微算機期末專題

組員：張娟鳴(E94086107)、侯秉逸(E24094172)、莫寶琳(F64081169)、陳詠君(H54084078)

1. 音樂播放器

a. 系統功能與原理說明

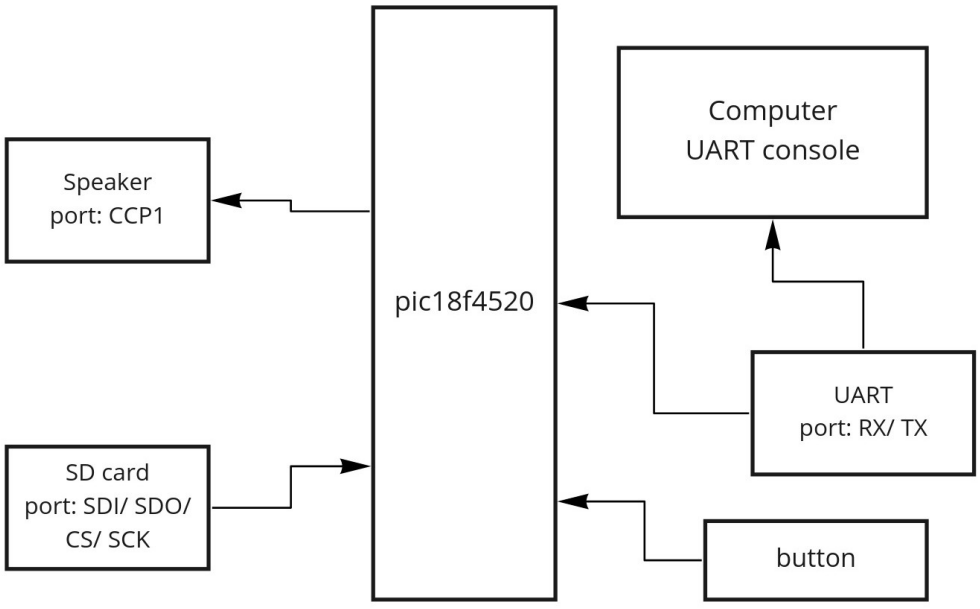
系統功能： 讀取SD卡裡面的檔案並轉換成類比訊號播出
原理： 在SPI模式下讀取SD卡內容，讀到後透過UART下達指令，進行SD卡讀寫函示庫初始化與讀寫；之後將文件轉成PWM載波訊號，再經低通濾波電路除去高頻雜訊，利用3.5mm音源線將訊號輸出至耳機。

b. 系統使用環境及對象

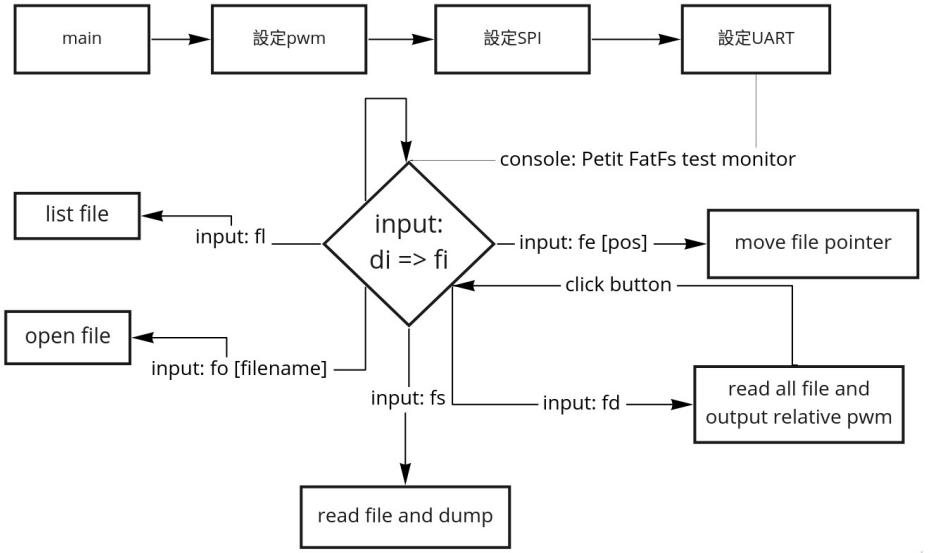
- 系統使用環境：MPLAB 5.20
- 語言：C
- 編譯器：xc8 2.32
- 燒錄器：Picket 4
- 晶片：PIC18F4520

c. 系統完整架構圖、流程圖、電路、設計

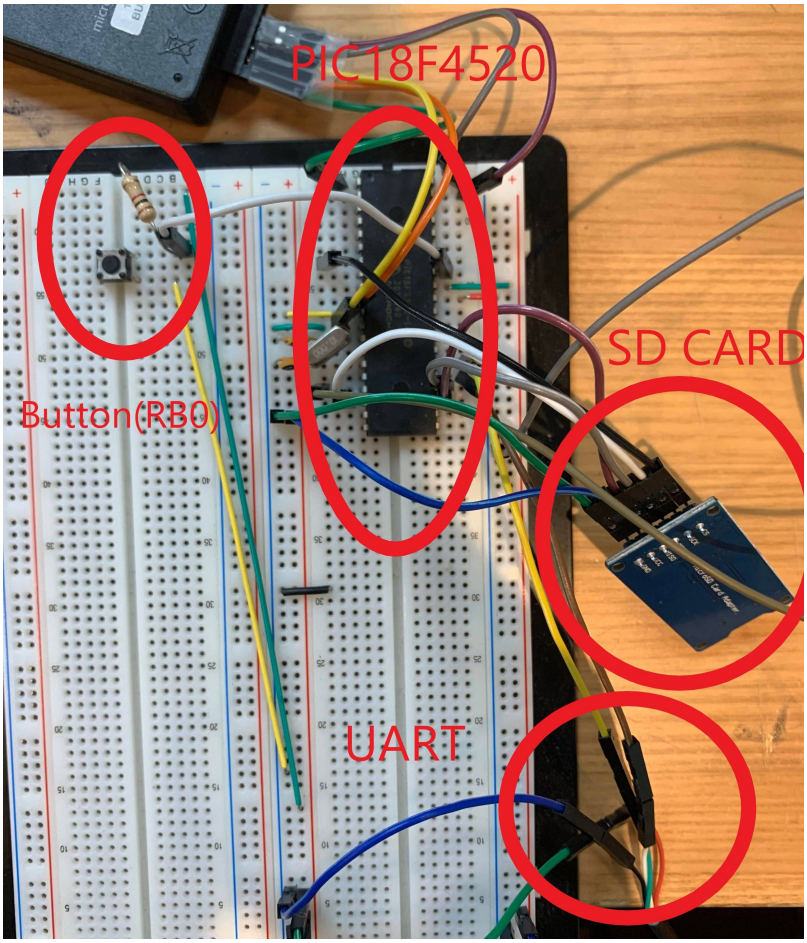
系統完整架構圖

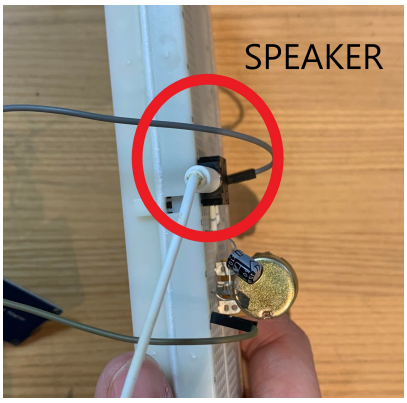


流程圖


main
設定pwm
設定SPI
設定UART
console: Petit FatFs test monitor
list file
input: fl
input:
di => fi
input: fe [pos]
move file pointer
click button
open file
input: fo [filename]
input: fs
input: fd
read all file and
output relative pwm
read file and dump
miro

電路


PIC18F4520
SD CARD
Button(RB0)
UART



設計

作品主要流程：

1. 初始化各個硬體
2. console印出 "Petit FatFs test monitor"
3. 初始化disk、掛載SD卡
4. 開啟檔案
5. 播放音樂
6. 如果播放中途按按鈕，會停止播放
7. 再輸入一次播放指令會繼續播放
8. 也可以重新播放，如果重新開檔案則會從頭播放

d. 系統開發工具、材料及技術

系統開發工具、材料

pic18f4520, Micro SD TF Card Memory Shield Module, 3.5mm耳機接頭, 石英振盪器, 電阻與電容, button

系統開發技術

- 單元項目：SPI、UART、Interrupt、PWM
- 進階項目：SD卡、低通濾波器、電子硬體元件（石英振盪器）

SPI介面+SD卡

- SD卡內容之讀取是利用SPI介面完成
- Fat32讀檔 reference
  di: `disk_initialize()`
  判斷是否為合法的sd card並初始化SD卡讀取機制。
  fi: `pf_mount(&fs)`
  判斷disk 是否被初始化，判斷sector 0(Boot sector) 判斷是否為合法file格式，計算file 的size，紀錄文件data的起始位置，初始化文件完成。
  fo: `pf_open(ptr)`
  判斷路徑是否合法，得到file開始的cluster(即基本的儲存單位)以及size，回傳開啟檔案成功。
  fd:

一開始將init_flag設為0，即第一次會跳過Fat32格式的前面44bytes，使用while迴圈連續讀值，將讀到的數值傳給pwm的duty cycle使用，每讀完一個Byte,delay 40us讓PWM做轉換。

PWM

- 將SD卡讀到的8 bit wave訊號轉為數字，做為pwm的duty cycle輸出；並根據音頻檔案的採樣率(我們採用16000Hz的音頻文件)，每過16000us更新一次duty cycle，更新週期越短，播放速度會變快，可以達成快轉的效果。
  - CCP1/RC2作為輸出： `TRISC = 0; LATC = 0;`
  - period set 62.5us `PR2 = 63;`
  - Timer2 Prescaler = 1:1 `T2CON=0x04;`
  - PWM mode configuration `CCP1CON = 0X3C;`

UART

- UART接serial port，讓使用者透過下指令方式與SD卡互動，可以進行更換檔案或讀取其他檔的hex碼等功能。
  `_XTAL_FREQ = 40MHz baudrate : 19200`

Interrupt

- 使用按鈕（接在RB0）作為中斷播放音樂的訊號，並紀錄暫停的地方以便下次播放。

電子硬體元件

- 我們外接石英振盪器作為系統oscillator，並把晶片振盪器來源設定為HSPLL。
  `#pragma config OSC = HSPLL`

低通濾波器

- 濾除電路的高頻雜訊，使音源輸出較為乾淨。

e. 周邊接口或library及API說明

周邊接口

- Micro SD TF Card Memory Shield Module

Library

- 我們使用以下GitHub repository的SD卡部份跟PWM部份，並添加自己想要的功能上去
  - Repo reference
  - 這個Repo主要是一些和PIC18F4520相關的應用，我們應用其中SD卡的部分

API說明

- 使用UART進行SD卡存取時的互動指令
  - 輸入di: 初始化disk
  - 輸入fi: 掛載SD卡
  - 輸入fl: 列出目錄

    - 輸入fo: 開啟檔案
    - 輸入fd: 播放音樂
    - 輸入fs: 單純顯示檔案內容，一次128 byte
    - 按下按鈕: 暫停音樂，這時如果:
        - 輸入fd: 從暫停處繼續播放
        - 輸入fo: 開啟另外檔案，先前播放紀錄會不見

f. 遇到的困難及如何解決

1. 問題：找不到合適的SD card library

描述： 一開始在找SD card的library時，只找到需要CCS compiler的library，跟我們理想上的需求不太一樣。
解決方法： 最後沒使用I2C介面，而是在GitHub找到了使用SPI介面搭配UART來讀取SD卡的library，缺點是SCK和SCL用到相同pin腳，我們沒辦法同時做要用到SPI和I2C介面的應用。

2. 問題：SPI和CCP的PWM模式接合問題

描述： 我們先做好SD卡讀檔的部份(SPI)再把讀到的值作為PWM的duty cycle，當我們嘗試把兩組程式合在一起的時候，卻一直無法顯示SD卡。
解決方法： 發現其實和硬體的初始化順序有一點關係；我們原先是先處理SPI相關的初始化，再做PWM和UART的初始化，這樣子UART能順利運作，但會找不到SD卡。
網路上有人分享用其他微處理器做的音樂播放器是先處理PWM的初始化再處理SPI，我們便依樣化葫蘆，結果真的能解決問題。

3. 問題：音樂不還原

描述： 成功組合SPI和PWM後，我們接上耳機播放，卻只聽到斷斷續續的聲音。
解決方法：

- 一開始我們以為是耳機本身阻抗太大，晶片產生的PWM訊號推不動，便修改RC濾波電路的阻值，也拿其他耳機測試；結果發現阻值改變時，只有辦法改變低通濾波器過濾高頻雜訊的效果，讓我們確定我們的輸出有確實。
- 後來我們查到可能是載波頻率出了問題；當載波頻率越高，聲音聽起來才越連續，我們想透過調整PWM週期以調高載波頻率，然而PIC18F4520的頻率最高只到40MHz，產生出來的還是不甚理想。
- 我們嘗試超頻將石英震盪器從10MHz換成16MHZ，使Fosc從40MHz變成48MHz；發現只要超過處理器最高的頻率限制，馬上出現非預期的行為，因此PWM載波頻率無法再提升。
- 最後靠調整更新頻率讓速度接近原曲，產生一個音調變高但節拍相同的音樂播放器，仍然會聽到聲音有一點斷斷續續（效果已比調整前還佳）。

---

2. Real Timer Clock Using RTC chip

- 因為前一個項目沒有達到預期結果，所以我們多做一組

a. 系統功能與原理說明

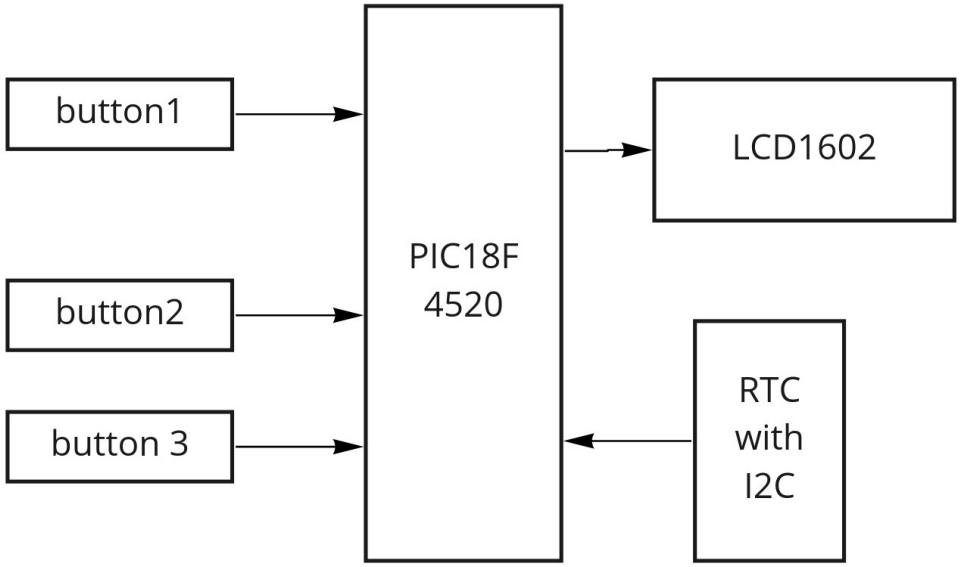
系統功能： 在LCD上顯示現在的年月日與時間
原理： 利用RTC 模組可以產生精確時間的特點，以及帶有電池供電以記憶並持續計時的特性來持續提供正確時間資料供LCD顯示

b. 系統使用環境及對象

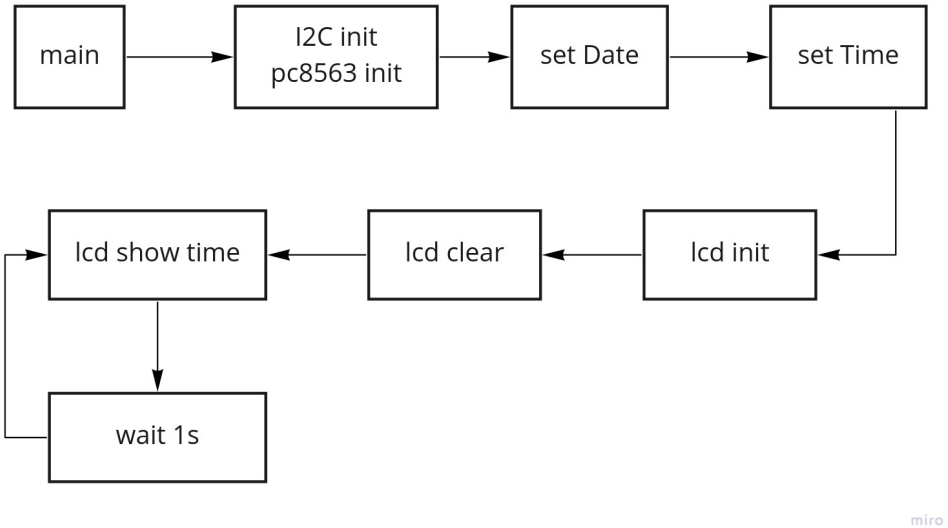
- 系統使用環境：MPLAB 5.20
- 語言：C
- 編譯器：xc8 2.32
- 燒錄器：Picket 4
- 晶片：PIC18F4520

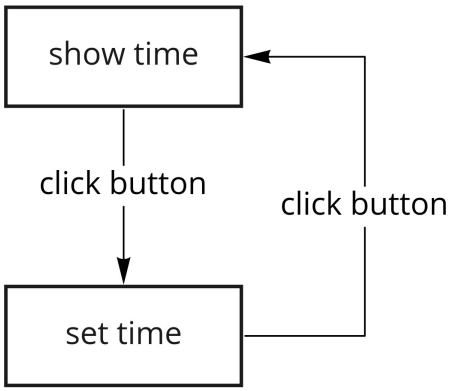
c. 系統完整架構圖、流程圖、電路、設計

系統完整架構圖



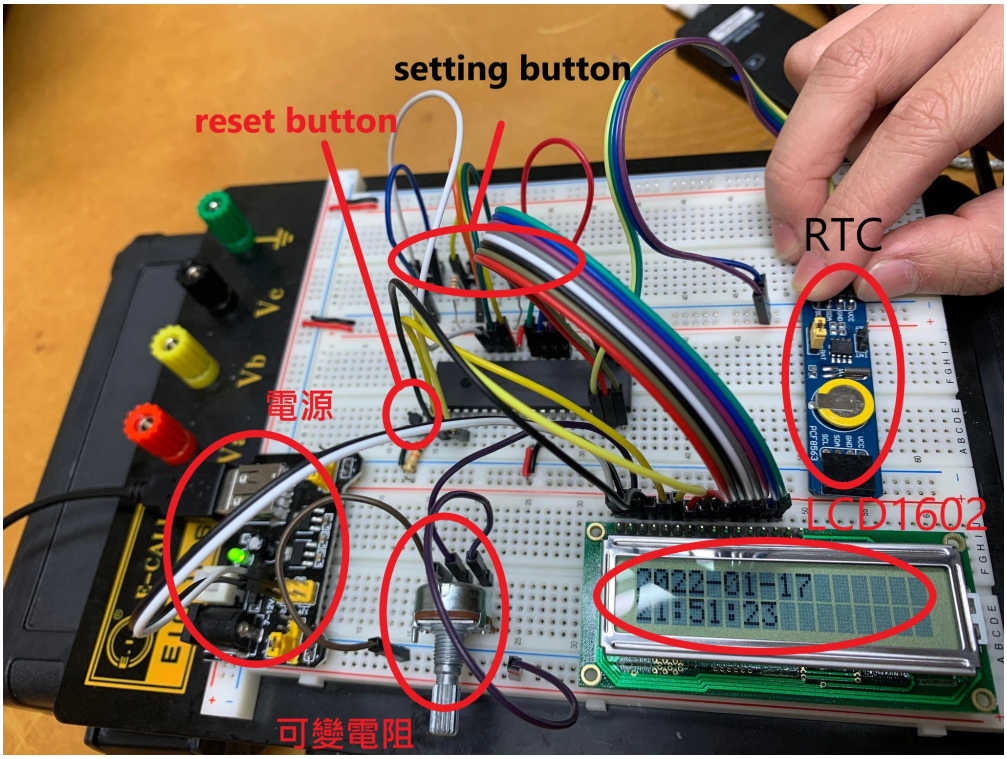
流程圖





- interrupt

電路



設計

1. 開啟電源，LCD1602上顯示時間
2. 按下中間的button，觸發interrupt，進入調整時間模式
3. 左右兩個buttom可以做年份的加減調整，調整好後按下中間的按鈕
4. 進入下一個選項的調整，調整完一輪後即可退出模式
5. 使用已經調整好的時鐘

d. 系統開發工具、材料及技術

系統開發工具、材料

PIC18F4520, PCF8563(RTC module), LCD1602(Liquid-Crystal displayer), button, 可變電阻

系統開發技術

- 單元項目舉例：Interrupt

- 進階項目舉例：I2C介面, 電子硬體元件（LCD monitor、Real time clock）

Interrupt

- 按鈕
  使用者可以透過按壓middle button 修改RTC模組的時間資料；透過多次按壓middle button依序切換修改的資料項目，並使用up及down button修改數值。
  按下button 時，觸發INT0 external interrupt，此時將該interrupt之enable bit設為not enable，並使用polled I/O做偵測，讓使用者透過按壓middle button依序切換修改項目。
  在最後一項修改完畢，離開interrupt servise routine之前，將INT0之enable bit設為enable，讓使用者之後能再次調整。
  調整數值時，up及down button為polled I/O，使用者按下按鈕時，可以在未超出該數值設定範圍時增加/減少數值大小。

I2C介面

- PCF8563(RTC chip)
  日期時間讀取：利用I2C輸出裝置的address選擇PCF8563裝置，再輸出該裝置register的位址；選擇想要PCF8563回傳的資料項目，並等待其回傳，接收該筆資料。
  日期時間設定：選擇該I2C裝置，傳送欲寫入項目相對應的register address，並傳送新的數值，以調整RTC之時間及日期。

電子硬體元件

- LCD1602液晶顯示螢幕
  利用LCD1602上方之parallel數據傳送腳位，並將RTC模組所讀入的數據經過適當轉換(BCD轉為數值)，再根據固定格式(yyyy-mm-dd及hh:mm:ss)顯示於LCD1602中。
- Real time Clock
  已經寫在I2C介面裡面，是時間的資料來源。
- 可變電阻
  調整LCD的解析度。

e. 周邊接口或library及API說明

周邊接口

1602LCD顯示螢幕、RTC的I2C介面、INT0 external interrupt

Library

- Repo reference1
  利用上列來源所提供之I2C_PCF8563函示庫，透過I2C介面使單片機與PCF8563溝通。該原始函示庫在取得時間、日期資料時是直接顯示於UART介面上之Serial port，將其修改為call by referene以在取得資料時修改程式中的資料變數。
- Repo reference2
  來源之Library為I2C介面之LCD1602函示庫，我們在實作時修改成parallel傳輸之LCD1602函示庫，利用該介面進行單片機與顯示螢幕的溝通，並建立多個method，以避免在控制LCD1602時直接撰寫command data之傳輸，減少開發時出現錯誤的機會，並可用於未來之專案。

API說明

- LCD1602之第一行顯示`yyyy-mm-dd`，第二行顯示`hh:mm:ss`，以傳達時間及日期。
- 利用mode button進入時間、日期調整模式，此時可以看到游標於年份欄位閃爍，表示正在調整年分(2021~2023)，並透過按壓mode button依序調整月份、日期、時、分、秒。
- 當使用者在調整時，透過按壓up/down button以進行數值之增減，調整完畢則按下mode button調整下一項目。
- 當全部調整完畢後，會回到顯示時間模式；使用者隨時可再次調整各項數值。

f. 遇到的困難及如何解決

1. 問題：程式燒不進微處理機

描述： 我們組有買供電並接上，打算將程式碼燒入晶片；設定完power on reset後如果關掉電源再打開，不是預期行為
解決方法： 我們後來有使用手動reset button來代替

3. 實際組員之分工項目

- 主題發想：張娟鳴、侯秉逸、莫寶琳、陳詠君
- 資料、函式庫藍本搜尋：張娟鳴、侯秉逸、莫寶琳、陳詠君
- 硬體設計及採購：侯秉逸
- 需求及程式設計：張娟鳴、侯秉逸、莫寶琳、陳詠君
- 功能測試與驗證：張娟鳴、侯秉逸、莫寶琳、陳詠君
- 文件撰寫：張娟鳴、侯秉逸、莫寶琳、陳詠君
- 影片錄製、剪輯：侯秉逸、莫寶琳、陳詠君

4. 程式碼與影片連結

原始碼下載連結
影片連結