SMART SKIN

By

Junaid Afzal

Sheeza Ali Akram

Usama Adeel

Submitted to the Faculty of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad

in partial fulfillment for the requirements of a B.E. Degree in

Telecom Engineering

JULY 2018

ABSTRACT

To expand the range of sensing modalities for always available input systems, we introduce Smart Skin, a novel input technique that allows the skin to be used as a finger input surface. The primary goal of Smart Skin is to provide an always available mobile input system - that is, an input system that does not require a user to carry or pick a device.

This enables several interactive modalities, including button-based hierarchical navigation, list based sliding navigation (similar to an iPod/Smartphone/MID), text/number entry (e.g. Telephone Number Keypad), and gaming (e.g. Tetris, Frogger).

CERTIFICATE FOR CORRECTNESS AND APPROVAL

This is officially state that the thesis work contained in this report

"Smart Skin – Using skin as an Input Surface"

Is carried out by:

Junaid Afzal, Sheeza Ali Akram and Usama Adeel

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelors of Electrical (Telecomm) Engineering from National University of Sciences and Technology (NUST).

Approved By:

_____

Asst. Prof. Dr. Mir Yasir Umair

EE Department

Military College of Signals, NUST

DATED: July, 2018

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our Faculty, without whose unflinching support and cooperation,

a work of this magnitude would not have been possible.

And our Parents for their support.

ACKNOWLEDGEMENTS

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. Whatever we have achieved, we owe it to Him, in totality. We are also thankful to our families for their continuous moral support which makes us what we are.

We are extremely grateful to our project supervisor Asst. Prof. Dr. Mir Yasir Umair from MCS who in addition to providing us with valuable technical help and guidance also provided us moral support and encouraged us throughout the development of the project.

We are highly thankful to all of our teachers and staff of MCS who supported and guided us throughout our course work. Their knowledge, guidance and training enabled us to carry out this whole work.

Finally, we are grateful to the faculty of Electrical(Telecomm) Department of the Military College of Signals, NUST.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

IP	Internet Protocol
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity
ADB	Android Debugging Bridge
MATLAB	Matrix Laboratory
IDE	Integrated Development Environment
AUTOCAD	Automated Computer-Aided Design

CHAPTER: 01

INTRODUCTION

1 INTRODUCTION

1.1 Overview

Smart Skin is a new skin-based interface that allows users to use their own arms and hands as touchscreens by sensing different ultra-low frequencies that are generated when knocking various parts of skin. In this prototype, menu graphics are projected on to the user's forearm and palm from a projector that is embedded in an armband. This band has an acoustic detector, which decides the part of the display that has to be activated by the touch. There are changes in bone density, mass, and size along with the filtering effects from joints and soft tissues means various skin spots are acoustically discrete. The sensor goes with ultra-low frequencies to particular skin spots, permitting the system to decide the skin button pressed by the user. Following this, the prototype system makes use of a wireless technology like Bluetooth to send the commands to the android device that is controlled.

1.2 Problem Statement

Many smartphone users keep their phones near at all times, and not being able to stash a phone in a pocket due to bigger size. Secondly, looking at your smartphone while lying in bed at night could wreak havoc on your vision. According to National Institute of Cancer (NIC), cell phones emit radiofrequency energy (radio waves). Tissues nearest to the antenna can absorb this energy and can cause cancer.

Keeping this in view we aim to design an armband that projects a user interface onto the skin, enabling users to control devices without carrying them. Projection displayed will

be of low frequency which will not harm one's skin and eyes (as there will be no eye contact involved with the gadget).

1.3 Approach

- Research about the microprocessor, bio-acoustic sensors and selection of hardware for the armband.
- Wireless interfacing between the microprocessor and android device to be operated.
- Research about the backend commands of android device to operate it through microprocessor.
- Selection of sensor for sensing bio-acoustic signals and classification of frequencies on different parts of the forearm
- Design of the armband and prototype building by assembling the hardware.
- Generating touch events on android phone according to the classified frequencies.
- Testing armband on skin of different people.
- Testing with latest available android phone.
- Debugging and improvement.

1.4 Scope

This project finds its scope in industry producing gizmo accessories like smart watches etc., as Smart Skin prototype has the similar applications and provides the user a new way of operating android phone. Considering the innovation side of the prototype, not just the android phone itself but this technology can further extend to control home automation with skin.

1.5 Aim & Objectives

1.5.1 Research Objectives

- Study and classification of different frequencies produced by tapping on different parts of forearm

1.5.2 Academic Objectives

- Working in the field of Signal Processing
- Programming Skills on Raspberry Pi (Practice on Python language)
- Wireless interfacing among Raspberry Pi and other devices to be operated
- GUI Development
- Machine Learning for Classification of Frequencies

1.5.3 Commercial Objectives

- Designing of the Armband for interfacing the skin with the device to be operated

1.5.4 Other Objectives

- To boost the mobile phone companies

So through this project we wish to integrate our theoretical knowledge with practicality to gain further insight and refine our skills in all the fields mentioned above.

1.5.5 Organization

This document is divided into five main sections, including:

- First section of thesis lays the abstract which describes the main details of Smart Skin, followed by the introduction section which specifies the problem statement, approach, scope and objectives.
- Second section summarizes the literature about the various resources read online regarding the project and the previous research on the topic.
- The third section emphasizes on the design and development part which illustrate the flow diagrams of different steps involved in Smart Skin as well as description of main modules.
- The fourth section is the analysis and evaluation part which gives the detail of signal processing results obtained from analog discovery as well as results obtained on android device by generating touch events on forearm.
- The fifth section compromises of the future work, further improvements and point out additional developments which can be made to enhance the scope of project
- In Appendix A we have added the Synopsis document of the project.
- Appendix B contains the code of Arduino for the classification of frequencies
- And Appendix C contains the code for ADB in python.

CHAPTER: 02

LITERATURE REVIEW

2 LITERATURE REVIEW

The literature available for this project is explained below:

2.1 Background Study

2.1.1 Acoustic Signals

When we tap our finger on the skin of forearm, acoustic waves are produced with distinct frequencies. Some portion of the acoustic wave is radiated into the air as sound waves, will not be captured by our prototype. Among the acoustic energy transmitted through the arm, one form of waves produced are Transverse Waves, which was created by the displacement of the skin from a finger impact. These waves appeared as ripples, which propagate outward from the point of contact. The amplitude of these waveforms is correlated to both the tapping force and to the volume and compliance of soft tissues under the impact area. In general, higher amplitude transverse waves are produced by tapping on soft regions than on the boney areas, as they have negligible compliance (e.g. wrist, palm, fingers).

In addition to transverse waves, some waves are transmitted inward, towards the skeleton. These Longitudinal (compressive) Waves travel through the soft tissues of the arm, exciting the bone, which is much less deformable then the soft tissue but can respond to mechanical excitation by rotating and translating as a rigid body. This excitation vibrates soft tissues surrounding the entire length of the bone, resulting in new longitudinal waves that propagate outward to the skin. These two separate forms of conduction – transverse waves moving directly along the arm surface, and longitudinal

waves moving into and out of the bone through soft tissues – because these mechanisms carry energy at different frequencies and over different distances.

In general, propagation of higher frequencies waves is more through bone than soft tissue, and conduction of energy in bones is more and can be carried over large distances than conduction in soft tissues. For our project we do not model any mechanism for conduction of energies and are not depending on these mechanisms for our work as from the analysis and research it has been concluded that the bi acoustic sensor we used will depend only on complex pattern of acoustic waves which results from the mixture of these mechanisms.

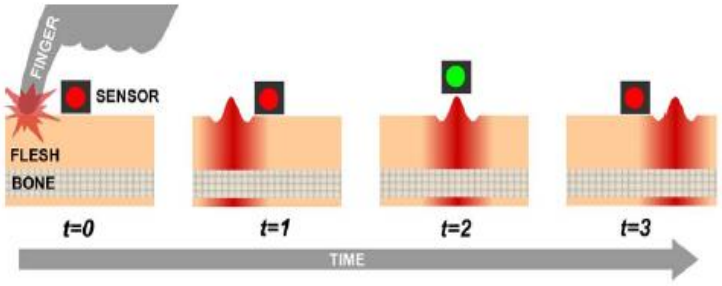

Figure 2-1: Transverse Wave Propagation

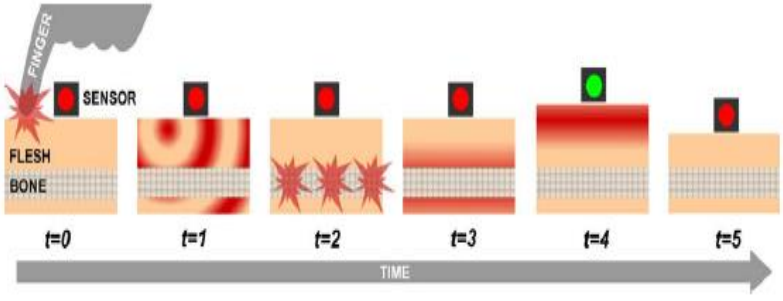

Figure 2-2: Longitudinal Wave Propagation

2.1.2 Wireless Interfacing/ Screen Sharing

Various methods were searched for sharing the screen of android phone with the Raspberry Pi wirelessly. After practicing some of the hacking techniques for sharing the screen we end up in learning the backend commands of android through ADB (android debugging bridge). Through ADB android phone can be wirelessly connected to microprocessor within a range of Bluetooth.

2.2 Existing Literature

[1]"Skinput: Appropriating the Body as an Input Surface". Microsoft Research Computational User Experiences Group. Retrieved 26 May 2010

The paper gives the concept of Skinput Technology, using our skin as an input surface. In this project, simple user interface on forearm was displayed on forearm and PC (personal computer) was used as a microprocessor. Also the frequency produced by knocking on various parts of skin are passed through the huge amplifier for classification. We are enhancing and modifying this Microsoft Research Project to solve the problem of the android user not carrying the gadgets along with them. Raspberry Pi is used instead of a PC and signal classification has to be done in Arduino by machine learning process instead of using large amplifiers making the armband easily wearable and feasible.

[2] P. Mistry, P. Maes. SixthSense – A Wearable Gestural Interface. In the Proceedings of SIGGRAPH Asia 2009, Sketch. Yokohama, Japan. 2009

The Sixth Sense project proposes a mobile, always available input/output capability by combining projected information with a color-marker-based vision tracking system. This approach is feasible, but suffers from serious occlusion and accuracy limitations. For

example, determining whether, e.g., a finger has tapped a button, or is merely hovering above it, is extraordinarily difficult. In the present work, we briefly explore the combination of on-body sensing with on-body projection.

Other approaches have taken the form of wearable computing. This typically involves a physical input device built in a form considered to be part of one's clothing. For example, glove-based input systems allow users to retain most of their natural hand movements, but are cumbersome, uncomfortable, and disruptive to tactile sensation.

CHAPTER: 03

DESIGN AND DEVELOPMENT

3 DESIGN AND DEVELOPMENT

3.1 Preliminary Design

The details for the design are given as below.

3.1.1 Technical Specification

The project consists of the following modules:

3.1.1.1 Hardware

- Raspberry Pi
- Arduino Nano
- Bio-acoustic sensor (Minisense100)
- Pico Projector
- Arm band
- Bluetooth Handsfree
- Analog Discovery 2

3.1.1.1.1 Raspberry Pi Zero

The Raspberry Pi is a series of small single-board computer. This microprocessor is used for interfacing all the components and managing their operation.

Figure 3-1: Raspberry Pi Zero

3.1.1.1.2 Arduino Nano

The Arduino Nano is a microcontroller board based on the ATmega328. It is similar to other versions of Arduino but it only lacks a DC power jack and works with a Mini-B USB cable instead of a standard one.

It converts the Analog Input from Bio-Acoustic Sensor to Digital Output. This Digital signal will be given to Raspberry Pi as it can only take distal input.

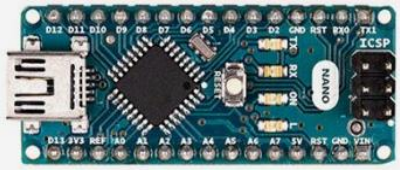

Figure 3-2: Arduino Nano

3.1.1.1.3 Minisense100:

The Minisense100 is a cantilever-type vibration sensor which provide high sensitivity at ultra-low frequencies.

It is used to sense the bio-acoustic signal frequencies produced by tapping on different parts of the forearm.

Figure 3-3: Minisense100

3.1.1.1.4 Pico Projector

Pico projectors are tiny battery powered projectors of small size as that of mobile phone or even smaller. These projectors can be embedded inside any device.

In our prototype Pico Projector is embedded inside the armband to show the GUI menu of Android Phone on the forearm.

Figure 3-4: Pico Projector

3.1.1.1.5 Bluetooth Handsfree

Bluetooth handsfree is a wireless technology for exchanging data over short distances. We used this in our project to attend calls and listen to music.

Figure 3-5: Bluetooth Handsfree

3.1.1.1.6 Analog Discovery 2

The Digilent Analog Discovery 2 is a USB oscilloscope and multi-function instrument that allows users to measure, visualize, generate, record, and control mixed-signal circuits of all kinds.

Figure 3-6: Analog Discovery 2

3.1.1.2 Software

- Python Thonny
- MATLAB
- Android Studio
- Waveform 2015
- AutoCAD

3.1.1.2.1 Python Thonny

Thonny is a Python Integrated Development Environment for beginners. It provides step-by-step code evaluation and detailed visualization of the called stack. It presents empty editor and python shell in installation package to eliminate the need for downloading and configuring other components.

It is used for Android Debugging Bridge (ADB) between the microprocessor and android phone for generating touch events on Android.

3.1.1.2.2 MATLAB

MATLAB is a desktop environment used for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly.

In MATLAB different algorithms work with our data and iteration is done until results are obtained. To automate or reproduce our work, it automatically generates a MATLAB program.

In our project we used MATLAB for Signal Processing like Peak Analysis, Average, Mean and Standard Deviation, Frequency Spectrum (FFT), Amplitude Variations and Coefficient of Variation for frequency filtration and signal classification.

3.1.1.2.3 Android Studio

Android Studio is used for making GUI for displaying the Android Phone Screen on forearm via Pico Projector.

3.1.1.2.4 Waveform 2015

It is used to acquire, visualize, analyze and interpret results from Analog Discovery 2.

3.1.1.2.5 AutoCAD

AutoCAD is a commercial automated computer-aided design (CAD) and drafting software application which is used to create 3D printable models. Several 3D printing service providers have partnership with Autodesk to print physical 3D models from 3D design.

In our project, it is used for modelling the design for the armband by considering the dimensions of all the components used in prototype.

3.2 Design Requirements and Specifications

Skinput is an input technology that uses bio-acoustic sensing to localize finger taps on the skin. When augmented with a Pico-projector, the device can provide a direct manipulation of GUI on the body.

This project appropriates the human body for acoustic transmission, allowing the skin to be used as an input surface. This skin-based interface allows users to use their own arms and hands as touch screens by sensing different ultra-low frequency sounds that are generated when knocking various parts of skin. These sound units are detected using a novel array of Bio-acoustic sensors. In our project we have targeted to use 5 sensors. Raspberry Pi and the Mobile device will be connected using Bluetooth/Wi-Fi technology.

Raspberry Pi will in turn control the functioning of the mobile phone. Pico Projector will be interfaced with Raspberry Pi to display the mobile screen on forearm.

3.3 Design Blocks

3.3.1 A/D Conversion

We will use Arduino Nano as A/D Convertor. The Bio-Acoustic Signal is first sensed and captured by Minisense100 which then send the signal to microcontroller for conversion because Raspberry Pi can't take analog input. This conversion is done by ATMega386 in Arduino Nano.

Figure 3-7: Block Diagram for A/D Conversion

3.3.2 Android Debugging Bridge

Android Debug Bridge (ADB) is a versatile command-line tool that lets us to communicate with a device. The ADB command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that is used to run a variety of commands on a device.

ADB.PY and TEST.PY files are used along with editing for executing function on Android

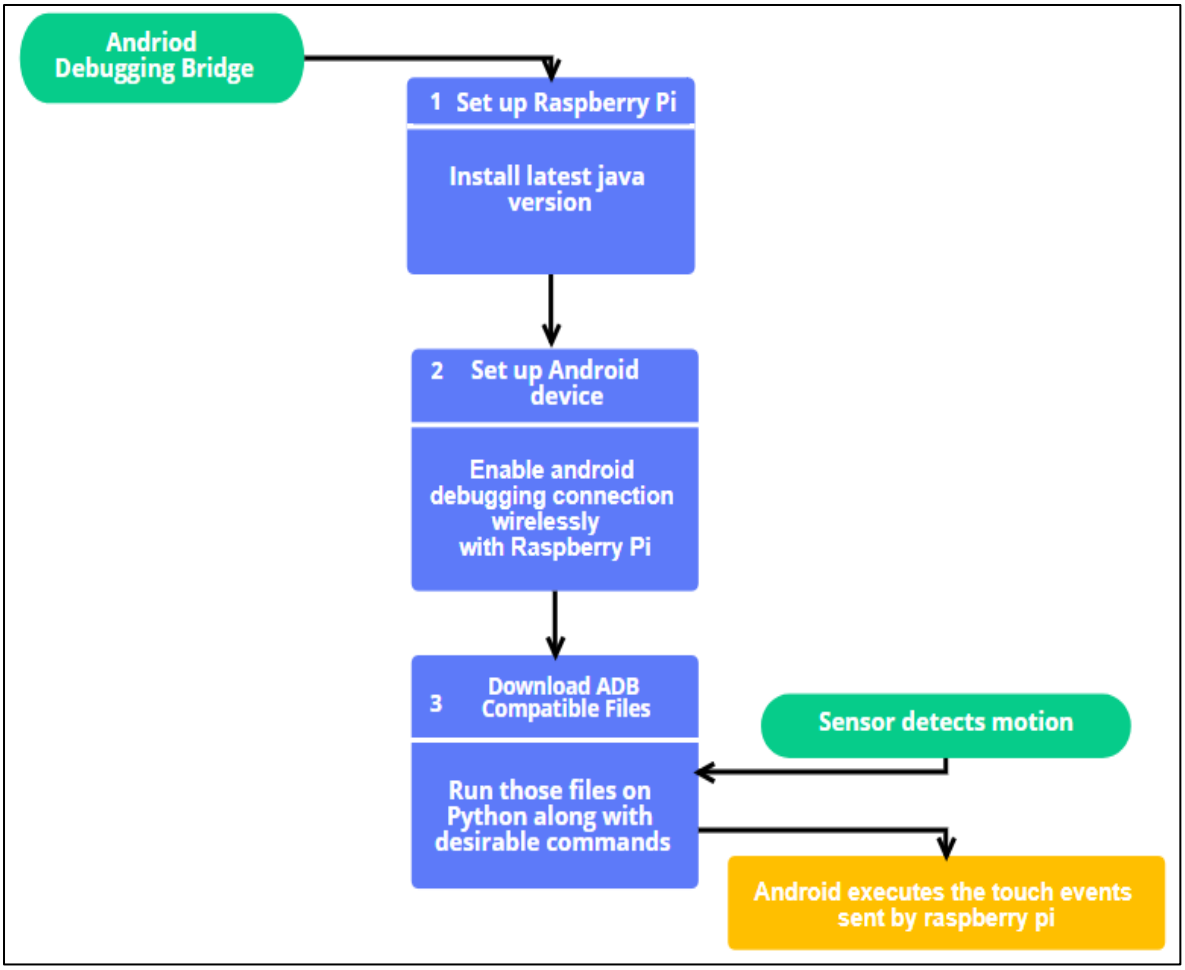

Figure 3-8: Block Diagram for Enabling Android Debugging Bridge

3.3.3 GUI Development

Android Studio is used for making Graphical User Interface development for displaying the Android Phone Screen on forearm via Pico Projector. Empty activities are made as we just have to display the user interface.

Figure 3-9: GUI Development Block Diagram

3.3.4 Signal Acquisition

Signals captured by Bio-Acoustic Sensor (Minisense100) will be send to Arduino Nano for A/D conversion and Frequency Filtration. After the Signal Processing, frequencies are classified from the correlations of the results. Digital output from will be send to Raspberry Pi which will send commands to Android Device.

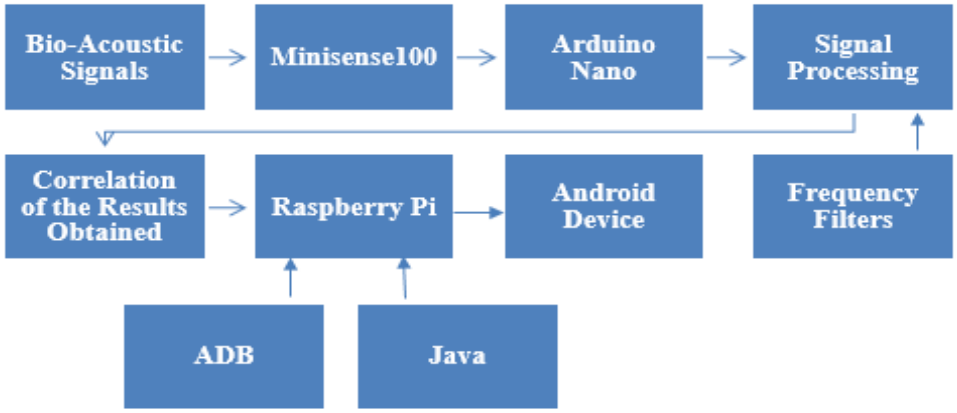

Figure 3-0-10: Block Diagram for Signal Acquisition

3.3.5 Frequency Filtration

Testing different frequencies to obtain distinct results at different skin locations so as to avoid overlapping of signals at different locations. Frequency Filtration will be done in Arduino in which machine learning algorithm is applied. Device will first take inputs from the user on various locations on the skin. The user will first train the machine according to the vibrations generated on his/her skin by touches. Machine Algorithm used six feature extractions to train the device and classifying frequencies.

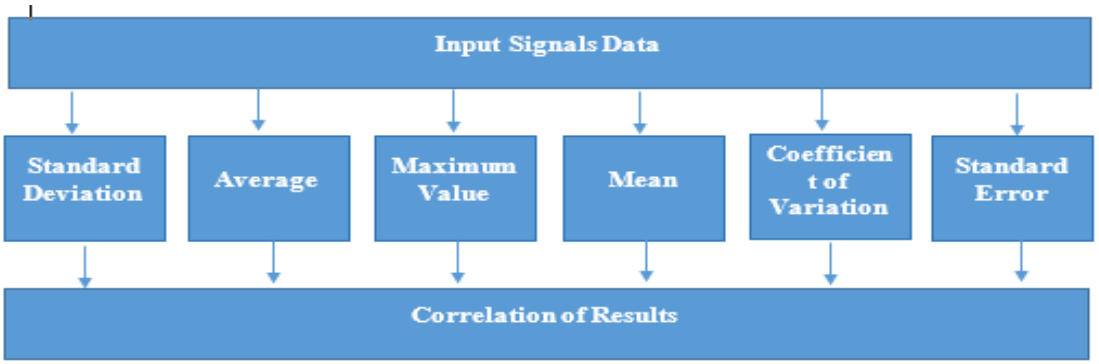

Figure 3-11: Block Diagram for Frequency Filtration

3.4 Block Diagram

This block diagram shows the basic working of the Smart Skin. Distinct frequencies produced by tapping on various parts of forearm are sensed by the array of Bio-Acoustic Sensors, from there the signals are fed into Arduino for A/D conversion and signal processing. Digital output from Arduino Nano will be send to Raspberry Pi Zero, which is wirelessly connected to Android Device though ADB. Raspberry Pi will send command to Android Device about the touch event on forearm. On the Android Phone that touch event operation will occur and the GUI of next display will be displayed on the forearm through Pico Projector.

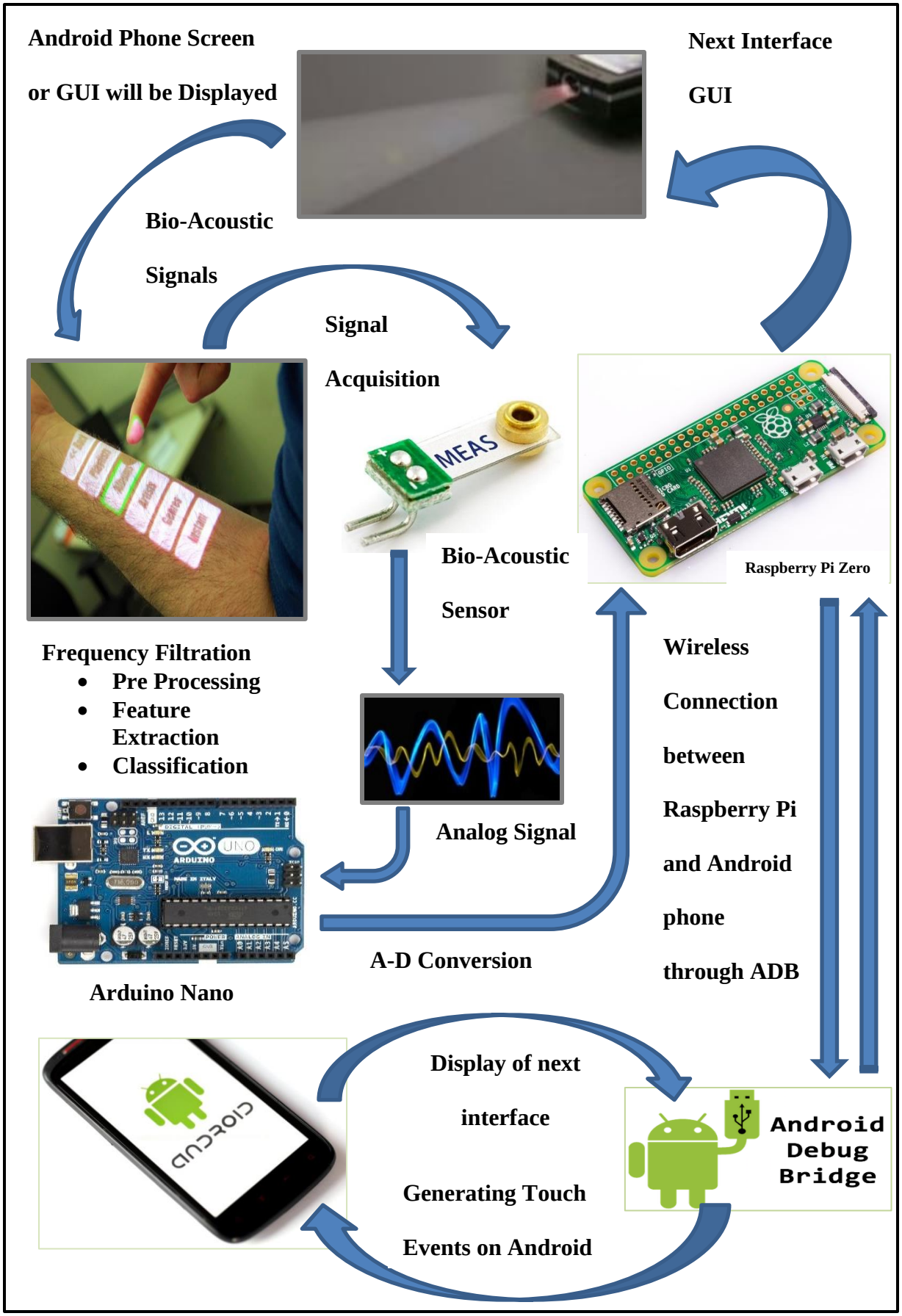

Figure 3-10-2: Block Diagram of Project

CHAPTER: 04

ANALYSIS AND EVALUATION

4 PROJECT ANALYSIS AND EVALUATION

Five input locations on the forearm and hand proved to be acoustically distinct. Input taken from these locations are easily classified by the sensors in armband. We can operate functions by tapping at the GUI of android phone projected at these 5 locations.

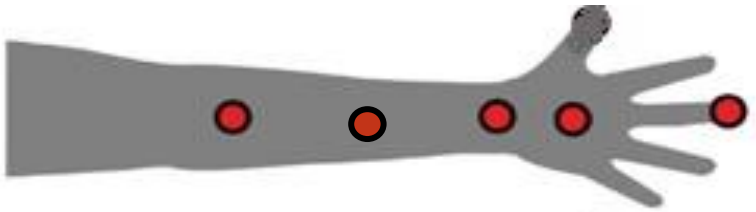

Figure 4-1: Five Input Locations on Forearm

4.1 Results from Analog Discovery

Output of the waveform on Waveform 2015 software from pocket size oscilloscope (Analog Discovery) when tapped on those 5 locations on forearm.

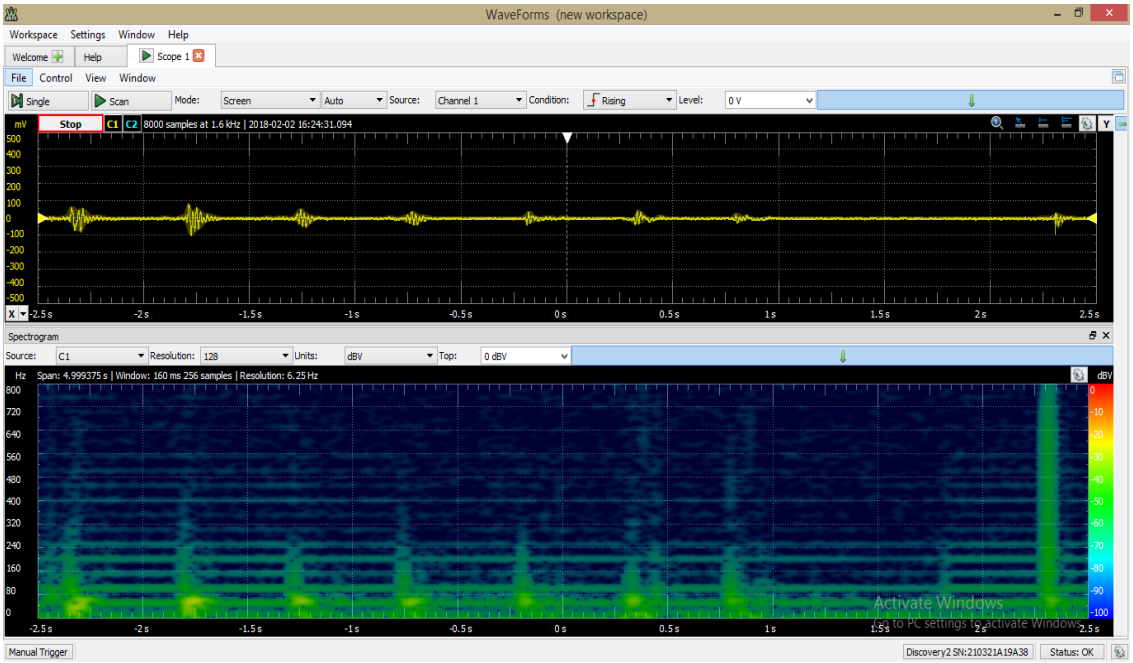

Figure 4-2: Waveform Signal obtained from Analog Discovery

4.2 Results obtained from Arduino Serial Plotter

Input locations and the result of signal waveform produced by tapping on those locations after machine learning algorithm (feature extraction and signal classification) on the Arduino Serial Plotter are given below. The x-axis of the waveform from serial plotter shows the threshold voltage and y-axis represents the frequency. The signal obtained after classification have different frequencies and threshold values.

For Location 1

The value of frequency produced by tapping on the forearm at location 1 is approximately 155Hz.

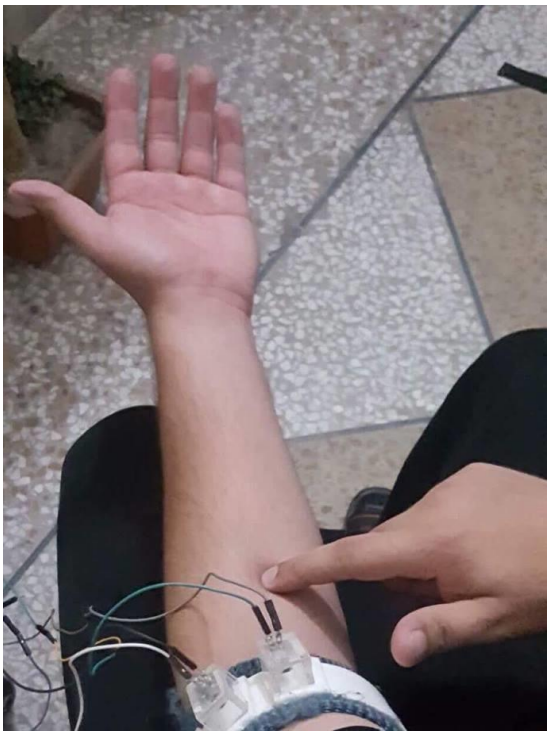

Figure 4-3: Input Location 1

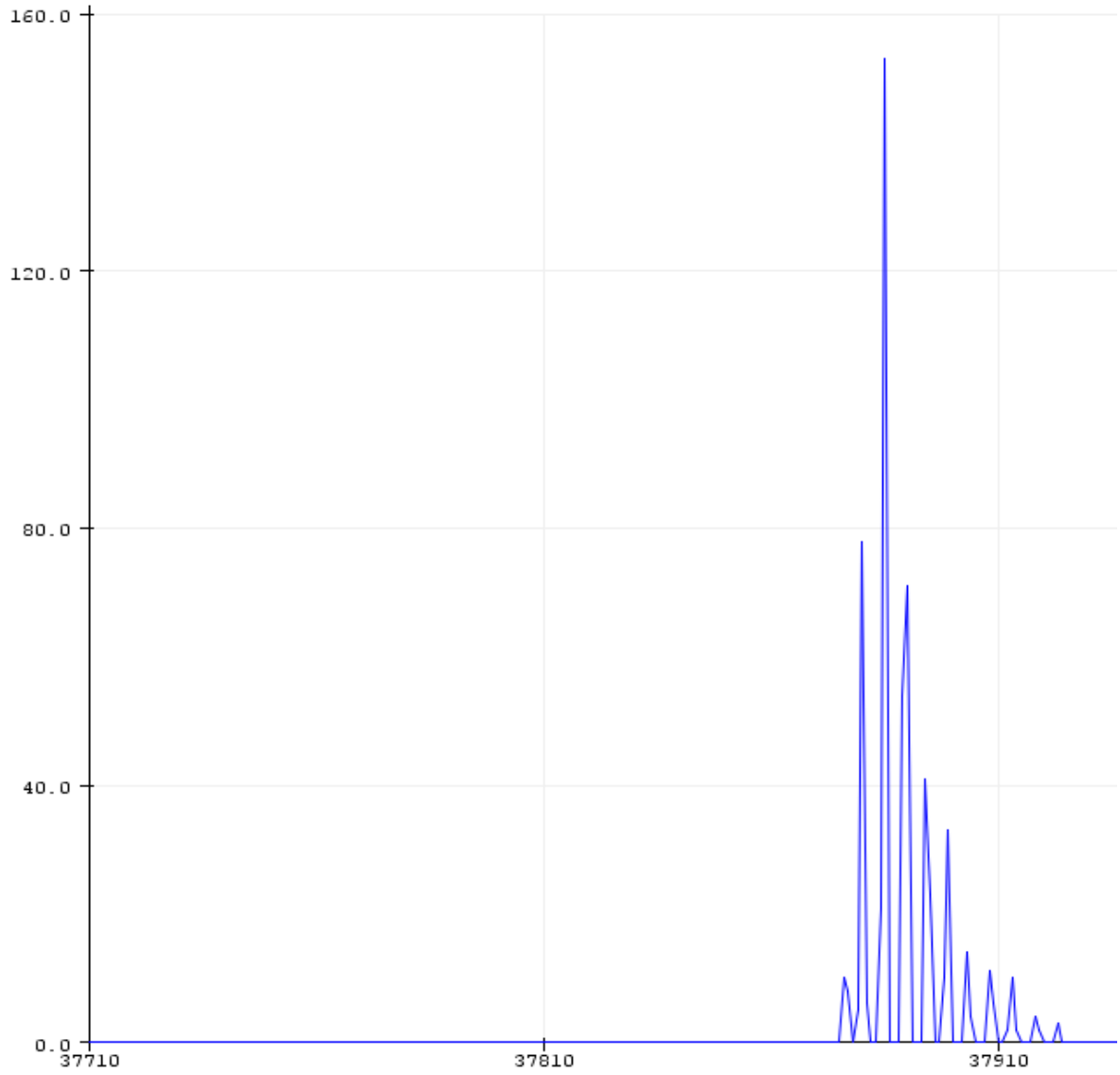

Figure 4-4: Output Signal for Location 1

For Location 2

The value of frequency produced by tapping on the forearm at location 2 is approximately 56Hz.

Figure 4-5: Input Location 2

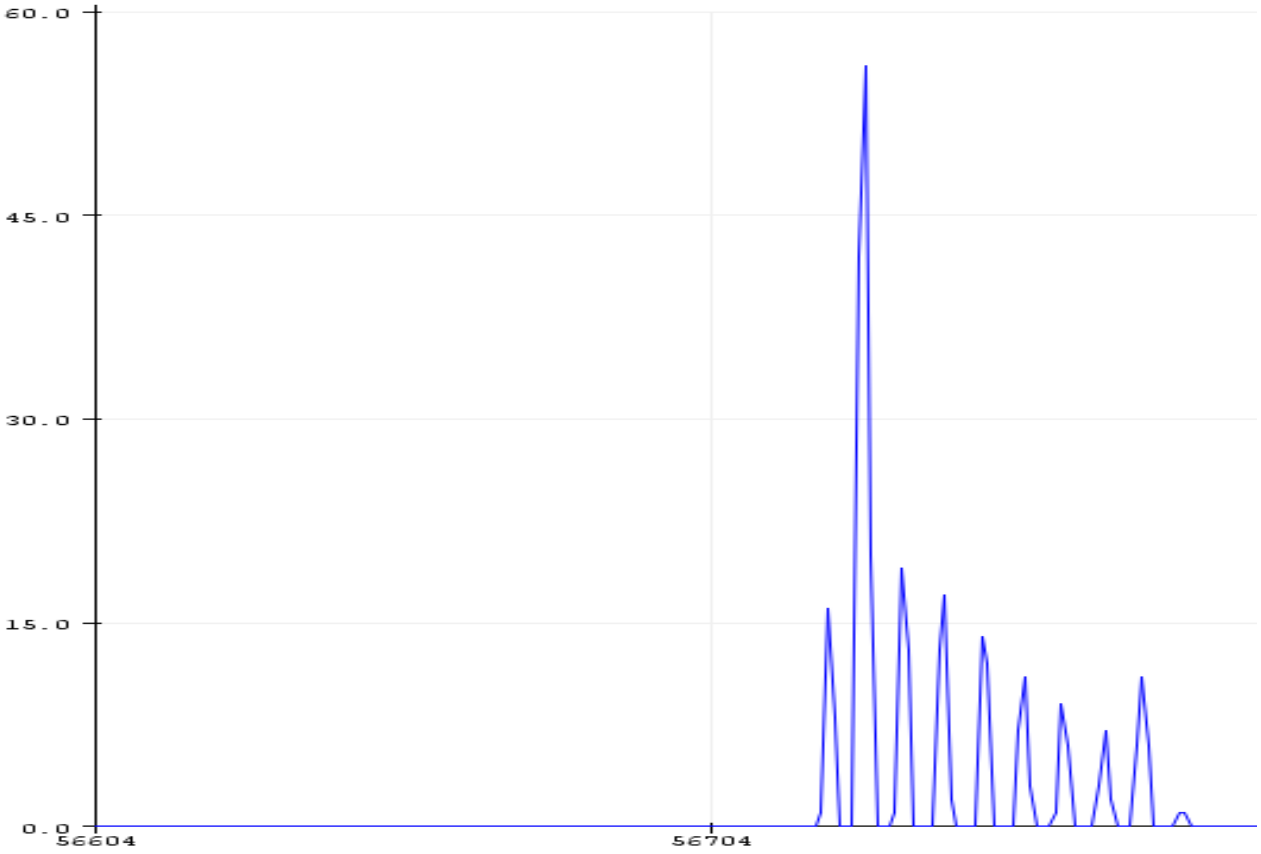

Figure 4-6: Output Signal for Location 2

For Location 3

The value of frequency produced by tapping on the forearm at location 3 is approximately 26Hz.

Figure 4-7: Input Location 3

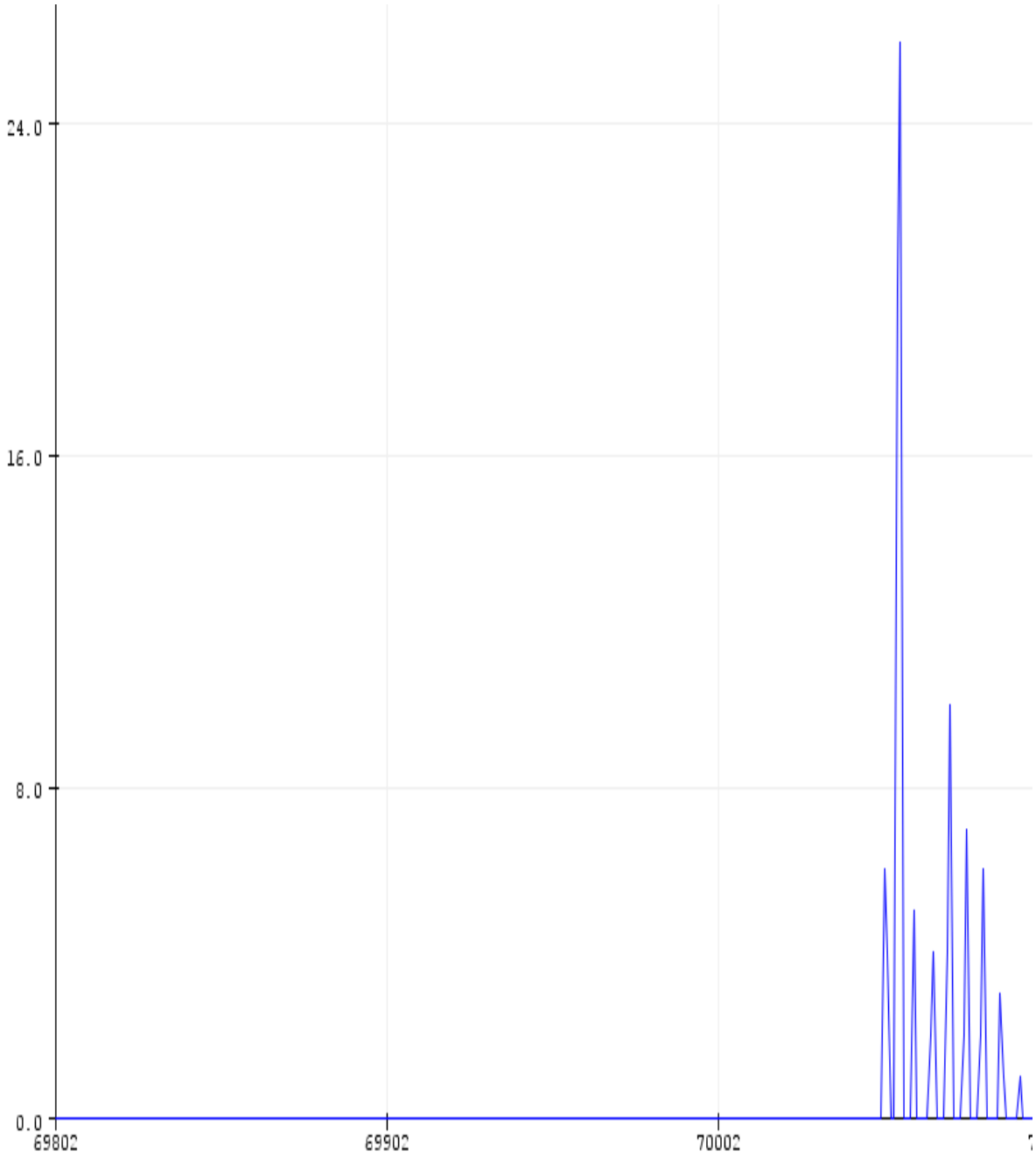

Figure 4-8: Output Signal for Location 3

For Location 4

The value of frequency produced by tapping on the forearm at location 4 is approximately 12Hz.

Figure 4-9: Input Location 4

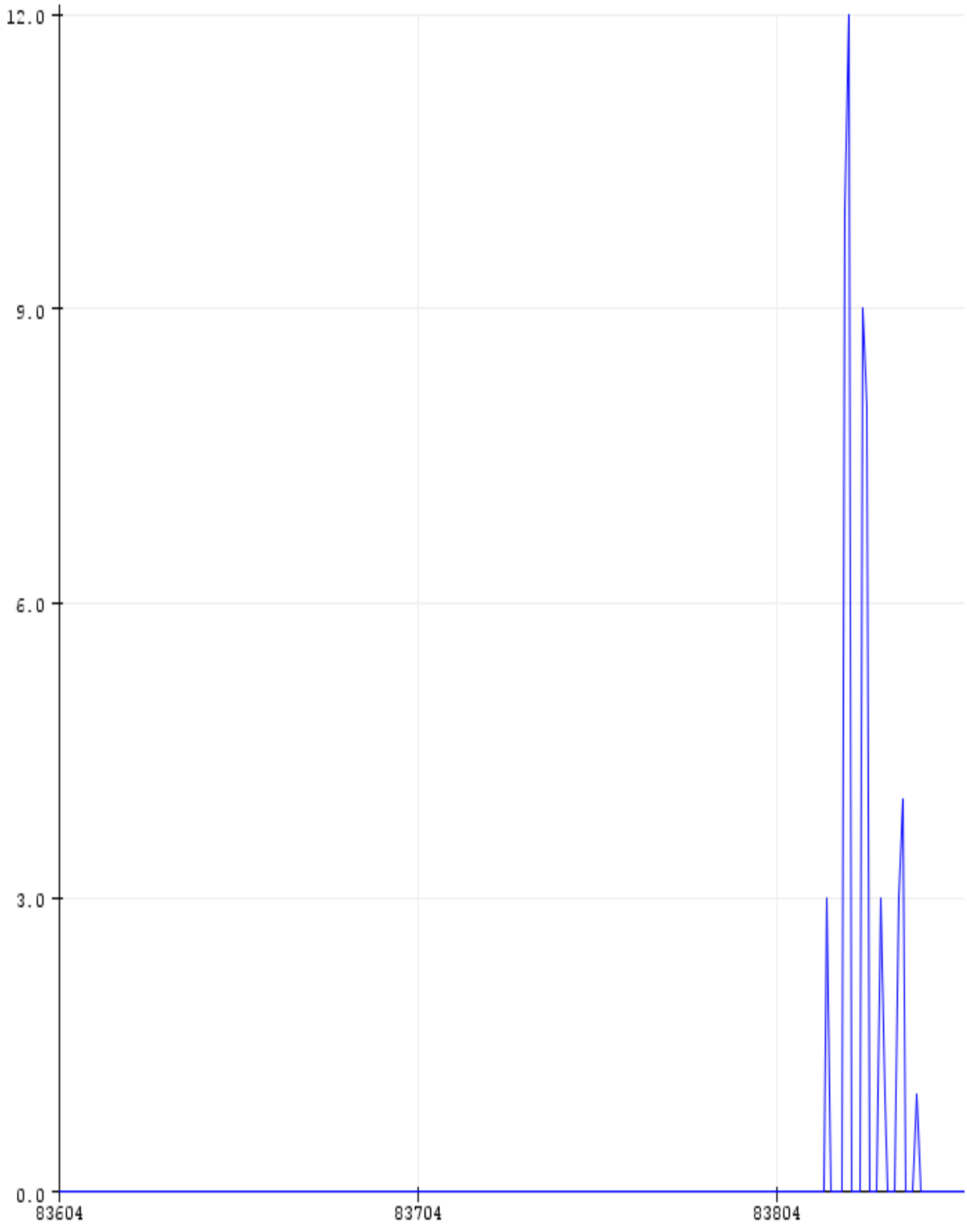

Figure 4-10: Output Signal for Location 4

For Location 5

The value of frequency produced by tapping on the forearm at location 2 is approximately 19Hz.

Figure 4-11: Input Location 5

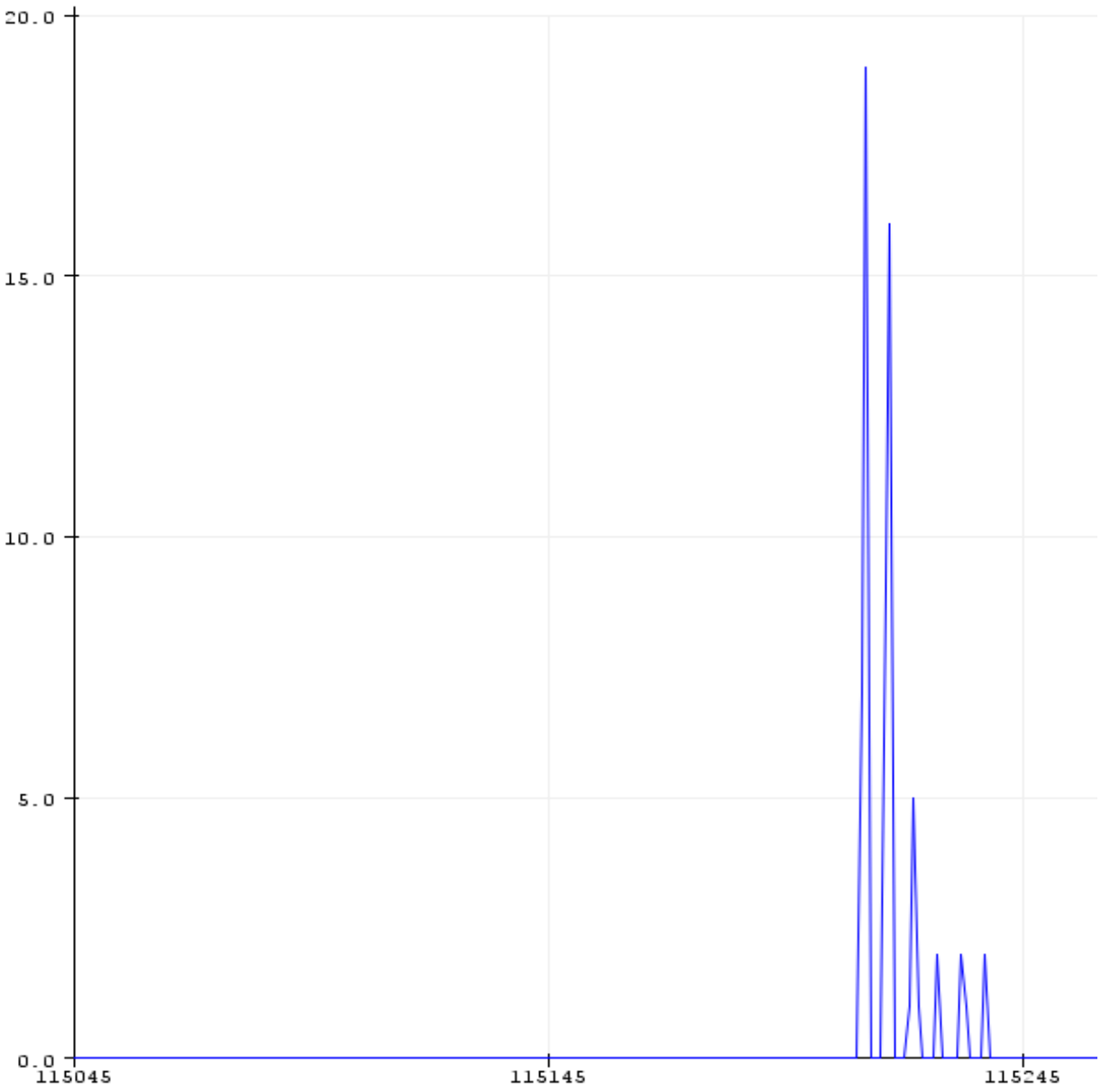

Figure 4-12: Output Signal for Location 5

4.3 Final Prototype

Array of Bio-Acoustic Sensors, Microprocessor and Microcontroller are encapsulated in the armband and final shape of the prototype is:

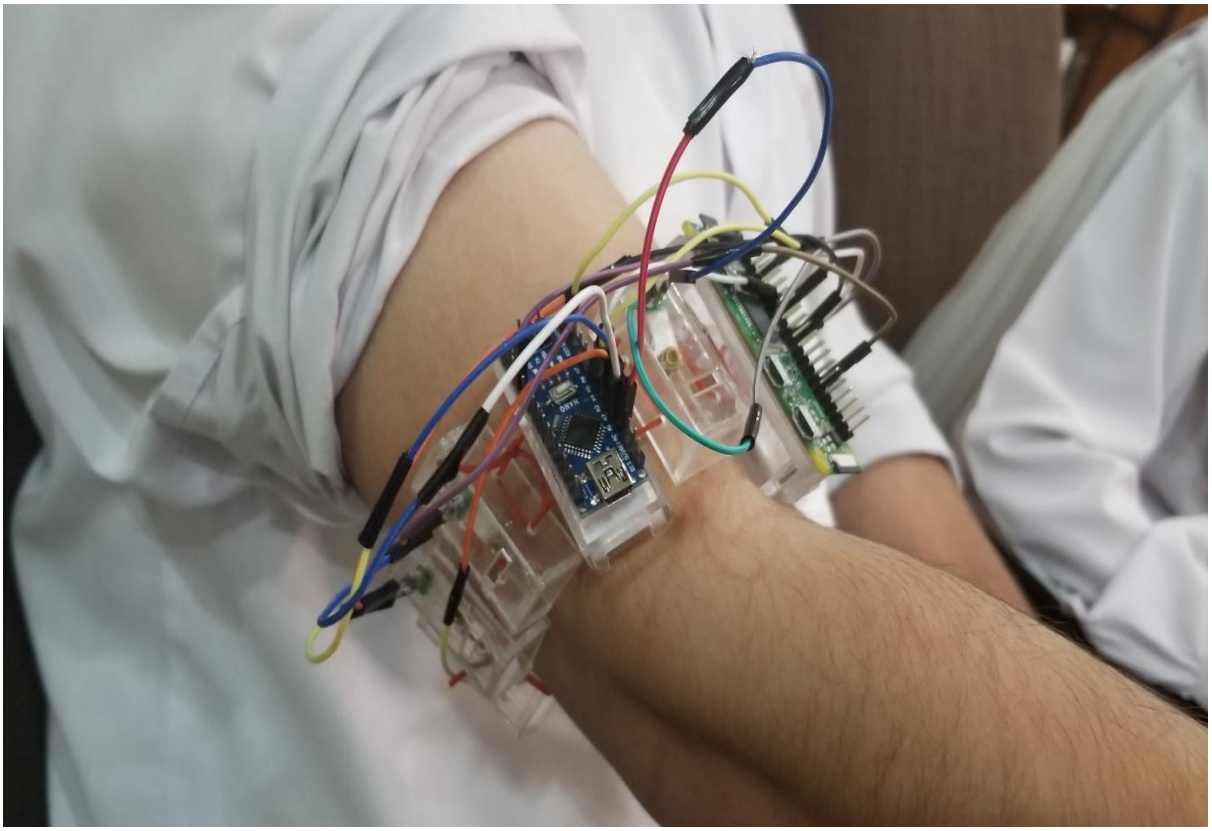

Figure 4-13: Final Prototype

CHAPTER: 05

FUTURE WORK

5 FUTURE WORK

Operating Android Device wirelessly through GUI is the basic goal and foundation of this project, but the project has some additional goals which can be achieved through further explorations and practical work.

- Better classification and filtration of frequencies obtained from forearm can be done to get more locations.
- Time delay between the touch events on forearm and events occurring on android device can be reduced by testing the prototype with different microprocessor and microcontroller, so that the time response issue between them can be resolved.
- Prototype can take the size of screen and automatically control all android phone without the predefined commands in script.
- This device can be made user specific and can be used for military communication.

CHAPTER: 06

CONCLUSION

6 CONCLUSION

6.1 Overview

The main purpose of this project is to develop an armband that would wirelessly interface the skin of forearm with the Android Device to be operated within the range of Bluetooth i.e. 100 meters.

This objective is achieved by configuration of bio-acoustic sensors (Minisense100) according to classified frequencies of those position of forearm where the finger taps on the GUI displayed. From sensors, signals are then processed in microcontroller for frequency filtration and better classification and sent to Raspberry Pi. Also, the microcontroller (Arduino) converts the analog input to digital output as Raspberry Pi can take only digital input. Raspberry Pi, that wirelessly connected to android device, send commands to the device through ADB. Display of next interface will then be displayed on the forearm. Through this prototype user can operate their android device through GUI.

6.2 Objectives Achieved/Achievements

- Wirelessly operating Android Device within the range of Bluetooth
- Compatible with most of the latest android running smart phones
- Providing an alternate way to operate gadget
- Eliminating need for carrying the android device everywhere
- Minimum risk to health

6.3 Limitations

There are certain limitations related to project:

- The android device and raspberry pi must be connected to same IP for wireless interfacing.
- This prototype can only be used with android devices.
- Accuracy issue as prototype has not been tested on various people.
- Due to time response issue between microprocessor and microcontroller delay occur.
- Issue of Arduino memory due to machine learning for larger sets of data can't be done.
- Arduino add unwanted noise to the input signal which is then filtered which further change the desired signal.

6.4 Future Research

In future, testing with the small size microprocessor will be done so that the size of the armband can be further reduced. Efforts can be done in order to increase the range by wirelessly interfacing the microprocessor and android device through Wi-Fi and through practical work frequencies on the forearm can be further classified through machine learning process so that sensor can sense more locations on forearm and we can do more operations on android device.

CHAPTER: 07

BIBLIOGRAPHY

7 BIBLIOGRAPHY

[1] Chris Harrison, Desney Tan, Dan Morris, Skinput: appropriating the body as an input surface, CHI '10 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, p.453-462, April 10 - 15, 2010, Atlanta, Georgia, USA

[2] P. Mistry, P. Maes. SixthSense – A Wearable Gestural Interface. In the Proceedings of SIGGRAPH Asia 2009, Sketch. Yokohama, Japan. 2009

[3] Ahmad, F., and Musilek, P. A Keystroke and Pointer Control Input Interface for Wearable Computers. In Proc. IEEE PERCOM '06, 2-11

[4] Deyle, T., Palinko, S., Poole, E.S., and Starner, T. Hambone: A Bio-Acoustic Gesture Interface. In Proc. ISWC '07. 1-8

[5] Lewis, R.J. Literature review of touch-screen research from 1980 to 1992. IBM Technical Report, 54.694. Aug 20, 1993

[6] Mistry, P., Maes, P., and Chang, L. WUW - wear Ur world: a wearable gestural interface. In CHI '09 Ext. Abst., 4111-4116

[7] Witten, I.H. and Frank, E. Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

CHAPTER: 08

APPENDICES

8 APPENDICES

8.1 Appendix A

Synopsis

Extended Title: USING YOUR SKIN AS INPUT SURFACE
Brief Description of The Project / Thesis with Salient Specifications: This project appropriates the human body for acoustic transmission, allowing the skin to be used as an input surface. This skin-based interface allows users to use their own arms and hands as touch screens by sensing different ultra-low frequency sounds that are generated when knocking various parts of skin. These sound units are detected using a novel array of Bio-acoustic sensors. Raspberry Pi and the Mobile device will be connected using Bluetooth/Wi-Fi technology. Raspberry Pi will in turn control the functioning of the mobile phone. Pico Projector will be interfaced with Raspberry Pi to display the mobile screen on arm.
Scope of Work: This project entails the study of vibrations generated when the finger is tapped on different parts of the forearm. We will set the frequencies of Bio-acoustic sensor in conduction with the vibrations produced. Bluetooth technology is used for the connection of Raspberry pi and Mobile phone. Programming will be done in Raspberry Pi for operating the mobile phone, displaying of the mobile phone screen through pico projector on the forearm, interfacing the skin and the mobile phone and for the connection through Bluetooth/Wi-Fi component.
Academic Objectives: • Designing of the Armband for interfacing the skin with the device to be operated • Working in the field of Signal Processing • Study and classification of different frequencies produced by tapping on different parts of forearm • Programming Skills on Raspberry Pi • Wireless Connection of Raspberry Pi and device to be operated

Application / End Goal Objectives:

- No need to interact with the gadget directly.
- New way of operating mobile phone in the absence of mobile phone
- Making phone calls by tapping the numbers flashed on your forearm
- Change the track of the music while jogging or walking
- Use of the input surface which is always available

Previous Work Done on The Subject:

In 2015 by R.Lawanya, Mrs. G.Sangeetha Lakshmi, in DKM College for Women, India

Material Resources Required:

- Raspberry Pi
- Bluetooth Module/WiFi Module
- Bio-acoustic sensor
- Pico Projector

No of Students Required: 03

Group Members:

- NC Junaid Afzal
- NC Sheeza Ali Akram
- NC Usama Adeel

Special Skills Required:

- Fabrication
- Signal Processing
- Raspberry Pi Coding

Approval Status **Supervisor Name & Signature**

_____

Assigned to:_____ **HoD Signature**
_____

R&D SC Record Status **File # _____ Coordinator Signature**
_____

8.2 APPENDIX B

Code in Arduino for Signal Processing

```
#include <ResponsiveAnalogRead.h>
 #include "QuickStats.h"
 void gm(float dd[], int rt);
 const int ANALOG_PIN1 = A1;
 const int ANALOG_PIN2 = A2;
 unsigned long time_since_last_reset = 0;
 ResponsiveAnalogRead analog1(ANALOG_PIN1, true, 0.9);
 ResponsiveAnalogRead analog2(ANALOG_PIN2, true, 0.9);
 QuickStats stats;
 struct node{
  float data;
  node* next;};
int LED_Pin1=12;
int LED_Pin2=11;
int LED_Pin3=13;
int LED_Pin4=9;
int LED_Pin5=8;
float region1[7];
float region2[7];
```

```
float region3[7];
float region4[7];
float region5[7];
int LED_Next_Region=10;
int rt;
void setup() {
 Serial.begin(1000000);
 pinMode(LED_Pin1, OUTPUT);
 pinMode(LED_Next_Region, OUTPUT);
 pinMode(LED_Pin2, OUTPUT);
 pinMode(LED_Pin3, OUTPUT);
 pinMode(LED_Pin4, OUTPUT);
 pinMode(LED_Pin5, OUTPUT);
 gm(dd1, 1);
 gm(dd2, 2);
 gm(dd3, 3);
 gm(dd4, 4);
 gm(dd5, 5);}
 void gm(float dd[], int rt)
{
 int i=0;
```

```
  node* head;
  head = NULL;
  node* temp = new node();
  temp->data=0;
  temp->next=NULL;
  head=temp;
  time_since_last_reset = millis();
  while ((millis() - time_since_last_reset) < 4000)
{
  analog1.update();
  analog2.update();
  Serial.print(analog1.getRawValue()+analog2.getRawValue());
  //Serial.print("\t");
  //Serial.print(analog1.getValue()+analog2.getValue());
  if(analog1.getRawValue() > 3 || analog2.getRawValue() > 3)
    {
    temp = new node();
    temp->data=analog1.getRawValue()+analog2.getRawValue();
    temp->next=NULL;
    i++;
    //Serial.print("\tchanged");
```

```
    node* temp1=head;
    while (temp1->next!= NULL)
    {
      temp1=temp1->next;}
      temp1->next=temp;}
      Serial.println("");
      delay(2);
    }
    node *temp2=head;
    float sl[i];
    int j=0;
    while (temp2->next!=NULL)
{
    sl[j]=temp2->data;
    j++;
    temp2=temp2->next;}
    for (int k=0;k<=j;k++)
     {
      Serial.println(sl[k]);}
    node* temp1;
    while(head != NULL)
```

```
  {
    temp1 = head;
    head = head->next;
    delete(temp1);
  }
if (rt == 1)
 {
Serial.println("Average: ");
dd1[0]=stats.average(sl,i);
Serial.println(stats.average(sl,i));
Serial.println("Maximum: ");
dd1[1]=stats.maximum(sl,i);
Serial.println(stats.maximum(sl,i));
Serial.println("Standard Deviation: ");
dd1[2]=stats.stdev(sl,i);
Serial.println(stats.stdev(sl,i));
Serial.println("Standard Error: ");
dd1[3]=stats.stderror(sl,i);
Serial.println(stats.stderror(sl,i));
Serial.println("Coefficient of Variation (%): ");
dd1[4]=stats.CV(sl,i);
```

```
Serial.println(stats.CV(sl,i));

Serial.println("Median: ");

dd1[5]=stats.median(sl,i);

Serial.println(stats.median(sl,i));

Serial.println("Variance: ");

dd1[6]=stats.stdev(sl,i)*stats.stdev(sl,i);

Serial.println(stats.stdev(sl,i)*stats.stdev(sl,i));}

else if (rt == 2)

{

Serial.println("Average: ");

dd2[0]=stats.average(sl,i);

Serial.println(stats.average(sl,i));

Serial.println("Maximum: ");

dd2[1]=stats.maximum(sl,i);

Serial.println(stats.maximum(sl,i));

Serial.println("Standard Deviation: ");

dd2[2]=stats.stdev(sl,i);

Serial.println(stats.stdev(sl,i));

Serial.println("Standard Error: ");

dd2[3]=stats.stderror(sl,i);

Serial.println(stats.stderror(sl,i));
```

```
Serial.println("Coefficient of Variation (%): ");
dd2[4]=stats.CV(sl,i);
Serial.println(stats.CV(sl,i));
Serial.println("Median: ");
dd2[5]=stats.median(sl,i);
Serial.println(stats.median(sl,i));
Serial.println("Variance: ");
dd2[6]=stats.stdev(sl,i)*stats.stdev(sl,i);
Serial.println(stats.stdev(sl,i)*stats.stdev(sl,i));}
else if (rt == 3)
{
Serial.println("Average: ");
dd3[0]=stats.average(sl,i);
Serial.println(stats.average(sl,i));
Serial.println("Maximum: ");
dd3[1]=stats.maximum(sl,i);
Serial.println(stats.maximum(sl,i));
Serial.println("Standard Deviation: ");
dd3[2]=stats.stdev(sl,i);
Serial.println(stats.stdev(sl,i));
Serial.println("Standard Error: ");
```

```
  dd3[3]=stats.stderror(sl,i);
  Serial.println(stats.stderror(sl,i));
  Serial.println("Coefficient of Variation (%): ");
  dd3[4]=stats.CV(sl,i);
  Serial.println(stats.CV(sl,i));
  Serial.println("Median: ");
  dd3[5]=stats.median(sl,i);
  Serial.println(stats.median(sl,i));
  Serial.println("Variance: ");
  dd3[6]=stats.stdev(sl,i)*stats.stdev(sl,i);
  Serial.println(stats.stdev(sl,i)*stats.stdev(sl,i));}
  else if (rt == 4)
{
  Serial.println("Average: ");
  dd4[0]=stats.average(sl,i);
  Serial.println(stats.average(sl,i));
  Serial.println("Maximum: ");
  dd4[1]=stats.maximum(sl,i);
  Serial.println(stats.maximum(sl,i));
  Serial.println("Standard Deviation: ");
  dd4[2]=stats.stdev(sl,i);
```

```
Serial.println(stats.stdev(sl,i));

Serial.println("Standard Error: ");

dd4[3]=stats.stderror(sl,i);

Serial.println(stats.stderror(sl,i));

Serial.println("Coefficient of Variation (%): ");

dd4[4]=stats.CV(sl,i);

Serial.println(stats.CV(sl,i));

Serial.println("Median: ");

dd4[5]=stats.median(sl,i);

Serial.println(stats.median(sl,i));

Serial.println("Variance: ");

dd4[6]=stats.stdev(sl,i)*stats.stdev(sl,i);

Serial.println(stats.stdev(sl,i)*stats.stdev(sl,i));}

else if (rt == 5)

 {

Serial.println("Average: ");

dd5[0]=stats.average(sl,i);

Serial.println(stats.average(sl,i));

Serial.println("Maximum: ");

dd5[1]=stats.maximum(sl,i);

Serial.println(stats.maximum(sl,i));
```

```
  Serial.println("Standard Deviation: ");
  dd5[2]=stats.stdev(sl,i);
  Serial.println(stats.stdev(sl,i));
  Serial.println("Standard Error: ");
  dd5[3]=stats.stderror(sl,i);
  Serial.println(stats.stderror(sl,i));
  Serial.println("Coefficient of Variation (%): ");
  dd5[4]=stats.CV(sl,i);
  Serial.println(stats.CV(sl,i));
  Serial.println("Median: ");
  dd5[5]=stats.median(sadlyf,i);
  Serial.println(stats.median(sl,i));
  Serial.println("Variance: ");
  dd5[6]=stats.stdev(sl,i)*stats.stdev(sl,i);
  Serial.println(stats.stdev(sl,i)*stats.stdev(sl,i));}
    digitalWrite(LED_Next_Region, HIGH);
    delay(2000);
    digitalWrite(LED_Next_Region, LOW);}
void loop() {
  int i=0;
  int l=0;
```

```
int m=0;

int n=0;

int a=0;

int b=0;

node* head;

head = NULL;

node* temp = new node();

temp->data=0;

temp->next=NULL;

head=temp;

unsigned long time_since_last_reset = 0;

time_since_last_reset = millis();

while ((millis() - time_since_last_reset) < 4000)

 {

analog1.update();

analog2.update();

Serial.print(analog1.getRawValue()+analog2.getRawValue());

//Serial.print("\t");

//Serial.print(analog1.getValue()+analog2.getValue());

if(analog1.getRawValue() > 3 || analog2.getRawValue() > 3)

  {
```

```
    temp = new node();
    temp->data=analog1.getRawValue()+analog2.getRawValue();
    temp->next=NULL;
    i++;
    //Serial.print("\tchanged");
    node* temp1=head;
    while (temp1->next!= NULL)
     {
      temp1=temp1->next;
     }
      temp1->next=temp;
}
       Serial.println("");
       delay(2);
}
      if (i != 0)
   {
      float sl[i];
      int j=0;
      node *temp2=head;
    while (temp2->next!=NULL)
```

```
  {
   sadlyf[j]=temp2->data;
   j++;
   temp2=temp2->next;
  }
   node* temp1;
       while(head != NULL)
 {
       temp1 = head;
       head = head->next;
       delete(temp1);}
 for (int k=0;k<=6;k++)
{
   if (k == 0)
  {
  if ((abs(stats.average(sl,i) - dd1[k]) < abs(stats.average(sl,i) - dd2[k])) &&
(abs(stats.average(sl,i) - dd1[k]) < abs(stats.average(sl,i) - dd3[k])) &&
(abs(stats.average(sl,i) - dd1[k]) < abs(stats.average(sl,i) - dd4[k])) &&
(abs(stats.average(sl,i) - dd1[k]) < abs(stats.average(sl,i) - dd5[k])))
  l++;
  else if ((abs(stats.average(sl,i) - dd2[k]) < abs(stats.average(sl,i) - dd1[k])) &&
(abs(stats.average(sl,i) - dd2[k]) < abs(stats.average(sl,i) - dd3[k])) &&
```

```
(abs(stats.average(sl,i) - dd[k]) < abs(stats.average(sl,i) - dd4[k])) &&
(abs(stats.average(sl,i) - dd2[k]) < abs(stats.average(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.average(sl,i) - dd3[k]) < abs(stats.average(sl,i) - dd2[k])) &&
(abs(stats.average(sl,i) - dd3[k]) < abs(stats.average(sl,i) - dd1[k])) &&
(abs(stats.average(sl,i) - dd3[k]) < abs(stats.average(sl,i) - dd4[k])) &&
(abs(stats.average(sl,i) - dd3[k]) < abs(stats.average(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.average(sl,i) - dd4[k]) < abs(stats.average(sl,i) - dd2[k])) &&
(abs(stats.average(sl,i) - dd4[k]) < abs(stats.average(sl,i) - dd1[k])) &&
(abs(stats.average(sl,i) - dd4[k]) < abs(stats.average(sl,i) - dd3[k])) &&
(abs(stats.average(sl,i) - dd4[k]) < abs(stats.average(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.average(sl,i) - dd5[k]) < abs(stats.average(sl,i) - dd2[k])) &&
(abs(stats.average(sl,i) - dd5[k]) < abs(stats.average(sl,i) - dd1[k])) &&
(abs(stats.average(sl,i) - dd5[k]) < abs(stats.average(sl,i) - dd3[k])) &&
(abs(stats.average(sl,i) - dd5[k]) < abs(stats.average(sl,i) - dd4[k])))

  b++;

}

  else if (k == 1)

  {

  if ((abs(stats.maximum(sl,i) - dd1[k]) < abs(stats.maximum(sl,i) - dd2[k])) &&
(abs(stats.maximum(sl,i) - dd1[k]) < abs(stats.maximum(sl,i) - dd3[k])) &&
(abs(stats.maximum(sl,i) - dd1[k]) < abs(stats.maximum(sl,i) - dd4[k])) &&
(abs(stats.maximum(sl,i) - dd1[k]) < abs(stats.maximum(sl,i) - dd5[k])))
```

```
 l++;

  else if ((abs(stats.maximum(sl,i) - dd2[k]) < abs(stats.maximum(sl,i) - dd1[k])) &&
(abs(stats.maximum(sl,i) - dd2[k]) < abs(stats.maximum(sl,i) - dd3[k])) &&
(abs(stats.maximum(sl,i) - dd2[k]) < abs(stats.maximum(sl,i) - dd4[k])) &&
(abs(stats.maximum(sl,i) - dd2[k]) < abs(stats.maximum(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.maximum(sl,i) - dd3[k]) < abs(stats.maximum(sl,i) - dd2[k])) &&
(abs(stats.maximum(sl,i) - dd3[k]) < abs(stats.maximum(sl,i) - dd1[k])) &&
(abs(stats.maximum(sl,i) - dd3[k]) < abs(stats.maximum(sl,i) - dd4[k])) &&
(abs(stats.maximum(sl,i) - dd3[k]) < abs(stats.maximum(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.maximum(sl,i) - dd4[k]) < abs(stats.maximum(sl,i) - dd1[k])) &&
(abs(stats.maximum(sl,i) - dd4[k]) < abs(stats.maximum(sl,i) - dd2[k])) &&
(abs(stats.maximum(sl,i) - dd4[k]) < abs(stats.maximum(sl,i) - dd3[k])) &&
(abs(stats.maximum(sl,i) - dd4[k]) < abs(stats.maximum(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.maximum(sl,i) - dd5[k]) < abs(stats.maximum(sl,i) - dd1[k])) &&
(abs(stats.maximum(sl,i) - dd5[k]) < abs(stats.maximum(sl,i) - dd2[k])) &&
(abs(stats.maximum(sl,i) - dd5[k]) < abs(stats.maximum(sl,i) - dd3[k])) &&
(abs(stats.maximum(sl,i) - dd5[k]) < abs(stats.maximum(sl,i) - dd4[k])))

  a++;

 }

 else if (k == 2)

{
```

```
  if ((abs(stats.stdev(sl,i) - dd1[k]) < abs(stats.stdev(sl,i) - dd2[k])) &&
(abs(stats.stdev(sl,i) - dd1[k]) < abs(stats.stdev(sl,i) - dd3[k])) && (abs(stats.stdev(sl,i) -
dd1[k]) < abs(stats.stdev(sl,i) - dd4[k])) && (abs(stats.stdev(sl,i) - dd1[k]) <
abs(stats.stdev(sl,i) - dd5[k])))

  l++;

  else if ((abs(stats.stdev(sl,i) - dd2[k]) < abs(stats.stdev(sl,i) - dd1[k])) &&
(abs(stats.stdev(sl,i) - dd2[k]) < abs(stats.stdev(sl,i) - dd3[k])) && (abs(stats.stdev(sl,i) -
dd2[k]) < abs(stats.stdev(sl,i) - dd4[k])) && (abs(stats.stdev(sl,i) - dd2[k]) <
abs(stats.stdev(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.stdev(sl,i) - dd3[k]) < abs(stats.stdev(sl,i) - dd2[k])) &&
(abs(stats.stdev(sl,i) - dd3[k]) < abs(stats.stdev(sl,i) - dd1[k])) && (abs(stats.stdev(sl,i) -
dd3[k]) < abs(stats.stdev(sl,i) - dd4[k])) && (abs(stats.stdev(sl,i) - dd3[k]) <
abs(stats.stdev(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.stdev(sl,i) - dd4[k]) < abs(stats.stdev(sl,i) - dd2[k])) &&
(abs(stats.stdev(sl,i) - dd4[k]) < abs(stats.stdev(sl,i) - dd1[k])) && (abs(stats.stdev(sl,i) -
dd4[k]) < abs(stats.stdev(sl,i) - dd3[k])) && (abs(stats.stdev(sl,i) - dd4[k]) <
abs(stats.stdev(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.stdev(sl,i) - dd5[k]) < abs(stats.stdev(sl,i) - dd2[k])) &&
(abs(stats.stdev(sl,i) - dd5[k]) < abs(stats.stdev(sl,i) - dd1[k])) && (abs(stats.stdev(sl,i) -
dd5[k]) < abs(stats.stdev(sl,i) - dd3[k])) && (abs(stats.stdev(sl,i) - dd5[k]) <
abs(stats.stdev(sl,i) - dd4[k])))

  b++;

}
```

```
  else if (k == 3)

  {

  if ((abs(stats.stderror(sl,i) - dd1[k]) < abs(stats.stderror(sl,i) - dd2[k])) &&
(abs(stats.stderror(sl,i) - dd1[k]) < abs(stats.stderror(sl,i) - dd3[k])) &&
(abs(stats.stderror(sl,i) - dd1[k]) < abs(stats.stderror(sl,i) - dd4[k])) &&
(abs(stats.stderror(sl,i) - dd1[k]) < abs(stats.stderror(sl,i) - dd5[k])))

  l++;

  else if ((abs(stats.stderror(sl,i) - dd2[k]) < abs(stats.stderror(sl,i) - dd1[k])) &&
(abs(stats.stderror(sl,i) - dd2[k]) < abs(stats.stderror(sl,i) - dd3[k])) &&
(abs(stats.stderror(sl,i) - dd2[k]) < abs(stats.stderror(sl,i) - dd4[k])) &&
(abs(stats.stderror(sl,i) - dd2[k]) < abs(stats.stderror(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.stderror(sl,i) - dd3[k]) < abs(stats.stderror(sl,i) - dd2[k])) &&
(abs(stats.stderror(sl,i) - dd3[k]) < abs(stats.stderror(sl,i) - dd1[k])) &&
(abs(stats.stderror(sl,i) - dd3[k]) < abs(stats.stderror(sl,i) - dd4[k])) &&
(abs(stats.stderror(sl,i) - dd3[k]) < abs(stats.stderror(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.stderror(sl,i) - dd4[k]) < abs(stats.stderror(sl,i) - dd2[k])) &&
(abs(stats.stderror(sl,i) - dd4[k]) < abs(stats.stderror(sl,i) - dd1[k])) &&
(abs(stats.stderror(sl,i) - dd4[k]) < abs(stats.stderror(sl,i) - dd3[k])) &&
(abs(stats.stderror(sl,i) - dd4[k]) < abs(stats.stderror(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.stderror(sl,i) - dd5[k]) < abs(stats.stderror(sl,i) - dd2[k])) &&
(abs(stats.stderror(sl,i) - dd5[k]) < abs(stats.stderror(sl,i) - dd1[k])) &&
(abs(stats.stderror(sl,i) - dd5[k]) < abs(stats.stderror(sl,i) - dd3[k])) &&
(abs(stats.stderror(sl,i) - dd5[k]) < abs(stats.stderror(sl,i) - dd4[k])))
```

```
 b++;
}
 else if (k == 4)
 {
 if ((abs(stats.CV(sl,i) - dd1[k]) < abs(stats.CV(sl,i) - dd2[k])) && (abs(stats.CV(sl,i) - dd1[k]) < abs(stats.CV(sl,i) - dd3[k])) && (abs(stats.CV(sl,i) - dd1[k]) < abs(stats.CV(sl,i) - dd4[k])) && (abs(stats.CV(sl,i) - dd1[k]) < abs(stats.CV(sl,i) - dd5[k])))
 l++;
 else if ((abs(stats.CV(sl,i) - dd2[k]) < abs(stats.CV(sl,i) - dd1[k])) && (abs(stats.CV(sl,i) - dd2[k]) < abs(stats.CV(sl,i) - dd3[k])) && (abs(stats.CV(sl,i) - dd2[k]) < abs(stats.CV(sl,i) - dd4[k])) && (abs(stats.CV(sl,i) - dd2[k]) < abs(stats.CV(sl,i) - dd5[k])))
 m++;
 else if ((abs(stats.CV(sl,i) - dd3[k]) < abs(stats.CV(sl,i) - dd2[k])) && (abs(stats.CV(sl,i) - dd3[k]) < abs(stats.CV(sl,i) - dd1[k])) && (abs(stats.CV(sl,i) - dd3[k]) < abs(stats.CV(sl,i) - dd4[k])) && (abs(stats.CV(sl,i) - dd3[k]) < abs(stats.CV(sl,i) - dd5[k])))
 n++;
 else if ((abs(stats.CV(sl,i) - dd4[k]) < abs(stats.CV(sl,i) - dd2[k])) && (abs(stats.CV(sl,i) - dd4[k]) < abs(stats.CV(sl,i) - dd1[k])) && (abs(stats.CV(sl,i) - dd4[k]) < abs(stats.CV(sl,i) - dd3[k])) && (abs(stats.CV(sl,i) - dd1[k]) < abs(stats.CV(sl,i) - dd4[k])))
 a++;
```

```
  else if ((abs(stats.CV(sadlyf,i) - dd5[k]) < abs(stats.CV(sl,i) - dd2[k])) &&
(abs(stats.CV(sl,i) - dd5[k]) < abs(stats.CV(sl,i) - dd1[k])) && (abs(stats.CV(sl,i) -
dd5[k]) < abs(stats.CV(sl,i) - dd3[k])) && (abs(stats.CV(sl,i) - dd5[k]) <
abs(stats.CV(sl,i) - dd4[k])))

  b++;

  }

  else if (k == 5)

  {

  if ((abs(stats.median(sl,i) - dd1[k]) < abs(stats.median(sl,i) - dd2[k])) &&
(abs(stats.median(sl,i) - dd1[k]) < abs(stats.median(sl,i) - dd3[k])) &&
(abs(stats.median(sl,i) - dd1[k]) < abs(stats.median(sl,i) - dd4[k])) &&
(abs(stats.median(sl,i) - dd1[k]) < abs(stats.median(sl,i) - dd5[k])))

  l++;

  else if ((abs(stats.median(sl,i) - dd2[k]) < abs(stats.median(sl,i) - dd1[k])) &&
(abs(stats.median(sl,i) - dd2[k]) < abs(stats.median(sl,i) - dd3[k])) &&
(abs(stats.median(sl,i) - dd2[k]) < abs(stats.median(sl,i) - dd4[k])) &&
(abs(stats.median(sl,i) - dd2[k]) < abs(stats.median(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.median(sl,i) - dd3[k]) < abs(stats.median(sl,i) - dd2[k])) &&
(abs(stats.median(sl,i) - dd3[k]) < abs(stats.median(sl,i) - dd1[k])) &&
(abs(stats.median(sl,i) - dd3[k]) < abs(stats.median(sl,i) - dd4[k])) &&
(abs(stats.median(sl,i) - dd3[k]) < abs(stats.median(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.median(sl,i) - dd4[k]) < abs(stats.median(sl,i) - dd2[k])) &&
(abs(stats.median(sl,i) - dd4[k]) < abs(stats.median(sl,i) - dd1[k])) &&
```

```
(abs(stats.median(sl,i) - dd4[k]) < abs(stats.median(sl,i) - dd3[k])) &&
(abs(stats.median(sl,i) - dd4[k]) < abs(stats.median(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.median(sl,i) - dd5[k]) < abs(stats.median(sl,i) - dd2[k])) &&
(abs(stats.median(sl,i) - dd5[k]) < abs(stats.median(sl,i) - dd1[k])) &&
(abs(stats.median(sl,i) - dd5[k]) < abs(stats.median(sl,i) - dd3[k])) &&
(abs(stats.median(sl,i) - dd5[k]) < abs(stats.median(sl,i) - dd4[k])))

  b++;}

 else if (k == 6)

{

  if ((abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd2[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd3[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd1[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd4[k])) &&
(abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd5[k])))

  l++;

  else if ((abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd2[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd3[k])) &&
(abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd4[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k])))

  m++;

  else if ((abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd3[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
```

```
dd3[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k])) &&
(abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd3[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd4[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd3[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k])))

  n++;

  else if ((abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd4[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd4[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k])) &&
(abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd4[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd3[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd4[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k])))

  a++;

  else if ((abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd2[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd5[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd1[k])) &&
(abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k]) < abs(stats.stdev(sl,i)*stats.stdev(sl,i) -
dd3[k])) && (abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd5[k]) <
abs(stats.stdev(sl,i)*stats.stdev(sl,i) - dd4[k])))

  b++;

}}

  if (l > m && l > n && l > a && l > b){

    digitalWrite(LED_Pin1, HIGH);

    delay(2);

    digitalWrite(LED_Pin1, LOW);  }

  else if (m > n && m > l && m > a && m > b){
```

```
    digitalWrite(LED_Pin2, HIGH);
    delay(2);
    digitalWrite(LED_Pin2, LOW);}
  else if (n > m && n > l && n > a && n > b){
    digitalWrite(LED_Pin3, HIGH);
    delay(2);
    digitalWrite(LED_Pin3, LOW);}
  else if (a > m && a > l && a > n && a > b){
    digitalWrite(LED_Pin4, HIGH);
    delay(2);
    digitalWrite(LED_Pin4, LOW);}
   else if (b > m && b > l && b > n && b > a)
{
    digitalWrite(LED_Pin5, HIGH);
    delay(2);
    digitalWrite(LED_Pin5, LOW);}}
    Serial.println("");
    delay(2);
}
```

8.3 APPENDIX C

Code in Python for ADB

```
import os

class ADB(object):

    def call(self, command):

        command_result = ''

        command_text = 'adb %s' % command

        results = os.popen(command_text, "r")

        while 1:

            line = results.readline()

            if not line: break

            command_result += line

        return command_result

    def devices(self):

        result = self.call("devices")

        devices = result.partition('\n')[2].replace('\n', '').split('\tdevice')

        return [device for device in devices if len(device) > 2]

    def upload(self, fr, to):
```

```
    result = self.call("push " + fr + " " + to)
    return result
def get(self, fr, to):
    result = self.call("pull " + fr + " " + to)
    return result
def install(self, param):
    data = param.split()
    if data.length == 1:
        result = self.call("install " + param[0])
    elif data.length == 2:
        result = self.call("install " + param[0] + " " + param[1])
    return result
def uninstall(self, package):
    result = self.call("shell pm uninstall " + package)
    return result
def clearData(self, package):
    result = self.call("shell pm clear " + package)
```

```
    return result

def shell(self, command):

    result = self.call("shell " + command)

    return result

def kill(self, package):

    result = self.call("kill " + package)

    return result

def start(self, app):

    pack = app.split()

    result = "Nothing to run"

    if pack.length == 1:

        result = self.call("shell am start " + pack[0])

    elif pack.length == 2:

        result = self.call("shell am start " + pack[0] + "/." + pack[1])

    elif pack.length == 3:

        result = self.call("shell am start " + pack[0] + " " + pack[1] + "/." + pack[2])

    return result
```

```
def screen(self, res):
    result = self.call("am display-size " + res)
    return result
def dpi(self, dpi):
    result = self.call("am display-density " + dpi)
    return result
def screenRecord(self, param):
    params = param.split()
    if params.length == 1:
        result = self.call("shell screenrecord " + params[0])
    elif params.length == 2:
        result = self.call("shell screenrecord --time-limit " + params[0] + " " + params[1])
    return result
def screenShot(self, output):
    self.call("shell screencap -p /sdcard/temp_screen.png")
    self.get("/sdcard/temp_screen.png", output)
    self.call("shell rm /sdcard/temp_screen.png")
```

```
from adb import ADB

debug = ADB()

import RPi.GPIO as GPIO

import time

channel1=17

channel2=27

channel3=22

channel4=10

channel5=9

GPIO.setmode(GPIO.BCM)

GPIO.setup(channel1, GPIO.IN,)

GPIO.setup(channel2, GPIO.IN,)

GPIO.setup(channel3, GPIO.IN,)

GPIO.setup(channel4, GPIO.IN,)

GPIO.setup(channel5, GPIO.IN,)

def callback1(channel1):

    if GPIO.input(channel1):
```

```
        print ("movement detected")
        debug.shell("input keyevent 26")
        debug.shell("input swipe 1000 1600 1000 3000")
def callback2(channel2):
    if GPIO.input(channel2):
        print ("movement detected")
        debug.shell("input keyevent 3")
def callback3(channel3):
    if GPIO.input(channel3):
        print ("movement detected")
        debug.shell("input tap 1272 2266")
        debug.shell("input tap 1220 1368")
def callback4(channel4):
    if GPIO.input(channel4):
        print ("movement detected")
        debug.shell("input tap 1272 2266")
        debug.shell("input swipe 1000 1500 500 2000")
```

```
        debug.shell("input tap 1220 1368")
def callback5(channel5):
    if GPIO.input(channel5):
        print ("movement detected")
        debug.shell("input swipe 1000 2200 1000 1600")
GPIO.add_event_detect(channel1, GPIO.BOTH, bouncetime=300)
GPIO.add_event_detect(channel2, GPIO.BOTH, bouncetime=300)
GPIO.add_event_detect(channel3, GPIO.BOTH, bouncetime=300)
GPIO.add_event_detect(channel4, GPIO.BOTH, bouncetime=300)
GPIO.add_event_detect(channel5, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel1, callback1)
GPIO.add_event_callback(channel2, callback2)
GPIO.add_event_callback(channel3, callback3)
GPIO.add_event_callback(channel4, callback4)
GPIO.add_event_callback(channel5, callback5)
while True:
    time.sleep(1)
```