

Implementation of a Wearable Feedback System Monitoring the Activities of Upper-extremities

by

Dong Yang

B.Eng., Xi'an Jiao Tong University, 2001

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Engineering

in the

School of Engineering Science
Faculty of Applied Sciences

© **Dong Yang 2015**

SIMON FRASER UNIVERSITY

Spring 2015

All rights reserved.

However, in accordance with the *Copyright Act of Canada*, this work may be reproduced, without authorization, under the conditions for "Fair Dealing." Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

Approval

Name: Dong Yang
Degree: Master of Engineering
Title: *Implementation of a Wearable Feedback System
Monitoring the Activities of Upper-extremities*
Examining Committee: Chair: Dr. Ash Parameswaran, P.Eng.
Professor

Dr. Carlo Menon, P.Eng
Senior Supervisor
Associate Professor

Fabio Campi
Supervisor
Lecturer

Date Defended/Approved: April 21st, 2015

Partial Copyright Licence



The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files (“Work”) (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU’s own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU’s rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author’s written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author’s knowledge, infringe upon anyone’s copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013

Abstract

In the previous work done by ZG.Xiao and C.Menon, the novel idea of using a strap with 8 Force Sensing Resistor (FSR) sensors for monitoring activities of upper-extremities was proposed. The goal of my research is to implement such a system for a low-cost, low power embedded system, a band module, together with a hand-held user interface for rehabilitation related application. For hand gesture classification, the Linear Discriminate Analysis (LDA) method is used. The training and predicting can be done on either a band module or a hand-held user interface. Two system configurations are proposed: real-time band module data sampling and hand-held user interface data analysis, or real-time band module data sampling and analysis. On top of this, the LDA algorithm in C language used in our system has been profiled on an Intel Galileo Gen2 board in order to evaluate its performance on 32-bit embedded platforms for the next generation of our system.

Keywords: Force Sensing Resistor; Linear Discriminate Analysis; embedded system; rehabilitation; profiling

Acknowledgements

I would like to thank my senior supervisor, Professor Carlo Menon, for this amazing opportunity, valuable suggestions and being very patient and supportive of my work. Also I would like to thank my supervisor, Fabio Campi, for guidance and suggestions throughout my project especially during writing of this report. And I would like to thank Professor Parameswaran as the chair of my supervisory committee.

I then would like to express my appreciation to Lukas-Karim Merhi for his coordination of this project, ZhenGang Xiao for his help and support as the original researcher of this topic, Tanner Frison, Pawan Tejwani, Bryan Chong for their hardware/software/debugging effort on the prototypes and other team members in MENRVA group at SFU for their support of my project.

I would also like to thank Renee Anne McCallum for her excellent proof reading on this report. And I would like to thank Catherine Louie for her advices on formatting this report.

Last but not least, I would like to thank all my family members for their support and understanding during my school years throughout the MEng program.

Table of Contents

Approval.....	ii
Partial Copyright Licence	iii
Abstract.....	iv
Acknowledgements.....	iv
Table of Contents.....	vi
List of Tables.....	viii
List of Figures.....	ix
List of Acronyms.....	x
Chapter 1. Introduction	1
1.1. Motivation.....	1
1.2. Project Scope.....	2
1.3. Technical Report Outline.....	3
Chapter 2. System Overview	4
2.1. Major Components.....	5
2.2. Training Process	6
Chapter 3. Hand-held User Interface	8
3.1. Hardware/Software Platform Selection	9
3.2. Development Tools	9
3.3. Major Software Components.....	10
3.3.1. SL4A	10
3.3.2. Webview	11
3.3.3. Gnu Octave Application for Android.....	11
3.3.4. Python Octave Communication	11
3.4. Python Script.....	14
3.4.1. Start-up	14
3.4.2. Main loop	16
3.5. HTML File for Webview	17
3.5.1. Javascript Start-up	17
3.5.2. Javascript Message Loop	18
3.6. Octave Script.....	19
Chapter 4. FSR Band Module	21
4.1. Module Overview.....	21
4.1.1. FSR Sensor Band	22
4.1.2. Arduino Pro Mini	23
4.1.3. OLED Display	24
4.1.4. Battery.....	25
4.1.5. Bluetooth Module	25

4.2. Firmware Development tools	26
4.3. Atmega328P Firmware	27
4.3.1. Graphic Library for OLED Display	27
4.3.2. Start-up	28
4.3.3. Main loop	29
Chapter 5. Linear Discriminate Analysis (LDA)	31
5.1. Theory	31
5.2. C Language Implementation	31
5.3. CROUT Algorithm for Matrix Determinant	34
Chapter 6. Profiling on Intel Galileo Gen2 Board	36
6.1. Linux Profiling Environment	37
6.2. Gprof	38
6.3. Profiling results	38
Chapter 7. Conclusions	39
Chapter 8. Future Work	40
8.1. Support more FSR sensors	40
8.2. Compile LDA C code in Android Native Development Kit (NDK) for PY4A	40
References	41
Appendix A. Development resources	44
Flot: Attractive JavaScript plotting for jQuery	44
Gnu Octave for Android	44
Python script debug process	44
Appendix B. Source Code	45
“dump.py”	45
“graph.html”	53
“octaveMain.java” diff output	56
“octaverc”	58
“k.m”	59
“Atmega328_AVRStudio_Test.c”	65
“crouc.c” diff output	72
“LDA_operators.h”	74
“LDA_operators.c”	75
“matrices.h”	79
“matrices.c”	80
“mycommon.h”	83

List of Tables

Table 2.1.	User settings for Android Application.....	6
Table 3.1.	Main Hardware Specification.....	9
Table 3.2.	Command list	12
Table 4.1.	Hardware summary	24
Table 6.1.	GProf results (sensor: 8, sample/class: 100)	38

List of Figures

Figure 2.1.	System Diagram.....	4
Figure 2.2.	Band Module.....	5
Figure 2.3.	User interface layout	5
Figure 2.4.	(a) Gesture number 1; (b) Gesture number 2	6
Figure 3.1.	Android Architecture.....	9
Figure 3.2.	Android Development Tools	10
Figure 3.3.	Android Application and Gnu Octave Interaction	12
Figure 3.4.	Python script start up flow chart	14
Figure 3.5.	Python script main loop flow chart.....	16
Figure 3.6.	Javascript Start-up flow chart	17
Figure 3.7.	Javascript message loop flow chart.....	18
Figure 3.8.	Octave script flow chart.....	19
Figure 4.1.	Band module block diagram.....	21
Figure 4.2.	System Schematic ^[13]	22
Figure 4.3.	OLED graphic display	24
Figure 4.4.	AVR Studio 6	26
Figure 4.5.	Atmel-ICE.....	26
Figure 4.6.	AVR firmware start-up flow chart.....	28
Figure 4.7.	AVR firmware main loop flow chart.....	29
Figure 5.1.	Calculation for linear discriminate coefficients flow chart	32
Figure 5.2.	Matrix inversion flow chart.....	33
Figure 5.3.	Laplace algorithm flow chart.....	34
Figure 5.4.	Crout algorithm flow chart	35
Figure 6.1.	Intel Galileo Gen2 Board	36
Figure 6.2.	Galileo System Diagram.....	37

List of Acronyms

ADC	Analog to Digital Converter
ADT	Android Development Tools
ANOVA	ANalysis Of VAriance
CES	Consumer Electronics Show
CSV	Comma-Separated Values
ELM	Extreme Learning Machine
FMG	Force MyoGraphy
FSR	Force Sensing Resistor
HTML	HyperText Markup Language
IDE	Integrated Development Environment
ISA	Instruction Set Architecture
LDA	Linear Discriminate Analysis
LED	Light-Emitting Diode
MAP	Maximum A Posteriori probability
OLED	Organic Light-Emitting Diode
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PY4A	Python for Android
SL4A	Scripting Layer for Android
SRAM	Static Random-Access Memory
SVM	Supported Vector Machine
UART	Universal Asynchronous Receiver/Transmitter
WMFT	Wolf Motor Function Test

Chapter 1.

Introduction

In this chapter, the background information of this project is introduced including the motivation, the project scope and the outline of this technical report.

1.1. Motivation

Patients affected by chronic diseases such as a stroke face several problems during their rehabilitation process:

Limited access to rehabilitation services. A survey done by Centers for Disease Control in the U.S. in 2005 revealed that only 31% of the stroke survivors received outpatient rehabilitation^[1]. Even when patients receive rehabilitation, it is often unstructured, hard to follow and it lacks clear goals for walking and functional use of the upper-extremities^{[2], [3]}.

Availability of home-based therapies. Oral instructions and evaluation methods used by physiotherapists in existing home-based therapies are often difficult to follow and hard to measure. Also, existing therapies rely on the patients themselves to achieve a certain number of accurate gestures on a daily basis, which is also a challenge. Still, there is evidence showing that increased activity significantly helps patients regain the ability to walk and increased functional use of the upper-extremities^{[3], [4], [5]}.

Clinical trials for the functional use of the upper-extremities. The effectiveness of clinical trials on the upper-extremities are often affected by the physical setting of the trial. The results of the frequently used Wolf Motor Function Test^[6] (WMFT) can be

interpreted subjectively. A more objective and easily accessed tool can be helpful for the clinical trial.

In the research done by ZG.Xiao and C.Menon^[7], a novel idea towards the development of a wearable feedback system using FSR sensors was proposed. The work introduces objective data analysis on Force MyoGraphy (FMG) signals to predict upper-extremities gestures. The key parts of the proposed feedback system are: Force Sensing Resistor (FSR) strap as hardware for FMG data collecting and a test protocol designed to classify 6 upper-extremities gestures for a drinking task.

The work presented in this report is aimed at developing software for a prototype system as a step forward in implementing the proposed wearable feedback system. In order to provide a monitoring device which can be easily used by patients at home, a low-cost, low-power FSR band module is put together to predict upper-extremities gestures in real-time. Also because the instant feedback of the system may promote more repetitive exercise by the patient, a display is added to the band module. Furthermore, to help facilitate the rehabilitation process guided by a physician or a physiotherapist during their collaboration with their patient, an Android application which plots real-time FMG data was used. It has user configurable settings and displays prediction result real-time is developed for tablet computer as hand-held user interface.

1.2. Project Scope

The development of the wearable feedback system involves a mechanical design which includes a strap with FSR sensors and Velcro, a case for the band module, and a Printed Circuit Board (PCB); hardware design including prototype circuits for the sensors, Organic Light-Emitting Diode (OLED) display and battery and software design including an embedded firmware for the band module, and an Android Application on Android Tablet.

My role in this project was mainly around software design focused on the following aspects:

- *Firmware development for the band module.*
- *Software development for the Android application.*
- *Discussed and implemented requirements reported from mechanical and hardware team.*
- *Feedback on the hand-held user interface are properly discussed and implemented.*

1.3. Technical Report Outline

The Technical Report is organized as follows:

- Chapter 2 provides a system overview and the system's configuration.
- Chapter 3 describes the hand-held user interface.
- Chapter 4 presents the FSR band module.
- Chapter 5 introduces the Linear Discriminate Analysis (LDA).
- Chapter 6 describes the profiling on Galileo Gen2 board.
- Chapter 7 presents future work.
- Appendix A has all the information related to the Android Application development.
- Appendix B has all the information related to the Atmega328 firmware development.
- Appendix C has all the source codes.

Chapter 2.

System Overview

In this chapter, the overview of the wearable feedback system is described with the major system components and their interactions. A greater detailed description of each major component is introduced in the following chapters.

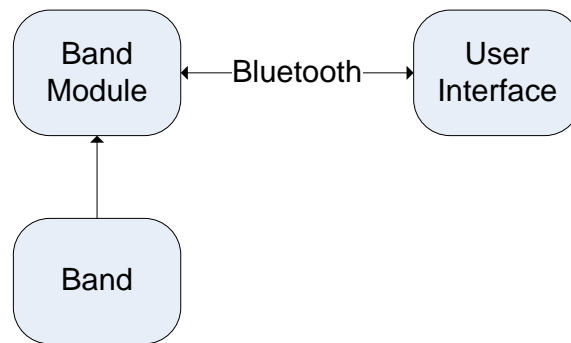


Figure 2.1. System Diagram

As shown in Figure 2.1, the wearable feedback system includes a band module for stand-alone application and a hand-held user interface for more detailed user interaction. The objective of the system is to not only promote upper-extremities exercises when a patient is fulfilling his rehabilitation at home, but also to provide more detail on in-clinic patient performance for physicians and physiotherapists.

2.1. Major Components

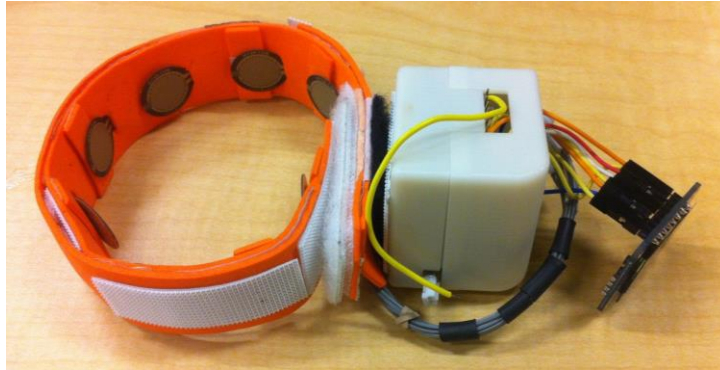


Figure 2.2. Band Module

The band module shown in Figure 2.2 is composed of the main enclosure for the electronics and a band made of FloTex foam where 8 FSR sensors are embedded. When a patient wears it, the Velcro on the band is properly attached and the band itself is a bit stretched.

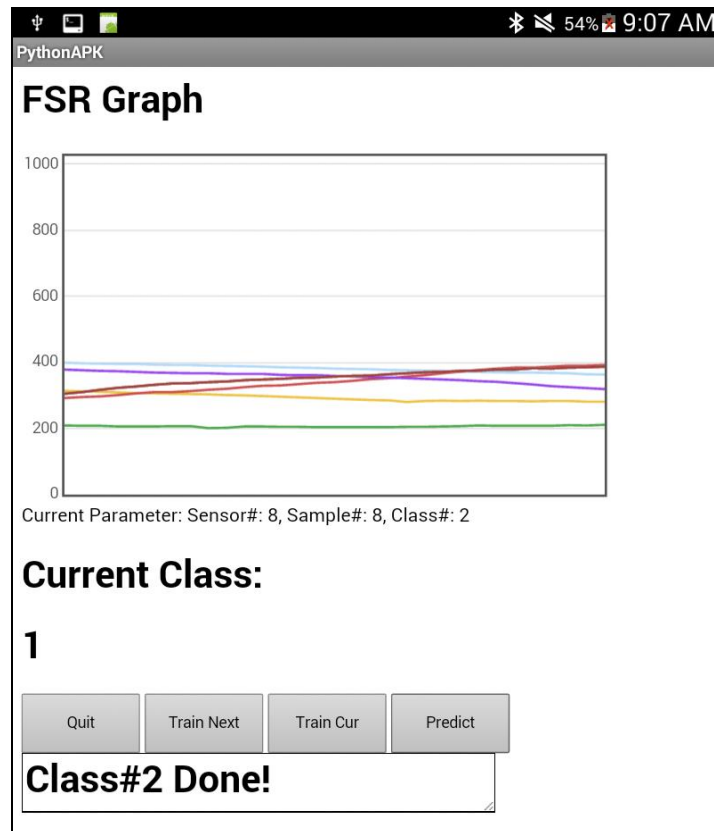


Figure 2.3. User interface layout

The hand-held user interface (shown in Figure 2.3) plots real time FMG data; does training and real time prediction with more user configurable parameters.

Table 2.1. User settings for Android Application

User Setting	User Options
Number of Sensors	2,3,4,5,6,7,8
Number of Samples	8,16,32
Number of Classes	2,3,4,5,6,7,8,9,10

Table 2.1 is the list of user settings. The number of sensors is settable and FMG data from the number of sensors are selected for training and predicting phases. The number of samples is the number of samples taken for each gesture during the training phase. The number of classes is the total number of gestures for the training phase.

2.2. Training Process

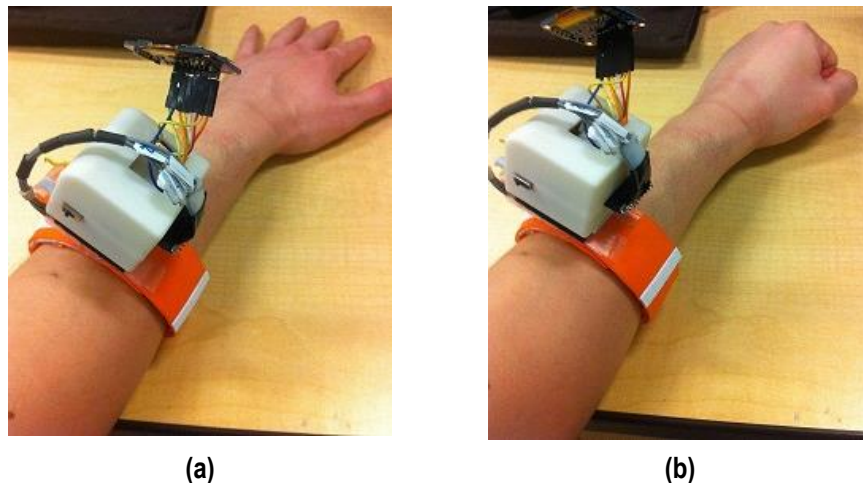


Figure 2.4. (a) Gesture number 1; (b) Gesture number 2

Before using the band module to evaluate their gesture, the patients need to perform a “training” phase to tune the machine learning algorithm. After the training phase, the patients can then perform the gestures that are properly trained and the system is able to count and notify patients the predicted gesture.

At the start of the training phase, the patients first put the band on their forearm; relax their forearm and keep that gesture for 3 seconds after the triggering training phase for gesture number 1 (shown in Figure 2.4 (a)). In this prototype, the training phase is triggered from a host PC through Bluetooth. A button can be added to allow the patients perform the triggering themselves. Then they relax their forearm, and changes their gesture, for example, make a fist (shown in Figure 2.4 (b)), and then keep the gesture for 3 seconds after the triggering training phase for gesture number 2.

After the training phase is done, the algorithm is then able to predict the current gesture number in real-time and present the predicted gesture number.

Chapter 3.

Hand-held User Interface

This chapter describes the user interface of the hand-held device used to control the wearable feedback system, and to collect/process its results. In particular, this chapter will introduce software implementation details and features. The objective of the hand-held user interface is to provide real-time FMG data to the physician and physiotherapist and to handle more hand gestures compared to home based rehabilitation.

As described in the previous section, two alternative methods for the utilization of the wearable band are made available: one is based on a host system, and one is “stand-alone”. In the host-controlled solution, the user (physician or physiotherapist) uses a tablet that collects all user information with a wireless (Bluetooth) interface and performs all the necessary processing steps to guide the user through the rehabilitation exercise.

In some cases though, it is not possible to assume that the users will have at their disposal a tablet device and/or the ability to use it especially in home based therapy. To complement the above strategy, an alternate strategy was introduced where the band is used as a stand-alone device, and some degree of processing power is embedded in the band by means of a small microcontroller with a local display. This chapter describes the tablet-based software development related to the first solution while Chapter 4 describes the microcontroller-based software development related to solution 2.

3.1. Hardware/Software Platform Selection

Because the Android Operation System is open source^[8] and covers a major section of the smart phone/tablet market^[9], it was chosen as the reference software development environment for this project.

Table 3.1. Main Hardware Specification

Operating System	Android Jelly Bean (4.3)
Display	7"
Camera	3 Mega Pixels
Battery	4000mAh
Wi-Fi	802.11a/b/g/n 2,4 + 5GHz

The Samsung Galaxy Tab3 was chosen as the hardware platform for this project because of its superior hardware feature. Its main hardware specifications are in Table 3.1.

3.2. Development Tools

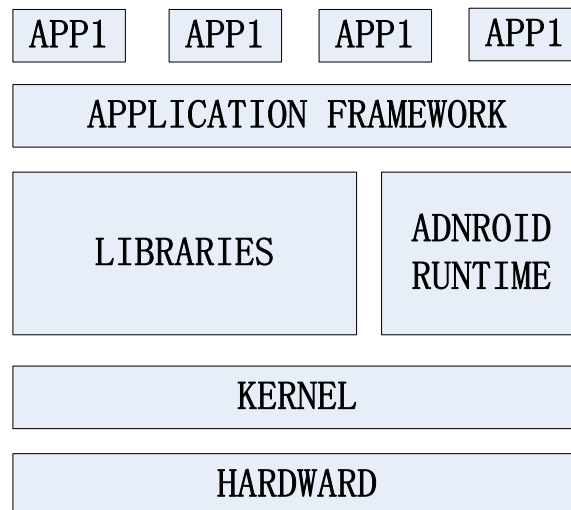


Figure 3.1. Android Architecture

Figure 3.1 is the architectural diagram of an android system. The user interface Application is running on the top level supported by an application framework in Android OS.

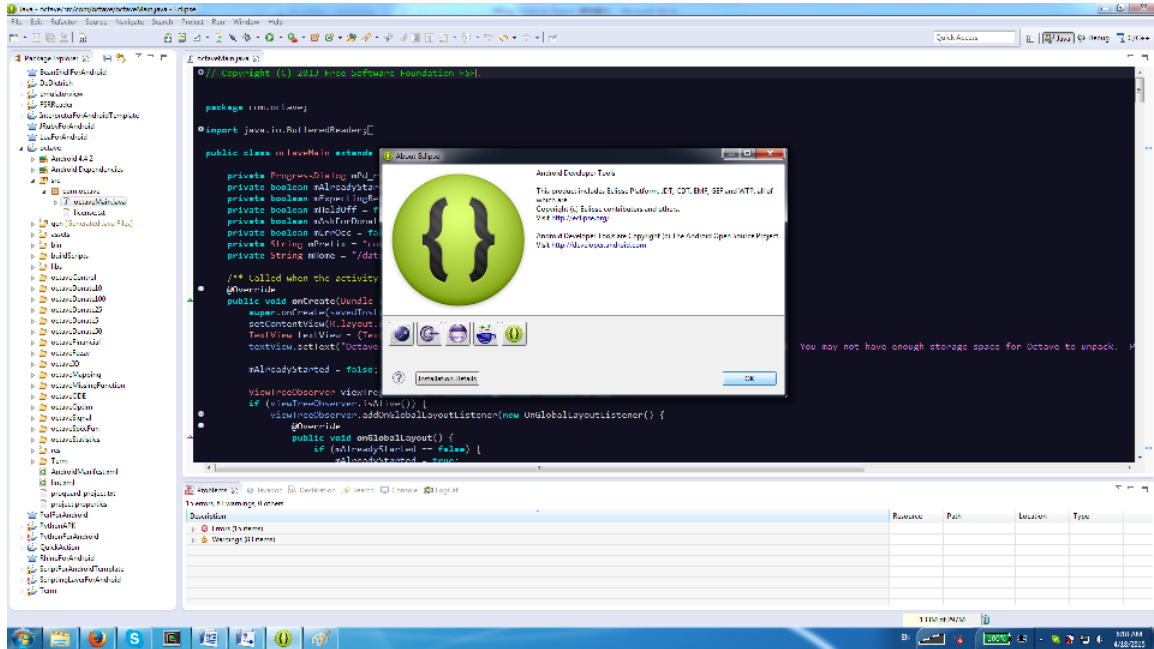


Figure 3.2. Android Development Tools

Figure 3.2 is a screen capture of Android Development Tools (ADT). In this Integrated Development Environment (IDE), the developer uses Java-like language for Android application development.

ADT is used in packaging the Python script, the supported library from Python for Android (PY4A) and the Scripting Layer for Android (SL4A) into an Android Application. It is also used to repackage the modified version of the GNU Octave for Android.

3.3. Major Software Components

3.3.1. SL4A

The first task that must be realized by the application software running on the tablet is to access FMG data sent from the band module through Bluetooth. A high level support for Android API is desirable for access functionalities such as Bluetooth. Once

the FMG data is appropriately collected on the host tablet, the system can focus more on data representation and the prediction process.

SL4A brings scripting languages to the Android by allowing editing and executing scripts and interactive interpreters directly on the Android device. These scripts have access to many of the APIs available to full-fledged Android applications, but with a greatly simplified interface. A number of scripting languages are supported by SL4a including Python. Python is chosen to be the scripting language for this particular application because of its popularity in the programming community and ease of use.

PY4A is a Python language support running on top of SL4A. The Python version used in this project is 2.7.1. The debugging process for the python script development is covered in Appendix A.

3.3.2. Webview

SL4A also provides support for web technologies to build a graphical user interface for native applications. Such a method is called Webview ^[10] which is a view that displays web pages. A HyperText Markup Language (HTML) file is created to perform real time FMG data plotting and also to allow the user to trigger the training and predicting phases of the classification process. The Webview and Python script communicates through messages via SL4A.

3.3.3. Gnu Octave Application for Android

Gnu Octave^[11] is an open-source software package for science computing and is compatible with Matlab language. An Android version of Gnu Octave has been developed by C.Champion^[12]. For this project, a script is developed to communicate with Python script and to train and predict on the FMG data.

3.3.4. Python Octave Communication

Comma-Separated Values (CSV) has been used as the reference format for the communication between the band module and the user interface. In order to ease the

interfacing work between the band module and Android application, all messages are in ASCII with the following format, with each Analog to Digital Convertor (ADC) value being a maximum of 4 digits from 0 to 1023:

[ADC#1],[ADC#2],[ADC#3],[ADC#4],[ADC#5],[ADC#6],[ADC#7],[ADC#8]

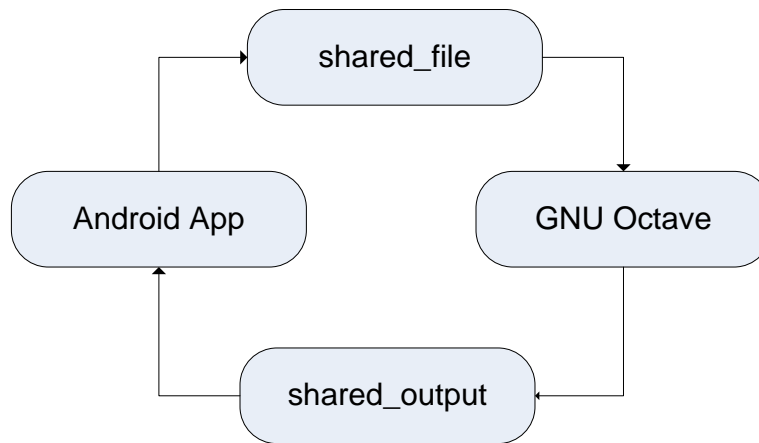


Figure 3.3. Android Application and Gnu Octave Interaction

As shown in Figure 3.3, once the data from the band module is received by the tablet by means of the Bluetooth interface, it will be loaded on to the Gnu Octave software through a file. Once the data is trained and starts predicting, the result will be passed back to the python script through another file so that the Python script can then pass the information to Webview for the user interface.

Table 3.2. Command list

Command name	Letter in file messaging
Data	d
Train	s
Predict	p

Table 3.2 is the list of commands and the letter in the file messaging. The following is the message format in the file to pass data from the Python script to Octave:

[Command][ADC#1], [ADC#2], [ADC#3], [ADC#4], [ADC#5], [ADC#6], [ADC#7], [ADC#8],[Time Stamp]

The following is the message format in the file to pass data from Octave back to the Python script:

[Command],[Predicted Class],[Time Stamp]

Note: time stamp format: YYYY-MM-DD_hh:mm:ss

3.4. Python Script

3.4.1. Start-up

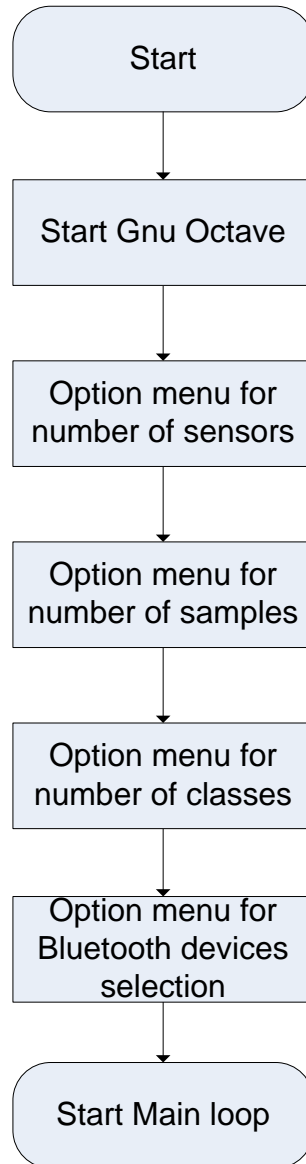


Figure 3.4. Python script start up flow chart

Figure 3.4 is the flow chart for the Android Application that is the result of the described work. At the beginning, the application starts Gnu Octave, waits 2 seconds, and then pops up a menu for the user to choose the relevant design options:

1. the number of sensors
2. the number of samples
3. the number of classes
4. Bluetooth device (Band Module) selection

Only one Bluetooth device (Band Module) can be connected at once due to the fact that Bluetooth is a point to point link network. Finally, the user interface starts with real time data plotting, and options for training as well as predicting. Detailed user settings are shown in Table 2.1.

3.4.2. Main loop

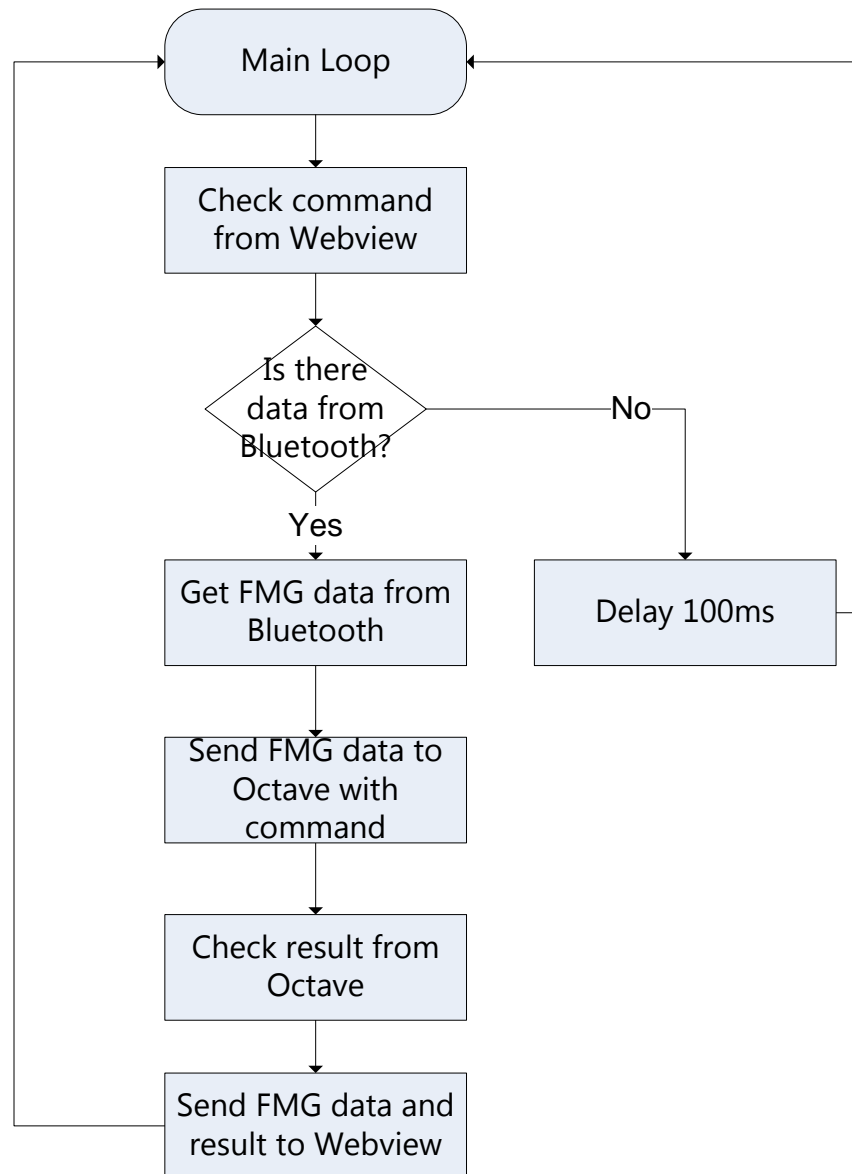


Figure 3.5. Python script main loop flow chart

Figure 3.5 describes the flow chart of the main loop in the Python script: the script first checks if there are commands from Webview. The commands are train, predict and quit. It then checks if Bluetooth data is available; if so, it gets FMG data from Bluetooth; if not, it delays 100ms and goes back to the start of the main loop. If there is Bluetooth data, it takes the data, sends the data to Octave with the command from

Webview. Next it checks the results coming back from Octave and finally it sends the data to Webview with result from Octave.

3.5. HTML File for Webview

The goal of the HTML file is to provide a platform for not only displaying FMG data and user feedback but also taking commands from the user. In order to change the content of the HTML layout, Javascript is used to interact with the Python script via SL4A.

3.5.1. Javascript Start-up

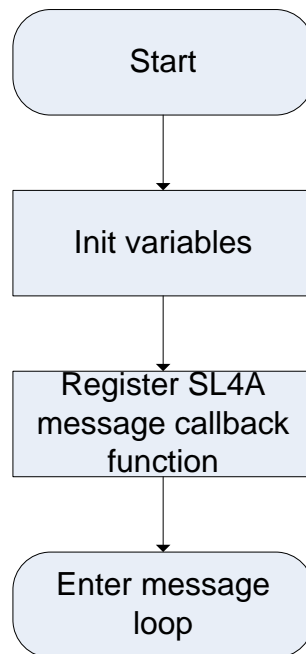


Figure 3.6. Javascript Start-up flow chart

Figure 3.6 shows the start-up sequence of the javascript: first of all, all variables are initialized; and then the SL4A message call back function is registered; finally, the main loop is entered.

3.5.2. Javascript Message Loop

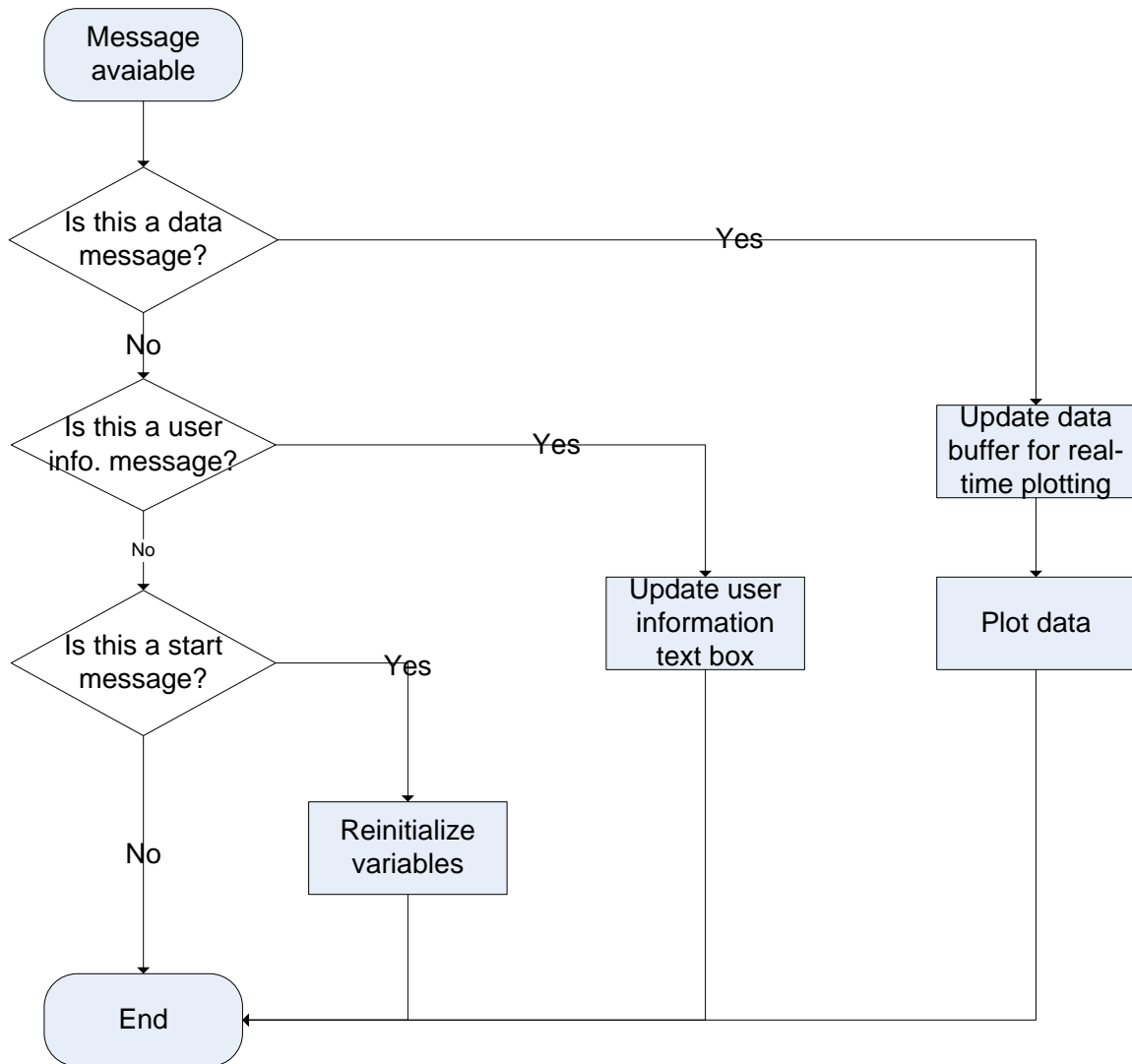


Figure 3.7. Javascript message loop flow chart

Figure 3.7 is the loop which javascript gets messages from or sends messages to the Python script. There are a total of 3 commands: start, data and user information. The start command re-initializes variables corresponding to user settings, the data command is used to plot FMG data on the web page, and the information command displays information to the user during the training and predicting phases.

3.6. Octave Script

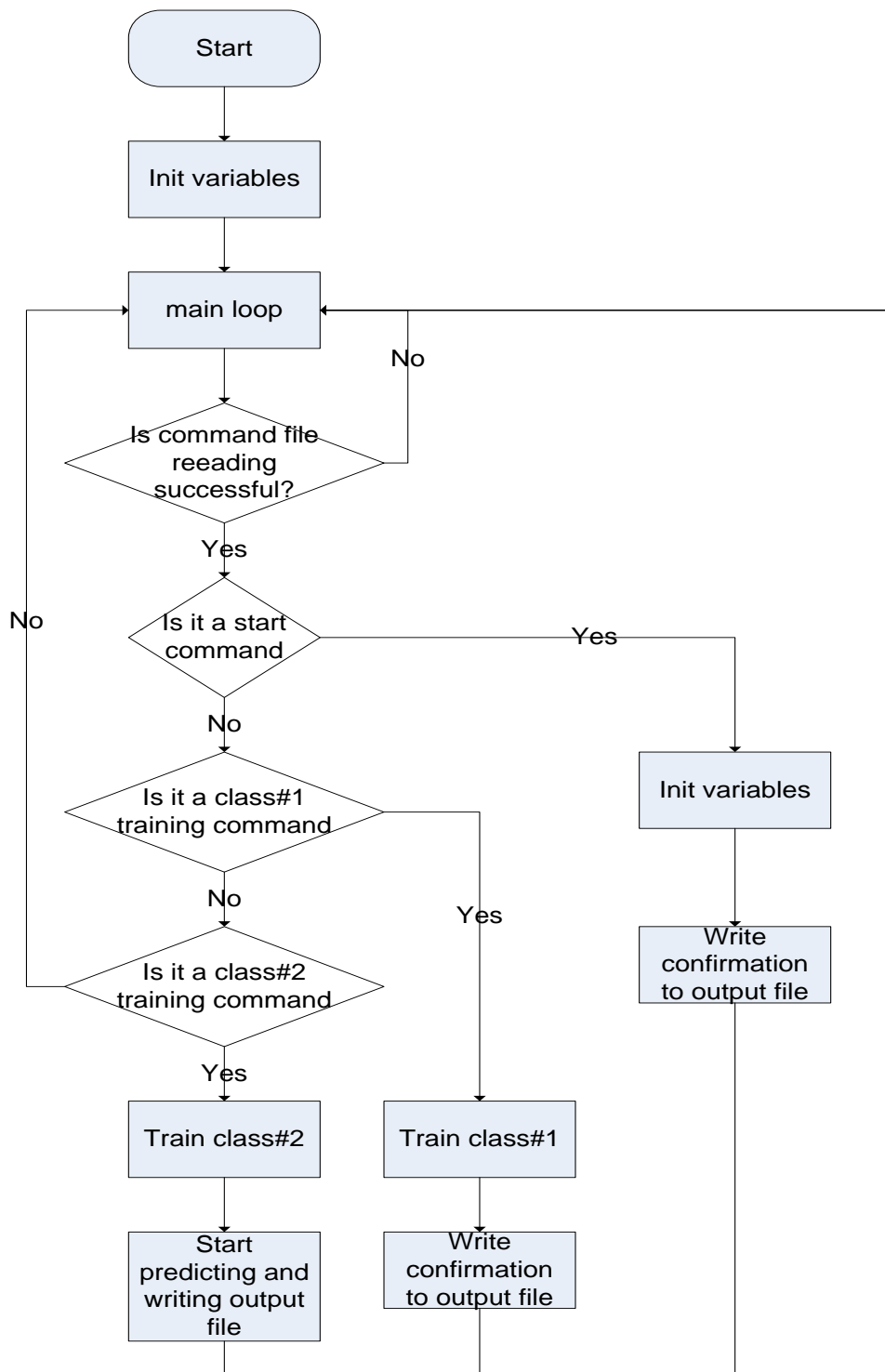


Figure 3.8. Octave script flow chart

Figure 3.8 is the flow chart of the Octave script running in GNU Octave on the Android. After initialization of all the variables, it detects if there is information in the command file from the Python script. If so, the command and data are parsed into Octave and they are processed and the results are output back to the Python script via the output file.

Chapter 4.

FSR Band Module

In this chapter, the FSR band module is introduced and details are covered on all major hardware components, the software development environment and how firmware is implemented. The objective of the band module is to provide a low-cost, low power device which not only monitors the rehabilitation process of the patient; but also gives instant feedback to promote more repetitive exercise.

4.1. Module Overview

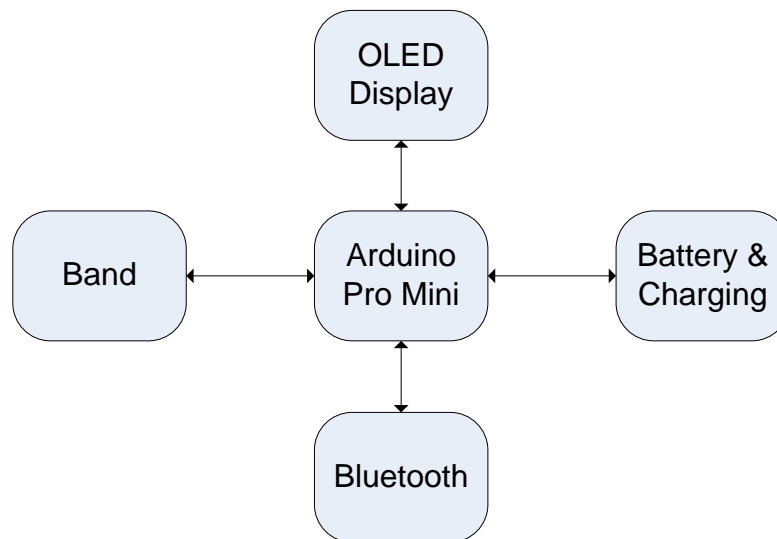


Figure 4.1. Band module block diagram

Figure 4.1 is the block diagram of the band module. There are five major hardware components: an Arduino Pro Mini board, the band, an OLED display, a battery and the Bluetooth module.

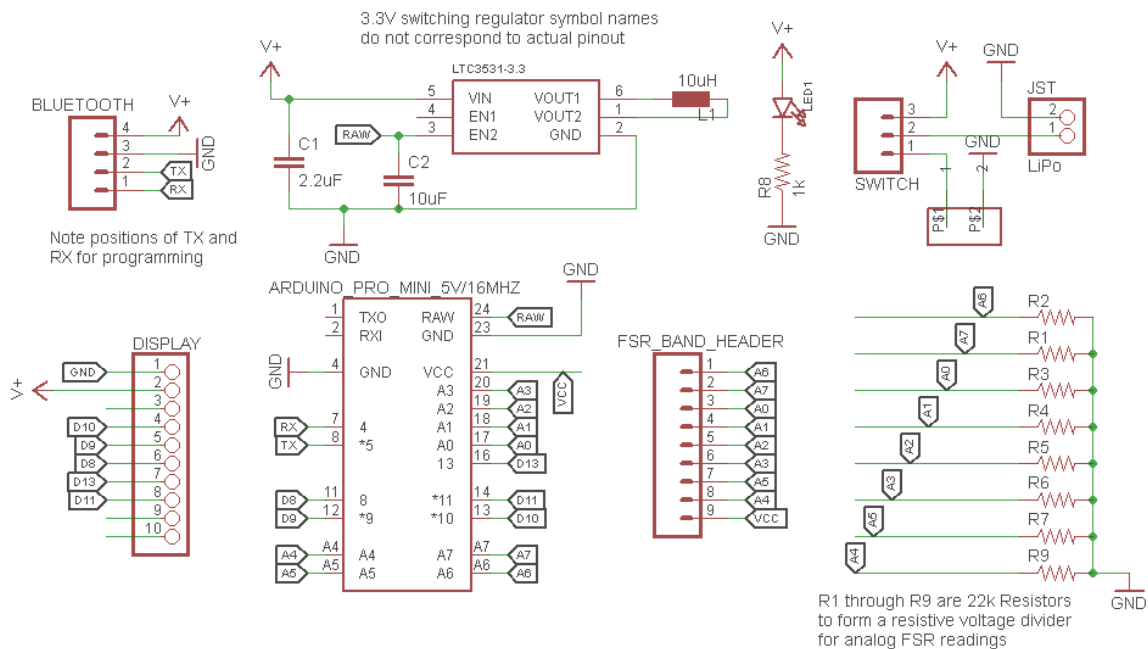


Figure 4.2. System Schematic^[13]

Figure 4.2 is the schematic of the prototype system. The component in the center of the schematic is Arduino Pro Mini. The connectors on the left are connector for Bluetooth and connector for display. The connector on the bottom is the connector for the band. The circuit on the right is the circuit for FSR sensors. And on the top, there are circuits for voltage regulator and battery. The following components/modules are directly powered by 3.7V from the battery: the display, the Bluetooth module, the switching regulator and the LED. The Arduino Pro Mini is powered by 3.3V from the output of the switching regulator. The FSR sensors are powered by 3.3V from the Arduino Pro Mini.

4.1.1. FSR Sensor Band

The band is made of FloTex foam with Velcro so that the user can easily wear it and take it off. Also the flexibility of the band helps holding the FSR sensors against the skin on the forearm in order to obtain good quality FMG signals. The 8 FSR sensors are placed on the band at equal distance so that there is no wearing preference.

Each FSR sensor is composed of a piezoelectric plate which changes its resistance when force is applied to its surface. It is light weight and can be embedded

into textiles ^[14] for wearable applications. The signal generated by the FSR sensor is called the FMG signal which is proven to be useful in human limb muscle monitoring including lower extremities ^[15] and upper extremities ^[16].

In Figure 2.2, a band made of FloTex is used to group 8 FSR sensors (embedded into the material) so that it can be worn at any location on the forearm of the patient. The band is holding the surface of the sensors against the skin in such way that contraction or relax of the muscle on the sensor area can be converted into a FMG signal.

In Figure 4.2, the circuit diagram of 8 FSR sensors is shown. Each FSR circuit is composed of a 22kOhm resistor in series with the FSR to form a voltage divider. Signals ADC 1 to ADC8 are output to the analog to digital pins on an embedded microcontroller chip that is used for signal processing in the digital domain. When force is applied to the surface of an FSR sensor, the ratio of the voltage divider is increased due to the fact that the resistance of the FSR is decreasing. Vice versa, when force is taken away from the surface of the FSR sensor, the ratio of the voltage divider is decreased due to the fact that the resistance of the FSR is increasing. As the result, the ADC reading of the voltage divider changes accordingly.

4.1.2. Arduino Pro Mini

Table 4.1. Hardware summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	5V-12V
Digital I/O Pins	14
Analog Input Pins	6
DC Current per I/O Pin	40mA
Flash Memory	32kB
SRAM	2kB
EEPROM	1kB
Clock Speed	16MHz

With the idea of low-cost and low power hardware in mind, an Arduino Pro Mini^[17] board is chosen as the main board for the band module. As a very popular fast prototyping hardware platform, Arduino Pro Mini is very competitive in terms of dimensions (0.7"x1.3") compared with other models in the family. Table 4.1 is the hardware summary of the Arduino Pro Mini board. Its 8MHz clock handles the computing power needed for this application.

4.1.3. OLED Display

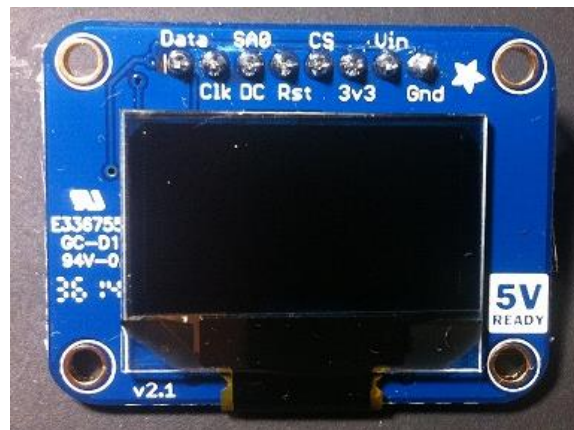


Figure 4.3. OLED graphic display

Figure 4.3 is the monochrome 0.96" 128x64 OLED graphic display^[18] from Adafruit. This display is chosen not only because of its small dimensions (about 1" diameter), but also because it is very readable due to the high contrast of the OLED

display. No backlight is required because the display makes its own light. This reduces the power required to run the OLED and is why the display has such high contrast. There are a total of 8 pins on the OLED module: “Data”, “Clk”, “SA0”, “Rst” and “CS” pins are for Serial Peripheral Interface (SPI) communication; and ‘Vin’ and ‘Gnd’ pins are for power.

4.1.4. Battery

For the power source of the system, the battery must be very light and high density in power. A polymer lithium ion battery^[19] is used for powering the system. It is only 22g with a power output from each cell of 3.7V at 1000mAh. With the assumption that the Arduino Pro Mini draws 10mA current and OLED Light-Emitting Diode (LED) draws 20mA current, the system (5V power supply) should be able to run at least 24 hours. After that, the battery needs to be recharged.

4.1.5. Bluetooth Module

In order to keep a reliable and robust wireless link between the band module and hand-held user interface, due to the fact that most of the tablet computer comes with Bluetooth support, a Bluetooth module is needed. The HC-06 Bluetooth module^[20] can only work in Slave mode and accepts only 3.3V on TX and RX pins. The modem works as a serial (RX/TX) pipe with a serial stream from 2400 to 115200bps.

4.2. Firmware Development tools

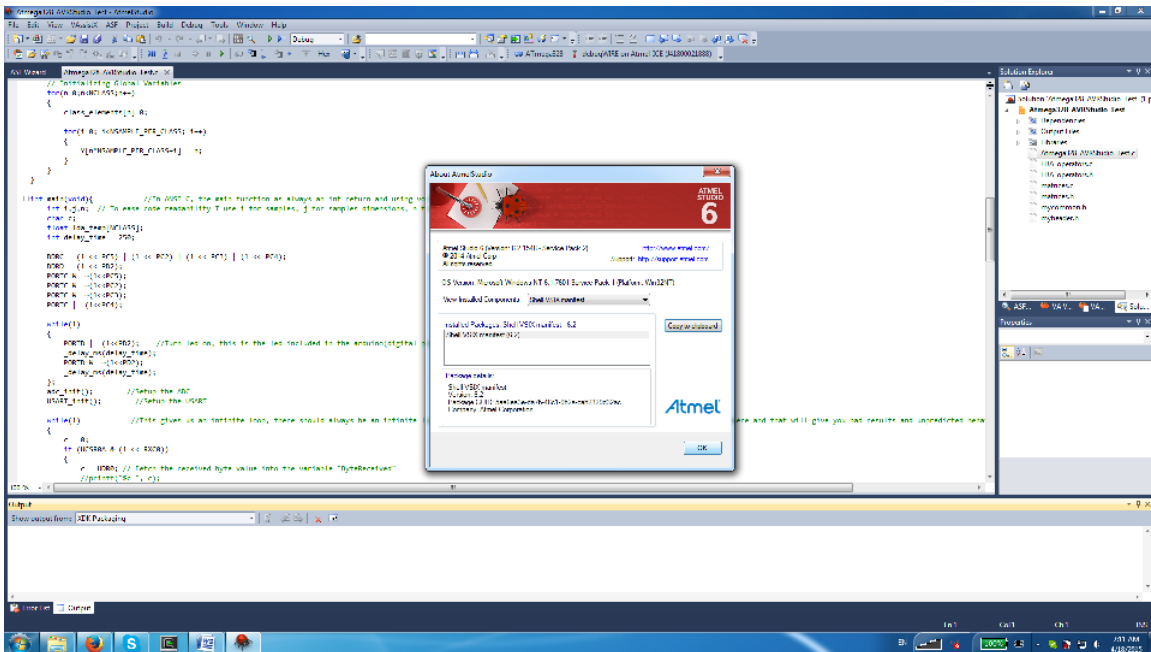


Figure 4.4. AVR Studio 6

Figure 4.4 is the layout of the Atmel IDE, the AVR Studio 6^[21], which is used for developing firmware for Atmega328P on the band module.



Figure 4.5. Atmel-ICE

AtemI-ICE^[22] (shown in Figure 4.5) is the debugger/programmer used for developing firmware for Atmega328P.

4.3. Atmega328P Firmware

4.3.1. Graphic Library for OLED Display

U8glib is a library with support of SSD1306, the Dot Matrix OLED Common Driver with Controller used on Monochrome 0.96" 128x64 OLED graphic display from Adafruit. The library supports AVR, ARM and Arduino platforms.

To program the OLED display using this library, there are 3 main steps:

1. Call an U8glib constructor
2. Add the "picture loop"
3. Write a graphics "draw" procedure

Call an U8glib constructor. `u8g_dev_ssd1306_adafruit_128x64_hw_spi` is used as the parameter for function `u8g_InitHWSPI()`.

Add the "picture loop". `draw_initial()` function is used at the start of the program to display the project name and author information, when the program is in the main loop, the `draw_lda()` function is called to display the current state of the program (IDLE, training and predicting) with the corresponding result.

4.3.2. Start-up

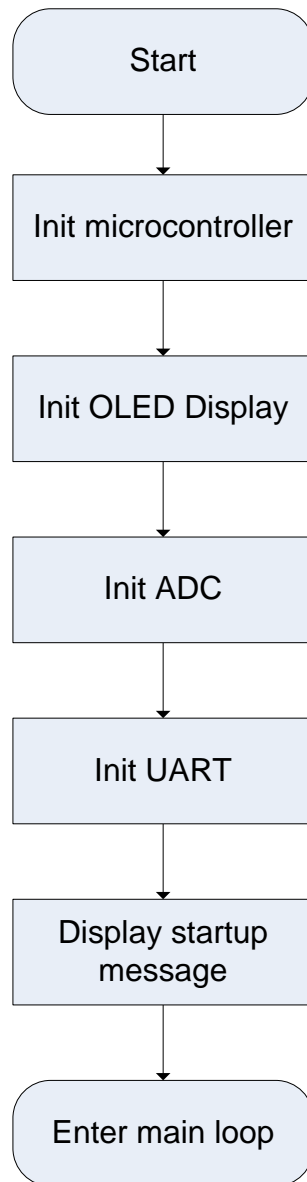


Figure 4.6. AVR firmware start-up flow chart

Figure 4.6 is the flow chart of the start-up process of the firmware. At first, the microcontroller is initialized, then the OLED graphic display library is initialized, after that, the ADC and Universal Asynchronous Receiver/Transmitter (UART) are initialized and the start-up message is displayed before entering the main loop.

4.3.3. Main loop

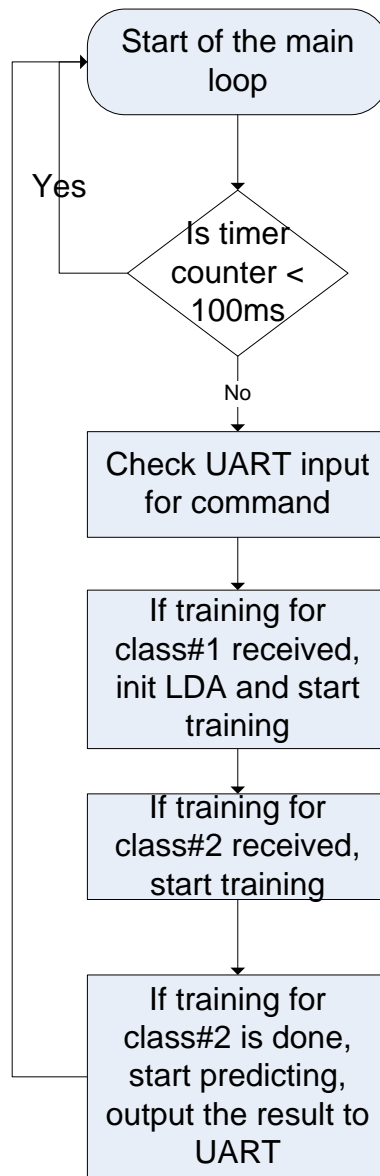


Figure 4.7. AVR firmware main loop flow chart

Figure 4.7 is the flow chart for the main loop in the firmware. At the start of the main loop, a timer counter is checked to make sure the rest of the routine is executed every 100ms. After that, if the timer check shows 100ms has passed, firmware checks the UART input for the user command. Then if command training for class number 1 is received, the LDA library is initialized and the training is started. Also if command training for class number 2 is received, the training is started. After training for class number 2 is done, predicting is started and the result is output to both the UART and the OLED graphic display.

More detailed description on LDA C code is covered in Chapter 5.

Chapter 5.

Linear Discriminate Analysis (LDA)

In this chapter, LDA is introduced with details on its theory, C language implementation and an improvement by using Crout algorithm to calculate matrix determinant. LDA is originally chosen as the classification algorithm for this application because it's less complex compared to other classification algorithms, such as Supported Vector Machine (SVM) and Extreme Learning Machine (ELM). Less complexity helps lower the cost of implementing the algorithm on a low-cost, low power microcontroller system.

5.1. Theory

LDA is related to Analysis of Variance (ANOVA) and Principal Component Analysis (PCA) ^[23]. It is used in statistics, pattern recognition and machine learning. It is the result of a Maximum A Posteriori probability (MAP) estimate with the assumption that the classes are normally distributed with the same variance ^[23]. It was originally proposed by R.Fisher ^[24] in his research solving plant taxonomic problems to maximize between class mean while minimizing within class variance without the assumptions of normal distribution and equal variance.

5.2. C Language Implementation

The algorithm used in this project was originally written by W.Dwinneel ^[25] in Matlab. Then it was converted to C language by F.Campi and A. Shamshuddin ^[26].

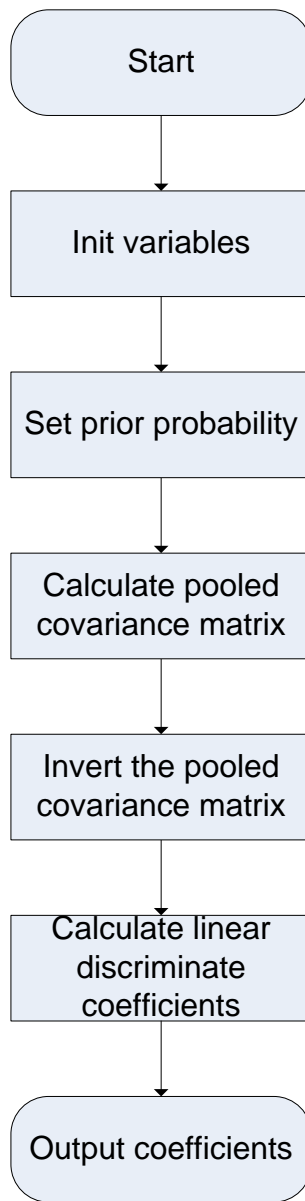


Figure 5.1. Calculation for linear discriminate coefficients flow chart

Figure 5.1 is the flow chart for calculating the LDA coefficients: at the start of the function, all global variables are initialized with their proper value; then prior probability for each class is set to be equal; after that, a pooled covariance matrix is calculated and inverted; and finally linear discriminate coefficients are calculated and output to the main loop.

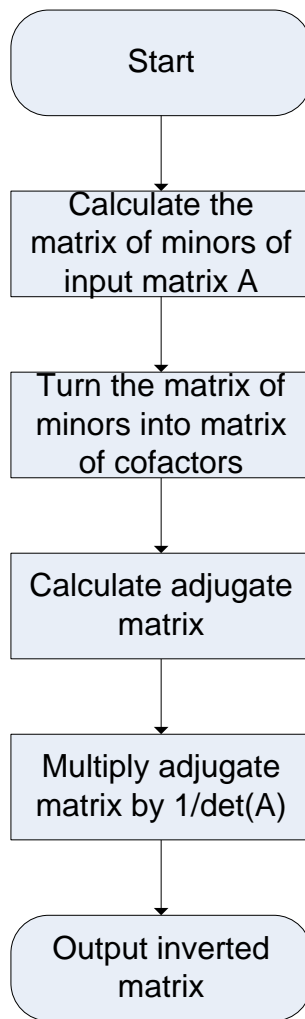


Figure 5.2. Matrix inversion flow chart

Figure 5.2 is the flow chart for matrix inversion: first of all, the matrix of minors of input matrix A is calculated; then it is turned into a matrix of cofactors; then the adjugate matrix is calculated and multiplied by $1/\det(A)$ ($\det(A)$ is the determinant of matrix A); and the product is the inverted matrix.

5.3. CROUT Algorithm for Matrix Determinant

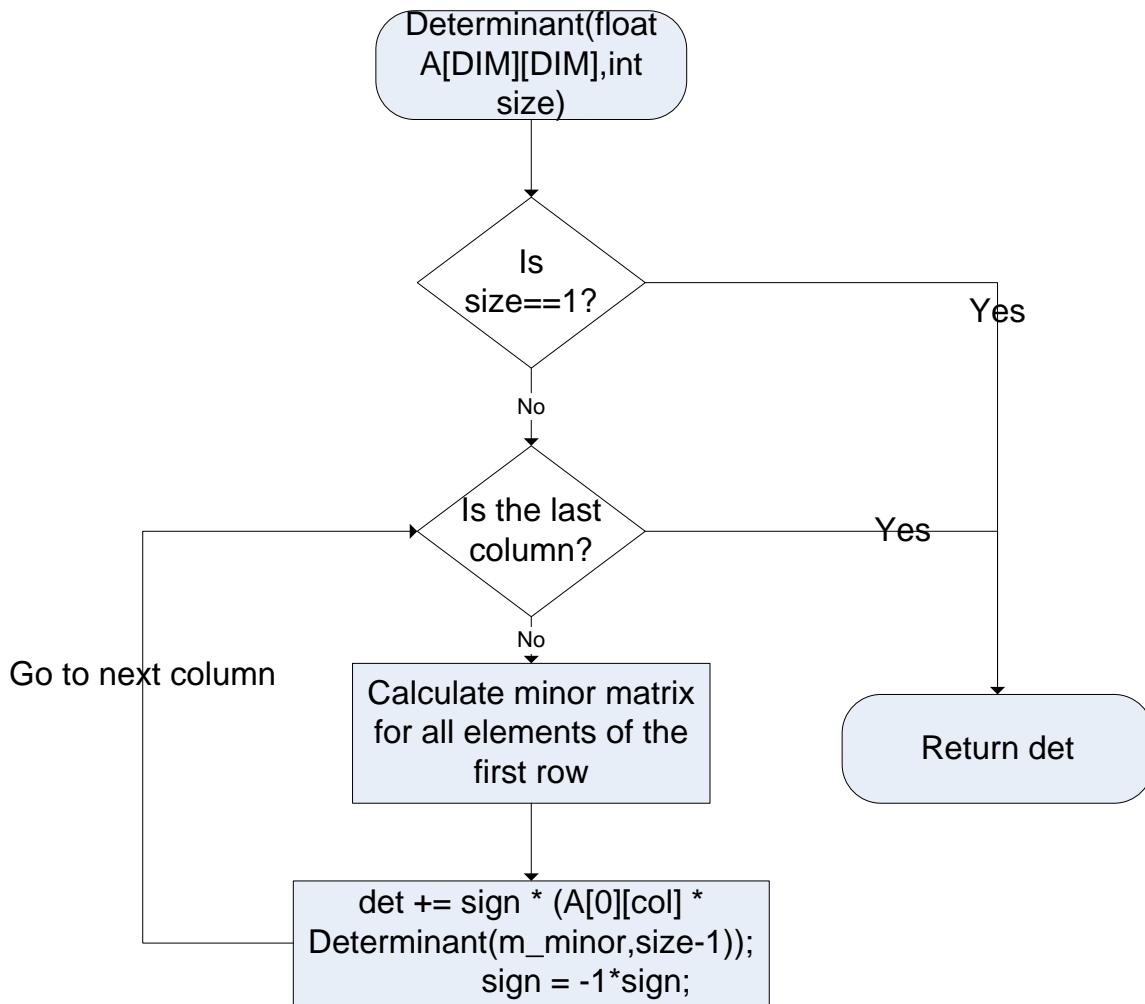


Figure 5.3. Laplace algorithm flow chart

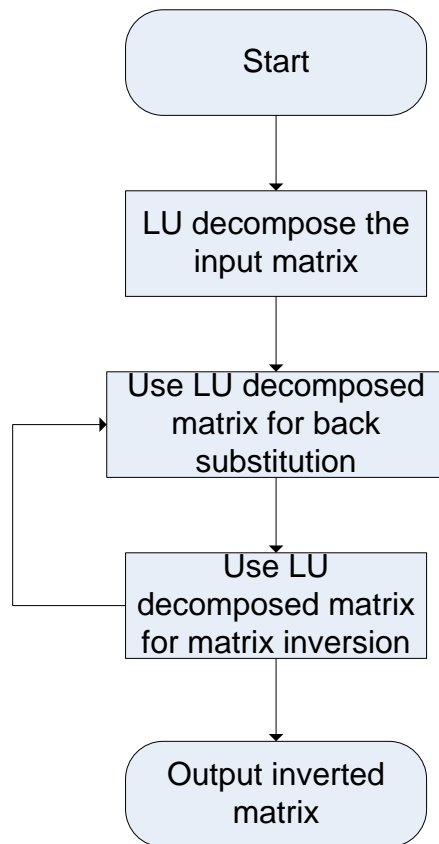


Figure 5.4. Crout algorithm flow chart

Originally the C code used the Laplace formula (Figure 5.3), a recursive method to get the determinant of a matrix. Improvement is made when using the Crout algorithm (Figure 5.4) to first decompose the matrix to upper and lower triangular matrices and then calculate their determinant by multiply all its diagonal elements. The Crout algorithm is non recursive therefore the number of calls to the function is reduced to get the inverse of a matrix especially when class numbers are high. In the Crout algorithm C implementation^[27], a singular matrix (non invertible) can be detected and the minimum matrix value is defined to avoid overflow.

More details on the difference between the Crout algorithm and the Laplace formula in terms of performance during profiling are covered in the next chapter.

Chapter 6.

Profiling on Intel Galileo Gen2 Board

Due to the maximum 6 FSR sensor allowed during training with the stand-alone band module, a more powerful microcontroller (32-bit versus 8-bit) is preferred for training and predicting with more FSR sensors. In this chapter, profiling work on an Intel Galileo Gen2 board is introduced including a Linux profiling environment, GProf and profiling results. The objective of the profiling work is to evaluate the performance of the LDA algorithm on a 32-bit microcontroller with a larger data set (class number, sensor number and sample per class number)

In January 2015, on an international Consumer Electronics Show (CES), Intel presented its Intel Curie Module^[28] designed for compact wearable technologies. In order to evaluate its performance as our next generation hardware platform, but not wanting to wait until it's available for the development community, an Intel Galileo Gen2 board^[29] is chosen because it has the same Quark microcontroller as the Curie Module.



Figure 6.1. Intel Galileo Gen2 Board

Figure 6.1 is the Intel Galileo Gen2 board. The following are the main hardware features:

- 400MHz 32-bit Intel Pentium Instruction Set Architecture (ISA)-compatible processor o 16 KBytes on-die L1 cache
- 512 KBytes of on-die embedded Static Random-Access Memory (SRAM)
- 10/100 Ethernet connector
- USB 2.0 Host connector
- USB Device connector, used for programming

6.1. Linux Profiling Environment

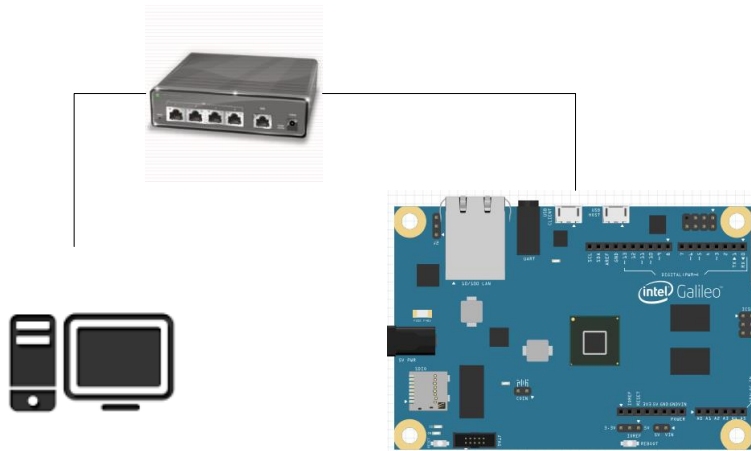


Figure 6.2. Galileo System Diagram

In order to have a development environment for the Intel Galileo Gen 2 board, a Debian Linux distribution specifically for the Galileo Gen 2 board called Galileo Debian is used. This project provides a SD card image of the Debian Wheezy for the Intel Galileo Gen 2 board. It uses the Linux 3.8.7 kernel that was released for this board, but is otherwise a full Debian system with C language development/debug tools such as the GNU Compiler Collection (GCC) ^[30], GNU Project Debugger (GDB) ^[31] and gprof ^[32]. Once the Linux system is up and running, the developer uses his host PC to remotely log into the Linux system using Secure SHell (SSH) as shown in Figure 6.2.

6.2. Gprof

Gprof is a Unix tool for application performance analysis. When an application is compiled with GCC, special instrumentation code is inserted by adding '-pg' option. The sample data is stored in 'gmon.out' and can be analyzed later by Gprof. With the '-s' option, several output files can be combined by Gprof for cumulative analysis on several runs of the application.

The output of Gprof are in two parts: flat profile and text call graph. The flat profile provides total execution time of each function and the percentage of the total running time. The functions list in flat profile is sorted by percentage. And the text call graph shows, for each function, who called it (parent) and who it called (child subroutines).

6.3. Profiling results

Table 6.1. GProf results (sensor: 8, sample/class: 100)

	Laplace formula	CROUT
Execution time (microseconds)	830	0
Number of calls (8 sensors)	623521	1

Note: 0 in the second row and third column means the execution time for dett() is less than 0.01ms.

Table 6.1 is the GProf results comparing the performance of the LDA by using the original Laplace formula and the CROUT algorithm for matrix determinant. In terms of execution time, CROUT algorithm takes less than 0.01ms versus 830 ms from Laplace formula for 8 sensors and 100sample/class. In terms of number of calls, the Crout algorithm takes only 1 call versus 623521 from the Laplace formula. This optimization speeds up the training stage of the LDA algorithm.

Chapter 7.

Conclusions

In this MEng project, software for a wearable feedback system has been implemented. This system helps stroke patients, their physicians and their physiotherapists during the rehabilitation process.

There are two hardware platforms: an Android tablet and a band module with microcontroller in it. Two system configurations are proposed: one is to use the band module alone in patients' home while they are doing their daily exercises, and the other one is to use both band modules with patients and Android application on a tablet with their physician or physiotherapist to present detailed information.

In the Android application, a Python script is developed to get data from the band module, display such data on the tablet and communicate with the Gnu Octave application on the same tablet for real-time gesture prediction. A HTML file is developed for real-time data plotting and presenting user information. The python script and HTML file are packed into a stand-alone Android application for ease of use.

In the band module, a firmware is developed to get data from the FSR sensors, send data through Bluetooth module to the tablet, use LDA algorithm for training and gesture predicting. The LDA algorithm is presented and an optimization is made to reduce the number of calls and time consumption.

Finally, the performance of LDA algorithm is profiled on an Intel Galileo Gen2 board to evaluate for the next generation hardware platform of this system. With more powerful hardware, more information can be processed by the LDA algorithm such as more sensor number, more sample number and more class number.

Chapter 8.

Future Work

In this chapter, future work is introduced. The objective of this future work is to not only improve the existing system in terms of performance, but also to reduce the complexity of the current software framework.

8.1. Support more FSR sensors

It is possible to put a greater number of smaller FSR sensors on to the band so that, with a much more powerful microcontroller on the band module, detailed information from different muscle groups can be used in gesture training and predicting.

8.2. Compile LDA C code in Android Native Development Kit (NDK) for PY4A

The complicity of the current combination of the Android Application and the Gnu Octave for Android solution can be greatly reduced by compiling the LDA C code in Android NDK^[33] as a module for PY4A and called directly in the Python script.

References

1. Centers for Disease Control. Outpatient rehabilitation among stroke survivors: 21 states and the District of Columbia, 2005. MMWR Morb Mortal Wkly Rep. 2007;56:504-507.
2. Management of Stroke Rehabilitation Working Group. VA/DoD clinical practice guidelines for the management of stroke rehabilitation. <http://www.rehab.research.va.gov/jour/10/479/pdf/VADODclinicalGuidelines479.pdf>. Accessed September 20, 2011.
3. Duncan P, Sullivan K, Behrman A, et al. Body-weight-supported treadmill rehabilitation program after stroke. N Engl J Med. 2011;364:2026-2036.
4. Duncan P, Studenski S, Richards L, et al. Randomized clinical trial of therapeutic exercise in subacute stroke. Stroke. 2003;34:2173-2180.
5. Wolf SL, Winstein CJ, Miller JP, et al. Effect of constraint-induced movement therapy on upper extremity function 3 to 9 months after stroke: the EXCITE randomized clinical trial. JAMA. 2006;296:2095-2104.
6. Wolf SL, Thompson PA, Morris DM, et al. The EXCITE Trial: attributes of the Wolf Motor Function Test in patients with subacute stroke. Neurorehabil Neural Repair. 2005;19:194-205.
7. Xiao ZG and Menon C (2014) Towards the development of a wearable feedback system for monitoring the activities of the upper-extremities, Journal of Neuroengineering and Rehabilitation, Vol.11, No.2, 13pp.
8. "Android Open Source Project" Internet: <https://source.android.com/>, [Apr. 17th, 2015].
9. "Smartphone OS Market Share, Q4 2014" Internet: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>, [Apr. 17th, 2015].
10. "WebView | Android Developers" Internet: <http://developer.android.com/reference/android/webkit/WebView.html>, Apr. 3rd, 2015 [Apr. 5th, 2015].
11. "GNU Octave" Internet: <https://www.gnu.org/software/octave/>, [Apr. 5th, 2015].

12. C. Champion. "Octave – Android Apps on Google Play" Internet: <https://play.google.com/store/apps/details?id=com.octave&hl=en>, Feb. 27th, 2015 [Apr. 5th, 2015].
13. T. Frison. "Re: Band module for Dong". Personal Communication (Apr. 8th, 2015).
14. J. Meyer, P. Lukowicz, and G. Tröster. Textile pressure sensor for muscle activity and motion detection. In ISWC 2006: Proceedings of the 10th IEEE International Symposium on Wearable Computers, October 2006.
15. P. Lukowicz, F. Hanser, C. Szubski, and W. Schobersberger. Detecting and interpreting muscle activity with wearable force sensors. In Pervasive 2006, pages 101–116, May 2006.
16. Ogris, G., Kreil, M., Lukowicz, P.: Using FSR based muscle activity monitoring to recognize manipulative arm gestures. In: Int. Symp. on Wear. Comp. (October 2007).
17. "Arduino Pro Mini" Internet: <http://www.arduino.cc/en/Main/arduinoBoardProMini>, [Apr. 17th, 2015].s
18. "Monochrome 0.96" 128x64 OLED graphic display" Internet: <https://www.adafruit.com/products/326>, [Apr. 5th, 2015].
19. "Polymer Lithium Ion Battery - 1000mAh" Internet: <https://www.sparkfun.com/products/339>, [Apr. 5th, 2015].
20. "Arduino with HC-05 bluetooth module" Internet: <http://www.martyncurrey.com/tag/hc-05/>, Oct. 27th, 2014 [Apr. 5th, 2015].
21. "Atmel Studio 6 - The Integrated Development Environment" Internet: <http://atmega32-avr.com/download-avr-studio-6/>, Oct 30th, 2012 [Apr. 5th, 2015].
22. "Atmel-ICE" Internet: <http://www.atmel.com/tools/atatmel-ice.aspx>, [Apr. 5th, 2015].
23. "Linear discriminant analysis" Internet: http://en.wikipedia.org/wiki/Linear_discriminant_analysis, Mar. 28th, 2015 [Apr. 5th, 2015].
24. R.A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems". *Annals of Eugenics*, 7: 179-188, (1936).
25. W. Dwinnell. "Data Mining in MATLAB: Linear Discriminant Analysis (LDA)". Internet: <http://matlabdatamining.blogspot.de/2010/12/linear-discriminant-analysis-lda.html>. Dec. 11, 2010 [Dec 4th, 2014].

26. F. Campi, A. Shamshuddin. "LDA C Code". Personal Communication (Jan 20th, 2015).
27. Internet: <http://algorithm.narod.ru/ln/crout.c>, [Dec 4th, 2014].
28. "Intel® Curie™ Module: Unleashing Wearable Device Innovation" Internet: <http://www.intel.com/content/www/us/en/wearables/wearable-soc.html>, [Apr. 17th, 2015].
29. "Intel® Galileo Gen 2 Development Board—Empower Your Prototype" Internet: <http://www.intel.com/content/www/us/en/do-it-yourself/galileo-maker-quark-board.html>, [Apr. 5th, 2015].
30. "GCC, the GNU Compiler Collection" Internet: <https://gcc.gnu.org/>, Mar. 20th, 2015 [Apr. 5th, 2015].
31. "GDB: The GNU Project Debugger" Internet: <https://www.gnu.org/software/gdb/>, Feb. 20th, 2015 [Apr. 5th, 2015].
32. "GNU gprof" Internet: <https://sourceware.org/binutils/docs/gprof/>, [Apr. 5th, 2015].
33. "Android NDK" Internet: <https://developer.android.com/tools/sdk/ndk/index.html>, [Apr. 17th, 2015].

Appendix A.

Development resources

Flot: Attractive JavaScript plotting for jQuery

<http://www.flotcharts.org/>

Gnu Octave for Android

<https://github.com/corbinlc/octave4android/commit/b3bb7f6de607a99840f8b4e6dd40ae12460d2563>

Python script debug process

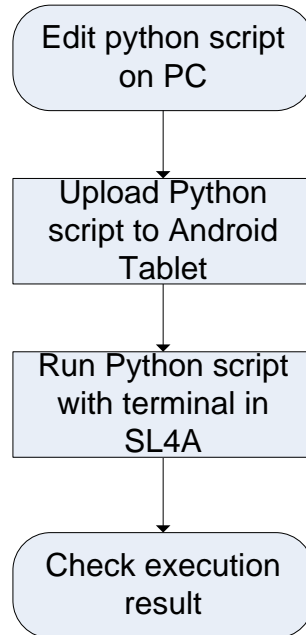


Figure A.1. Debug process for Python script

Figure A.1 is a typical debug process for Python scripts: the script is edited on a host PC first. Then by using ADT, the script is uploaded onto the Android device, and then it is executed with a terminal so that debug/error message can be displayed. At the end of the process, either the execution is successful or it exits with debug/error messages that can eventually help track down problems.

U8glib setup in AVR Studio 6

To properly set up AVR Studio for u8glib, please follow the following link:

<https://code.google.com/p/m2tklib/wiki/as6>

Appendix B.

Source Code

“dump.py”

```
# Python script for FSR sensor system
# Python 2.7, Android 4.0.3
#
# System overview:
#
# Arduino + BT shield <--> Python for Android <--> web view
#
# Python for Android --> file in /storage/emulated/0/shared_file -->
Octave script
#
# Python for Android <-- file in /storage/emulated/0/shared_output <--
Octave script
#
# Dong Yang, 2015-1-2

import android
import time
import datetime
import random
import json
import os
import sys, string
import socket
import select

#DEBUG_PYTHON_OCTAVE = True
DEBUG_PYTHON_OCTAVE = False
octaveInstalled = True

DATA_NUM = 8
# buffer for real time plotting
PLOT_DATA = []
# data channel number supported
PLOT_DATA_NUM = DATA_NUM
# prepare matrix for data plotting
for i in range(PLOT_DATA_NUM):
    PLOT_DATA.append([])
# total points in graph
TOTAL_POINTS = 30
# feedback message length from Octave
OCTAVE_OUTPUT_SIZE = 19

#fmt = '%Y-%m-%d %H:%M:%S'
# time stamp format
fmt = '%H:%M:%S'
# file handle
f = 0
```

```

shared_foler = '/storage/emulated/legacy/'
file_to_octave = 'shared_file'
file_from_octave = 'shared_output'
CMD = 'a'

if DEBUG_PYTHON_OCTAVE:
    webviewDir = 'file:///'+shared_foler+'sl4a/scripts/graph.html'
else:
    webviewDir =
'file:///data/data/com.android.python27/files/graph.html'

sensorNumOpt = ["1","2","3","4","5","6","7","8"]
sensorNumOptInt = range(1, 8+1)
sensorNumDefault = 7
sampleNumOpt = ["8","16","32"]
sampleNumOptInt = [8,16,32]
sampleNumDefault = 0
classNumOpt = ["2","3","4","5","6","7","8"]
classNumOptInt = range(2, 8+1)
classNumDefault = 0

def get_device_list(droid):
    droid.dialogCreateHorizontalProgress('Search For Bluetooth
Devices', 'Progress', 100)
    droid.dialogShow()

    device_list = []

droid.eventRegisterForBroadcast("android.bluetooth.device.action.FOUND"
, True)
droid.bluetoothDiscoveryStart() # start BT discovery
timeout = 0
while timeout<2000:
    timeout = timeout + 1
    droid.dialogSetCurrentProgress(timeout/20)
    event = droid.eventPoll(1).result # new device detected!
    if not event:
        continue
    print event

    if event[0] != None:
        bt_str = event[0]['data']
        if bt_str.find('null')>=0:
            print '[WARNING]: replace null with None in device
name'

            print 'original:', bt_str
            bt_str = bt_str.replace('null', 'None')
            print 'new:      ', bt_str
            device_list.append(eval(bt_str))
        else:
            break

droid.dialogDismiss()

```



```

droid.eventUnregisterForBroadcast("android.bluetooth.device.action.FOUND")
    return device_list

def cleanUpExit():
    global octaveInstalled

    print octaveInstalled
    if octaveInstalled:
        # clean up
        file_open_failure = False
        try:
            f = open(shared_folder+file_to_octave, 'w', 0)
        except IOError:
            print 'File open failed'
            file_open_failure = True

        if not file_open_failure:
            f.close()
    else:
        sock.close()
        sock2.close()

    sys.exit()

def getUserChoice(title, options, defaultNum, positive_text):
    droid.dialogCreateAlert(title)
    droid.dialogSetSingleChoiceItems(options, defaultNum)
    droid.dialogSetPositiveButtonText(positive_text)
    droid.dialogShow()
    r = droid.dialogGetResponse().result
    r = droid.dialogGetSelectedItem().result[0]
    droid.makeToast("%s: %s" % (title, options[r]))
    while True: # Wait for events from the menu.
        response=droid.eventWait(10000).result
        if response==None:
            break
        if response["name"]=="dialog":
            break
    return r

droid = android.Android()

if not DEBUG_PYTHON_OCTAVE:
    octaveName = ""
    octaveVer = ""
    r = droid.getPackageVersion("com.octave")
    if r.error == None:
        octaveName = r.result
    ver = droid.getPackageVersionCode("com.octave")
    if r.error == None:
        octaveVer = r.result

```

```

    droid.makeToast("Octave Packet Name: %s Version: %s" % (octaveName,
octaveVer))

    if octaveName!="" and octaveVer!="" and octaveName!=None and
octaveVer!=None:
        droid.startActivity("android.intent.action.MAIN", None, None,
None, False, 'com.octave', 'com.octave.octaveMain')
    else:
        octaveInstalled = False

if octaveInstalled:
    try:
        f = open(shared_foler+file_to_octave, 'w', 0)
        f.close()
    except IOError:
        print 'File open failed'
else:
    UDP_IP_UBUNTO = "192.168.56.102"
    UDP_IP_RCV = "192.168.56.101"
    UDP_PORT = 12345
    UDP_PORT_RCV = 12346

    sock = socket.socket(socket.AF_INET, # Internet
        socket.SOCK_DGRAM) # UDP

    sock2 = socket.socket(socket.AF_INET, # Internet
        socket.SOCK_DGRAM) # UDP

    try:
        sock2.bind((UDP_IP_RCV, UDP_PORT_RCV))
        sock2.setblocking(0)
    except socket.error, e:
        print e

sensorNum = 4
sampleNum = 5
classNum = 6

if not DEBUG_PYTHON_OCTAVE:
    sensorNumInt = getUserChoice("Sensor Number", sensorNumOpt,
sensorNumDefault, "Conitnue")
    sensorNum = sensorNumOptInt[sensorNumInt]
    sampleNumInt = getUserChoice("Sample Number", sampleNumOpt,
sampleNumDefault, "Conitnue")
    sampleNum = sampleNumOptInt[sampleNumInt]
    classNumInt = getUserChoice("Class Number", classNumOpt,
classNumDefault, "Conitnue")
    classNum = classNumOptInt[classNumInt]

droid.toggleBluetoothState(True) # turn on bluetooth

d_list = get_device_list(droid)
if d_list == []:
    droid.makeToast("No Bluetooth Device Found, Exit")
    cleanUpExit()

```

```

dev_list = []
for d in d_list:
    dev_list.append("Name: %s, Address: %s" %
(d["android.bluetooth.device.extra.NAME"],
d["android.bluetooth.device.extra.DEVICE"]))

devNum = getUserChoice("Bluetooth Device List", dev_list, 0, "Connect")
deviceName = d_list[devNum]["android.bluetooth.device.extra.NAME"]
deviceAddr = d_list[devNum]["android.bluetooth.device.extra.DEVICE"]

ret = droid.bluetoothConnect('00001101-0000-1000-8000-00805F9B34FB',
deviceAddr)
if ret.error == None:
    droid.makeToast("Bluetooth Device (%s) Successfully Connected" %
deviceName)
else:
    droid.makeToast("Bluetooth Device (%s) Connect Failed, Exit" %
deviceName)
    cleanUpExit();

droid.webViewShow(webviewDir)

time.sleep(2)

# send start info to webview
da={}
da['name'] = 'start'
da['sensorNum']=sensorNum
da['sampleNum']=sampleNum
da['classNum']=classNum
droid.eventPost("respond",json.dumps(da))

while True:
    datai = []
    cpos = 0
    # check data from BT
    if droid.bluetoothReadReady().result == True:
        try:
            MESSAGE = str(droid.bluetoothReadLine().result)
        except UnicodeEncodeError:
            print "UnicodeEncodeError"
            continue
        # process BT data
        da = MESSAGE.split(',');
        if len(da)==8 and da[0]!='':
            for i in range(DATA_NUM):
                try:
                    datai.append(int(da[i]))
                except ValueError:
                    print da

            d = datetime.datetime.now()
            d_string = '%s:%6d\r' % (d.strftime(fmt),d.microsecond)
            #print datai, d_string

```

```

# initialization
if CMD=='a':
    datai[0] = sensorNum
    datai[1] = sampleNum
    datai[2] = classNum

if octaveInstalled:
    file_open_failure = False
    try:
        f = open(shared_foler+file_to_octave, 'w', 0)
    except IOError:
        print 'File to Octave open failed'
        file_open_failure = True

    if not file_open_failure:
        # write BT data to file
        f.write("%c%4d,%4d," % (CMD, datai[0], datai[1]))
        f.write("%4d,%4d," % (datai[2], datai[3]))
        f.write("%4d,%4d," % (datai[4], datai[5]))
        f.write("%4d,%4d," % (datai[6], datai[7]))
        f.write(d_string)
        f.write("\r");
        f.close()
    else:
        try:
            sock.sendto(CMD, (UDP_IP_UBUNTO, UDP_PORT));
            for i in range(DATA_NUM):
                sock.sendto("%4d," % datai[i], (UDP_IP_UBUNTO,
UDP_PORT))

                if i==DATA_NUM-1:
                    sock.sendto(d_string, (UDP_IP_UBUNTO,
UDP_PORT));
        except socket.error, e:
            print e

if CMD=='s' or CMD=='t' or CMD=='r':
    CMD = 'd'

from_octave_failure = False
if octaveInstalled:
    try:
        f = open(shared_foler+file_from_octave, 'r', 0)
    except IOError:
        print 'File from Octave open failed'
        from_octave_failure = True

    if not from_octave_failure:
        # get Octave result from file
        f.seek(0, os.SEEK_END)
        size = f.tell()
        if (size>=OCTAVE_OUTPUT_SIZE):
            try:
                f.seek(-OCTAVE_OUTPUT_SIZE, os.SEEK_END)
            except IOError:
                from_octave_failure = True

```

```

        rline = f.readline()
        if len(rline) != OCTAVE_OUTPUT_SIZE:
            from_octave_failure = True
        else:
            from_octave_failure = True
    else:
        try:
            rline, svr_info = sock2.recvfrom(1024) # buffer size is
1024 bytes
        except socket.error, e:
            from_octave_failure = True

    if not from_octave_failure:
        #print rline
        c = rline[0]
        rs = rline[1:].split(',')
        try:
            num = int(rs[0])
        except ValueError:
            print rs
        if c=='c':
            cpos = num
        elif c=='s':
            da = {}
            da['name'] = 'msg'
            da['data'] = 'Training for Class#' + str(num+1)
            droid.eventPost("respond", json.dumps(da));
        elif c=='t':
            da = {}
            da['name'] = 'msg'
            da['data'] = 'Class#' + str(num) + ' Done!'
            droid.eventPost("respond", json.dumps(da));
        elif c=='a':
            CMD = 'd'
            # scape this round
            continue
        elif c=='x':
            break

    if octaveInstalled:
        f.close()

# send result to webview
if CMD!='a' and len(datai)>0:
    # display results
    da={}
    da['name'] = 'data'
    da['cp']=cpes
    da['wh']=datai
    droid.eventPost("respond",json.dumps(da))

e = droid.eventWaitFor('check', 100).result
if e==None:
    continue
ei = eval(e['data']);

```

```
if ei['name']=='train next':
    CMD = 's'
if ei['name']=='train current':
    CMD = 'r'
elif ei['name']=='predict':
    CMD = 't'
# exit if user clicked quit
elif ei['name']=='quit':
    CMD = 'x'

cleanUpExit()
```

“graph.html”

```
<html>
  <head>
    <title></title>
    <!--
      <script language="javascript" type="text/javascript"
src="jquery.min.js"></script>
      <script language="javascript" type="text/javascript"
src="jquery.flot.min.js"></script>
    -->
    <script language="javascript" type="text/javascript"
src="file:///data/data/com.android.python27/files/jquery.min.js"></scri
pt>
    <script language="javascript" type="text/javascript"
src="file:///data/data/com.android.python27/files/jquery.flot.min.js"><
/script>
    <!-- (c) 2011 Emant Pte Ltd -->
  </head>
  <body>
    <h1>FSR Graph</h1>
    <div id="egraph"
style="width:500px;height:300px;"></div>
    <ul>Current Parameter: <ul id="pa"></ul>
    <h1>Current Class: <h1 id="cp"/></h1>
    <div>
      <button id="Quit" onClick='quit();'
style="height:50px; width:100px">Quit</button>
      <button id="Train Next"
onClick='startTrainNext();' style="height:50px; width:100px">Train
Next</button>
      <button id="Train Current"
onClick='startTrainCurrent();' style="height:50px; width:100px">Train
Cur</button>
      <button id="predict" onClick='startPredict();'
style="height:50px; width:100px">Predict</button>
    </div>
    <textarea id="msgTextBox" cols="30" rows="1"
style="height:50px; width:400px; font-size: 2em; font-weight: bold;
font-family: Verdana, Arial, Helvetica, sans-serif; border: 1px solid
black"></textarea>
    <script type="text/javascript">
      var data = [];
      var res = [];
      var total_data_points = 30;
      var data_num_max = 8;
      var data_num = 8;
      var timerOn = 0;
      var timerInterval = 100;
      var myFlag = true;
      var droid = new Android();

      var options = {
```

```

        series: {
            shadowSize: 0 // Drawing is faster
without shadows
        },
        yaxis: {
            min: 0,
            max: 1024
        },
        xaxis: {
            show: false
        }
    };

    // initialize matrix for data plotting
    for (var i=0;i<data_num;i++) {
        data[i] = [];
        res[i] = [];
    }

    function getBtData(wh) {
        for (var i=0; i<data_num; i++) {
            if (data[i].length >
total_data_points)
                data[i] =
data[i].slice(1);

            data[i].push(wh[i]);

            // Zip the generated y values
            res[i] = [];
            for (var j = 0; j <
data[i].length; ++j) {
                res[i].push([j,
data[i][j]])
            }
        }

        return res;
    }

    var display = function(result) {
        var json_data = eval("(" + result.data +
        ")");
        if(json_data.name=='data') {
            if(myFlag) {
                $.plot($("#egraph"),
getBtData(json_data.wh), options);
            }
            document.getElementById("cp").innerHTML = json_data.cp;
        } else if(json_data.name=='msg') {
            document.getElementById("msgTextBox").value = json_data.data;
        } else if(json_data.name=='start') {

```



```

        data_num = json_data.sensorNum;
        for (var i=0;i<data_num;i++) {
            data[i] = [];
            res[i] = [];
        }
        document.getElementById("pa").innerHTML
= "Sensor#: " + json_data.sensorNum + ", Sample#: " +
json_data.sampleNum + ", Class#: " + json_data.classNum;
    }
}

droid.registerCallback("respond", display);

function quit(){
    clearInterval(timerOn);
    droid.eventPost("check",
{"\name\":"quit\"});
}

function startTrainNext(){
    droid.eventPost("check",
{"\name\":"train next\"});
}

function startTrainCurrent(){
    droid.eventPost("check",
{"\name\":"train current\"});
}

function startPredict(){
    droid.eventPost("check",
{"\name\":"predict\"});
}
}
</script>
</body>
</html>

```

“octaveMain.java” diff output

```
23d22
< import java.io.FileOutputStream;
26d24
< import java.io.InputStream;
28d25
< import java.io.OutputStream;
38d34
< import android.content.res.AssetManager;
154,191d149
<     private void copyAsset(String filename, String target_name) {
<         AssetManager assetManager = getAssets();
<         InputStream in = null;
<         OutputStream out = null;
<         try {
<             in = assetManager.open(filename);
<             File outFile = new
File(Environment.getExternalStorageDirectory().getAbsolutePath() +
"/freeRoot", target_name);
<             out = new FileOutputStream(outFile);
<             copyFile(in, out);
<         } catch (IOException e) {
<             Log.e("tag", "Failed to copy asset file: " + filename,
e);
<         }
<         finally {
<             if (in != null) {
<                 try {
<                     in.close();
<                 } catch (IOException e) {
<                     // NOOP
<                 }
<             }
<             if (out != null) {
<                 try {
<                     out.close();
<                 } catch (IOException e) {
<                     // NOOP
<                 }
<             }
<         }
<     }
<
<     private void copyFile(InputStream in, OutputStream out) throws
IOException {
<         byte[] buffer = new byte[1024];
<         int read;
<         while((read = in.read(buffer)) != -1){
<             out.write(buffer, 0, read);
<         }
<     }
<
200,204d157
```

```
<             } else {
<                 //delete freeRoot to get a clean update
<                 exec("rm -f " +
Environment.getExternalStorageDirectory().getAbsolutePath()+"/freeRoot/
octaverc");
<                 exec("rm -f " +
Environment.getExternalStorageDirectory().getAbsolutePath()+"/freeRoot/
k.m");
<                 exec("rm -f " +
Environment.getExternalStorageDirectory().getAbsolutePath()+"/freeRoot/
LDA.m");
206,210d158
<
<             // copy script
<             copyAsset("octaverc", ".octaverc");
<             copyAsset("k.m", "k.m");
<             copyAsset("LDA.m", "LDA.m");
```

“octaverc”

```
cd /storage/sdcard0/freeRoot  
k  
exit
```

“k.m”

```
% Octave/Matlab script for data processing using LDA algorithm
% getting data from /storage/emulated/0/shared_file
% writting result to /storage/emulated/0/shared_output
%
% Dong Yang, 2014-1-2

% turn off buffer while print
more off

CLASS_NUM = 2;
SENSOR_NUM = 8;
SAMPLE_NUM = 8;

DATA_LEN = 56;

time_str = '00:00:00:000000';
home_dir_socket = '/home/octave/Octave'
cmd_file_str = '/storage/emulated/legacy/shared_file';
output_file_str = '/storage/emulated/legacy/shared_output';
cmd = '0';
cnt = 0;
posture = 0;
tData = [];
w = [];
y = [];

home_dir = pwd;
if strcmp(home_dir, home_dir_socket)
    comm_socket = 1
else
    comm_socket = 0
endif

if comm_socket==1
    ANDROID_IP = "192.168.56.101";
    ANDROID_PORT = 12346;
    server_info = struct("addr", ANDROID_IP, "port", ANDROID_PORT);

    s=socket(AF_INET, SOCK_DGRAM, 0);
    bind(s,12345);
    s2=socket(AF_INET, SOCK_DGRAM, 0);
    connect(s2, server_info);
    len=56;
    rcv data = [];
    rcv_len = 0;
else
    fh = fopen(output_file_str, "w");
    if !is_valid_file_id(fh)
        'shared_output is not valid'
    endif
    fclose(fh)
end
```

```

endif

testing = false;

while (1)

    % get user command
    c = kbhit(1);
    if c=='x'
        disp('Exit');
        break;
    endif

    if comm_socket==1
        [data,count]=recv(s,len);
        fflush(stdout);

        rcv_data = [rcv_data data];
        rcv_len = rcv_len + count;
        if rcv_len>DATA_LEN+1
            ['wrong data size:', rcv_len, ' ', rcv_data]
            rcv_data = [];
            rcv_len = 0;
            continue;
        elseif rcv_len==DATA_LEN+1
            line = char(rcv_data);
            rcv_data = [];
            rcv_len = 0;
        else
            continue;
        endif
    else
        f = fopen(cmd_file_str, "r");
        if !is_valid_file_id(f)
            'shared_file is not valid'
            continue;
        endif

        % get BT data
        ll = fseek(f, -DATA_LEN-2, 'eof');
        if (ll==-1)
            %'Fseek failed'
            fclose(f);
            continue;
        endif
        line = fgetl(f);
        if line==-1
            'line==-1, Error'
            continue;
        endif
        fclose(f);
    endif

    si = size(line, 2);
    if (si~=DATA_LEN)

```

```

        'Line Length Incorrect:'
        si
        line
        continue;
    endif

    %line
    t_cmd = line(1);
    %[cmd t_cmd]

    if t_cmd=='x'
        disp('Exit');
        break;
    endif

    % continue process user input
    if c=='s'
        t_cmd = c;
    elseif c=='t'
        printf('start testing');
        t_cmd = c;
    elseif c=='r'
        printf('restart training with class#%d', posture+1);
        t_cmd = c;
    endif

    if cmd == '0' && (t_cmd=='s' || t_cmd=='t' || t_cmd=='a' ||
t_cmd=='r')
        cmd = t_cmd;
        if t_cmd == 's' && (posture==0 || posture==CLASS_NUM)
            tData = [];
            y = zeros(SAMPLE_NUM, 1);
            for i=2:CLASS_NUM
                y = [y; (i-1).*ones(SAMPLE_NUM, 1)];
            end
            size(y)
            testing = false;
            posture = 0;
        elseif t_cmd=='r'
            if posture~=0
                if posture== 1
                    tData = [];
                else
                    tData = tData(1:(posture-1)*SAMPLE_NUM,:);
                    cmd = 's';
                endif
                posture--;
                testing = false;
            else
                cmd = '0'
            endif
        endif
    endif
    line = substr(line, 2, DATA_LEN-1);

```

```

% transfer data into matrix
tline = strtrunc(line, 39);
li = strsplit(line, ",");
si = size(li);
if si(1, 1)==1 && si(1, 2)==9 && strcmp(time_str, li{1,9})==0
    if comm_socket==0
        fh = fopen(output_file_str, "w");
        if !is_valid_file_id(fh)
            'shared_output is not valid'
            continue;
        endif
    endif
endif

time_str = li{1,9};
datas = str2double(strsplit(tline, ","));
if cmd=='s'
    if cnt++<SAMPLE_NUM
        tData = [tData; datas(1:SENSOR_NUM)]
        if comm_socket==1
            s_ret = sprintf('s%2d,%s', posture, time_str);
            send(s2, s_ret);
        else
            fprintf(fh, 's%2d,%s', posture, time_str);
        endif
    elseif posture<CLASS_NUM-1
        posture++;
        cmd = '0';
        cnt = 0;

        if comm_socket==1
            s_ret = sprintf('t%2d,%s', posture, time_str);
            send(s2, s_ret);
        else
            fprintf(fh, 't%2d,%s', posture, time_str)
        endif

        printf('Training for Class#%d is done, hit s again for
training the next class\r\n', posture);
    else
        posture++;
        printf('training is done, hit t start testing\r\n');

        if comm_socket==1
            s_ret = sprintf('t%2d,%s', posture, time_str);
            send(s2, s_ret);
        else
            fprintf(fh, 't%2d,%s', posture, time_str)
        endif

        cmd = '0';
        cnt = 0;
        w = LDA(tData, y)
    endif
elseif cmd=='t'
    if size(w, 2)==SENSOR_NUM+1

```



```

        testing = true;
    else
        printf('Incorrect w size: %d\r\n', size(w, 2));
    endif
    cmd = '0';
elseif cmd=='a'
    SENSOR_NUM = datas(1);
    SAMPLE_NUM = datas(2);
    CLASS_NUM = datas(3);
    cnt = 0;
    posture = 0;

    cmd = '0';

    if comm_socket==1
        s_ret = sprintf('a 0,%s', time_str);
        send(s2, s_ret);
    else
        fprintf(fh, 'a 0,%s', time_str);
    endif
    testing = false;
endif

if testing
    pyo = [1 datas(1:SENSOR_NUM)]*w'
    [pym, pyml] = max(pyo)

    if comm_socket==1
        s_ret = sprintf('c%2d,%s', pyml, time_str);
        send(s2, s_ret);
    else
        fprintf(fh, 'c%2d,%s', pyml, time_str);
    endif
endif

if comm_socket==0
    fclose(fh);
endif
else
    %Line Parse Error"
    %si
    %[time_str, " ", li{1,9}]
endif

% sleep for 100ms
usleep(100000);

endwhile

if comm_socket==1
    s.close();
    s2.close();
else
    % clean up
    fh = fopen(output_file_str, "w");

```

```
fprintf(fh, 'x 0,%s', time_str);  
if is_valid_file_id(fh)  
    fclose(fh);  
endif  
endif
```

“Atmega328_AVRStudio_Test.c”

```
/*
 * Atmega328_AVRStudio_Test.c
 *
 * Created: 2/27/2015 9:53:52 PM
 * Author: Dong Yang
 */

/* wiring oled <-> Arduino Uno:
 * 'data' <-> 11
 * 'Clk' <-> 13
 * 'a0' <-> 8
 * 'Rst' <-> 9
 * 'Cs' <-> 10
 */
#include "u8g.h"

#include <avr/io.h> //This contains definitions for all the
registers locations and some other things, must always be included

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "LDA_operators.h"

#define BAUDRATE 9600
#define MYUBRR (((((F_CPU * 10) / (16L * BAUDRATE)) + 5) / 10) - 1)

#define STATE_IDLE (0)
#define STATE_TRAIN (1)
#define STATE_PREDICT (2)

static int uart_putchar(unsigned char c, FILE *stream);
static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL,
_FDEV_SETUP_WRITE);

float lda_class[NCLASS][NSAMPLE][DIM];
int class_elements[NCLASS];

float lda_priors[NCLASS];
float lda_coeff[NCLASS][DIM+1];

int gClass[NSAMPLE];

float lda_result[NSAMPLE][NCLASS];

uint16_t adc_value[6]; //Variable used to store the value
read from the ADC
char buffer[5]; //Output of the itoa function
int nClass = 0;
int nSample = 0;
int nState = STATE_IDLE;
```

```

int pClass = 0;

u8g_t u8g;
char str_buf1[13] = "";
char str_buf2[13] = "";

void adc_init(void);           //Function to initialize/configure the
ADC
uint16_t read_adc(uint8_t channel); //Function to read an arbitrary
analog ic channel/pin
void USART_init(void);        //Function to initialize and
configure the USART/serial
void USART_send( unsigned char data); //Function that sends a char
over the serial port
void USART_putstring(char* StringPtr); //Function that sends a
string over the serial port

void u8g_setup(void)
{

u8g_SetPinOutput(PN(1,5));u8g_SetPinOutput(PN(1,3));u8g_SetPinOutput(PN
(1,2));u8g_SetPinOutput(PN(1,1));u8g_SetPinOutput(PN(1,0));
//u8g_InitSPI(&u8g, &u8g_dev_ssd1306_128x64_hw_spi, PN(1, 5), PN(1,
3), PN(1, 2), PN(1, 1), PN(1,0));
u8g_InitHWSPI(&u8g, &u8g_dev_ssd1306_adafruit_128x64_hw_spi, PN(1,
2), PN(1, 0), PN(1, 1));

u8g_SetColorIndex(&u8g, 1);
u8g_SetFont(&u8g, u8g_font_8x13);
}

void sys_init(void)
{
#ifdef __AVR__
/* select minimal prescaler (max system speed) */
CLKPR = 0x80;
CLKPR = 0x00;
#endif
}

static int uart_putchar(unsigned char c, FILE *stream)
{
if (c == '\n')
uart_putchar('\r', stream);

while(!(UCSR0A & (1<<UDRE0)));
UDR0 = c;

return 0;
}

void initLDA()
{
int n=0;
int i=0;

```

```

nClass = 0;

// Initializing Global Variables
for(n=0;n<NCLASS;n++)
{
    class_elements[n]=0;

    for(i=0; i<NSAMPLE_PER_CLASS; i++)
    {
        gClass[n*NSAMPLE_PER_CLASS+i] = n;
    }
}

void draw_project_name(void)
{
    u8g_SetFont(&u8g, u8g_font_6x10);
    //u8g_DrawStr(&u8g, 0, 15, "Hello World!");
    u8g_DrawStr(&u8g, 0, 20, "Wearable Electronic");
    u8g_DrawStr(&u8g, 0, 40, "Platform Project for");
    u8g_DrawStr(&u8g, 0, 60, "FSR Strap");
}

void draw_author(void)
{
    u8g_SetFont(&u8g, u8g_font_6x10);
    //u8g_DrawStr(&u8g, 0, 15, "Hello World!");
    u8g_DrawStr(&u8g, 0, 20, "SIMON FRASER");
    u8g_DrawStr(&u8g, 0, 40, "UNIVERSITY");
    u8g_DrawStr(&u8g, 0, 60, "version 1.2.1");
}

void clearScreen()
{
    u8g_FirstPage(&u8g);
    do
    {
        } while ( u8g_NextPage(&u8g) );
}

void draw_initial()
{
    int cnt = 0;

    while(cnt++<100)
    {
        u8g_FirstPage(&u8g);
        do
        {
            draw_project_name();
        } while ( u8g_NextPage(&u8g) );
    }

    cnt = 0;
}

```

```

while(cnt++<100)
{
    u8g_FirstPage(&u8g);
    do
    {
        draw_author();
    } while ( u8g_NextPage(&u8g) );
}

void draw_lda()
{
    if(nState == STATE_IDLE)
    {
        sprintf(str_buf1, "Ready for");
        sprintf(str_buf2, "Training");
    }
    else if(nState == STATE_TRAIN)
    {
        sprintf(str_buf1, "Training");
        sprintf(str_buf2, "for c# %d", nClass);
    }
    else if(nState == STATE_PREDICT)
    {
        sprintf(str_buf1, "Predicted", pClass);
        sprintf(str_buf2, "class# %d", pClass);
    }

    u8g_FirstPage(&u8g);
    do
    {
        u8g_SetFont(&u8g, u8g_font_10x20);
        u8g_DrawStr(&u8g, 0, 20, str_buf1);
        u8g_DrawStr(&u8g, 0, 40, str_buf2);
    } while ( u8g_NextPage(&u8g) );
}

int main(void){ //In ANSI C, the main function as always an
int return and using void will give you an warning
    int i,j,n; // To ease code readability I use i for samples, j for
samples dimensions, n for classes
    char c;
    float lda_temp[NCLASS];
    int adc = 0;

    //DDRB = (1 << PB5);
    //Configure timer
    // Target Timer Count = (((Input Frequency / Prescaler) / Target
Frequency) - 1)
    //Input frequency is 16Mhz
    //Target frequency is 10Hz
    TCCR1B = (1<<CS12); //1:256 prescaler

    sys_init();

```

```

u8g_setup();

adc_init();          //Setup the ADC
USART_init();       //Setup the USART

draw_initial();

while(1) {
    //Read timer value and act according with it
    if(TCNT1 < 6249){          //Our pre-calculated timer count
        continue;
    }

    //PORTB ^= (1<<PB5);      //Toggle the led state
    TCNT1 = 0;                //Reset the timer value

    c = 0;
    if (UCSR0A & (1 << RXC0))
    {
        c = UDR0; // Fetch the received byte value into the
variable "ByteReceived"
        //printf("%c ", c);
    }

    if(c=='s')
    {
        initLDA();
        nState = STATE_TRAIN;
    }
    else if(c=='t')
    {
        nState = STATE_TRAIN;
    }
    else
    {
        draw_lda();
    }

    for(i=0; i<6; i++) {
        adc = read_adc(i);
        adc_value[i] = adc;          //Read one ADC channel
        if(nSample<NSAMPLE_PER_CLASS && nState==STATE_TRAIN) {

lda_class[gClass[nClass*NSAMPLE_PER_CLASS+nSample]][class_elements[nClass]]
ss]][i] = adc;
        }
        //printf("%4d,", adc);
    }

    if(nSample<NSAMPLE_PER_CLASS && nState==STATE_TRAIN) {
        class_elements[nClass]++;
        nSample++;
        if(nSample==NSAMPLE_PER_CLASS) {
            nClass++;
            nState = STATE_IDLE;

```

```

        if(nClass>=NCLASS)
        {
            // In case we dont have prior probability info, we
determine class probability
            // as #class elements / #total samples
            for(n=0;n<NCLASS;n++) lda_priors[n] =
(float)class_elements[n]/NSAMPLE;

            // Running the LDA Coefficients calculation
//lda_coefficients(lda_coeff,lda_class,class_elements,lda_priors);
lda_coefficients_N(lda_coeff,lda_class,class_elements,lda_priors);
            nClass = 0;
            nState = STATE_PREDICT;
        }
        nSample = 0;
    }
}
#endif
if 1
    if(nState!=STATE_PREDICT)
    {
        //printf("\r\n");
        continue;
    }

    for(j=0; j<NCLASS; j++)
    {
        lda_temp[j] = lda_coeff[j][0];
        for(n=0; n<DIM; n++)
        {
            lda_temp[j] = lda_temp[j] +
lda_coeff[j][n+1]*adc_value[n];
        }

        if(lda_temp[0]>=lda_temp[1])
        {
            pClass = 0;
        }
        else
        {
            pClass = 1;
        }
        //printf("%4d\r\n", pClass);
#endif
    }
    return 1;
}

void adc_init(void){
    ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)); //16Mhz/128 = 125Khz
the ADC reference clock
    ADMUX |= (1<<REFS0); //Voltage reference from Avcc (5v)
    ADCSRA |= (1<<ADEN); //Turn on ADC
}

```



```

    ADCSRA |= (1<<ADSC);           //Do an initial conversion because
this one is the slowest and to ensure that everything is up and running
}

uint16_t read_adc(uint8_t channel){
    ADMUX &= 0xF0;                 //Clear the older channel that was
read
    ADMUX |= channel;             //Defines the new ADC channel to be
read
    ADCSRA |= (1<<ADSC);         //Starts a new conversion
    while(ADCSRA & (1<<ADSC));   //Wait until the conversion is
done
    return ADCW;                 //Returns the ADC value of the chosen
channel
}

void USART_init(void)
{
    UBRR0H = (uint8_t)(MYUBRR>>8);
    UBRR0L = (uint8_t)(MYUBRR);
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00); /* 8-bit data */

    stdout = &mystdout; //Required for printf init
}

```

“crout.c” diff output

```
4c4
< int LU_decompos(float **a,int n,int *indx,int *d,float *vv);
---
> int LU_decompos(double **a,int n,int *indx,int *d,double *vv);
17c17
< void LU_backsub(float **a,int n,int *indx,float *b);
---
> void LU_backsub(double **a,int n,int *indx,double *b);
29c29
< void LU_invert(float **a,int n,int *indx,float **inv,float *col);
---
> void LU_invert(double **a,int n,int *indx,double **inv,double
*col);
42c42
< float LU_determ(float **a,int n,int *indx,int *d);
---
> double LU_determ(double **a,int n,int *indx,int *d);
57,59d56
<
< #include "mycommon.h"
<
63c60
< int LU_decompos(float **a,int n,int *indx,int *d,float *vv) {
---
> int LU_decompos(double **a,int n,int *indx,int *d,double *vv) {
65c62
< float big,sum,temp;
---
> double big,sum,temp;
88c85
< for(k=0;k<j;k++) sum-=a[i][k]*a[k][j];
---
> for(k=0;k<=j;k++) sum-=a[i][k]*a[k][j];
112c109
< void LU_backsub(float **a,int n,int *indx,float *b) {
---
> void LU_backsub(double **a,int n,int *indx,double *b) {
114c111
< float sum;
---
> double sum;
133c130
< void LU_invert(float **a,int n,int *indx,float **inv,float *col) {
---
> void LU_invert(double **a,int n,int *indx,double **inv,double *col) {
143c140
< float LU_determ(float **a,int n,int *indx,int *d) {
---
> double LU_determ(double **a,int n,int *indx,int *d) {
145c142
< float res=(float)(*d);
```

```

---
> double res=(double) (*d);
148,181d144
< }
<
< int Inverse_N(float inv[DIM][DIM], float orig[DIM][DIM], int size)
< {
<     float *mo[DIM], *mi[DIM];
<     float vv[DIM];
<     int i, d;
<     int indx[DIM];
<     int ret;
<     float col[DIM];
<
<     for (i = 0; i < size; i++)
<     {
<         mo[i] = orig[i];
<         mi[i] = inv[i];
<     }
<
<     //showmat("Original Matrix:", mo, size);
<
<     ret = LU_decompos(mo, size, indx, &d, vv);
<     if (!ret)
<     {
<         printf("LU_decompos() failed, return!");
<         return 0;
<     }
<     //printf("LU decompos(): %d\r\n", ret);
<
<     //showmat("Matrix after LU:", mo, size);
<
<     LU_invert(mo, size, indx, mi, col);
<
<     //showmat("Matrix after inv:", mi, size);
<
<     return 1;

```

“LDA_operators.h”

```
// Matrices.h
// Header file for library LDA_operators.c

#define NSAMPLE 20
#define DIM 3
#define NCLASS 2
#include "mycommon.h"

void vect_mean(float mean[DIM], float v[NSAMPLE][DIM], int elements);
void vect_var(float var[DIM], float v[NSAMPLE][DIM], int elements);
void vect_cov(float cov[DIM][DIM], float v[NSAMPLE][DIM], int
elements);
void pooled_cov(float pool_cov_matrix[DIM][DIM], float
classes[NCLASS][NSAMPLE][DIM], int elements[NCLASS]);
void lda_coefficients(float coeff[NCLASS][DIM+1], float
classes[NCLASS][NSAMPLE][DIM], int elements[NCLASS], float
priors[NCLASS]);
```

“LDA_operators.c”

```
/*
*****
LDA_operators.C
*****
*/
//fcampi@sfu.ca
// Collection of N-Dimensional Matrix operators used for LDA
calculation

#include <math.h>
#include "matrices.h"
#include "LDA_operators.h"

void vect_mean(float mean[DIM], float v[NSAMPLE][DIM], int elements) {
    int i,j;
    for(j=0;j<DIM;j++) mean[j]=0;
    for(i=0;i<elements;i++)
        for(j=0;j<DIM;j++)
            mean[j] += v[i][j]/elements;
}

void vect_var(float var[DIM], float v[NSAMPLE][DIM], int elements) {
    int i,j;
    float mean[DIM];
    for(j=0;j<DIM;j++) var[j]=0;
    vect_mean(mean,v,elements);
    for(i=0;i<elements;i++)
        for(j=0;j<DIM;j++)
            var[j] += (v[i][j]-mean[j])*(v[i][j]-
mean[j])/(elements-1);
}

void vect_cov(float cov[DIM][DIM], float v[NSAMPLE][DIM], int elements)
{
    // cov(i,j) = mean(xi.*xj) - mean(xi)*mean(xj);
    int x,y,i;
    float mean[DIM];
    // Initializing Covariance Matrix
    for(x=0;x<DIM;x++) for(y=0;y<DIM;y++) cov[x][y]=0;
    // Calculating Mean Vector
    vect_mean(mean,v,elements);
    // Determining Covariance Matrix
    for(x=0;x<DIM;x++)
        for(y=0;y<DIM;y++){
            for(i=0;i<elements;i++)
                cov[x][y] +=
(v[i][x]*v[i][y])/elements;
            cov[x][y] -= mean[x]*mean[y];
            cov[x][y] = cov[x][y] * elements /
(elements-1);
        }
}
```

```

    }
}

void pooled_cov(float pool_cov_matrix[DIM][DIM],float
classes[NCLASS][NSAMPLE][DIM],int elements[NCLASS]) {
    // The pooled Covariance Matrix is a DIMxDIM matrix that is derived
    // by the combination
    // of covariance matrices for all the classed in the set of
    // samples, according to the equation:
    // PooledCov = PooledCov + ((nGroup(n) - 1) / (NSAMPLES - NCLASS)
). * cov(Input(Group, :));
    int n,x,y;
    float cov[DIM][DIM];
    // Initializing Output Matrix
    for(x=0;x<DIM;x++) for(y=0;y<DIM;y++) pool_cov_matrix[x][y]=0;

    for(n=0;n<NCLASS;n++) {
        // Calculating Covariance Matrices for each class
        vect_cov(cov,classes[n],elements[n]);
        // Deriving Pooled Covariance
        for(x=0;x<DIM;x++)
            for(y=0;y<DIM;y++)
                pool_cov_matrix[x][y] += ( ((float)elements[n]-1) /
(float) (NSAMPLE-NCLASS)) * cov[x][y];
    }
}

void lda_coefficients(float coeff[NCLASS][DIM+1], float
classes[NCLASS][NSAMPLE][DIM],
int elements[NCLASS], float priors[NCLASS]) {

    int x,y,n;
    float poolcov[DIM][DIM], poolcovInv[DIM][DIM];
    float mean[DIM], temp[DIM];

    // Calculating Pooled Cov Matrix, that is a combination of the Cov
    // Matrices for each class
    pooled_cov(poolcov, classes, elements);

    // Inverting Pooled Cov Matrix
    Inverse(poolcovInv,poolcov,DIM);
    //ShowMat("poolcovInv", poolcovInv, DIM);

    // Calculating linear discriminant coefficients :
    // Step 1: W(n,2:end) = GroupMean(n,:) * inv(PooledCov)
    // Step 2: W(n,1) = -0.5 * [W(n,2:end) * GroupMean(n,:)'] +
log(PriorProb(n));
    for(n=0;n<NCLASS;n++) {
        for(y=0;y<DIM;y++) temp[y]=0;
        vect_mean(mean,classes[n],elements[n]);

        // Step 1
        for(x=0;x<DIM;x++) for(y=0;y<DIM;y++) temp[x] += mean[y] *
poolcovInv[y][x];
    }
}

```

```

        // Step 2
        coeff[n][0] = log(priors[n]);
        for(y=0;y<DIM;y++) {
            coeff[n][0] += -0.5 * temp[y] * mean[y];
            coeff[n][y+1] = temp[y];
        }
    }

void reset_array(float a[DIM])
{
    int y;
    for(y=0;y<DIM;y++) a[y]=0;
}

void cal_temp(float temp[DIM], float mean[DIM], float
poolcovInv[DIM][DIM])
{
    int x,y;
    for(x=0;x<DIM;x++) for(y=0;y<DIM;y++) temp[x] += mean[y] *
poolcovInv[y][x];
}

void lda_coefficients_N(float coeff[NCLASS][DIM+1], float
classes[NCLASS][NSAMPLE][DIM],
                        int elements[NCLASS], float priors[NCLASS]) {

    int x,y,n;
    float poolcov[DIM][DIM], poolcovInv[DIM][DIM];
    float mean[DIM], temp[DIM];

    // Calculating Pooled Cov Matrix, that is a combination of the Cov
    Matrices for each class
    pooled_cov(poolcov, classes, elements);

    // Inverting Pooled Cov Matrix
    Inverse_N(poolcovInv,poolcov,DIM);
    //ShowMat("poolcovInv", poolcovInv, DIM);

    // Calculating linear discriminant coefficients :
    // Step 1: W(n,2:end) = GroupMean(n,:) * inv(PooledCov)
    // Step 2: W(n,1) = -0.5 * [W(n,2:end) * GroupMean(n,:)'] +
log(PriorProb(n));
    for(n=0;n<NCLASS;n++) {
        //for(y=0;y<DIM;y++) temp[y]=0;
        reset_array(temp);
        vect_mean(mean,classes[n],elements[n]);

        // Step 1
        cal_temp(temp, mean, poolcovInv);

        // Step 2
        coeff[n][0] = log(priors[n]);
        for(y=0;y<DIM;y++) {
            coeff[n][0] += -0.5 * temp[y] * mean[y];

```

```
        coeff[n][y+1] = temp[y];  
    }  
}
```


“matrices.h”

```
// Matrices.h
// Header file for library matrices.c
#include "mycommon.h"
#define DIM 2
float Determinant(float A[DIM][DIM], int size);
void Inverse(float B[DIM][DIM], float A[DIM][DIM], int size);
void Transpose(float B[DIM][DIM], float A[DIM][DIM], int size);
```

“matrices.c”

```

/*****
*****
      MATRICES.C
*****
*****/
//fcampi@sfu.ca
// Collection of N-Dimensional Matrix operators

// Note: This set of operators is planned for small environments where
// no malloc is available.
// Matrix spaces are allocated statically! This is because we do not
// want
// to use MALLOC. Since we need to build sub-adjoint matrices to
// calculate the cofactor matrix,
// without malloc we need to define them of fixed width! Of course, this
// can require high memory
// occupation if size becomes too big

#include <stdio.h>
#include <math.h>
#include "matrices.h"

/* calculate determinate of generic n-sized matrix using the recursive
Laplace's formula:
   The determinant of a matrix is the sum of any row or column the
   elements multiplied by their co-factors
   The cofactor  $i, j$  is the minor  $i, j$   $\times (-1)^{i+j}$ . In the following, we
   just choose row 0 as reference */
float Determinant(float A[DIM][DIM], int size)
{
    float sign=1, det=0, m_minor[DIM][DIM];
    int i, j, m, n, col;
        // End-of-Recursion Condition
    if (size==1)
        return (A[0][0]);
    else {
        for (col=0; col<size; col++){
            m=0; n=0;
            // Calculating "Minor Matrix" for all elements of
the first row (row 0)
            // we parse the current matrix and eliminate Row 0
and Column Col
            for (i=0; i<size; i++)
                for (j=0; j<size; j++) {
                    m_minor[i][j]=0;
                    if (i != 0 && j != col) {
                        m_minor[m][n]=A[i][j];
                        if (n<(size-2)) n++;
                        else { n=0; m++; }
                    }
                }
        }
    }
}

```

```

        det += sign * (A[0][col] * Determinant(m_minor, size-1));
        sign = -1*sign;
    }
    }
    return (det);
}

// Transpose the input matrix
void Transpose(float B[DIM][DIM], float A[DIM][DIM], int size){
    int i,j;
    for(i=0;i<size;i++)
        for(j=0;j<size;j++)
            B[i][j] = A[j][i];
}

// Calculate the inverse matrix using the Cofactors Matrix
// Methodology:1) calculate the Matrix of Minors of the input matrix A
//              2) Turn that into the Matrix of Cofactors (Apply signs
to the above)
//              3) Calculate then the Adjugate (Transpose the above)
//              4) Multiply that by 1/Determinant(A).

void Inverse(float B[DIM][DIM], float A[DIM][DIM], int size) {
    float minor[DIM][DIM], cofactor[DIM][DIM];
    float det;
    int p,q,m,n,i,j;
    // Calculate the cofactor sub-Matrix for all elements of the
Input matrix A
    for (q=0;q<size;q++)
        for (p=0;p<size;p++) {
            m=0;n=0;
            for (i=0;i<size;i++)
                for (j=0;j<size;j++){
                    if (i != q && j != p) {
                        minor[m][n]=A[i][j];
                        if (n<(size-2)) n++;
                        else { n=0;m++; }
                    }
                }
            //ShowMat("minor", minor, size-1);
            //det= Determinant(minor, size-1);
            //printf("%f\r\n", det);
            //det = dett(minor, size-1);
            //printf("%f\r\n", det);
            cofactor[q][p] = pow(-1,q + p) *
Determinant(minor, size-1);
        }

    det=Determinant(A, size);
    Transpose(B, cofactor, size);
    for (q=0;q<size;q++) for (p=0;p<size;p++) B[p][q]=B[p][q]/det ;
}

void ShowMat(char* n, float m[DIM][DIM], int size)
{

```

```

int i, j;

printf("%s\r\n", n);
for(i=0; i<size; i++)
{
    for(j=0; j<size; j++)
    {
        printf("%f ", m[i][j]);
    }
    printf("\r\n");
}
}

#if 0
float Det;
//Matlab: M=[ 1 2 3.2 1 1 ; 2 3 4.4 5 7.3 ; 1 1 0 8.2 1.4 ; 1 3 2 2 5;
0 2 1 3.1 2];
float B[5][5],M[5][5]={ {1,2,3.2,1,1}, {2,3,4.4,5,7.3},
{1,1,0,8.2,1.4}, {1,3,2,2,5}, {0,2,1,3.1,2}};
int main(){
    ShowMat("M[]", M, DIM);
    Det = Determinant(M,DIM); // Expected result -177.1280
    printf("%f\r\n", Det);
    Det = dett(M, DIM);
    printf("%f\r\n", Det);
    Inverse(B,M,DIM);
    ShowMat("B[]", B, DIM);
    Inverse_N(B,M,DIM);
    ShowMat("B[]", B, DIM);
    // Expected result from matlab (Will be slightly different in 32-
bit)
    // 0.3779   -0.3508    0.4757    0.8481   -1.3616
    // 0.2445   -0.4374    0.0846    0.5457    0.0508
    // 0.1336    0.3051   -0.1933   -0.5505    0.3312
    // -0.0309   0.0646    0.0701   -0.1684    0.1515
    // -0.2633   0.1847   -0.0967   -0.0095    0.0488
}
#endif

```

“mycommon.h”

```
#ifndef COMMON_H
#define COMMON_H

#define DIM 6
#define NCLASS 2
#define NSAMPLE_PER_CLASS (10)
#define NSAMPLE (NCLASS*NSAMPLE_PER_CLASS)

#endif
```