

香港大學工程學院主辦

Organised by the Faculty of Engineering, The University of Hong Kong

地震探測器設計比賽

Earthquake Detector Design Competition

文字報告

Text Report

高級組：第 63 隊 Senior Category: Team No. 63

420 Enhance Your Calm

香港神託會培基書院
Stewards Pooi Kei College

組員：

麥澤榮 Mak Chak Wing

吳灝民 Ng Ho Man

王俊禧 Wong Chun Hei James

黃達濠 Wong Tat Ho

老師：

柳已丞老師

Mr. Harry Lau

00 目錄

01	引言	09.01	分辨真假地震及地震實體波種類
02	這是什麼？	09.02	分辨地震強度
03	運作原理	10	其他資料
	03.01 加速度計運作原理	11	反思
	03.02 如何分辨真假地震	11.01	可改善之處
04	所使用的元件	11.02	成就達成
	04.01 不使用智能手機加速度計原因	12	總結
05	軟件設計	13	鳴謝
	05.01 Android 應用程式	14	附錄
	05.02 網頁	14.01	分工
06	硬件設計	14.02	參考資料
07	操作方法	14.03	電路圖
08	應用範疇	14.04	原始碼
09	測試		

01 引言

在現今科技的世代，測量地震的方法很多，有的更是很便宜就能做到的，但現今很多的地震儀都未普及化，而且一些落後地方都需要比較便宜而實用的地震警報設施，去避免更多的慘況，所以我們這次會將地震儀去適用於不同的人士，有可能生活在城市，有可能生活在沒有網絡的地方，都能夠即時發出警號給他們，去提醒他們離開，所以我們會討論一下我們這個裝置如何實現到這些目標，去用什麼電子元件，去測量地震，並講述我們的研究和測試，更會探討到用戶的兼容性，好處和可改善之處等等，以作未來的發展用途，去真正提供給有需要的人去使用，去避免地震帶來的災難。

02 這是什麼？

地震探測器是能夠探測地殼震動、並紀錄地震數據的儀器。從古代的候風地動儀（圖 02），到現代化的先進電子儀器，地震探測器測量地震強度的方法很多元化。雖然地震探測器沒法預測地震，可是仍在地震學中有重要的功用，例如分析地震波的頻率、利用多個地震探測器來推測震央的位置等等。它繪製的圖表大多以 x - 軸為時間軸；並以 y - 軸為強度表示地震的數據。可以透過圖表看到地震震動的頻率。

我們 420 Enhance Your Calm 團隊設計的地震探測器，以加速度計測試地震強度、以及使用內置的地震資料及圖表分辨震動是否由地震所引起。這個方式能夠減低地震探測器意外被激活而引起的誤差。



圖 02 ↑ 候風地動儀
(網上圖片)

03 運作原理

現時的地震探測器都以電子方式量度地震強度。因此可藉著可靠的電子零件，再為其加入結合物理邏輯的電腦程式，來測量地震。以下列出了我們的地震探測器所使用的三個基本的物理定律以及原理。

03.01 加速度計運作原理

加速度計的原理，是運用牛頓物理第二定律 ($F = ma$ 或者 $a = F / m$)。即是說加速度（單位： $m \cdot s^{-2}$ ）與受到的合力（英語：Resultant force、單位：牛頓（英語：Newton））成正比，與其質量（單位：kg）成反比。但是所謂的「加速度計」其實是在測量其受的力，而並非加速度，是藉著測量施予其的力再以牛頓第二定律間接計算出加速度。因此在測量之前必須先校準，以免得到不準確的數據。

（有關校準地震探測器的詳情請參閱：07 操作方法）

03.02 如何分辨真假地震

我們設計的地震探測器的一大賣點，是地震探測器如何分辨震動是否地震所引起。城市週遭則有許多非地震的震動（例如：重型車輛駛過、人群在附近遊走、惡意觸碰等），郊野則有好奇的小動物在觸碰地震探測器。若沒有這功能，這些都會影響其準繩度。因此加入這項功能，能大大提高警報誤鳴的機會。以下則為其分辨震動的原理。

利用短期平均值（英語：Short term average / STA），我們可以過濾一些非地震的震動，以避免因非地震的震動激活地震警報。

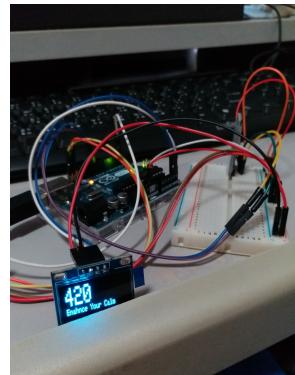
（有關短期平均值如何過濾假地震請參閱：09.01 分辨真假地震及地震實體波種類）

04 所使用的元件

要儲存、處理分析地震測量的數據，我們需要使用卡片式電腦以及其他元件來測量、運算以及分析數據。當中我們使用了 Arduino UNO R3 原廠開發板、MPU6050 加速度計、SSD1306 OLED 顯示屏、HC-05 藍牙傳輸元件以及蜂鳴器等。以下為元件使用的簡述：

→ Arduino UNO R3 原廠開發板

利用 Arduino UNO R3 原廠開發板，我們可以自行編程，達至地震測量、紀錄數據等功能。它連接著 MPU6050 加速度計、SSD1306 OLED 顯示屏、HC-05 藍牙傳輸元件以及蜂鳴器等。



→ MPU6050 加速度計

MPU6050 加速度計讓我們可以量度地震對加速度計數值的改變，從而推斷出地震的震幅和地震波的類型，並透過簡單的 Arduino 邏輯編程來分辨地震的真實性。

→ SSD1306 OLED 顯示屏

使用 SSD1306 OLED 顯示屏，我們可以更清晰地顯示我們的震幅和地震波的類型。加上 OLED 顯示屏比其他的顯示屏更省電，從而減少地震探測器的用電量。

圖 04.OLED ↑ SSD1306 OLED 顯示屏正在顯示「420 Enhance Your Calm」字樣

→ HC-05 藍牙傳輸元件

透過使用 HC-05 藍牙傳輸元件，地震探測器能將數據傳送到用家的智能裝置，令用家可以遠距閱覽實時地震探測數據。

→ LTE12-05 蜂鳴器

當地震探測器探測到有地震時，探測器上的蜂鳴器便會發出聲響，提醒附近的人民知道地震的發生並且呼籲緊急撤離。

04.01 不使用智能手機加速度計原因

由於我們的地震探測器需要長時間運行，如果我們智能手機內置的加速度計，智能手機的用電量會提高。由於我們有些應用程式會經常使用，導致龐大的用電量，當加上全時間的加速度計的數據紀錄和處理，便會提升智能手機的用電量，用家需要經常為智能手機充電以應付日常的需要。除此之外，由於我們每天有不同的運動，如行走，跑步。這些活動會被智能手機的加速度計紀錄，干擾了加速度計紀錄地震的功能，大大影響地震探測器的準確性。相反而言，如果使用獨立功能的地震探測器，裝置只會量度地下的震動，可以減少外來的干擾。因此，獨立的地震探測器比智能手機更有效量度和紀錄地震。

05 軟件設計

為方便使用者閱讀地震探測器讀取的數據，我們編寫了Android 應用程式、以及網頁，讓使用者可在較遠的現場、甚至其他地方閱讀數據以及圖表。

值得一提的事，透過不同方向（x - 軸、y - 軸、z - 軸）的加速度，我們可以分辨地震實體波是屬於 P 波或是 S 波。因此我們在編程時為地震探測器加入分辨實體波類型的邏輯，並在地震時提供相關資訊。

05.01 Android 應用程式

Android 應用程式介面清晰明瞭、簡單易用，是讀者能一眼看得出地震資料。透過藍牙連接地震探測器後，應用程式會顯示 x - 軸和 y - 軸（水平）的合成加速度（英語：Resultant Acceleration）以及 z - 軸（垂直）的加速度。並在探測到有地震時顯示地震實體波的類型（即 P 波和 S 波）以及發出警報。提升使用者在地震發生時要注意人身及財物安全。

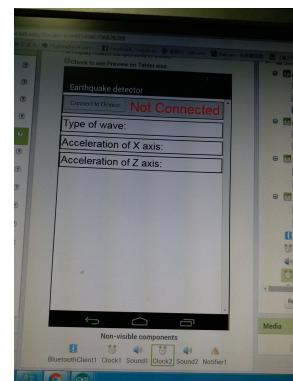


圖 05.01a ↑ 正在編寫
Android 應用程式

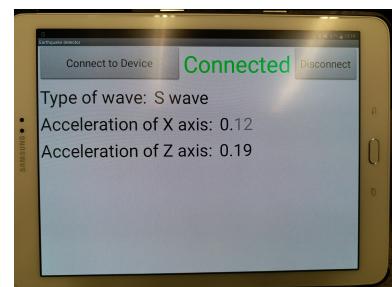


圖 05.01b ↑ Android 應用程式

05.02 網頁

我們的網頁會有三個不同的功能。

1. 顯示實時數據如地震波的類型和震幅，使用家能在遠距離知悉裝置的所在地有沒有地震。
2. 將所收集的數據轉換成圖表，讓用家能透過圖表知悉地震的狀況。
3. 儲存所收集到的數據並顯示他們，從而讓用家使用過去的數據進行分析並採取適當的措施。

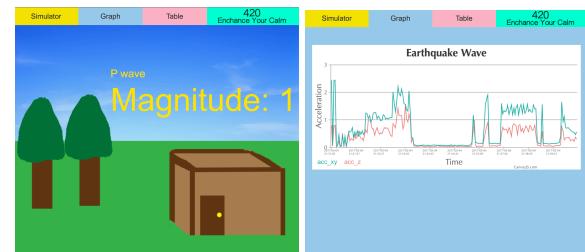


圖 05.02 ↑ 網頁截圖

使用者可以透過網頁，在遠距離閱覽地震探測器所探測的數據。並提供原始數據以及地震波圖表的即時顯示。

06 硬件設計

我們設計的地震探測器則是輕巧、靈活。方便戶外測量、靈活更換測量位置。

利用鐳射切割技術，我們可以將透明膠版切割成適合地震探測器的大小、切割出連接線和儲存裝置的插孔（例如：電源線、SD 卡卡槽）、甚至在地震探測器表面刻上字樣。

所使用的元件價錢合理，而且功能甚多、外觀簡潔，適合長遠的大量生產，惠及有需要的人，例如須實地考察的地理學生等等。

07 操作方法

1. 檢查 Arduino UNO R3 是否已接駁 USB 電源。
2. 初始化地震探測器。

注意：初始化地震探測器時地震探測器會自動校準，因此接駁電源後須靜待，直至顯示屏顯示「Done」字樣。

3. 使用流動裝置（例如：智能手機、平板電腦等）透過藍牙連接地震探測器，以讀取其地震數據。如遇上地震，裝置的屏幕亦會顯示地震的種類（P 波 / S 波）。

08 應用範疇

地震探測器主要功能為實時探測地震強度，以及即時繪製圖表。其設計輕巧、便攜性、以及分辨真假地震的功能，適合作靈活的戶外地震測量。

這地震探測器更可以在香港使用，雖然香港甚少發生地震，但有時仍會受鄰近地區（例如中國大陸、南亞等地區）的地震波及。所以我們可以使用地震探測器蒐集有關數據。由於現時本港蒐集地震的地震較為少，所以我們的裝置可以在多個地方加設，使地震的數據有所增多。由於地震的數據可以被紀錄和分析，可以幫助未來研究地殼板塊移動。所以在港使用地震探測器有一定的重要性。

09 測試

為確保地震探測器能夠準確地分辨震動是否由地震引起、以及測量地震強度，我們需要在事前為地震探測器進行測試。以下為測試的方法、過程、以及其結果。

我們 420 Enhance Your Calm 團隊設計的地震探測器，設有分辨真假地震的功能。因此我們的測試方法分兩個階段。第一階段為測試分辨真假地震的功能，第二階段測試地震探測器在震動台上不同強度的表現。測試方法為以強度幅度、震動已擺放地震探測器的震動台，然後分析結果。

09.01 分辨真假地震及地震實體波種類

地震探測器必須能夠分辨真假地震。真正的地震是持續性的，而人為震動（如重型車輛經過）或惡意觸碰通常只會維持短時間或只有數下。如果直接以加速度大過某個數值去激活警報的話，必然有很多誤報。

因此，我們的探測器能計算過去 10 個加速度的短期平均值（即過去半秒的地震強度數據），並測試出一個激活警報的數值，便能有效減少因非地震引起的震動而意外激活警報的機會。

在測試時，我們除了以手動的方法去模擬短期或人工的震動，亦建造了震動台去製造持續的地震。結果顯示我們的探測器能探測到震動是否持續，並分辨該震動是地震還是惡意觸碰。

另外，我們亦嘗試上下震動或左右震動探測器。結果顯示透過不同方向（x - 軸、y - 軸、z - 軸）的加速度，我們的探測器能分辨地震實體波是屬於 P 波或是 S 波。

09.02 分辨地震強度

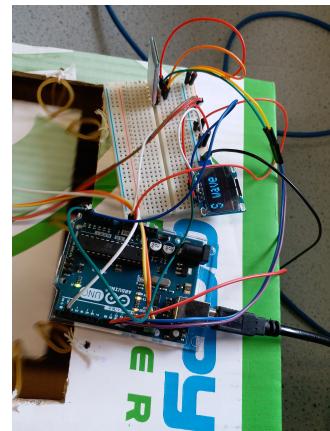


圖 09.01
←
SSD1306 OLED
顯示屏正在顯示地震實體波的類型

利用加速度計所量度的加速度大小，我們可以推斷地震的強度，因此利用電位器（或稱可變電阻，英語：Potentiometer / Variable Resistor）以及摩打等元件製作了一個震動台（英語：Shake table），並用它來測試地震探測器。

震動台能夠控制震動幅度，從而令探測器能夠有穩定的震動來源。我們把地震探測器固定在震動台上，並連接上電源。震動台右面能夠調整震幅大小。因此我們在測試時能夠獲得準確的數據，從而探測地震的幅度。

在測試過程中，我們將震動台的震幅慢慢調低，而地震探測器得出的圖表（圖 09.02）也有明顯的變化反應震幅變小。表示我們的地震探測器有辨別地震強度的功能。

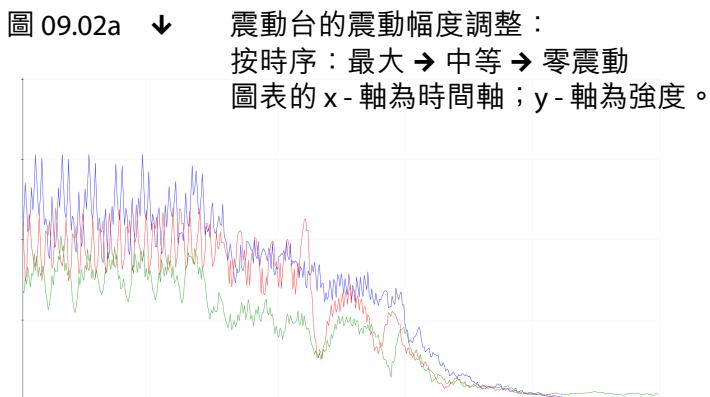


圖 09.02b ↑ 震動台

10 其他資料

地震探測器：	長	10.6 釐米	震動台：	長	31.0 釐米（約 A4 長度）
	闊	8.6 釐米		闊	22.0 釐米（約 A4 間度）
	高	4.6 釐米		高	13.0 釐米
	重量	296 公克		重量	773 公克
地震探測器名稱：	使用電壓	5.00 伏特		使用電壓	5.00 伏特
		420 - 17 Forbidden			

11 反思

參加是次地震探測器設計比賽，學懂了不少有關地震的知識之外，還在設計地震探測器時學懂與人相處、互相妥協、以及團隊合作。以下為我們可改善之處以及達成的成就。

11.01 可改善之處

這次我們最缺乏的就是時間分配和工作分配，作為學生的我們，需要兼顧不同方面的發展，有時會拖慢到研發的進程，令隊友們等待，令到進程變得緩慢，製作途中浪費了很多時間，而且我們起初是沒有對地震探測器的研發一個好好的規劃，令隊友不知道自己要做什麼。

11.02 成就達成

但最後很感恩的是我們各盡其力，對不同的部門作出幫助，還不分你我去完成這個地震探測器，大家都能從中了解到編程，電路和設計的知識，還了解到團隊合作的重要性，完成一個地震探測器不能夠只靠一個人，而是互相依靠，一同去付出的成果，所以最後大家都能完成了這個地震探測器。

12 總結

透過這次比賽，我們希望能夠把地震的知識普及給身邊的人，地震其實離我們並不是很遠，在大地震中我們常常感受到餘震。但香港只有天文台才有地震探測器。一般市民對地震的認知並不多，甚至覺得地震和自己毫不相關。但地震其實距離我們很近。我們希望在這個比賽中，普及人們對地震的認知和提高人們的警覺。

而在製造地震探測器中，我們不停地研究如何透過程式來顯示數據。科技發展迅速，我們希望我們的經歷能幫助 STEM 教育的發展，並為自己的學弟學妹提供一個 STEM 教育的範例。

我們更想藉着這次的比賽去宣揚科技對人類的生活的重要性，我們相信如果世上沒有了科技，很多事情會變得很麻煩，嚴重的話甚至會影響到人類的存亡，所以我們最後希望這個裝置能幫助到有需要的人，去逃過死神的威脅。

13 鳴謝

1. 主辦單位
2. 柳已丞老師
3. 香港神託會培基書院

14 附錄

14.01 分工

麥澤榮：報告編寫、外觀設計
吳灝民：報告編寫、數據演示、製作震動台
王俊禧：編寫程式、整合電路
黃達濠：編寫程式、為地震探測器進行測試

14.02 參考資料

OLED I2c Display With Arduino, by Jean0x7BE in arduino.

<http://www.instructables.com/id/Monochrome-096-i2c-OLED-display-with-arduino-SSD13/>

mp3無損解碼板 TF卡 U盤 MP3解碼播放器模塊 自帶功放(H6B4) , 淘寶網。

[https://item.taobao.com/item.htm?id=521749240901&ali_refid=a3_430582_1006:1103831585:N: %E8%93%9D%E7%89%99mp3%E8%A7%A3%E7%A0%81%E6%9D%BF: 725f9b307476aaf2471c125f7e4769f4&ali_trackid=1_725f9b307476aaf2471c125f7e4769f4&spm=a230r. 1.14.1.jCkAZQ#detail](https://item.taobao.com/item.htm?id=521749240901&ali_refid=a3_430582_1006:1103831585:N: %E8%93%9D%E7%89%99mp3%E8%A7%A3%E7%A0%81%E6%9D%BF: 725f9b307476aaf2471c125f7e4769f4&ali_trackid=1_725f9b307476aaf2471c125f7e4769f4&spm=a230r. 1.14.1.jCkAZQ&ali_refid=a3_430582_1006:1103831585:N: %E8%93%9D%E7%89%99mp3%E8%A7%A3%E7%A0%81%E6%9D%BF: 725f9b307476aaf2471c125f7e4769f4&ali_trackid=1_725f9b307476aaf2471c125f7e4769f4&spm=a230r. 1.14.1.jCkAZQ#detail)

Mp3 Play From SD Card With Arduino, by sezgingul in audio.

<http://www.instructables.com/id/Audio-Playback-From-SD-Card-With-Arduino/>

Using the SD library to log data, Arduino.

<https://www.arduino.cc/en/Tutorial/Datalogger>

How to Use a Buzzer (or Piezo Speaker) - Arduino Tutorial, by codebender_cc in arduino.

<http://www.instructables.com/id/How-to-use-a-Buzzer-Arduino-Tutorial/>

Controlling AC Light Using Arduino With Relay Module, by ostin jos in arduino.

<http://www.instructables.com/id/Controlling-AC-light-using-Arduino-with-relay-modu/>

MyShake: A smartphone seismic network for earthquake early warning and beyond, Qingkai Kong, Richard M. Allen, Louis Schreier, Young-Woo Kwon.

<http://advances.sciencemag.org/content/2/2/e1501055/tab-pdf>

STA/LTA trigger Algorithm

ftp://hazards.cr.usgs.gov/Eq_Effects/GeekPack/Procedures-Configs-Info/1_Dataloggers/K2-Altus/Sta-Lta.pdf

14.03 電路圖

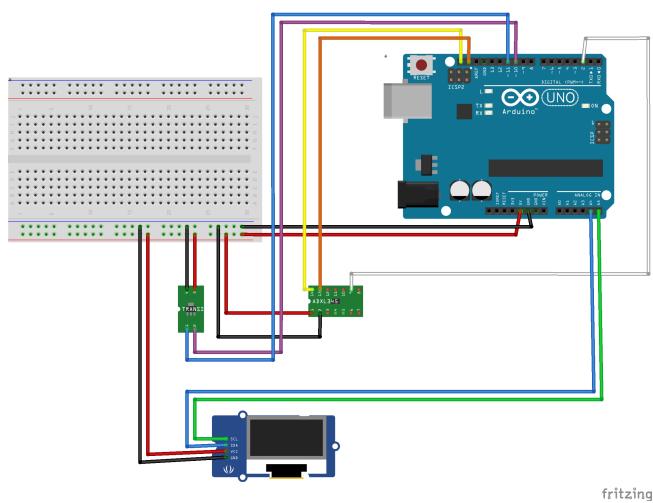


圖 14.03a ↑ 地震探測器的電路圖

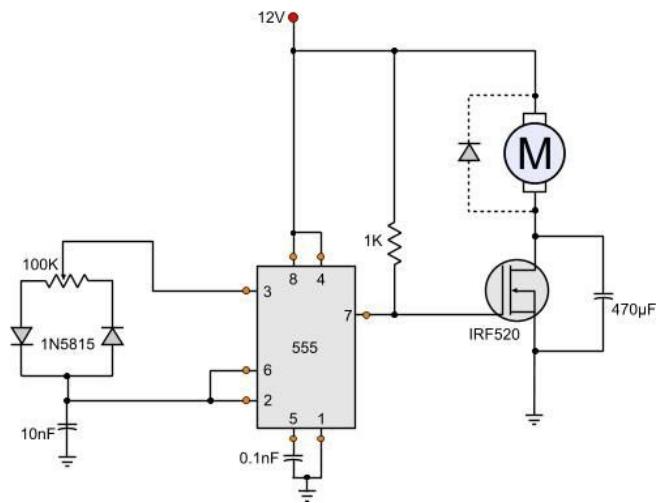


圖 14.03b ↑ 震動台的電路圖 (網上圖片)

14.04 Arduino 原始碼

↓ 地震探測器的 Arduino 原始碼

```
1) // I2Cdev and MPU6050 must be installed as libraries,
2) // or else the .cpp/.h files
3) // for both classes must be in the include path of
4) // your project
5) #include "I2Cdev.h"
6) #include "MPU6050.h"
7) #include <SPI.h>
8) #include <Adafruit_GFX.h>
9) #include <Adafruit_SSD1306.h>
10) #include "Wire.h"
11) #include <SoftwareSerial.h>
12) #include "pitches.h"

13) int m_state=0;      // 0 for no earthquake; 1 for
14) light earthquake; 2 for moderate earthquake; 3 for
15) large earthquakes;
16) float xy_raw=0;           // raw value
17) of P wave
18) float xy=0;             //acceleration
19) value of P wave
20) float BTtime=0;

21) /*          BT          */
22) SoftwareSerial BTserial(10, 11); // RX, TX
23) /*          BT          */
24)
25) /*          Accelerometer      */
26)
27) // class default I2C address is 0x68
28) // specific I2C addresses may be passed as a
29) parameter here
30) // AD0 low = 0x68 (default for InvenSense evaluation
31) // board)
32) // AD0 high = 0x69
33) MPU6050 accelgyro;
34) //MPU6050 accelgyro(0x69); // <-- use for AD0 high
35)
36) int16_t ax, ay, az;
37) int16_t gx, gy, gz;
38) // int16_t Tmp;
39)
40)
41) // uncomment "OUTPUT_READABLE_ACCELGYRO" if you want
42) to see a tab-separated
43) // list of the accel X/Y/Z and then gyro X/Y/Z values
44) // in decimal. Easy to read,
45) // not so easy to parse, and slow(er) over UART.
46) #define OUTPUT_READABLE_ACCELGYRO
47)
48) bool blinkState = false;
49)
50) float total_x = 0; /* calibrating */
51) float calx = 0; /* calibrating */
52) float acc_x = 0; /* calibrating */
53)
54) float total_y = 0; /* calibrating */
55) float caly = 0; /* calibrating */
56) float acc_y = 0; /* calibrating */
57)
58) float total_z = 0; /* calibrating */
59) float calz = 0; /* calibrating */
60) float acc_z = 0; /* calibrating */
61)
62) float i = 0; /* calibrating */
63)
64)
65)
66) /* running average */
67) const int numReadings=10;
68) int readindex = 0;
69) float readings_x[numReadings];
70) float totalx = 0;
71) float average_x = 0;
72) float readings_y[numReadings];
73) float totaly = 0;
74) float average_y = 0;
75) float readings_z[numReadings];
76) float totalz = 0;
77) float average_z = 0;
78) float ini_x;
79) float ini_y;
80) float ini_z;
81) int n = 0;
82) /* running average */
83)
84) /*          OLED          */
85) #define OLED_RESET 4
86) Adafruit_SSD1306 display(OLED_RESET);
87)
88) #define NUMFLAKES 10
89) #define XPOS 0
90) #define YPOS 1
91) #define DELTAY 2
92)
93)
94) #define LOGO16_GLCD_HEIGHT 16
95) #define LOGO16_GLCD_WIDTH 16
96)
97)
98) #if (SSD1306_LCDHEIGHT != 32)
99) #error("Height incorrect, please fix
Adafruit_SSD1306.h!");
100) #endif
101) /*          OLED          */
102)
103)
104) void setup() {
105) // put your setup code here, to run once:
106) Serial.begin(9600);
107) BTserial.begin(9600);
108) pinMode(9, OUTPUT); //buzzer
109)
110)
111) /*          OLED          */
112) // by default, we'll generate the high voltage from
113) the 3.3v line internally! (neat!)
114) display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C (for the 128x32)
115) // init done
116) // Clear the buffer.
117) display.clearDisplay();
118)
119) // show logo
display.setTextSize(3);
```

```

120) display.setTextColor(WHITE);
121) display.setCursor(0,0);
122) display.println("420");
123) display.setTextSize(1);
124) display.setTextColor(WHITE);
125) display.println("Enhance Your Calm");
126) display.display();
127) delay(2000);
128) display.clearDisplay();

129)
130) /*          OLED          */
131)
132) Wire.begin();
133) // initialize device
134) Serial.println("Initializing I2C devices...");
135) display.setTextSize(1);
136) display.setTextColor(WHITE);
137) display.setCursor(0,0);
138) display.println("Initializing");
139) display.display();

140) /*          Accelerometer      */
141) accelgyro.initialize();
142) // Set up offsets, first calibration (by
143) MPU6050_calibration)
144) // 49 502 2077 76 -53 29
145) accelgyro.setXAccelOffset(49);
146) accelgyro.setYAccelOffset(502);
147) accelgyro.setZAccelOffset(2077);
148) accelgyro.setXGyroOffset(76);
149) accelgyro.setYGyroOffset(-53);
150) accelgyro.setZGyroOffset(29);

151)
152) // verify connection
153) Serial.println("Testing device connections...");
154) Serial.println(accelgyro.testConnection() ?
155) "MPU6050 connection successful" : "MPU6050 connection
failed");
156) Serial.println("Calibrating..");
157) display.println("Calibrating");
158) display.display();

159)
160)     for (i=0 ; i<200 ; i++) {
161) // second calibration
162)         accelgyro.getAcceleration(&ax, &ay, &az);
163) //read data
164)         total_x = total_x + ax ;
165)         total_y = total_y + ay ;
166)         total_z = total_z + az ;
167)         delay(50);
168)         // // Serial.print("the value is ");
169)         Serial.println(ax);
170)         calx = 16384-total_x/i; //take average and
let it be the value of 16
171)         caly = 16384-total_y/i; //take average and
let it be the value of 16
172)         calz = 16384-total_z/i; //take average and
let it be the value of 16
173)         //Serial.println(calx);
174)         //Serial.println(calz);
175)         delay(1000);
176)         Serial.println("Done!");
177)         display.println("Done!");
178)         display.display();
179)         delay(1000);
180)         display.clearDisplay();
181)         /*          Accelerometer      */
182)         /*
183)             for (int readindex = 0; readindex<numReadings;
readindex++){ //reset value
184)                 readings_x[readindex] = 0;
185)                 readings_y[readindex] = 0;
186)                 readings_z[readindex] = 0;
187)             }
188)             /* running average */
189)             ini_x = ax;
190)             ini_y = ay;
191)             ini_z = az;
192)             Serial.println(ini_x);
193)             Serial.println(ini_y);
194)             Serial.println(ini_z);
195)         }

196) void loop() {
197) // put your main code here, to run repeatedly:
198)
199) /*          Accelerometer      */
200) // read raw accel/gyro measurements from device
201) accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy,
&gz);
202) // Tmp=Wire.read()<<8|Wire.read(); // 0x41
(TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

203) // these methods (and a few others) are also
available
204) //accelgyro.getAcceleration(&ax, &ay, &az);
205) //accelgyro.getRotation(&gx, &gy, &gz);

206) #ifdef OUTPUT_READABLE_ACCELGYRO
207) // display tab-separated accel/gyro x/y/z
values
208) // Serial.print("a/g:\t");
209) // Serial.print(ax); // Serial.print("\t");
210) // Serial.print(ay); // Serial.print("\t");
211) // Serial.print(az+calz); //
212) Serial.print("\t");
213) // Serial.print(gx); // Serial.print("\t");
214) // Serial.print(gy); // Serial.print("\t");
215) // Serial.println(gz);
216) #endif

217)
218) #ifdef OUTPUT_BINARY_ACCELGYRO
219)     Serial.write((uint8_t)(ax >> 8));
220)     Serial.write((uint8_t)(ax & 0xFF));
221)     Serial.write((uint8_t)(ay >> 8));
222)     Serial.write((uint8_t)(ay & 0xFF));
223)     Serial.write((uint8_t)(az >> 8));
224)     Serial.write((uint8_t)(az & 0xFF));
225)     Serial.write((uint8_t)(gx >> 8));
226)     Serial.write((uint8_t)(gx & 0xFF));
227)     Serial.write((uint8_t)(gy >> 8));
228)     Serial.write((uint8_t)(gy & 0xFF));
229)     Serial.write((uint8_t)(gz >> 8));
230)     Serial.write((uint8_t)(gz & 0xFF));
231) #endif

232) /*
233)     running_average();
234)     OLED_Display();
235)     if (millis()-BTtime>=1000) {
236)         BT_Display();
237)         BTtime = millis();
238)     }
239) // delay(100);
240)
241) 
```

```

242)
243)
244) void running_average(void) {
245)
246)     totalx=totalx-readings_x[readindex]; //clear old
247)     values
248)     totaly=totaly-readings_y[readindex]; //clear old
249)     values
250)     totalz=totalz-readings_z[readindex]; //clear old
251)     values
252)     readings_x[readindex]=abs(ax - ini_x); //take absolute value
253)     readings_y[readindex]=abs/ay - ini_y); //take absolute value
254)     readings_z[readindex]=abs/az - ini_z); //take absolute value
255)     totalx=totalx+readings_x[readindex]; //add new values
256)     totaly=totaly+readings_y[readindex]; //add new values
257)     totalz=totalz+readings_z[readindex]; //add new values
258)     readindex++;
259)     if (readindex >= numReadings) readindex=0; //doing averages
260)     average_x=totalx/numReadings;
261)     average_y=totaly/numReadings;
262)     average_z=totalz/numReadings;
263)     if (average_x <= 0){ //omit -ve values
264)         average_x = 0;
265)     }
266)     if (average_y <= 0){ //omit -ve values
267)         average_y = 0;
268)     }
269)     acc_x = (average_x)/16384*9.8066 ; //raw to acceleration
270)     acc_y = (average_y)/16384*9.8066 ; //raw to acceleration
271)     acc_z = (average_z)/16384*9.8066 ; //raw to acceleration
272)     Serial.print("Raw value:");Serial.print("\t");
273)     Serial.print(average_x); Serial.print("\t");
274)     Serial.print(average_y); Serial.print("\t");
275)     Serial.println(average_z);
276)     Serial.print("Acceleration:"); Serial.print("\t");
277)     Serial.print(acc_x); Serial.print("\t");
278)     Serial.print(acc_y);
279)     Serial.print("\t");Serial.println(acc_z);
280)     xy_raw= sqrt((average_x*average_x) +
281)     (average_y*average_y));
282)     xy= sqrt((acc_x*acc_x) + (acc_y*acc_y));
283)     Serial.print("xy= "); Serial.print("\t");
284)     Serial.println(xy);
285)     Serial.print("xy_raw= "); Serial.print("\t");
286)     Serial.println(xy_raw);
287)     /* M1 */
288)
289) }
290) else{
291)     noTone(9); //buzzer stop
292)     digitalWrite(9, LOW); //buzzer stop
293)     m_state=0;
294) }
295) }
296) else if ((xy_raw >=1000 && xy_raw <2000) ||
297) (average_z >=800 && average_z <1300)) {
298)     n=n+1 ;
299)     if (n>20){
300)         Serial.println("warning, magnitude 2");
301)         tone(9, NOTE_G3); //buzzer play
302)         digitalWrite(9, HIGH); // buzzer play
303)         m_state=2;
304)     }
305)     else{
306)         noTone(9); //buzzer stop
307)         digitalWrite(9, LOW); //buzzer stop
308)         m_state=0;
309)     }
310) }
311) else if (xy_raw >=2000 or average_z >=1500 ) {
312)     n=n+1 ;
313)     if (n>20){
314)         Serial.println("warning, magnitude 3");
315)         tone(9, NOTE_G3); //buzzer play
316)         digitalWrite(9, HIGH); // buzzer play
317)         m_state=3;
318)     }
319)     else{
320)         noTone(9); //buzzer stop
321)         digitalWrite(9, LOW); //buzzer stop
322)         m_state=0;
323)     }
324)     else{
325)         n=0 ;
326)         m_state=0;
327)     }
328) /* M3 */
329) }
330)
331)
332) }
333)
334) void OLED_Display(void) {
335)     // OLED print type of wave
336)     // xy= sqrt((acc_x*acc_x) + (acc_y*acc_y));
337)     /* M1 */
338)     if (m_state==1){
339)         if (xy > acc_z) {
340)             display.clearDisplay();
341)             display.setTextSize(2);
342)             display.setTextColor(WHITE);
343)             display.setCursor(0,0);
344)             display.println("P wave");
345)             display.setTextSize(2);
346)             display.println("M1");
347)             display.display();
348)     }
349)     else {
350)         display.clearDisplay();
351)         display.setTextSize(2);
352)         display.setTextColor(WHITE);
353)         display.setCursor(0,0);
354)         display.println("S wave");
355)         display.setTextSize(2);
356)         display.println("M1");
357)         display.display();
358)     }

```

```

359)
360)     }
361) }
362) /* M1 */
363) /* M2 */
364) if (m_state==2){
365)     if (xy > acc_z) {
366)         display.clearDisplay();
367)         display.setTextSize(2);
368)         display.setTextColor(WHITE);
369)         display.setCursor(0,0);
370)         display.println("P wave");
371)         display.setTextSize(2);
372)         display.println("M2");
373)         display.display();
374)
375)     }
376) else {
377)     display.clearDisplay();
378)     display.setTextSize(2);
379)     display.setTextColor(WHITE);
380)     display.setCursor(0,0);
381)     display.println("S wave");
382)     display.setTextSize(2);
383)     display.println("M2");
384)     display.display();
385)
386) }
387) }
388) /* M2 */
389) /* M3 */
390) if (m_state==3){
391)     if (xy > acc_z) {
392)         display.clearDisplay();
393)         display.setTextSize(2);
394)         display.setTextColor(WHITE);
395)         display.setCursor(0,0);
396)         display.println("P wave");
397)         display.setTextSize(2);
398)         display.println("M3");
399)         display.display();
400)
401) }
402) else {
403)     display.clearDisplay();
404)     display.setTextSize(2);
405)     display.setTextColor(WHITE);
406)     display.setCursor(0,0);
407)     display.println("S wave");
408)     display.setTextSize(2);
409)     display.println("M3");
410)     display.display();
411)
412) }
413) }
414) /* M3 */
415) else if (n<20) {
416)     display.clearDisplay();
417)     display.setTextSize(3);
418)     display.setTextColor(WHITE);
419)     display.setCursor(0,0);
420)     display.println("Stable");
421)     display.display();
422)     m_state=0;
423) }
424)
425) }
426)
427)
428) void BT_Display(void) {
429)     // OLED print type of wave
430)     // xy= sqrt((acc_x*acc_x) + (acc_y*acc_y));
431)     if(m_state==0) {
432)         BTserial.print("stable");
433)         BTserial.print("|");
434)         BTserial.print(xy);
435)         BTserial.print("|");
436)         BTserial.print(acc_z);
437)         BTserial.print("|");
438)         BTserial.print(m_state);
439)         BTserial.print("|");
440)     }
441)     else if(m_state!=0) {
442)         if (xy > acc_z) {
443)             BTserial.print("P");
444)             BTserial.print("|");
445)             BTserial.print(xy);
446)             BTserial.print("|");
447)             BTserial.print(acc_z);
448)             BTserial.print("|");
449)             BTserial.print(m_state);
450)             BTserial.print("|");
451)         }
452)         else {
453)             BTserial.print("S");
454)             BTserial.print("|");
455)             BTserial.print(xy);
456)             BTserial.print("|");
457)             BTserial.print(acc_z);
458)             BTserial.print("|");
459)             BTserial.print(m_state);
460)             BTserial.print("|");
461)         }
462)     }
463) }

```

▼ 震動台的 Arduino 原始碼

```

1) byte data;
2) void setup() {
3) }
4) void loop () {
5) data = analogRead(0) / 4;
6) analogWrite(9, data);
7) delay(5);
8) }

```