

期末專題報告：觸摸鋼琴

王靖婷 R12921A13、吳璵汐 T13901203

一、 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。

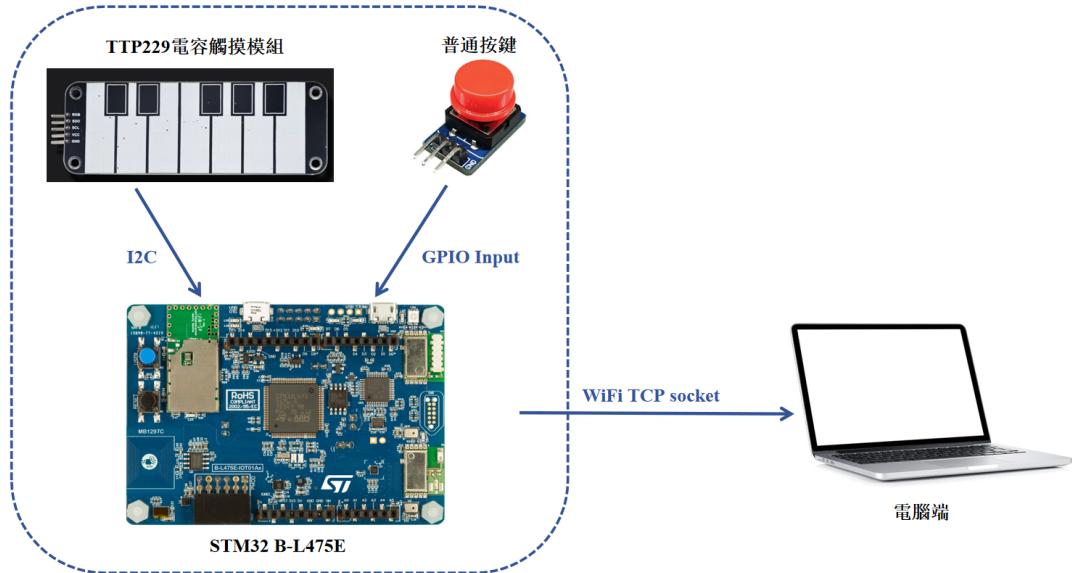
輸入介面為一組鋼琴鍵盤形狀的電容式觸控感測器，連接至 B-L475E-IOT01A 開發板，用以偵測對應各個音符的觸控訊號。當使用者觸碰按鍵時，開發板偵測對應的觸控訊號並處理，隨後透過 Wi-Fi client-server 架構 (TCP protocol) 將資料無線傳送至電腦。在 Host 端 (Mac 筆電)，一個以 Python 實作的 Server 負責接收這些訊號，並利用音色合成技術播放對應的音符聲音。

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。

二、 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。

```

// 初始化 SCL 为输出, SDO 为输入
void TTP229_GPIO_Init(void)
{
    __HAL_RCC_GPIOB_CLK_ENABLE();

    GPIO_InitTypeDef GPIO_InitStruct = {0};

    // PB8 = SCL
    GPIO_InitStruct.Pin = GPIO_PIN_8;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    // PB9 = SDO
    GPIO_InitStruct.Pin = GPIO_PIN_9;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}

uint16_t TTP229_ReadKeys(void)
{
    uint16_t data = 0;

    for (int i = 0; i < 16; i++)
    {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET); // CLK low
        HAL_Delay(1);
        if (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9) == GPIO_PIN_RESET)
        {
            data |= (1 << i);
        }
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET); // CLK high
        HAL_Delay(1);
    }
    return data;
}

```

(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。

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    switch (GPIO_Pin)
    {
        case GPIO_PIN_0:
        {
            GPIO_PinState key_state = HAL_GPIO_ReadPin(BUTTON1_GPIO_PORT, BUTTON1_PIN);
            // 控制 LED2
            if (key_state == GPIO_PIN_RESET)
            {
                BSP_LED_On(LED2); // 按下，点亮
            }
            else
            {
                BSP_LED_Off(LED2); // 松开，熄灭
            }
            uint8_t tx_buf[2];
            tx_buf[0] = 0xAA; // 自定义标识符: PA0 按键事件
            tx_buf[1] = (key_state == GPIO_PIN_RESET) ? 0x01 : 0x00;

            uint16_t sentLen;
            if (Socket >= 0)
            {
                WIFI_SendData(Socket, tx_buf, sizeof(tx_buf), &sentLen, WIFI_WRITE_TIMEOUT);
            }

            break;
        }
    }
}

void EXTI0_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

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)。

四、作法—Python (Techniques--Python)

1、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 0x00000000000010101，以 16 進位來表示是 0x0015。

2、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。

3、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。

4、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。

```
# pip3 install pyFluidSynth
fs = fluidsynth.Synth() # Initialize FluidSynth with the SoundFont
fs.start() # start the audio driver

sfid = fs.sfload("./FluidR3_GM/FluidR3_GM.sf2") # load the piano SoundFont
fs.program_select(0, sfid, 0, 0) # channel, sfid, bank, preset
```

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。

```
for bit in range(12):
    if pow(2, bit) & key_value > 0 and pow(2, bit) & status == 0:
        if bit == 0:
            fs.noteon(0, 61 + cur_note, 100)
        elif bit == 1:
            fs.noteon(0, 63 + cur_note, 100)
        elif bit == 2:
            fs.noteon(0, 66 + cur_note, 100)
        elif bit == 3:
            fs.noteon(0, 68 + cur_note, 100)
        elif bit == 4:
            fs.noteon(0, 70 + cur_note, 100)
        elif bit == 5:
            fs.noteon(0, 60 + cur_note, 100)
        elif bit == 6:
            fs.noteon(0, 62 + cur_note, 100)
        elif bit == 7:
            fs.noteon(0, 64 + cur_note, 100)
        elif bit == 8:
            fs.noteon(0, 65 + cur_note, 100)
        elif bit == 9:
            fs.noteon(0, 67 + cur_note, 100)
        elif bit == 10:
            fs.noteon(0, 69 + cur_note, 100)
        elif bit == 11:
            fs.noteon(0, 71 + cur_note, 100)
```

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：

FluidSynth_program_number	Instrument
0	piano
24	guitar
40	violin
56	trumpet

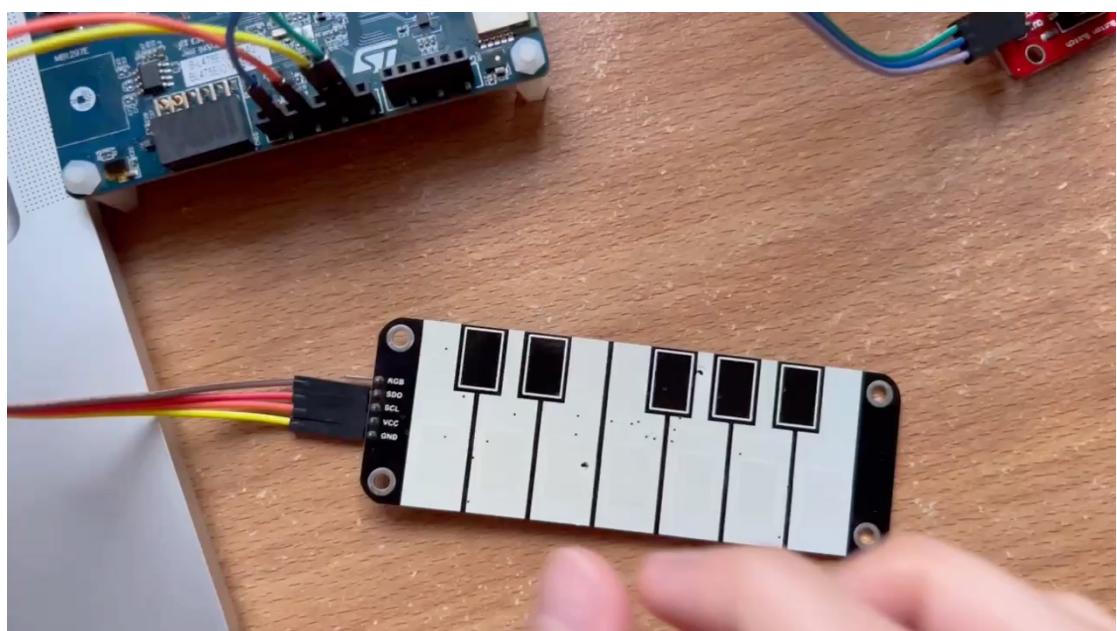
- 特殊音高轉換：在短時間內，如果 user 按了所有的白鍵，則音高升高八度；
如果 user 按了所有的黑鍵，則音高降低八度。

五、 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



2、 Demo2 - 用喇叭的聲音演奏

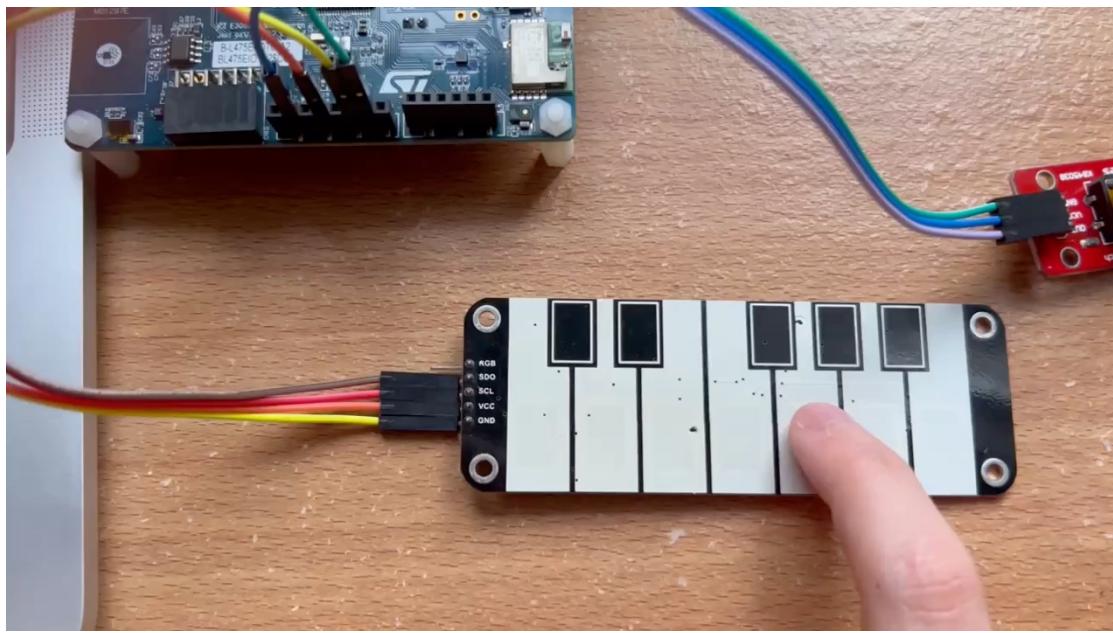
<https://drive.google.com/file/d/1-L4dd9nkjXrQ2tjeXhBq54hmAglAVwZy/view?usp=sharing>



3、 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。

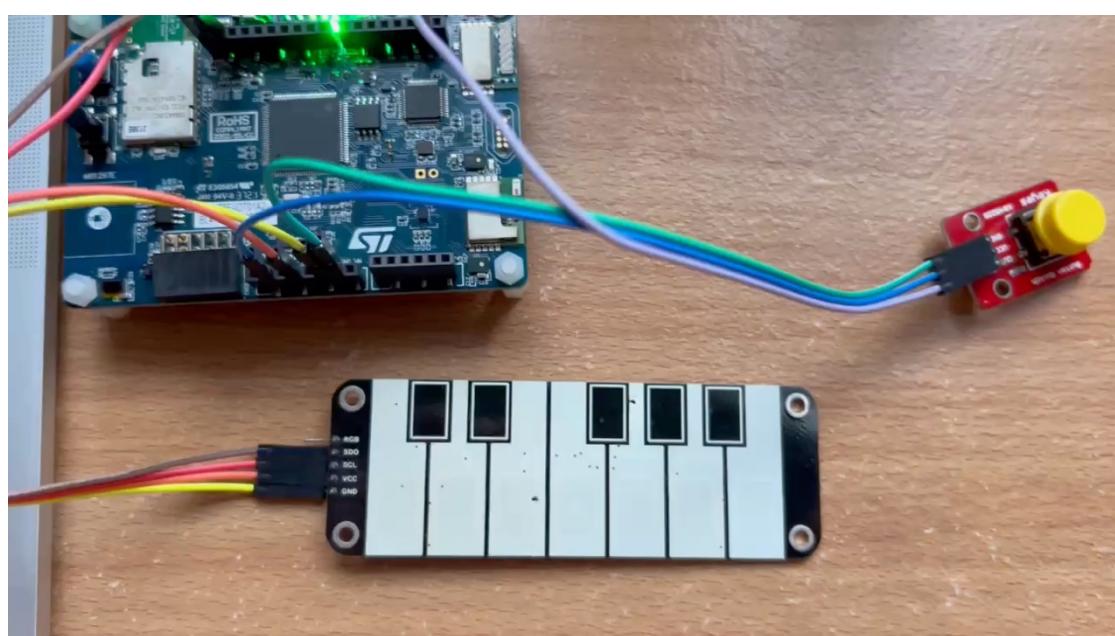
https://drive.google.com/file/d/1Kf6iW6z3dfWfC-Y_ROHzWsU9ogIqkmen/view?usp=sharing



4、 Demo4 - tone-switch

只要觸摸所有白鍵，音高就會升高八度。

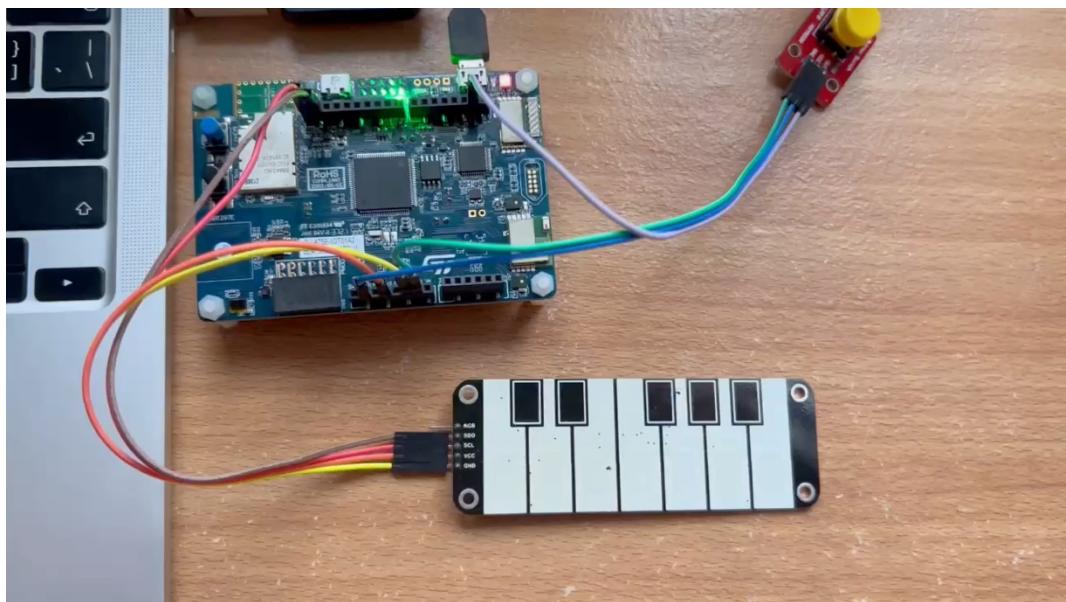
<https://drive.google.com/file/d/1izU3Jj2NNnbYq6NScLGoPSjJZVzSafxX/view?usp=sharing>



5、 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>



六、 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>

七、 其他連結 (Other Resources)

- Github

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

- PPT

<https://docs.google.com/presentation/d/1nPx8sj5PwT81z5MP1jm1LiKZmTs3vkSe5fo1KD3GMvU/edit?usp=sharing>