AC 2012-5251: NORMAL AND ABNORMAL EKGS AND HEART SOUNDS

Dr. Mohammad Rafiq Muqri, DeVry University, Pomona
Prof. Shih Ek Chng, DeVry University, Pomona

EET Professor, DeVry University

Mr. Furqan Muqri, University of Texas

Southwestern Medical Center at Dallas

Ms. Aceela Muqri, VCU School of Medicine

©American Society for Engineering Education, 2012
Normal and Abnormal EKGs and Heart Sounds:
Development of a diagnostic tool

Abstract

Electrocardiogram (EKG, or ECG) is a transthoracic interpretation of the electrical activity of the heart externally recorded by skin electrodes and captured over time. It can detect hypertrophy, heart block, fibrillation, electrolyte abnormalities, rhythm problems and other cardiac conditions. Heart murmurs are abnormal sounds during your heart beat cycle and can be heard with stethoscope. Mastery and expertise in clinical EKG (Electrocardiogram) interpretation is one of the most desirable clinical skills in bioengineering and medicine. It can probably only be achieved if one acquires a well rounded experience in understanding the pathophysiology, clinical status of the patient, and correlation with specific EKG findings.

This paper presents the development and application of an innovative medical diagnostic tool for EKG monitoring, which can be used by engineering technology, health care, and medical students for quick health screening and cardiologic health assessment. Students progressively learn to monitor and interpret the conventional noninvasive electrocardiography by leveraging the power of java’s graphical user interface and data structures.

The paper explains the laboratory setup of a basic 3-lead EKG monitoring station using modern data acquisition tool and software for EKG feature extraction. Students will begin their analysis by looking at rate, rhythm, axis, hypertrophy, and infarction and correlate the characteristic appearance on the EKG with existing conditions, certain pathology, and drug or electrolyte effects. A diagnostic tool using Java and Objective-C programming is then developed. The graphical user interface will be used to correlate specific EKG findings with pathology of the heart and selectively demonstrate the diagnosis of certain cardiologic health screening and assessment parameters.

This learning and teaching module can be instrumental in progressive learning for BMET and EET students, by enhancing their understanding of clinical EKG instrumentation, parameters extraction and their relationship with differential diagnosis of the patients. It will give them a form of intellectual development, enhancing their skill-set, and challenging their creativity.

This paper thereby serves as an interesting way to expose engineering technology, health care and medical students to this fascinating topic and gives them exposure to EKG instrumentation, and java programming while having fun learning the EKG interpretation, algorithms, early heart monitoring, diagnosis and intervention.

Instrument Setup

A typical experimental setup for EKG signal analysis used in our laboratory is described next. A high performance general-purpose biomedical instrumentation amplifier and a data acquisition system is required for pre and post processing of EKG signal. This amplifier with high input
impedance and high CMRR allow low noise recording of bio-potential ECG signal. It is usually equipped with High-Pass and Low-Pass filter to remove signal pickup from body movement or breathing and noise riding on the baseline of input signal. For testing the instrumentation amplifier and data acquisition system purposes, an Agilent function generator set to output nominal amplitude and frequency cardiac signal is used.

![Cardiac Signal Generation](image1.png)

**Nominal Cardiac Signal Generation**

![Cardiac Signal GPIB](image2.png)

**Nominal Cardiac Signal captured through GPIB**

The preprocessing of raw data is done by hardware build within the amplifier unit. In actual acquisition of ECG signal, a basic three-lead cable with isolation amplifier is needed to allow safe surface recording of ECG signal from human subjects. Further preprocessing of ECG signal to reduce the effect of noise can be done with Biomedical Workbench software from National Instrument.
The lower part of the diagram shows how NI ElVIS II Workstation is used for data acquisition with a laptop or desktop PC running on Biomedical Workbench software from National Instrument. The Biomedical Startup Kit 3.0 version can be downloaded from NI. (https://decibel.ni.com/content/docs/DOC-12646). It includes an executable and a source code version. Executable version is used in this paper.
EKG Feature Extraction

If different data acquisition system is used, NI Biomedical Workbench can convert various biosignal file format to TDMS file. Typical biosignal file formats include LVM, ABF, MIT-BIH Database, HL7, ACQ, EMG, aECG and EDF. If NI ELVIS II is used as described in this paper, EKG signal is acquired through the analog input channels by running the Bio-signal Logger. The Online Bio-signal Noise Reduction Data Logger is a pre-processing application that uses adaptive filters to further reduce the effect of noise. Extraction of various features including Heart Rate, PR Interval, QT Interval, QRS Width and Amplitude Statistics are accomplished through the ECG Feature Extractor.
Clinical Parameters

Cardiac murmurs can be divided into three categories based on where they occur in the cardiac cycle, systolic, diastolic and continuous. Short, quiet systolic murmurs are generally benign. Long systolic murmurs, diastolic murmurs and continuous murmurs are generally pathologic. (Two continuous murmurs that are benign are mammary soufflé and cervical venous hum.)

The most obvious of the heart sounds are the first and second sounds, or S1 and S2, which demarcate systole from diastole. S1 is the sound which marks the approximate beginning of systole, and is created when the increase in intra-ventricular pressure during contraction exceeds the pressure within the atria, causing a sudden closing of the tricuspid and mitral, or AV valves. The ventricles continue to contract throughout systole, forcing blood through the aortic and pulmonary, or semi-lunar valves. At the end of systole, the ventricles begin to relax, the pressures within the heart become less than that in the aorta and pulmonary artery, and a brief back flow of blood causes the semi-lunar valves to snap shut, producing S2.

A modern stethoscope consists of two earpieces connected by tubing to a chest piece which usually has both diaphragm and bell attachments. Earpieces should be angled forwards to match the direction of the physician's external auditory meati. The bell is used to hear low-pitched sounds. Use for mid-diastolic murmur of mitral stenosis or S3 in heart failure. The diaphragm, by filtering out low-pitched sounds, highlights high-pitched sounds. Use it for analyzing the second heart sound, ejection and mid-systolic clicks and for the soft but high-pitched early diastolic murmur of aortic regurgitation. Examination of Heart sounds and murmurs are utilized in conjunction with EKGs by physicians for clinical diagnosis of patients and cardiologic assessment.

The standard 12-lead electrocardiogram is a representation of the heart's electrical activity recorded from electrodes on the body surface. You will notice that the 12-lead ECG provides
spatial information about the heart's electrical activity in three approximately orthogonal directions:
Right - Left
Superior - Inferior
Anterior - Posterior

Each of the twelve leads (the “limb leads” and the “precordial” leads) as indicated below, represents a particular orientation in space (RA = right arm; LA = left arm, LF = left foot):
The six standard limb leads (the bipolar and the augmented) depict cardiac electrical events from six angles in the vertical plane, while the six precordial leads (the “chest leads”) depict electrical events in the horizontal plane.

Bipolar limb leads (frontal plane):
Lead I: RA (-) to LA (+) (Right Left, or lateral).
Lead II: RA (-) to LF (+) (Superior Inferior).
Lead III: LA (-) to LF (+) (Superior Inferior)

Augmented unipolar limb leads (frontal plane):
Lead aVR: RA (+) to [LA & LF] (-) (Rightward).
Lead aVL: LA (+) to [RA & LF] (-) (Leftward).
Lead aVF: LF (+) to [RA & LA] (-) (Inferior).

Unipolar (+) chest leads (horizontal plane) comprise of the following:
Leads V1, V2, V3: (Posterior Anterior)
Leads V4, V5, V6: (Right Left, or lateral)
The placement of chest Leads V1 thru V6 is listed next.
V1: right 4th intercostal space
V2: left 4th intercostal space
V3: halfway between V2 and V4
V4: left 5th intercostal space, mid-clavicular line
V5: left 5th intercostal space, anterior-axillary line
V6: horizontal to V5, mid-axillary line
There is a systemic approach to ECG interpretation which is typically based on nine parameters: Rate, Rhythm, P wave morphology, PR interval, QRS complex (Axis, Voltage, Duration, morphology), QT interval, ST segment, T wave, and U wave. The list of ECG diagnoses was derived from a recently published statement of the American College of Cardiology/American Heart Association (ACC/AHA) Committee to Develop a Clinical Competence Statement on Electrocardiography and Ambulatory Electrocardiography.

Let us talk about ECG waves and intervals. The P wave corresponds to atrial systole, the sequential activation (depolarization) of the right and left atria. The PR interval represents the time interval from onset of atrial depolarization (P wave) to onset of ventricular depolarization.
QRS complex). The PP interval corresponds to duration of atrial cycle (an indicator of atrial rate).

The QRS complex corresponds to ventricular systole, the right and left ventricular depolarization. The T wave represents the ventricular repolarization. The cause of the inconsistently present U wave is wave is controversial. The QT interval corresponds to duration of ventricular muscle depolarization. The RR interval represents the duration of ventricular cardiac cycle (an indicator of ventricular rate).

The six standard limb leads are depicted below:

It is beyond the scope of this paper to provide a comprehensive presentation of basic electrocardiography, vectorcardiography, and pathophysiology. Discussion of electrophysiologic mechanisms underlying changes in the surface ECG are minimized in the following discussion and it is assumed that the audience has a basic understanding of ECG vector analysis (calculating mean axes).

The java algorithms used place emphasis upon the differential diagnosis based on the criteria of rhythm, P wave, PR interval, QRS complex, ST, T and U wave changes. We therefore introduced the students to the basics of the ECG interpretation and successfully attempted to incorporate some of these algorithms using the java data structures and advanced GUI features.

Development of User Interface

Objective-C is an elegant reflective language (superset of the ANSI version of C) with added Smalltalk style object oriented concepts which optimizes Model-view-Controller (MVC) software architecture. Just like the C code, we define header files and source files so as to separate public declarations from the code implementation details.

The java language is primarily used when developing Android applications (though it does not actually get compiled to byte code). Both the Android and the iPhone also provide a way of declaring user preferences in XML. Both platforms provide a default user interface for editing a person’s respective preferences.

Arrays in Java are data structures consisting of related data items of the same type. Arrays can be considered as fixed-length entities, although at times an array reference with proper syntax may be reassigned to a new array of a different length. On the other hand we have dynamic data structures, such as stacks, queues, trees and lists that can shrink or grow as programs execute.

An Android user interface is composed of hierarchies of objects called Views. A View is a drawable object used as an element in your UI layout, such as a buttons, images, text labels etc. In contrast to Apple’s iPhone, Android’s UI builder is fairly inefficient and probably the only way to get around is to code all of the UIs directly into the XML. However I noticed that integrated JavaDoc and content seem to truly facilitate the path to learn an unfamiliar API. Android also includes a set of C/C++ libraries used by various components of the Android system through application framework. One of the key benefit of Android is being open source, but it also means that when looking at sample code, it is not all organized in the same way.
The main point is that every great iOS iPhone or Android app starts with a brilliant idea, but translating that idea into actions requires planning. So before you can write any code, you have to take the time to explore the possible techniques and technologies. The core infrastructure of an iOS app is built from objects in the UIKit framework. There are some resources that must be present in all iOS apps. Most apps include images, sounds, and other types of resources for presenting the app’s content, but the App store may also require some specific resources. You may like to refer to iOS App Programming Guide for further details.

Objective-C also provides a dot syntax for accessing accessor methods which are used to get and set the state of objects. Why dot notation? Quite simple, Objective-C offers synergy with the syntax for C structs which look a lot like objects with @property except for two differences. First, we can’t send messages to C structs, because they have no methods. Second, C structs are almost never allocated in the heap, since we don’t use pointers to access them. Let us demonstrate this with an example:

typedef struct
{    float x;
    float y;
} CGPoint  //Makes our C struct seem like an object with @property, but you can’t send message
@interface Torpedo
@property CGPoint position;  // @property access looks just like C struct member access
@end
@interface Ship : Vessel
@property float height;
@property CGPoint center;
-(BOOL) getsHitByTorpedo : (Torpedo *) torpedo;  // Instance method begins with dash
//Returns whether the passed torpedo would hit the receiving ship
@end
@implementation Ship
@synthesize height, width, center ;
- (BOOL) getsHitByTorpedo : (Torpedo *) torpedo ;
{
    float LeftEdge  =  self. center. x -  self. Width /2;
    float RightEdge  =    . . . ;
    return ((torpedo. position. x  >= leftEdge ) && (torpedo. position. x <= rightEdge ) &&
    (torpedo. position. y  >= topEdge ) && (torpedo. position. y <= topEdge )
}
@end
Cocoa is an application environment for both the MAC OS X operating system and iOS, the operating system used on Multi-Touch devices such as iPhone, iPad, and iPad touch. It consists of a suite of object oriented libraries, a runtime system, and an integrated development system. In other words, Cocoa’s basic application framework provides the infrastructure for event driven behavior and for management of applications, and windows. It also offers for your applications, a rich collection of ready-made user-interface objects which are available in the interface builder’s library. Is it not nice to simply drag an object from the library onto the surface of your interface, configure its attributes, and connect it to other objects?
Retain and Release. An object knows how many times it is referenced, thanks to its retain count.
In order to increase the retain count of an object, all you have to do is to send the object a retain message, and to decrease the retain count of an object, you send the object a release message:

\[ \text{anObject retain}; \]
\[ \text{anObject release}; \]

Here is just a sample of Cocoa’s user-interface objects:
- Navigation bars
- Page controls
- Windows
- Text fields
- Labels
- Image views
- Date pickers
- Segmented controls
- Table views
- Progress indicators
- Sliders
- Buttons

Cocoa’s software is developed primarily by using two developer applications, namely Xcode and interface builder, although it is possible to develop Cocoa applications without using these applications at all. Beginning with Xcode 3.1 and the introduction of iOS, when we create a software project, we must choose a platform SDK. The platform SDK contains everything that is required for developing software for a given platform and operating-system release. The SDK for iOS (like MAC OS X SDK) consists of frameworks, libraries, header files and system tools but includes a platform-specific compiler and other tools. There is also a separate SDK for iOS Simulator. All SDKs include build settings and project templates appropriate to their platform.

Xcode is the engine that powers Apple’s integrated development environment (IDE) for Mac OS X and iOS. It allows us to create and manage projects; and specify platforms, target requirements, dependencies, and build configurations. Xcode allows us to write source code in editors with features such as syntax coloring, automatic indenting, navigating and searching through the components of a project, including header files and documentation. It allows to build the project, debug it locally, in iOS Simulator, or remotely in a graphical source-level debugger.

Open Development Process

Some people might want to avoid having to learn Xcode to get started programming with Objective-C using the terminal application, at least in the beginning. For them, the first step is to start the Terminal application, which is located in the Application folder, stored under Utilities on the Mac computer. The Objective-C language enables the user to define a set of ordered data items known as an array. The purpose of this program (which illustrates the use of character arrays) is to show how general Objective C programming techniques can be easily introduced at an early stage to engineering technology students.

First Program in Objective-C

Let us begin by creating a directory to store your program examples. Make a directory called Aprograms. Next in this directory, using vi editor, type the first program called main1.m and follow these steps:
mkdir Aprograms
$ cd APrograms
$ vi main1.m

Type <i> to go to INSERT mode, then enter the code snippet as depicted below. After, you finished typing, you want to save the main1.m program, so now press the <Esc> key to go to control mode. Typing :wq takes you back to bash shell command prompt ($).

```
import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
  int i, j, k;
  char word1[] = { 'W', 'E', 'L', 'C', 'O', 'M', 'E' };
  char word2[] = { 'T', 'O' };
  char word3[] = { 'A', 'S', 'K', 'M', 'E' };

  for (i = 0 ; i < 7 ; ++i)
    NSLog (@"%c", word1[i]);
  printf("\n");
  for (j = 0 ; j < 2 ; ++j)
    NSLog (@"%c", word2[j]);
  printf("\n");
  for (k = 0 ; k < 4 ; ++k)
    NSLog (@"%c", word3[k]);
  [pool drain];
  return 0;
}
```

The first line of your Objective-C program tells the compiler to locate and process a system file (include the information) into the program. Note that, the NS is used for Apple’s classes. It stands for NeXTStep. NeXTStep being the operating system Mac OS X was based on when Apple bought NeXT, Inc.

The first statement in our program reads
```
NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
```
It reserves space in memory for an autorelease pool. Here an alloc short for allocate message is sent to NSAutoReleasePool class requesting that a new instance be created. The init message then is sent to the newly created object to get it initialized.

The second last statement in our program reads
```
[pool drain];
```
It releases the allocated memory pool and the associated objects.

The command to compile and run main1.m are as follows:
```
$ gcc -framework Foundation main1.m -o pg1
$./pg1   lets you execute (run) your executable file name pg1
```
Now that you have seen the output of first program, using bash shell terminal Window utility:

W
E
L
C
O
M
E
T
O
A
S
E
E

Since developing, testing, and integrating both Android and iPhone applications are huge topic, we will focus mainly on EKG interpretation by leveraging the power of Java graphical user interface (GUI) and its data structures. So as to not be overly verbose in our descriptions of abnormal and normal EKGS and heart sounds, we shall simply only tell you our success story.

The next program demonstrates the power of java’s advanced GUIs and Graphics. In this program JLPViewer.java, we create an array of strings consisting of the names of the images. We also want the user to select only one EKG (item) image at a time from pJL (pictureJList), so we list pJL to single selection mode by using the method setSelectionMode together with the constant ListSelectionModel.SINGLE_SELECTION.
When the user selects an image from pJL, the program displays the corresponding EKG image using the JLabel object dPicJL (displayPictureJList). We use the class ImageIcon and create an array of images which include ekg1, ekg2, ekg3, ekg4, ekg5 etc.

There are five items in the JList. When the program executes, it displays only three of these image items in the list at a time. Therefore, we want to attach a vertical scroll bar to the JList object pJL, so that user can scroll to select an EKG not currently shown in the list. We set the pane layout to null and specify the size and location of the GUI components. These statements are provided in the complete program listing given next.

`/* @(#)JLPViewer.java */`  
* JLPViewer application  
*  
* @author  
* @version 1.00 2012/01/11  
*  
* import java.awt.*;  
* import java.awt.event.*;  
* import javax.swing.*;  
* import javax.swing.event.*;  
*  
* public class JLPViewer extends JFrame implements ListSelectionListener {  
*  
* private String[] pNames = {"ekg 1", "ekg 2", "ekg 3", "ekg 4","ekg 5" } ;  
* private ImageIcon[] pics = {new ImageIcon("ekg1.gif"),new ImageIcon("ekg2.gif"),new ImageIcon("ekg3.gif"),new ImageIcon("ekg4.gif"),new ImageIcon("ekg5.gif")};  
* private BorderLayout lBL ;  
* private JList pJL ;  
* private JScrollPane sJS ;  
* private JLabel promptJL ;  
* private JLabel dPicJL ;  
* private JLabel infoJL ;  
* public JLPViewer( )  
* {  
*   super ("EKG Viewer");  
*   Container pane = getContentPane();  
*   pane.setLayout(null);  
*   promptJL = new JLabel(" Select an ElectroCardiogram",SwingConstants.CENTER);  
*   promptJL.setSize(350,10);  
*   promptJL.setLocation(10,10);  
*   pane.add(promptJL);  
*   pJL = new JList(pNames);  
*   pJL.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);  
*   pJL.addListSelectionListener(this);  
*   sJS = new JScrollPane(pJL);  
*   pane.add(pJL);  
*   pane.add(sJS);  
* }  
*  
* */
sJS.setSize(350, 60);
sJS.setLocation(10, 20);
pane.add(sJS);
dPicJL = new JLabel(pics[4]);
dPicJL.setSize(350, 350);
dPicJL.setLocation(10, 50);
pane.add(dPicJL);
infoJL = new JLabel(pNames[4], SwingConstants.CENTER);
infoJL.setSize(350, 20);
infoJL.setLocation(10, 380);
pane.add(infoJL);
setSize(480, 540);
setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);

public static void main(String args[]) {
    JLPViewer pV = new JLPViewer();
}
public void valueChanged(ListSelectionEvent e) {
    dPicJL.setIcon(pics[pJL.getSelectedIndex()]);
    infoJL.setText(pNames[pJL.getSelectedIndex()]);
    repaint();
}
}
Here is another snapshot of the Java program output, which serves the purpose of EKG parameters interpretation and demonstrates the user-friendly advanced GUI capabilities of this fascinating EKG diagnostic tool.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.text.DecimalFormat;

public class EkgInterpret extends JFrame implements ActionListener {

    static DecimalFormat twoDigits = new DecimalFormat("0.00");
    static String[] yourChoicesItems = {
            "Atrial Rate: 71",
            "Ventricular rate: 71",
            "Rhythm: Sinus rhythm with second degree SA type I",
            "PR interval:(msec) 170",
            "P wave: Normal",
            "QRS Complex Axis: -45",
            "QRS Duration:(msec) 90",
            "Morphology: Q waves in I, III, and aVF, and V1 to V5",
            "STsegment: Hyperacute ST segment elevation in V1 to V6",
            "T wave: Inverted in I, aVL and V2 to V6",
            "QT interval: 320",
            "Diagnosis: Sinus rhythm with second degree SA block type I",
            "\tLAD, acute anterior and lateral MI, and probable old inferior MI"};
    static double[] yourChoicesParameters = {71, 71, 170, -45, 90, 320};

    private JList yourChoices;
    private JTextArea info;
    private Container pane;

    public EkgInterpret() {
        super("Welcome to EKG Interpretation Lab");
        pane = getContentPane();
        pane.setBackground(new Color(0, 200, 200));
        pane.setLayout(new BorderLayout(5, 5));
        JLabel yourChoicesJLabel = new JLabel("EKG DIAGNOSIS MENU");
        pane.add(yourChoicesJLabel, BorderLayout.NORTH);
        yourChoicesJLabel.setFont(new Font("Dialog", Font.BOLD, 20));
        yourChoices = new JList(yourChoicesItems);
        pane.add(new JScrollPane(yourChoices), BorderLayout.WEST);
        yourChoices.addActionListener(this);
    }

    public static void main(String[] args) {
        EkgInterpret e = new EkgInterpret();
    }
}
yourChoices.setFont(new Font("Courier",Font.BOLD,16));
info = new JTextArea();
pane.add(info,BorderLayout.EAST);
info.setFont(new Font("Courier",Font.PLAIN,14));
JButton button = new JButton("Interpretation Completed");
pane.add(button,BorderLayout.SOUTH);
button.addActionListener(this);
setSize(640, 500);
setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
private void displayInfo()
{
    int[] listArray = yourChoices.getSelectedIndices();
    // double Axis = -45.00;
    double LAD;
    double RAD;
    double AD = 0.5;
    info.setEditable(false);
    info.setText("\n");
    for(int index = 0; index < listArray.length; index++)
    { AD = AD + yourChoicesParameters[listArray[index]];
        info.append("EKG DIAGNOSIS MENU\n\n");
        info.append("------------- Welcome --------------\n\n");
        for(int index = 0; index < listArray.length; index++)
        { 
            info.append(yourChoicesItems[listArray[index]] + "\n");
       }
        info.append("\n");
        info.append("VOLTAGE:\n\t" + twoDigits.format(AD) + "\n");
        info.append("Thank you - Have a nice Day\n\n");
        yourChoices.clearSelection();
        repaint();
    }
    public void actionPerformed(ActionEvent event)
    {
        if (event.getActionCommand().equals("Interpretation Completed"))
            displayInfo();
    }
    public static void main(String[] args)
    {
        EkgInterpret eint = new EkgInterpret();
    }
}
In conclusion, it can be stated that with proper guidance, monitoring and diligent care, the engineering technology students can be exposed earlier to Java data structures, GUI interface, Cocoa framework, X-Code, iPhone or Android development platform. This will go a long way in motivating them, eliminating their fear, improving their understanding and enhancing their quality of education. With proper mentoring, capable tutelage, and guidance, these burgeoning and talented young students will not only learn Java data structures, objective-C and advanced java GUIs but also have fun applying this knowledge in EKG interpretation, early heart monitoring, and diagnosis for beginning smart phone apps development.

Bibliography

13. Skill Modules, Advanced Physical Diagnosis Learning and Teaching at the Bedside, Department of Medicine, University of Washington, http://depts.washington.edu/physdx.html
14. ZhijunGu, NI Biomedical Start up Kit 3.0, July 2010, https://decibel.ni.com/content/docs/DOC-12646