Jiacheng Gu
Matriculation Number: S1573210
BEng Hons Project

Sensory Assistive Device for Visually-Impaired and ASD Users

27 April 2017

Original Mission Statement

Project Background:

Facial expressions and body language play a vitally important role in modern daily communications and social interaction. However, many people have difficulty in receiving or interpreting these expressions and responding properly (for example those with visual impairments or autism spectrum disorders). Limited feedback from these non-verbal cues means that these people are more likely to misunderstand others' opinions and emotions. This leads to gradual degradation in communication skills and increased social isolation.

Project Overview:

The purpose of the project is to design, build and test a system consisting of a camera, image analysis software and a wearable device with haptic feedback (Figure 1). The camera or camera glasses (this could even be the camera on the smartphone) would take real-time videos of the user's conversation partner, and transmit the video to a device with image analysis and emotion detection software [1] (usually a computer or a smartphone) via a USB cable or Bluetooth. Then the wearable device will send reconfigurable feedback to the user to intuitively inform them of the real-time emotion of the conversation partner. Primary feedback will be programmable haptic vibration patterns, but can also be extended to changing colors, patterns or sounds.

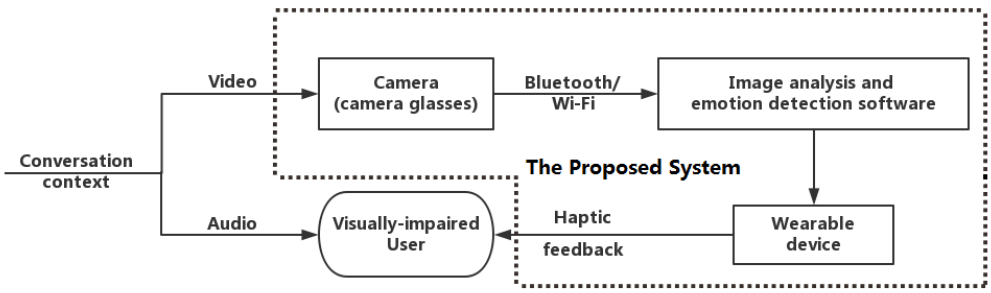


Figure 1: The block diagram for the proposed system

Project Tasks:

The general purpose of this project is to continue the efforts of previous project students and to complete the construction (in both hardware and software) of the haptic wristband device. Existing known problems will be solved, and performance improvements will be implemented.

Preparatory Tasks:

1. Learn to code ***Intel Edison*** with ***C/C***++.
2. Learn to drive vibration motor with ***drv2605***.
3. Learn to program the Android app on ***Android Studio*** by ***Java***.
4. Study the ***psychological*** relationship between colors and emotions.

Main Tasks:

1. Enable programmable gradual transitions between colors on ***OLED screen***.
2. Create a database to match vibration and display patterns with different emotions.
3. Crowd-source a questionnaire among the public on the relationships between vibration and display models with different emotions, and use machine-learning to update the database.
4. Integrate and test the haptic wristband (developed at University of Edinburgh) with the camera glasses and emotion detection software (developed by collaborators at University of St Andrews).

Extension:

1. Try to develop the app for existing wearable device like Apple watch (from Apple) or SWR50 (from SONY).
2. Investigate the possibility of using the cloud-based database for color and vibration pattern storage and ranking.

3. Perform evaluation of system performance in collaboration with collaborating psychologists at the University of Stirling.

Background Knowledge:

- C/C++ for Edison and Arduino, and java for android
- Electronics: microcontrollers; Bluetooth and wireless communication; haptic motors
- Psychology: Human perception of emotions

Resources:

- Dr. Juan Ye, University of St Andrews (Computer Science, face recognition, and machine learning)
- Dr. Ross Goutcher, University of Stirling (Psychology)
- SMC FabLab+, TLG, and TLD
- Android Studio, IDEs

Abstract

The wristband has been rebuilt, to solve the existing connection problems between the breakouts and microcontroller and enable the gradually changing between different colors when displaying color patterns on the OLED screen. The device is aimed at helping people with visual impairments or autism spectrum disorders (ASD) by giving real-time haptic and color feedback based on his/her conversation partner's current emotion states.

The Android app has been modified. The user can easily create and test the vibration and color patterns, in addition, questionnaires can be crowd-sourced on the smartphone. As the cloud service is no longer available, Excel data management is added to this app.

Declaration of Originality

I declare that this thesis is my

original work expects where stated.

Statement of Achievement

In this project, I successfully created the haptic wristband: an essential part of an advanced communication aiding system [REF] (other parts includes: Camera or camera glass, facial expression detector) for visually-impaired or ASD people. In addition, solving most of the existing bugs and problems.

The whole project includes hardware design, software design, and structural design of the wristband. I chose all the appropriate components for the wristband and integrated everything. I learned Inter-Integrated Circuit (I^2C) and Serial Peripheral Interface Bus (SPI Bus) communication based on Arduino library and write the program in *Arduino IDE* to drive the wristband.

In order to develop the Android app to work with the wristband, I learned about the Bluetooth communication between HC-05 and Android phone. I also learned to use the jxl.jar library to write, read, and edit local Excel file on an android phone.

When making the shell for the wristband, I learned how to design a 3-D object.

This project enables me to combine all kinds of knowledge together to create a real wearable device and significantly improves my Java and Android programming ability.

Context

Original Mission Statement i
Project Background: i
Project Overview: i
Project Tasks: ii
Extension: ii
Background Knowledge: iii
Resources: iii
Abstract iv
Declaration of Originality v
Statement of Achievement vi
Context vii
List of Symbols x
Glossary xi
Chapter 1 Introduction and background 1
1.1 Background 1
1.2 Identification of the Problem 1
1.2.1 Auditory Sense 2
1.2.2 Tactile Sense 2
1.2.3 Color Signals 3
1.2.4 Thermal Signals 3
1.3 Project Aims and Objectives 4
1.4 Current Hardware Status 5
1.5 Project Overview 6
Chapter 2 Hardware design 7
2.1 System Framework 7
2.2 Microcontrollers 7
2.3 Vibration Motor & Motor Driver 10
2.3.1 Vibration Motors 10
2.3.2 Motor Driver DRV2605 10
2.4 OLED Screen SSD1331 11
2.5 Bluetooth module HC-05 13
2.6 Battery & Charging Module 13
2.6.1 Battery 13
2.6.2 Charging module 14

2.7 (Reserved) Accelerometer ADXL345 & Buttons 14
2.8 Hardware System Integration 15
Chapter 3 Hardware Programming & System Design 18
3.1 System Structure 18
3.2 Bluetooth Communication 19
3.3 DRV2605 20
3.4 SSD1331 OLED Screen 22
3.5 Internal Memory on Arduino - EEPROM 25
3.6 (Reserved) ADXL345 & Buttons 27
3.6.1 ADXL345 27
3.6.2 Buttons 29
Chapter 4 Software Design 30
4.1 General Design 31
4.1.1 Bluetooth Design 31
4.1.2 Write, Read and Edit Excel File 33
4.2 Homepage 35
4.3 Wristband Settings 35
4.4 Library Vibration Patterns 36
4.5 Real-Time Vibration Patterns 40
4.6 Color Patterns 41
4.7 Questionnaires 43
4.7.1 Personal information 43
4.7.2 Select Emotion for Displayed pattern 44
4.7.3 Pattern Comparison 46
4.7.4 See Questionnaire Data 47
Chapter 5 System Integration & Testing 48
5.1 Mechanical Design 48
5.2 Assembling 48
5.3 Testing 49
5.3.1 Battery 49
5.3.2 Pattern playing and uploading 50
Chapter 6 Pathways to Impact 51
6.1 Current Position in Research & Development Process 51
6.2 Impacts of the Project 52
6.2.1 SOCIAL IMPACT 52
6.2.2 ECONOMIC IMPACT 52

6.2.3 Academic Impact 52
Chapter 7 Conclusions & Future work 53
7.1 Conclusion 53
7.2 Future Work 53
Acknowledgments 54
References 55
Appendices 58
A.1 DRV2605 Waveform Library Effects List 58
A.2 Laser Cutting Design (A4 size) 59
A.3 Example Method for Creating Library-Vibration-Pattern-Detail Excel File 60

List of Symbols

t_s: Battery life of wristband in standby mode

t_p: Battery life of wristband if keep playing vibration and color pattern

Glossary

ASD: Autism spectrum disorders

SAT: Sensory Assistive Technology

SAD: Sensory Assistive Device

LRA: Linear Resonant Actuator

ERM: Eccentric Rotating Mass (Vibration Motor)

I^2C: Inter-Integrated Circuit

SPI: Serial Peripheral Interface Bus

OLED: organic light-emitting diode

UART: Universal Asynchronous Receiver/Transmitter

USB: Universal Serial Bus

DC: Direct Current

PWM: Pulse-Width Modulation

I/O: Inputs/Outputs

EEPROM: Electrically Erasable Programmable Read-only Memory

RAM: Random-Access Memory

UI: User Interface

Chapter 1 Introduction and background

1.1 Background

Although it might be ignored by people, facial expression, and body language, is of vital significance to daily communications and social interactions. According to the Mehrabian's findings in 1967 [2] [3], nonverbal communication represents two-thirds of communication, in addition, 83% of the nonverbal communication is based on visual cues. In another word, facial expression and body language account for around 55% in social communication and interactions.

Therefore, losing or misunderstanding those visual cues might result in gradual degradation in communication skills and increasing social isolation.

Unfortunately, according to the WHO (2014) [4], 285 million people are estimated to be visually impaired worldwide of which 39 million are blind and 246 have low vision. It also showed that 80% of all visual impairment could be prevented or cured. However, there is no cure for autism spectrum disorder (ASD), which would affect one's social interaction, communication, and behaviors. CDC's Studies (Centers for Diseases Control and Prevention) in Asia, Europe, and North America have identified that 1% to 2% of individuals suffer from ASD [5]. Similarly, in the UK, about 1 person in 100 has ASD, according to a survey from NHS in 2016 [6].

Considering that such huge amount of people have visual impairments or ASD, Sensory Assistive Technology (SAT) and Sensory Assistive Device (SAD) is urgently demanded.

1.2 Identification of the Problem

Various SADs have been invited to help visually-impaired and autism. They replace visual cues with tactile sense, auditory sense, color and thermal signals.

1.2.1 Auditory Sense

One of the most popular substitutions for a visual cue is auditory sense. Numerous acoustic assistive devices are available because more and more devices have a smart audio system now. However, applications are limited because they would lead to distraction during daily conversations.

1.2.2 Tactile Sense

Tactile sense is another common solution for both visually-impaired and ASD patients. The blind began to use tactile sense to feel the world from a long time ago. After Braille is invited , people with visual-impairment could 'read' books as normal people. Nowadays, haptic devices are widely used, most phones have built-in vibration motors to give haptic feedback to the user.

Several projects focused on giving haptic feedback to the visually-impaired user currently. The VibroGlove [7] (according to *Figure 1-1*), with many small Linear Resonant Actuator (LRA) vibration motor fixed on several fingers, can play combinations of different vibration to represent different information.

However, as there are more than ten small vibration motors on one glove, it is susceptible that whether the user can easily remember the complex combination of these vibration patterns.

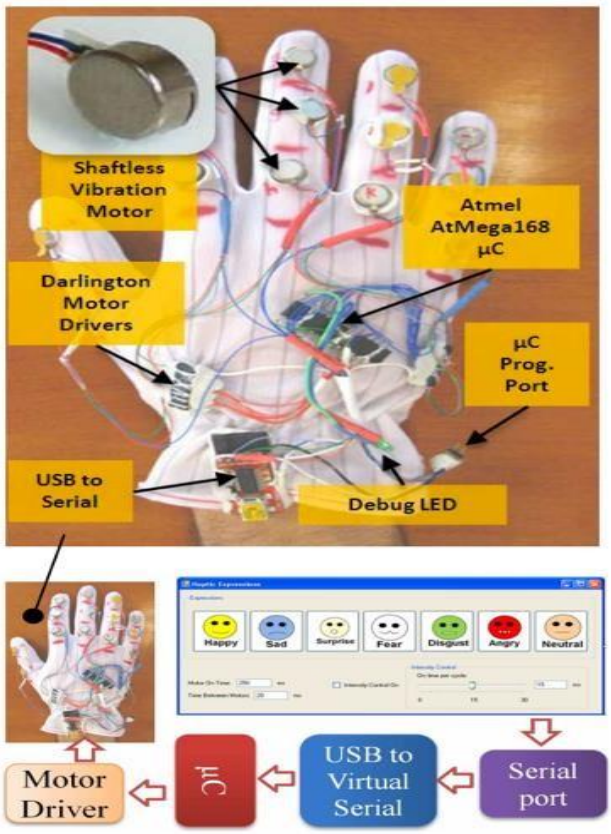


Figure 1-1: Architecture of the VibroGlove [7]

Another project is a chair [8] showed in Figure 1-2, with many vibration motors on the back.

However, it can only display four different patterns: Happy, Sad, Angry and Neutral. Ponderosity is clearly another main drawback of this haptic chair.

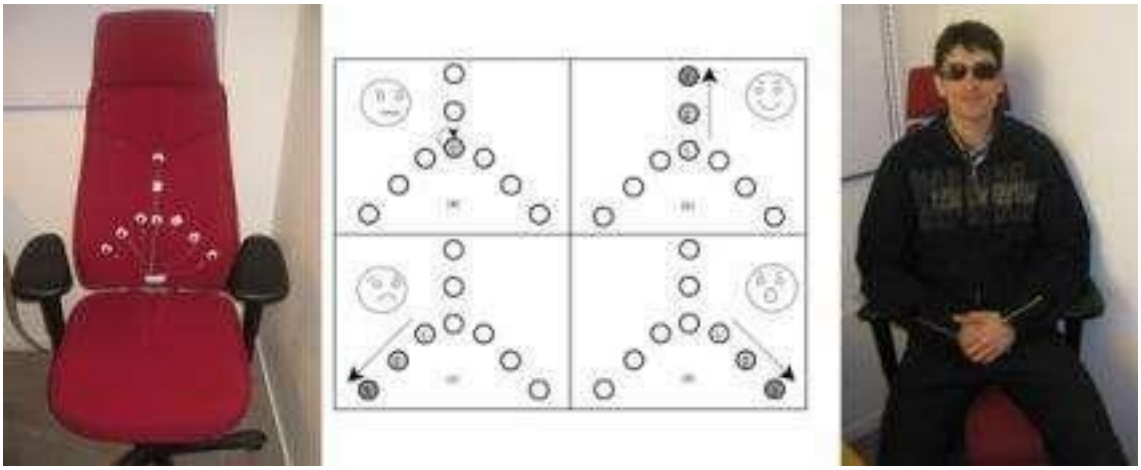


Figure 1-2： The chair with vibration motors [8]

1.2.3 Color Signals

Although color feedback SAD is not available yet, some research shows that colors could be matched to specific emotions, for example, Plutchik's emotion wheel [9] showed in *Figure 1-3*. In this wheel, hue is related to the emotion type, and saturation and value are related to the intensity of that emotion.

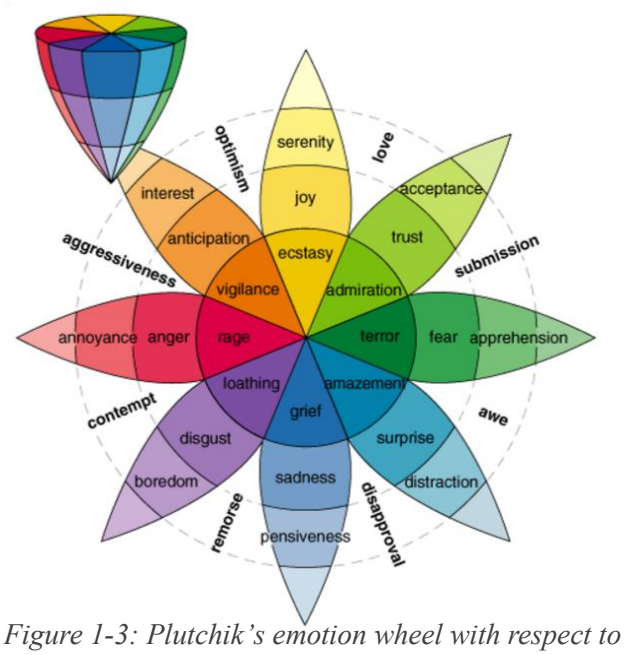


Figure 1-3: Plutchik's emotion wheel with respect to the relationship between color hue and saturation

1.2.4 Thermal Signals

The limitation of using the thermal signal as feedback is that the physical size of the device would be large. Also, although heater is easy to be integrated into a portable device, cooler is not, even small a semiconductor chilling plate would require a heat sink for heat radiating, let alone the compression engine.

1.3 Project Aims and Objectives

Due to the aforementioned limitations of current SAT for visually-impaired and ASD people, this project is aimed at designing, building and testing a system consisting of a camera, image analysis software and a wearable device with haptic and color feedback (*Figure 1-4*). This project is in conjunction with J. Ye from the University of St Andrews and R. Goucher from the University of Sterling. The camera or camera glasses [10] would take real-time videos of the user's conversation partner, and transmit the video to a device with image analysis and emotion detection software [10]. Then the wearable device will send reconfigurable feedback to the user to intuitively inform them of the real-time emotion of the conversation partner. Primary feedback is programmable haptic vibration patterns and color patterns, but can also be extended to sounds.

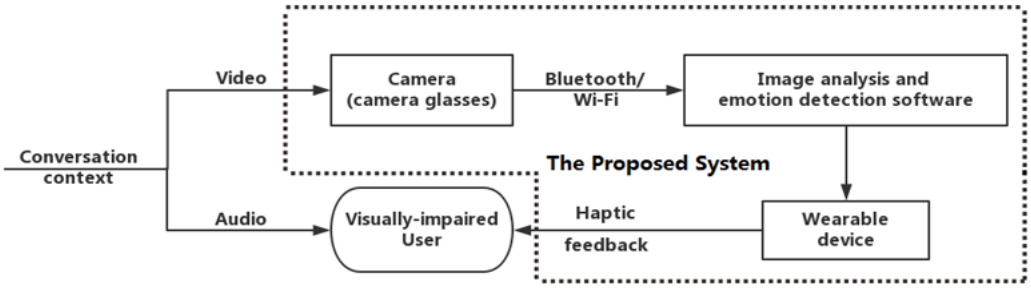


Figure 1-4: The proposed system

The sub-project of the above system started one year ago by Qisong Wang [11], which focused on designing a wristband as a wearable haptic feedback device that would play six basic emotions (Happiness, Fear, Surprise, Sadness, Disgust, and Anger) to the user. (*Figure 1-5*)

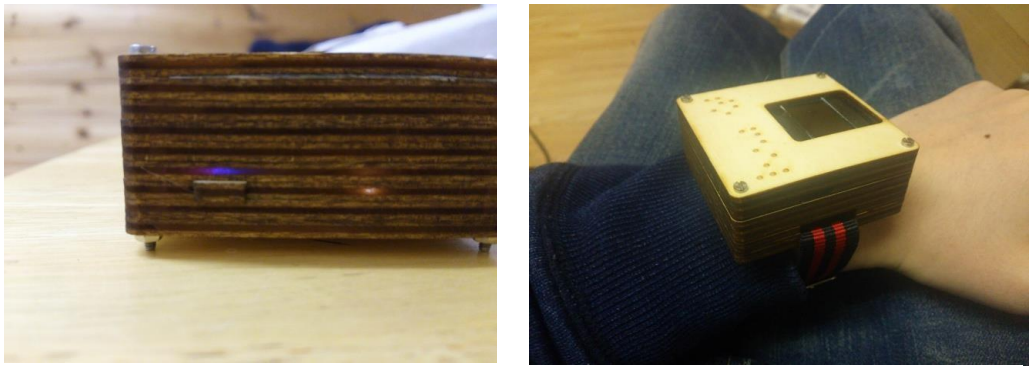


Figure 1-5: The haptic wristband designed by Qisong Wang [11]

1.4 Current Hardware Status

Qisong created an excellent haptic wristband based on Edison [12], with haptic feedback driven by DRV2605, and color pattern displayed by SSD1331. However, the connection between Edison and other components relies on the ribbon cables [11] (*Figure 1-6*) that insert into the socket; it is important to note that those electronic components are all integrated together on a wristband, along with an Eccentric Rotating Mass (ERM) motor. Vibration would gradually break these connections. After months' testing and demonstrations, the socket got loose while the inner connection status is invisible, as a result, the wristband was barely working and hardly repairable.

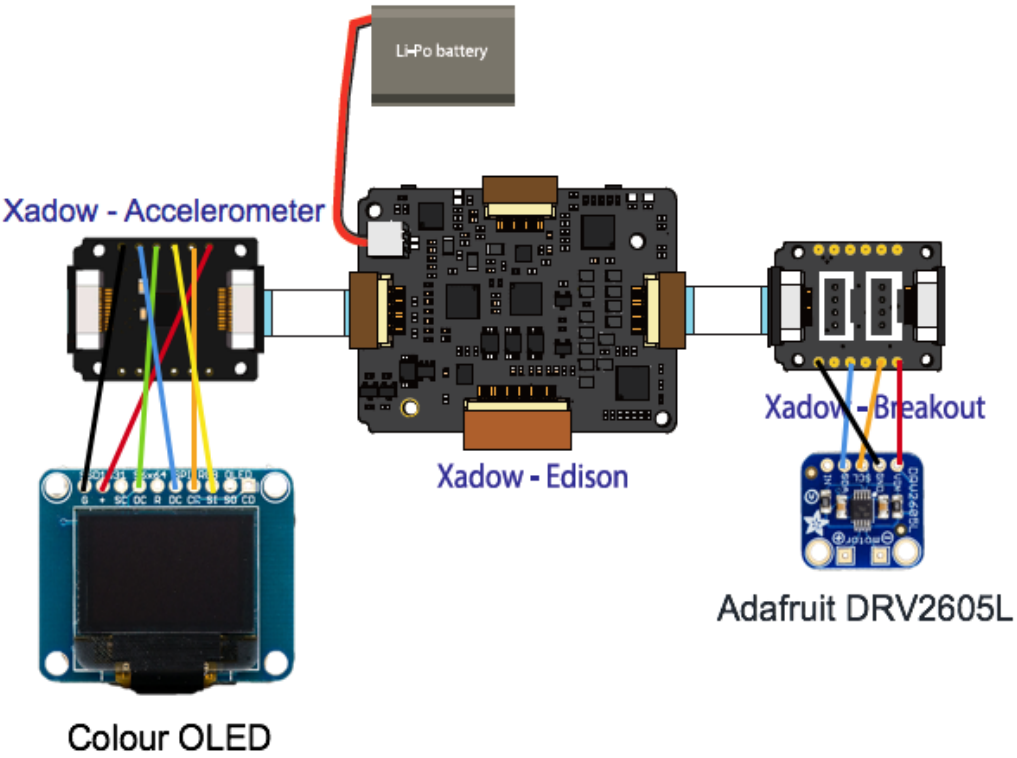


Figure 1-6: Wristband hardware developed by Qisong Wang, with ribbon cable connections [11]

Besides, as Microsoft has closed the free online-cloud, therefore the pattern data could no longer be saved online, and hence, local data saving and managing on the Android device is urgently needed as well.

1.5 Project Overview

This report would mainly present about my project issues: fixing and solving existing connection problems, enabling gradually changing between different colors, adding questionnaire section to the Android app [11] and attempt to get access to android's local memory.

Chapter 2 Hardware design

2.1 System Framework

According to my project overview in section 1.5, the overall system framework would be similar to Qisong's hardware design [11]. The microcontroller with internal memory drives the haptic vibration module and the color display module, and communicate with the Android device and the facial expression detection module via a wireless connection (Bluetooth or Wi-Fi). (*Figure 2-1*)

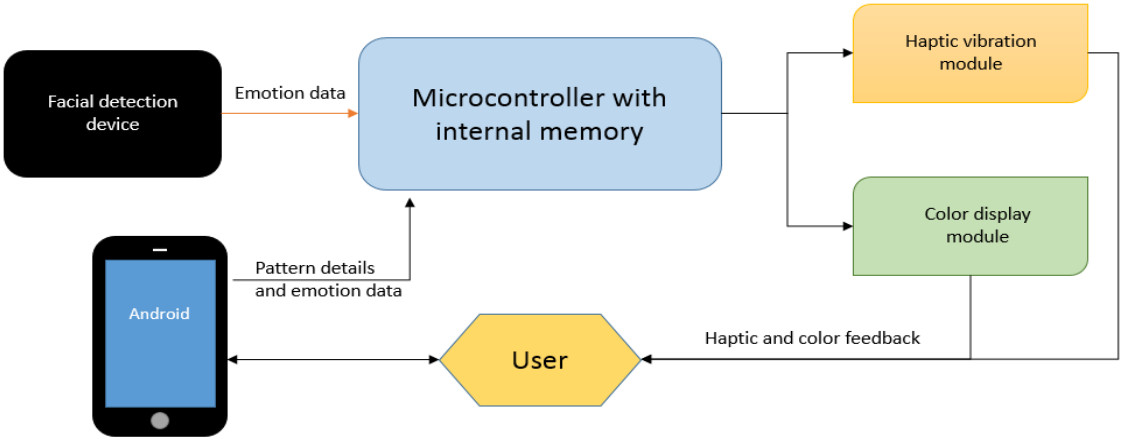


Figure 2-1: System Framework for the wristband

2.2 Microcontrollers

As aforementioned in section 1.2 and section 1.3, the haptic wristband ought to be wearable and portable. Therefore the physical size of the microcontroller ought to be small. As the microcontroller no longer needs to upload pattern data to the cloud, the Wi-Fi is not necessary. For future analysis and studies on the relationship between different vibration/color patterns and emotions, I decided to save all the pattern detail on the phone, along with the crowd-sourced data from questionnaires. As a result, powerful microcontrollers like Edison and Raspberry Pi are not the first choice (Many features and hardware resources were never used on Edison). In Qisong's project [11], I^2C and SPI communication is used to drive the motor driver and the organic light-emitting diode (OLED) screen respectively. I continued to using the DRV2605 (section 2.3) and SSD1331 (section 2.4), so I2C and SPI are required on the

microcontroller. If no on-board Bluetooth or Wi-Fi module is available on the microcontroller, the Universal Asynchronous Receiver/Transmitter (UART) port is needed to integrate a Bluetooth (section 2.5) or Wi-Fi module.

In conclusion, the desired microcontroller should have I^2C, SPI and UART port (or built-in Bluetooth/Wi-Fi module), with low power consumption and small physical size.

In *Table2-1*, features of Arduino pro mini [13] (*Figure 2-2*) and Redbear Duo [14] (*Figure 2-3*) is presented.

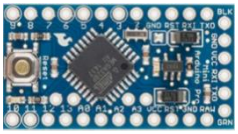


Figure 2-2: Arduino pro mini [13]

Figure 2-3: Redbear Duo [14]

Microcontroller	Arduino pro mini	Redbear Duo
Physical size(mm)	33.5*18.0*3.9	38.80*20.40*3.83
Power supply (*)	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)	3.5 - 8 V
Circuit Operating Voltage	3.3 V or 5 V	3.3 V
Digital I/O Pins	14	18
PWM Pins	6	13
UART, I^2C, SPI	1, 1, 1	2, 1 , 2
Clock Speed	8 MHz (3.3V versions) or 16 MHz (5V versions)	120MHz
On-board Bluetooth	No	Yes
EEPROM	512B	1MB

Table 2-1: Features of Arduino pro mini [13] and Redbear Duo [14]

*: Arduino pro mini can be powered by VCC pin on six pin header, or RAW pin with a voltage regulator which can accept voltage up to 12V DC. It can allow power supply changing between RAW pin and VCC pin when working. [13]

Redbear Duo can be powered by Micro USB port, the Vin pin or 3V3 pin. But only one power source can be powered at a time. Otherwise, the board will be damaged. [14]

Note that the library [15] for SSD1331 OLED screen from Adafruit only support the original

```
void Adafruit_SSD1331::begin(void) {
    // set pin directions
    pinMode(_rs, OUTPUT);

    if (_sclk) {
        pinMode(_sclk, OUTPUT);
        sclkportreg = portOutputRegister(digitalPinToPort(_sclk));
        sclkpin = digitalPinToBitMask(_sclk);

        pinMode(_sid, OUTPUT);
        sidportreg = portOutputRegister(digitalPinToPort(_sid));
        sidpin = digitalPinToBitMask(_sid);
    } else {
        // using the hardware SPI
        SPI.begin();
        SPI.setDataMode(SPI_MODE3);
    }
```

Figure 2-4: Codes in <Adafruit_SSD1331.cpp>

```
#define portOutputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_output_PGM + (P))) )
#define portInputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_input_PGM + (P))) )
```

Figure 2-5: Codes in <Arduino.h>

Arduino board. In begin() function in file <Adafruit_SSD1331.cpp>(Available in the CD in the last page of this thesis, shown in *Figure 2-4*), the function named 'portOutputRegister' is not available in Redbear's head file [14] <application.h>. Defining this function in <application.h> according to Arduino's head file <Arduino.h> [13] (*Figure 2-5*) did not solve the problem.

Although the Redbear has more I/O ports, they are not used in this project, the on-board Bluetooth and Wi-Fi can be replaced by HC-05 Bluetooth module (in section 2.5) via UART. Arduino pro mini also has advantages in the power supply, physical dimension. Let alone the problem mentioned above. As a result, Arduino pro mini is chosen for this project.

2.3 Vibration Motor & Motor Driver

2.3.1 Vibration Motors

Nowadays, two basic types of vibration motor are widely used. An Eccentric Rotating Mass (ERM) vibration motor [16] (shows in *Figure 2-6*) uses a small unbalanced mass on a DC motor; it vibrates when the mass is rotating with the rotor. Another one calls Linear Resonant Actuator (LRA) [17] (*Figure 2-7*), which contains a small magnetic mass attached to a spring and is similar to a loudspeaker, vibration can be created when the magnetic mass is driven by the coil.

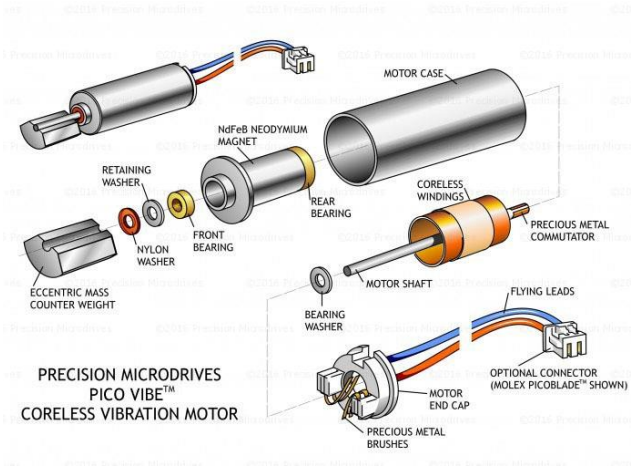


Figure 2-6: Eccentric Rotating Mass (ERM) vibration motor [16]

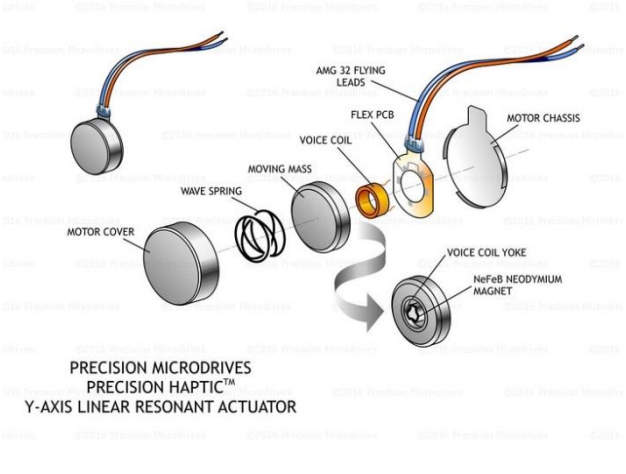


Figure 2-7: Linear Resonant Actuator (LRA) [17]

The vibration of LRA is far slighter than that of an ERM motor. Therefore LRAs are used in the VibroGlove [7]. ERM motors, however, are common in mobile phones. Its vibration intensity is strong enough to vibrate the whole portable device like phones, pagers, and wristband in this project.

2.3.2 Motor Driver DRV2605

As the ERM motor is used, it could be driven directly by a MOSFET (or 'H-Bridge' circuit), with Pulse-Width Modulation (PWM) from Arduino's digital ports.

However, the better solution is to use an integrated haptic motor driver. The DRV2605 device

(*Figure 2-8*) is a vibration motor driver which designed to give 'extremely-flexible haptic control' of ERM motors and LRA via I^2C [18].

The connections can be seen in the following *Table 2-2*:

Ports on DRV2605	Ports on Arduino	Description
SDA	A4	data port for I^2C
SCL	A5	clock port for I^2C
GND	GND	ground
VIN	VCC	V+

Table 2-2: Connections for DRV2605

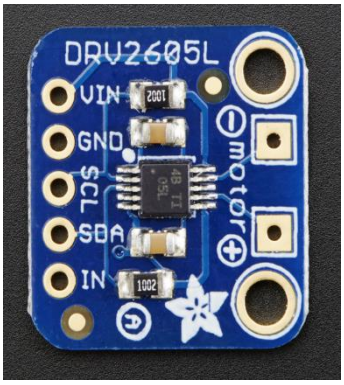


Figure 2-8: DRV2605 [18]

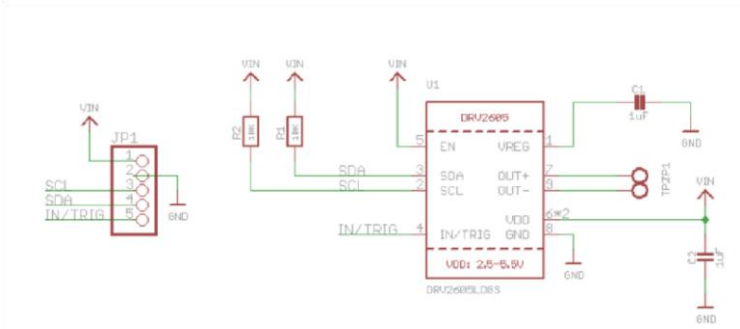


Figure 2-9: Schematics of DRV2605 [18]

Due to the large starting current the motor need when begin spinning from the still, so a large capacitor is added between VIN and GND of DRV2605.(*Figure 2-9*)

DRV2605 provides a library of 123 different vibration effects (available in Appendices A.1). The motor can vibrate based on the value of an 8-bit integer (0 for non-vibration while 255 for strongest vibration intensity). All these features would be presented in detail in section 3.3.

2.4 OLED Screen SSD1331

As mentioned in section 1.2.3, color patterns could be matched to emotions. Although RGB LEDs could be used to display color patterns, an OLED screen would be a better choice because

it can not only play color patterns but also display texts or even images on the screen.

SSD1331 OLED screen (*Figure 2-10*) from Adafruit is an ultra-small display with high-contrast 16-bit color (5-6-5 bits for red-green-blue) and wide viewing angle. The visible portion of the OLED screen is 0.96" diagonal and contains 96x64 RGB pixels [19]. The microSD card can be used for loading a bitmap and display it; this feature is not utilized in this project. The connections can be seen in the following *Table 2-3*:

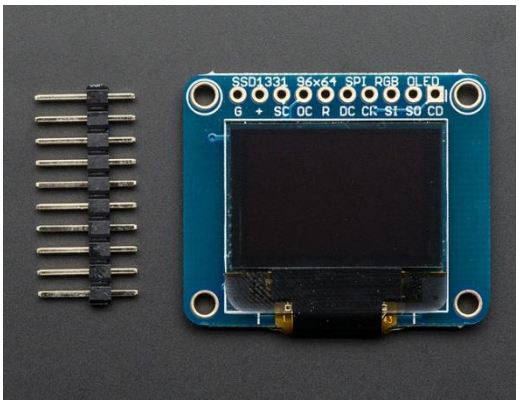


Figure 2-10: SSD1331 OLED screen [19]

Ports on OLED screen	Ports on Arduino	Description
G	GND	ground
+	VCC	V+
SC	---	SD Card Chip Select
OC	D10	OLED Chip Select
R	D9	Reset
DC	D8	Data/Command
CK	D13	Serial Clock
SI	D11	Master Out Slave In
SO	---	Master In Slave Out
CD	---	SD Detect

Table 2-3: Connections for OLED screen [19]

2.5 Bluetooth module HC-05

HC-05 Bluetooth module [20] (*Figure 2-11*) is used to enable Arduino to communicate with the Android phone. It is based on the Cambridge Silicon Radio BC417 2.4 GHz Bluetooth Radio chip [21] and can be set to either master or slave model.

The connections can be seen in the following *Table 2-4*:

Ports on HC-05	Ports on Arduino	Description
VCC	V+(*)	V+
GND	GND	Ground
TXD	D1	Transmit
RXD	D0	Receive
KEY	---	Mode select
LED	---	LED

Table 2-4: Connections for HC-05 [20]

Figure 2-11: HC-05

*: Note that V+ is the output port of the Lithium-ion battery changing module, instead of the VCC ports on Arduino. This Bluetooth module cannot connect to the VCC together with DRV2605. Because of the limitation of the output power of Arduino' I/O, the vibration motor's starting current would lead to the instability of HC-05 module: the Bluetooth connection is likely to be broken. (Adding start-up capacitor makes no difference.)

2.6 Battery & Charging Module

2.6.1 Battery

As a wearable device, the battery should be re-chargeable. To power all the existing electronic components, a Lithium Polymer Battery is selected in this project. The battery voltage vary

between 3 V (0%) to 4.2 V (100%), average voltage is 3.7 V.

In this project, LP-443440-IS-3 [22] Rechargeable Battery is used, with a capacity of 600mAh at 0.2C. The dimension of the battery is 41x34x4.2 (mm), and it fits the wristband well.

2.6.2 Charging module

However, overcharging or over-discharging would result in irreversible damage to the battery.

TELESKY 1A Li-Battery charging module [23] (*Figure 2-12*) is used in this project. This module can be powered via a MicroUSB port or 5 V DC input. Charging voltage is 4.2 V ± 1%, and current is 1000 mA. Discharging threshold voltage is 3 V, and maximum discharging current is 3 A. Battery's positive and negative pole should be connected to B+ and B- respectively.

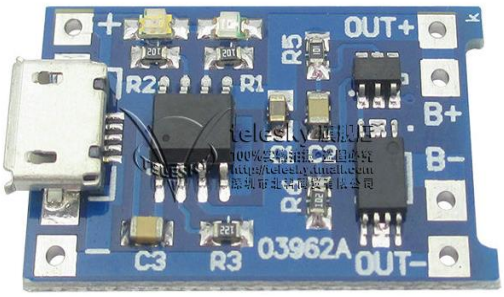


Figure 2-12: TELESKY 1A Li-Battery

2.7 (Reserved) Accelerometer ADXL345 & Buttons

Accelerometer ADXL345 (*Figure 2-13*) is used for detecting taps on the wristband to implement working status transition in Qisong's wristband [11]. ADXL345 is a low-power 3-axis accelerometer with high resolution and up to ± 16g measurement [24].

However, misdetection happens in actual operation. In addition, the large surface area is available on the wristband so that buttons (or switches) can be applied instead of accelerometers for the application above.

In this project, the wristband's operation status only depends on the received Bluetooth data. Therefore both accelerometer and button are unnecessary. However, the accelerometer can be

used to detect acceleration in X, Y and Z axis, to analyze the motion state of the wristband, and features like 'raise to wake' can be realized in the future. So the accelerometer and button (Figure 2-14) are reserved in the new wristband.

Figure 2-13: Accelerometer ADXL345 [24]

Figure 2-14: Button

The ADXL345 is driven by I^2C. So the connections are the same as DRV2605 motor driver. As shows in *Table 2-5*.

Ports on ADXL345	Ports on Arduino	Description
SDA	A4	data port for I^2C
SCL	A5	clock port for I^2C
GND	GND	ground
VIN	VCC	V+

Table 2-5: Connections for ADXL345 [24]

The button is connected to the VCC and D6 on Arduino.

2.8 Hardware System Integration

Based on the hardware chosen in this section, the circuit diagram (Figure 2-15) is designed.

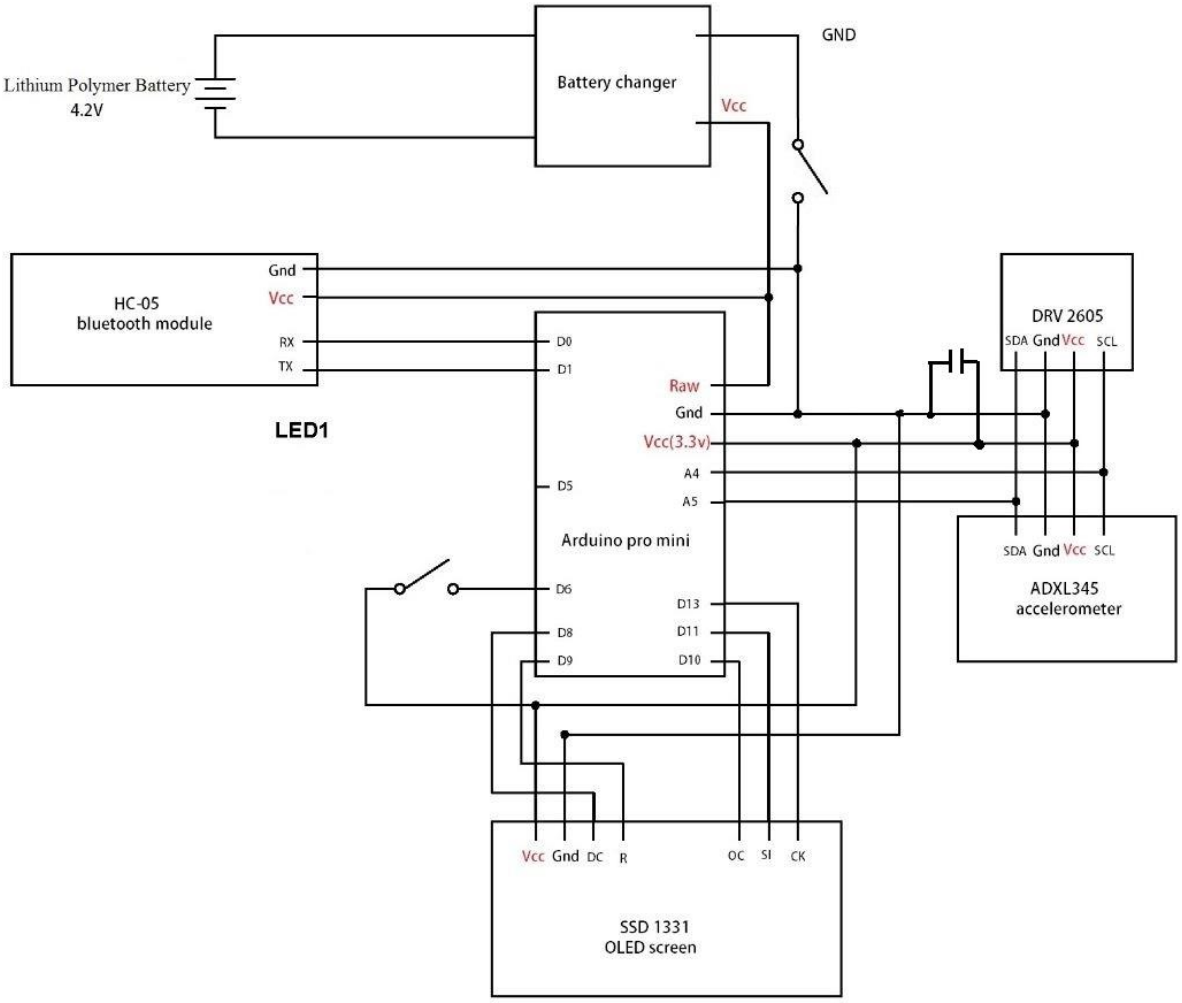


Figure 2-15: Circuit Diagram

The testing circuit on the breadboard is in *Figure 2-16*.

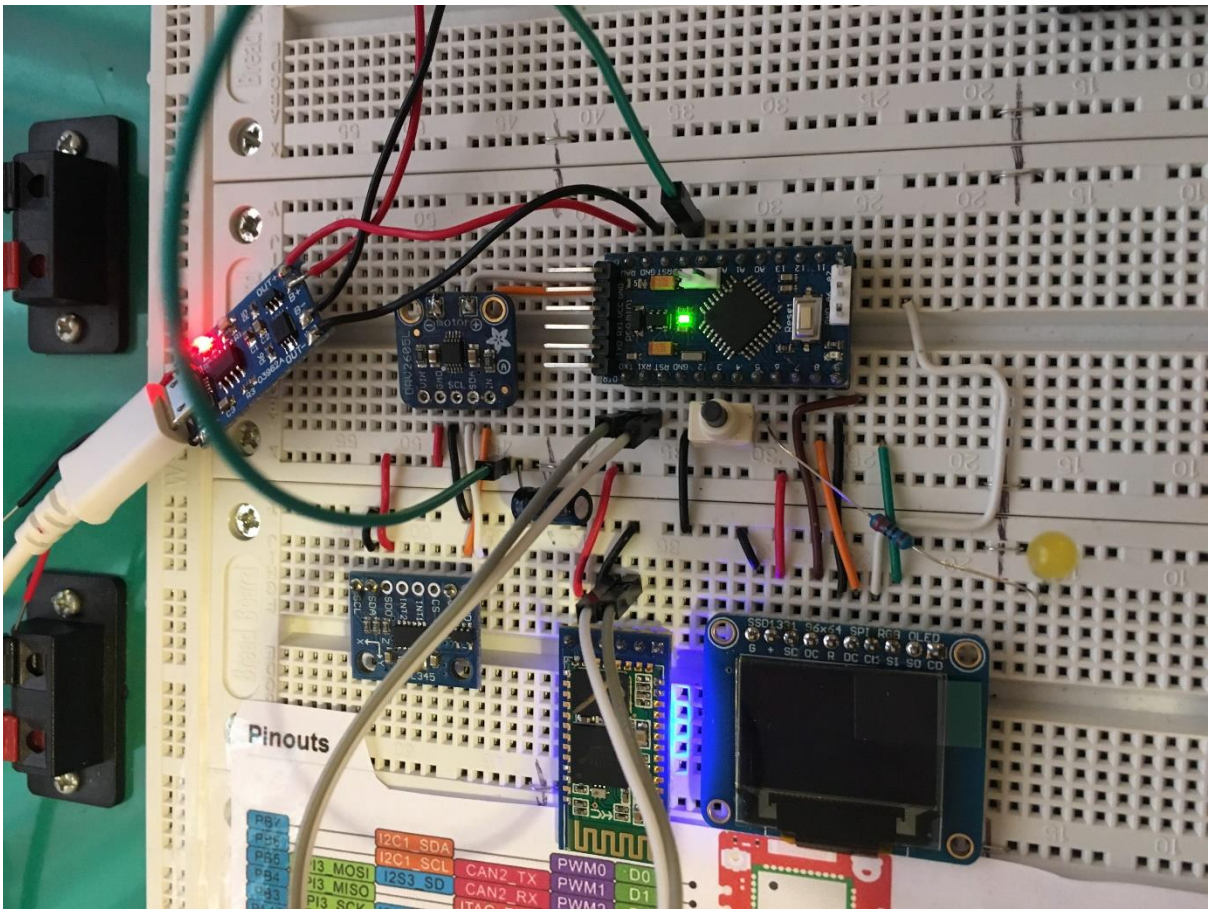


Figure 2-16: System on breadboard

Chapter 3 Hardware Programming & System Design

3.1 System Structure

The wristband system only operates when receiving Bluetooth data from Android phone or emotion detection device. Three kinds of data are recognizable for the wristband:

1. Play data: Start with 'P', followed by the pattern type ('L' represents library vibration patterns, 'R' means real-time vibration patterns, and 'C' represents color patterns), and its pattern sequence. When receiving play data, wristband would play the relevant pattern immediately. This kind of data is designed for questionnaires (section 4.7) and creating new patterns (section 4.4).
2. Uploading data: Start with 'U', followed by play data. Uploading data is send as a whole, 18 pieces of data (*) would be sent to the wristband together. Once receiving uploading data, the wristband would save it in local memory. This kind of data represents vibration and color patterns for different emotions.
3. Demo data: Start with 'D', followed by first two characters of 9 emotion types. The wristband would play the vibration and color pattern (saved in 2.uploading data) matches the emotion when receiving this kind of data.

*: The image analysis and emotion detection software [1] can detect up to ten different emotions: *happy, sad, angry, contempt, surprised, disgust, fear, nod, shake, and neutral.*

Considering that during an ordinary conversation, neutral takes most of the time, so the wristband would not play any pattern when the emotion is neutral. Therefore, nine vibration patterns and nine color patterns need to be saved in Arduino's local memory.

The system structure is shown in *Figure 3-1*.

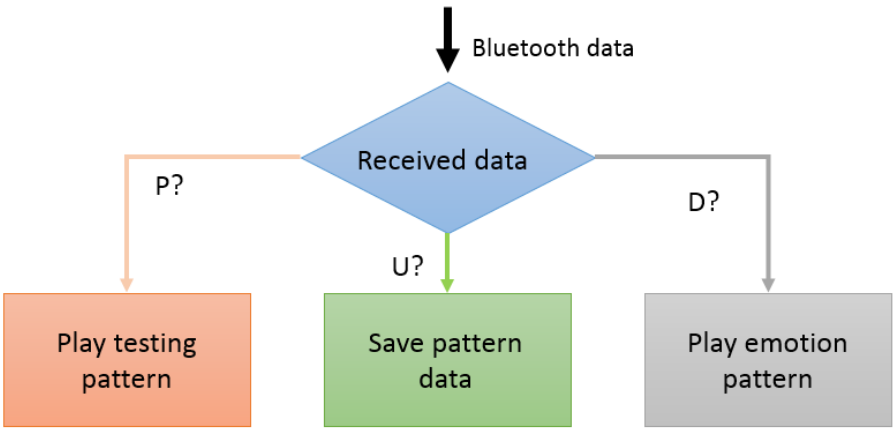


Figure 3-1: Wristband structure diagram

The Arduino program and relevant libraries are available in the CD. Arduino IDE 1.8.2 is recommend.

3.2 Bluetooth Communication

Arduino receive Bluetooth data from HC-05 module via UART, as a result, those data can be simply collected by using 'Serial' function. Following codes are designed for collecting and saving data in a string array 'mstring()':

```
 /*************receive data from phone*************/
if (Serial.available() > 0)
{mstring = "";}
  while(Serial.available() > 0)
  {
    isrec = true;
    command = ((byte)Serial.read());
    if(command == ':')
    {break; }
    else
    {
      mstring += command;
      }
    delay(2);
  }
```

3.3 DRV2605

This advanced vibration motor driver is controlled via I^2C, so the head file <Wire.h> [25] needs to be included in this project.

```
#include <Wire.h>
#include "Adafruit_DRV2605.h"
Adafruit_DRV2605 drv;
```

Before using various functions in this library, 'begin()' function should be run at 'setup()' section of the Arduino program.

```
 drv.begin();
 drv.selectLibrary(1);
```

Examples are available in DRV2605's library [26].

```
  // I2C trigger by sending 'go' command
  drv.setMode(DRV2605_MODE_INTTRIG); // default, internal trigger when sending GO command

  drv.selectLibrary(1);
  drv.setWaveform(0, 84);  // ramp up medium 1, see datasheet part 11.2
  drv.setWaveform(1, 1);  // strong click 100%, see datasheet part 11.2
  drv.setWaveform(2, 0);  // end of waveforms
```

'setMode()' function can set DRV2605 in different working modes. Many modes are defined in <Adafruit_DRV2605.h>

```
  #define DRV2605_MODE_INTTRIG  0x00
  #define DRV2605_MODE_EXTTRIGEDGE  0x01
  #define DRV2605_MODE_EXTTRIGLVL  0x02
  #define DRV2605_MODE_PWMANALOG  0x03
  #define DRV2605_MODE_AUDIOVIBE  0x04
  #define DRV2605_MODE_REALTIME  0x05
  #define DRV2605_MODE_DIAGNOS  0x06
  #define DRV2605_MODE_AUTOCAL  0x07
```

In this project, only library mode (DRV2605_MODE_INTTRIG) and real-time mode (DRV2605_MODE_REALTIME) is used. Play() functions for these two modes are defined:

```
void playLV(int effects[],int ptnnumb) {           //library vibration pattern play function
  for (unsigned int i = 0; i < ptnnumb; i++) {
    drv.setWaveform(i, effects[i] + 1);          // set the whole sequence
    if (i >= 10) break;
  }
  drv.setWaveform(ptnnumb, 0);                  //set effect numbers
  drv.go();                                    // play
}
```

The library pattern play function can play a sequence of effect (available in Appendices A.1) that saved in array 'effects[]', the ptnnumb is the number of effects in that array. It can play 10 effect in maximum. After set the whole sequence, it plays it by applying function 'drv.go()';

```
void playRV(int effects[],int ptnnumb, int time) {      //real time vibration play function
  for (unsigned int i = 0; i < ptnnumb; i++) {
    drv.setRealtimeValue(effects[i]/2);              //play
    delay(time);
  }
  drv.setRealtimeValue(0x00);                       //stop
}
```

Alternatively, vibration pattern is play based on the value of each number in array 'effects[]',

The larger each 8-bit integer is, the stronger vibration would be. Delay for 'time' milliseconds insure the real-time sequence to be recognizable

The received vibration pattern data in 'mstring()' (section 3.1) is in this format :

PL0,12,67,21,35,117

or

PR0,15,45,68,90,121,117,115,101,87,65,32,22,22,0

The first sequence starts with 'PL', represents '**P**lay' '**L**ibrary' vibration patterns, the following numbers are the library effects.

Similarly, 'PR' means '**P**lay' '**R**eal-time' vibration patterns, the rest numbers are values for the real-time sequence.

Following code for play library vibration pattern is an example:

```
if(mstring[0] == 'P'){       //for play patterns
```

```
  switch(mstring[1]){
    case 'L':        // receive the Library vibration pattern
    /*library vibration pattern */
      for(i = 0; i<=lnumb;i++){
        effect[i] = 0;
      }
      lnumb = 0;
      for(i = 2;i<mstring.length() ;i++){
        if(mstring[i] != ','){         //different effects in a pattern is divided by ','
          effect[lnumb] = effect[lnumb] * 10 +la;
          la = mstring[i] - '0';
        }else{
          effect[lnumb] = effect[lnumb] * 10 +la;
          la = 0;
          lnumb++;                        // number of the effect in library pattern
        }
      }
      la = 0;
      /********here we got the effect and it number = lnumb  ****************/
      drv.setMode(DRV2605_MODE_INTTRIG);
      playLV(effect,lnumb);
    break;
```

3.4 SSD1331 OLED Screen

In order to drive this OLED screen, <SPI.h> [27] needs to be included, along with the library [15] for SSD1331, including <Adafruit_SSD1331.h> and <Adafruit_GFX.h>.

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1331.h>
#include <SPI.h>
```

<Adafruit_SSD1331.h> is the basic head file of SSD1331 screen, which defines some low-level drawing actions and commonds:

```
// drawing primitives!
  void drawPixel(int16_t x, int16_t y, uint16_t color);
  void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color);
  //void fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t fillcolor);
  void pushColor(uint16_t c);
  // commands
  void begin(void);
  void goHome(void);
```

```
void goTo(int x, int y);
void reset(void);
```

In another head file:<Adafruit_GFX.h>, numerous high-level functions are defined, followed are some examples:

```
virtual void
  // It's good to implement those, even if using transaction API
  drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color),
  drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color),
  fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color),
  fillScreen(uint16_t color),
  // Optional and probably not necessary to change
  drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color),
  drawRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);
void
  drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color),
  setCursor(int16_t x, int16_t y),
  setTextColor(uint16_t c),
  setTextColor(uint16_t c, uint16_t bg),
  setTextSize(uint8_t s),
  setTextWrap(boolean w),
  cp437(boolean x=true),
  setFont(const GFXfont *f = NULL),
```

Following codes are written in Arduino's 'setup()' section to display the initialization interface:

```
display.fillScreen(0x0000);        //full screen black
display.setCursor(5, 5);
display.setTextColor(0xF800);      // text color is red
display.setTextSize(1);
display.println("Sensory Assistive Device");  //display text "Sensory Assistive Device"
delay(500);
display.setCursor(5, 30);
display.setTextColor(0xFFE0);        //text color is yellow
display.setTextSize(1);
display.println("with haptic feedback");     //display text "with haptic feedback"
delay(2000);
display.fillScreen(0x0000);
```

The initialization interface is shown in *Figure 3-1*.

Figure 3-1: Initialization Interface

Color pattern data (designed in android app, details available in section 4.5) is like:

PCrrr1,ggg1,bbb1,td1,ts1# rrr2,ggg2,bbb2,td2,ts2#...

'rrr1' represent the 8-bit red color value for the first color, similarly, 'ggg1' and 'bbb1' means color value for blue and green. 'td' is the duration for the first color (ms), and 'ts' is the time for the first color to change to the second color in this color pattern (this value is used to realize gradually changing between different colors). '#' represent an end for one color.

The color-pattern-display function:

```
/************ oled display function *******************/
void oledplay(int code[5][5],int ptnnumb){
   for(i = 0; i < ptnnumb ; i++){                          // transfer 24-bits color(R,G,B
all 8 bits) to 16 bits color (R 5 bits, G 6 bits, B 5 bits)
     color[i] = ((code[i][0]/8)<<11) + ((code[i][1]/8)<<5) + (code[i][2]/8);
   }
   for(j = 0 ; j < ptnnumb - 1 ;j ++){
     if(code[j][4] <= 200){                     // if not gradually change to next color
       switch_times[j] = 0;
     }else{        // gradually change: calculate the switching steps  and color difference
for each step
       switch_times[j] = code[j][4] / 150 ;
       diff_r[j] = ( color[j+1] & 0xF800  - color[j] & 0xF800   ) >>11;
       diff_g[j] = ( color[j+1] & 0x07E0  - color[j] & 0x07E0 ) >>5;
       diff_b[j] = ( color[j+1] & 0x001F  - color[j] & 0x001F  );
```

```
        diff[j]   = ((diff_r[j] / switch_times[j]) << 11) + ((diff_g[j] / switch_times[j] )
<< 5) + (diff_b[j] / switch_times[j]);
      }
    }
    for(i = 0; i<ptnnumb; i++){
        display.fillRect(5,5,40,50,color[i]);            //display the color according
        delay(code[i][3]);                              //duration
        if(i == ptnnumb - 1 )                   // for the last color, no switching is needed
          break;
        for( j = 0; j <= switch_times[i] ;j++){          // swithcing time
          display.fillRect(5,5,40,50,color[i] + diff[i] * j);
        }
    }
    display.fillRect(5,5,40,50,0x0000);        //to minimize the refresh time , only display
in a small rectangle
}
```

First of all, the 24-bit color (red, green, blue takes 8 bits respectively) needs to be transferred to 16-bit color (5 bits for red and blue, 6 bits for green).

Because of the limitation of the OLED screen's refresh frequency, actual smooth, gradual change is very different to realize. A possible solution is to calculate the difference between two colors' value. Then divide it into several parts ('diff[] in above codes'), then add 'diff[]' to the previous color and display it for a short time, after running this loop for several times, the displayed color changes gradually to next one. In addition, the screen takes the time to refresh, and the more pixels it scans, the longer it would take. As a result, to minimize the refresh time, only a part of the screen is used to display color pattern.

3.5 Internal Memory on Arduino - EEPROM

In order to save pattern data on the Arduino, Electrically Erasable Programmable Read-only Memory (RRPROM) is used, which could keep saved data when the board is turned off. Head file <EEPROM.h> [28] is used in this section.

As there are nine different emotions to be detected [1], nine vibration patterns and nine color patterns need to be saved in EEPROM.

For Arduino pro mini, 512B ROM is available [13], and each byte can only hold a value from 0 to 255. To make the most of the available resource, each pattern takes 26 bytes of ROM, the first byte records its pattern type and the sequence number, and rest bytes record the sequence. The first byte of library vibration pattern equals 100 + number of this pattern's effects, for the real-time vibration pattern, it is 200 + number of sequences. Similarly, the color patterns' first byte equals 0 + number of colors.

Example codes for saving library vibration patterns are followed:

```
case 'L':
          /*library vibration pattern
            *
             */
            for(i = 0; i<=lnumb;i++){
              effect[i] = 0;
            }
            lnumb = 0;
            for(i = 3;i<mstring.length() ;i++){
              if(mstring[i] != ','){
                effect[lnumb] = effect[lnumb] * 10 +la;
                la = mstring[i] - '0';
              }else{
                effect[lnumb] = effect[lnumb] * 10 +la;
                la = 0;
                lnumb++;
              }
            }
            /********here we got the effect and it number = lnumb  ****************/
            codetosave[uploadnumb - 1][0] = 100 + lnumb;      //   if the first bit of the
code is between 101-199  then it is a library pattern
            for(k=1; k<=lnumb;k++){
              codetosave[uploadnumb - 1][k] = effect[k - 1];
            }
          break;

          if(uploadnumb == 18){
       uploadnumb = 0;
       isuploaded = true;
   /***
     Write the uploading pattern details to EEPROM
     These data would still be there even powered off
```

```
***/
    for(j = 0; j < 18;j++){
      for(i = 0; i < 26; i++){
        EEPROM.write(j * 26 + i, codetosave[j][i]);
        }
      }
}
```

Similar to write data into EEPROM, when the wristband is powered on, it reads data from EEPROM:

```
for(j = 0; j < 18;j++){
  for(i = 0; i < 26; i++){
    codetosave[j][i] = EEPROM.read(j * 26 + i);
  }
}
```

3.6 (Reserved) ADXL345 & Buttons

3.6.1 ADXL345

Although accelerometer is not used in this project, it can be used to realize tap detection or raise to wake functions.

Similar to DRV2605, <Wire.h> is required for I^2C. In 'setup()' loop, following codes can be used to initialize the different functions of this module:

```
adxl.powerOn();
//set activity/ inactivity thresholds (0-255)
adxl.setActivityThreshold(75); //62.5mg per increment
adxl.setInactivityThreshold(75); //62.5mg per increment
adxl.setTimeInactivity(10); // how many seconds of no activity is inactive?

//look of activity movement on this axes - 1 == on; 0 == off
adxl.setActivityX(1);
adxl.setActivityY(1);
adxl.setActivityZ(1);

//look of inactivity movement on this axes - 1 == on; 0 == off
adxl.setInactivityX(1);
```

```
adxl.setInactivityY(1);
adxl.setInactivityZ(1);

//look of tap movement on this axes - 1 == on; 0 == off
adxl.setTapDetectionOnX(1);
adxl.setTapDetectionOnY(1);
adxl.setTapDetectionOnZ(1);
```

Note that the vibration from the ERM motor would add tremendous noise to the acceleration data, but the vibration is perpendicular to the motor's rotating axle, so most of the noise can be removed by disabling detection on two axes. For example, if the motor shaft is parallel with the X-axis of the accelerometer, the following codes can be applied:

```
adxl.setTapDetectionOnX(1);
adxl.setTapDetectionOnY(0);
adxl.setTapDetectionOnZ(0);
```

However, in this case, the accelerometer could only detect taps along the x-axis.

Alternatively, a better solution is to disable all the detect action when playing vibration pattern, and enable it after vibration. For example:

```
// Disable tapping detection during vibraton
adxl.setInterrupt(ADXL345_INT_SINGLE_TAP_BIT, false);
adxl.setInterrupt(ADXL345_INT_DOUBLE_TAP_BIT, false);

drv.setWaveform(0, effect[mode]);  // play effect
drv.setWaveform(1, 0);       // end waveform
// play the effect!
delay(100);
drv.go();
delay(100);

// Enbale tapping detection agian
adxl.setInterrupt(ADXL345_INT_SINGLE_TAP_BIT, true);
adxl.setInterrupt(ADXL345_INT_DOUBLE_TAP_BIT, true);
```

Following code is a good example to get acceleration value of three axis:

```
    double xyz[3];
    double ax,ay,az;
    adxl.getAcceleration(xyz);
    ax = xyz[0];
    ay = xyz[1];
    az = xyz[2];
```

These value can be used to realize ‘raise to wake’ in future work

3.6.2 Buttons

Code for button detection is followed:

```
int i=digitalRead(6);                  //read value on D6 port
    if (i == HIGH){
                          // button pressed
    }else{
                          // button released
    }
```

The reserved button can be used in future works.

Chapter 4 Software Design

As described in project overview in section 1.5, more functionalities need to be added to the existing Android app. First of all, the Android app can be used to crowd-source questionnaires, and the developer ought to be able to go through these data freely. Considering that the cloud service from Microsoft has been closed, the collected data has to be saved on Android device. As well as the detailed information for the designed vibration and color pattern. Qisong's app only support six emotions, and it is represented with pictures with facial expression, this might lead to misunderstanding for ASD users. Take aforementioned points into consideration, re-building the app would be easier than modifying it.

The following details ought to be included for each pattern: name, emotion type (one of the nine emotions), pattern type (library vibration pattern, real-time vibration pattern or color pattern), and its pattern sequence. Therefore, saving those data in Excel file or SQLite database are feasible solutions. Considering that these data would be analyzed on Matlab (e.g. using the neutral net to analyze the corresponding relation between emotions and vibration patterns), Excel format seems to be the best choice.

As this Android project is huge and complicated, it is impossible and unnecessary to go into every single detail of the Android Studio design. Also, a certain level of knowledge and experience in Java and Android development is essential to understand this chapter.

The source code of this project is available on the CD on the last page, and on Github as well: https://github.com/tony950309/sentire

Some information for this Android project:

Required Android system: Android 4.0.4 or above

Development Environment: Android Studio 2.3.1

Hardware requirement: Android phone or tablet with Bluetooth adapter

Third-party libraries:

```
compile files('libs/jxl.jar')
compile 'com.android.support:cardview-v7:25.3.1'
compile 'com.akexorcist:bluetoothspp:1.0.0'
compile 'com.skyfishjy.ripplebackground:library:1.0.1'
compile 'com.github.blackfizz:eazegraph:1.2.2@aar'
compile 'com.nineoldandroids:library:2.4.0'
compile 'com.github.QuadFlask:colorpicker:0.0.10'
compile 'com.github.szugyi:Android-CircleMenu:1.1.1'
compile 'com.github.gabrielemariotti.cards:cardslib-core:2.1.0'
compile 'com.github.gabrielemariotti.cards:cardslib-cards:2.1.0'
testCompile 'junit:junit:4.12'
```

4.1 General Design

4.1.1 Bluetooth Design

Bluetooth communication [29] requires uses permission of peripheral Bluetooth equipment on Android devices, so the following permission declaration in AndroidManifest.xml is necessary:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Following two packages need to be imported:

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
```

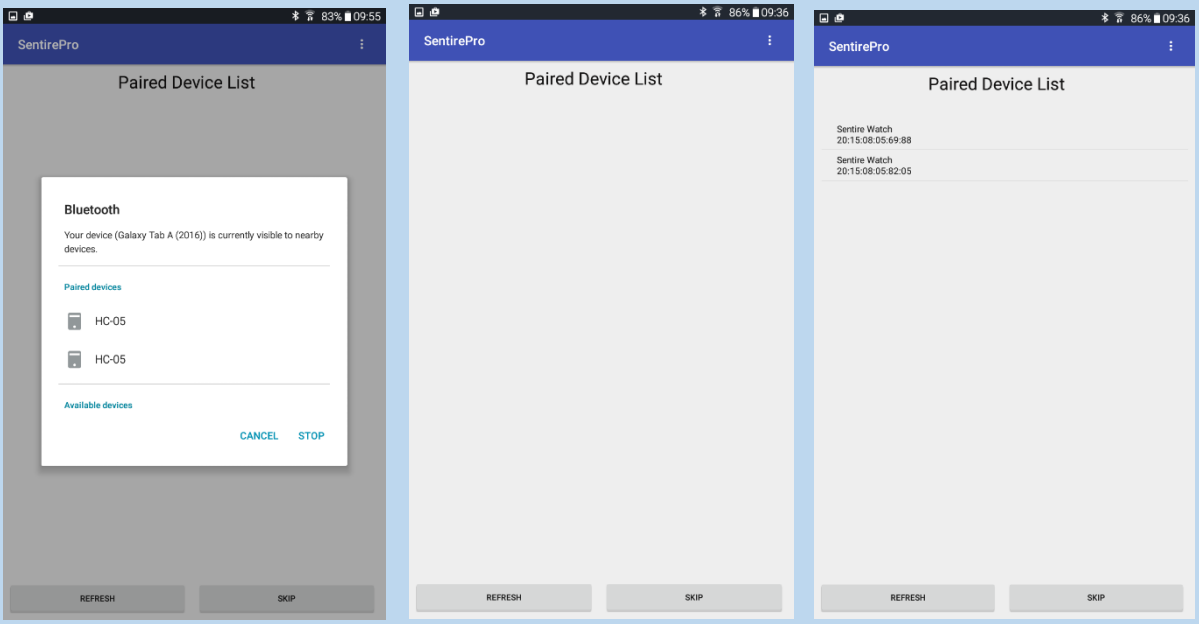


Figure 4-1: (a) (b) (c)

For the new device that has not been paired, the app would ask the user to select an available device to pair with. (Figure 4-1(a)) The user would be able to view the paired devices list by clicking the REFRESH button, then click the device to pair. Or click SKIP button to go to Homepage directly.

Following method can detect with the Bluetooth Adapter on the Android device is available:

```
if (myBluetooth == null) {
    //Show a mensag. that the device has no bluetooth adapter
    Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
Toast.LENGTH_LONG).show();
}
else if (!myBluetooth.isEnabled()) {
    //Ask to the user turn the bluetooth on
    Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(turnBTon, 1);
}
```

To get the Bluetooth address of the paired devices, these codes are used:

```
pairedDevices = myBluetooth.getBondedDevices();
ArrayList list = new ArrayList();

if (pairedDevices.size() > 0) {
    for (BluetoothDevice bt : pairedDevices) {
        list.add("Sentire Watch  " /*+ bt.getName() */ + "\n" + bt.getAddress());
//Get the device's name and the address
    }
```

Above code would add the paired device address and its name ‘Sentire Watch’ to the listview as well.

Once the user clicks on the paired device, the Android app would run following method to try to get connected with the selected device:

```
try {
    if (btSocket == null || !isBtConnected) {
//get the mobile bluetooth device
        myBluetooth = BluetoothAdapter.getDefaultAdapter();
//connects to the device's address and checks if it's available
        BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(SUPER_address);
```

```
//create a RFCOMM (SPP) connection
        btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
        BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
        btSocket.connect();//start connection
    }
} catch (IOException e) {
    ConnectSuccess = false;//if the try failed, you can check the exception here
}
```

4.1.2 Write, Read and Edit Excel File

jxl.jar [30] library is used to create, write, read or edit an Excel file. Uses permission of external storage should be declared in AndroidManifest.xml as well:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Following packages are used for this section:

```
import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.write.WriteException;
```

As the app is not able to read a nonexistent Excel file, so the Excel file should be created first, as well as the sheet 1:

```
File file=new File(Environment.getExternalStorageDirectory(),"Cptn.xls");
//  create the sheet ,name it
WritableWorkbook book = Workbook.createWorkbook(file);
WritableSheet sheet = book.createSheet("My library vibration patterns", 0);
```

To add a string or an integer to a cell, Label and jxl.write.Number can be used:

```
Label mystring = new Label(0, 0, "Total pattern number :");
sheet.addCell(mystring);
jxl.write.Number mynumb = new jxl.write.Number(3, 0, ptn_numbC);
sheet.addCell(mynumb);
```

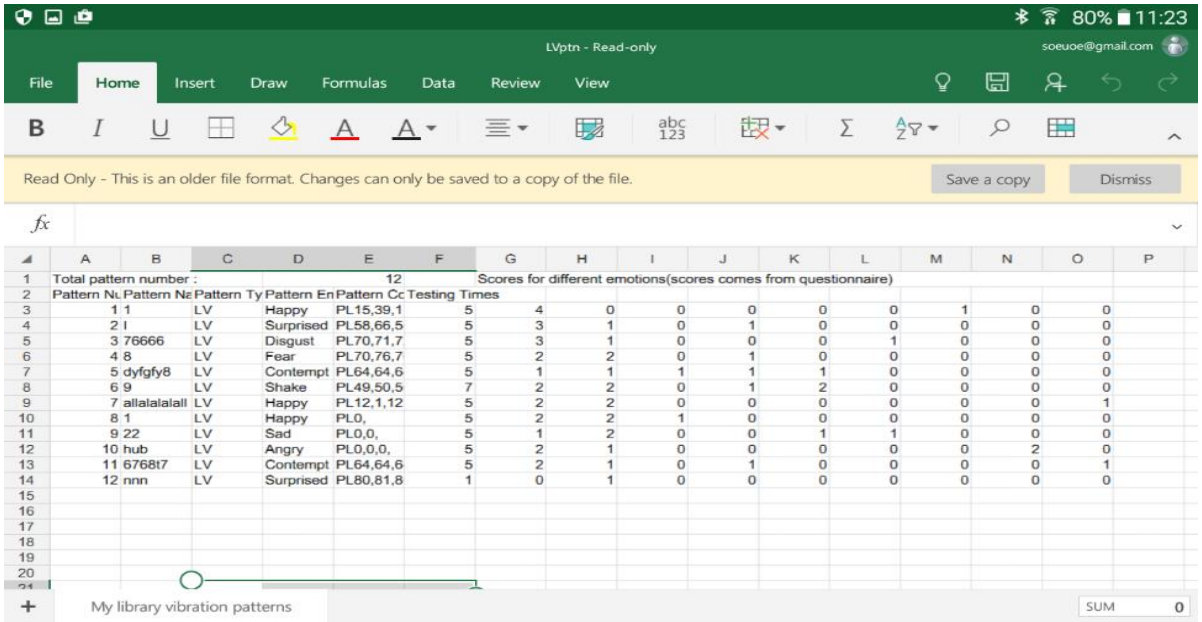
The two number (3 and 0) in jxl.write.Number(3, 0, ptn_numbC) means column 4 and row 1

respectively. (It starts from 0)

After adding all the data into cells, the app would write all cells into the Excel file with the following method:

```
book.write();
book.close();
```

An example of creating an Excel file to save library vibration pattern detail is attached in Appendices A.3. The generated Excel file is shown in *Figure 4-2*.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Total pattern number :				12		Scores for different emotions(scores comes from questionnaire)									
2	Pattern Nu	Pattern Na	Pattern Ty	Pattern En	Pattern Cc	Testing Times										
3	1	1	LV	Happy	PL15,39,1	5	4	0	0	0	0	0	1	0	0	
4	2	I	LV	Surprised	PL58,66,5	5	3	1	0	1	0	0	0	0	0	
5	3	76666	LV	Disgust	PL70,71,7	5	3	1	0	0	0	1	0	0	0	
6	4	8	LV	Fear	PL70,76,7	5	2	2	0	1	0	0	0	0	0	
7	5	dyfgfy8	LV	Contempt	PL64,64,6	5	1	1	1	1	1	0	0	0	0	
8	6	9	LV	Shake	PL49,50,5	7	2	2	0	1	2	0	0	0	0	
9	7	allalalalall	LV	Happy	PL12,1,12	5	2	2	0	0	0	0	0	0	1	
10	8	1	LV	Happy	PL0,	5	2	2	1	0	0	0	0	0	0	
11	9	22	LV	Sad	PL0,0,	5	1	2	0	0	1	1	0	0	0	
12	10	hub	LV	Angry	PL0,0,0,	5	2	1	0	0	0	0	0	2	0	
13	11	6768t7	LV	Contempt	PL64,64,6	5	2	1	0	1	0	0	0	0	1	
14	12	nnn	LV	Surprised	PL80,81,8	1	0	1	0	0	0	0	0	0	0	

Figure 4-2: LVptn.xls File

Reading data from an existing Excel file is similar:

```
File file=new File(Environment.getExternalStorageDirectory(),"Cptn.xls");
Workbook workbook = Workbook.getWorkbook(file);
//
Sheet sheet = workbook.getSheet(0);
//  get total number of existing ptn numbs
Cell cell = sheet.getCell(3, 0);// read from row 1 column 4
existingPtnNumbC = Integer.valueOf(cell.getContents()).intValue();
```

One shortcoming of the jxl.jar library is that it could not write data in one cell, which means to edit some of the cells, all the data should be read and saved in RAM. Then write them back to the same file after editing.

4.2 Homepage

In Homepage, users can go to any other activity by clicking on the corresponding button. If the Bluetooth connection is broken for some reasons; it can be reconnected if the user press 'RECONNECT' button. There is also a help menu provide for users. In order to start another activity when click the corresponding button, following method can be used:

```
i = new Intent(Homepage.this, seeq.class);
startActivity(i);
```

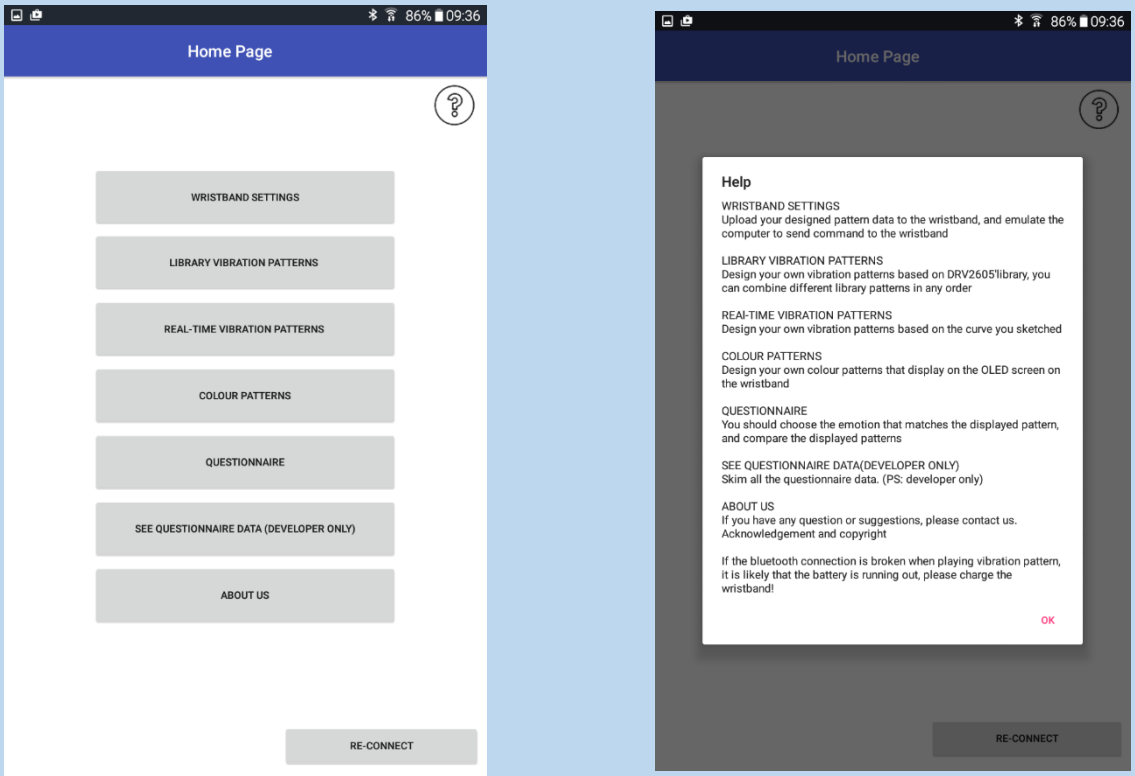


Figure 4-3: *(a)* *(b)*

4.3 Wristband Settings

In this activity, the user can initialize the app data. Which means all the pattern that has been designed would all be delete, including the file that saved the pattern for uploading to Arduino (These pattern sequence starts with 'U', as mentioned in section 3.1). The user would also be

able to clear the scores (for example, in the questionnaire, if one pattern is chosen to be Happy emotion, its happy score would plus 1) for all the pattern. The more vital functionality is to upload the emotion pattern to Arduino (starts with 'U', as mentioned in section 3.1). All these operations would require a password '0620', which can avoid misoperation.

The user can play all the pattern by press play button in the listview. They can enter demo activity (*Figure 4-4 (c)*) by press 'DEMO' as well

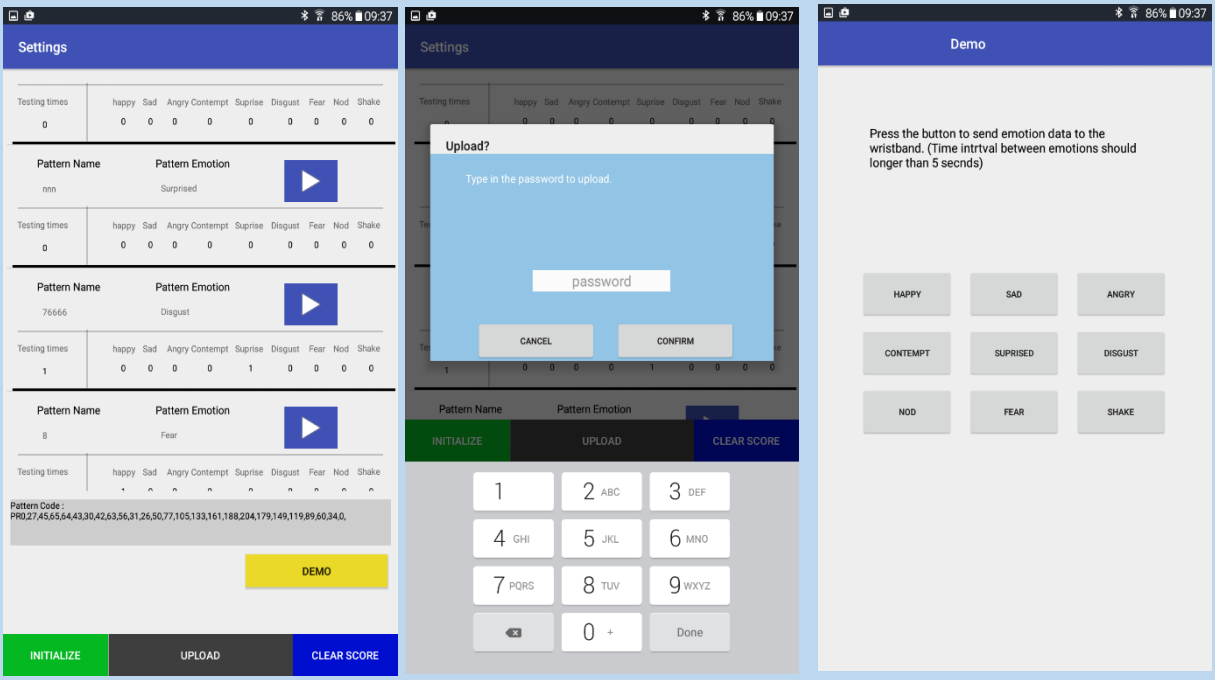


Figure 4-4: (a) *(b)* *(c)*

4.4 Library Vibration Patterns

In the first activity (*Figure 4-5 (a)*), all the designed library vibration patterns are available; the user can play it, change the name and emotion for selected pattern. They can add new effects to the existing pattern by clicking 'EDIT CODE'. 'Clear' button would delete all the created library vibration patterns. When pressed 'Create New', it would go to the second activity (*Figure 4-5 (c)*), in which new library pattern can be created and saved.

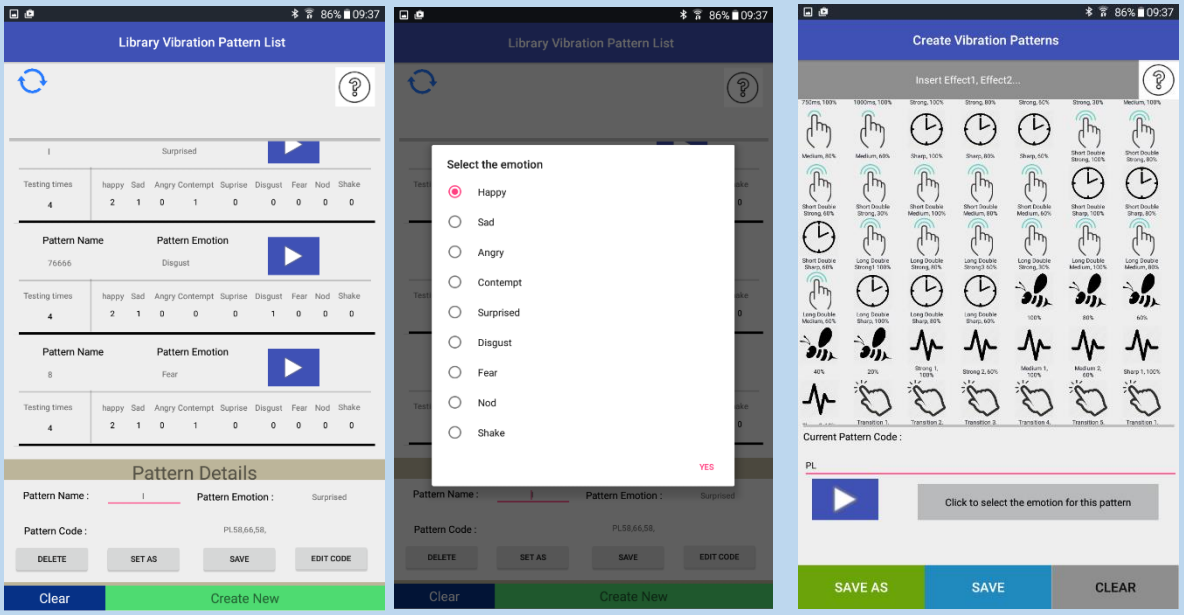


Figure 4-5: (a) *(b)* *(c)*

Note that when creating a new pattern, 'SAVE AS' and 'SAVE' are totally the same, however, when editing existing pattern, 'SAVE' button would overwrite the previous pattern with new effect sequence, while 'SAVE AS' button would save it as a new pattern.

Once created a new pattern, the user can view it and play it in previous activity (*Figure 4-5 (a)*) after pressing the refresh button on the left-top corner first.

In the first activity, a customized listview is required, to display all the pattern details and enable users to play it with an imageButton. (*Figure 4-6*)

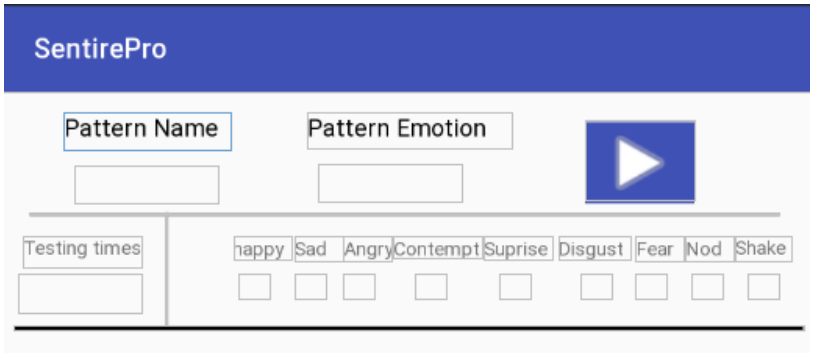


Figure 4-6: Customized Listview

This listview is defined in a java class file:

```
public savedptn_class(String ptnname,String ptntype,String ptnemot,String ptncode,
                      int ptntimes,
                      int scorehap,int scoresad,int scoreang,int scorecon,
                      int scoresup,int scoredis,int scorefea,int scorenod,
```

```
                    int scoresha,int imgid){
    this.ptnname = ptnname;
    this.ptntype = ptntype;
    this.ptnemot = ptnemot;
    this.ptncode = ptncode;
    this.ptntimes = ptntimes;
    this.scorehap = scorehap;
    this.scoresad = scoresad;
    this.scoreang = scoreang;
    this.scorecon = scorecon;
    this.scoresup = scoresup;
    this.scoredis = scoredis;
    this.scorefea = scorefea;
    this.scorenod = scorenod;
    this.scoresha = scoresha;
    this.imgid = imgid;
}
```

And an adapter is used to play these data in the listview:

```
public class savedptn_adapter extends BaseAdapter {
    private List<savedptn_class> objects;
    public savedptn_adapter(List<savedptn_class> objects)
    {this.objects=objects; }
    @Override
    public int getCount() {
        return objects.size();
    }

    @Override
    public Object getItem(int i) {
        return objects.get(i);
    }
    @Override
    public long getItemId(int i) {
        return i;
    }
    public View getView(int position, View converView, final ViewGroup parent){
        final savedptn_class ptnclass = objects.get(position);
```

A ViewHolder is applied to optimize the listview by saving the list item information in a buffer. If the information does not change, the app would not re-load the information again:

```
if(converView==null)
        {
            holder=new ViewHolder();
            View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.savedptnlist, null);
            holder.tvname = (TextView) view.findViewById(R.id.tvptnname);
            converView=view;
            converView.setTag(holder);
        }else
        {
            holder= (ViewHolder) converView.getTag();
        }
        holder.tvname.setText(ptnclass.getPtnname());
        holder.ivbtPlay.setImageResource(ptnclass.getImgid());
        holder.ivbtPlay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                BTsend(ptnclass.getPtncode());
                Toast.makeText(parent.getContext(),"code :
"+ptnclass.getPtncode(),Toast.LENGTH_SHORT).show();
//              BTsend(ptnclass.getPtncode());
            }
        });
```

For the second activity (*Figure 4-5 (c)*), when clicking on the gridView item, the relevant effect number would be added to the pattern sequence, the following girdView's OnClickListerner shows how it works:

```
gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v,
                            int position, long id) {
        if (effectSequence.isEmpty()) {
            effectSequence += String.valueOf(position);
        } else {
            effectSequence += (",," + String.valueOf(position));
        }
        current_ptncode += (String.valueOf(position) + ",");
        if(current_ptncode.length() > 30){  //if the sequence is too long
            isvalid = false;
        }else{
            isvalid = true;
        }
        if(isvalid){
```

```
            tveffect.setText(effectSequence);
            etLVCptncode.setText(current_ptncode);
        }else{
            msg("Invalid drawing! The sequence might be too complex or too long to
play!" + "\n"+
                    "Current code length is " +
Integer.toString(current_ptncode.length()) +
                    ". Try to make it shorter than 30." + "\n" +
                    "Otherwise the sequence might not be played perfectly.");
        }
    }
});
```

Here created the pattern sequence that mentioned in section 3.1 and section 3.3 (e.g. 'PL0,12,67,121,55,66')

4.5 Real-Time Vibration Patterns

The first activity for the real-time pattern is very similar to that in the previous section (*Figure 4-7 (a)*). The second activity for creating real-time pattern used the third-party library: 'eazegraph-1.2.2' [31].

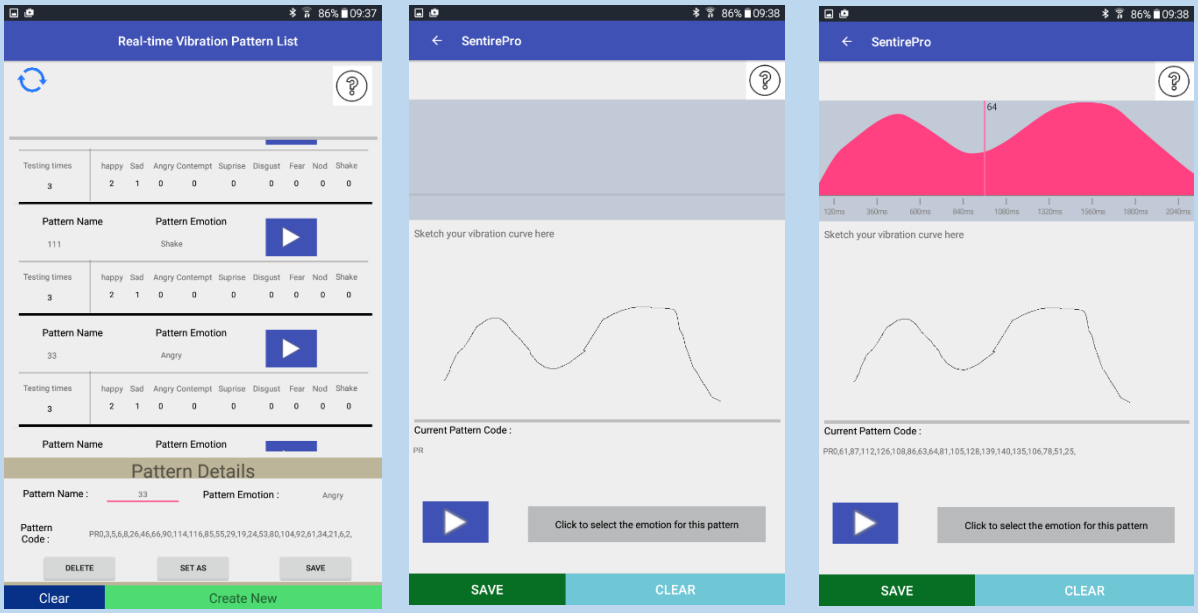


Figure 4-7: (a) *(b)* *(c)*

These packages are imported for creating real-time vibration pattern:

```
import org.eazegraph.lib.charts.ValueLineChart;
import org.eazegraph.lib.models.ValueLinePoint;
import org.eazegraph.lib.models.ValueLineSeries;
```

Class 'MyPlantView' allows users to sketch their own curves, to call this method, following codes are used:

```
myPaintView = (MyPaintView) findViewById(R.id.paintView);
mCubicValueLineChart = (ValueLineChart) findViewById(R.id.cubiclinechart);
mCubicValueLineChart.setUseDynamicScaling(false);
mCubicValueLineChart.setMaxZoomY(255);
mCubicValueLineChart.setIndicatorTextColor(getResources().getColor(R.color.colorPri
maryDark));
```

Then get the value for the curve:

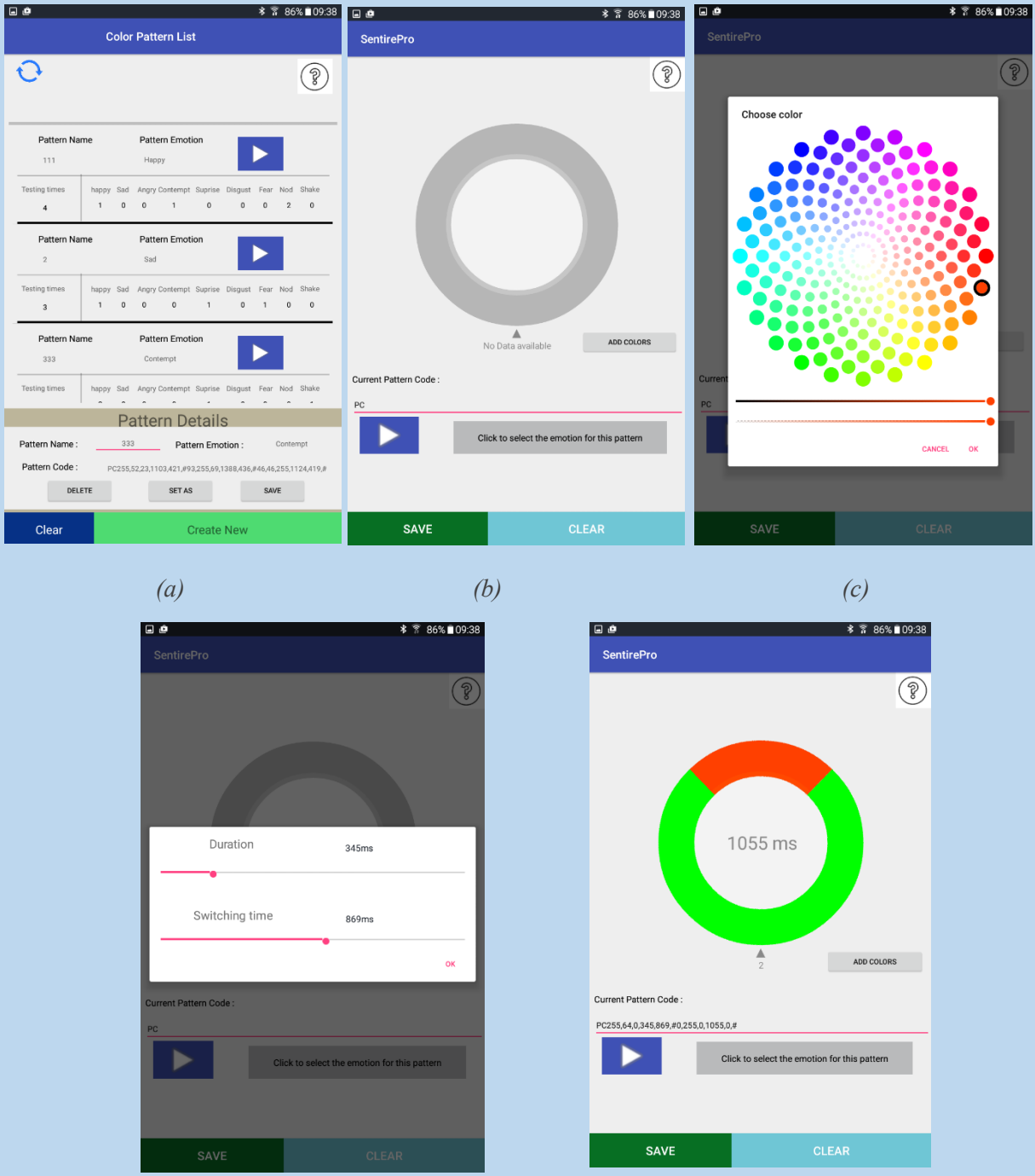
```
mCubicValueLineChart.clearChart();
amplitudes = myPaintView.getAmplitudes(50);
amplitudes.add(0, 0);
amplitudes.add(0);

ValueLineSeries series = new ValueLineSeries();
series.setColor(getResources().getColor(R.color.colorAccent));

for(int i=0; i < amplitudes.size(); i = i + 3) {
    mValue += amplitudes.get(i).toString()+",";
    series.addPoint(new ValueLinePoint(((new Integer(i*40)).toString()+"ms"),
amplitudes.get(i)));
}
mCubicValueLineChart.addSeries(series);
mCubicValueLineChart.startAnimation();
```

4.6 Color Patterns

The first activity that allows users to see designed color pattern is similar to the previous section as well (*Figure 4-8 (a)*). To create a new color pattern, the color picker is used.



(a) *(b)* *(c)*

(d) *(e)*

Figure 4-8:

The required packages are followed:

```
import com.flask.colorpicker.ColorPickerView;
import com.flask.colorpicker.builder.ColorPickerClickListener;
import com.flask.colorpicker.builder.ColorPickerDialogBuilder;
```

```
import org.eazegraph.lib.charts.PieChart;
import org.eazegraph.lib.models.PieModel;
```

In order to get the value select color in *Figure 4-8 (c)*, these methods are used:

```
ColorPickerDialogBuilder
    .with(colcreate.this)
    .setTitle("Choose color")
    .initialColor(mColor)
    .wheelType(ColorPickerView.WHEEL_TYPE.FLOWER)
    .density(12)
    .setPositiveButton("ok", new ColorPickerClickListener() {
        @Override
        public void onClick(DialogInterface dialog,
                            int selectedColor, Integer[] allColors) {
            mColor = selectedColor;
```

Users could set the duration for the added color, together with the switching time this color takes to change to next color. (*Figure 4-8 (d)*).

4.7 Questionnaires

4.7.1 Personal information

To better analyze the relationship between emotions and vibration/color patterns for a different group of people, the personal information is also required in this activity. However, the user can skip this step if they refuse to provide private information. (*Figure 4-9 (a)*)

This personal information would be transmitted to next activity as a part of questionnaire data, and would be saved in an Excel file. To achieve this, the variable used to keep the data should be defined as public:

```
public static String quesNationality,quesAge,quesGender,quesDisability;
```

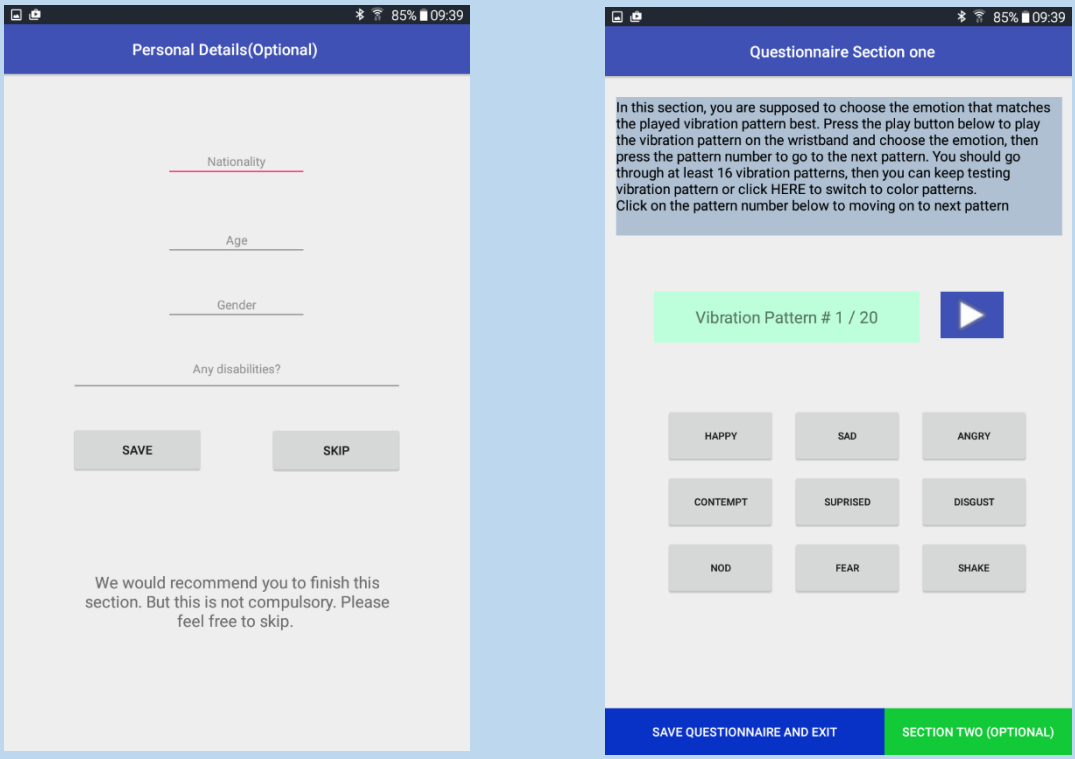


Figure 4-9: *(a)* *(b)*

4.7.2 Select Emotion for Displayed pattern

In this activity (*Figure 4-9 (b)*), users are supposed to choose the emotion that matches the played vibration or color pattern best. And press the pattern number to test next pattern. The user should test at least 16 vibration patterns and ten color patterns. They can keep testing vibration pattern until finishing them all, or press the context on the top to switch to color pattern.

As mentioned in section 4.7.1, the personal information would be available in this activity in writeQdetail() method (which is designed for saving this person's answers, as well as his/her personal information in an individual Excel file). Following codes are used (Note that the name of previous activity is ques_info):

```
// personal details
mystring = new Label(1, 1, ques_info.quesNationality);
sheet.addCell(mystring);
mystring = new Label(2, 1, ques_info.quesAge);
sheet.addCell(mystring);
mystring = new Label(3, 1, ques_info.quesGender);
sheet.addCell(mystring);
mystring = new Label(4, 1, ques_info.quesDisability);
sheet.addCell(mystring);
```

Considering that there would likely be many people taking this questionnaire, so their questionnaire data should be saved in different files. Therefore, another Excel file ('Questionnairenumber.xls') is created to record how many persons have finished the questionnaire (used to decide the name of Excel file that saves questionnaire information). It also records how many patterns has been tested in the previous questionnaire, so that in the next questionnaire, the system could start with the pattern followed.

Followed is an example:

Assume that there are 30 library vibration patterns, 20 real-time vibration patterns, and 25 color patterns. The first person took the questionnaire test 8 library vibration patterns, 8real-time vibration patterns and 14 color patterns. Then in 'Questionnairenumber.xls', four numbers would be recorded. The first one is the how many persons have finished the questionnaire; it is 1; the second number is how many library vibration patterns has been played, it is 8; similarly, the third number is 8, and the fourth number is 14. Then in the second questionnaire, the file name would be 'Questionnaire2.xls', it would play library vibration pattern from the ninth one, and so on.

Followed are the codes to read the 'Questionnairenumber.xls' file:

```
 Cell cell = sheet.getCell(0, 0);
n = Integer.valueOf(cell.getContents()).intValue();
cell = sheet.getCell(1,0);
lastLVnumb = Integer.valueOf(cell.getContents()).intValue();
cell = sheet.getCell(2,0);
lastRTVnumb = Integer.valueOf(cell.getContents()).intValue();
```

```
cell = sheet.getCell(3,0);
lastCnumb = Integer.valueOf(cell.getContents()).intValue();
```

Screenshot for the ‘Questionnairenumber.xls’ and ‘Questionnaire**N**.xls’ is in *Figure 4-10*. (**N** is an integer)

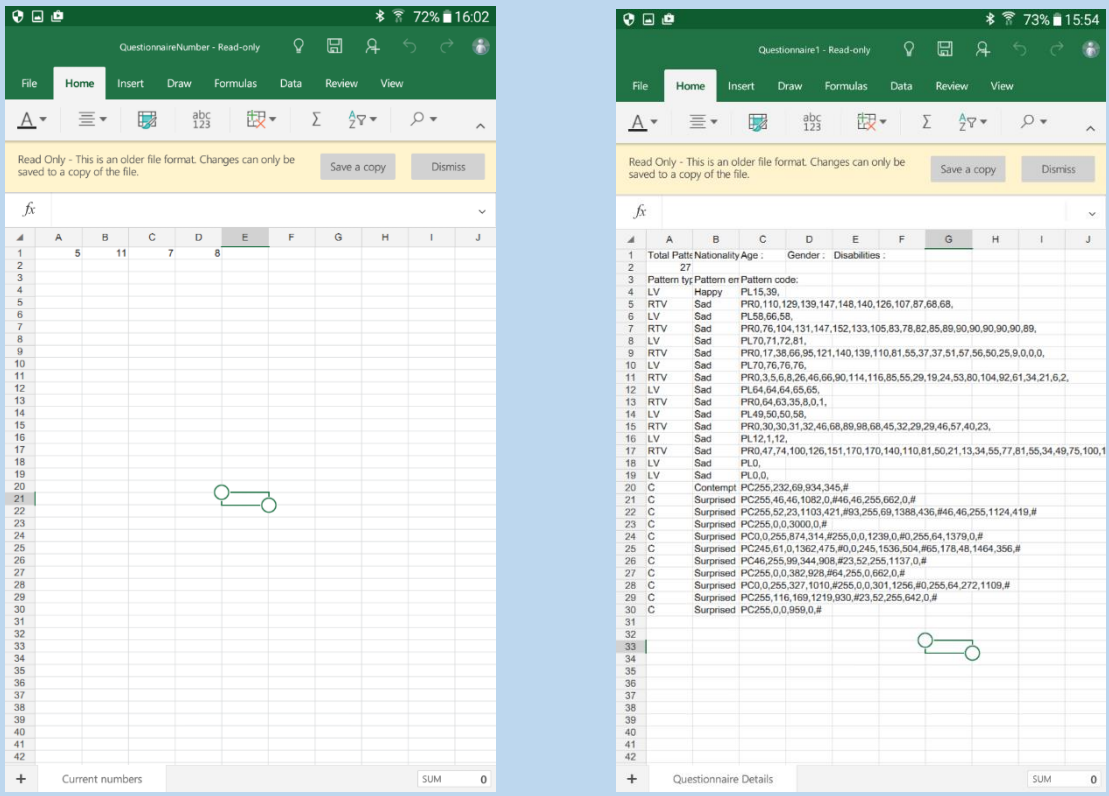


Figure 4-10: *(a)* *(b)*

4.7.3 Pattern Comparison

In this activity, questionnaire detail is not saved in ‘Questionnaire**N**.xls’. It only adds scores for the designed patterns. The following method is used to generate a random pattern number to compare:

```
private int myrand(int max){   // random number: 0 to max -1
    Random random=new Random();
    return random.nextInt(max);
}
```

The layout is shown in *Figure 4-11 (a)*.

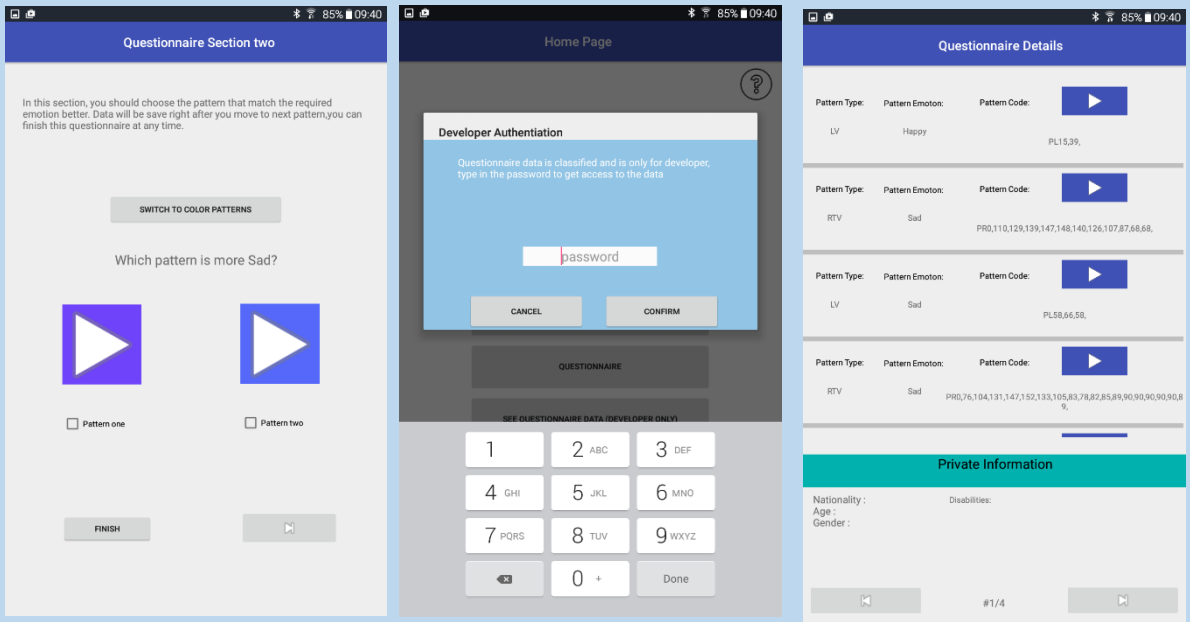


Figure 4-11: (a) *(b)* *(c)*

4.7.4 See Questionnaire Data

This activity requires Developer Authentication, and the password is '0620'. A similar listview is used as in section 4.4. It can display all the pattern details, and wristband would play the pattern once the developer clicks on the play button. Developers could go through all the questionnaire file with the previous and next button on the bottom. (*Figure 4-11 (a)*)

Following method is used to achieve reading different Excel files:

```
readthis("Questionnaire" + Integer.toString(curnumb) + ".xls");
```

Chapter 5 System Integration & Testing

5.1 Mechanical Design

Due to the size and kinds of the electronic components are different from Qisong's design, a new shell for the wristband requires to be created. 2mm thick acrylic plate is used to build the shell. On an A4 size acrylic plate, 12 pieces of 60x50(mm) layer is sketched for laser cutting (*Figure 5-1*).

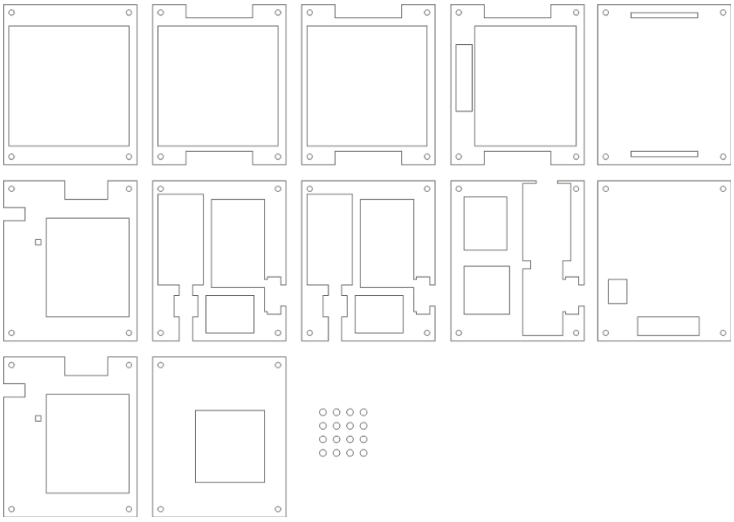


Figure 5-1: Laser cut design

The full-size design is available in Appendices A.2.

5.2 Assembling

The Lithium Polymer battery and the ERM vibration motor are attached to the bottom layer of the wristband (*Figure 5-2*). Considered the vibration that created by the motor, hot glue is used to stick the ERM to the acrylic plate.

Above the battery and motor lie the battery charging module, the accelerometer and the motor driver (*Figure 5-3*). The I^2C wires of the accelerometer and the driver are connected together (Green and yellow wires in *Figure 5-4*). The switch and the button are attached here as well.

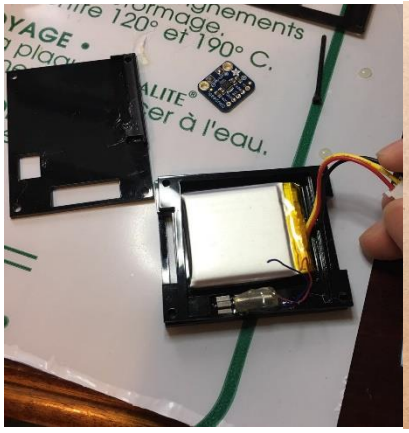


Figure 5-2: Battery and motor

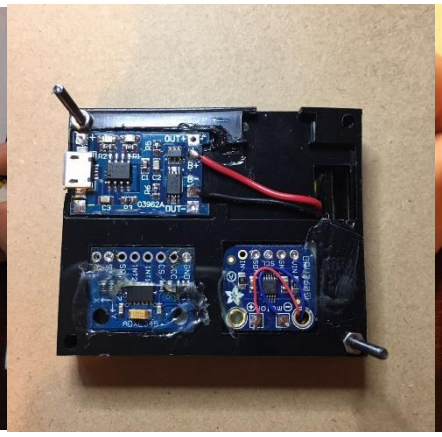


Figure 5-3: Charging module, accelerometer and motor driver

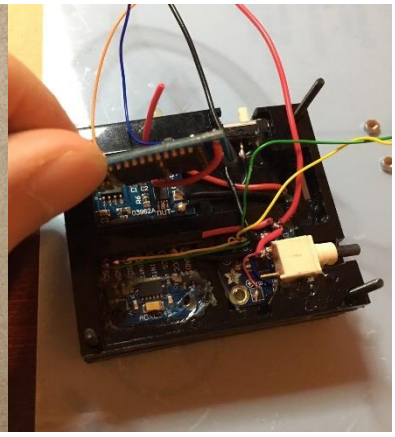


Figure 5-4: I^2C connections

The microcontroller and the Bluetooth module is fixed on the upper layer. (*Figure 5-5*). The blue and orange wires enable the UART communication between Arduino and HC-05.

The OLED screen adheres to the top layer, as shown in *Figure 5-6*. M2*32 Screws and nuts are used to assemble this wristband (Figure 5-7).

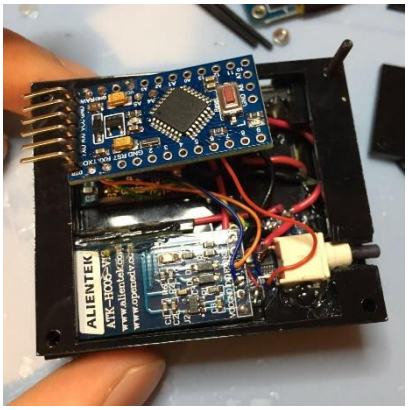


Figure 5-5: Arduino and HC-05

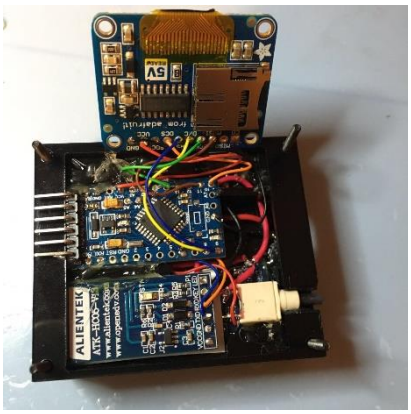


Figure 5-6: OLED screen

Figure 5-5: The wristband

5.3 Testing

5.3.1 Battery

The MicroUSB socket on battery charging module is used for charging the Lithium Polymer battery. If a 5V 1000mA adapter is applied, it will take around 80 minutes to charge the battery

from 3.6V to 4.18V.

Calculate the system total power consumption is very different as it mainly depends on how frequently the patterns are played.

Theologically, the approximate battery life for standby mode would be:

$$t_s = battery\ \ energy \div (Arduino's\ \ power + HC - 05's\ \ power[20] + OLED's\ \ power[19])=$$

$$(3.7V \times 600mAh) \div (3.3V \times 25mA + 3.7V \times 50mA + 3.3V \times 25mA)=6.34\text{h}$$

Assuming the vibration pattern is always playing, then:

$$t_p = (3.7V \times 600mAh) \div (3.3V \times 25mA + 3.7V \times 50mA + 3.3V \times 25mA + 3.3V \times 75mA)=3.7\text{h}$$

However, when testing the wristband in actual operation, the battery life is shorter.

In standby mode (Bluetooth connected), the wristband could only work for around 5 hours. Once playing Happy emotion every 20 seconds, with library vibration sequence: PL12,1,12 and color sequence: PC255,0,0,959,9,#, the wristband worked for 2 hours and 30 minutes.

5.3.2 Pattern playing and uploading

More than 8 hours' testing shows that the integrated SAD system can give accurate haptic and color feedback to the user. Users can easily create new vibration or color pattern at their will. The developers and researchers could get access to all the saved pattern details and questionnaire data on the Android device. All those data saved in Excel file can be easily imported into Matlab for further psychological analysis.

However, as all the play functions (for DRV2605 and SSD1331) are achieved by loops, so the playing pattern would not be interrupted when next data is received. If the user keeps sending data when the previous play function has not been finished. The Arduino might run out of its Random-Access Memory (RAM), causing the instability of the hardware system. To avoid this situation, the user should only send Bluetooth data after the previous pattern is finished playing.

Chapter 6 Pathways to Impact

6.1 Current Position in Research & Development Process

This project is current between 'Concept for a product' and 'Further development'. 'Regulatory framework' is partly done as well.

It started from the initial research of J. Ye, P. Hands and R. Goutcher [1], and ended up with a prototype wearable device along with a companion Android app.

This project aims at developing a mature SAT system for visually-impaired or ASD users, which can convert visual cues that not available for users to haptic or color feedback. This SAD system would hopefully improve conversation quality of the target population, avoid degradation in their communication skills and reduce social isolation.

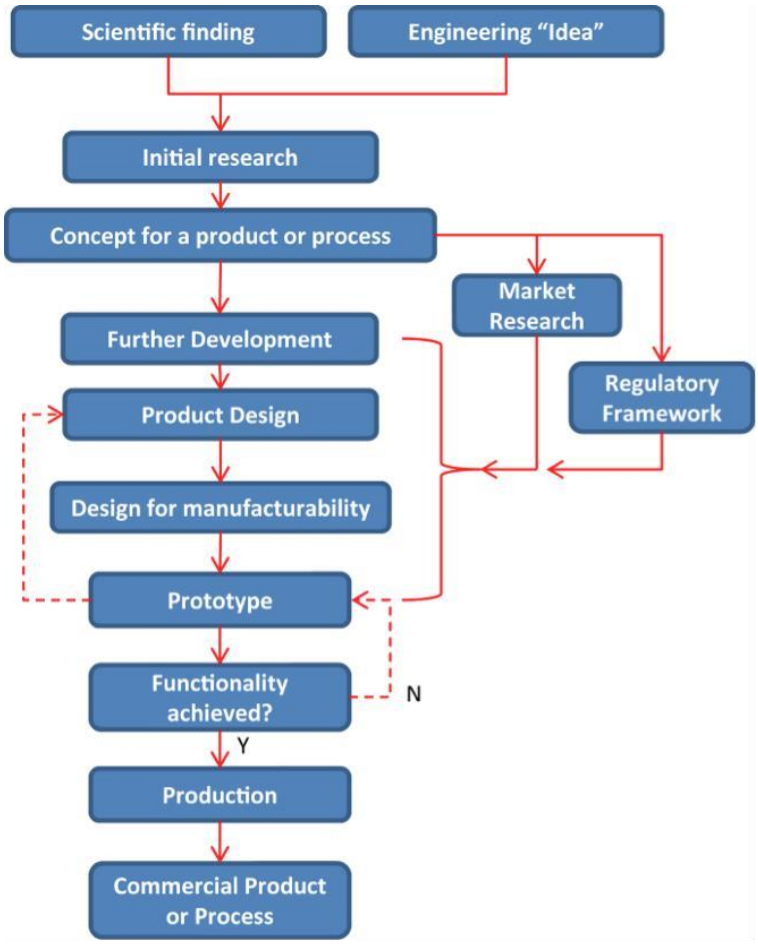


Figure 5-1: Typical Commercial Product Development Flow [31]]

My efforts and works successfully pushed the whole project into the 'Further development' step. To achieve the proposed SAD system, more development and research needs to be done. For example, run the questionnaire among the target population to crowd-source more accurate and reliable database on the relationship between emotions and different vibration and color patterns. Because visually-impaired people may have different opinions on colors and emotions compared to normal person.

6.2 Impacts of the Project

6.2.1 SOCIAL IMPACT

As mentioned above, this project aims at helping visually-impaired and ASD people to improve their communication and life quality. They would build up confidence during the daily conversation and fit better in the world.

6.2.2 ECONOMIC IMPACT

As described in the introduction (section 1.1), there are an enormous amount of visually-impaired people (and people with ASD) in the world. Also, very limited SADs is available now, and existing SAD system is bulky (section 1.2). Therefore once this SAD system is commercialized in the future, the vast number of potential customer would ensure the profit of this product.

6.2.3 Academic Impact

Hopefully, a relevant paper would be published recently in this area, which would likely to arouse wide attentions and interests in the related field.

This device also provides an easy-to-use tool for psychologists to study the relationships between human emotions and vibrations or colors, and the possibility of converting some of the human's nature senses to another one.

Chapter 7 Conclusions & Future work

7.1 Conclusion

To conclude, this project succeeds in building a new wearable haptic device with an Android app, fixing all the existing defects and problems. The gradual change between colors is also achieved as expected. This prototype SAD system can generate recognizable vibration and color feedback that represent different emotions. With the Android app, the new alerting pattern could be designed with high usability and flexibility; developers could use the integrated system to crowd-source questionnaires easily for deeper research.

7.2 Future Work

For this project, crowd-source more questionnaire data is of vital importance and should be done in next step. The new alerting pattern could be designed after analyzing these data on a psychological level. Integration of the entire system (with the camera glasses) could be done with our collaborator at the University of St Andrews and the University of Sterling.

For this wristband, the developer could try to realize some more features. For example: Enable users to interrupt the play process by sending new data. Use the accelerometer to achieve ‘raise to wake’, which would save power consumption on OLED screen. Get access to the rest power in the battery and display it on the screen, which could remind the user to charge the battery.

For the Android app, an automatic ranking based on scores of different emotions can be achieved. Also, as the UI does not look pretty, some more art design could be applied to it.

Acknowledgments

I would like to acknowledge my supervisor: Dr. Philip Hands for his continually support and constructive suggestions on this project. I would like to acknowledge my project examiner Dr. Les Haworth for his valuable advice and feedback on my project.

I would like to thank Dr. Juan Ye and Dr. Ross Goutcher for their help when designing the app. I would like to thank Dr. Adam Stokes for his help on laser cutting.

I would like to acknowledge Qisong Wang for his help and guidance on building the Android app. I would like to thank Yang Guo for her help on Android Studio and the tools to build the wristband. I would like to thank Iain for his help on creating the wristband.

References

[1] L. Murray, R. Goutcher, P. Hands, J. Ye. "Capturing Social Cues with Imaging Glasses". : University of St Andrews; 2016.

[2] A. Mehrabian and S. R. Ferris, "Inference of attitudes from nonverbal communication in two channels," *Journal of consulting psychology,* vol. 31, no. 3, p. 248, 1967.

[3] A. Mehrabian and M. Wiener, "Decoding of inconsistent communications," *Journal of personality and social psychology,* vol. 6, no. 1, p. 109, 1967.

[4] World Health Organization, "Visual impairment and blindness," WHO Media Centre, August 2014. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs282/en/ . [Accessed 26 April 2017].

[5] Centers for Diseases Control and Prevention, "Autism Spectrum Disorder (ASD)", Centers for Diseases Control and Prevention, 31 March 2016. [Online]. Available: http://www.cdc.gov/ncbddd/autism/data.html . [Accessed 26 April 2017].

[6] National Health Service, "Autism spectrum disorder (ASD)", Autism spectrum disorder, March 2016. [Online]. Available: http://www.nhs.uk/conditions/Autistic-spectrum-disorder/Pages/Introduction.aspx . [Accessed 26 April 2017].

[7] S. Krishna and et al., "VibroGlove: an assistive technology aid for conveying facial expressions," in CHI'10 Extended Abstracts on Human Factors in Computing Systems, ACM, 2010.

[8] S. u. Réhman, "Expressing Emotions through Vibration for Perception and Control," Umeå University, Umeå , 2010.

[9] R. Plutchik, "The Nature of Emotions Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice," *American Scientist,* vol. 84, no. 1, pp. 344-350, 2001.

[10] J. Ye, P. Hands and R. Goutcher, *Understanding Interaction between Human and Machine Sensory Perception,* Edinburgh: Unpublished, 2015.

[11] Q. Wang, *Sensory Assistive Device Interpreting Video for Visually-Impaired Users*, Edinburgh: Unpublished, 2016.

[12] Intel Corporation, “Intel Edison Compute Module,” Intel Corporation, 1 Jan 2016. [Online]. Available: https://software.intel.com/en-us/iot/hardware/edison . [Accessed 26 April 2017].

[13] ARDUINO, Arduino Pro Mini, ARDUINO, 2017. [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardProMini . [Accessed 26 April 2017].

[14] RedBear, “Redbear Duo”, RedBear, 2016. [Online]. Available: https://redbear.cc/product/wifi-ble/redbear-duo.html . [Accessed 26 April 2017].

[15] Adafruit, “0.96" mini Color OLED”, Adafruit, May 2015, [Online]. Available: https://learn.adafruit.com/096-mini-color-oled/wiring . [Accessed 26 April 2017].

[16] Precision Microdrives, “Understanding ERM Vibration Motor Characteristics,” Precision Microdrives, 1 Janurary 2016. [Online]. Available: https://www.precisionmicrodrives.com/vibration-motors/eccentric-rotating-mass-vibration-motors-erms . [Accessed 26 April 2017].

[17] Precision Microdrives, “Driving Linear Resonance Vibration Actuators,” Precision Microdrives, 1 January 2016. [Online]. Available: https://www.precisionmicrodrives.com/vibration-motors/linear-resonant-actuators-lras . [Accessed 26 April 2017].

[18] Adadruit, “Adafruit DRV2605 Haptic Controller Breakout”, Adafruit, December 2014. [Online]. Available: https://learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout/downloads . [Accessed 26 April 2017].

[19] Adafruit, “0.96" mini Color OLED”, Adafruit, May 2015, [Online]. Available: https://learn.adafruit.com/096-mini-color-oled/overview . [Accessed 26 April 2017].

[20] ALIENTEK, “ATK-HC05-V11”, ALIENTEK, December 2012. [Online]. Available: http://www.openedv.com/thread-12486-1-1.html . [Accessed 26 April 2017].

[21] Cambridge Silicon Radio Limited, “Single Chip Bluetooth® v2.0+EDR System”, Cambridge Silicon Radio Limited, July 2005. [Online]. Available: http://yourduino.com/docs/CSR-BC417-datasheet.pdf . [Accessed 26 April 2017].

[22] 陈声宇, “Specification of Li-polymer Rechargeable Battery”, , Jan, 2012. [Online]. Available: http://www.farnell.com/datasheets/1666649.pdf?_ga=1.189825512.270953201.1486150287 . [Accessed 26 April 2017].

[23] TELESKY, “TELESKY 1A Li-ion battery charging & protection board, charging + protection two-in-one, charging module (锂电池充电与保护一体板 充电+保护 二合一 充电模块)”, TELESKY, . [Online]. Available: https://world.tmall.com/item/41345485036.htm?spm=a312a.7700824.w4011-8679163093.36.dRt9tf&id=41345485036&rn=3d25ba2ea7962d44d8c347f896b2a8e4&abbucket=13 . [Accessed 26 April 2017].

[24] Seeed Studio, “Accelerometer ADXL345 Library,” Seeed Studio, 20 April 2014. [Online]. Available: https://github.com/Seeed-Studio/Accelerometer_ADXL345 . [Accessed 26 April 2017].

[25] Arduino, “Wire Library,” Arduino, 2017. [Online]. Available: https://www.arduino.cc/en/reference/wire . [Accessed 26 April 2017].

[26] Adafruit, “Adafruit DRV2605 Haptic Controller Breakout”, Adafruit, 11 April 2016. [Online]. Available: https://learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout/wiring-and-test . [Accessed 26 April 2017].

[27] Arduino, “SPI Library,” Arduino, 2017. [Online]. Available: https://www.arduino.cc/en/reference/SPI . [Accessed 26 April 2017].

[28] Arduino, “EEPROM Library,” Arduino, 2017. [Online]. Available: https://www.arduino.cc/en/Reference/EEPROM . [Accessed 26 April 2017].

[29] Autodesk, “How To: Create an Android App With Android Studio to Control LED”, Deyson in arduino, 2016. [Online]. Available: http://www.instructables.com/id/Android-Bluetooth-Control-LED-Part-2/ . [Accessed 26 April 2017].

[30] Demo Source and Support, “Download jxl.jar : jxl « j « Jar File Download”, Demo Source and Support, . [Online]. Available: http://www.java2s.com/Code/Jar/j/Downloadjxljar.htm . [Accessed 26 April 2017].

[31] Blackfizz, “EazeGraph”, Github.Inc, 2017. [Online]. Available: https://github.com/blackfizz/EazeGraph . [Accessed 26 April 2017].

Appendices

A.1 DRV2605 Waveform Library Effects List

EFFECT ID NO.	WAVEFORM NAME	EFFECT ID NO>	WAVEFORM NAME	EFFECT ID NO.	WAVEFORM NAME
1	Strong Click - 100%	42	Long Double Sharp Click Medium 2 – 80%	83	Transition Ramp Up Long Smooth 2 – 0 to 100%
2	Strong Click - 60%	43	Long Double Sharp Click Medium 3 – 60%	84	Transition Ramp Up Medium Smooth 1 – 0 to 100%
3	Strong Click - 30%	44	Long Double Sharp Tick 1 – 100%	85	Transition Ramp Up Medium Smooth 2 – 0 to 100%
4	Sharp Click - 100%	45	Long Double Sharp Tick 2 – 80%	86	Transition Ramp Up Short Smooth 1 – 0 to 100%
5	Sharp Click - 60%	46	Long Double Sharp Tick 3 – 60%	87	Transition Ramp Up Short Smooth 2 – 0 to 100%
6	Sharp Click - 30%	47	Buzz 1 – 100%	88	Transition Ramp Up Long Sharp 1 – 0 to 100%
7	Soft Bump - 100%	48	Buzz 2 – 80%	89	Transition Ramp Up Long Sharp 2 – 0 to 100%
8	Soft Bump - 60%	49	Buzz 3 – 60%	90	Transition Ramp Up Medium Sharp 1 – 0 to 100%
9	Soft Bump - 30%	50	Buzz 4 – 40%	91	Transition Ramp Up Medium Sharp 2 – 0 to 100%
10	Double Click - 100%	51	Buzz 5 – 20%	92	Transition Ramp Up Short Sharp 1 – 0 to 100%
11	Double Click - 60%	52	Pulsing Strong 1 – 100%	93	Transition Ramp Up Short Sharp 2 – 0 to 100%
12	Triple Click - 100%	53	Pulsing Strong 2 – 60%	94	Transition Ramp Down Long Smooth 1 – 50 to 0%
13	Soft Fuzz - 60%	54	Pulsing Medium 1 – 100%	95	Transition Ramp Down Long Smooth 2 – 50 to 0%
14	Strong Buzz - 100%	55	Pulsing Medium 2 – 60%	96	Transition Ramp Down Medium Smooth 1 – 50 to 0%
15	750 ms Alert 100%	56	Pulsing Sharp 1 – 100%	97	Transition Ramp Down Medium Smooth 2 – 50 to 0%
16	1000 ms Alert 100%	57	Pulsing Sharp 2 – 60%	98	Transition Ramp Down Short Smooth 1 – 50 to 0%
17	Strong Click 1 - 100%	58	Transition Click 1 – 100%	99	Transition Ramp Down Short Smooth 2 – 50 to 0%
18	Strong Click 2 - 80%	59	Transition Click 2 – 80%	100	Transition Ramp Down Long Sharp 1 – 50 to 0%
19	Strong Click 3 - 60%	60	Transition Click 3 – 60%	101	Transition Ramp Down Long Sharp 2 – 50 to 0%
20	Strong Click 4 - 30%	61	Transition Click 4 – 40%	102	Transition Ramp Down Medium Sharp 1 – 50 to 0%
21	Medium Click 1 - 100%	62	Transition Click 5 – 20%	103	Transition Ramp Down Medium Sharp 2 – 50 to 0%
22	Medium Click 2 - 80%	63	Transition Click 6 – 10%	104	Transition Ramp Down Short Sharp 1 – 50 to 0%
23	Medium Click 3 - 60%	64	Transition Hum 1 – 100%	105	Transition Ramp Down Short Sharp 2 – 50 to 0%
24	Sharp Tick 1 - 100%	65	Transition Hum 2 – 80%	106	Transition Ramp Up Long Smooth 1 – 0 to 50%
25	Sharp Tick 2 - 80%	66	Transition Hum 3 – 60%	107	Transition Ramp Up Long Smooth 2 – 0 to 50%
26	Sharp Tick 3 – 60%	67	Transition Hum 4 – 40%	108	Transition Ramp Up Medium Smooth 1 – 0 to 50%
27	Short Double Click Strong 1 – 100%	68	Transition Hum 5 – 20%	109	Transition Ramp Up Medium Smooth 2 – 0 to 50%
28	Short Double Click Strong 2 – 80%	69	Transition Hum 6 – 10%	110	Transition Ramp Up Short Smooth 1 – 0 to 50%
29	Short Double Click Strong 3 – 60%	70	Transition Ramp Down Long Smooth 1 – 100 to 0%	111	Transition Ramp Up Short Smooth 2 – 0 to 50%
30	Short Double Click Strong 4 – 30%	71	Transition Ramp Down Long Smooth 2 – 100 to 0%	112	Transition Ramp Up Long Sharp 1 – 0 to 50%
31	Short Double Click Medium 1 – 100%	72	Transition Ramp Down Medium Smooth 1 – 100 to 0%	113	Transition Ramp Up Long Sharp 2 – 0 to 50%
32	Short Double Click Medium 2 – 80%	73	Transition Ramp Down Medium Smooth 2 – 100 to 0%	114	Transition Ramp Up Medium Sharp 1 – 0 to 50%
33	Short Double Click Medium 3 – 60%	74	Transition Ramp Down Short Smooth 1 – 100 to 0%	115	Transition Ramp Up Medium Sharp 2 – 0 to 50%
34	Short Double Sharp Tick 1 – 100%	75	Transition Ramp Down Short Smooth 2 – 100 to 0%	116	Transition Ramp Up Short Sharp 1 – 0 to 50%
35	Short Double Sharp Tick 2 – 80%	76	Transition Ramp Down Long Sharp 1 – 100 to 0%	117	Transition Ramp Up Short Sharp 2 – 0 to 50%
36	Short Double Sharp Tick 3 – 60%	77	Transition Ramp Down Long Sharp 2 – 100 to 0%	118	Long buzz for programmatic stopping – 100%
37	Long Double Sharp Click Strong 1 – 100%	78	Transition Ramp Down Medium Sharp 1 – 100 to 0%	119	Smooth Hum 1 (No kick or brake pulse) – 50%
38	Long Double Sharp Click Strong 2 – 80%	79	Transition Ramp Down Medium Sharp 2 – 100 to 0%	120	Smooth Hum 2 (No kick or brake pulse) – 40%
39	Long Double Sharp Click Strong 3 – 60%	80	Transition Ramp Down Short Sharp 1 – 100 to 0%	121	Smooth Hum 3 (No kick or brake pulse) – 30%
40	Long Double Sharp Click Strong 4 – 30%	81	Transition Ramp Down Short Sharp 2 – 100 to 0%	122	Smooth Hum 4 (No kick or brake pulse) – 20%
41	Long Double Sharp Click Medium 1 – 100%	82	Transition Ramp Up Long Smooth 1 – 0 to 100%	123	Smooth Hum 5 (No kick or brake pulse) – 10%

Figure A.1 DRV2605 Waveform Library Effects List [18]

A.2 Laser Cutting Design (A4 size)

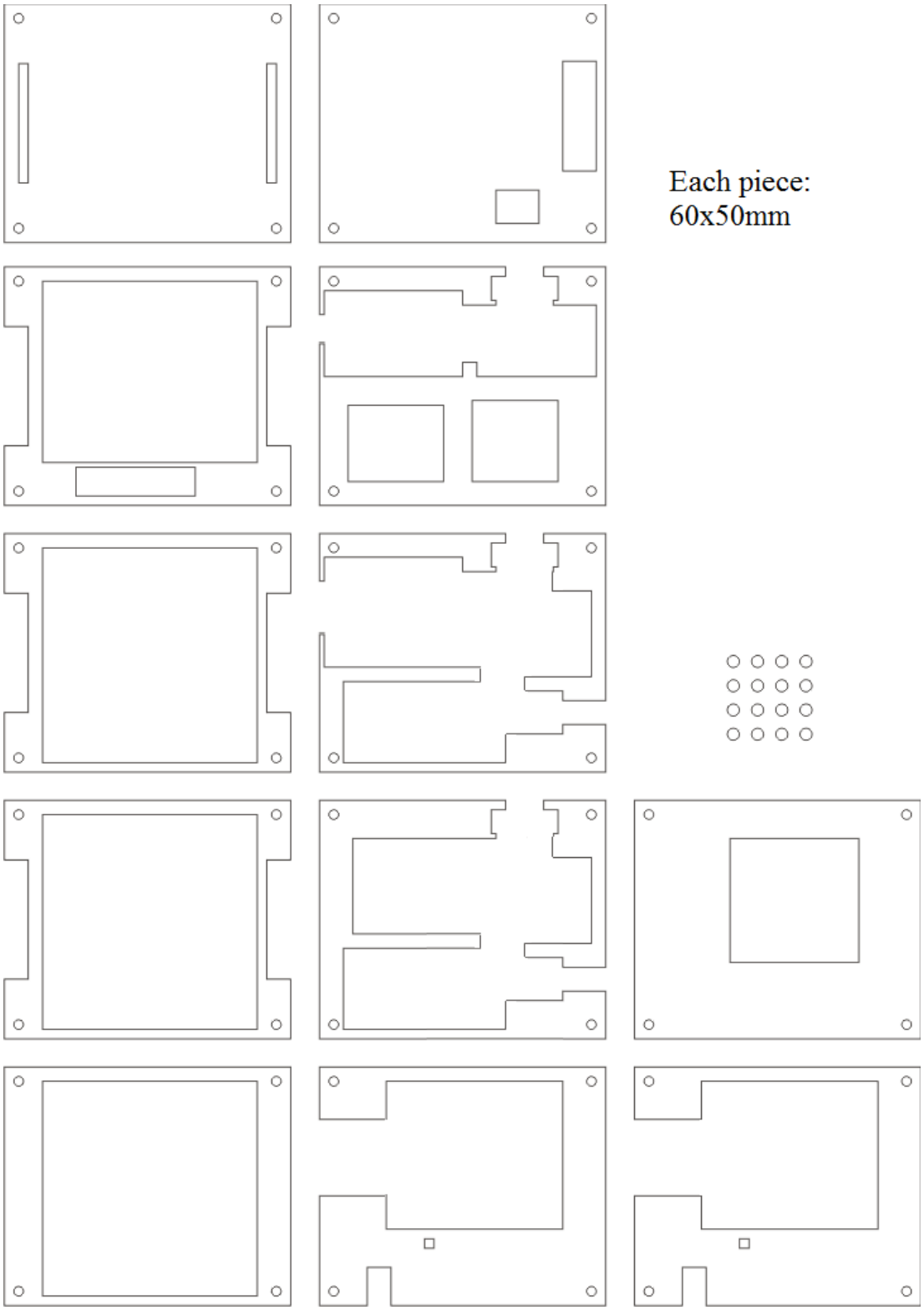


Figure A.2 Laser Cutting Design (A4 size)

A.3 Example Method for Creating Library-Vibration-Pattern-Detail Excel File

```
private void writeCExcel() {
    try {
        //  open the file
        File file=new File(Environment.getExternalStorageDirectory(),"Cptn.xls");
        //  create the sheet ,name it
        WritableWorkbook book = Workbook.createWorkbook(file);
        WritableSheet sheet = book.createSheet("My library vibration patterns", 0);

        sheet.mergeCells(0,0,2,0);//  merge the first 3 cell on row 1 and cell 4,5
        sheet.mergeCells(3,0,4,0);//    first 3:column  first 0: row
        sheet.mergeCells(6,0,14,0);
        //  row 1: total number of current ptn
        Label mystring = new Label(0, 0, "Total pattern number :");
        sheet.addCell(mystring);
        jxl.write.Number mynumb = new jxl.write.Number(3, 0, ptn_numbC);
        sheet.addCell(mynumb);
        mystring = new Label(6, 0, "Scores for different emotions(scores comes from
questionnaire)");
        sheet.addCell(mystring);

        //  labels for different data
        mystring = new Label(0, 1, "Pattern Number");
        sheet.addCell(mystring);
        mystring = new Label(1, 1, "Pattern Name");
        sheet.addCell(mystring);
        mystring = new Label(2, 1, "Pattern Type");
        sheet.addCell(mystring);
        mystring = new Label(3, 1, "Pattern Emotion");
        sheet.addCell(mystring);
        mystring = new Label(4, 1, "Pattern Code");
        sheet.addCell(mystring);
        mystring = new Label(5, 1, "Testing Times");
        sheet.addCell(mystring);


        //  write data

        for (int row = 2; row <= ptn_numbC + 1; row++)
        {
```

```
            mynumb = new jxl.write.Number(0, row, myptn_numbC[ row - 2 ]);
            sheet.addCell(mynumb);
            mystring = new Label( 1,row, myptn_nameC[ row - 2 ]);
            sheet.addCell(mystring);
            mystring = new Label( 2,row, myptn_typeC[ row - 2 ]);
            sheet.addCell(mystring);
            mystring = new Label( 3,row, myptn_emotC[ row - 2 ]);
            sheet.addCell(mystring);
            mystring = new Label( 4,row, myptn_codeC[ row - 2 ]);
            sheet.addCell(mystring);
            mynumb = new jxl.write.Number(5, row, myptn_timesC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(6, row, shappyC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(7, row, ssadC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(8, row, sangryC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(9, row, scontemptC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(10, row, ssupriseC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(11, row, sdisgustC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(12, row, sfearC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(13, row, snodC[ row - 2 ]);
            sheet.addCell(mynumb);
            mynumb = new jxl.write.Number(14, row, sshakeC[ row - 2 ]);
            sheet.addCell(mynumb);

        }
        book.write();
        book.close();
    } catch (WriteException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```