期末專題報告：觸摸鋼琴

王靖婷R12921A13、吳璵汐T13901203

1. **動機/介紹（Motivation/Introduction）**

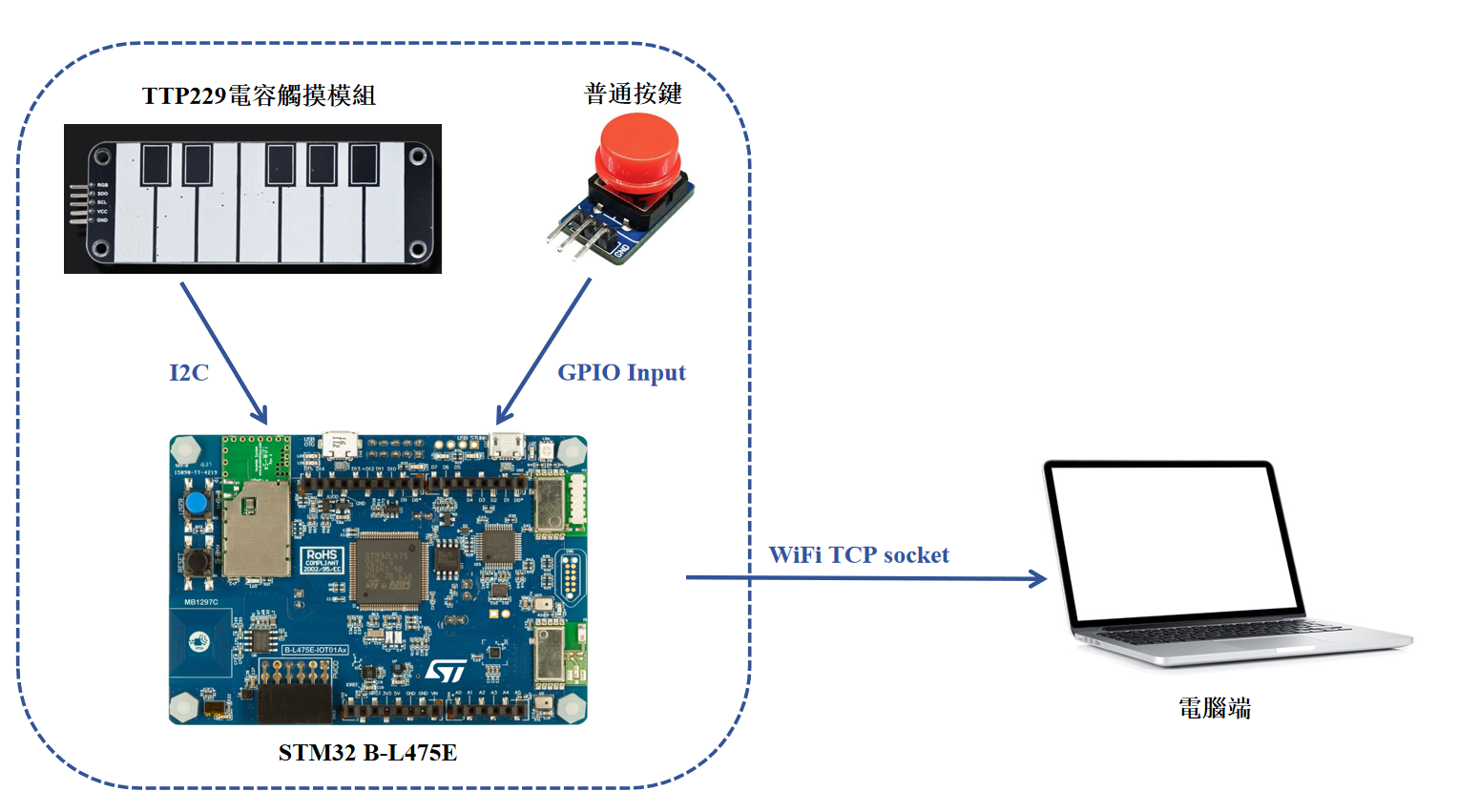
在本專案中，我們設計並實作了一套基於 STM32 B-L475E-IOT01A的攜帶式數位鋼琴系統。我們的動機是希望開發一個可實際彈奏的輕巧樂器裝置，具備實體按鍵、音高控制，以及音色切換功能，結合嵌入式硬體與電腦端音訊合成技術，打造具互動性的音樂體驗平台。因此，我們開發出一個原型，結合了電容式觸控感測、無線通訊以及軟體音訊合成技術。

輸入介面為一組鋼琴鍵盤形狀的電容式觸控感測器，連接至 B-L475E-IOT01A開發板，用以偵測對應各個音符的觸控訊號。當使用者觸碰按鍵時，開發板偵測對應的觸控訊號並處理，隨後透過 Wi-Fi client-server 架構（TCP protocol）將資料無線傳送至電腦。在 Host端（Mac 筆電），一個以 Python 實作的 Server負責接收這些訊號，並利用音色合成技術播放對應的音符聲音。

我們的目標是建立一個功能完整、可獨立運作的鋼琴輸入系統，支援實體按鍵、音高控制，並具備音色切換功能。本專案展示了嵌入式硬體、無線網路與互動音訊軟體的整合，提供了一個具表現力與擴充性的音樂互動平台。

1. **系統架構（System Overview）**

我們的目的是利用**STM32 B-L475E-IoT01A1開發板**，建構一個**互動式鋼琴控制與聲音生成系統**。 系統架構包含一個**TTP229電容觸摸鋼琴模組**（晶片型號：**8229BSF**）和一個**普通按鍵**，實現複合型輸入控制。目標是即時識別這兩種輸入信號，並通過**WiFi**將其發送到 Host 端， 由電腦通過Python program完成聲音播放、音色切換和音高轉換等功能。



### **1、**硬體組成****

* **STM32 B-L475E-IoT01A1：** 作為主控制晶片，負責GPIO pin腳配置、感測器 data採集、按鍵中斷回應，以及WiFi data發送。
* **TTP229電容觸摸模組**：一個12鍵的電容式觸摸模組，用於模擬鋼琴鍵盤，範圍為一個音階，檢測 User觸摸輸入。
* **普通按鍵（GPIO輸入）**: 用於控制音色切換，通過配置**外部中斷（EXTI）**來檢測 Interrupt事件。

### **2、**系統功能****

* **鋼琴按鍵檢測**： 通過GPIO對TTP229模組進行串行讀取，識別 User觸摸的按鍵， 轉換為一個16位的 bit array。 當檢測到有按鍵按下（非0值）時， STM32立即通過WiFi將對應 data 發送給電腦。並透過 python 功能實作，實現音高轉換。
* **物理按鍵中斷輸入**：普通按鍵配置為外部中斷模式（EXTI），可分別回應**按下**和**鬆開**兩種事件，可用於控制音色切換。
* **無線數據傳輸**： 數據通過**WiFi TCP socket**以位元組流形式 (byte stream)發送。STM32採用事件驅動機制（not polling），僅在檢測到有效輸入變化時利用 Interrupt進行data發送，減少功耗與CPU佔用。
* **電腦端回應處理：** Host端運行Python program接收數據。 對於TTP229數據， 根據接收到的按鍵值播放對應鋼琴音，並透過特殊事件觸發音高轉換。對於物理按鍵事件，按下按鈕則可以切換音色。

**三、 作法 -- STM32（Techniques--STM32）**

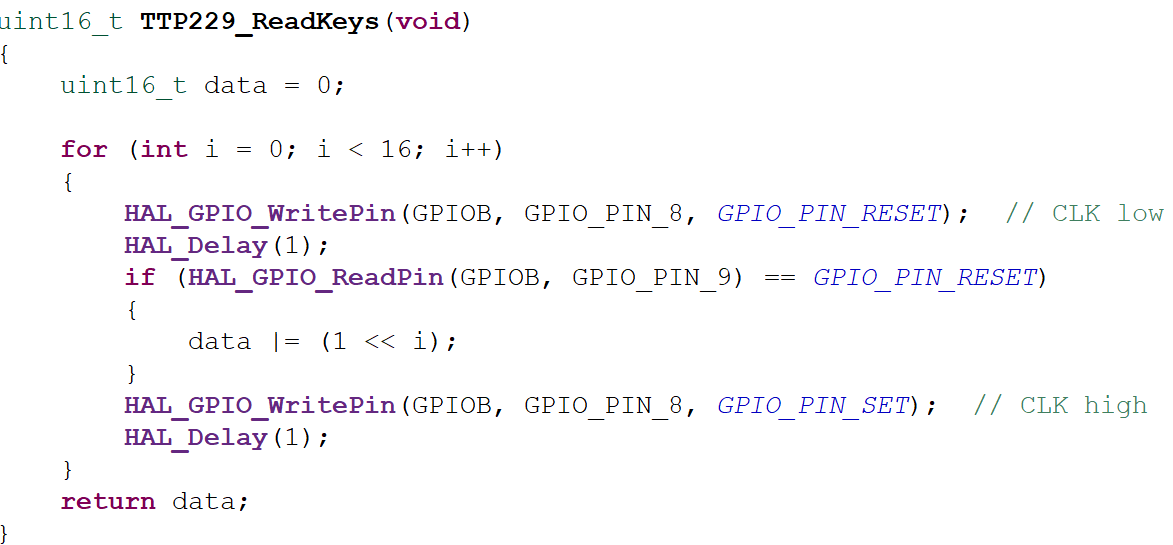
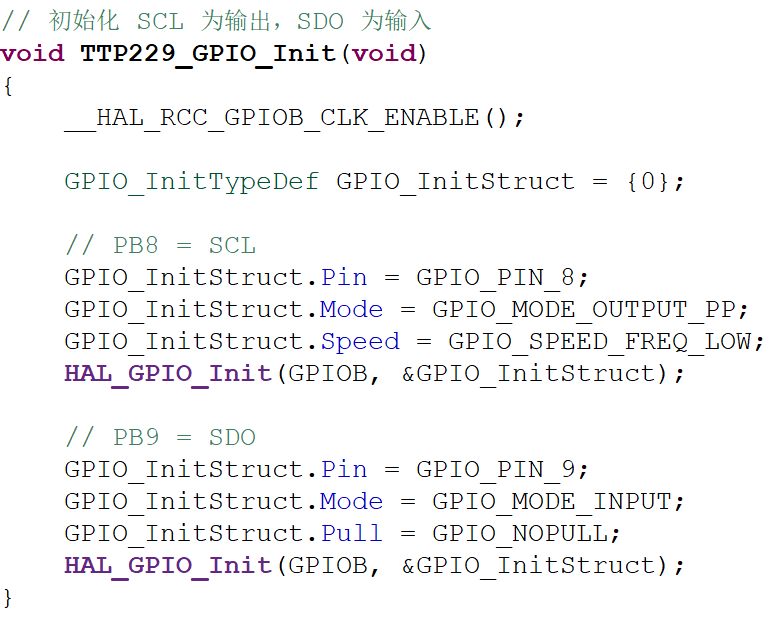
### ****1、GPIO輸入與控制****

#### （1）TTP229電容式鋼琴模組讀取

**原理**：TTP229-BSF模組通過SCL/SDO（I2C）以串行方式輸出觸摸按鍵信號。

**實現方式**：將PB8配置為**推挽輸出**（模擬時鐘SCL）；將PB9配置為**浮空輸入**（讀取數據位SDO）。

通過for迴圈模擬16次時鐘脈衝，逐位讀取數據並合成為16位鍵值，若鍵值非零，即代表有鍵被按下。



#### （2）普通按鍵輸入（PA0）

初始設置為GPIO輸入，後切換為外部中斷EXTI模式。

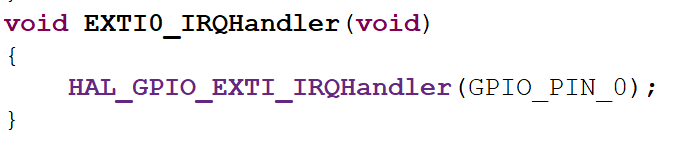
可識別**按下**（低電平）和**鬆開**（高電平）兩個狀態。

### ****2、外部中斷EXTI配置****

為提高系統回應性， 普通按鍵採用**中斷觸發方式**：使用HAL庫將PA0配置為EXTI中斷線（EXTI0）；在stm32l4xx\_it.c中定義EXTI0\_IRQHandler()並在main.c中實現HAL\_GPIO\_EXTI\_Callback()；可檢測按鍵按下與釋放，區別在於讀取HAL\_GPIO\_ReadPin()返回的是GPIO\_PIN\_RESET（低）或GPIO\_PIN\_SET（高）；

每次中斷觸發後可直接通過WiFi發送狀態位元組。





### **3、**WiFi網路通信****

使用官方提供的es\_wifi驅動庫；配置為**TCP Client**模式，連接本地電腦端指定IP與端口； 成功連接後，使用WIFI\_SendData()函數發送按鍵資訊。 **數據格式**：TTP229數據：2位元組（按鍵編碼）。普通按鍵：自定義首位元組標識（例如0xAA），第二位元組表示狀態（0x01表示按下，0x00表示鬆開）。所有數據以位元組流（byte stream）形式通過TCP socket發送。

**4、串口除錯與日誌輸出**

使用USART1配合printf實現串口輸出；用於輸出當前按鍵狀態、WiFi狀態、調試資訊等，有助於開發調試與系統驗證。

**5、程式結構優化處理**

主迴圈while(1)不再進行輪詢處理， 所有鍵值檢測操作均由中斷機制驅動完成（包括物理中斷和定時器中斷），數據發送也轉由事件觸發調用，提高系統回應效率與資源利用率，讓使用者有更佳的體驗 (QoE)。

1. **作法—Python(Techniques--Python)**
2. **STM32 輸入格式**

**從 STM32 接收到的格式為 16 位元的 bit array，其中第1到第12個 bit (LSB)控制鋼琴按鍵輸入，因為總共有12個按鍵，而第13個 bit 控制普通按鈕，舉例來說，如果 STM32 傳送過來的是 C 大調音階（CEG），則 bit array 為 0x0000000000010101，以 16 進位來表示是 0x0015。**

1. **Wifi網路通信**

利用 python內建的 socket 網路通訊模組，建立 TCP連線，並使用 client-server 架構。在電腦端建立 local server， STM32 作為 client，連接到 Python server，發送按鍵訊號，python 接收到訊號後，則會處理訊號，並根據訊號控制聲音合成器。

1. **FluidSynth 聲音合成器**

FluidSynth 是一個開源的軟體聲音合成器（real-time software synthesizer），能根據 MIDI 訊號 和 SoundFont 音色庫（.sf2 檔案） 產生高品質的樂器聲音。功能如下：

* 播放 MIDI 音樂檔或即時輸入的 MIDI 訊號。
* 載入並使用 SoundFont (.sf2) 音色庫。
* 即時產生聲音（real-time synthesis）。
* 支援多聲部（polyphony）、多音軌（multichannel）。
* 支援音量、音高、音色、彎音等 MIDI 控制。

FluidSynth 的基本架構如下：以播放小提琴音色的 C4 為例：

* MIDI 訊號輸入：按下 C4 音符、換樂器為小提琴。
* 查詢 SoundFont：從 .sf2 音色library找出 C4 對應的聲音 Sample。
* 即時合成音訊：使用 DSP將 Sample轉為音訊波形。
* 輸出聲音：將聲音播放出來。

**pyFluidSynth 是 Python 的一個模組，用來控制 FluidSynth，這個模組讓我們可以用簡單的 Python 程式碼達成類似 MIDI 鍵盤輸出的功能，而不需硬體音源。**

**FluidR3\_GM.sf2 library是一個廣泛使用的 SoundFont 2（.sf2） 格式音色庫，它依據 General MIDI 標準，內含了 128 種標準樂器音色與打擊樂器。FluidR3\_GM.sf2 包含了以下好處：**

* **包含多種常用樂器音色：鋼琴、吉他、小提琴、管樂器、打擊樂等。**
* **適用於 MIDI 音樂、數位合成與互動式音樂應用。**
* **開源且免費，可自由用於學術用途。**
* **是 FluidSynth 的官方推薦 SoundFont。**

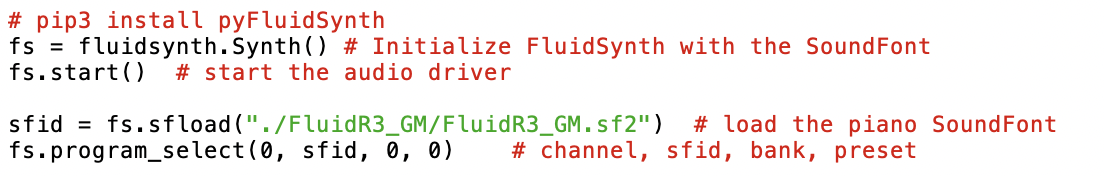
**在本專案中，FluidR3\_GM.sf2 負責提供鋼琴音色、吉他音色、小提琴音色、喇叭音色（透過按鈕切換），達成類似真實樂器的聲音輸出。**

1. **Python 實作**

**結合以上幾點，Python program 的流程控制如下：**

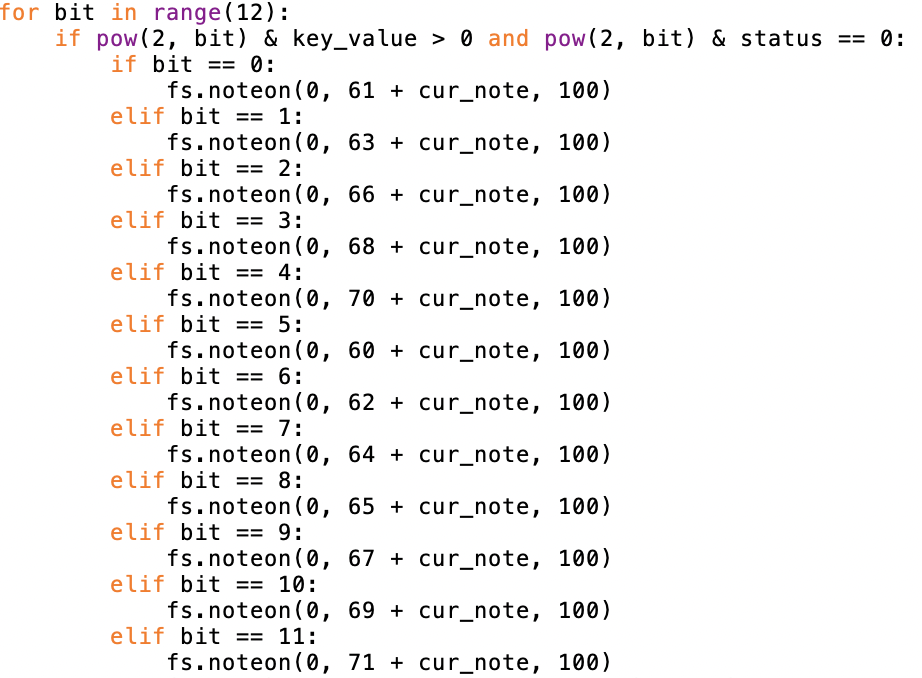
1. **Python 作為 Wifi Server，透過 TCP socket接收 STM32 的資料。**
2. **按照STM32對應的資料格式進行 decode。**
3. **控制 pyFluidSynth 模組，以 FluidR3\_GM.sf2 library 實現 real-time playback。**

**以下程式用來初始化pyFluidSynth 模組，並選擇樂器編號為0的鋼琴。**



**fs.program\_select(0, sfid, 0, 0) 透過最後一個參數可選擇樂器，這裡設為0（鋼琴）。**

**fs.noteon(0, 61 + cur\_note, 100) 第二個參數選擇音高，第三個參數控制音量，這裡選擇C4，音量100。**

**fs.noteoff(0, 61 + cur\_note) 停止目前聲音輸出。****三**

**程式有三個狀態變數：**

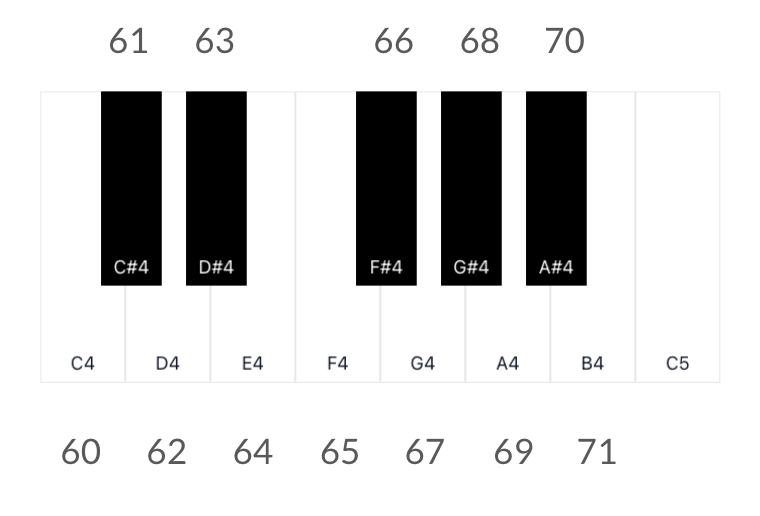
**status記住目前被觸摸的按鍵。**

**cur\_instruments 記住目前的樂器。**

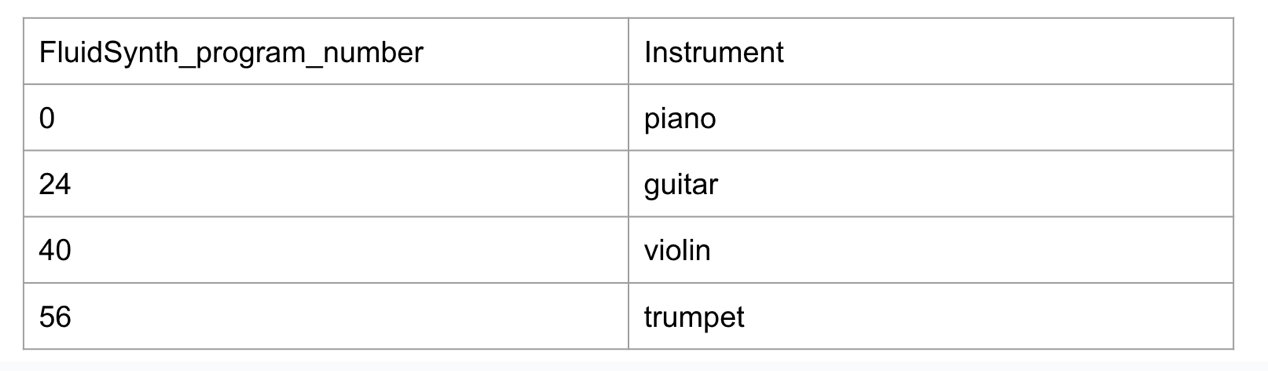
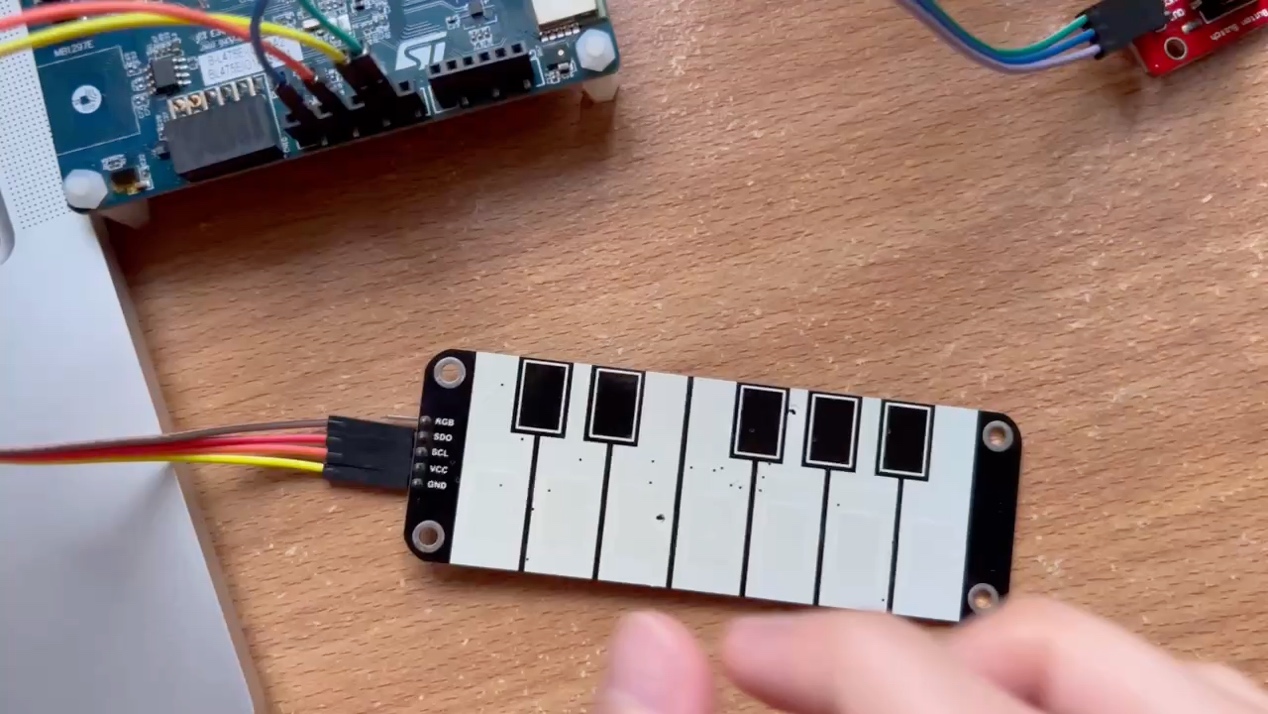
**cur\_note 記住目前的音高。**

**Python 實作出以下三個個功能：**

* **一般鋼琴按鍵輸入：利用12個bit來輸出對應的音階。**



* **普通按鈕輸入：只要接收到普通按鈕輸入（第13個bit），則程式控制 python 變換音色。本專案使用四種樂器，分別為鋼琴、吉他、小提琴、喇叭，只要按一次普通按鈕，就會循環變成下一種樂器。下圖為各樂器在 FluidR3\_GM.sf2 library 中的編號：**



* **特殊音高轉換：在短時間內，如果user 按了所有的白鍵，則音高升高八度 ; 如果user 按了所有的黑鍵，則音高降低八度。**

1. **實作（Demo）**

我們實作出一個功能完整、可獨立運作的鋼琴輸入系統，支援實體按鍵、音高控制，並具備音色切換功能。STM32 實作改成 interrupt 之後，latency 降低，performance 也變好。

1. **Demo1 - 用鋼琴的聲音演奏**

<https://drive.google.com/file/d/16ZgBjUYqB748SH20Cn4z6KcTeuiMq_Gc/view?usp=sharing>

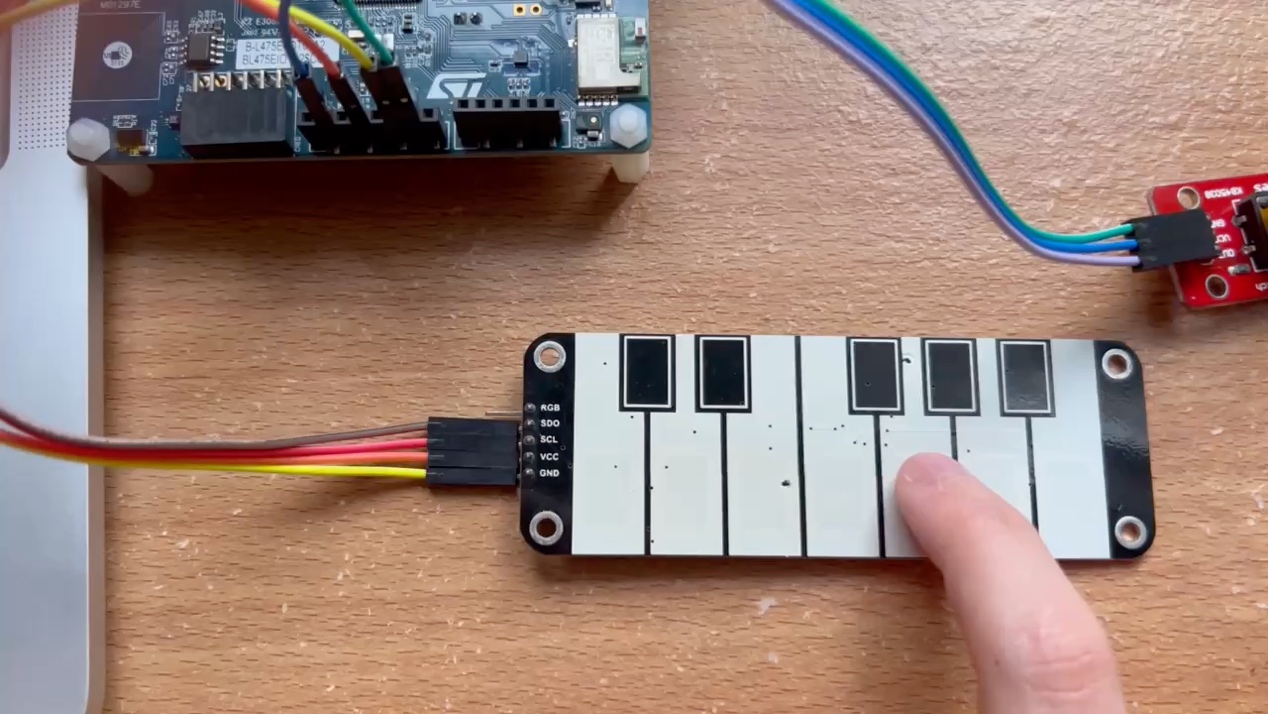
1. **Demo2 - 用喇叭的聲音演奏**

<https://drive.google.com/file/d/1-L4dd9nkjXrQ2tjeXhBq54hmAglAVwZy/view?usp=sharing>

1. **Demo3 - sound-switch**

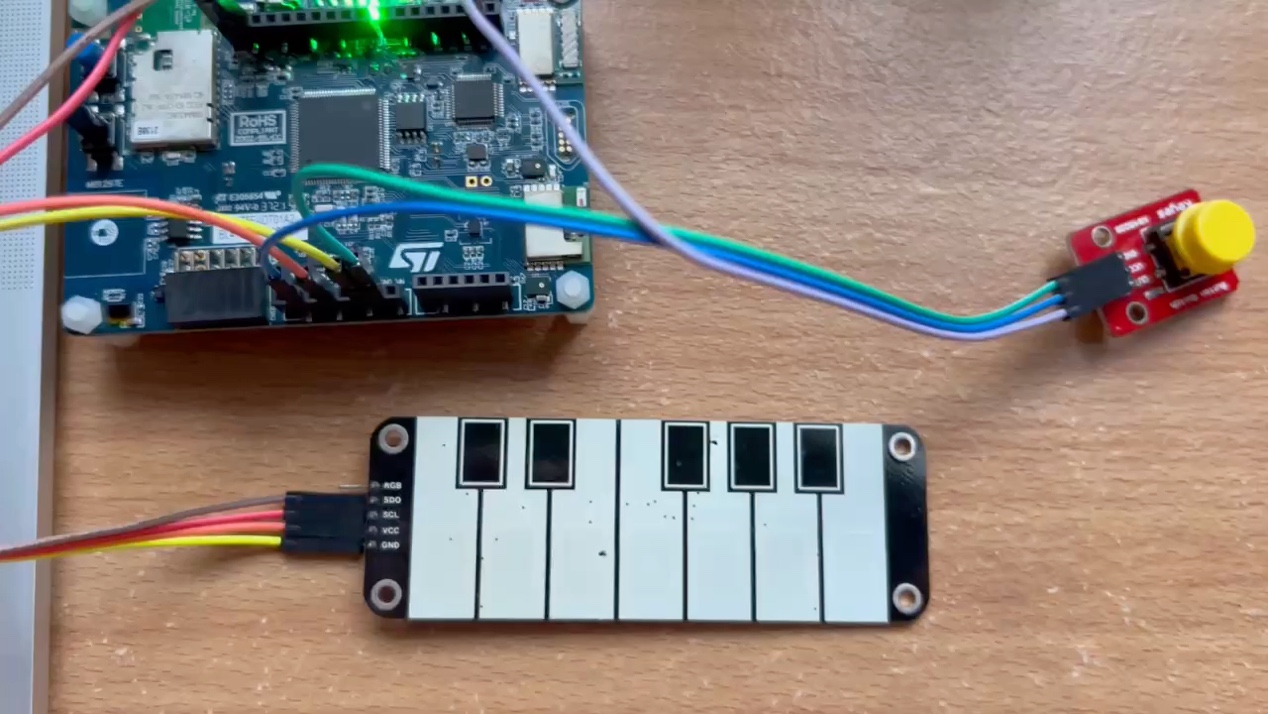
利用按鈕觸發音色的轉換，循環： 鋼琴->吉他->小提琴->喇叭。

<https://drive.google.com/file/d/1Kf6iW6z3dfWfC-Y_ROHzWsU9ogIqkmen/view?usp=sharing>

****

1. **Demo4 - tone-switch**

只要觸摸所有白鍵，音高就會升高八度。

****<https://drive.google.com/file/d/1izU3Jj2NNnbYq6NScLGoPSjJZVzSafxX/view?usp=sharing>

1. **Demo5- tone-switch**

只要觸摸所有黑鍵，音高就會降低八度。

****<https://drive.google.com/file/d/1Lgy6O1EArwtrYpgiLMWCJiZxRL0fguS7/view?usp=sharing>

1. **參考資料（References）**

**完美鋼琴:**

<https://play.google.com/store/apps/details?id=com.gamestar.perfectpiano&hl=zh_TW>

<https://apps.apple.com/tw/app/%E5%AE%8C%E7%BE%8E%E9%8B%BC%E7%90%B4/id942937409>

1. **其他連結 (Other Resources)**

* **Github**

<https://github.com/B07505045/touch_piano-final-project>

* **PPT**

<https://docs.google.com/presentation/d/1nPx8sj5PwT81z5MPljm1LiKZmTs3vkSe5fo1KD3GMvU/edit?usp=sharing>