ESLab Final Project Report: Tetris Battle

B07901090 劉紀霆, B08901169 梁正, B08901123 何景盛

Github: https://github.com/Gting6/NTUEE_EMBED_LAB/tree/main/Final

摘要與動機

在大約十年前 Facebook 剛開始流行的時候，因為手機遊戲沒有很發達，因此大家會上 FB 線上玩一款連線對戰遊戲叫做 “Tetris Battle”，遊戲規則和俄羅斯方塊大致雷同，在一定時間內，消除最多橫線的玩家獲勝。在 2019 年的時候，這款遊戲無預警宣布停止更新，也結束了我們兒時的回憶，因此，我們希望藉由上課所學的技巧，將這款遊戲重現出來。

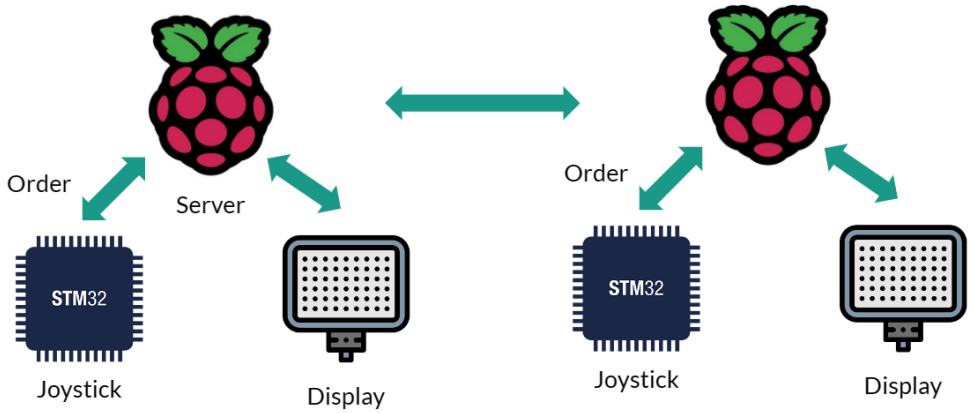
在這份 Project 中，我們成功的實做了可兩人連線的 **Tetris Battle** 對戰遊戲，並延伸加入搖桿、**LED** 面板，最後成果遊戲畫面十分順暢且延遲低。我們利用 RPI 當作 Server、運算邏輯，STM32 作為搖桿，LED Board 當作螢幕，大量運用上課所學的 Socket Programming, Threading, Mutex / Lock, I/O Multiplexing, RPI Control LED 等等技巧來實現。

DEMO 連結: https://youtu.be/rcX75-qD13I
投影片連結: ESLab DEMO

遊玩方法

向右旋轉: 保留方塊
向上晃: 旋轉方塊
向下晃: 方塊下墜
按左按鈕: 方塊向左移動
按右按鈕: 方塊向右移動

專案架構



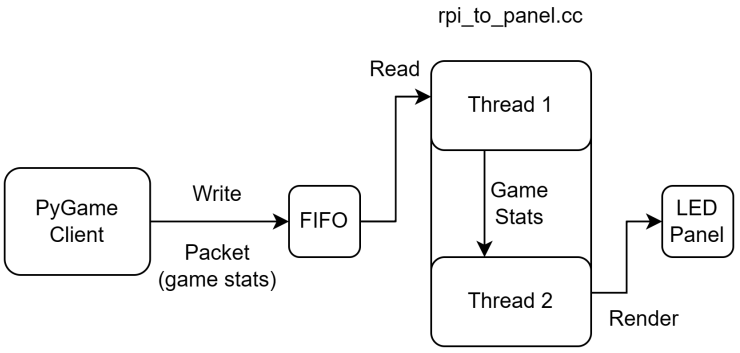
使用一台 RPI 同時作為 Tetris Server + Tetris Client，另外一台 RPI 作為 Tetris Client。STM32 作為 JoyStick，LED Board 作為 Display。裝置之間的綠色箭頭皆是用 Wifi 進行溝通。

實作細節

1. 遊戲操控：
   我們以STM32來當作搖桿進行操控，因此需要用STM32的Gyroscope和Accelerometer來判斷向上晃，向下晃，向右轉這三種手勢，還有外加左右兩顆按鈕來實現。
   - 向右轉: 以Gyroscope Y軸的角加速度做積分來取角度，可能角度超過某個值就代表有旋轉。
   - 向上晃和向下晃: 以Gyroscope X軸和accelerometer z軸的兩個值來做判斷，我們向上晃時會有一些向上的角度和加速度，向下晃會有俯角和向下的加速度，用這兩個值來進行判斷。
   - 左右按鈕: 以STM32的InterruptIn的fall和rise來實現。

   難點與解決
   在一開始做向上向下晃是用Accelerometer z軸的加速度來進行判斷，收到一個信號做一下clip，加速度超過某閾值就代表State 1，加速度少過某閾值就代表State -1，其他就代表State 0，然後再用FSM來改變State進而做判斷，可能向上晃是從1 => -1 => 1 => 0，向下晃是-1 => 1 => -1 => 0。我們發現這方法有幣端，可能因為收到的信號不是3個為一組(有可能是1 => -1 => 0)，這時候FSM就會出錯。因此，我們修改了這方法，變成用加速度和角速度共同來判斷。在實作的過程中，我們也知道mbed的callback function只能有一個input argument，解決方法是用一個struct來當input argument。

2. 遊戲畫面顯示：
   我們以一塊大小為64 x 64 pixels的LED全彩面板模組來做為我們的遊戲畫面顯示方式，參考第三方提供的API [1]來操控LED面板。
   在程式的部分，我們使用兩個thread，一個thread用來讀取FIFO，接收來自PyGame Client端的Packet (Status)，並隨後以其更新程式內定義之全域變數Stats；另外一個thread則是不斷的將Stats內的畫面矩陣顯示於LED屏幕上，這部分是使用第三方提供的API [1]來達成。



在優化程式的穩定性部分，我們使用pthread_mutex以及I/O Multiplexing (Select)來確保資料讀寫的順暢並不致過度堵塞。

3. RPI + 遊戲邏輯
   如前所述，程式分為兩個部分：Tetris Server 以及 Tetris Client。

   Tetris Server:
   - 不斷和 Tetris Client 利用 Socket Programming 交換畫面資訊。
   - 根據 Client 傳來的資訊，調整並回傳控制兩個玩家的遊戲狀態，分為"IDLE"、"等待另外一位玩家"、"遊戲進行中"、"獲勝"、"失敗"、"平手"。

   Tetris Client: 同時執行三個 Thread
   - STM32 Listner: 利用上課所教的 Socket Programming，透過 Wi-Fi 去不斷接收 STM32 傳來的指令，接收到指令之後便寫入一個和其他 thread 共用的 Queue 之中。
   - Main: 處理俄羅斯方塊的邏輯，每一秒鐘會去讀 60 次 Queue 的資訊，如果 Queue 為非空，則讀取指令，進行畫面的更新並利用 Python 的 ctype library，將畫面的 data 寫入 FIFO 給 LED，每當畫面有所更新，或者時間超過 1 毫秒，便會送出其最新資料給 Tetris Server，並且獲得對手的最新資料。
     - 附註：利用 FIFO 和 LED 溝通時，我們定義的資料傳輸如下

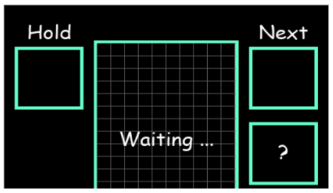
```
[('player0_main', c_int*20*10),
 ('player0_shift', c_int*4*4),
 ('player0_next', c_int*4*4),
 ('player0_pts', c_int),
 ('player0_cbs', c_int),
 ('player1_main', c_int*20*10),
 ('player1_shift', c_int*4*4),
 ('player1_next', c_int*4*4),
 ('player1_pts', c_int),
 ('player1_cbs', c_int),
 ('game', c_int),
 ('time', c_int)]
```

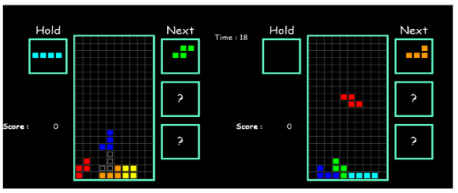
   - Timer Count: 接收到 Tetris Server 更新 “game” state 成為遊戲進行之後，便開始進行倒數計時並畫在畫面中間。

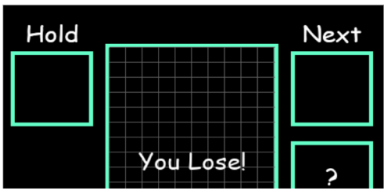
   遊戲設計:
   - 畫面由 10 x 20 的 2D array + 兩個 4 x 4 的 2D array 構成，每一格儲存數字，代表該格所要表示的顏色。顏色共有九種，分別代表黑色、各方塊的顏色、陰影等等。
   - 根據畫面判斷一個指令是否合理，如在最右邊邊界時候，指令輸入向右則無反應，詳細規則和 Tetris Battle 官網定義相同。

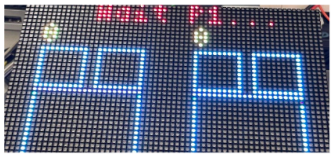
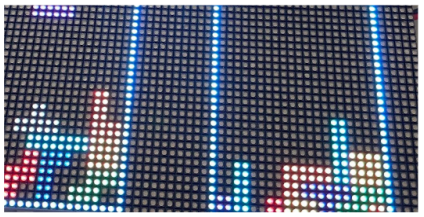
- 支援旋轉、左、右、速降、保留五種遊戲指令，並且每一秒鐘俄羅斯方塊會自動向下掉落一格。

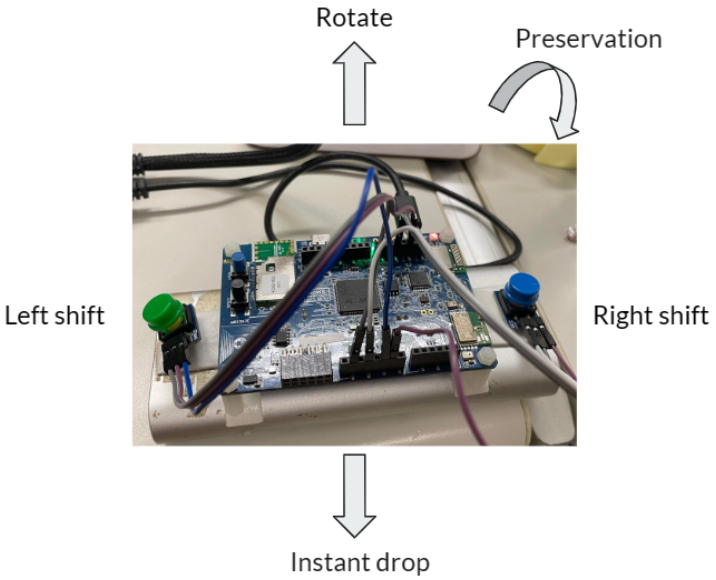
部分成果截圖







參考資料:

[1] https://github.com/hzeller/rpi-rgb-led-matrix
[2] https://github.com/hzeller/rpi-rgb-led-matrix
[3] Tetris Battle | Tetris
[4] Socket Programming in Python (Guide) – Real Python
[5] threading — Thread-based parallelism — Python 3.11.1 documentation
[6] ctypes — A foreign function library for Python — Python 3.11.1 documentation
[7] https://github.com/NTUEE-ESLab/2020-Pikachu-volleyball?fbclid=IwAR2OjNHmujj6J_A953Bmx-v4b5ptkjRtIlE-pMu8KrmLBQ7Pl4QlVhcfSbs
[8] Linux Tutorial: POSIX Threads (cmu.edu)