Final Project Report

106062314 蔡政諺

- **Goal**

  數位鬧鐘

- **Components**

  1. LCD

     做為鬧鐘的螢幕，顯示時間、日期、星期等資訊，也可以即時顯示環境亮度、環境噪音音量。

  2. Buzzer (DAC)

     鬧鐘的鈴聲。

  3. Photo-resistor (Sensor)

     用來感應環境光，若光線變暗就會調暗螢幕亮度，反之則調亮螢幕亮度。

  4. Microphone (Sensor)

     用來感應環境噪音，若太吵會增強鬧鐘鈴聲，反之則減弱鈴聲。

  5. Keypad (Sensor)

     用來設定鬧鐘時間，或是關掉鬧鐘。

  6. DS1302 (RTC)

     提供時間、日期、星期等資訊。

- **Code Explanation**

  這次實作有用到 DS1302 的函式庫(DS1302.cpp、DS1302.h)。第一次 compile 要在 setup()將星期、時間、日期設定好。後續幾次就可以將這段 code 註解掉。

```
rtc.writeProtect(false);
rtc.setDOW(WEDNESDAY);
rtc.setTime(13, 30, 0);
rtc.setDate(15, 1, 2020);
```

  這次 project 總共定義了 3 個 state。CLOCK 是時鐘模式，會顯示現在時間，SETTING 是設定模式，給使用者設定鬧鐘，RINGING 是鬧鐘響的 state。

```
#define CLOCK 0
#define SETTING 1
#define RINGING 2
```

這次我用 FreeRTOS 實作，共設定 3 個 task。ControlTask 負責處理所有的 sensor/input ，DisplayTask 負責 LCD 的顯示 ，BuzzerTask 負責控制蜂鳴器的音高。

```
xTaskCreate(ControlTask, (const portCHAR *)"CONTROL", 110, NULL, 1, NULL);
xTaskCreate(DisplayTask, (const portCHAR *)"DISPLAY", 120, NULL, 1, NULL);
xTaskCreate(BuzzerTask, (const portCHAR *)"BUZZER", 45, NULL, 1, NULL);
```

在 ControlTask 裡面，會讀取 Photo resistor 和 Microphone 的數值，Keypad 會根據 state 不同，做不同的事。在 CLOCK 時，會將 RTC 算出來的時間放進全域變數中，此外如果按下井字鍵，就會進入設定模式；SETTING 中共有三頁，第一頁是顯示現在設定的鬧鐘時間，使用者可以透過數字鍵去修改，第二頁是即時偵測環境噪音的音量，第三頁是即時偵測環境亮度；RINGING 時，則可以按井字鍵，關掉鬧鐘的鈴聲。

```
void ControlTask() {
  /* Handle all inputs(sensors) */
  for (;;) {
    vTaskDelay(10);
    /* Photo Resistor */
    brightness = analogRead(A0) / 100;
    if (brightness >= 5) lcd.backlight();
    else lcd.noBacklight();
    /* Microphone */
    volume = analogRead(microphone) / 12;
    Serial.println(analogRead(microphone));
    /* Keypad */
    key = myKeypad.getKey();
    switch (state) {
      case CLOCK:
        strcpy(WeekdayStr, rtc.getDOWStr());
        strcpy(DateStr, rtc.getDateStr());
        strcpy(TimeStr, rtc.getTimeStr());
        if (key == '#') {
          state = SETTING;
        }
        if (strcmp(TimeStr, alarmTimeStr) == 0 || key == '*') {
          state = RINGING;
          alarmMillis = millis();
        }
        break;
```

在 DisplayTask 裡面，也會根據 state 的不同，顯示不同的資訊。在 CLOCK 時，顯示 RTC 提供的時間等資訊；在 SETTING 時，如果是第一頁會顯示當前的鬧鐘時間，第二頁顯示環境音量，第三頁顯示環境亮度。

```
void DisplayTask() {
  for (;;) {
    vTaskDelay(10);
    switch (state) {
      case CLOCK:
        lcd.clear();
        lcd.setCursor(0, 1);
        lcd.print(DateStr);
        lcd.setCursor(13, 1);
        lcd.print(WeekdayStr);
        lcd.setCursor(4, 0);
        lcd.print(TimeStr);
        break;
      case SETTING:
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(msg);
        lcd.setCursor(0, 1);
        if (page == 0) lcd.print(alarmTimeStr);
        else if (page == 1) {
          for (int i = 0; i < 10; i++) {
            if (i < volume) lcd.write(0);
            else lcd.write(1);
          }
        }
```

在 BuzzerTask 裡面會用到鬧鐘鈴聲，我將音樂寫在全域變數 music 裡，用 millis 來控制一個一個音輪流播放。

```
const int music[16] = {C, D, E, F, G, G, A, F, E, E, D, D, C, C, C, C};
```

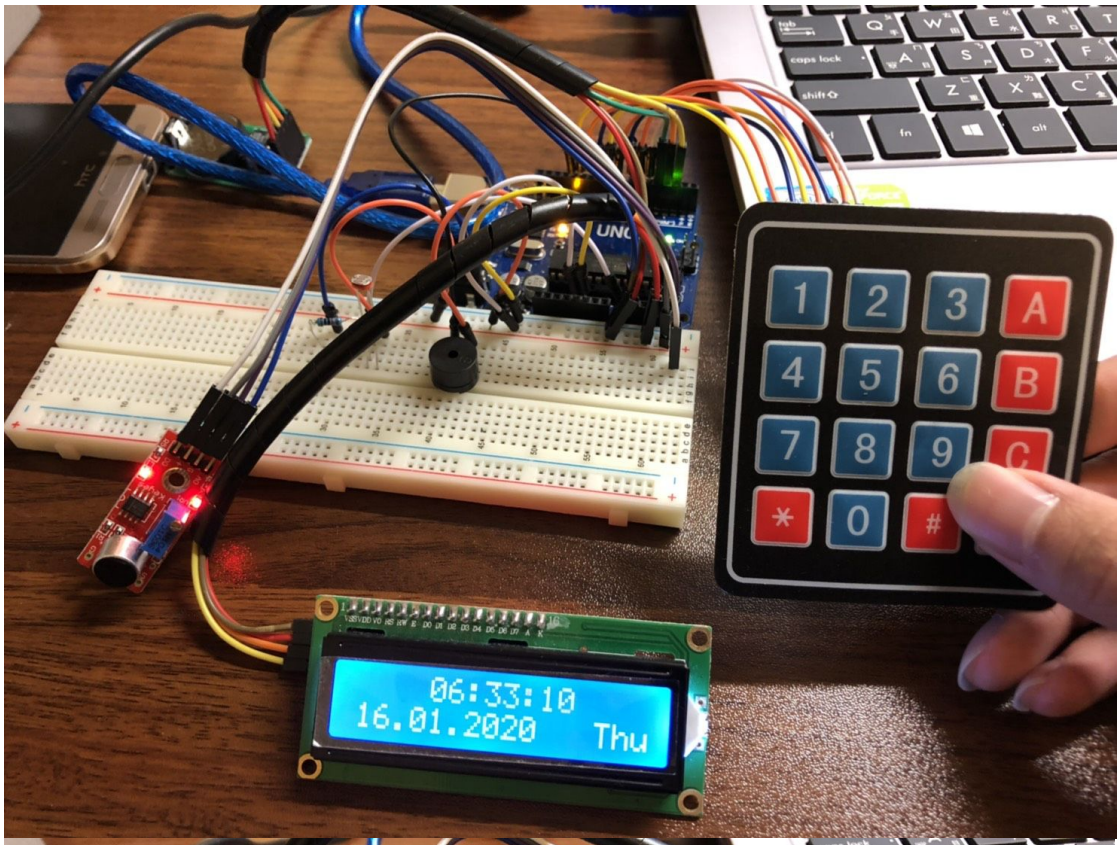
在 BuzzerTask 裡面，則是控制蜂鳴器的頻率。裡面會根據偵測到的音量大小，決定鈴聲的音高。如果環境噪音較小聲，就會低八度播放；但太吵就會高八度播放。

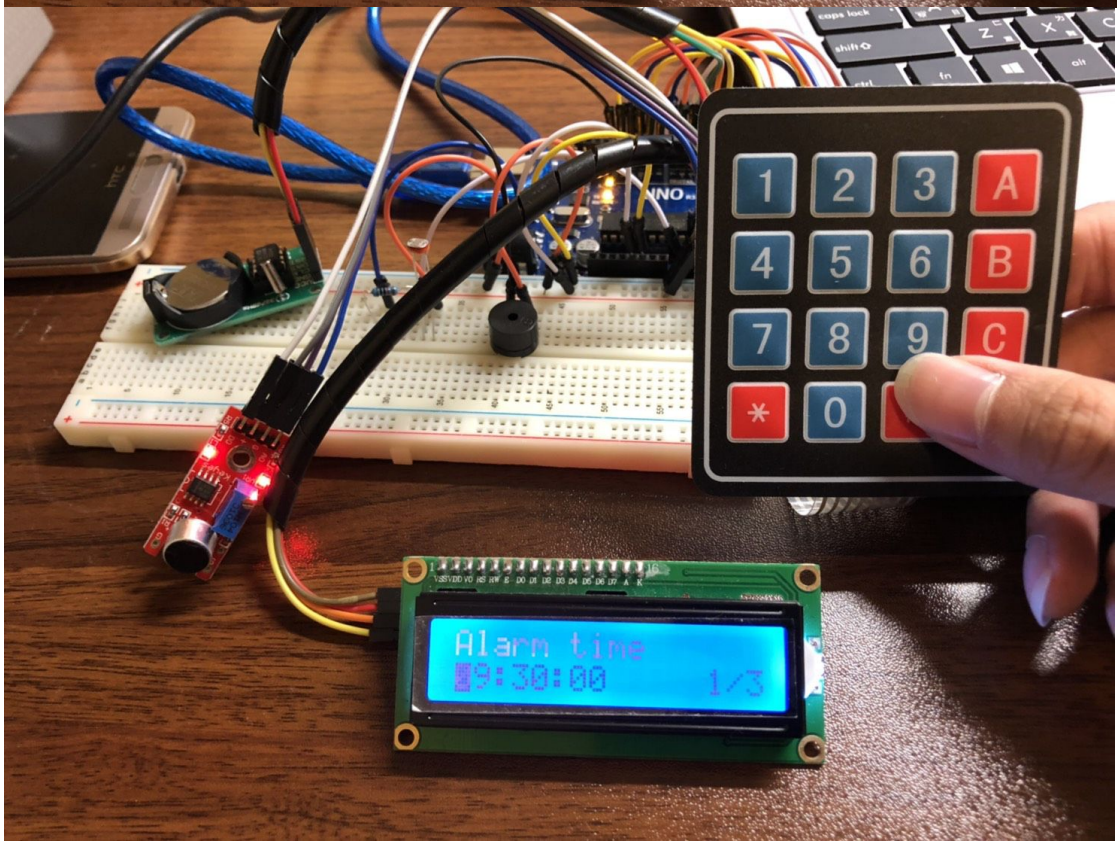
```
void BuzzerTask() {
  for (;;) {
    vTaskDelay(10);
    noTone(buzzer);
    i = 0;
    if(volume > 5) {
      while(i < 16 && state == RINGING){
        if (millis() > alarmMillis + 500){
          tone(buzzer, music[i++]);
          alarmMillis = millis();
        }
      }
    } else {
      while(i < 16 && state == RINGING){
        if (millis() > alarmMillis + 500){
          tone(buzzer, music[i++] / 2);
          alarmMillis = millis();
```

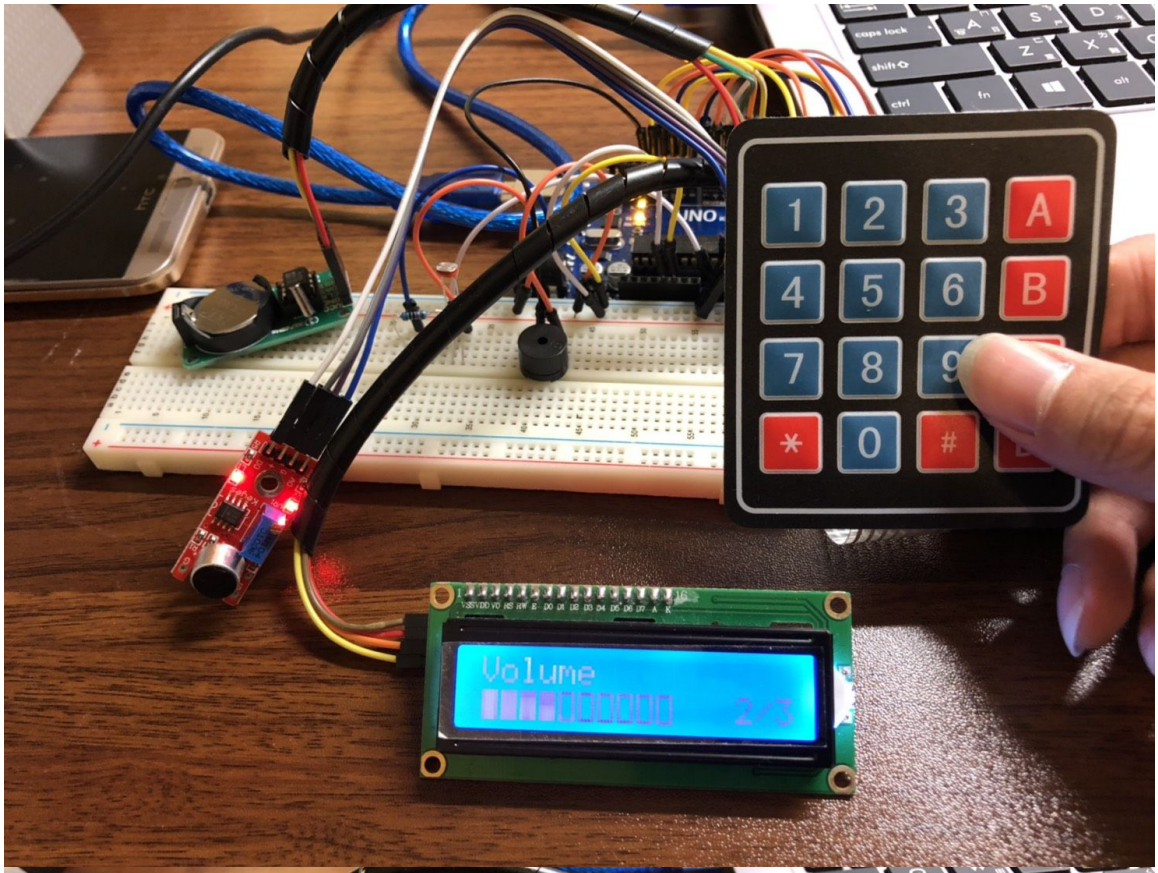
- **Encountered Difficulties**

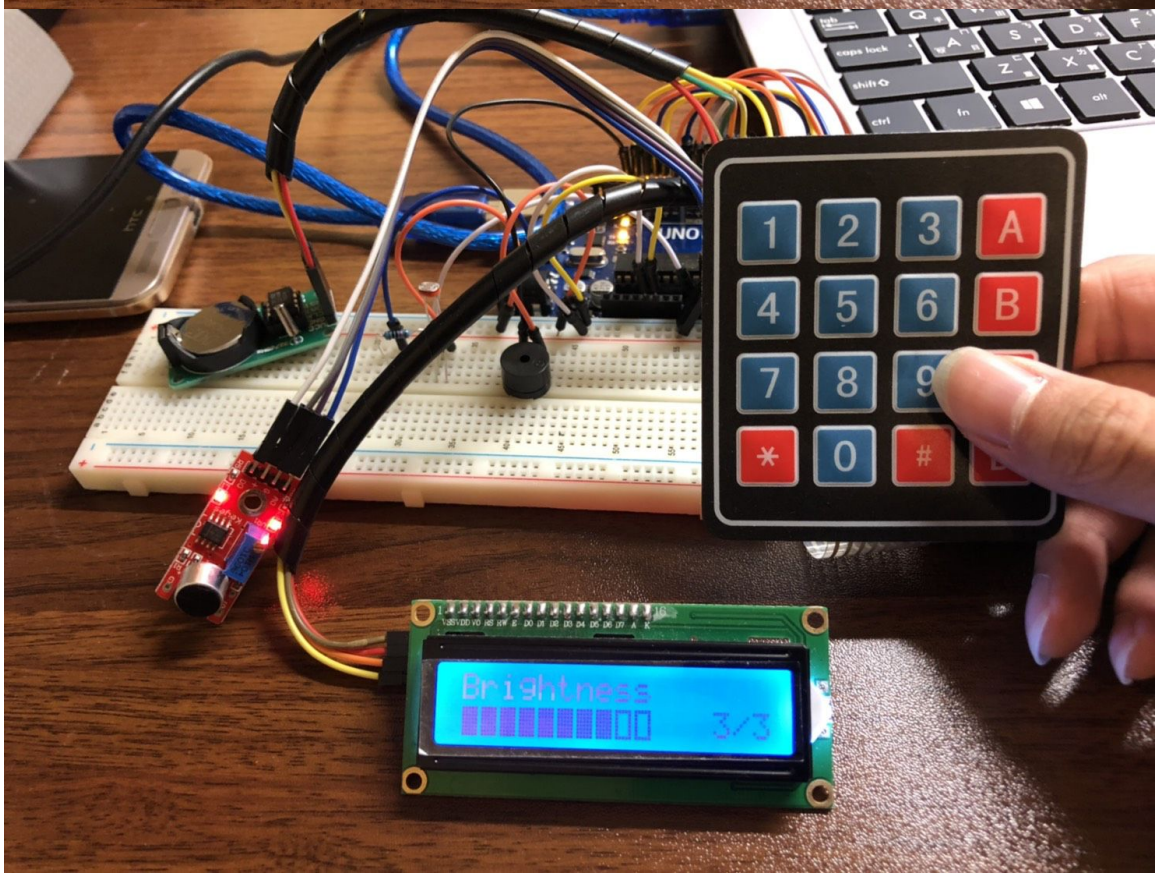
  這次 project 大部分的時間都是在處理 FreeRTOS，單一個 task 如果 stack 給得太少，stack 就會 overflow；但如果 stack 開太大，佔據太多資源，Arduino 同樣會爆掉，因此就必須把各個 task 的 stack 大小盡量壓到最低值，而這只能靠暴力法一次又一次嘗試各個 task 最剛好的 stack 大小。做到後來覺得為了一點點分數寫 FreeRTOS，還真是不划算。

- **Result**

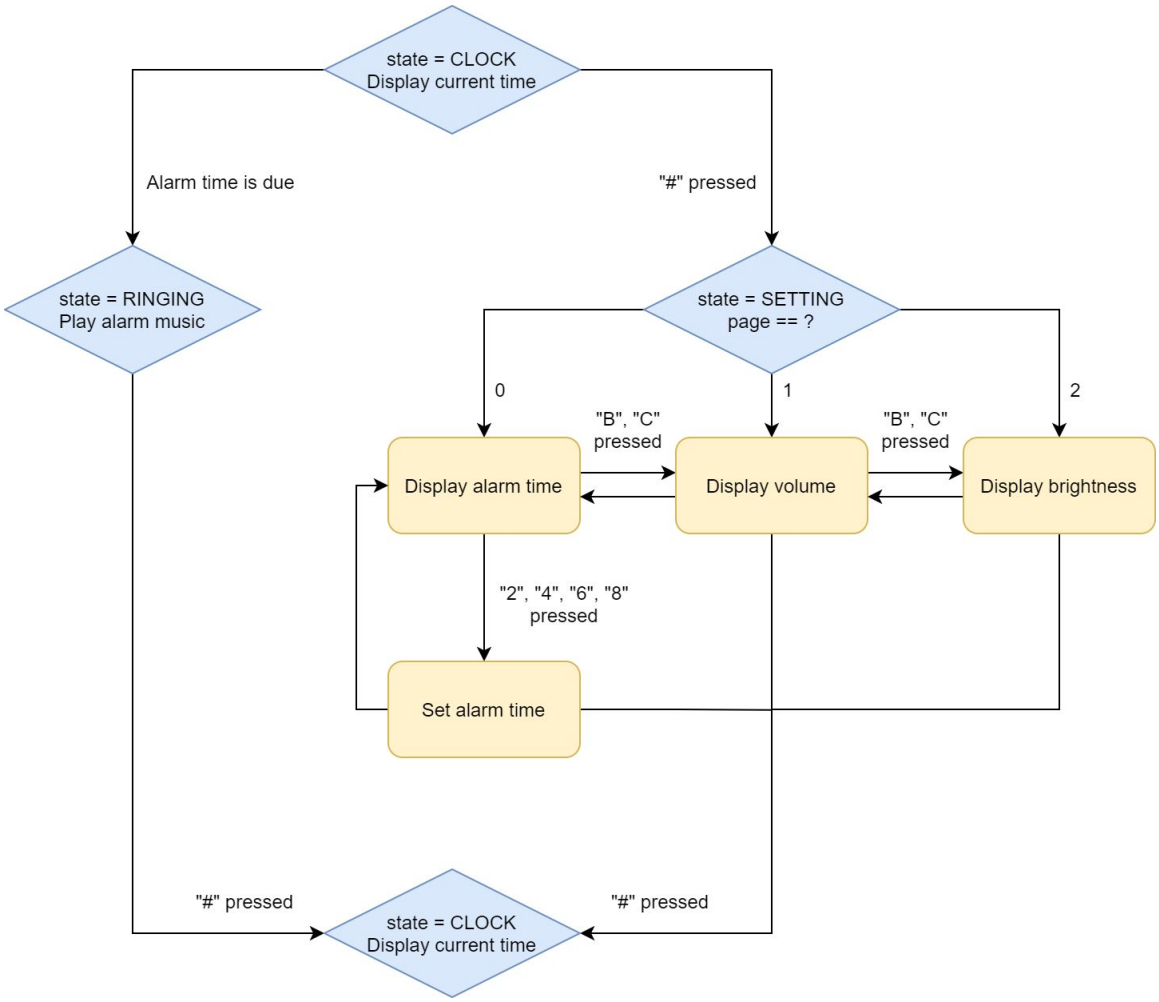





1 2 3 A
4 5 6 B
7 8 9
* 0 #
Volume
2/3


1 2 3 A
4 5 6 B
7 8 9
* 0 #
Brightness
3/3

- **Flow Chart**


state = CLOCK
Display current time
Alarm time is due
"#" pressed
state = RINGING
Play alarm music
state = SETTING
page == ?
0
1
2
"B", "C"
pressed
"B", "C"
pressed
Display alarm time
Display volume
Display brightness
"2", "4", "6", "8"
pressed
Set alarm time
"#" pressed
"#" pressed
state = CLOCK
Display current time