

# Flappy Bird Using STM32 board Sensor

b06203017 李俊諺 b06504016 林家宏 b06701214 王群博

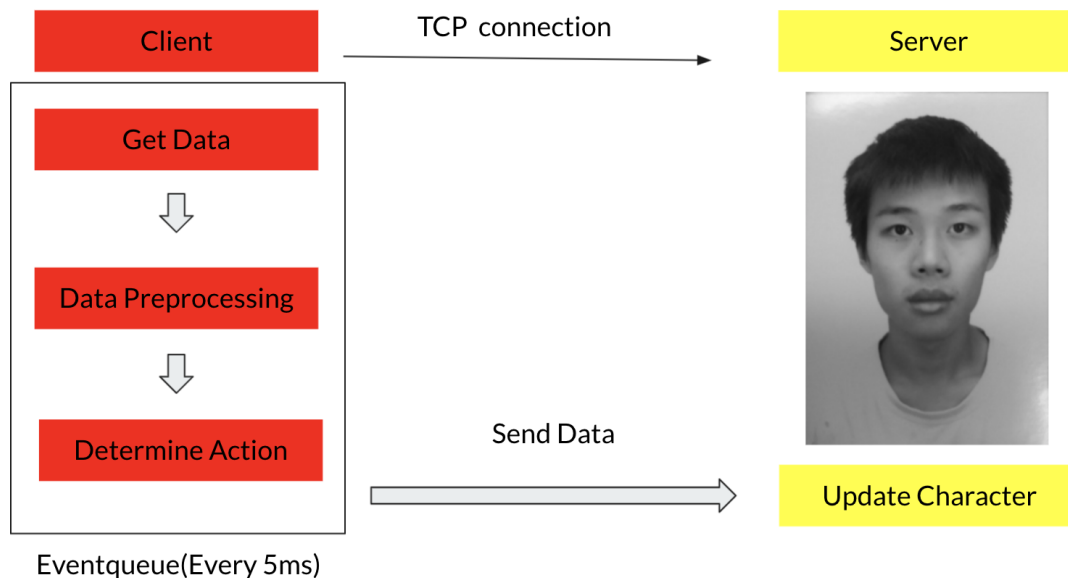
## 1) 動機

Flappy bird是一款藉由空白鍵來操作遊戲角色的遊戲，其操作方法如影片連結(<https://www.youtube.com/watch?v=l69adfEqwC0>)。從影片連結可以看到當按空白鍵時，遊戲角色就會飛起一瞬間並迅速向下(原來的遊戲有重力設定)。而原遊戲由於只需要藉由簡單的空白鍵指令就可以遊玩，玩家不免覺得有些無聊。因此為了增進遊戲體驗讓玩家更身歷其境，我們改用stm32 board來操作遊戲角色，讓玩家有親自駕駛遊戲角色的體驗(類似wii的玩法)，並添加更多操縱方式。

## 2) 設計細節

我們的做法是將遊戲的重力設定拿掉，如此一來就能讓遊戲角色做出更多動作，而我們是利用手部上下傾斜stm32 board來操縱遊戲角色的上下移動，詳細的細節如下

流程圖：



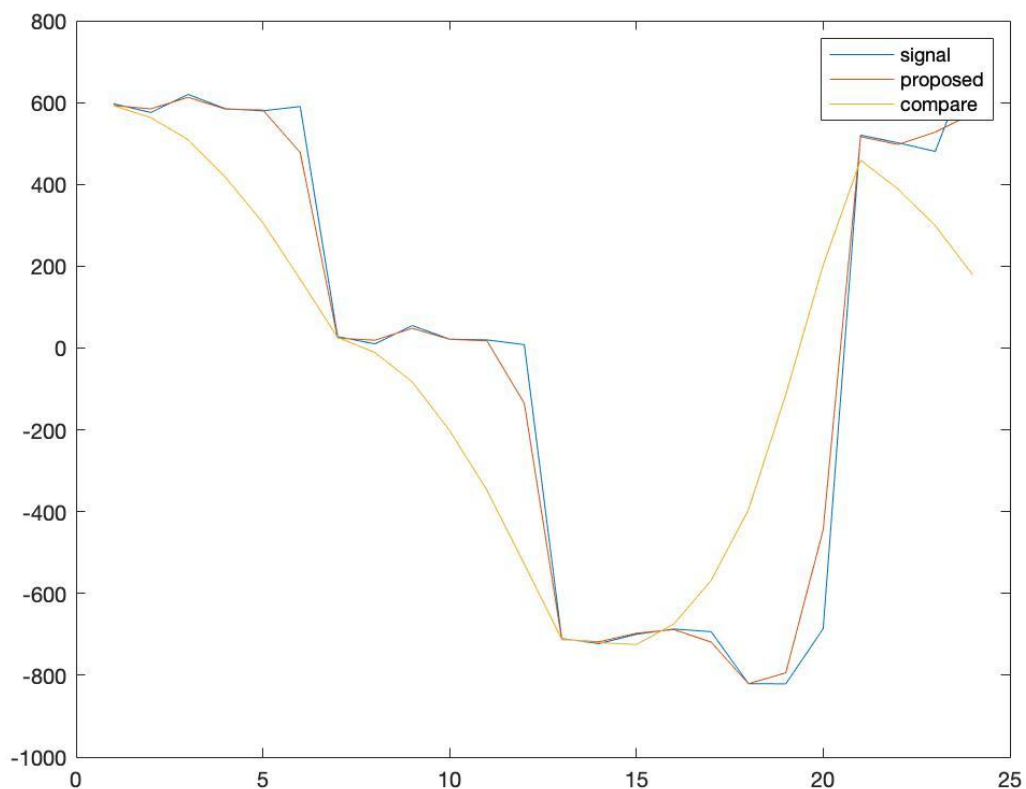
### ● 資料處理

資料處理的部份是在client端執行，首先三軸加速度計負責接收xyz軸的數據。在接收完數據之後，就進入資料前處理的階段，這邊我們資料前處理的方式是用queue先把資料存起來並將存在queue裡的數據做加權平均。這樣處理資料的好處是它能有效避

免雜訊影響遊戲的操作，在測試的過程中，我們試了多種長度的queue及各樣的加權參數來產出最終數據，在最後我們認為長度為2的queue能達到最好的效果(加權參數為0.2跟0.8)，我們的設定是讓新進的資料占有較大的權重。

- 不同資料處理方法比較

在處理資料時，可能因為有手抖等等情形導致資料會有誤判的問題，我們在做的時候有比較許多不同的方法來處理訊號，利用較長的impluse response可能會造成edge的消失，而太短impluse response會造成訊號較不平滑，在最後取捨後是選用 $h = [0.2, 0.8]$  (proposed)，而對照組則是選用 $h = [0.05, 0.1, 0.15, 0.2, 0.25, 0.25]$  (compare)，以下為用matlab實驗的結果



- 判斷操控

當資料經過queue做完處理後，接下來就進入資料判定的階段，我們改良後的遊戲玩法有五種操作方式，這五種分別為操控遊戲角色大幅向上，正常幅度向上，維持高度，正常幅度向下以及大幅向下。而要怎麼達成這些功能呢？我們利用y軸的加速度變化來判

斷，當把stm32 board大幅上擺時，則遊戲角色也會跟著大幅上移，其它四個操作方式也是依此原理。值得注意的是我們設計的方式是只有在按下stm32 board的button後(利用DigitalIn這個interface來讀值)，才會觸發向上以及向下的機制，如此一來就能避免誤判的情形。而當不按button時，遊戲角色只會保持不動。而詳細的threshold設定如下表：

| Threshold                          | Value           | Discription |
|------------------------------------|-----------------|-------------|
| $\text{data}[y] < -500$            | $\text{up} = 4$ | 大幅下降        |
| $-500 \leq \text{data}[y] \leq 50$ | $\text{up} = 3$ | 小幅下降        |
| $-50 \leq \text{data}[y] \leq 50$  | $\text{up} = 2$ | 不動          |
| $50 < \text{data}[y] \leq 500$     | $\text{up} = 1$ | 小幅上升        |
| $500 < \text{data}[y]$             | $\text{up} = 0$ | 大幅上升        |

- 傳送資料

從上列數據可以看到我們將判定完的資料數據以0, 1, 2, 3, 4來表示，並把up值傳到server，我們這樣做的好處是server能接收到非常乾淨的資料，降低收到雜訊的機率。而因為這遊戲非常講求上下擺動的準確性，所以在資料傳輸方面我們是採用TCP連接，因為TCP能確保封包完整的傳到server，而UDP無法保證能將封包完整的傳到server。而在client端接收資料以及傳輸資料的部分，我們是將這些event送進eventqueue去排程，而這邊我們的設定是讓eventqueue每5毫秒call\_every一次。

- 其他貢獻

另外在client端要提的是我們程式的寫法，我們的寫法是建立兩個class，一個是Sensor class(負責接收資料)，另一個是Wifi class(負責傳輸資料)，這樣結構化的寫法能增進程式的可讀性，讓以後要參考我們code的學弟妹能輕易看懂。而在server端我們是採用multithreaded的方式。我們總共用了兩個thread，一個thread負責遊戲畫面的執行，另一個thread則負責接收client端所傳來的資料，如此以來遊戲就能順暢地進行。另外還要提的一點是當我們沒按button時，queue裡的資料會清空，這樣的作法能避免queue存到沒

必要的數值而影響遊戲角色的操作。

### 3) 成果

總結一下，在這次的project我們所用到課堂上學習的知識有wifi, eventqueue, multithread, DigitalIn以及3-axis accelerate sensor。而我們設計的演算法能以低延遲，高精確度的操控來玩遊戲

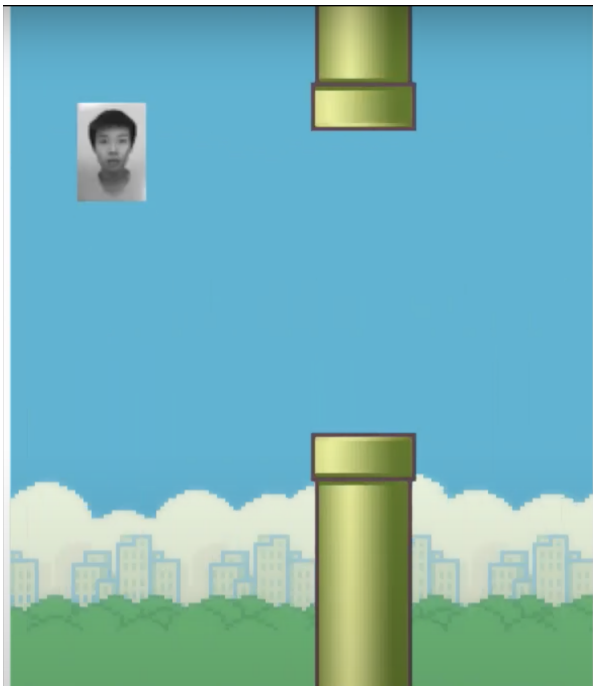
- demo影片：

<https://youtu.be/YWr3OLMTgC4>

[https://youtu.be/JaakABC\\_8xs](https://youtu.be/JaakABC_8xs)

<https://youtu.be/XWOblXyJOK4>

- ScreenShot from gameplay



- Github連結：

[https://github.com/AlexLee1999/ESlab\\_final\\_flappy\\_bird](https://github.com/AlexLee1999/ESlab_final_flappy_bird)

### 4) 參考資料：

- Reference of the Game:

<https://www.youtube.com/watch?v=UZg49z76cLw>

- Mbed OS Reference Documents

[https://os.mbed.com/teams/ST/code/DISCO\\_L475VG\\_IOT01-Sensors-BSP/](https://os.mbed.com/teams/ST/code/DISCO_L475VG_IOT01-Sensors-BSP/)

[https://os.mbed.com/teams/ST/code/BSP\\_B-L475E-IOT01/](https://os.mbed.com/teams/ST/code/BSP_B-L475E-IOT01/)