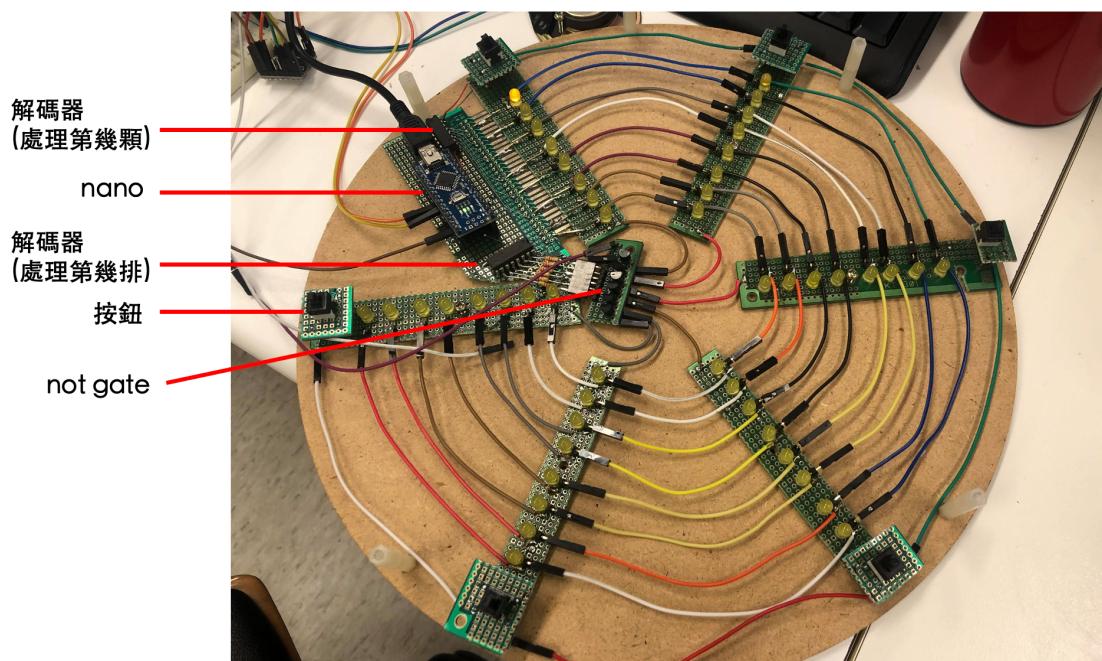
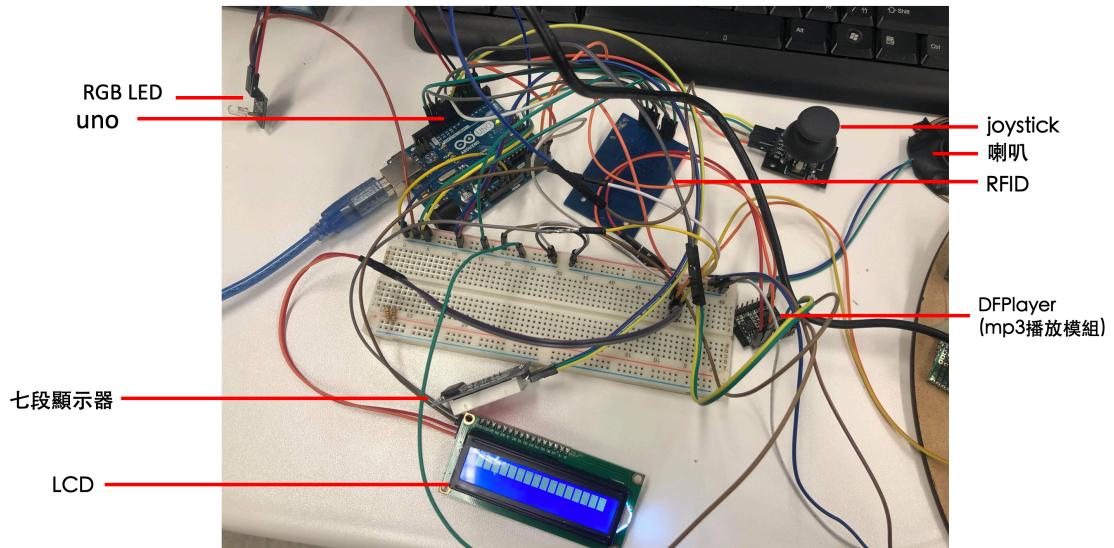


# 嵌入式系統概論 Term Project Report

## Aim

107062308 唐晏湄

### 一、架構介紹



此次實作使用 finite state machine 的概念來寫，所以我就用這樣的流程來介紹。

#### 1. GET\_USER

在這個階段會等待使用者在 RFID 上刷卡片，目前實作有 user1 跟 user2，分別對應兩張不同的卡片。讀取到卡片後就會在 LCD 上顯示出使用者名稱以及歷史最高分，接著進入 START\_GAME。

#### 2. START\_GAME

- (1) 在這邊也可以刷另外一張卡，切換成另外一個使用者。此時就會把 state 切換成 GET\_USER，來判斷現在讀到的使用者是哪一個。
- (2) 或是可以搖動搖桿來選擇要遊玩的歌曲，目前實作有兩首歌可供選擇。當切換到不同首歌時 LCD 上也會顯示出現在切換到哪一首歌。
- (3) 等到確定好要遊玩的歌曲後，就可以按下 joystick 的按鈕。接著 RGB 就會輪流亮起藍、紅、黃、綠三個顏色各一秒，等綠燈一亮起時就表示遊戲開始。播放音樂(DFPlayer)，傳送開始訊號給 nano，並進到 PLAYING 的階段。

### 3. PLAYING

- (1) 在這個階段 uno 會和 nano 頻繁溝通，使用的 protocol 是 I2C。首先 nano 會接到 uno 傳送過來的開始訊號，就知道後面傳過來的訊息都是要顯示的音符。在接到後面的訊息時 nano 就會操縱 LED 一個一個亮起來。當 LED 從內圈亮到最外圈時，若是 nano 偵測到使用者有在正確的時間點擊按鈕，就會回傳訊息給 uno 要加分。uno 接到 nano 的訊息後就會加分，並將當前的分數顯示在七段顯示器上。重複以上的過程，直到音樂播完，這一輪遊戲結束。音樂播完後 uno 會再發一個結束的訊號給 nano，讓 nano 可以知道結束，並把一些相關的變數歸零。
- (2) 這個音樂遊戲的譜面是由我自己設計的。我先寫了一個程式會顯示我每次按下按鈕的時間，距離音樂開始播放的起始時間間隔多久。而這些間隔時間就是我希望音符被玩家點擊到的時間。因此再扣掉音符從最內圈移動到最外圈的時間，就是音符要顯示的時機，也就是 uno 要傳送訊號給 nano 的時機。所以進入 PLAYING 的狀態後，uno 就會比較 millis() 讀到的當前時間，減掉開始播音樂的時間後，是否已經該放出下一個音符。如此一來便可以達到我們想要的效果。
- (3) 由於我們的板子上總共有  $6 \times 8 = 48$  個 LED 燈要控制，但是 nano 沒有那麼多腳位可以接，因此我們使用解碼器來解決這個問題。我們可以只用三個腳位輸出二進位的數字(因此範圍剛好為 0~7)，再把這個輸出的值傳進解碼器中，就可以拿到 6 個(column)或是 8 個(row)bit 的 output，有了 column 跟 row 的值，我們便能準確控制要亮哪一個位置的燈。而中間的 not gate 是因為要亮 LED 的話，兩邊接的腳必須要是一邊高電位一邊低電位。但我們的解碼器解出來在指定的位置都會是高電位，因此必須將結果通通反向(not)，才能讓想要的 LED 順利亮起來。

原先我們的預想是每次只會有一排的 LED 亮起來，所以玩家需要判斷燈亮不同位置，而選擇不同的按鈕來點擊。但很可惜我們在 demo 的前一天晚上發現控制 column 的那個解碼器壞掉，會導致同

時會有多排的同樣位置的燈亮起來，會讓玩家搞不清楚哪顆才是真正要打的音符。所以最後就改成每次都是亮一整圈的 LED，六個按鈕任選一個按即可。沒辦法達到我們原先預想的效果。

- (4) 判斷有沒有得分只要在偵測到玩家按下按鈕的瞬間，判斷現在是否有最外圈的 LED 燈正在亮著即可。
- (5) uno 會去讀取 DFPlayer 的一個 **busy signal**，來判斷現在是否還在播放音樂。若是發現已經播完了，就會發送結束訊號給 nano，然後把 state 切換成 CLEAR。

#### 4. CLEAR

最後音樂播完會在 LCD 上顯示最終成績三秒，並且判斷是否這個成績有比當前使用者的最高紀錄還要高。如果有，就更新最高紀錄。因此回到原先顯示當前使用者以及最高紀錄的畫面時，就可以看到最高紀錄會被更新。然後把七段顯示器、相關變數等歸零，最後回到 START\_GAME。

## 二、遇到的困難

### 1. 在 nano 中如何儲存要顯示的音符？

我後來在 nano 中開一個 48 格的陣列，來紀錄當前正在顯示的音符。陣列中會紀錄這個音符的 row 跟 column，若是這一格是空的就會設-1。每次有新的音符需要顯示，就會遍歷這個陣列尋找空的位置，然後把該音符的位置寫進去。而每次要亮燈時都會遍歷這個陣列，若是發現這格有紀錄音符的位置，就會去呼叫亮燈的 function 來把該位置的 LED 亮起來。等到音符掉落到最外圈時，會在 timer interrupt 中把該音符所在的位置清空。

### 2. 讀取 DFPlayer 的 busy signal

由於這個 **busy signal** 實在開始播歌後隔一小段時間才會出現 **busy** 的訊號，所以在寫程式時不可以直接在 play song 的 function 的下一行就馬上去讀取 **busy signal**。此時就會讀到 DFPlayer 還是 IDLE 的狀態，會導致錯誤的結果。正確的方式應該要在 play song 的下一行 delay 一小段時間，然後再去讀取。

簡言之是寫軟體和寫硬體不太一樣的觀念，但因為我寫軟體習慣了，所以當時這個 bug 花了我非常多的時間.....。

### 3. 各種接觸不良、杜邦線斷掉等等問題

經常發生因為線路接觸不良，導致無法正常運行。而我一開始還沒意識到會是硬體的問題，所以會一直去檢查自己的 code 到底哪裡有 bug。最後看了很久還是看不出來，花了很多時間，結果後來把線路壓一壓就正常了。或是板子上的 led 會不規則的亂閃，但我把我的程式碼看過好幾遍、Serial print 出所有可以印的地方，都發現印出來的東西都是正常的。試了一整個晚上，最後發現只是因為線路接觸不良。LCD 一開始沒有辦法正常亮燈也是因為和 uno 溝通的杜邦線斷掉了。一開始會沒有注

意到有可能是硬體的問題，到最後習慣了，發現運行狀況不對會習慣先把線路壓一壓再繼續檢查。

### 三、心得

此次的 **project** 其實做得非常趕。在期末考週前我只有把硬體(**nano**, **LED** 等電路)的部分先完成，大部分的 **coding** 都是到考完期末考後才開始。而且還剛好和 **os** 的最後一次作業的時間重疊，所以沒有太多時間可以好好準備，壓力很大。但完成後看到成品還是讓人很有成就感！這次的經驗也讓我學到更多 **arduino** 相關的應用，也對 **I2C** 的運用更加熟練，仍舊收穫良多。雖然這學期的 **loading** 非常重，嵌入式對我來說也是一堂壓力滿大的課，但也從中學習不少，很多在 **os** 學到的知識也都有應用在這堂課上呢！  
也很謝謝助教陪我們一整個學期，每次有問題也都不厭其煩的協助我，這一學期也辛苦你們了！