

Contents

[The design of the hardware - 歐育成、楊梓暄](#_3fsigiz5l4ah)

[Component:](#_ymxogae094im)

[Board circuit welding:](#_rtc4md65vtil)

[Battery holder inside of wood:](#_hy9zkf5p7hbj)

[預計內部配置：](#_q02msc4vx3k7)

[球拍手柄加工：](#_dvn1z5ly2iv0)

[Combining all components:](#_pcrc8hc3yig4)

[Arduino code battery charging related mentions:](#_5cgff2ej96ni)

[The implementation of data communication and database - 許少麒](#_ludszp9djva6)

[Arduino IDE 開發環境與程式設定說明](#_upqxj8z2selt)

[開發環境與硬體](#_lfopzoncqbkl)

[IMU參數設定](#_fz3y2hghaskp)

[資料收集資訊](#_voq6preebbii)

[BLE設定](#_l3aybsneyavb)

[電池設定](#_ao4beq3bq6rq)

[The design of UI (APP) - 李東霖、許少麒](#_yrj7qkn2mv1f)

[Development environment](#_cfqq0mqp925n)

[Flutter pub packages](#_61b89ekybnnh)

[Flutter project 注意事項](#_rcviz6ky9gaa)

[firebase.json, google-services.json](#_vcxc7ti2f1iq)

[assets / badminton\_model.tflite](#_qshok2h4jkeg)

[android / settings.gradle.kts](#_s1lxkvfx0h7n)

[BLE 掃瞄及連線 - 許少麒](#_mkw5ybwgam5d)

[傳送資料到雲端 - 許少麒](#_13qg39al19m)

[即時偵測 Sensor 資訊，以 Line Chart 顯示 - 李東霖](#_m2964srg0y6c)

[模型嵌入 App & 感測資料處理與模型判斷應用 - 李東霖、許少麒](#_uide47ltcnaa)

[Data Collection & Labeling - 楊梓暄](#_k8722elo47gd)

[The design of classification model - 楊梓暄、許少麒](#_r4ehsq3qz53p)

[模型目標](#_ggbjiu1ihdai)

[資料前處理與 Frame 切割](#_yor9uukscq25)

[輸入資料格式](#_rejlzbar11k0)

[模型架構](#_f80gvfj68j33)

[模型訓練](#_ujsrk9cgr2s6)

[模型部署](#_gepxwwa5lhrx)

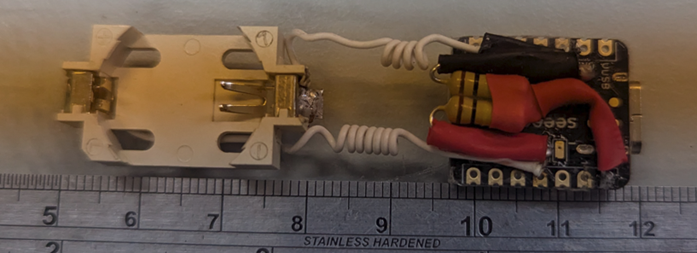
[類別標籤](#_3x5afzn8cbki)

# The design of the hardware - 歐育成、楊梓暄

## Component:

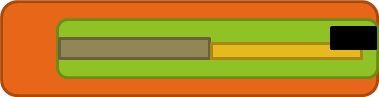
|  |  |  |
| --- | --- | --- |
| Board:  **Seeed XIAO nRF52840 Sense** | 20\*17.5\*5 mm |  |
| Battery w/ Battery Holder :  **LIR2032** | 32\*16\*6 mm |  |
| Badminton racket |  |  |

## Board circuit welding:

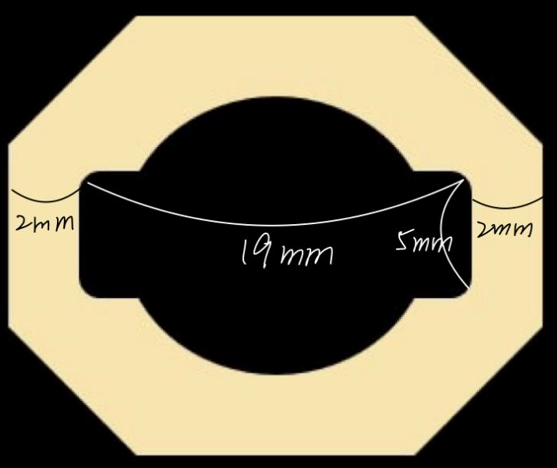


## Battery holder inside of wood:

### 預計內部配置：

* 黑色：type c 接口 5\*8.5\*3.5 mm
* 綠色：原手柄開孔 35(深)\*12\*12 mm
* 灰色：電池 20\*20\*3(厚) mm
* 黃色：板子 20\*17.5\*5(含USB厚度) mm
* 橙色：手柄

### 球拍手柄加工(楊梓暄)：



## Combining all components:

## 

## 

## 

## 

## 

## 

## 

## Arduino code battery charging related mentions:

* The voltage is checked every **minute**.
* To make charging more efficient, when the voltage is insufficient (< **3.7V**), it switches to **high current** charging (**P0\_13**, OUTPUT pin is **LOW**).

//高低電流切換

void updateChargingMode(float batteryVoltage) {

if (batteryVoltage < 3.7) {

if (!isHighCurrentCharging) {

pinMode(P0\_13, OUTPUT);

digitalWrite(P0\_13, LOW); // 切換到高電流充電

isHighCurrentCharging = true;

Serial.println("Switched to HIGH current charging.");

}

} else {

if (isHighCurrentCharging) {

pinMode(P0\_13, OUTPUT);

digitalWrite(P0\_13, HIGH); // 切換回低電流充電

isHighCurrentCharging = false;

Serial.println("Switched to LOW current charging.");

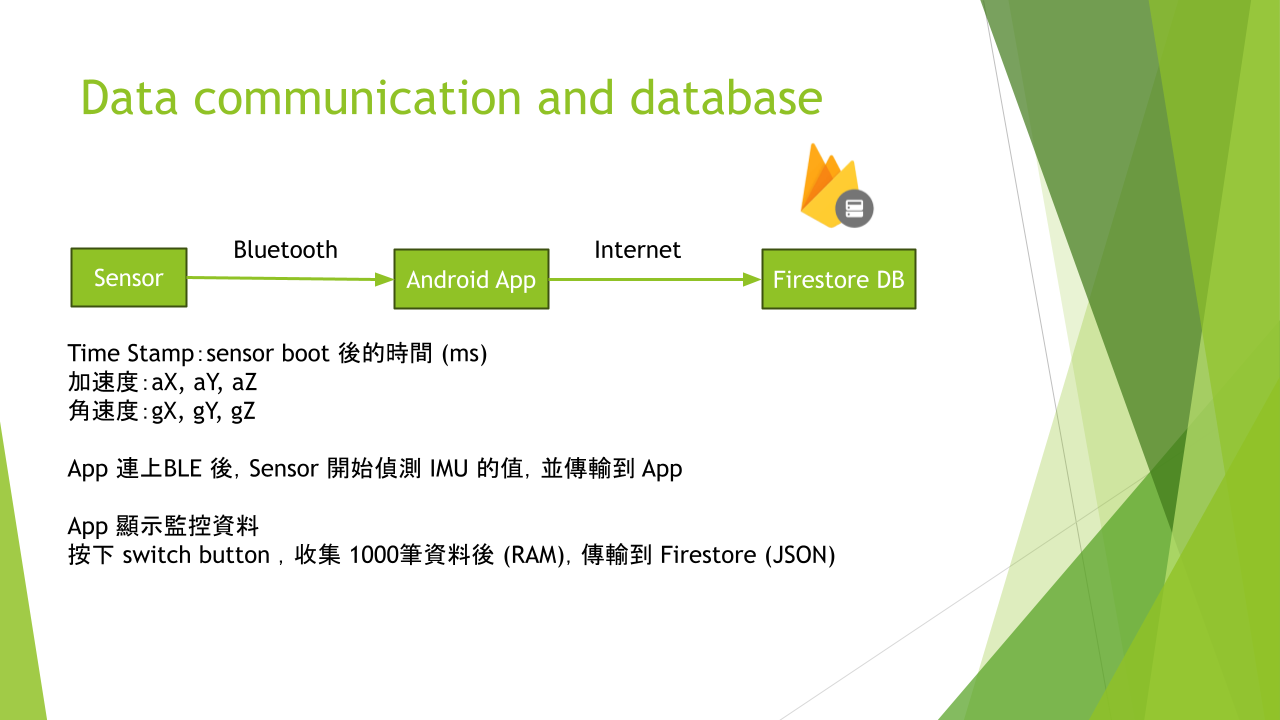
}

}

}

# The implementation of data communication and database - 許少麒

相關程式碼：IMU\_Capture\_BLE\_v9.ino



## Arduino IDE 開發環境與程式設定說明

### 開發環境與硬體

* 開發板 (Board)：Studio XIAO nRF52840 Sense
* 配置 (Configuration)：Seeed nRF52 mbed-enabled Boards (v2.9.3)
* 使用函式庫 (Libraries)：
  + ArduinoBLE (v1.4.0)
  + Seeed Arduino LSM6DS3 (v2.0.4)

### IMU參數設定

* Sensor：LSM6DS3TR
* Device Address：0x6A
* 加速度計：
  + 資料輸出頻率（ODR）：416Hz
  + 量測範圍（Full Scale）：±16G
  + 帶寬設定（Bandwidth）：100Hz
* 陀螺儀：
  + 資料輸出頻率（ODR）：416Hz
  + 量測範圍（Full Scale）：±2000 dps
  + 帶寬設定（Bandwidth）：400Hz
* I²C傳輸速率：400kHz

### 資料收集資訊

* 資料收集頻率：每 20ms 傳送一次
* 單筆資料格式（30 bytes）：
  + timestamp（時間戳 , 4 bytes）
  + aX, aY, aZ（加速度三軸, 各 4 bytes）
  + gX, gY, gZ（角速度三軸, 各 4 bytes）
  + rawVoltage（電壓讀值, 2 bytes）
* 電壓讀取：每 1 分鐘透過 analogRead(A0) 進行一次電壓測量

### BLE設定

* BLE裝置名稱：SmartRacket
* Service UUID：0769bb8e-b496-4fdd-b53b-87462ff423d0
* Characteristic UUID：8ee82f5b-76c7-4170-8f49-fff786257090
* 每 20ms 發送一筆資料（包括timestamp、加速度、角速度、電壓資訊）
* 每筆資料大小：30 bytes

### 電池設定

* 設定 P0.13 (D13) 為高電位 (HIGH)：

➔ 啟用低電流充電模式，避免因電流過大影響電池與板子安全。

* 設定 P0.14 (D14) 為低電位 (LOW)：

➔ 保持 ADC 功能開啟，避免在充電過程中 P0.31 引腳損壞。

# The design of UI (APP) - 李東霖、許少麒

專案程式碼 <https://github.com/bigheadfjuee/iot_imu_ble>

## Development environment

* Flutter 3.29.3
* Dart 3.7.2
* Android SDK - API Level 35
* Android SDK Build-Tools 36
* Android SDK Platform-Tools 35.0.2
* NDK 29.0.13599879 rc2
* CMake 4.0.2

## Flutter pub packages

* flutter\_blue\_plus 連接 BlueTooth 的套件
* firebase\_core, firebase\_database, cloud\_firestore 將 Sensor 資料上傳到 Firebase Cloud Fire Store
* fl\_chart 顯示 Sensor 波型的套件
* tflite\_flutter 將 Tensor Flow Lite 放入 Android 手機使用的套件

## Flutter project 注意事項

### firebase.json, google-services.json

需要配合 <https://console.firebase.google.com/> 雲端專案使用

### assets / badminton\_model.tflite

訓練後的模型要輸出為 Tensor Flow Lite 的格式，並放到 assets 資料夾下

### android / settings.gradle.kts

因為有使用 tflite\_flutter 套件，在 gradle 中要加入下面的內容，不然 build apk 時會出錯

dependencies {

implementation("org.tensorflow:tensorflow-lite:2.10.0")

implementation("org.tensorflow:tensorflow-lite-gpu:2.10.0")

implementation("org.tensorflow:tensorflow-lite-gpu-api:2.10.0")

}

## BLE 掃瞄及連線 - 許少麒

UI相關程式碼：lib/pages/ble\_scan\_page.dart

李東霖 - 小修改：將裝置名稱為 SmartRacket 的設備 (Arduino中可自訂) 排在前面



### 

## 傳送資料到雲端 - 許少麒

相關程式碼

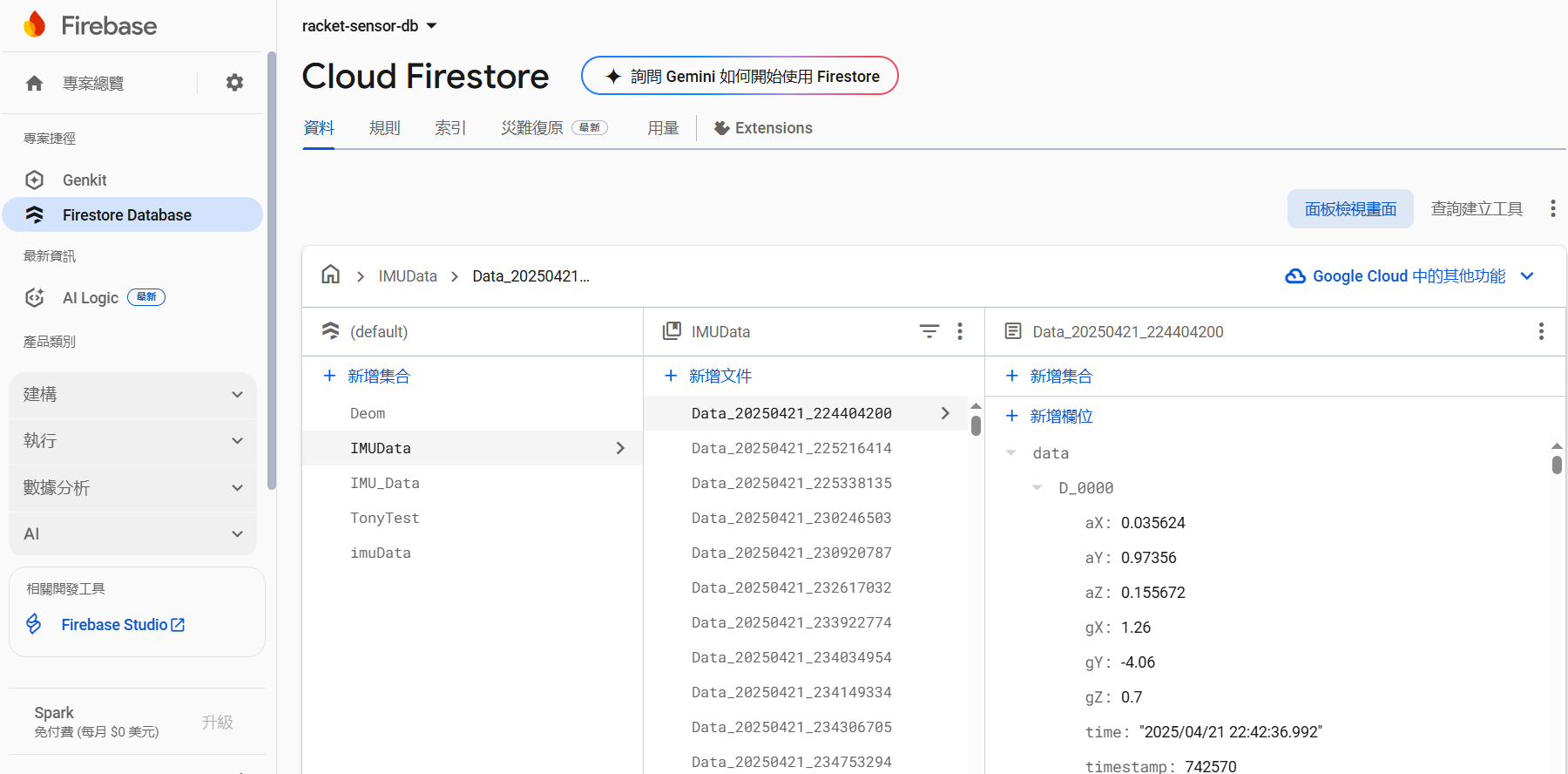
lib/components/ble\_data\_manager.dart 處理由藍牙收到的 IMU 即時資料，使用 NotifyListener 的方式

lib/components/ble\_data\_receiver\_page.dart 上傳至 Firebase 雲端的功能

### 

Firestore連結

<https://console.firebase.google.com/project/racket-sensor-db/firestore/databases/-default-/data/~2FIMUData~2FData_20250421_224404200?fb_gclid=Cj0KCQjwiqbBBhCAARIsAJSfZkaCX4ZXoakJ00ZIN0Z8NXxpDFh6DktTXnc63GAC0p_bn9LjmtWWD2AaAnZ2EALw_wcB>



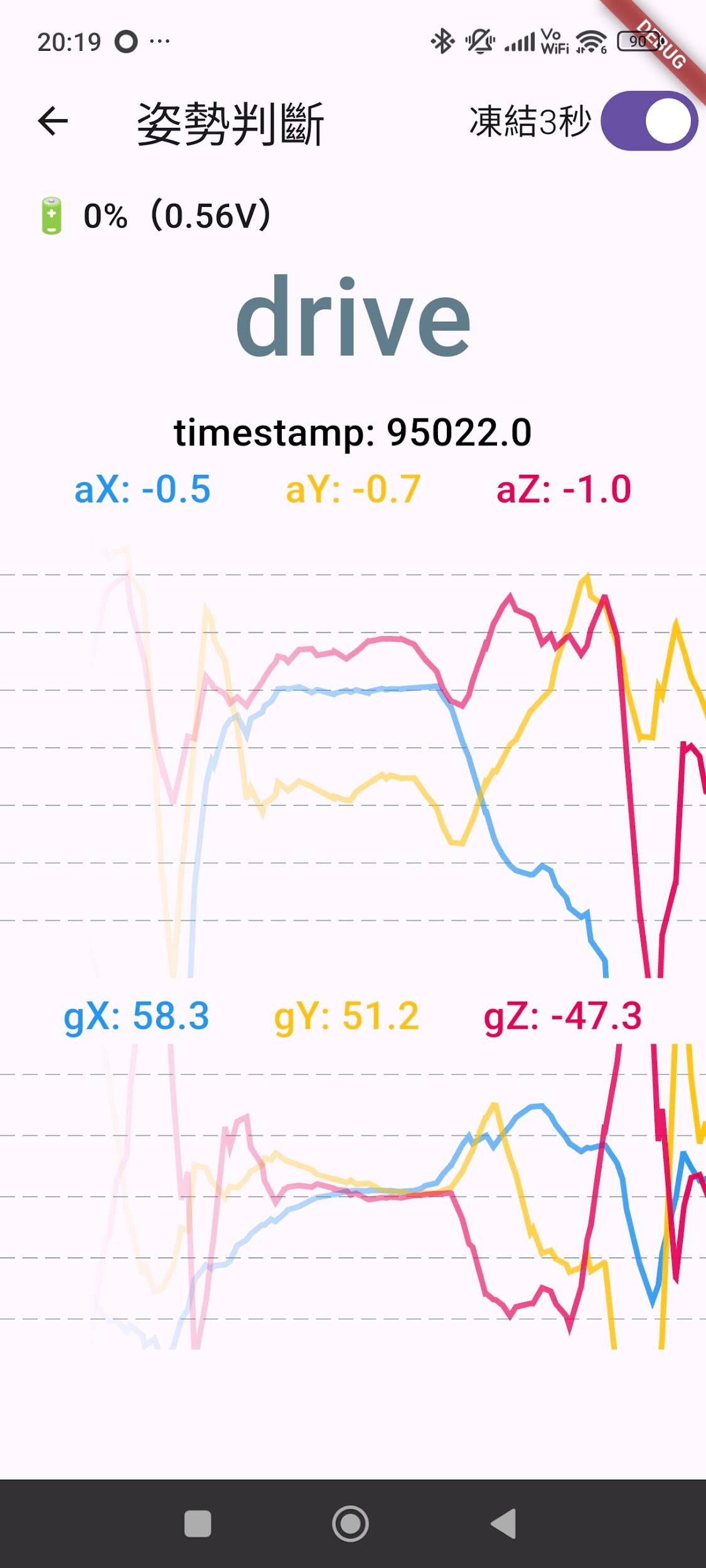
Firestore 資料結構

* Collection：IMUData
* Document：Data\_yyyyMMdd\_HHmmssSSS
* 內容：
  + data 欄位是 Map 形式
  + Key 為 D\_0000 ~ D\_0999
  + Value 為單筆 IMU 感測資料



## 即時偵測 Sensor 資訊，以 Line Chart 顯示 - 李東霖

程式碼：lib/components/line\_page.dart



### 

## 模型嵌入 App & 感測資料處理與模型判斷應用 - 李東霖、許少麒

相關程式碼 lib/components/ble\_data\_posture.dart

目前邏輯與實作方式說明

在系統以固定 40 筆 IMU資料為一個輸入批次，當緩衝區收集到 40 筆資料時，便組成模型所需的四維輸入格式 [1, 40, 6, 1]，並直接呼叫 TFLite 模型進行推理一次。模型會輸出三個機率值（對應到姿勢類別：drive、other、smash），系統會找出其中機率最大的索引值，轉換成對應的球種名稱並更新畫面。

若模型預測結果為 drive 或 smash，畫面會凍結顯示該姿勢 3 秒鐘，3 秒後自動清空結果，恢復為預設的 ---。但在凍結期間內，如果有新的非 other 姿勢出現，畫面會馬上更新為該新結果，並重新計算 3 秒倒數。系統不會對過去的結果進行統計或投票，而是直接根據當下 40 筆的推理結果顯示姿勢。

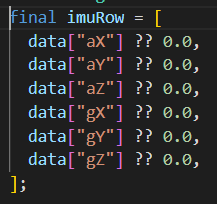
Interpreter

Tensor Flow Lite 檔案

1. 載入 Model

IMU Data

run



2. 持續輸入資料



3.推論結果

轉成結果 [drive, other, smash] 顯示

40 sample 為一組

日後再精進

目前的做法是將每 40 筆感測資料，使用滑動視窗（位移 1 筆）方式作為模型的輸入，並以第一個被判斷為非 other 的姿勢作為顯示結果。若有新產生的非 other 結果，系統會立即更新姿勢並重新計算 3 秒的凍結時間。為了進一步提升穩定性，日後可考慮在「位移 40 次」的過程中，將所有推理結果加總後，取出出現次數最多的姿勢，作為最終顯示結果。若模型本身準確度足夠，這種統計方式有機會讓最終判斷更精確。

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

40 筆

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

IMU

[ ax, ay, az,

gx, gy, gz ]

output[ a, b, c]

40個output 結果

x 40 次

[ sum(ouput[0]),

sum(ouput[1]),

sum(output[2]) ]

……..

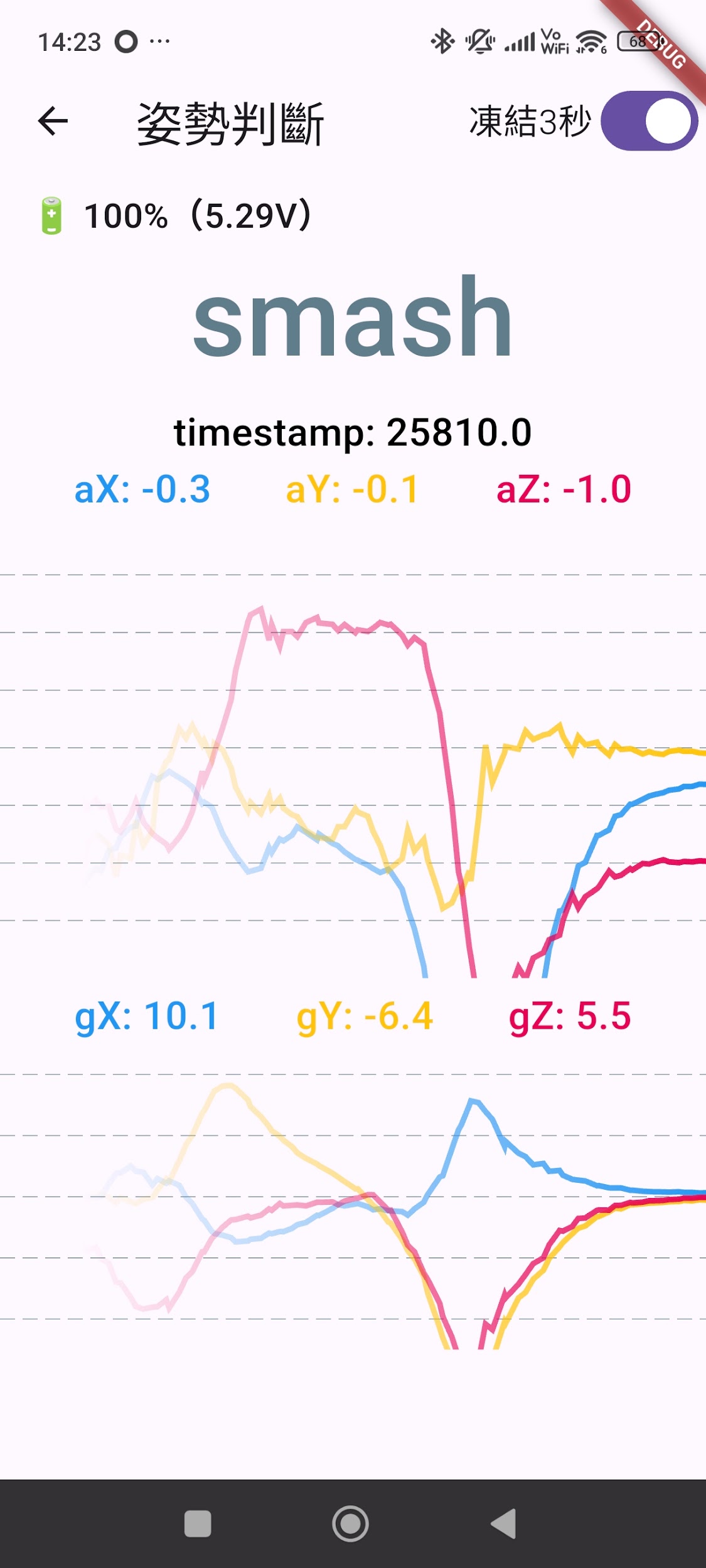
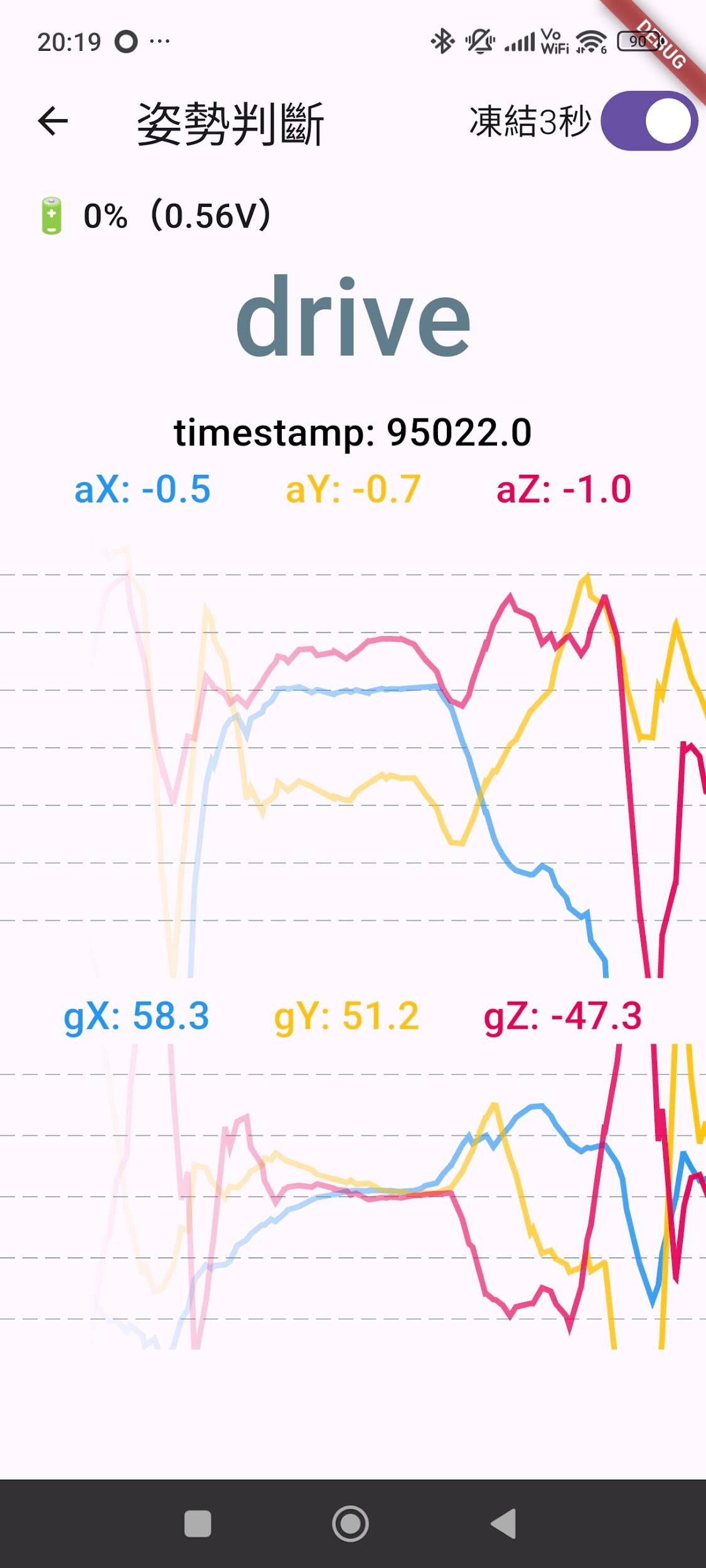
……..

選40次加總後最大的

當作最後判斷結果

實作模型的輸入資料，輸出結果

通常為 Other，若有偵測到發球姿勢會顯示 drive，偵測到殺球姿勢則為 smash



# 

# Data Collection & Labeling - 楊梓暄

資料收集:https://drive.google.com/drive/folders/1o8LODBSVJFH1U4MGCzjMudopOV3ubo2V

資料影檔:https://drive.google.com/drive/folders/1NZYH\_0yP\_bxdvzpZEIDBN2vpKSe83XDy

標記程式:

* mark\_label\_by\_time\_new.py

輸出圖檔程式:

* create\_img\_to\_word.py

DEMO影片 (楊梓暄、許少麒) :

https://drive.google.com/file/d/1bjbaYI3h-fRR41D2Crcyp3tjSIGnCCh7/view?usp=drive\_link

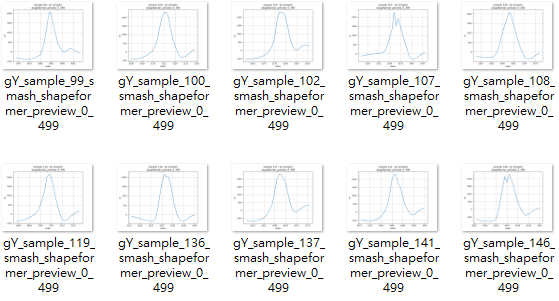
資料收集:

* 收集smash/drive兩種球種，部分使用打球者拋球後打，部分使用第二人拋球後打

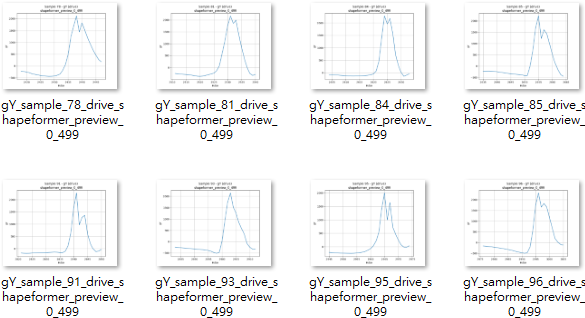
資料標記方式:

* 比對過後選擇gY作為參考軸
* 合併所有excel後讀取指定資料夾，依照time欄位過濾出指定時間
* 利用標準差計算出突增值
* 在gY欄找到突增值，若有連續超過突增值的，則找到最高點的第一筆
* 往前抓19筆，往後抓20筆，加上當筆資料，總共40筆，依照設定標記上，smash/drive
* 剩下的抓連續40筆標記為other
* 移除時間間隔不在10~40的資料段
* 匯出資料成圖檔確認標記結果

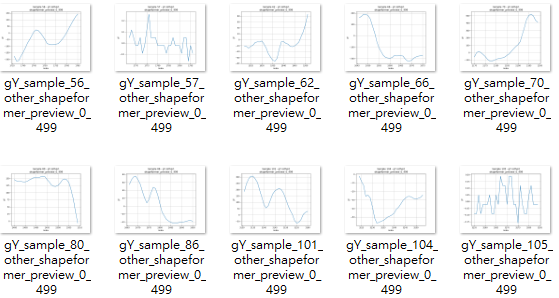
smash



drive



other



# The design of classification model - 楊梓暄、許少麒

相關程式碼：train\_badminton\_model.py

## 模型目標

分類模型的目標是透過 40 筆連續的 IMU 感測資料，即時辨識羽球擊球動作 (drive、other、smash)。

## 資料前處理與 Frame 切割

* 資料來自 Excel 檔案 output\_label\_all.xlsx，內容包含時間序列的 IMU 感測資料與標籤。
* 以 40 筆連續資料 為一個 frame，stride = 40（每次不重疊取 40 筆）進行分段。
* 針對三類別資料（drive、smash、other）分別切 frame，並利用 隨機過採樣/下採樣 平衡每個類別到 2000 frame。
* 最終資料集進行隨機排列，並以 80% 作為訓練集、20% 作為驗證集。

## 輸入資料格式

* 每個 frame 含有 40 筆時間點，每筆資料有 6 個特徵：
  + 加速度 (aX, aY, aZ)
  + 角速度 (gX, gY, gZ)
* 最終輸入格式為 4 維張量：Shape: (批次大小, 40, 6, 1)

## 模型架構

該模型為 卷積神經網路 (CNN)，專門用於擷取 IMU 資料的空間與時間特徵。

模型層結構：

* Conv2D 卷積層（16 filters，2x2 kernel，ReLU activation function）
* Batch Normalization
* Dropout (0.2)
* Conv2D 卷積層（32 filters，2x2 kernel，ReLU activation function）
* Dropout (0.1)
* Flatten
* Dense 全連接層（64 units，ReLU，L2 正則化）
* 輸出層（softmax，3 分類）

## 模型訓練

* 損失函式：categorical cross-entropy
* 優化器：Adam
* 訓練週期：50 epochs
* 批次大小：64
* 驗證比例：20%（驗證模型效能）

## 模型部署

* 訓練完成後，將模型以 Keras 格式 (badminton\_cnn\_model.h5) 儲存。
* 為了能在 Android App 中執行，即時推論時使用，將模型轉換為 TFLite 格式 (badminton\_model.tflite)。

## 類別標籤

* 模型最終輸出三類別：drive、other、smash。
* 透過 LabelEncoder 保留類別對應，確保 App 部署時結果一致。