2018/11/17

1071 單晶片系統實作

作業 2：整點報時時鐘設計或兼具鬧鐘與時鐘功能之電子時鐘設計

中國文化大學 資訊工程學系 3B A5216047 政弘 王

目錄

圖目錄

一、 題目

1. 基本題：

   參考課本 4-9 節數位時鐘設計之接線與程式碼，使用 Arduino Uno 微控板與 KTduino 實驗版，設計一個具有報時功能的時鐘(顯示時：分)24 小時制，要求如下：(1)具有調整時、分的功能；(2)每隔 5 分鐘從蜂鳴器發出聲音；(3)每小時發出一段簡短音樂報時。(4)為了展示效果，請將時間速度加快 10 倍，也就是將 0.1 秒視為 1 秒。

2. 進階題：

   參考第 4 章及第 5 章內容，使用 Arduino Uno 微控板與 KTduino 實驗版設計一個兼具鬧鐘與時鐘功能之電子時鐘。平常具有時間顯示功能。在使用者啟動鬧鐘功能後，當所設定的時間一到，就會重複播放一段音樂，直到關掉鬧鐘功能。

作業報告內容：

(1) 封面

(2) 目錄(含報告目錄、圖目錄，要利用 Word 自動產生)

(3) 題目

(4) 程式碼(要有註解)

(5) 靜態展示(成果照片)

(6) 動態展示(拍成影片及配音說明)

(7) 學習心得

二、 基本題

1. 說明

題目： 數位時鐘(基本版)

接腳:

P11B(1~4)	：	分別接微控板的 2 、3 、4 、5 腳
P11A(a~g)	：	分別接微控板的 6 、7 、8 、9 、10 、11 、12 腳
P14	：	接微控板的 13 腳
DD	：	接微控板的 A0 腳
P5-1	：	接微控板的 A1 腳
P6-5~P6-8	：	分別接微控板的 A2 、A3 、A4 、A5 腳
P6-1	：	接 GND(低態動作)

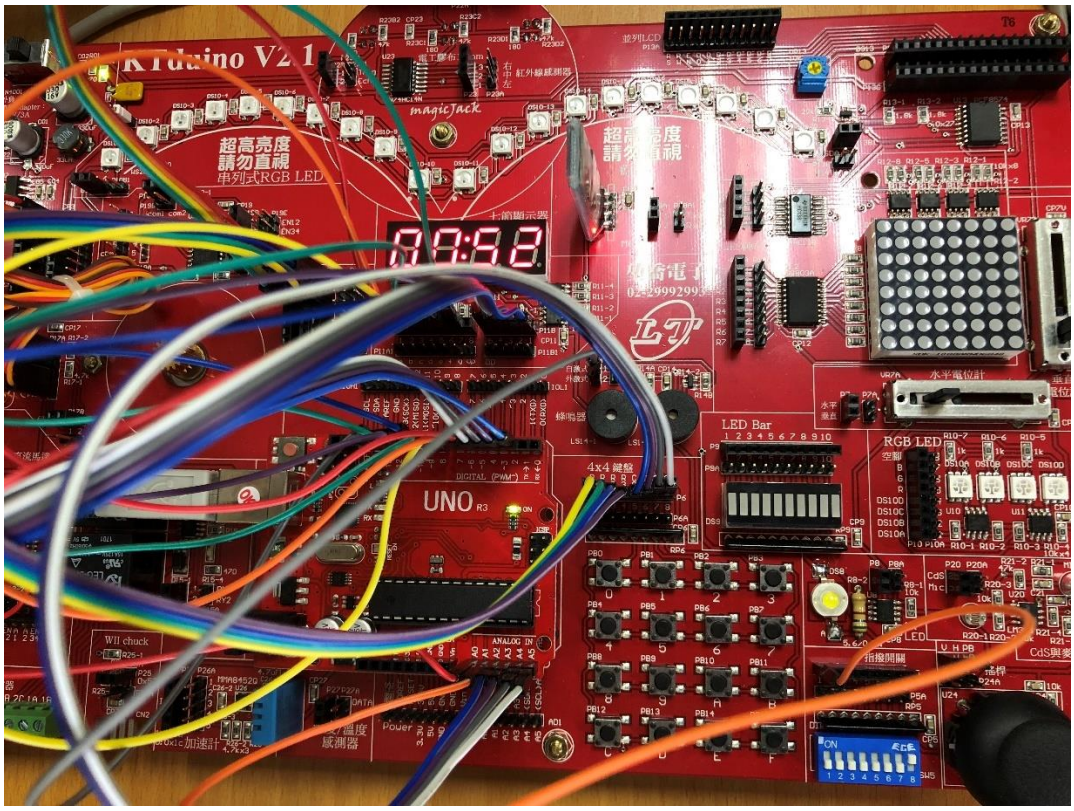
功能:

(1)具有調整時、分的功能

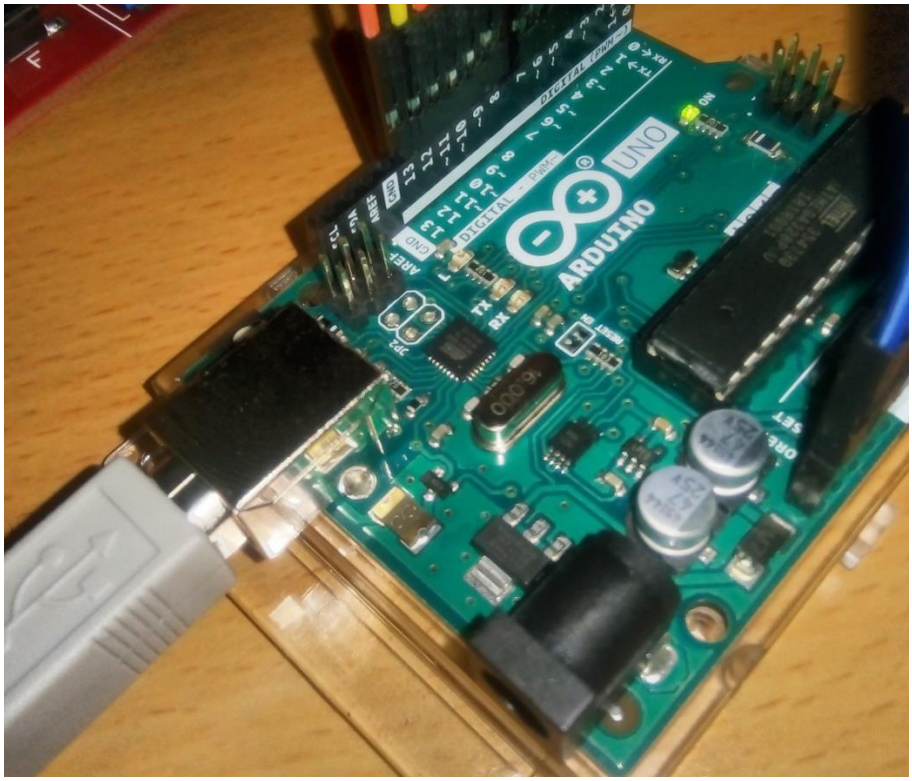
(2)每隔 5 分鐘從蜂鳴器發出聲音

(3)每小時發出一段簡短音樂報時

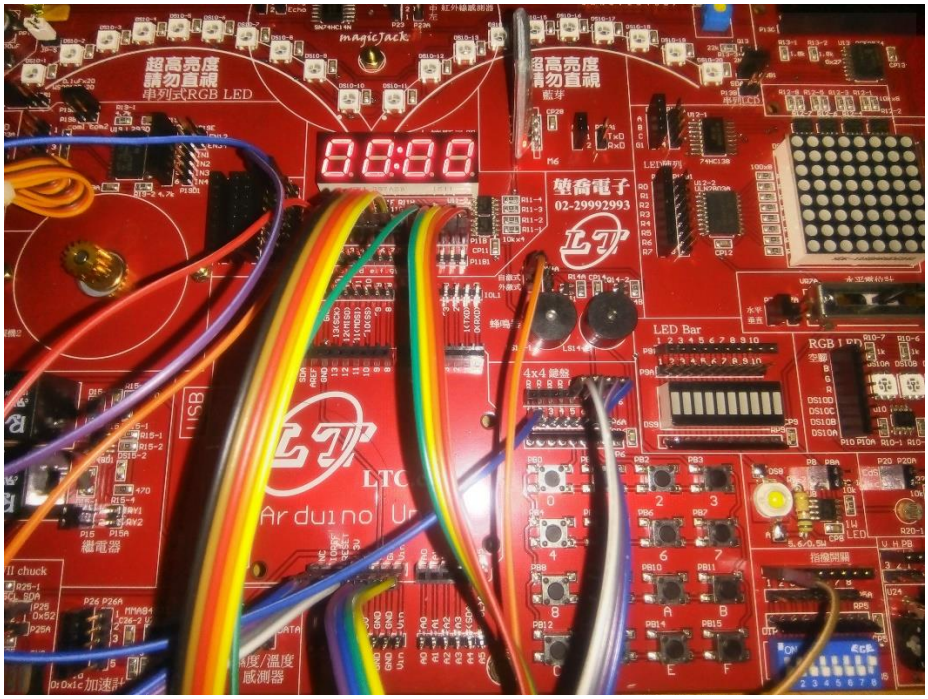
2. 成品



圖表 1 數位時鐘(基本題)成品-課程套件



圖表 2 數位時鐘(基本題)成品-UNO 板



圖表 3 數位時鐘(基本題)成品- KTduino 實驗版

3.Pseudocode

```
4-digit 7-segment:
Reference: http://www.ardumotive.com/4-digit-7seg-display-en.html
order by (a, b, c, d, e, f, g)
order by (g, f, e, d, c, b, a)
Digit:      Binary:         Hexadecimal:
N/AAA       0bXgfe dcba
0           0b1100 0000     0xC0
1           0b1111 1001     0xF9
2           0b1010 0100     0xA4
3           0b1011 0000     0xB0
4           0b1001 1001     0x99
5           0b1001 0010     0x92
6           0b1000 0010     0x82
7           0b1111 1000     0xF8
8           0b1000 0000     0x80
9           0b1001 0000     0x90

Shape:
      AAA
    F     B
      GGG
    E     C
      DDD

//-----------------------------------------
```

圖表 4 四段七節顯示器編碼設計虛擬碼

```
Declare pins

void setup(){
    Setup Pins;
}
```

圖表 5 虛擬碼片段一(宣告接腳及設定系統)

```
void loop(){
    if(switch ON){
        always{
            if(Add hour){
                Exception 1 : 23->00
                Exception 2 : 09->10
                General     : ++hour
            }

            if(Subtract hour){
                Exception 1 : 00->23
                Exception 2 : 10->09
                General     : --hour
            }

            if(Add minute){
                Exception 1 : 59->00
                Exception 2 : 09->10
                General     : ++minute
            }

            if(Subtract minute){
                Exception 1 : 00->59
                Exception 2 : 10->09
                General     : --minute
            }

            if(switch OFF){
                break;
            }
        }
    }else{
        always{
            if(++seconds == 60){
                seconds = 0;

                ++minute

                if(minute == 60){
                    minute = 0;

                    Add hour{
                        Exception 1 : 23->00
                        Exception 2 : 09->10
                        General     : ++hour
                    }
                    Play music
                }

                if(minute%5 == 0){
                    beep
                }
            }

            if(switch ON){
                break;
            }
        }
    }
}
```

圖表 6 虛擬碼片段二(主程式)

構思說明:

將程式分為變數的宣告及定義、初始化系統和主要運行程式進行思考。

變數的宣告及定義: 簡單化為接腳宣告，避免資料宣告以提升安全性。

初始化系統: 設定接腳。

主要運行程式: 判斷模式，然後執行期。

4. 程式碼

主程式

```
// Include library(s):
#include "Clock.h" //My library : 處理時鐘運作作業(詳情請看header檔)
//-------------------------------------------
```

圖表 7 主程式-引入函數庫(基礎題)

```
// Declare variables:

// Output pins:
const int scan[] = {2, 3, 4, 5};                // 宣告掃描信號接腳
const int SEG[] = {6, 7, 8, 9, 10, 11, 12};     // 宣告顯示信號接腳(g~a)
const int BZ = 13;                              // 宣告蜂鳴器接腳
const int DD = A0;                              // 宣告閃秒LED接腳(":")
// ===========================================

// Input pins
const int DIPSW[] = {A1};                       // 宣告指撥開關接腳
const int PB[] = {A2, A3, A4, A5};              // 宣告按鍵接腳
// ===========================================

// Data:
Clock* clock;   // 宣告時鐘物件
// ===========================================

//-------------------------------------------
```

圖表 8 主程式-接腳宣告及時鐘物件宣告(基礎題)

```
// Initializing system
void setup(){
    clock = new Clock(SEG, scan, PB, 4, DIPSW, 1, BZ, DD);  //Instance clock
}
//-------------------------------------------
```

圖表 9 主程式-系統初始化(實例時鐘物件) (基礎題)

```
// Main function
void loop(){
    clock->runWork();// run clock
}
//-------------------------------------------
```

圖表 10 主程式-主核心運作程式(基礎題)

時鐘物件

```
#ifndef CLOCK_H
#define CLOCK_H

// include librarys:
#include "Arduino.h"                // Ardruino base library
#include "FourDigitSevenSegment.h"  // My Library : 處理四段七節顯示器的部分作業(詳情請看header檔)
#include "MyMusic.h"                // My Library : 處理音樂的部分作業(詳情請看header檔)
//-------------------------------------------
using namespace std;

class Clock{
public: ···
private: ···
protected:
};
#endif // !CLOCK_H
```

圖表 11 時鐘物件的標頭檔(架構及引用的函數庫)

```
// Constructor: input(SEG pins(a~g), scanner pins, PB pins, DIPSW pins, and buzzer pin)
Clock(const int* SEG, const int* scan, const int* PB, int PB_length, const int* DIPSW, int DIPSW_length, const int buzzer, const int DD);
// ========================================

// Deconstructor
~Clock();
// ========================================
```

圖表 12 時鐘物件的標頭檔-Public(建構子及解構子)

```
// Functions:

// Setting Functions:

// Setting DIP switch pins
void set_DIPSW_pins(const int* pins, int length);

// Setting PB pins
void set_PB_pins(const int* pins, int length);

// Setting Buzzer pins
void set_Buzzer_pin(const int pin);

// Setting DD pins
void set_DD_pin(const int pin);
// - - - - - - - - - - - - - - - - - - - - - - -

// Clock work Functions:

// 調整模式
void adjustment();

// 計時模式
void timing();
// - - - - - - - - - - - - - - - - - - - - - - -

// The others functions:

// beep beep counts times
double beep(int pin, int counts);

// 讀取指撥開關
void DIP_swtich();
// - - - - - - - - - - - - - - - - - - - - - - -

// Run
void runWork();
// - - - - - - - - - - - - - - - - - - - - - - -

// ========================================
```

圖表 13 時鐘物件的標頭檔-Public(函數)

```
// Functions:

// Time work Functions:

// Add one hour
void add_hour(int* m_time/*m_time: time array*/);

// Subtract one hour
void subtract_hour(int* m_time/*m_time: time array*/);

// Add one minute
void add_minute(int* m_time/*m_time: time array*/);

// Subtract one minute
void subtract_minute(int* m_time/*m_time: time array*/);
// - - - - - - - - - - - - - - - - - - - - - -

// Setup functions:

// Setup all PB
void setupPB();

// Setup all dip switch
void setupDIPSW();

// Setup buzzer
void setupBuzzer();

// Setup DD
void setupDD();
// - - - - - - - - - - - - - - - - - - - - - -

// ========================================
```

圖表 14 時鐘物件的標頭檔-Private(函數)

```
// Pins:
int BZ;      // 宣告蜂鳴器接腳
int DD;      // 閃秒LED接腳(":")
int* PB;     // 按鍵接腳
int* DIPSW; // 指撥開關接腳
// ========================================
```

圖表 15 時鐘物件的標頭檔-Private(接腳變數)

```
// Data:
FourDigitSevenSegment* FDSS;    // 四段七節顯示器處理物件
MyMusic music;                  // 音樂處理物件
int disp[4] = {0, 0, 0, 0};     // 顯示緩衝區陣列(buffer)
unsigned long long x0,x1,x2;    // 時間變數
boolean phase0 = true;          // 工作旗標(flag)：phase0=false計時模式(ON), phase0=true調整模式(OFF)
double seconds=0;               // 秒變數
bool sec = false;               // 閃秒LED旗標(flag)
// ========================================
```

圖表 16 時鐘物件的標頭檔-Private(資料變數)

```
// Constructor: input(SEG pins(a~g), scanner pins, PB pins, DIPSW pins, and buzzer pin)
Clock::Clock(const int* SEG, const int* scan, const int* PB, int PB_length, const int* DIPSW, int DIPSW_length, const int buzzer, const int DD){
    FDSS = new FourDigitSevenSegment(SEG, scan);    //Instance FDSS

    this->PB = new int[ PB_length ];        // Instance PB array
    this->DIPSW = new int[ DIPSW_length ];  // Instance DIPSW array

    // Setting pins:
    set_PB_pins(PB, PB_length );
    set_DIPSW_pins(DIPSW, DIPSW_length );
    set_Buzzer_pin(buzzer);
    set_DD_pin(DD);
    // ========================================
}
//----------------------------------------
```

圖表 17 時鐘物件的建構子

```
// Deconstructor
Clock::~Clock(){

}
//------------------------------------------------
```

圖表 18 時鐘物件的解構子

```
// Setting DIP switch pins
void Clock::set_DIPSW_pins(const int* pins, int length){
    for(int i=0; i<length; ++i){
        DIPSW[i] = pins[i];
    }

    setupDIPSW();
}
//-----------------------------------------
```

圖表 19 時鐘物件的設定開關接腳函數

```
// Setting PB pins
void Clock::set_PB_pins(const int* pins, int length){
    for(int i=0; i<length; ++i){
        PB[i] = pins[i];
    }

    setupPB();
}
//-----------------------------------------
```

圖表 20 時鐘物件的設定按鈕接腳函數

```
// Setting Buzzer pins
void Clock::set_Buzzer_pin(const int pin){
    BZ = pin;

    setupBuzzer();
}
//---------------------------------------------
```

圖表 21 時鐘物件的設定蜂鳴器接腳函數

```
// Setting DD pins
void Clock::set_DD_pin(const int pin){
    DD = pin;

    setupDD();
}
//---------------------------------------------
```

圖表 22 時鐘物件的設定":"LED 接腳函數

```
// 調整模式
void Clock::adjustment(){
    while(phase0){
        seconds = 0;      // 秒數歸零
        digitalWrite(DD, false); // 閃秒固定亮著

        // 時遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[0])){    // 若PB[0]被按下(時遞增)
            add_hour(disp);
        }

        // 時遞減調整(按鍵為低態動作)
        if(!digitalRead(PB[1])){    // 若PB[1]被按下(時遞減)
            subtract_hour(disp);
        }

        // 分遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[2])){    // 若PB[2]被按下(分遞增)
            add_minute(disp);
        }

        // 分遞減調整(按鍵為低態動作)
        if(!digitalRead(PB[3])){    // 若PB[3]被按下(分遞減)
            subtract_minute(disp);
        }

        FDSS->scanner(25, disp);    // 連續掃描25週(100ms，每秒約10次)

        DIP_swtich();   // 讀取指撥開關
    }
}
//---------------------------------------------
```

圖表 23 時鐘物件的調整模式運行函數

```
// 計時模式
void Clock::timing(){
    while(!phase0){
        FDSS->scanner(2, disp); // 連續掃描2週(8ms)
        x1 = millis();          // 查詢時間
        x2=x1-x0;               // 計算時間差

        if(x2>=500){    // 0.5秒時間差
            x0=x1;      // 記錄時間

            // 切換閃秒:
            sec=!sec;
            digitalWrite(DD, sec);
            // - - - - - - - - - - - - - - - - - - - - -

            // 處理時間
            if(sec){
                if(++seconds>=60){      // 秒數加1
                    seconds-=60;        // 若秒數滿60則歸零
                    if(disp[0]==9){     // 若分之個位數滿9
                        disp[0]=0;      // 則分之個位數歸零

                        if(disp[1]==5){ // 若分之十位數滿5
                            disp[1]=0;  // 則分之十位數歸零

                            add_hour(disp);     // Add one hour

                            digitalWrite(DD, 1);    // 閃秒固定暗著
                            FDSS->closeAll();       // 連續掃描2週(8ms)

                            seconds+=music.LittleStar(BZ);  // Play Music: Little Star and fix delay time
                        }else{
                            ++disp[1];  //否則分之十位數加1
                        }

                    }else{
                        ++disp[0];      //否則分之個位數加1
                    }

                    if(disp[0]==0 || disp[0]==5){   // 如果分能被五整除
                        seconds += beep(BZ, 2);     // 叫兩聲且修正時間
                    }

                }
            }

        }

        DIP_swtich();   // 讀取指撥開關
    }
}
//-----------------------------------------
```

圖表 24 時鐘物件的計時模式運行函數

```
// 讀取指撥開關
void Clock::DIP_swtich(){
    if(digitalRead(DIPSW[0])){ // 讀取指撥開關true：調整模式
        phase0 = true;
    }else{                     // false：計時模式
        phase0 = false;
    }
}
//-----------------------------------------
```

圖表 25 時鐘物件的讀取指撥開關函數

```
// Run
void Clock::runWork(){
    DIP_swtich();    // 讀取指撥開關
    adjustment();    // 調整模式
    timing();        // 計時模式
}
// - - - - - - - - - - - - - - - - - - - - - - - -
```

圖表 26 時鐘物件的主運行函數

```
// Add one hour
void Clock::add_hour(int* m_time/*m_time: time array*/){
    if(m_time[3]==2 && m_time[2]==3){   // 若已23時
        m_time[3]=0;                    // 時之十位數歸零
        m_time[2]=0;                    // 時之個位數歸零
    }else if(m_time[2]==9){             // 若時之個位數等於9
        m_time[2]=0;                    // 時之個位數歸零
        ++m_time[3];                    // 時之十位數遞增(進位)
    }else{
        ++m_time[2];                    // 否則時之個位數遞增
    }
}
//-----------------------------------------
```

圖表 27 時鐘物件的加一小時函數

```
// Subtract one hour
void Clock::subtract_hour(int* m_time/*m_time: time array*/){
    if(m_time[3]==0 && m_time[2]==0){   // 若已00時
        m_time[3]=2;                    // 時之十位數調整為2
        m_time[2]=3;                    // 時之個位數調整為3
    }else if(m_time[2]==0){             // 若時之個位數等於0
        m_time[2]=9;                    // 時之個位調整為9
        --m_time[3];                    // 時之十位數遞減(借位)
    }else{
        --m_time[2];                    // 否則時之個位數遞減
    }
}
//-----------------------------------------
```

圖表 28 時鐘物件的減一小時函數

```
// Add one minute
void Clock::add_minute(int* m_time/*m_time: time array*/){
    if(m_time[1]==5 && m_time[0]==9){    // 若已59分
        m_time[1]=0;                     // 分之十位數歸零
        m_time[0]=0;                     // 分之個位數歸零
    }else if(m_time[0]==9){              // 若分之個位數等於9
        m_time[0]=0;                     // 分之個位數歸零
        ++m_time[1];                     // 分之十位數遞增(進位)
    }else{
        ++m_time[0];                     // 否則分之個位數遞增
    }
}
//-------------------------------------------
```

圖表 29 時鐘物件的加一分鐘函數

```
// Subtract one minute
void Clock::subtract_minute(int* m_time/*m_time: time array*/){
    if(m_time[1]==0 && m_time[0]==0){    // 若已00分
        m_time[1]=5;                     // 分之十位數調整為5
        m_time[0]=9;                     // 分之個位數調整為9
    }else if(m_time[0]==0){              // 若分之個位數等於0
        m_time[0]=9;                     // 分之個位調整為9
        --m_time[1];                     // 分之十位數遞減(借位)
    }else{
        --m_time[0];                     // 否則分之個位數遞減
    }
}
//-------------------------------------------
```

圖表 30 時鐘物件的減一分鐘函數

```
// beep beep counts times
double Clock::beep(int pin, int counts){
    double delayTime = 0;
    for(int i=0; i<counts; ++i){     // 執行counts次
        tone(pin, 1000, 100);        // 發聲(1kHz,0.1秒)
        delay(100);                  // 靜音(0.1秒)
        delayTime += 0.2;
    }

    return delayTime;
}
//-------------------------------------------
```

圖表 31 時鐘物件的 beep beep 叫函數

```
// Setup all PB
void Clock::setupPB(){
    for(int i=0; i<(sizeof(PB)/sizeof(PB[0])); ++i){
        pinMode(PB[i], INPUT);
    }
}
//-------------------------------------------
```

圖表 32 時鐘物件的按鈕接腳設置函數

```
// Setup all dip switch
void Clock::setupDIPSW(){
    for(int i=0; i<(sizeof(DIPSW)/DIPSW[0]); ++i){
        pinMode(DIPSW[i], INPUT);
    }
}
//-------------------------------------------
```

圖表 33 時鐘物件的開關接腳設置函數

```
// Setup buzzer
void Clock::setupBuzzer(){
    pinMode(BZ, OUTPUT);
}
//-------------------------------------------
```

圖表 34 時鐘物件的蜂鳴器接腳設置函數

```
// Setup DD
void Clock::setupDD(){
    pinMode(DD, OUTPUT);
}
//-------------------------------------------
```

圖表 35 時鐘物件的":"LED 接腳設置函數

四段七節顯示器物件

```
  7   #ifndef FOURDIGITSEVENSEGMENT_H
  8   #define FOURDIGITSEVENSEGMENT_H
  9
 10   // include librarys:
 11   #include "Arduino.h"    //Ardruino base library
 12   //-----------------------------------------
 13   using namespace std;
 14
 15   class FourDigitSevenSegment
 16   {
 17 ⊞ public: ···
 71 ⊞ private: ···
102   protected:
103   };
104
105   #endif // !FOURDIGITSEVENSEGMENT_H
```

圖表 36 四段七節顯示器物件的標頭檔(架構及引用的函數庫)

```
// Constructors:

// Constructor: default
FourDigitSevenSegment(/* args */);

// Constructor: input(SEG pins(a~g) and scanner pins)
FourDigitSevenSegment(const int* SEG_pins, const int* scan_pins);

// Constructor: input(SEG pins(a~g and dp) and scanner pins)
FourDigitSevenSegment(const int* SEG_pins, const int* scan_pins, int dp);
// ========================================

// Deconstructor:
~FourDigitSevenSegment();
// ========================================
```

圖表 37 四段七節顯示器物件的標頭檔-Public(建構子及解構子)

```
// Functions:

// Setting Functions:

// Setting Individual Segments Illuminated pins(a~g)
void set_SEG_pins(const int* pins/*pins: SEG pins array(g~a)*/);

// Setting Individual Segments Illuminated pins(a~g, and dp)
void set_SEG_pins(int g, int f, int e, int d, int c, int b, int a, int dp);

// Setting Scanner pins
void set_scan_pins(const int* pins/*scan_pins: scanner pins array*/);

// Setting Scanner pins
void set_scan_pins(int pin0, int pin1, int pin2, int pin3);

// Setting Individual Segments Illuminated pins(dp)
void set_dp_pin(int dp/*dp: SEG pins array(dp)*/);
// - - - - - - - - - - - - - - - - - - - - - - -

// Getting Functions:

// Getting length of SEG
int get_SEG_size();

// Getting length of scan
int get_scan_size();
// - - - - - - - - - - - - - - - - - - - - - - -

// The Others Functions:

// Scan x times and display numbers
void scanner(int x/*x: Scanning x times*/, int* disp/*disp: display numbers array*/);

// Close all LED
void closeAll();
// ========================================
```

圖表 38 四段七節顯示器物件的標頭檔-Public(函數)

```
//Functions:

// Display numbers
void setNumber(int number);

// setup Individual Segments Illuminated pins(a~g)
void setupSEG();

// setup scanner pins
void setupScan();

// setup Individual Segments Illuminated pins(dp)
void setupDp();
// ========================================
```

圖表 39 四段七節顯示器物件的標頭檔-Private(函數)

```
// Pins:
int* SEG;        // 宣告顯示信號接腳陣列(a~g)
int dp;          // 宣告顯示信號接腳(dp)
int* scan;       // 宣告掃描信號接腳陣列(scanner)
// ========================================
```

圖表 40 四段七節顯示器物件的標頭檔-Private(接腳變數)

```
// Data:
const int SEG_size = 7;          // length of SEG
const int scan_szie = 4;         // length of scan

const int SEG_code[10] = {       // 宣告七節顯示碼陣列
    0xc0,0xf9,0xa4,0xb0,0x99,    // 0~4
    0x92,0x82,0xf8,0x80,0x90     // 5~9
};
// ========================================
```

圖表 41 四段七節顯示器物件的標頭檔-Private(資料變數)

```
// Constructor: default
FourDigitSevenSegment::FourDigitSevenSegment(/* args */)
{
    SEG = new int[SEG_size];    // Instance SEG array
    scan = new int[scan_szie];  // Instance scan array

    // Make default scanner pins
    for(int i=0; i<get_scan_size(); ++i){
        scan[i] = i+2;
    }

    // Make default Individual Segments Illuminated pins
    for(int i=0; i<get_SEG_size(); ++i){
        SEG[i] = i+6;
    }

    // Setup pins:
    setupScan();
    setupSEG();
    // ============================================
}
//--------------------------------------------
```

圖表 42 四段七節顯示器物件的建構子 1

```
// Constructor: input(SEG pins(a~g) and scanner pins)
FourDigitSevenSegment::FourDigitSevenSegment(const int* SEG_pins, const int* scan_pins)
{
    SEG = new int[SEG_size];    // Instance SEG array
    scan = new int[scan_szie];  // Instance scan array

    // Setting pins:
    set_SEG_pins(SEG_pins);
    set_scan_pins(scan_pins);
    // ==========================================
}
//--------------------------------------------
```

圖表 43 四段七節顯示器物件的建構子 2

```
// Constructor: input(SEG pins(a~g and dp) and scanner pins)
FourDigitSevenSegment::FourDigitSevenSegment(const int* SEG_pins, const int* scan_pins, int dp)
{
    SEG = new int[SEG_size];    // Instance SEG array
    scan = new int[scan_szie];  // Instance scan array

    // Setting pins:
    set_SEG_pins(SEG_pins);
    set_scan_pins(scan_pins);
    set_dp_pin(dp);
    // ==========================================
}
//--------------------------------------------
```

圖表 44 四段七節顯示器物件的建構子 3

```
// Deconstructor
FourDigitSevenSegment::~FourDigitSevenSegment()
{
}
//--------------------------------------------
```

圖表 45 四段七節顯示器物件的解構子

```
// Setting Individual Segments Illuminated pins(a~g)
void FourDigitSevenSegment::set_SEG_pins(const int* pins/*pins: SEG pins array(g~a)*/){
    for(int i=0; i<get_SEG_size(); ++i){
        SEG[i] = pins[i];
    }

    setupSEG();
}
//--------------------------------------------
```

圖表 46 四段七節顯示器物件的設定 SEG 接腳函數 1

```
// Setting Individual Segments Illuminated pins(a~g, and dp)
void FourDigitSevenSegment::set_SEG_pins(int g, int f, int e, int d, int c, int b, int a, int dp){
    SEG[0] = g;
    SEG[1] = f;
    SEG[2] = e;
    SEG[3] = d;
    SEG[4] = c;
    SEG[5] = b;
    SEG[6] = a;
    setupSEG();

    set_dp_pin(dp);
}
//--------------------------------------------
```

圖表 47 四段七節顯示器物件的設定 SEG 接腳函數 2

```
// Setting Scanner pins
void FourDigitSevenSegment::set_scan_pins(const int* pins/*scan_pins: scanner pins array*/){
    for(int i=0; i<get_scan_size(); ++i){
        scan[i] = pins[i];
    }

    setupScan();
}
//--------------------------------------------
```

圖表 48 四段七節顯示器物件的設定 scan 接腳函數 1

```
// Setting Scanner pins
void FourDigitSevenSegment::set_scan_pins(int pin0, int pin1, int pin2, int pin3){
    scan[0] = pin0;
    scan[1] = pin1;
    scan[2] = pin2;
    scan[3] = pin3;

    setupScan();
}
//----------------------------------------
```

圖表 49 四段七節顯示器物件的設定 scan 接腳函數 2

```
// Setting Individual Segments Illuminated pins(dp)
void FourDigitSevenSegment::set_dp_pin(int dp/*dp: SEG pins array(dp)*/){
    this->dp = dp;
    setupDp();
}
//----------------------------------------
```

圖表 50 四段七節顯示器物件的設定 dp 接腳函數

```
// Getting length of SEG
int FourDigitSevenSegment::get_SEG_size(){
    return SEG_size;
}
//----------------------------------------
```

圖表 51 四段七節顯示器物件的取得 SEG array size 函數

```
// Getting length of scan
int FourDigitSevenSegment::get_scan_size(){
    return scan_szie;
}
//----------------------------------------
```

圖表 52 四段七節顯示器物件的取得 scan array size 函數

```
// Scan x times and display numbers
void FourDigitSevenSegment::scanner(int x/*x: Scanning x times*/, int* disp/*disp: display numbers array*/){
    for(int i=0; i<x; ++i){             // Do x times
        for(int j=0; j<get_scan_size(); ++j){
            if(disp[j]>9 || disp[j]<0){ // Prevent Error
                return;
            }

            closeAll();                 // Close all LED(Prevent afterimage)

            setNumber(disp[j]);         // Display numbers

            digitalWrite(scan[j], 0);   //Output Scanning signal
            delay(1);                   //delay 1ms
        }
    }
}
//----------------------------------------
```

圖表 53 四段七節顯示器物件的 LED 顯示函數

```
// Close all LED
void FourDigitSevenSegment::closeAll(){
    for(int k=0; k<get_scan_size(); ++k){
        digitalWrite(scan[k], 1);   // Close kth LED (低態動作)
    }
}
//----------------------------------------------
```

圖表 54 四段七節顯示器物件的 LED 關閉函數

```
// Display numbers
void FourDigitSevenSegment::setNumber(int number){
    if(number>9 || number<0){   // Prevent Error
        return;
    }

    int DISP = SEG_code[number];    // Getting SEG code of number

    for(int k=0; k<get_SEG_size(); ++k){
        if(bitRead(DISP, k)){
            digitalWrite(SEG[k], 1);    // Close LED(a,b,c,...,g) which SEG code is HIGH(低態動作)
        }else{
            digitalWrite(SEG[k], 0);    // Open LED(a,b,c,...,g) which SEG code is LOW(低態動作)
        }
    }
}
//----------------------------------------
```

圖表 55 四段七節顯示器物件的設定 LED 顯示數字函數

```
// setup Individual Segments Illuminated pins(a~g)
void FourDigitSevenSegment::setupSEG(){
    for(int i=0; i<get_SEG_size(); ++i){
        pinMode(SEG[i], OUTPUT);
    }
}
//----------------------------------------
```

圖表 56 四段七節顯示器物件的 SEG 接腳設置函數

```
// setup scanner pins
void FourDigitSevenSegment::setupScan(){
    for(int i=0; i<get_scan_size(); ++i){
        pinMode(scan[i], OUTPUT);
    }
}
//----------------------------------------
```

圖表 57 四段七節顯示器物件的 scan 接腳設置函數

```
// setup Individual Segments Illuminated pins(dp)
void FourDigitSevenSegment::setupDp(){
    pinMode(dp, OUTPUT);
}
//----------------------------------------
```

圖表 58 四段七節顯示器物件的 dp 接腳設置函數

音樂物件

```
 8   #ifndef MYMUSIC_H
 9   #define MYMUSIC_H
10
11   // Include librarys:
12   #include "Arduino.h"                    // Ardruino base library
13   //-----------------------------------------
14   using namespace std;
15
16   class MyMusic{
17   public:···
34   private:···
95   protected:
96   };
97   #endif // !MYMUSIC_H
```

圖表 59 音樂物件的標頭檔(架構及引用的函數庫)

```
// Constructor:
MyMusic();
// ===========================================

// Deconstructor:
~MyMusic();
// ===========================================
```

圖表 60 音樂物件的標頭檔-Public(建構子及解構子)

```
// Funcrions:

// 播放樂曲: 小星星
double LittleStar(int buzzer/*buzzer: pin of buzzer*/);

// 播放樂曲: 小蜜蜂
double Bee(int buzzer/*buzzer: pin of buzzer*/);
// ===========================================
```

圖表 61 音樂物件的標頭檔-Public(函數)

```
// Functions:
// Play Music and get delay time
double playMusic(int buzzer/*buzzer: pin of buzzer*/, int* tones/*tones: tones array*/, int tones_size/*tones_size: size of tones array*/, double* beat/*beat: beat array*/, int tempo/*tempo: tempo*/);
// ===========================================
```

圖表 62 音樂物件的標頭檔-Private(函數)

```
// tones:
const int pause        = 0;         // 休止符
// 低音
const int Low_C        = 262;       // C(Do)
const int Low_C_sharp  = 277;       // C#
const int Low_D        = 294;       // D(Re)
const int Low_D_sharp  = 311;       // D#
const int Low_E        = 330;       // E(Mi)
const int Low_F        = 349;       // F(Fa)
const int Low_F_sharp  = 370;       // F#
const int Low_G        = 392;       // G(So)
const int Low_G_sharp  = 415;       // G#
const int Low_A        = 440;       // A(La)
const int Low_A_sharp  = 466;       // A#
const int Low_B        = 494;       // B(Si)
// 中音
const int Mid_C        = 523;       // C(Do)
const int Mid_C_sharp  = 554;       // C#
const int Mid_D        = 587;       // D(Re)
const int Mid_D_sharp  = 622;       // D#
const int Mid_E        = 659;       // E(Mi)
const int Mid_F        = 698;       // F(Fa)
const int Mid_F_sharp  = 740;       // F#
const int Mid_G        = 784;       // G(So)
const int Mid_G_sharp  = 831;       // G#
const int Mid_A        = 880;       // A(La)
const int Mid_A_sharp  = 932;       // A#
const int Mid_B        = 988;       // B(Si)
// 高音
const int High_C       =  1046;     // C(Do)
const int High_C_sharp =  1109;     // C#
const int High_D       =  1175;     // D(Re)
const int High_D_sharp =  1245;     // D#
const int High_E       =  1318;     // E(Mi)
const int High_F       =  1397;     // F(Fa)
const int High_F_sharp =  1480;     // F#
const int High_G       =  1568;     // G(So)
const int High_G_sharp =  1661;     // G#
const int High_A       =  1760;     // A(La)
const int High_A_sharp =  1865;     // A#
const int High_B       =  1976;     // B(Si)
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

圖表 63 音樂物件的音調變數-Private

```
// beat:
const double Quasihemidemisemiquaver    = 0.03125;  // 一百二十八分音符(美式英文: Hundred twenty-eighth note)
const double Hemidemisemiquaver         = 0.0625;   // 六十四分音符(美式英文: Sixty-fourth note)
const double Demisemiquaver             = 0.1818;   // 二十二分音符(美式英文: Thirty-second note)
const double Semiquaver                 = 0.25;     // 十六分音符(美式英文: Sixteenth note)
const double Quaver                     = 0.5;      // 八分音符(美式英文: Eighth note)
const double Crotchet                   = 1;        // 四分音符(美式英文: Quarter note)
const double Minim                      = 2;        // 二分音符(美式英文: Half note)
const double Semibreve                  = 4;        // 全音符(美式英文: Whole note)
// - - - - - - - - - - - - - - - - - - - - - - -
```

圖表 64 音樂物件的節拍變數-Private

```
// Constructor
MyMusic::MyMusic(){

}
//-----------------------------------------------------
```

圖表 65 音樂物件的建構子

```
// Deconstructor
MyMusic::~MyMusic(){

}
//-----------------------------------------------------
```

圖表 66 音樂物件的解構子

```
// 播放樂曲: 小星星
double MyMusic::LittleStar(int buzzer/*buzzer: pin of buzzer*/){

    int tempo = 300;
    double beat[] = {
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim
    };

    int tones[] = {
        Mid_C, Mid_C, Mid_G, Mid_G, Mid_A, Mid_A, Mid_G,  // Twinkle, twinkle, little star,
        Mid_F, Mid_F, Mid_E, Mid_E, Mid_D, Mid_D, Mid_C,  // How I wonder what you are!
        Mid_G, Mid_G, Mid_F, Mid_F, Mid_E, Mid_E, Mid_D,  // Up above the world so high,
        Mid_G, Mid_G, Mid_F, Mid_F, Mid_E, Mid_E, Mid_D,  // Like a diamond in the sky.
        Mid_C, Mid_C, Mid_G, Mid_G, Mid_A, Mid_A, Mid_G,  // Twinkle, twinkle, little star,
        Mid_F, Mid_F, Mid_E, Mid_E, Mid_D, Mid_D, Mid_C   // How I wonder what you are!
    };

    // Play Music and return delay time
    return playMusic(buzzer, tones, sizeof(tones)/sizeof(tones[0]), beat, tempo);
}
//-------------------------------------------
```

圖表 67 音樂物件的播放音樂:小星星函數

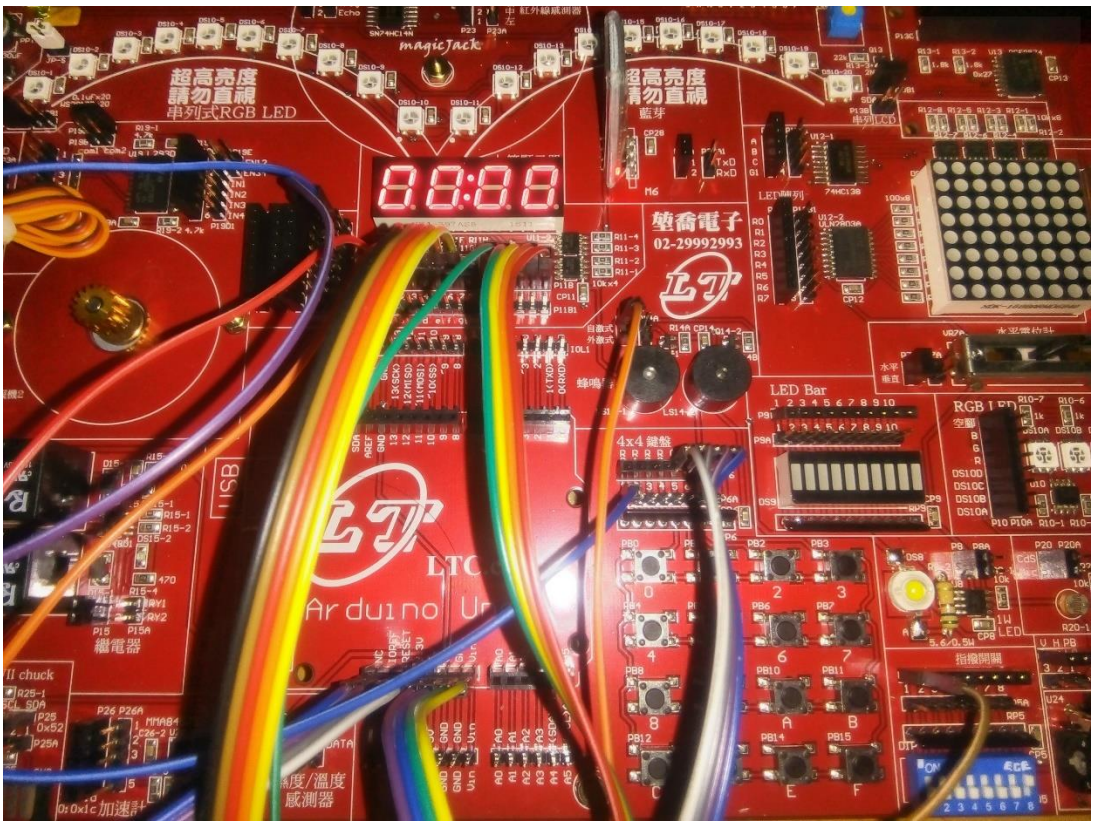
```
// 播放樂曲: 小蜜蜂
double MyMusic::Bee(int buzzer/*buzzer: pin of buzzer*/){

    int tempo = 300;
    double beat[] = {
        Crotchet, Crotchet, Minim, Crotchet, Crotchet, Minim, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Minim, Crotchet, Crotchet, Minim, Crotchet, Crotchet, Crotchet, Crotchet, Semibreve,
        Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Crotchet, Minim,
        Crotchet, Crotchet, Minim, Crotchet, Crotchet, Minim, Crotchet, Crotchet, Crotchet, Crotchet, Semibreve
    };

    int tones[] = {
        Mid_G, Mid_E, Mid_E, Mid_F, Mid_D, Mid_D, Mid_C, Mid_D, Mid_E, Mid_F, Mid_G, Mid_G, Mid_G,
        Mid_G, Mid_E, Mid_E, Mid_F, Mid_D, Mid_D, Mid_C, Mid_E, Mid_G, Mid_G, Mid_E,
        Mid_D, Mid_D, Mid_D, Mid_D, Mid_D, Mid_E, Mid_F, Mid_E, Mid_E, Mid_E, Mid_E, Mid_F, Mid_G,
        Mid_G, Mid_E, Mid_E, Mid_F, Mid_D, Mid_D, Mid_C, Mid_E, Mid_G, Mid_G, Mid_C
    };

    // Play Music and return delay time
    return playMusic(buzzer, tones, sizeof(tones)/sizeof(tones[0]), beat, tempo);
}
//-------------------------------------------
```

圖表 68 音樂物件的播放音樂:小蜜蜂函數

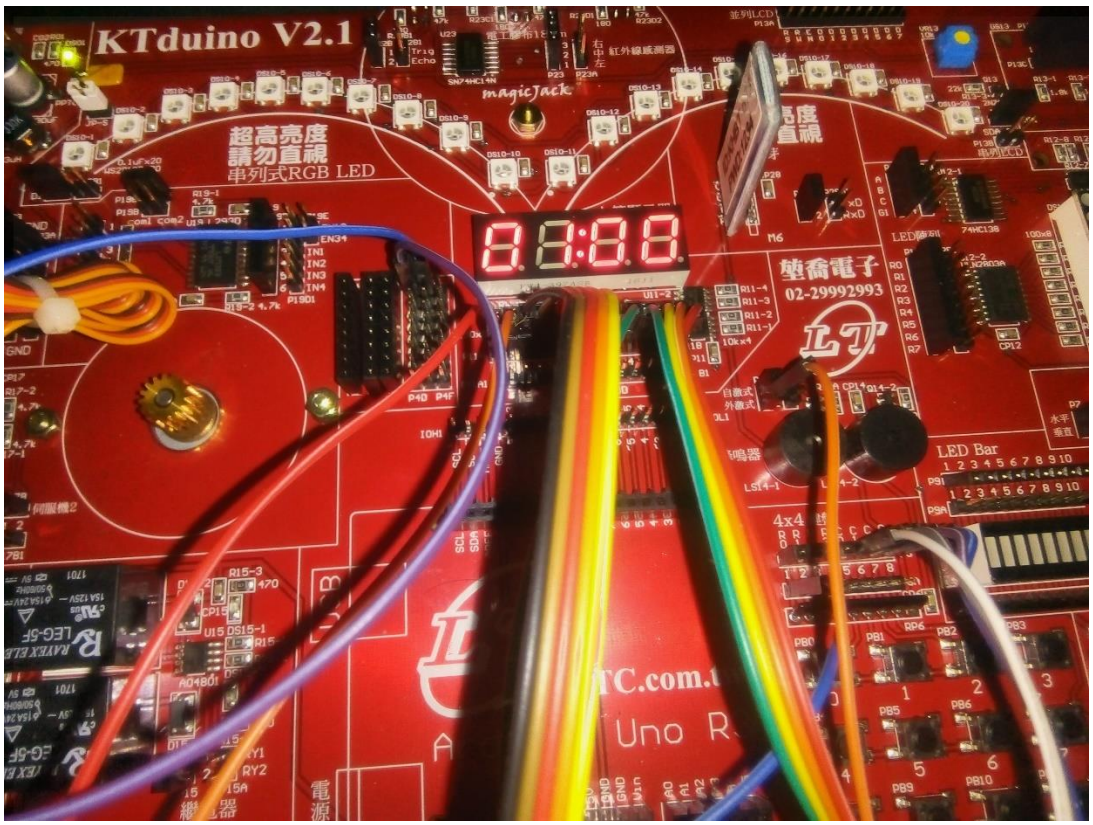
```
// Play Music and get delay time
double MyMusic::playMusic(int buzzer, int* tones, int tones_size, double* beat, int tempo){
    double delay_time = 0; // fixed delay time

    for(int i=0; i<tones_size; ++i){

        if(tones[i] != 0){                                  // if the tone is not pause.
            tone(buzzer, tones[i], beat[i]*tempo);          // play one tone with one beat
        }

        delay_time += ( ((double)tempo) /1000 )*beat[i];    // Computing fix time

        delay(beat[i]*tempo);   // Waitting one beat
        noTone(buzzer);         // Close buzzer (Prevent afterimage)
    }

    return delay_time;          // return fix time
}
//-----------------------------------------
```

圖表 69 音樂物件的音樂播放函數

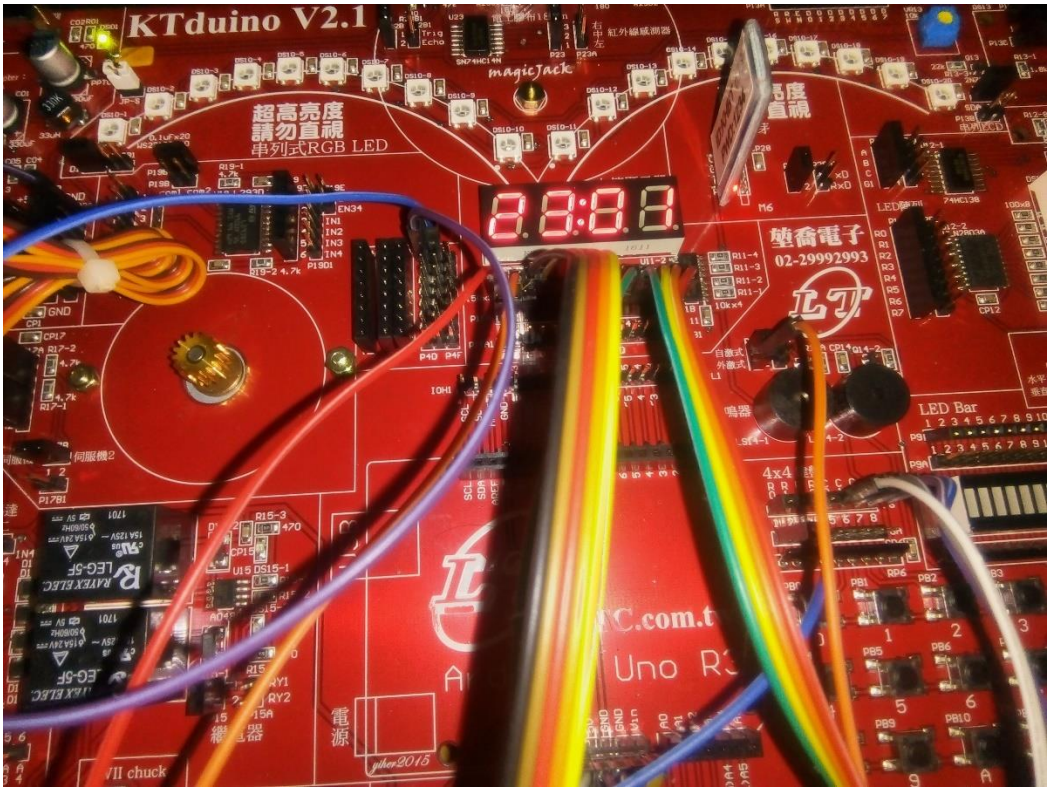
5. 靜態展示



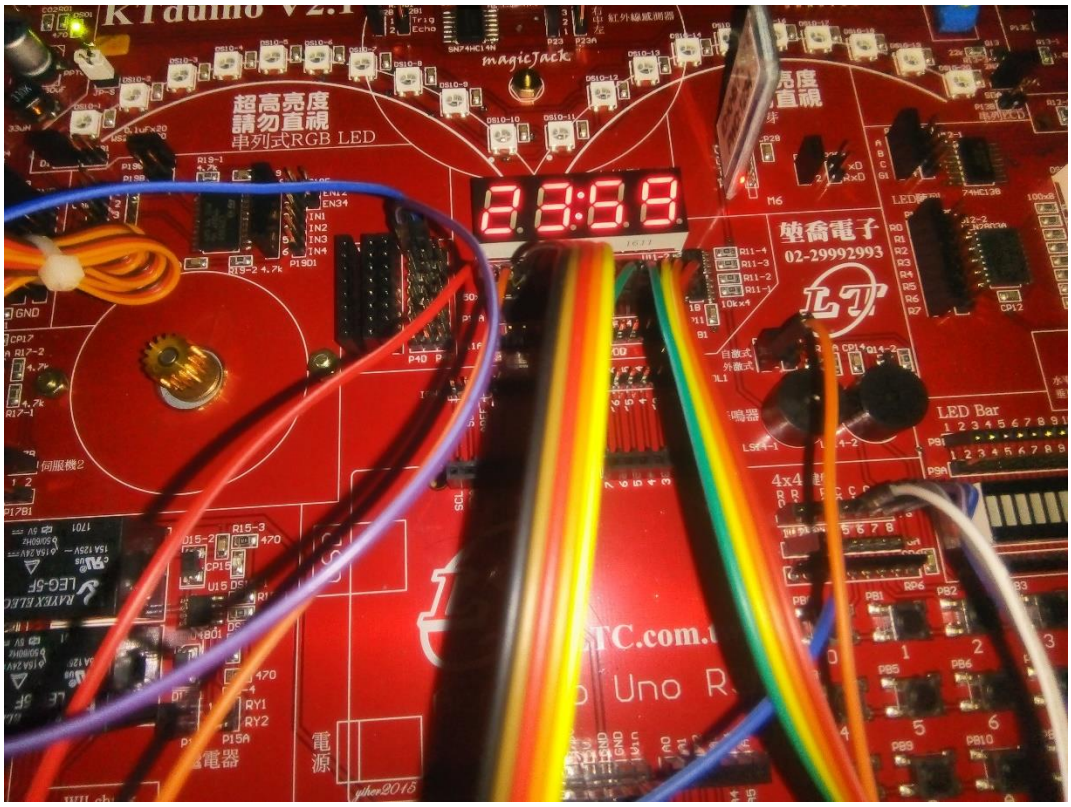
圖表 70 運行初始狀態



圖表 71 時的調整(00 到 01)



圖表 72 分的調整(00 到 01)及時的調整(00 到 23)



圖表 73 分的調整(00 到 59)

6. 動態展示

YouTube

Google Cloud

三、 進階題

1.說明

題目： 兼具鬧鐘與時鐘功能之電子時鐘(進階版)

接腳:

P11B(1~4)	：分別接微控板的 2 、3 、4 、5 腳
P11A(a-g)	：分別接微控板的 6 、7 、8 、9 、10 、11 、12 腳
P14	：接微控板的 13 腳
DD	：接微控板的 A0 腳
P5-1~P5-3	：分別接微控板的 A1 、A2 、A3 腳
P6-5~P6-6	：分別接微控板的 A4 、A5 腳
P6-1	：接 GND(低態動作)

功能:

(1)具有調整時、分的功能

(2)每隔 5 分鐘從蜂鳴器發出聲音

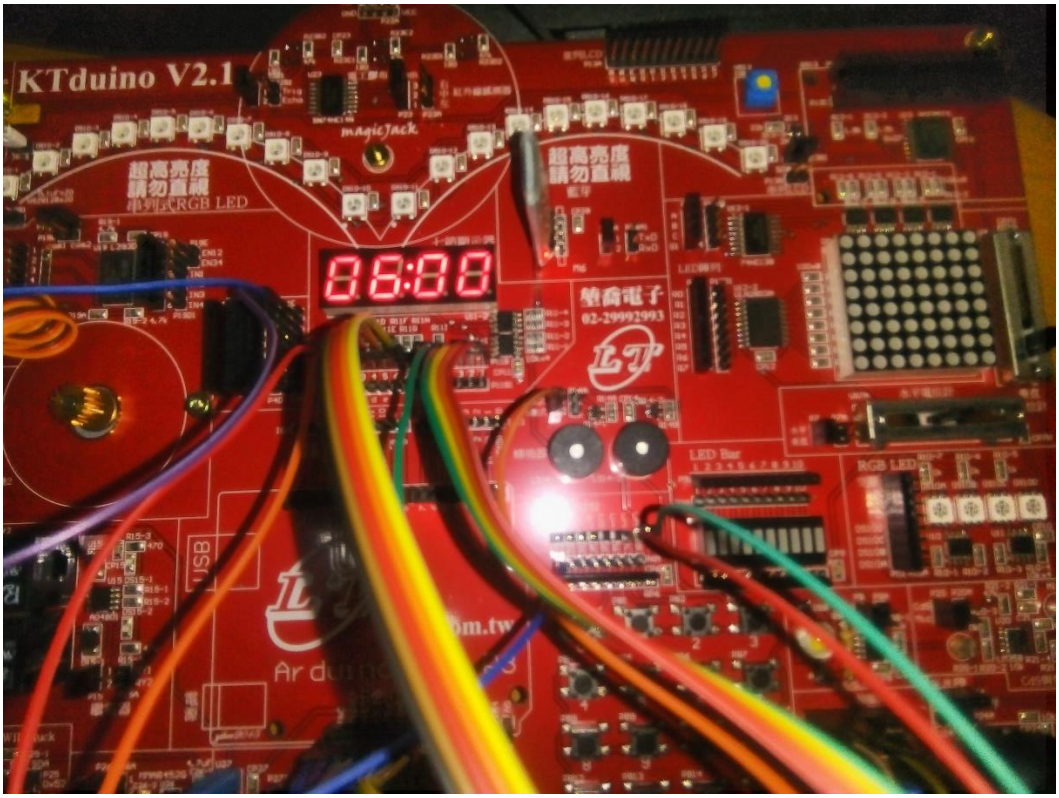
(3)每小時發出一段簡短音樂報時

(4)設定鬧鐘

(5)重複播放一段音樂，直到關掉鬧鐘功能

2. 成品

圖表 74 數位時鐘(進階題)成品-UNO 板



圖表 75 數位時鐘(進階題)成品- KTduino 實驗版

3. Pseudocode

```
Declare pins

void setup(){
    Setup Pins;
}

void loop(){
    if(Setting Alarm){
        adjustment_alarm();
    }else{
        if(adjustment){
            adjustment();
        }else{
            timing();
        }
    }
}

void timing(){
    while(it is timing model){
        if(time to AND it is alarm){
            play one part
        }
        run clock
    }
}
```

圖表 76 進階題虛擬碼

構思說明:

將程式分為變數的宣告及定義、初始化系統和主要運行程式進行思考。
變數的宣告及定義: 簡單化為接腳宣告，避免資料宣告以提升安全性。
初始化系統: 設定接腳。
主要運行程式: 判斷模式，然後執行期。
核心技術: 同步(synchronization)技術理念(用於邊播音樂邊執行計時)

4. 程式碼

主程式

```
// Include library(s):
#include "Clock.h" //My library : 處理時鐘運作作業(詳情請看header檔)
//-------------------------------------------
```

圖表 77 主程式-引入函數庫(進階題)

```
// Declare variables:

// Output pins:
const int scan[] = {2, 3, 4, 5};                // 宣告掃描信號接腳
const int SEG[] = {6, 7, 8, 9, 10, 11, 12};     // 宣告顯示信號接腳(g~a)
const int BZ = 13;                              // 宣告蜂鳴器接腳
const int DD = A0;                              // 宣告閃秒LED接腳(":")
// ===========================================

// Input pins
const int DIPSW[] = {A1, A2, A3};       // 宣告指撥開關接腳
const int PB[] = {A4, A5};              // 宣告按鍵接腳
// ===========================================

// Data:
Clock* clock;   // 宣告時鐘物件
// ===========================================

//-------------------------------------------
```

圖表 78 主程式-接腳宣告及時鐘物件宣告(進階題)

```
// Initializing system
void setup(){
    clock = new Clock(SEG, scan, PB, 2, DIPSW, 3, BZ, DD);  //Instance clock
}
//-------------------------------------------
```

圖表 79 主程式-系統初始化(實例時鐘物件) (進階題)

```
// Main function
void loop(){
    clock->runWork();// run clock
}
//-----------------------------------------------
```

圖表 80 主程式-主核心運作程式(進階題)

鬧鐘物件

```
 7   #ifndef CLOCK_H
 8   #define CLOCK_H
 9
10   // include librarys:
11   #include "Arduino.h"                // Ardruino base library
12   #include "FourDigitSevenSegment.h"  // My Library : 處理四段七節顯示器的部分作業(詳情請看header檔)
13   //-----------------------------------------
14   using namespace std;
15
16   class Clock{
17   public: ···
73   private: ···
153  protected:
154  };
155  #endif // !CLOCK_H
```

圖表 81 鬧鐘物件的標頭檔(架構及引用的函數庫)

```
// Constructor: input(SEG pins(a~g), scanner pins, PB pins, DIPSW pins, and buzzer pin)
Clock(const int* SEG, const int* scan, const int* PB, int PB_length, const int* DIPSW, int DIPSW_length, const int buzzer, const int DD);
// ========================================

// Deconstructor
~Clock();
// ========================================
```

圖表 82 鬧鐘物件的標頭檔-Public(建構子及解構子)

```
// Functions:

// Setting Functions:

// Setting DIP switch pins
void set_DIPSW_pins(const int* pins, int length);

// Setting PB pins
void set_PB_pins(const int* pins, int length);

// Setting Buzzer pins
void set_Buzzer_pin(const int pin);

// Setting DD pins
void set_DD_pin(const int pin);
// - - - - - - - - - - - - - - - - - - - - - - -

// Clock work Functions:

// 調整模式
void adjustment();

// 鬧鐘調整模式
void adjustment_alarm();

// 計時模式
void timing();
// - - - - - - - - - - - - - - - - - - - - - - -

// The others functions:

// beep beep counts times
double beep(int pin, int counts);

// 讀取指撥開關
void DIP_swtich();
// - - - - - - - - - - - - - - - - - - - - - - -

// 讀取鬧鐘模式是否開啟
void isAlarm();
// - - - - - - - - - - - - - - - - - - - - - - -

// Run
void runWork();
// - - - - - - - - - - - - - - - - - - - - - - -

// ===========================================
```

圖表 83 鬧鐘物件的標頭檔-Public(函數)

```
// Functions:

// Time work Functions:

// Add one hour
void add_hour(int* m_time/*m_time: time array*/);

// Subtract one hour
void subtract_hour(int* m_time/*m_time: time array*/);

// Add one minute
void add_minute(int* m_time/*m_time: time array*/);

// Subtract one minute
void subtract_minute(int* m_time/*m_time: time array*/);
// - - - - - - - - - - - - - - - - - - - - - - -

// Setup functions:

// Setup all PB
void setupPB();

// Setup all dip switch
void setupDIPSW();

// Setup buzzer
void setupBuzzer();

// Setup DD
void setupDD();
// - - - - - - - - - - - - - - - - - - - - - - -

// ========================================
```

圖表 84 鬧鐘物件的標頭檔-Private (函數)

```
// Pins:
int BZ;      // 宣告蜂鳴器接腳
int DD;      // 閃秒LED接腳(":")
int* PB;     // 按鍵接腳
int* DIPSW; // 指撥開關接腳
// ========================================
```

圖表 85 鬧鐘物件的標頭檔-Private(接腳變數)

```
// Base data:
FourDigitSevenSegment* FDSS;    // 四段七節顯示器處理物件
int disp[4] = {0, 0, 0, 0};     // 顯示緩衝區陣列(buffer)
int goal[4] = {0, 0, 6, 0};     // 鬧鈴目標時間，預設06:00
unsigned long long x0,x1,x2;    // 時間變數
int phase0 = 2;                 // 工作旗標(flag)：phase0=0計時模式, phase0=1調整模式, phase0=2鬧鐘調整模式
boolean alarm = true;           // 工作旗標(flag)：alarm=false鬧鐘模式開(ON), alarm=true鬧鐘模式關(OFF)
double seconds=0;               // 秒變數
bool sec = false;               // 閃秒LED旗標(flag)
// ========================================

// Data of music:
// beat
double beat[49] = {
    1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2,
    1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 4,
    1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2,
    1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 4
};
// - - - - - - - - - - - - - - - - - - - - -

// tones
int tones[49] = {
    784, 659, 659, 698, 587, 587, 523, 587, 659, 698, 784, 784, 784,
    784, 659, 659, 698, 587, 587, 523, 659, 784, 784, 659,
    587, 587, 587, 587, 587, 659, 698, 659, 659, 659, 659, 659, 698, 784,
    784, 659, 659, 698, 587, 587, 523, 659, 784, 784, 523
};
// - - - - - - - - - - - - - - - - - - - - -

// tempo
int tempo = 300;
// - - - - - - - - - - - - - - - - - - - - -

// size of tones
int tones_size = sizeof(tones)/sizeof(tones[0]);
// - - - - - - - - - - - - - - - - - - - - -
// ========================================
```

圖表 86 鬧鐘物件的標頭檔-Private(資料變數)

```
// Constructor: input(SEG pins(a~g), scanner pins, PB pins, DIPSW pins, and buzzer pin)
Clock::Clock(const int* SEG, const int* scan, const int* PB, int PB_length, const int* DIPSW, int DIPSW_length, const int buzzer, const int DD){
    FDSS = new FourDigitSevenSegment(SEG, scan);    //Instance FDSS

    this->PB = new int[ PB_length ];        // Instance PB array
    this->DIPSW = new int[ DIPSW_length ];  // Instance DIPSW array

    // Setting pins:
    set_PB_pins(PB, PB_length );
    set_DIPSW_pins(DIPSW, DIPSW_length );
    set_Buzzer_pin(buzzer);
    set_DD_pin(DD);
    // ========================================
}
//----------------------------------------
```

圖表 87 鬧鐘物件的建構子

```
// Deconstructor
Clock::~Clock(){

}
//------------------------------------------------
```

圖表 88 鬧鐘物件的解構子

```
// Setting DIP switch pins
void Clock::set_DIPSW_pins(const int* pins, int length){
    for(int i=0; i<length; ++i){
        DIPSW[i] = pins[i];
    }

    setupDIPSW();
}
//----------------------------------------
```

圖表 89 鬧鐘物件的設定開關接腳函數

```
// Setting PB pins
void Clock::set_PB_pins(const int* pins, int length){
    for(int i=0; i<length; ++i){
        PB[i] = pins[i];
    }

    setupPB();
}
//----------------------------------------
```

圖表 90 鬧鐘物件的設定按鈕接腳函數

```
// Setting Buzzer pins
void Clock::set_Buzzer_pin(const int pin){
    BZ = pin;

    setupBuzzer();
}
//-------------------------------------------
```

圖表 91 鬧鐘物件的設定蜂鳴器接腳函數

```
// Setting DD pins
void Clock::set_DD_pin(const int pin){
    DD = pin;

    setupDD();
}
//-------------------------------------------
```

圖表 92 鬧鐘物件的設定":"LED 接腳函數

```
// 調整模式
void Clock::adjustment(){
    while(phase0==1){
        seconds = 0;                // 秒數歸零
        digitalWrite(DD, false);    // 閃秒固定亮著

        // 時遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[0])){    // 若PB[0]被按下(時遞增)
            add_hour(disp);
        }

        // 分遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[1])){    // 若PB[1]被按下(分遞增)
            add_minute(disp);
        }

        FDSS->scanner(25, disp);    // 連續掃描25週(100ms，每秒約10次)

        DIP_swtich();   // 讀取指撥開關
    }
}
//-------------------------------------------
```

圖表 93 鬧鐘物件的調整模式運行函數

```
// 鬧鐘調整模式
void Clock::adjustment_alarm(){
   while(phase0==2){
        seconds = 0;                // 秒數歸零
        digitalWrite(DD, false);    // 閃秒固定亮著

        // 時遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[0])){    // 若PB[0]被按下(時遞增)
            add_hour(goal);
        }

        // 分遞增調整(按鍵為低態動作)
        if(!digitalRead(PB[1])){    // 若PB[1]被按下(分遞增)
            add_minute(goal);
        }

        FDSS->scanner(25, goal);    // 連續掃描25週(100ms，每秒約10次)
        FDSS->closeAll();           // Close all LED(Prevent afterimage)

        DIP_swtich();   // 讀取指撥開關
   }
}
//-----------------------------------------
```

圖表 94 鬧鐘物件的鬧鐘調整模式運行函數

```
// 計時模式
void Clock::timing(){
    int index = 0;              // index of tones
    boolean activity = false;   // flag (Can it be an alarm?)

    while(phase0==0){
      if(!activity){
        FDSS->scanner(2, disp); // 連續掃描2週(8ms)
      }else{
        FDSS->closeAll();       // 關閉鬧鐘顯示
      }

        x1 = millis();          // 查詢時間
        x2=x1-x0;               // 計算時間差

        if(x2>=500){    // 0.5秒時間差
            x0=x1;      // 記錄時間

            // 切換閃秒:
            sec=!sec;
            digitalWrite(DD, sec);
            // - - - - - - - - - - - - - - - - - - - - -
```

圖表 95 鬧鐘物件的計時模式運行函數 part A(秒差紀錄片段)

```
        // 處理時間
        if(sec){
            if(++seconds>=60){       // 秒數加1
                seconds-=60;         // 若秒數滿60則歸零
                if(disp[0]==9){      // 若分之個位數滿9
                    disp[0]=0;       // 則分之個位數歸零

                    if(disp[1]==5){ // 若分之十位數滿5
                        disp[1]=0;  // 則分之十位數歸零

                        add_hour(disp);     // Add one hour
                    }else{
                        ++disp[1];  //否則分之十位數加1
                    }

                }else{
                    ++disp[0];      //否則分之個位數加1
                }
            }
        }

    }
    isAlarm();
```

圖表 96 鬧鐘物件的計時模式運行函數 part B(計時片段)

```
        if(alarm){  // It is alarm

            // Check time
            if(disp[0]==goal[0] && disp[1]==goal[1] && disp[2]==goal[2] && disp[3]==goal[3]){
                activity = true;
            }


            if(activity){                   // If time to ,play one beat of music

                FDSS->closeAll();           // Close all LED(Prevent afterimage)

                if(index >= tones_size){    // Checking if index is over range
                    index = 0;              // if index is over range, make index to 0
                }

                if(tones[index] != 0){                          // if the tone is not pause.
                    tone(BZ, tones[index], beat[index]*tempo);      // play one tone with one beat
                }

                seconds += ( ((double)tempo) /1000 )*beat[index];   // Computing fix time

                delay(beat[index]*tempo);   // Waitting one beat
                noTone(BZ);                 // Close buzzer (Prevent afterimage)
                ++index;                    // Moving index to next one
            }

        }else{                              // If it is not alarm, it will not play music
            activity = false;
        }

        DIP_swtich();   // 讀取指撥開關
    }
}
//-----------------------------------------
```

圖表 97 鬧鐘物件的計時模式運行函數 part C(音樂同步作業片段)

```
// 讀取指撥開關
void Clock::DIP_swtich(){
    if(digitalRead(DIPSW[1])){          // 讀取指撥開關(DIPSW[1]) : 鬧鐘調整模式
        phase0 = 2;
    }else if(digitalRead(DIPSW[0])){    // 讀取指撥開關(DIPSW[0]) : 調整模式
        phase0 = 1;
    }else{                              // 讀取指撥開關(DIPSW[0]) : 計時模式
        phase0 = 0;
    }
}
//-------------------------------------------
```

圖表 98 鬧鐘物件的讀取指撥開關函數

```
// 讀取鬧鐘模式是否開啟
void Clock::isAlarm(){
    if(digitalRead(DIPSW[2])){  // It is alarm.
        alarm = true;
    }else{                      // It is not alarm.
        alarm = false;
    }
}
//-------------------------------------------
```

圖表 99 鬧鐘物件的讀取鬧鈴開關函數

```
// Run
void Clock::runWork(){
    DIP_swtich();           // 讀取指撥開關
    adjustment_alarm();     // 鬧鐘調整模式
    adjustment();           // 調整模式
    timing();               // 計時模式
}
// - - - - - - - - - - - - - - - - - - - - - - - -
```

圖表 100 鬧鐘物件的主運行函數

```
// Add one hour
void Clock::add_hour(int* m_time/*m_time: time array*/){
    if(m_time[3]==2 && m_time[2]==3){    // 若已23時
        m_time[3]=0;                     // 時之十位數歸零
        m_time[2]=0;                     // 時之個位數歸零
    }else if(m_time[2]==9){              // 若時之個位數等於9
        m_time[2]=0;                     // 時之個位數歸零
        ++m_time[3];                     // 時之十位數遞增(進位)
    }else{
        ++m_time[2];                     // 否則時之個位數遞增
    }
}
//-------------------------------------------
```

圖表 101 鬧鐘物件的加一小時函數

```
// Subtract one hour
void Clock::subtract_hour(int* m_time/*m_time: time array*/){
    if(m_time[3]==0 && m_time[2]==0){    // 若已00時
        m_time[3]=2;                     // 時之十位數調整為2
        m_time[2]=3;                     // 時之個位數調整為3
    }else if(m_time[2]==0){              // 若時之個位數等於0
        m_time[2]=9;                     // 時之個位調整為9
        --m_time[3];                     // 時之十位數遞減(借位)
    }else{
        --m_time[2];                     // 否則時之個位數遞減
    }
}
//-------------------------------------------
```

圖表 102 鬧鐘物件的減一小時函數

```
// Add one minute
void Clock::add_minute(int* m_time/*m_time: time array*/){
    if(m_time[1]==5 && m_time[0]==9){    // 若已59分
        m_time[1]=0;                     // 分之十位數歸零
        m_time[0]=0;                     // 分之個位數歸零
    }else if(m_time[0]==9){              // 若分之個位數等於9
        m_time[0]=0;                     // 分之個位數歸零
        ++m_time[1];                     // 分之十位數遞增(進位)
    }else{
        ++m_time[0];                     // 否則分之個位數遞增
    }
}
//-------------------------------------------
```

圖表 103 鬧鐘物件的加一分鐘函數

```
// Subtract one minute
void Clock::subtract_minute(int* m_time/*m_time: time array*/){
    if(m_time[1]==0 && m_time[0]==0){    // 若已00分
        m_time[1]=5;                     // 分之十位數調整為5
        m_time[0]=9;                     // 分之個位數調整為9
    }else if(m_time[0]==0){              // 若分之個位數等於0
        m_time[0]=9;                     // 分之個位調整為9
        --m_time[1];                     // 分之十位數遞減(借位)
    }else{
        --m_time[0];                     // 否則分之個位數遞減
    }
}
//---------------------------------------------
```

圖表 104 鬧鐘物件的減一分鐘函數

```
// beep beep counts times
double Clock::beep(int pin, int counts){
    double delayTime = 0;
    for(int i=0; i<counts; ++i){     // 執行counts次
        tone(pin, 1000, 100);        // 發聲(1kHz,0.1秒)
        delay(100);                  // 靜音(0.1秒)
        delayTime += 0.2;            // Computing fix time
    }

    return delayTime;                // return fix time
}
//---------------------------------------------
```

圖表 105 鬧鐘物件的 beep beep 叫函數

```
// Setup all PB
void Clock::setupPB(){
    for(int i=0; i<(sizeof(PB)/sizeof(PB[0])); ++i){
        pinMode(PB[i], INPUT);
    }
}
//---------------------------------------------
```

圖表 106 鬧鐘物件的按鈕接腳設置函數

```
// Setup all dip switch
void Clock::setupDIPSW(){
    for(int i=0; i<(sizeof(DIPSW)/DIPSW[0]); ++i){
        pinMode(DIPSW[i], INPUT);
    }
}
//---------------------------------------------
```

圖表 107 鬧鐘物件的開關接腳設置函數

```
// Setup buzzer
void Clock::setupBuzzer(){
    pinMode(BZ, OUTPUT);
}
//-------------------------------------------
```

圖表 108 鬧鐘物件的蜂鳴器接腳設置函數

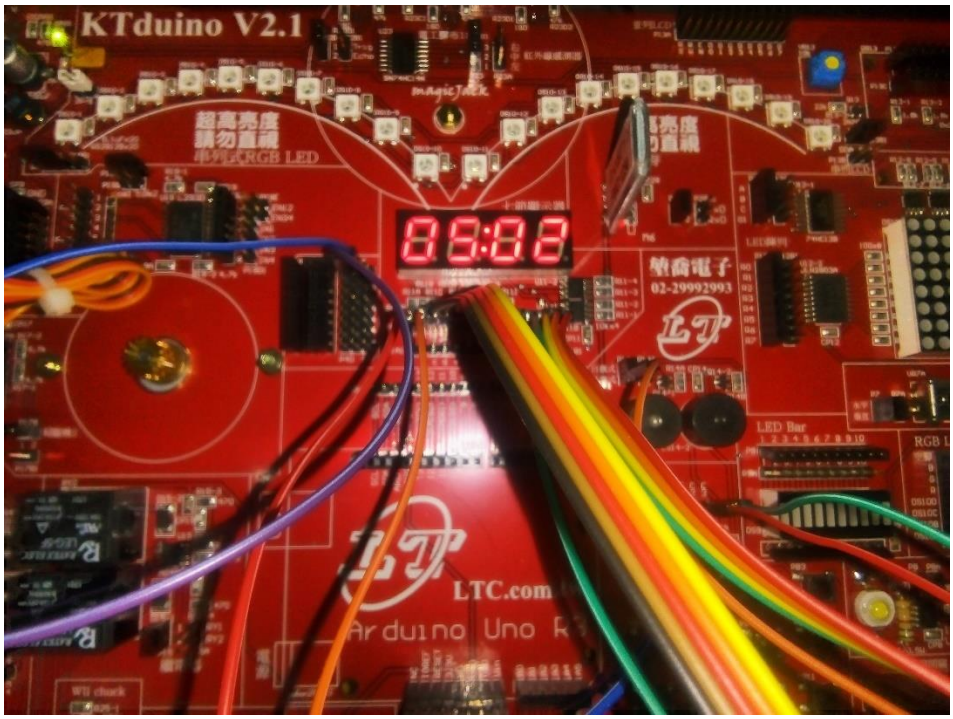
```
// Setup DD
void Clock::setupDD(){
    pinMode(DD, OUTPUT);
}
//-------------------------------------------
```

圖表 109 鬧鐘物件的":"LED 接腳設置函數

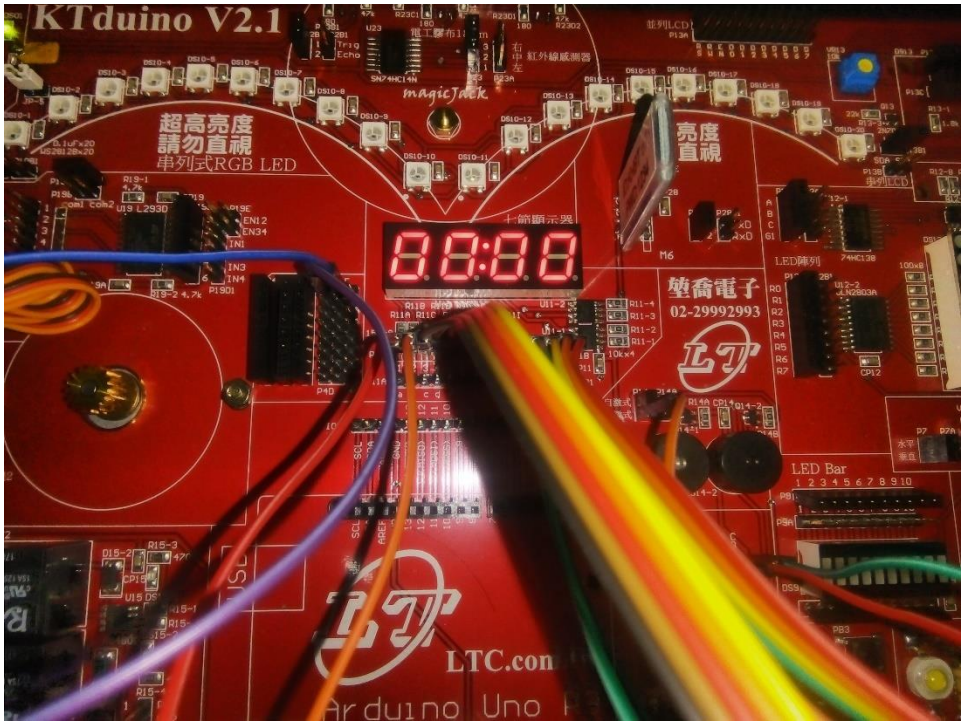
四段七節顯示器物件

同基本題(點此查看)

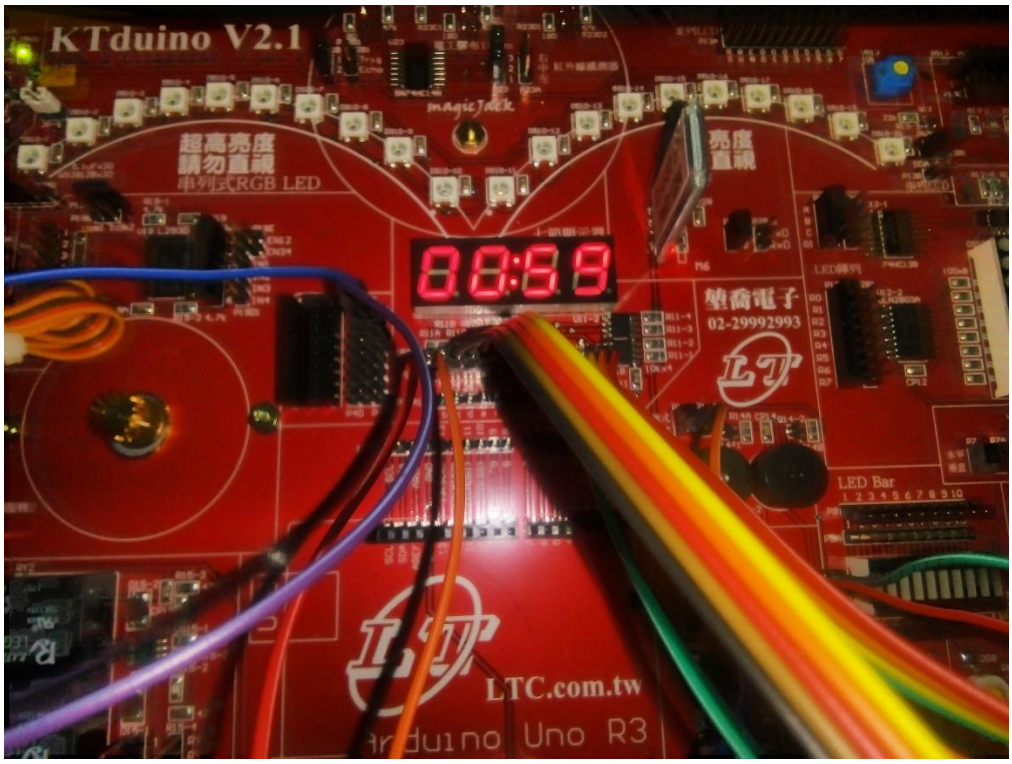
5. 靜態展示



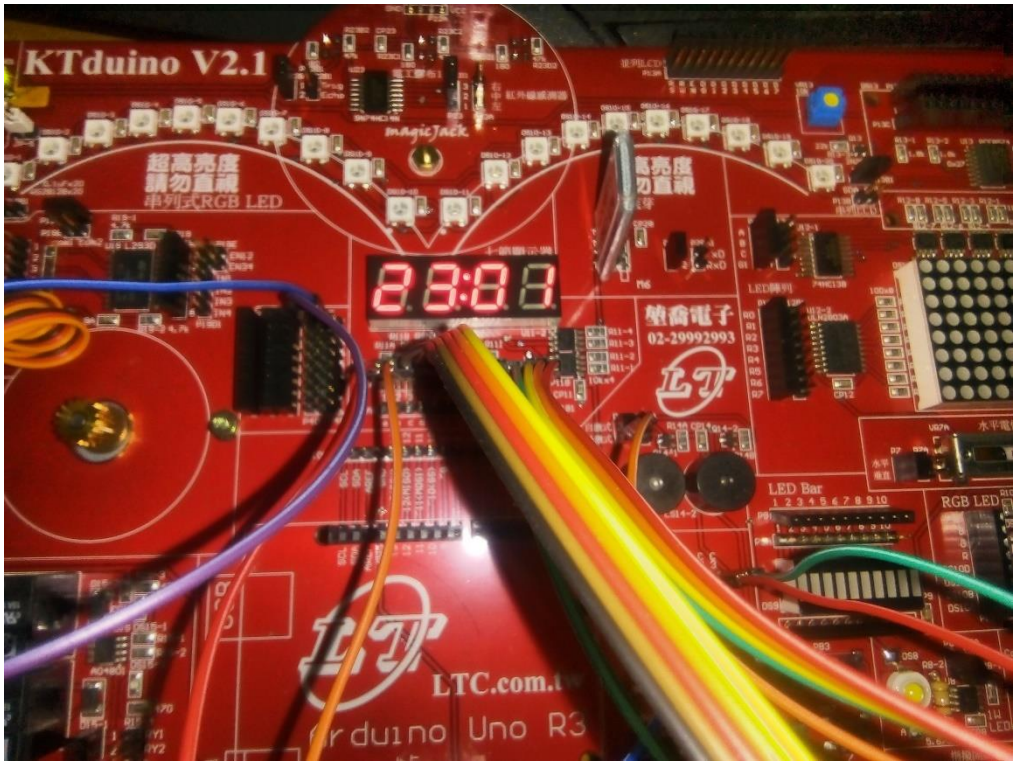
圖表 110 鬧鈴時分的調整(預設 06:00)



圖表 111 時間設定模式(預設 00:00)



圖表 112 時間設定模式分的調整



圖表 113 時間設定模式時的調整

6. 動態展示

YouTube

Google Cloud

四、 學習心得

在這一次的專案設計上我採用了物件式的設計模式，使得程式具有可移植姓，例如:基本題和進階題都利用的四段七節顯示器物件就是一個例子。而在 Clock 物件在原始設計理念上他會成為鬧鐘物件的父物件，但是因為已經做完才想起來所以就放棄了。而在進階題中因為音樂要持續播放且隨時能切斷及時間系統要能夠保持運行的單晶片前提，所以我參考了網路同步技術(synchronization)的概念來保持音樂完整性和時間的運行完整性使得成果能較接近理想。