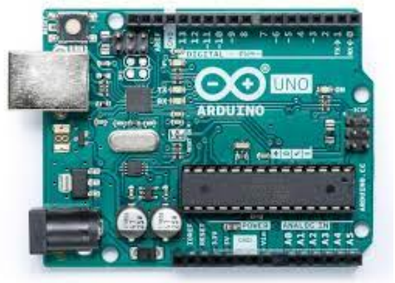
盲人小助手

姓名：林宥任
學號：B0829025
指導教授：吳世琳教授

架構



當小於20cm
則響起

當加速度大於15時

ThingSpeak

LINE Notify

流程

當使用者前方有物體時
(目前是設定為20cm以內)

物體在膝蓋處

物體在腰處

當使用者劇烈碰撞或摔倒時 → 加速度>15 → 傳送緊急訊息給使用者家屬

LINE Notify

程式碼

project

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_MPU6050.h>
#include <Process.h>
#include <Bridge.h>

//設定閾值，當加速度大於這個值時，傳送數據到ThingSpeak
#define THRESHOLD 15
String APIKey = "JXQW0PILW72EQ8IP";
Adafruit_MPU6050 mpu;
unsigned long timer = 0;

void setup() {
  //初始化串口、I2C通訊、Bridge橋
  Serial.begin(9600);
  Wire.begin();
  Bridge.begin();

  //初始化MPU-6050
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  //計算總加速度
  float totalAcceleration = sqrt(a.acceleration.x * a.acceleration.x +
                                 a.acceleration.y * a.acceleration.y +
                                 a.acceleration.z * a.acceleration.z);
  Serial.println(totalAcceleration);
  //如果總加速度超過閾值，發送數據到ThingSpeak
  if (totalAcceleration > THRESHOLD && millis() - timer > 10000) {
    String url = "https://api.thingspeak.com/update";    // form the string for the URL parameter

    // Send the HTTP POST request
    Process p;
    Serial.print("\n\nSending data... ");

    String cmd = "curl --silent --request POST --header \"X-THINGSPEAKAPIKEY: " +APIKey;
    cmd = cmd + "\" --data \"field1=" + totalAcceleration;
    cmd = cmd + "\" http://api.thingspeak.com/update";
    p.runShellCommand(cmd);
    Console.println(cmd);
    p.close();

    Serial.println(cmd);
    Serial.println("done!");

    // If there's incoming data from the net connection, send it out the Serial:
    while (p.available()>0) {
      char c = p.read();
      Serial.write(c);
    }
  }

  delay(100);
}
```

project2

```
#define trigPin 9 // 超聲波傳感器 trig pin
#define echoPin 10 // 超聲波傳感器 echo pin
#define BeeperPin 13 // LED 腳位
#define Do 523
#define Re 587
#define Mi 659
#define Fa 698
#define So 784
#define La 880
#define Si 988


int melody[7] = {Do, Re, Mi, Fa, So, La, Si};

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(BeeperPin, OUTPUT);
  Serial.begin(9600); // 建立 Serial 連接，可以觀察超聲波傳感器返回的距離數據
}

void loop() {
  digitalWrite(trigPin, LOW); // 先發送一個 LOW 訊號
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH); // 發送一個 10 毫秒的高訊號
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH); // 讀取回傳的脈衝寬度
  int distance = duration * 0.034 / 2; // 計算距離，公式為距離 = 脈衝寬度 * 速度 / 2，其中速度為聲速（約為 34 cm/ms）

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance < 20) { // 當距離小於 10 cm 時亮起 LED
    for (int i = 0;i < 8; i++){
      tone(BeeperPin, melody[i]);
      delay(250);
    }
  } else { // 否則熄滅 LED
    noTone(BeeperPin);
    delay(2000);
  }

  delay(500); // 延遲 500 毫秒，避免短時間內大量讀取超聲波傳感器數據
}
```

project3

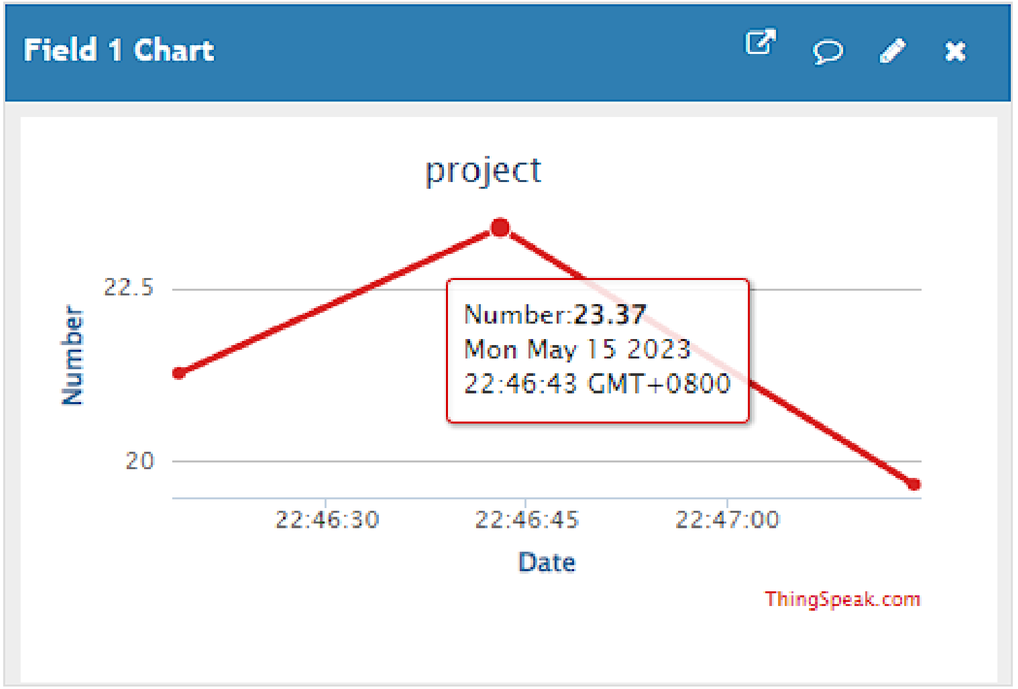
```
#define trigPin 9 // 超聲波傳感器 trig pin
#define echoPin 10 // 超聲波傳感器 echo pin
#define BeeperPin 13 // LED 腳位


void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(BeeperPin, OUTPUT);
  Serial.begin(9600); // 建立 Serial 連接，可以觀察超聲波傳感器返回的距離數據
}

void loop() {
  digitalWrite(trigPin, LOW); // 先發送一個 LOW 訊號
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH); // 發送一個 10 毫秒的高訊號
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH); // 讀取回傳的脈衝寬度
  int distance = duration * 0.034 / 2; // 計算距離，公式為距離 = 脈衝寬度 * 速度 / 2，其中速度為聲速（約為 34 cm/ms）

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance < 20) { // 當距離小於 10 cm 時亮起 LED
    tone(BeeperPin, 988);
    delay(250);
  } else { // 否則熄滅 LED
    noTone(BeeperPin);
  }

  delay(500); // 延遲 500 毫秒，避免短時間內大量讀取超聲波傳感器數據
}
```

sendmes.py

C: > Users > user > OneDrive > 桌面 > project > sendmes.py

```
import requests
import json
import time

# 設定閾值，當加速度大於這個值時，發送Line Notify訊息
THRESHOLD = 15

# 設定ThingSpeak和Line Notify的API Key
THINGSPEAK_API_KEY = "ZEQ6YIPZXVYPE264"
LINE_NOTIFY_TOKEN = "CidxRRiUl0YBVySIJxATudfu263dGCadc6M8LbCarsZ"

# 設定Line Notify發送的訊息格式
LINE_MESSAGE = "發生劇烈碰撞，使用者可能有危險"

def main():
    last_updated = 0
    while True:
        # 從ThingSpeak讀取最新的數據
        url = f"https://api.thingspeak.com/channels/2146591/fields/1.json?api_key={THINGSPEAK_API_KEY}"
        response = requests.get(url)
        data = json.loads(response.text)
        acceleration = float(data["feeds"][-1]["field1"])
        updated_at = data["feeds"][-1]["created_at"]
        updated_timestamp = time.mktime(time.strptime(updated_at, "%Y-%m-%dT%H:%M:%SZ"))

        # 如果數據大於閾值，並且更新時間大於上一次更新時間，則發送Line Notify訊息
        if acceleration > THRESHOLD and updated_timestamp > last_updated:
            message = LINE_MESSAGE.format(acceleration)
            payload = {"message": message}
            headers = {"Authorization": f"Bearer {LINE_NOTIFY_TOKEN}"}
            response = requests.post("https://notify-api.line.me/api/notify", data=payload, headers=headers)
            last_updated = updated_timestamp
            print("Line Notify訊息已發送")

        time.sleep(10)

if __name__ == "__main__":
    main()
```

DEMO



【盲人小助手】發生劇烈碰撞，使用者可能有危險

下午 6:33

THANK YOU