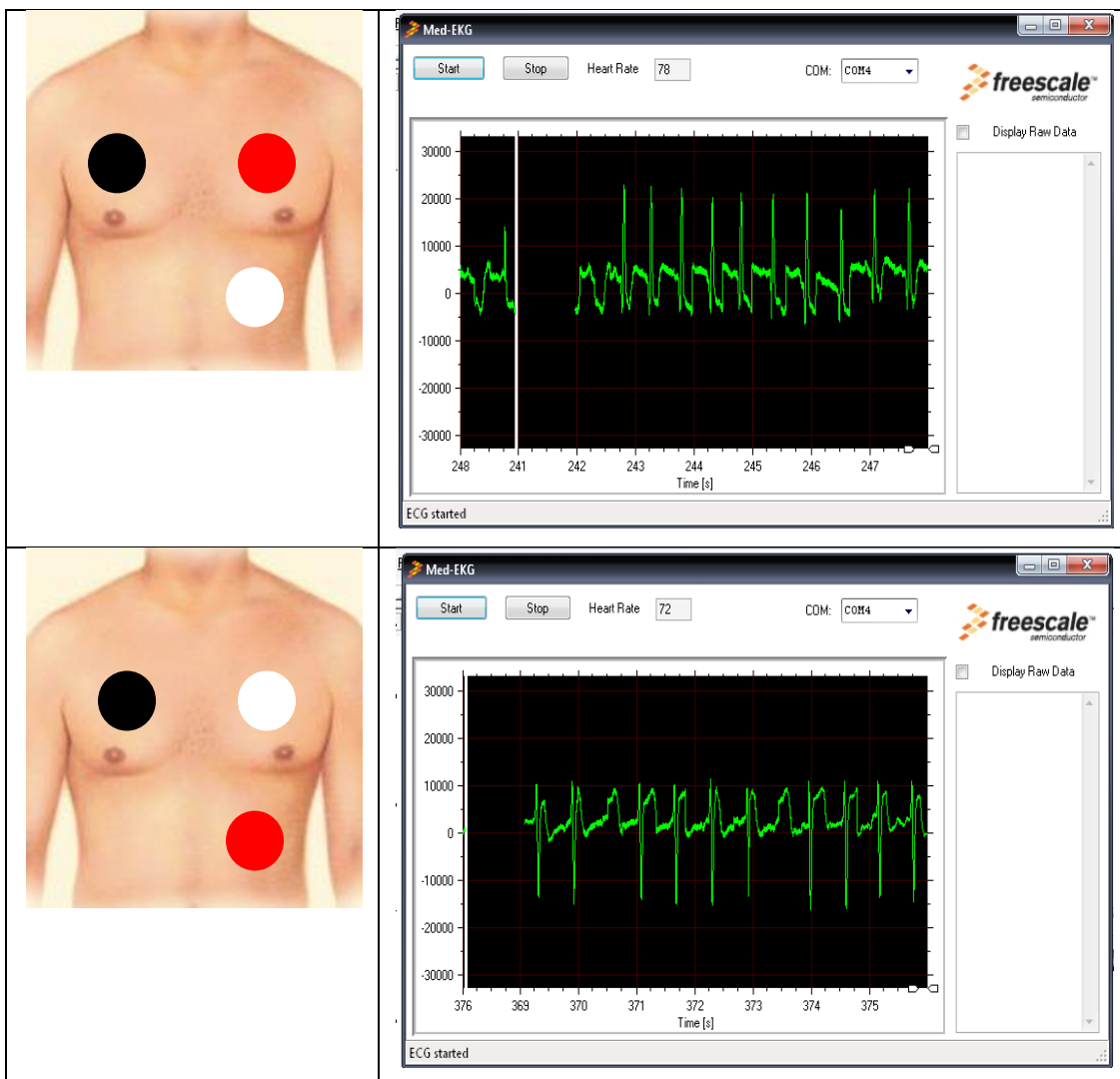


Figure 5.14: Cardiac Axis

It can be observed that different orientation of electrode will generate different pattern of ECG signal due to different cardiac axis especially the QRS complex. The cardiac axis refers to the mean direction of the wave of ventricular depolarisation in the vertical plane, measured from a zero reference point. The zero reference point looks at the heart from the same viewpoint as lead I. An axis lying above this line is given a negative number, and an axis lying below the line is given a positive number.

Theoretically, the cardiac axis may lie anywhere between 180° and -180° . The normal range for the cardiac axis is between -30° and 90° . An axis lying beyond -30° is termed left axis deviation, whereas an axis $> 90^\circ$ is termed right axis deviation. (Francis et. al , 2003) When the orientation of ECG electrodes changes, the QRS complex obviously change in shape corresponding to the new cardiac axis formed. Different orientation of electrodes will give different interelectrode distance which affects the signal strength and determine whether can output sufficiently reliable and strong ECG signal. (Puurtinen, Hyttinen & Malmivuo, 2005) The P wave and T wave will change according to the orientation of electrodes too but the changes of shape not obviously observed. The ECG waveform which is closest to the typical one will be white electrode at the right chest, black at the left abdominal and lastly the red at the left chest. It has clear waveform (P,QRS,T waves) with lesser noise in the graph. If this project will be implemented into medical server, choosing this electrode orientation will have best results compared to red electrode at right chest, black electrode at left abdominal and white electrode at left chest.

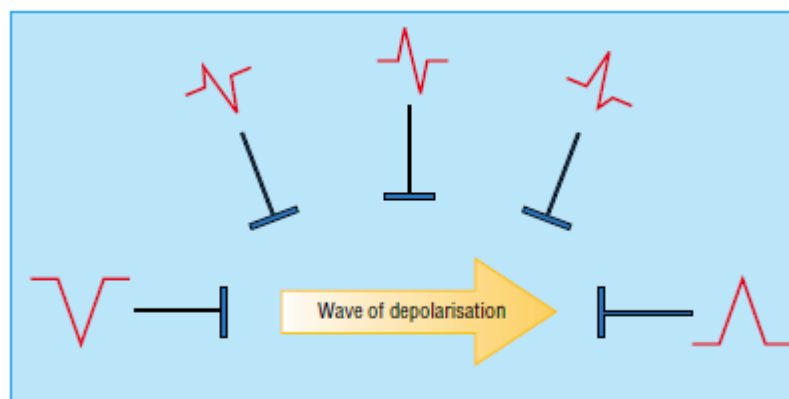


Figure 5.15: Shape of QRS complex depends on orientation of electrode



Figure 5.16: Best orientation of electrodes that output closest results to typical ECG waveform (LC- red, RC-white, LA- black)

5.4 Conversion Factor of ADC level to Voltage level

The x and y axis of the graph in MED-ECG are represented by time and ADC level respectively. The y axis range from -30000 to 30000 whereas the x axis parameter can be adjusted by pulling to the left by user. To convert the ADC level to the voltage level for convenient of diagnosis, follow the equation below:

$$V_{\text{signal}} = [(\text{Digital Value}) * (3.3 \text{ V})] / [(2^{16}) * (\text{Gain1}) * (\text{Gain2}) * (\text{Gain3})]$$

V_{signal} is the desired voltage output. Digital Value is the ADC results (obtain from raw data) Gain1 is 10x and 100x, depending of position of jumpers J3 and J4 of the MED-ECG. Gain2 range from 2x, 3x, 4x, 4.5x, 6x, 6.5x, 8.5x, 9x, 13x and 17x depending on the push button on TWR-MCF51MM (SW1, SW2 and SW4). Default Gain2 value is 9x. Gain3 is an optional fixed amplification block that can be selected, adds a gain of 3.3x. To calculate the voltage output correctly, the DSC function also has to disable by modify the source code to avoid any adding of gain and amplification of signal. 3.3V is the operating voltage and 2^{16} is due to 16 bits structure of ECG data. This information is not available in any literature yet and adapted from Freescale's support team.

5.5 Noise issue

As mentioned in earlier section, noise issue is a interference factors that hard to get rids of. The main sources of noise are (Enrique and Hartmann, 2003):

- Power-line interference: 50–60 Hz pickup and harmonics from the power mains
- Electrode contact noise: variable contact between the electrode and the skin, causing baseline drift
- Motion artifacts: shifts in the baseline caused by changes in the electrode-skin impedance
- Muscle contraction: electromyogram-type signals (EMG) are generated and mixed with the ECG signals
- Respiration, causing drift in the baseline
- Electromagnetic interference from other electronic devices, with the electrode wires serving as antennas, and
- Noise coupled from other electronic devices, usually at high frequencies.

The patient's conditions such as oily skin will prevent accurate data capturing. To solve it, scrub the attachment site thoroughly with alcohol to clean up oil and dust before apply the ECG electrode. ECG electrodes are optimally placed directly on dry skin and quality of ECG signal relies greatly on the skin condition. Loose connection of ECG electrodes will generate output with noise. Other noise such as muscle contraction, respiration and motion artifacts rely on the patient's conditions. Thus, every patient might have different ECG signal due to the stated noise factors.

As the prototype is connecting with USB port of computer, it is unavoidable add up some power line noise to the ECG data. The noise effect is more obviously seen in multiple USB device connected at computer simultaneously at the time of testing and output similar results in figure 5,14. To reduce the power line supply noise, try to unplug all unnecessary USB devices that are connected in the time of testing. The author also tried to use other USB port on the testing computer when encounter such problem and normally it solve the problem. If problem persists, try to test on other computer and laptop that has lesser power line supply noise. Sometime the computer or laptop's background programs will influence the ECG results, thus close all the unwanted programs during testing will aids obtain better ECG data.

It should be no surprise that noise can be a problem in ECG analysis. Fortunately, the signal-to-noise ratio is usually quite good in a person at rest. In an active person, however, there can be substantial low frequency (< 15 Hz) noise due to electrode motion, and high frequency (> 15 Hz) noise due to skeletal muscle activity. (Enrique and Hartmann, 2003) In addition, there is the possibility of noise at 60 Hz and its harmonics due to power-line noise results output like figure 5.14.

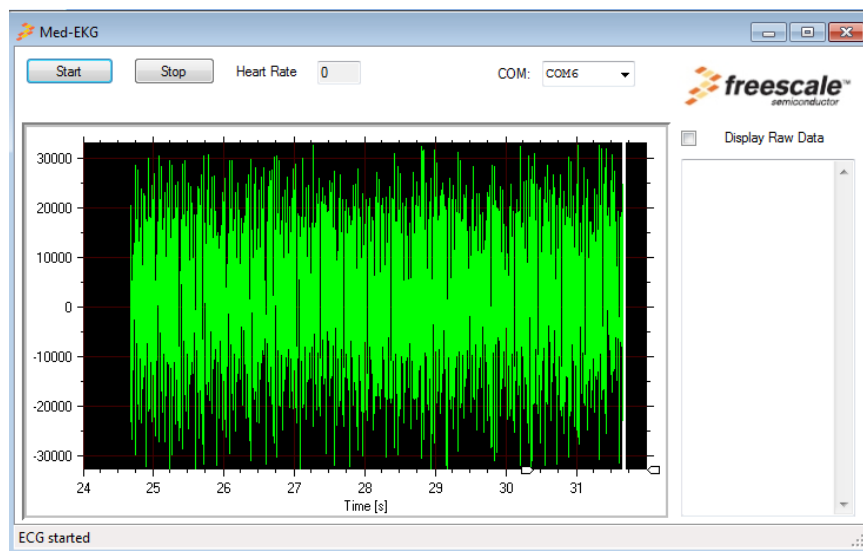


Figure 5.17: 60 Hertz Noise

As shown in the figure 5.17, the MED-ECG GUI record amplified noise which has high ADC level value (up to 30000). Normal ADC level for ECG varied from 8000 to 20000 only. No heart rate been recorded has the graph “bombarded” by continuous noise data (high amplitude), thus the GUI unable to estimate the heart rate. For heart beat detection, even a quarter of the standard distance, as the QRS amplitude detected can be used as estimation of heart beat. (Puurtinen, Hyttinen & Malmivuo, 2005) Figure 5.15 show raw data of incoming raw data with high value ADC, the display raw data column has been selected to display raw data. Figure 5.18 show example of 60 Hz noise (Erik Cheeve, 2005), the pattern of the ECG graph similar to figure 5.16 which justified that the captured noise is 60 Hz noise. The author has reviewed this problem with the manufacturer’s engineer via live chat. They justified that the noise is generated either by the power line or ECG electrodes.

The author then try on purposely loosening the ECG electrodes when capturing ECG data using MED-ECG and the results obtained was similar with figure 5.16.

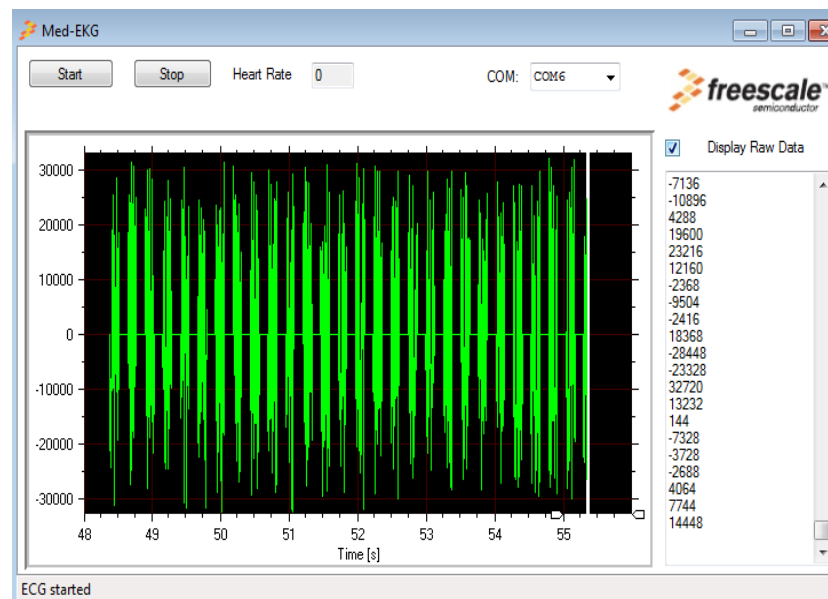


Figure 5.18: Value of ADC level during noise interference

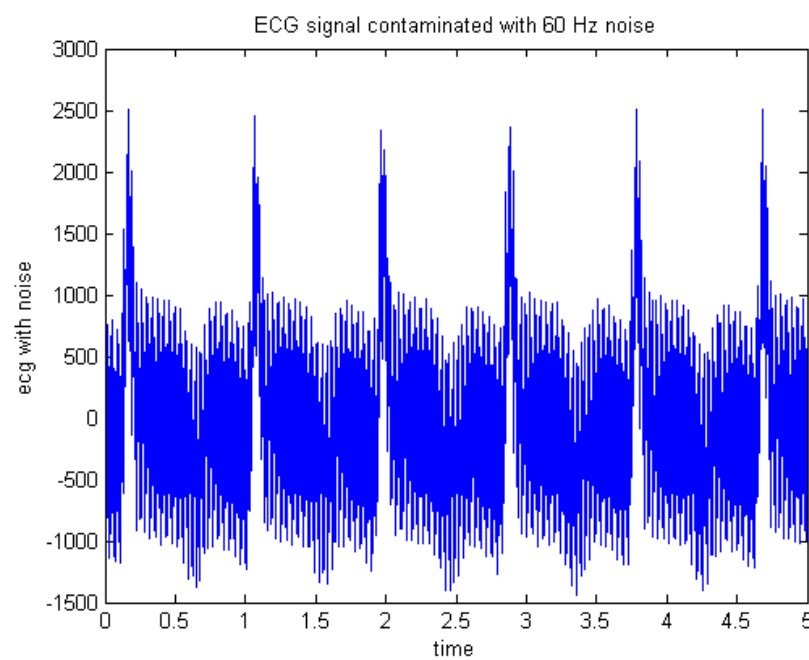


Figure 5.19: Example of 60 Hertz noise

5.6 XBee Buffer Issue

To ensure successful transmission of data via XBee module, the compatibility settings of module XBee module must be complementary as mentioned earlier. Initially the author set baud rate at 115200 instead of 9600 for wireless transmission and unrealized any possible problem might caused due to high baud rate. The XBee module at the receiver site stagnant (didn't receive any incoming data) after approximately 30 seconds of receiving ECG data from transceiver site. The author troubleshoots the problem and finally found out that the XBee module at the receiver site has reached its buffer stage. As explained in earlier section, once the DO Buffer reaches capacity (receive data), any additional incoming RF data is lost. (Digi Technologies, 2008) This occurs due to continuous data transfer from transceiver site at high baud rate until the memory of XBee module at the receiver site run off and the XBee module cannot receive anymore data after all the memory consumed.

There are two ways to solve this problem such as changing the stop bits to two instead of one and apply flow control. (John Foster, 2001) Another way suggested by Digi Technologies, 2008 which send data that are smaller than the DI buffer size is not practical as changing the file size of ECG data unaccomplished. Theory says that connect an XBee to a PC at 115,200 baud should be able to get away with one stop bit. In practice, any element of noise or other signal degradation is likely to lead to data loss. The baud rates are already set to the best obtainable values, so to prevent the problem the option is to configure the PC to send two stop bits. This can be configured using X-CTU program. Sending two stop bit slowing its transmission rate to one that the XBee can reliably accept. (John Foster, 2001) The first method didn't solve this problem perfectly as the problem still persists but the time of buffer stage achieved in XBee module extended. This is probably due to longer waiting period for receiver site (stop bits increment) but the baud rate is still too fast and consumed all the memory of XBee as time progress.

Flow control is necessary on any serial line if the rate of transmission is such that the receiving buffer cannot always be guaranteed to be emptied at a rate which prevents it from overflowing. There are two flow control mechanisms: hardware

flow control and software flow control. Hardware flow control is used between an XBee and its host. Software flow control is used between two hosts that are communicating via XBees. The XBees themselves do not take any notice of the software flow control characters. (John Foster, 2001) Hardware flow control require redesign the interfacing board as SKXBee board didn't support any hardware flow control. In this case, software flow control is a better and direct method to apply flow control. TWR-MCF51MM is the site where apply the programming modification and control the data. There are two crystals are provided on the board for clocking the TWR-MCF51MM256 such as 16 MHz crystal connected to XTAL2 and EXTAL2 for system clocking and 32.768kHz crystal connected to XTAL1 and EXTAL1 for TOD usage. (Freescale Inc, 2010) 16MHz crystal on board allows the author to modify the baud rate of the serial port to desired baud rate but involved modification of programming code (specifically on SCI command part). As a results, slowing down interface baud rate did help buffer overflow problem as the data is coming in slower than the over the air transmission rate and avoid data build up in the buffer. There is no buffer issue after software flow control applied. The baud rate slowed to 9600 instead of applying 115200. ECG data managed to send out and received in a continuous manner until the author stop the application.

CHAPTER 6

CONCLUSION

6.1 Comments on the project

The developed ECG prototype system was able to measure and output subject's ECG data in a GUI. Wireless transmission of captured ECG data has been established but the medical server has yet to develop. The prototype system is ready to use to develop with medical server which was not developed in this project. Programming the Freescale's medical development kit is the most challenging tasks in this project. The ECG prototype system programmed to capture process and output the ECG data using Code Warriors Development Studio. The author has presented the placements of ECG electrodes influence the pattern of captured ECG data in this report. Noise issue is another challenging aspect in ECG data capturing. Reducing the noise in ECG prototype system such as attach ECG electrode firmly, use low power line noise computer, clean attachment site of patient before applying electrodes, etc are important to ensure better quality of ECG data. As a conclusion, this project was succeeded with an ECG prototype system that able to capture ECG data and establish wireless transmission with a receiver at the other end. Both primary and secondary objective has been achieved! However, there are still some possible improvements which will be discussed in next section.

6.2 Possible Improvement and Recommendations for Future Works

There are improvements that can be applied in this project to improve the prototype system such introducing medical server, apply conductive gel, implement the XBee Pro module instead of XBee module, implement validation on the prototype system, developed wearable prototype system and construct the MAX232 circuitry on PCB board.

Introducing medical server at the receiver site of prototype system able to further enhances the function of prototype system. Designed medical server will able to diagnose possible complications of the heart via ECG results and enable alarm settings when threshold activated. The medical server can be developed using Labview and Visual Basic or any related development software. Once the medical server has been developed, running prototype system validation testing on real human might be a good way to determine the reliability of the system. No subject testing carry out in this project as no medical server been developed. Development of wearable ECG prototype system is another adding bonus to this system for convenient of data measurement. Follow the best orientation of electrodes as mentioned in discussion part to obtain ECG waveform that closest to typical one.

To provide are more range for data transmission from transmitter to receiver of ECG prototype system, the Xbee module need change to Xbee Pro module. ECG data would be transmit until 1.6 km and 50mw power transmit which is can transmit more length of data compared to Xbee that only 100m and 1.25mW for power transmit. The author has tried to test the maximum range which the XBee can receive and transmit data in the project too. He found out that obstacles along the transmission pathway will disrupt the wireless transmission. He is in opinion of changing XBee module to XBee Pro module might be a good try to improve wireless transmission quality for development of medical server. Construct MAX232 circuitry into PCB boards will secure the connections of components on it. There are no conductive gels used in this project as disposable ECG electrodes already have some conductive gel on it. The electrodes' connection can further enhanced if apply more conductive gel at attachment site.

REFERENCES

- Aily. (2009). Design and Implementation of ECG circuit. University Technology Malaysia, 18-19.
- Anwar. (2005). 3-Lead Wireless ECG, Electronic Design Project. 6-7.
- Brenda, Beasley & Michael. (2003). Understanding 12-lead EKGs 2nd edition. Upper Saddle River, New Jersey: Prentice Hall.
- Carlos, Americas & Guadalajara. (2010). Heart Rate Monitor and Electrocardiograph Fundamentals. Freescale Inc, 1-22.
- Chai, M. L. (2008). Keep the arteries unclogged. National Heart Association Malaysia.
- Cromwell, L. & Weibell, F. J. (2005). Design and Implementation of a Telemedicine System Using Bluetooth Protocol and GSM/GPRS Network, for Real-Time Remote Patient Monitoring. Technology and HealthCare Association.
- Crawford, M. H. (1999). ACC/AHA Guidelines for ambulatory electrocardiography. *Journal of the American College of Cardiology*, 34, 912-48.
- Cytron Technologies. (2008). SKXBee User`s Manual.
- Digi Technologies. (2008). Xbee datasheet.
- Digi Technologies. (2008). User`s Manual of X-CTU.

- Enrique & Hartmann. (2003). ECG Front-End Design is simplified with microconverter. Analog Device Inc. 1-3.
- Erik, C. (2005). Fundamentals of Digital Systems. Department of Engineering Swarthmore College. 76-77.
- Francis et al (2003). ABC of clinical Electrocardiograph. BMJ book. 22-30.
- Freescale Inc. (2010). User's Manual of TWR-MCF51MM kit.
- Freescale Inc. (2008). User's Manual of TWR-ELEV.
- Freescale Inc. (2008). User's Manual of TWR-SER.
- Freescale Inc. (2010). User's Manual of MED-ECG.
- Freescale Inc. (2010). Schematics Diagram of Freescale's Medical Development Kit.
- John Foster. (2001). XBee Cookbook Issue 1.3 for Series 1 (Freescale) with 802.15.4 Firmware. Digi Technologies. 1-22.
- Kong, K. Y., Ng, K.W. & Ong, K. (2000). Web-Based Monitoring of Real-Time ECG Data. *Computers in Cardiology*, 27, 189-192.
- Li & Zheng. (1995). *IEEE Trans. Biomedical Engineering*. 42-43.
- Meystre, (2005), The Current State of Telemonitoring: A Comment on the Literature, Telemedicine and e-Health Association, 1-5.
- Mohamed, M. (2000). The Problems and Challenges of the Aging Population. *Malaysia. Malaysian Journal of Medical Sciences*, 7, 1-3

- Meystre, S. (2005). The Current State of Telemonitoring: A Comment on the Literature. *Telemedicine and e-Health*. 5-7.
- Norris. (2002), *Essential of Telemedicine and Telecare*, Department of Management Science and Information System University of Auckland New Zealand.
- Nor Syahidatul et al., (2008), TRG Wireless ECG Sensor for Medical Healthcare Application, 2008 Student Conference on Research and Development (SCORED 2008), 26-27.
- Tracy. (2003). *Diagnosis and Study of the ECG Pattern*, Nursecom Education Technology. Chapter 4.
- Patrick et al. (2002), *Electrocardiogram (EKG) Data Acquisition and Wireless Transmission*. Southern Polytechnic State University.
- Puurtinen, M., Hyttinen, J. & Malmivuo, J. (2005). Optimizing bipolar electrode location for wireless ECG measurement – analysis of ECG signal strength and deviation between individuals. *Tampere University of Technology*. 1-4.
- Saritha & Sukanya. (2008). ECG Signal Analysis Using Wavelet Transforms. *Bulg. J. Phys.* Chapter 35: 68–77.
- Shimuzu. (2005). *Telemedicine by Mobile Communication*. *IEEE Engineering in Medicine and Biology*.
- Tanriverdi, H. & Iacono, C. S. (1999). Diffusion of Telemedicine: A Knowledge Barrier Perspective. *Telemedicine Journal*. 5(3).
- Thakor, N. V. & Webster, J. G. (1980). Ground-Free ECG Recording with Two Electrodes. *IEEE Trans on Biomed Eng.* 27, 12, 699-704.

Telemedicine Research Centre. (1999). What is Telemedicine?, Oregon Health Sciences University, Retrieval date on 12 August 2010 , From <http://tie.telemed.org/WhatIsTelemedicine.asp>

Texas Instrumentation Inc. (2004). Datasheet for MAX232.

Venkat, B. (2002). ZigBee and Bluetooth – Competitive or Complementary. ZigBee Alliance.

Venkat, B. (2002). ZigBee Overview. ZigBee Alliance.

WHO. (2004). Atlas of Heart Disease and Stroke.

Yayasan Jantung Malaysia (2009), Articles of Interest about your HEART, Retrieved on 11 August 2010, from <http://www.yjm.org.my/newsmaster.cfm?&menuid=31&action=view&retrieveid=3>

APPENDICES

Appendix A: Pin Layout Table for Freescale's Medical Development Kit

Medical Connector 2x10 Pin Header Connections (TWR-MCF51MM)

MCF51MM256 Signal	Pin	Pin	MCF51MM256 Signal
MOSFET Q6 (Pin 3)- Power (3.3V)	1	2	GND
PTD4/SD4/RGPIOP10/TPM1CH2	3	4	PTD5/SD5/RGPIOP11/TPM1CH3
DADP0	5	6	DADM0
PCT4/KBIP7/CMPP0/ADP8	7	8	DAXO_E
OUT1	9	10	OUT2
INP1-	11	12	INP2-
PTA4/INP1+	13	14	PTA7/INP2+
VINP1	15	16	VINP2
VINN1/DADM2	17	18	VINN2/DADM3
TRIOUT1/DADP2	19	20	TRIOUT2/DADP3

Medical Connector 2x10 Pin Header Connections (MED-ECG)

MED-ECG signal	Pin	Pin	MED-ECG Signal
Vdd (3.3V)	1	2	GND
ECG_IIC_SDA	3	4	ECG_IIC_SCL
MM_AdcEcgSignal	5	6	GND
MM_AdcBaseline	7	8	MMDac
OUT1	9	10	OUT2
INP1-	11	12	INP2-

INP1+	13	14	INP2+
VNP1	15	16	VINP2
VINN1	17	18	VINN2
TRIOUT1	19	20	TRIOUT2

Appendix B: Programming Files and Functions for Freescale's Medical Development Kit

Programming File	Functions
tpm.c	Configures Real Time Counter (RTC) for Timer Implementation.
main.c	Main function of programming.
IICV1Driver.c	Address control of data and interrupt flag from MED-ECG.
OPAMP.c	Control of op-amp gain on MED-ECG.
EcgDsc.c	Applying of digital signal conditioning filtration on ECG data.
Ecg.c	Processing of ECG data.
ADC.c	Analog to digital conversion.
AverageFilter.c	Filtering of ECG file.
DAC.c	Digital to Analog conversion.
Ekg.c	Outputting the ECG data
Usb_descriptor.c	USB standard (Freescale stack).
MCF51MM256.c	standard library of MCF51MM microcontroller.

Header File

IICV1Driver.h
OPAMP.h
EcgDsc.h
Ecg.h
ADC.h
AverageFilter.h
DAC.h
Ekg.h
MCF51MM256.h

USB class file and driver

Usb_framework.c
Usb_class.c
Usb_cdc.c
Usb_cdc_pstn.c
Usb_driver.c
Usb_dci.c

Appendix C: Default Jumper Settings for Freescale's Medical Development Kit

TWR-MCF51MM default jumper settings

Jumper Name	Default Setting on TWR-MCF51MM Module
J16	1 and 2
J15	1 and 2
J12	Open
J3	Open
J8	Open
J7	Open
J5	Open
J6	1 and 2
J1	1 and 2
J2	1 and 2
J10	2 and 3
J19	1 and 2, 3 and 4, 5 and 6
J26	2 and 3
J25	6 and 5, nothing on pin 4, 3, 2, 1
J24	Open
J11	1 and 2
J9	1 and 2
J4	1 and 2
J18	1 and 2, 3 and 4, 5 and 6, 7 and 8, 9 and 10, 11 and 12, 13 and 14
SW3	Slider 1 at OFF (Position 1)
	Slider 2 at OFF (Position 2)

MED-ECG default jumper settings

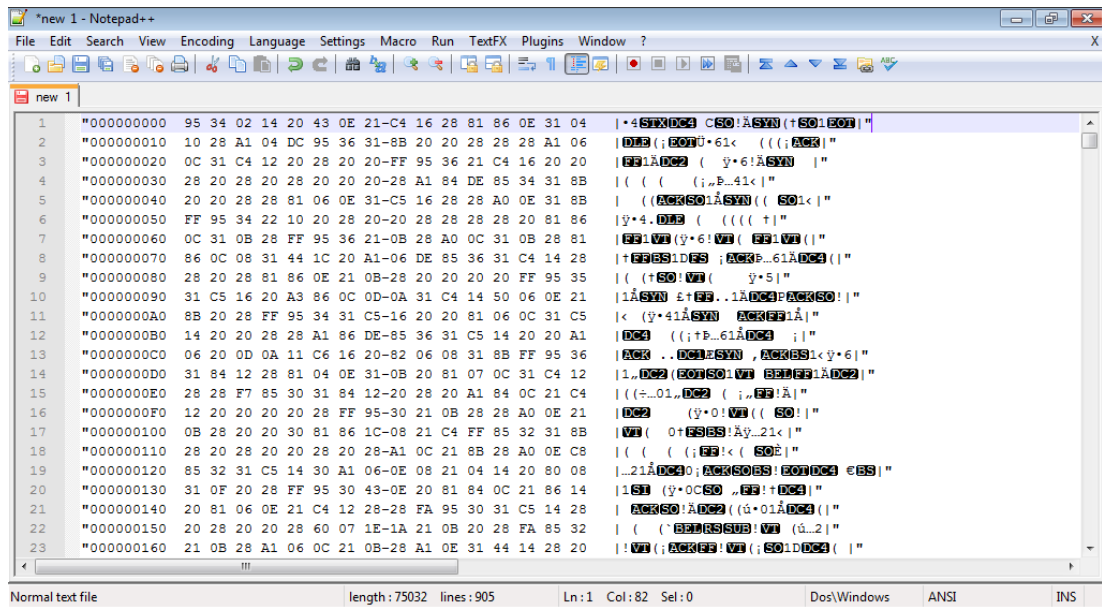
Jumper Name	Default Setting on TWR-MCF51MM Module
J2	1 and 2
J3	2 and 3
J4	2 and 3
J5	Open
J6	2 and 3
J7	2 and 3
J8	2 and 3
J9	2 and 3
J10	All open (J10 connects to DSC programmer. It is not needed for this demo.)
J11	2 and 3

TWR-SER default jumper settings

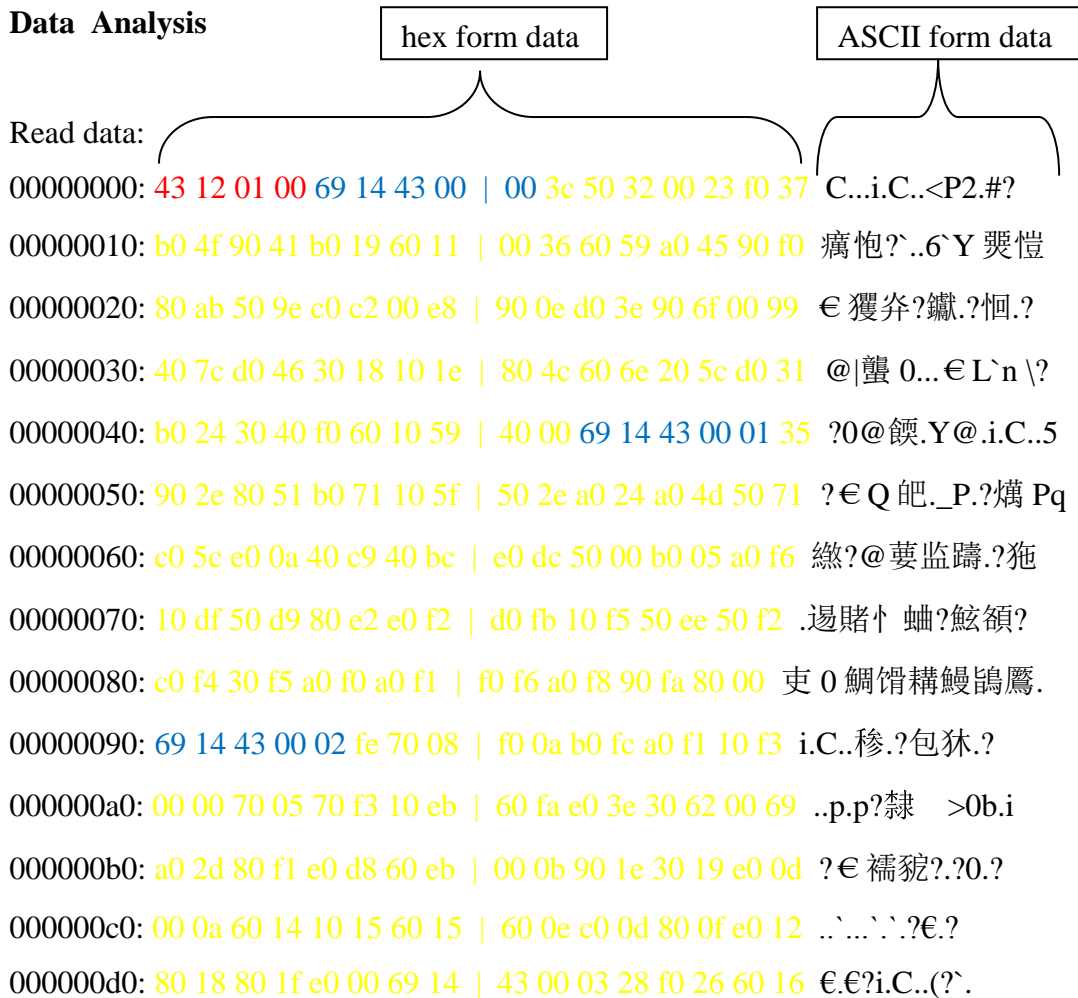
Jumper	Option	Setting	Description
J2	Ethernet PHY Clock Select	1-2	25 MHz
		3-4	50 MHz
		5-6	CLOCKOUT0
J3	CLOCKIN0 Driver Select	1-2	Route 25MHz clock to CLOCKIN0
		2-3	Route 50MHz clock to CLOCKIN0
J5	CAN Selection Options	1-2	Put CAN transceiver into sleep mode
		3-4	Connect Sleep pin to CAN pin (B43)
		5-6	Connect RXD pin to CANRX pin (B41)
		7-8	Connect TXD pin to CANTX pin (B42)
		9-10	Apply 120ohm termination resistor
J6	Ethernet PHY Interrupt Select	1-2	IRQ_H
		3-4	IRQ_F
		5-6	IRQ_D
		7-8	IRQ_B
J10	USBVBUS Select	1-2	Supply 5V on USB Connector (Host Mode)
		2-3	Source 5V from USB (Bus-powered Device)

Jumper	Option	Setting	Description
J11	USB OTG Interrupt Select	1-2	IRQ_H
		3-4	IRQ_F
		5-6	IRQ_D
		7-8	IRQ_B
J12	Ethernet PHY Configuration	1-2	Pull-up PHYAD2; PHY Address Select
		3-4	Pull-up PHYAD1; PHY Address Select
		5-6	Pull-down PHYAD0; PHY Address Select
		7-8	Pull-up CONFIG2; Loopback Select
		9-10	Pull-up CONFIG0; RMI Select
		11-12	Pull-up ISO; Isolation Mode Select
		13-14	Pull-down SPEED; 10Mbps Select
		15-16	Pull-down DUPLEX; Half-duplex Select
J13	Misc RS232/485 Config	1-2	Connect RS485 Receive En and Driver En
		3-4	Connect RS485 RX+ to TX+; Loopback
		5-6	Connect RS485 RX- to TX-; Loopback
		7-8	Enable ELE_CTS (A9) as RS232 CTS
		9-10	Supply 5V on DB9 pin 6
J15	RS232 / RS485 Select	1-2	RS232
		2-3	RS485
J16	USB Mode Select	1-2	Host Mode—supply 5V to VBUS
		3-4	Device Mode—source 5V from VBUS
		5-6	OTG Mode—VBUS controlled by OTG Charge Pump
J17	RS232 / RS485 RX Select	1-2	RS232
		2-3	RS485
J18	RS232 / RS485 RTS Select	1-2	RS232
		2-3	RS485
J19	RS232 / RS485 TX Select	1-2	RS232
		2-3	RS485

Appendix E: Notepad++ (convert ASCII to hex) and Analysis of hex data



Data Analysis



```

000000e0: 80 0f 00 16 90 23 90 22 | c0 0b 20 04 e0 19 90 5f € ...??? .?悒
000000f0: 30 7d c0 80 c0 44 f0 0d | c0 f6 f0 07 b0 25 d0 38 0}纒績?丽???
00000100: 40 36 10 2a 10 27 60 0b | e0 ef 30 c7 40 bd 10 c1 @6.*.^.囡 0 莽??
00000110: 80 e3 00 f5 c0 20 70 41 | e0 51 f0 00 43 13 00 00 € ?跽 pA 鄭?C...
00000120: 00 00 00 00 00 00 00 00 | 00 00 00

```

```

.....
.....
.....

```

Comments

Red Number is the acknowledge for Start command.

Blue Numbers is the header for ECG data.

Yellow Numbers are ECG data. They are the RAW data displayed in GUI.

For example :0x3c50 is the first data, 0x3200 is the second data.....

Appendix F: Pinout for XBee and XBee Pro

Pin	Name	Direction	Description	AT	PR
1	VCC	–	Power supply		
2	DOUT	Output	UART data out		
3	DIN / $\overline{\text{CONFIG}}$	Input	UART data in		7
4	DO8	Output	Digital output 8		
5	$\overline{\text{RESET}}$	Input	Module reset (at least 200nS)		
6	PWM0 / RSSI	Output	PWM output 0 / RX signal strength indicator		
7	PWM1	Output	PWM output 1		
8	(reserved)		Do not connect		
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin sleep control line or digital input 8	D8	6
10	GND	–	Ground		
11	AD4 / DIO4	Either	Analog input 4 or digital I/O 4	D4	0
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear to send flow control or digital I/O 7	D7	
13	ON / $\overline{\text{SLEEP}}$	Output	Module status indicator		
14	VREF	Input	Voltage reference for AD inputs		
15	Associate / AD5 / DIO5	Either	Associated indicator, analog input 5 or digital I/O 5	D5	
16	$\overline{\text{RTS}}$ / AD6 / DIO6	Either	RTS flow control, analog input 6 or digital I/O 6	D6	5
17	AD3 / DIO3	Either	Analog input 3 or digital I/O 3	D3	1
18	AD2 / DIO2	Either	Analog input 2 or digital I/O 2	D2	2
19	AD1 / DIO1	Either	Analog input 1 or digital I/O 1	D1	3
20	AD0 / DIO0	Either	Analog input 0 or digital I/O 0	D0	4

Appendix G: Programming Source Code

Only Ecg.c and Ekg.c will be display in this section. There are other source codes like file name (.c, .h file) display in Appendix B. For others source codes, please refer to the project CD.

Ecg.c

```
#include "ECG.h"

//public variables
UINT8 Ecg_HeartRate = 0;
UINT8 EcgDataBuffer[ECG_DATA_BUFFER_LENGTH];

//private
static UINT8 QRSFound = 0;
static UINT16 RealTimeHeartRate;
static UINT16 EcgHeartRateSum;
static UINT16 EcgHeartRateArray[ECG_ARRAY_LENGTH];
static UINT16 CopyOfEcgHeartRateArray[ECG_ARRAY_LENGTH];
static UINT8 EcgCurrentArrayPosition = 0;
static UINT16 EcgFirstSample = 0;
static UINT16 EcgSecondSample = 0;
static UINT16 EcgThirdSample = 0;
static UINT16 samplesBetweenPulses = 0;
static UINT8 HeartBeatOcurrred;
static UINT8 IsEcgSignalReady;

/* Private functions */
static void StateIdle(void);
static void StateMeasuring(void);
static void TimerSampleAdc_Event(void);

static void PerformControlAlgorithm(void);
static void SendGraphDataToPc(void);
static void ClearAllVariables(void);
static void CalculateHeartRateMedian(void);

/* Main state machine */
void (*const EcgStateMachine[]) (void) =
{
    StateIdle,
    StateMeasuring
};

/* Private Macros */

typedef enum
{
    STATE_IDLE,
    STATE_MEASURING
} EcgStates_e;

/* Private variables */
static MovingAverage_uint16_t FeedbackSignal, EcgSignal;
static UINT8 EcgActualState = STATE_IDLE;
static TIMER_OBJECT TimerEcgSampleAdc;
static UINT8 TimerEcgSampleAdcIndex;
static UINT8 EcgActualEvent = EVENT_ECG_NONE;

static UINT8 IsHeartRateMode = FALSE;
static IsNewEcgSampleReady = FALSE;
```

```

static UINT16 PulseDetectedWatchDog = 0;

/*Francisco Variables */
static UINT8 n = 100;
static UINT16 PwmValue = 0;

/*****

/* Private function definitions */

static void StateIdle(void)
{
    //do nothing
}

/*----- definitions and variables -----*/
#define SAMPLES_NUMBER 15

#define CENTER_LOW 32768-3000 //1.45V
#define CENTER_HI 32768+3000 //1.75V

#define LOW_LIMIT_1 32768-12000 //1.0V
#define HI_LIMIT_1 32768+12000 //2.2V
#define LOW_LIMIT_2 32768-22000 //0.5V
#define HI_LIMIT_2 32768+22000 //2.7V

#define FEW_CORRECTION 19 //0.015V -> with 3.3 gain, it adds about 0.05V
#define MORE_CORRECTION 38 //0.030V -> with 3.3 gain, it adds about 0.10V
#define MUCH_CORRECTION 76 //0.060V -> with 3.3 gain, it adds about 0.20V
//#define FEW_CORRECTION 38 //0.03V -> with 3.3 gain, it adds about 0.1V
//#define MORE_CORRECTION 114 //0.09V -> with 3.3 gain, it adds about 0.3V
//#define MUCH_CORRECTION 190 //0.16V -> with 3.3 gain, it adds about 0.5V

UINT8 SampleCounter = 0, i_index, j_index;
UINT16 median_array[SAMPLES_NUMBER], temp_var, median_val;

/*-----*/

static void PerformControlAlgorithm(void)
{
    if ( SampleCounter < SAMPLES_NUMBER)
    {
        median_array[SampleCounter] = FeedbackSignal.Result;
        SampleCounter++;
    }
    else
    {
        SampleCounter = 0;

        //ordering the data

        for (i_index = 0; i_index <= SAMPLES_NUMBER - 2; i_index++)
        {
            for (j_index = i_index + 1; j_index <= SAMPLES_NUMBER - 1; j_index++)
            {
                if (median_array[i_index] > median_array[j_index])
                {
                    temp_var = median_array[i_index];
                    median_array[i_index] = median_array[j_index];
                    median_array[j_index] = temp_var;
                }
            }
        }
    }
}

```

```

    }
}

//obtaining the median
median_val = median_array[SAMPLES_NUMBER/2];

//compensate according to the value of the median
if ( (median_val < CENTER_LOW) || (median_val > CENTER_HI) )
{
    if ( (median_val < CENTER_LOW)    &&
         (median_val > LOW_LIMIT_1)   &&
         (DACDAT0 < (4094 - FEW_CORRECTION)))
    {
        DACDAT0 += FEW_CORRECTION;
    }

    if ( (median_val > CENTER_HI)    &&
         (median_val < HI_LIMIT_1)   &&
         (DACDAT0 > (FEW_CORRECTION + 1)))
    {
        DACDAT0 -= FEW_CORRECTION;
    }

    if ( (median_val < LOW_LIMIT_1)   &&
         (median_val > LOW_LIMIT_2)   &&
         (DACDAT0 < (4094 - MORE_CORRECTION)))
    {
        DACDAT0 += MORE_CORRECTION;
    }

    if ( (median_val > HI_LIMIT_1)    &&
         (median_val < HI_LIMIT_2)    &&
         (DACDAT0 > (MORE_CORRECTION + 1)))
    {
        DACDAT0 -= MORE_CORRECTION;
    }

    if ( (median_val < LOW_LIMIT_2)   &&
         (DACDAT0 < (4094 - MUCH_CORRECTION)))
    {
        DACDAT0 += MUCH_CORRECTION;
    }

    if ( (median_val > HI_LIMIT_2)    &&
         (DACDAT0 > (MUCH_CORRECTION + 1)))
    {
        DACDAT0 -= MUCH_CORRECTION;
    }

}

//if the median are in the range, the DAC provides 1.6V
else
{
    DACDAT0 = 2047; //1.6V output
}
}
}

```

```

static void StateMeasuring(void)
{

```

```

if (IsNewEcgSampleReady) //the timer is reading periodically
{
    {
        UINT16 feedbackSignalRaw, ecgSignalRaw;
        IsNewEcgSampleReady = FALSE;

        //read ADC signals and average values
        feedbackSignalRaw = ADC_Read16b(ADC_CHANNEL_FEEDBACK_SIGNAL);
        ecgSignalRaw = ADC_Read16b(ADC_CHANNEL_ECG_SIGNAL);

        MovingAverage_PushNewValue16b(&FeedbackSignal, feedbackSignalRaw);
        MovingAverage_PushNewValue16b(&EcgSignal, ecgSignalRaw);

        TimerEcgSampleAdcIndex = AddTimerQ(&TimerEcgSampleAdc);
    }

    PerformControlAlgorithm();
}

#ifndef ECG_DSC

    samplesBetweenPulses++;           //increment sample sampleCounter between pulses
    PulseDetectedWatchDog++;         //this variable increments every
ECG_SAMPLE_PERIOD (2ms)

    if (PulseDetectedWatchDog > MAX_TIME_WITHOUT_PULSES)
    {
        Ecg_HeartRate = 0;           //set HR = 0
    }

    EcgFirstSample = EcgSecondSample;
    EcgSecondSample = EcgThirdSample;
    EcgThirdSample = EcgSignal.Result;

    //If the slope of the signal is big, there is a peak

    if ((EcgThirdSample < EcgFirstSample) && ((EcgFirstSample - EcgThirdSample) >
HR_SLOPE_THRESHOLD))
    {
        //Peak detected
        QRSFound++;
        PulseDetectedWatchDog = 0;

        if (samplesBetweenPulses > 75)           //check if current peak is not to
close to previous peak (Max HR = 200)
        {
            IsEcgSignalReady = TRUE;
            HeartBeatOcurrred = TRUE;

            if (QRSFound > 4)                       //
Find 4 Pulses Before Considering Signal is Ready
            {
                UINT8 i;

                //4 pulses has been found
                RealTimeHeartRate = 60000 / (samplesBetweenPulses *
ECG_SAMPLING_PERIOD);           //Calculate HR

                //Shift samples of FIFO

                for (i = OLDEST_ELEMENT; i > NEWEST_ELEMENT; i--)
                {

```

```

        EcgHeartRateArray[i] = EcgHeartRateArray[i-1];
    }

    //insert new sample into FIFO
    EcgHeartRateArray[NEWEST_ELEMENT] =

RealTimeHeartRate;

    //Call the function to calculate heart rate
    CalculateHeartRateMedian();

    }
    else
    {
        //less than 4 QRs
    }
}

else if (
    ((samplesBetweenPulses > 10) && (samplesBetweenPulses < 500)) ||
    ((EcgFirstSample - EcgSecondSample) > 400) ||
    (EcgSignal.Result > 1750)
)
{
    //time between peaks is shorter
    IsEcgSignalReady = FALSE;
}

samplesBetweenPulses = 0;
}

#endif

if (!IsHeartRateMode)
{
    SendGraphDataToPc();
}

#endif
}

#define ZERO_SIGNAL        0x00

static void SendGraphDataToPc(void)
{
    static UINT8 actualPosition = 0;

    //store data and send it when the buffer is full

    if (actualPosition < ECG_DATA_BUFFER_LENGTH)
    {
        //if (IsEcgSignalReady)
        {
            //we need to convert unsigned values to signed values to display them on GUI
            INT16 signedEcgSignal = (INT16)(EcgSignal.Result - 0x8000);

            EcgDataBuffer[actualPosition++] = (UINT8)(signedEcgSignal >> 8);
            //higher byte
            EcgDataBuffer[actualPosition++] = (UINT8)(signedEcgSignal & 0x00FF);
            //lower byte
        }/*
    }
    else

```



```

        {
            EcgDataBuffer[actualPosition++] = (UINT8)(ZERO_SIGNAL >> 8);
//higher byte
            EcgDataBuffer[actualPosition++] = (UINT8)(ZERO_SIGNAL & 0x00FF);
//lower byte
        }*/
    }
    else
    {
        //buffer is full, call the EVENT_ECG_DATA_READY event
        EcgActualEvent = EVENT_ECG_DIAGNOSTIC_MODE_NEW_DATA_READY;
        actualPosition = 0;
    }
}

```

```

static void TimerSampleAdc_Event(void)
{
    IsNewEcgSampleReady = TRUE;
}

```

```

/*****
* Public functions
*****/

```

```

/* call this only once at the beginning of the application */

```

```

void Ecg_Init(void)
{
    TimerEcgSampleAdc.msCount = ECG_SAMPLING_PERIOD;
    TimerEcgSampleAdc.pfnTimerCallback = TimerSampleAdc_Event;
}

```

```

UINT8 Ecg_DiagnosticModeStartMeasurement(void)

```

```

{
    UINT8 status = FALSE;

    if (EcgActualState == STATE_IDLE)
    {
        ADC_Init16b(1 << ADC_CHANNEL_FEEDBACK_SIGNAL |
            1 << ADC_CHANNEL_ECG_SIGNAL);

        //TPM1_Init();    //TPM is not used in MM version, DAC is used instead

        //TO DO:

        //DAC_Init();
        //OpAmps_Init();

        ClearAllVariables();

        EcgActualState = STATE_MEASURING;
        IsHeartRateMode = FALSE;

        //start timer to sample ADC
        TimerEcgSampleAdcIndex = AddTimerQ(&TimerEcgSampleAdc);
        status = TRUE;
    }
    else
    {
        status = FALSE;
    }
}

```

```

        return status;
    }

static void ClearAllVariables(void)
{
    UINT8 i;

    Ecg_HeartRate = 0;
    QRSFound = 0;
    RealTimeHeartRate = 0;
    EcgHeartRateSum = 0;
    EcgCurrentArrayPosition = 0;
    EcgFirstSample = 0;
    EcgSecondSample = 0;
    EcgThirdSample = 0;
    samplesBetweenPulses = 0;
    HeartBeatOccurred = 0;
    IsEcgSignalReady = 0;
    EcgCurrentArrayPosition = 0;
    PulseDetectedWatchDog = 0;

    //clear arrays
    for (i = 0; i < ECG_ARRAY_LENGTH; i++)
    {
        EcgHeartRateArray[i] = 0;
    }
}

void Ecg_DiagnosticModeStopMeasurement(void)
{
    RemoveTimerQ(TimerEcgSampleAdcIndex);
    EcgActualState = STATE_IDLE;
}

void Ecg_PeriodicTask(void)
{
    /* State machine handler */
    EcgStateMachine[EcgActualState]();

    /* Event handler */

    if (EcgActualEvent != EVENT_ECG_NONE)
    {
        if (Ecg_Events[EcgActualEvent] != NULL)
        {
            Ecg_Events[EcgActualEvent](); //execute registered event
            EcgActualEvent = EVENT_ECG_NONE;
        }
    }
}

```

```

static void CalculateHeartRateMedian(void)
//order HeartRate values in array and average samples in the middle
{
    UINT8 startIndex;
    UINT8 smallestIndex;
    UINT8 currentIndex;
    UINT8 tempStoreValue;
    UINT8 i;
    static UINT16 Ecg_HeartRate_MedianSum = 0;
    static UINT8 median_counter = 0;

    //Create a copy of the arrays

    for (i = 0; i < ECG_ARRAY_LENGTH; i++)
    {
        CopyOfEcgHeartRateArray[i] = EcgHeartRateArray[i];
    }

    // Order array values in ascending order
    for (startIndex = 0; startIndex < ECG_ARRAY_LENGTH; startIndex++)
    {
        smallestIndex = startIndex;

        for (currentIndex = startIndex + 1; currentIndex < ECG_ARRAY_LENGTH; currentIndex++)
        {
            if (CopyOfEcgHeartRateArray[currentIndex] <
CopyOfEcgHeartRateArray[smallestIndex])
            {
                smallestIndex = currentIndex;
            }
        }

        tempStoreValue = (UINT8) CopyOfEcgHeartRateArray[startIndex];

        CopyOfEcgHeartRateArray[startIndex] = CopyOfEcgHeartRateArray[smallestIndex];
        CopyOfEcgHeartRateArray[smallestIndex] = tempStoreValue;
    }

    //Obtaining the median
    if (median_counter < 4)
    {
        Ecg_HeartRate_MedianSum += CopyOfEcgHeartRateArray[ECG_ARRAY_LENGTH/2];
        median_counter++;
    }
    else
    {
        Ecg_HeartRate = (UINT8)(Ecg_HeartRate_MedianSum / 4);
        median_counter = 0;
        Ecg_HeartRate_MedianSum = 0;
    }
}

```

ekg.c

```

/*****
* Includes
*****/
#include "hidef.h" /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include "types.h" /* Contains User Defined Data Types */
#include "usb_cdc.h" /* USB CDC Class Header File */
#include "ekg.h" /* Virtual COM Application Header File */

```

```

#include "SerialCommands.h"
#include "OPAMP.h"
#include "DAC.h"
#include "EcgDsc.h"
#include "Ecg.h"

#if (defined _MCF51MM256_H) || (defined _MCF51JE256_H)
#include "exceptions.h"
#endif

#if (defined(_MC9S08MM128_H))
#define DEBOUNCE_TIME 10000
#endif
#if (defined _MCF51MM256_H)
#define DEBOUNCE_TIME 25000
#endif

/* skip the inclusion in dependency state */
#ifndef __NO_SETJMP
#include <stdio.h>
#endif
#include <stdlib.h>
#include <string.h>

/*****
 * Local Types - None
 *****/
typedef enum
{
    NO_MEASUREMENT,
    GLU_MEASUREMENT,
    BPM_MEASUREMENT,
    BPM_LEAK_TEST,
    ECG_MEASUREMENT,
    SPR_MEASUREMENT,
    HEIGHT_MEASUREMENT,
    WEIGHT_MEASUREMENT,
    TEMPERATURE_MEASUREMENT
}EkgCommand_e;

/*****
 * Local Functions Prototypes
 *****/
static void USB_App_Callback(uint_8 controller_ID,
                             uint_8 event_type, void* val);
static void USB_Notify_Callback(uint_8 controller_ID,
                                uint_8 event_type, void* val);
static void Virtual_Com_Recv_Serial_Data(void);

static void Virtual_Com_Send_Serial_Data(void);

static void EcgDiagnosticModeStartMeasurementReq(void);
static void EcgDiagnosticModeStopMeasurementReq(void);
static void EcgDiagnosticModeNewDataReadyInd(void);
static void SCII_Init(void);
static void SCISendData(const byte SendData[],byte ssize);

#ifndef SEND_SINE_WAVE
static void TimerSendDummyData_Event(void);
#endif

/*****
 * Constant and Macro's - None
 *****/
const pFunc_t Ecg_Events[] =

```

```

{
    NULL,
    //EVENT_ECG_NONE,
    NULL, //EVENT_ECG_HEART_RATE_MEASUREMENT_COMPLETE_OK,
    NULL, //EVENT_ECG_HEART_RATE_MEASUREMENT_ERROR,
    NULL, //EcgHeartBeatOccurredInd,
    //EVENT_ECG_HEART_BEAT_OCCURRED,
    EcgDiagnosticModeNewDataReadyInd
    //EVENT_ECG_DIAGNOSTIC_MODE_NEW_DATA_READY
};

```

```

const pFunc_t EcgDsc_Events[] =
{
    NULL,
    //EVENT_ECG_NONE,
    NULL, //EVENT_ECG_HEART_RATE_MEASUREMENT_COMPLETE_OK,
    NULL, //EVENT_ECG_HEART_RATE_MEASUREMENT_ERROR,
    NULL, //EcgHeartBeatOccurredInd,
    //EVENT_ECG_HEART_BEAT_OCCURRED,
    EcgDiagnosticModeNewDataReadyInd
    //EVENT_ECG_DIAGNOSTIC_MODE_NEW_DATA_READY
};

```

```

const pFunc_t ExecuteCommandReq[] =
{
    NULL, //GLU_START_MEASUREMENT,
    NULL, //GLU_ABORT_MEASUREMENT,
    NULL, //GLU_START_CALIBRATION,
    NULL,
    //GLU_BLOOD_DETECTED,
    NULL,
    //GLU_MEASUREMENT_COMPLETE_OK,
    NULL,
    //GLU_CALIBRATION_COMPLETE_OK,

    NULL, //BPM_START_MEASUREMENT,
    NULL, //BPM_ABORT_MEASUREMENT,
    NULL,
    //BPM_MEASUREMENT_COMPLETE_OK,
    NULL,
    //BPM_MEASUREMENT_ERROR,

    NULL, //BPM_START_LEAK_TEST,
    NULL, //BPM_ABORT_LEAK_TEST,
    NULL,
    //BPM_LEAK_TEST_COMPLETE,

    NULL, //ECG_HEART_RATE_START_MEASUREMENT,
    NULL, //ECG_HEART_RATE_ABORT_MEASUREMENT,
    NULL,
    //ECG_HEART_RATE_MEASUREMENT_COMPLETE_OK,
    NULL,
    //ECG_HEART_RATE_MEASUREMENT_ERROR,
    NULL,
    //ECG_HEART_BEAT_OCCURRED,

    EcgDiagnosticModeStartMeasurementReq,
    //ECG_DIAGNOSTIC_MODE_START_MEASUREMENT,
    EcgDiagnosticModeStopMeasurementReq,
    //ECG_DIAGNOSTIC_MODE_STOP_MEASUREMENT,
    EcgDiagnosticModeNewDataReadyInd,
    //ECG_DIAGNOSTIC_MODE_NEW_DATA_READY,

```

```

NULL, //TMP_READ_TEMPERATURE,
NULL, //HGT_READ_HEIGHT,
NULL, //WGT_READ_WEIGHT,

NULL, //SprStartMeasurementReq,
//SPR_START_MEASUREMENT,
NULL, //SprAbortMeasurementReq,
//SPR_ABORT_MEASUREMENT,
NULL,
//SPR_MEASUREMENT_COMPLETE_OK,
NULL,
//SPR_MEASUREMENT_ERROR,

NULL,
//SPR_DIAGNOSTIC_MODE_START_MEASUREMENT,
NULL,
//SPR_DIAGNOSTIC_MODE_STOP_MEASUREMENT,
NULL,
//SPR_DIAGNOSTIC_MODE_NEW_DATA_READY,
NULL,
//SPR_DIAGNOSTIC_MODE_MEASUREMENT_COMPLETE_OK,
NULL,
//SPR_DIAGNOSTIC_MODE_MEASUREMENT_ERROR,

NULL, //PoxStartMeasurementReq,
//POX_START_MEASUREMENT,
NULL, //PoxAbortMeasurementReq,
//POX_ABORT_MEASUREMENT,
NULL,
//POX_MEASUREMENT_COMPLETE_OK,
NULL,
//POX_MEASUREMENT_ERROR,

NULL,
//PoxDiagnosticModeStartMeasurementReq//POX_DIAGNOSTIC_MODE_START_MEASUREMENT,
T,
NULL,
//PoxDiagnosticModeStopMeasurementReq//POX_DIAGNOSTIC_MODE_STOP_MEASUREMENT,
NULL,
//POX_DIAGNOSTIC_MODE_NEW_DATA_READY
NULL, //BPM_SEND_PRESSURE_VALUE_TO_PC,
NULL, //SYSTEM_RESTART,
NULL //BPM_DATA_READY = 0xFF,
};

const UINT8 AmpGain[] =
{
Gain2,
Gain3,
Gain4,
Gain4half,
Gain6,
Gain6half,
Gain8half,
Gain9,
Gain13,
Gain17
};

/*****
* Global Variables
*****/
/*****
* Global Functions Prototypes
*****/
void TestApp_Init(void);

```

```

/*****
 * Local Variables
 *****/
#ifdef _MC9S08JS16_H
#pragma DATA_SEG APP_DATA
#endif
/* Virtual COM Application start Init Flag */
static volatile boolean start_app = FALSE;
/* Virtual COM Application Carrier Activate Flag */
static volatile boolean start_transactions = FALSE;
/* Receive Buffer */
static uint_8 g_curr_rcv_buf[DATA_BUFF_SIZE];
/* Send Buffer */
static uint_8 g_curr_send_buf[DATA_BUFF_SIZE];
/* Receive Data Size */
static uint_8 g_rcv_size;
/* Send Data Size */
static uint_8 g_send_size;

static UINT16 PacketIdNumber = 0;
static UINT8 ActualMeasurement;

static UINT8 u8Gain_index = 7;
static UINT16 u16Debounce = 0;

#ifdef SEND_SINE_WAVE
static TIMER_OBJECT TimerSendDummyData;
static UINT8 TimerSendDummyDataIndex;
#endif
/*****
 * Local Functions
 *****/

/*****
 *
 * @name TestApp_Init
 *
 * @brief This function is the entry for the Virtual COM Loopback app
 *
 * @param None
 *
 * @return None
 *****/
*****
 * This function starts the Virtual COM Loopback application
 *****/
*****

void TestApp_Init(void)
{
    uint_8 error;

    /*****
    Initialize things for Ekg
    *****/
    DAC12_Vin_SWtrig();//set DAC to ouput 1.6 base from 3.3v reference
    opamp1_gp_mode();
    opamp2_noninverting_mode(Gain9);
    TRIAMP1C0_HighMode();
    TRIAMP2C0_HighMode();

    PTCDD_PTCDD6 = 0;
    PTCPE_PTCPE6 = 1;
    PTEPE_PTEPE4 = 1; //input pin on sw4

    PTED_PTED7 = 0; //active low for medical port enabling
    PTEDD_PTEDD7 = 1; //pin as output

```

```

//indicator LEDs initialization
PTFD_PTFD1 = 0;
PTFD_PTFD2 = 0;
PTFDD_PTFDD1 = 1;
PTFDD_PTFDD2 = 1;

EcgDsc_Init();
    Ecg_Init();

#ifdef SEND_SINE_WAVE
    TimerSendDummyData.msCount = 64;
        TimerSendDummyData.pfnTimerCallback = TimerSendDummyData_Event;
#endif
/*****/

g_rcv_size = 0;
g_send_size = 0;
DisableInterrupts;
#if (defined _MCF51MM256_H) || (defined _MCF51JE256_H)
    usb_int_dis();
#endif
/* Initialize the USB interface */
error = USB_Class_CDC_Init(CONTROLLER_ID,USB_App_Callback,
    NULL,USB_Notify_Callback);
if(error != USB_OK)
{
    /* Error initializing USB-CDC Class */
    return;
}
SCI1_Init();
EnableInterrupts;
    #if (defined _MCF51MM256_H) || (defined _MCF51JE256_H)
        usb_int_en();
    #endif
}

/*****
 *
 * @name    TestApp_Task
 *
 * @brief   Application task function. It is called from the main loop
 *
 * @param   None
 *
 * @return  None
 *
 *****/
*****
* Application task function. It is called from the main loop
*****/
void TestApp_Task(void)
{
    /* call the periodic task function */
    USB_Class_CDC_Periodic_Task();

    EcgDsc_PeriodicTask();

    Ecg_PeriodicTask();

    /* check whether enumeration is complete or not */
    if((start_app==TRUE)// && (start_transactions==TRUE))
    {
        Virtual_Com_Recv_Serial_Data();
        Virtual_Com_Send_Serial_Data();
    }
}

```



```

        /* check if SW4 is press to reduce the gain */
        if(!PTED_PTED4 & !u16Debounce)
        {
            if (u8Gain_index) u8Gain_index--;
            PTFD_PTFD2 = 1;
            opamp2_noninverting_mode(AmpGain[u8Gain_index]);
            u16Debounce = DEBOUNCE_TIME;
        }

        /* check if SW2 is press to increase the gain */
        if(!PTCD_PTCD6 & !u16Debounce)
        {
            if (u8Gain_index<9) u8Gain_index++;
            PTFD_PTFD1 = 1;
            opamp2_noninverting_mode(AmpGain[u8Gain_index]);
            u16Debounce = DEBOUNCE_TIME;
        }

        /* check if SW2 and SW4 are released */
        if(PTED_PTED4 & PTCD_PTCD6 & u16Debounce)
        {
            if (u8Gain_index==0) PTFD_PTFD1 = 0;
            else if (u8Gain_index==9) PTFD_PTFD2 = 0;
            else
            {
                PTFD_PTFD1 = 0;
                PTFD_PTFD2 = 0;
            }
        }

        if (u16Debounce) u16Debounce--;
    }

    /*****
    *
    * @name    Virtual_Com_Recv_Serial_Data
    *
    * @brief   Implements Loopback COM Port
    *
    * @param   None
    *
    * @return  None
    *
    *****/
    * Receives data from USB Host and process it
    *****/
    static void Virtual_Com_Recv_Serial_Data(void)
    {
        if(g_recv_size)
        {
            //there is new data
            if (g_curr_recv_buf[PACKET_TYPE] == REQ)
            {
                if (ExecuteCommandReq[g_curr_recv_buf[COMMAND_OPCODE]] != NULL)
                {
                    //check if OPCCODE is in a valid range
                    if ((g_curr_recv_buf[COMMAND_OPCODE] <=
LAST_COMMAND))
                    {
                        ExecuteCommandReq[g_curr_recv_buf[COMMAND_OPCODE]]();
                    }
                }
            }
            //else
            //{

```

```

        //packet type is not a request
        //}
        g_recv_size = 0;
    }
    return;
}

/*****
 *
 * @name    Virtual_Com_Send_Serial_Data
 *
 * @brief   Send serial data in g_curr_send_buf to USB port
 *
 * @param   None
 *
 * @return  None
 *
 *****/
* Send data to USB Host
*****/
static void Virtual_Com_Send_Serial_Data(void)
{
    uint_8 status = 0;
    if(g_send_size)
    {
        /* Send Data to USB Host*/
        uint_8 size = g_send_size;
        g_send_size = 0;

        if(g_send_size > DATA_BUFF_SIZE)
        {
            asm (NOP);
        }

        status = USB_Class_CDC_Interface_DIC_Send_Data(CONTROLLER_ID, &g_curr_send_buf[0], size);
        SCISendData(&g_curr_send_buf[0], size);
        if(status != USB_OK)
        {
            /* Send Data Error Handling Code goes here */
        }
    }
}

/*****
 *
 * @name    USB_App_Callback
 *
 * @brief   This function handles Class callback
 *
 * @param   controller_ID : Controller ID
 * @param   event_type    : Value of the event
 * @param   val           : gives the configuration value
 *
 * @return  None
 *
 *****/
* This function is called from the class layer whenever reset occurs or enum
* is complete. After the enum is complete this function sets a variable so
* that the application can start.
* This function also receives DATA Send and RECEIVED Events
*****/
static void USB_App_Callback (
    uint_8 controller_ID, /* [IN] Controller ID */
    uint_8 event_type,    /* [IN] value of the event */
    void* val             /* [IN] gives the configuration value */

```

```

)
{
    UNUSED (controller_ID)
    UNUSED (val)
    if(event_type == USB_APP_BUS_RESET)
    {
        start_app=FALSE;
    }
    else if(event_type == USB_APP_ENUM_COMPLETE)
    {
        start_app=TRUE;
    }
    else if((event_type == USB_APP_DATA_RECEIVED))
    //&&      (start_transactions == TRUE))
    {
        /* Copy Received Data buffer to Application Buffer */
        USB_PACKET_SIZE BytesToBeCopied;
        APP_DATA_STRUCT* dp_rcv = (APP_DATA_STRUCT*)val;
        uint_8 index;
        BytesToBeCopied = (USB_PACKET_SIZE)((dp_rcv->data_size > DATA_BUFF_SIZE) ?
            DATA_BUFF_SIZE:dp_rcv->data_size);
        for(index = 0; index<BytesToBeCopied ; index++)
        {
            g_curr_rcv_buf[index]= dp_rcv->data_ptr[index];
        }
        g_rcv_size = index;
        (void)USB_Class_CDC_Interface_DIC_Recv_Data(CONTROLLER_ID, NULL, 0);
    }
    else if((event_type == USB_APP_SEND_COMPLETE)&&
        (start_transactions == TRUE))
    {
        /* Previous Send is complete. Queue next receive */
        (void)USB_Class_CDC_Interface_DIC_Recv_Data(CONTROLLER_ID, NULL, 0);
    }

    return;
}

/*****
 *
 * @name      USB_Notify_Callback
 *
 * @brief     This function handles PSTN Sub Class callbacks
 *
 * @param    controller_ID  : Controller ID
 * @param    event_type     : PSTN Event Type
 * @param    val            : gives the configuration value
 *
 * @return    None
 *
 *****/
/* This function handles USB_APP_CDC_CARRIER_ACTIVATED and
 * USB_APP_CDC_CARRIER_DEACTIVATED PSTN Events
 *****/

static void USB_Notify_Callback (
    uint_8 controller_ID, /* [IN] Controller ID */
    uint_8 event_type,   /* [IN] PSTN Event Type */
    void* val            /* [IN] gives the configuration value */
)
{
    UNUSED (controller_ID)
    UNUSED (val)
    if(start_app == TRUE)
    {
        if(event_type == USB_APP_CDC_CARRIER_ACTIVATED)

```

```

    {
        start_transactions = TRUE;
    }
    else if(event_type == USB_APP_CDC_CARRIER_DEACTIVATED)
    {
        start_transactions = FALSE;
    }
}
return;
}

void EcgDiagnosticModeStartMeasurementReq(void)
{
    if (ActualMeasurement == NO_MEASUREMENT)
    {
        //execute command and send confirm
        g_curr_send_buf[g_send_size++] = CFM;
        g_curr_send_buf[g_send_size++] =
        ECG_DIAGNOSTIC_MODE_START_MEASUREMENT;
        g_curr_send_buf[g_send_size++] = 1; //data bytes

#ifdef SEND_SINE_WAVE
        g_curr_send_buf[g_send_size++] = ERROR_OK;
        TimerSendDummyDataIndex = AddTimerQ(&TimerSendDummyData);
#else
    #ifdef ECG_DSC
        if ( (EcgDsc_DiagnosticModeStartMeasurement() == TRUE) &&
            (Ecg_DiagnosticModeStartMeasurement() ==TRUE) )
        {
            g_curr_send_buf[g_send_size++] = ERROR_OK;
        }
        else
        {
            g_curr_send_buf[g_send_size++] = ERROR_BUSY;
        }
    #else
        if (Ecg_DiagnosticModeStartMeasurement() == TRUE)
        {
            g_curr_send_buf[g_send_size++] = ERROR_OK;
        }
        else
        {
            g_curr_send_buf[g_send_size++] = ERROR_BUSY;
        }
    #endif
#endif

    Virtual_Com_Send_Serial_Data();

    ActualMeasurement = ECG_MEASUREMENT;
}

}

void EcgDiagnosticModeStopMeasurementReq(void)
{
    if (ActualMeasurement == ECG_MEASUREMENT)
    {
#ifdef SEND_SINE_WAVE
        RemoveTimerQ(TimerSendDummyDataIndex);
#else
    #ifdef ECG_DSC
        EcgDsc_DiagnosticModeStopMeasurement();

```

```

        Ecg_DiagnosticModeStopMeasurement();
    #else
        Ecg_DiagnosticModeStopMeasurement();
    #endif
#endif

    g_curr_send_buf[g_send_size++] = CFM;
    g_curr_send_buf[g_send_size++] = ECG_DIAGNOSTIC_MODE_STOP_MEASUREMENT;
    g_curr_send_buf[g_send_size++] = 0; //data bytes

    Virtual_Com_Send_Serial_Data();
    ActualMeasurement = NO_MEASUREMENT;
}
}
void EcgDiagnosticModeNewDataReadyInd(void)
{
    static UINT16 IdNumber = 0;

#ifdef ECG_DSC
    if (ActualMeasurement == ECG_MEASUREMENT)
    {
        UINT8 i=0;

        //Send indication
        g_curr_send_buf[g_send_size++] = IND;
        g_curr_send_buf[g_send_size++] = ECG_DIAGNOSTIC_MODE_NEW_DATA_READY;
        g_curr_send_buf[g_send_size++] = DATA_LENGTH_FROM_DSC-1;
        //data from DSC (64) + packetID

        g_curr_send_buf[g_send_size++] = (UINT8) (IdNumber >> 8);
        g_curr_send_buf[g_send_size++] = (UINT8) (IdNumber & 0x00FF);

        IdNumber++;

        //copy data from DSC to outbuffer
        i = DATA_START_POSITION;

        while (i<DATA_END_POSITION)
        {
            g_curr_send_buf[g_send_size++] = DataFromDsc[i+1]; //swap data bytes
            g_curr_send_buf[g_send_size++] = DataFromDsc[i];
            i+=2;
        }

        g_curr_send_buf[g_send_size++] = EcgDsc_HeartRate;

        //send data
        Virtual_Com_Send_Serial_Data();
    }
#else
    if (ActualMeasurement == ECG_MEASUREMENT)
    {
        UINT8 i=0;

        //Send indication
        g_curr_send_buf[g_send_size++] = IND;
        g_curr_send_buf[g_send_size++] = ECG_DIAGNOSTIC_MODE_NEW_DATA_READY;
        g_curr_send_buf[g_send_size++] = ECG_DATA_BUFFER_LENGTH + 2 + 1;
        //data from ECG + packetID + ECG_HeartRate

        g_curr_send_buf[g_send_size++] = (UINT8) (IdNumber >> 8);
        //add packetId for error handling
        g_curr_send_buf[g_send_size++] = (UINT8) (IdNumber & 0x00FF);

        IdNumber++;
    }
#endif
}

```

```

//copy data from DSC to outbuffer
i = 0;

while (i<ECG_DATA_BUFFER_LENGTH)
{
    g_curr_send_buf[g_send_size++] = EcgDataBuffer[i++]; //copy ECG data to
OutBuffer
    g_curr_send_buf[g_send_size++] = EcgDataBuffer[i++];
}

g_curr_send_buf[g_send_size++] = Ecg_HeartRate;
//send data
Virtual_Com_Send_Serial_Data();
}
#endif
}

#ifdef SEND_SINE_WAVE
//this is the timer event for sending a dummy sine wave

void TimerSendDummyData_Event(void)
{
    static const UINT16 SinX[] =
    {
        1250, 1349, 1448, 1545, 1640, 1733, 1823, 1910, 1992, 2069, 2142, 2208, 2269, 2323, 2371,
        2411, 2444, 2470, 2488, 2498, 2500, 2494, 2481, 2459, 2430, 2394, 2350, 2300, 2243, 2179, 2110, 2036, 1956,
        1872, 1784, 1693, 1598, 1502, 1404, 1305, 1206, 1107, 1009, 912, 818, 726, 638, 553, 473, 398, 328, 264, 206,
        155, 111, 73, 44, 21, 7, 0, 2, 11, 28, 53, 85, 124, 171, 225, 284, 350, 422, 499, 581, 667, 756, 849, 944, 1041,
        1140, 1239
    };

#define SIN_X_LAST_ELEMENT    79
static const UINT8 EcgBufferSize = 64;
static const UINT8 SinXTimerPeriod = 64;

if (ActualMeasurement == ECG_MEASUREMENT)
{
    static UINT8 SinXArrayActualElement = 0;

    //Send indication
    g_curr_send_buf[g_send_size++] = IND;
    g_curr_send_buf[g_send_size++] = ECG_DIAGNOSTIC_MODE_NEW_DATA_READY;
    g_curr_send_buf[g_send_size++] = EcgBufferSize + 2;
//data bytes

    g_curr_send_buf[g_send_size++] = (UINT8) (PacketIdNumber >> 8);
    g_curr_send_buf[g_send_size++] = (UINT8) (PacketIdNumber & 0x00FF);

    PacketIdNumber++;

    while (g_send_size < (EcgBufferSize + DATA_PACKET + 2))
    {
        g_curr_send_buf[g_send_size++] = SinX[SinXArrayActualElement] >> 8;

        g_curr_send_buf[g_send_size++] = SinX[SinXArrayActualElement] & 0x00FF;

        if (SinXArrayActualElement == SIN_X_LAST_ELEMENT)
        {
            SinXArrayActualElement = 0;
        }
        else

```

```

        {
            SinXArrayActualElement++;
        }
    }

    Virtual_Com_Send_Serial_Data();

    TimerSendDummyDataIndex = AddTimerQ(&TimerSendDummyData);
}

}
#endif

void SWdelay(void)
{
    UINT16 debounce= 50000;
    while (debounce) debounce--;
}

/* add SCI1 channel to transport data */

static void SCII_Init(void)
{
    SCI1C1 = 0x00;
    SCI1C2 = 0x00;
    SCI1C3 = 0x00;
    SCI1BD = 24000000UL/9600/16; /* 9600 bps*/
}

static void SCISendData(const byte SendData[],byte ssize)
{
    byte i;

    SCI1C2_TE = 1;

    for(i = 0;i < ssize; i++)
    {
        while(SCI1S1_TDRE == 0)
        {
            ;
        }
        SCI1D = SendData[i];
    }
    SCI1C2_TE = 0;
}

/* EOF */

```

Appendix H: Schematics Diagram of Freescale's Medical Development Kit