中央大學資訊工程學系

106-2 嵌入式系統設計
期末專題報告

智慧時鐘

Smart Clock

指導教授：陳慶瀚

學生：陳韋佑 (106522033)

林後昇 (106522054)

2018 年 6 月 22 日

摘要

目前台灣市面上還沒有自動校時時鐘，經過市場調查發現日本早已有時間矯正電波來接收本地的正確時間，但台灣還沒有這方面的設施與設備，因此本專題主要目的在 MCU(STM32)上與 Wi-Fi 模組(NodeMCU)建立一個網路路由再從網路校時系統(NTP)獲得當地的正確時刻或者利用可連網設備加入 IoT 網域底下設定時間，最後再透過馬達去驅動翻頁數字盤來實驗顯示時間的功能。

本專題進行過程首先了解 Wi-Fi 模組、結構組成及網路封包原理，並研究如何從 NodeMCU 上開 Http Server， 從手機上的網頁端發送 HTTP GET Request 過去，最後再實現馬達驅動把翻頁數字盤翻轉至正確的時間，來實現嵌入式系統與物聯網(IoT)的智慧應用。

目錄

第 1 章　前言 5

第 2 章　原理與方法 6

2.1　網路時間協定 6

2.2　翻頁鐘 7

2.3　UART 介面 7

第 3 章　硬體架構 8

3.1　架構圖 8

3.2　微控制器 8

3.3　Wi-Fi 模組 9

3.4　磁簧繼電器 9

第 4 章　系統設計 10

4.1　Grafcet (STM32) 10

4.2　STM32 核心程式碼 10

4.3　NodeMCU 核心程式碼 16

第 5 章　系統整合驗證 20

第 6 章　結論與心得 21

第 7 章　Reference 21

圖目錄

圖 2.1 NTP 伺服器時鐘校正原理示意圖 6
圖 2.2 NTP 階層架構示意圖 6
圖 2.3 實驗用翻頁鐘圖 7
圖 2.4 UART Frame 格式圖 7
圖 3.1 智慧時鐘架構圖 8
圖 3.2 NodeMCU 微控制器圖 8
圖 3.3 ESP8266 無線 Wi-Fi 模組 9
圖 3.4 磁簧繼電器原理圖 9
圖 3.5 MD-5 磁簧繼電器圖 9
圖 4.1 STM32 Grafcet 流程圖 10
圖 5.1 Smart Clock 接線圖 20
圖 5.2 Smart Clock 產品完成圖 20

第 1 章　前言

智慧時鐘主要的特色就是有物聯網(Internet of Thing, IoT)、智慧自動校時，這些特色可讓它有更多的應用，時鐘不再只是一個看時間的產品，更有利於使用者在不用智慧型手機的情況下馬上掌握資訊。嵌入式系統常被用於高效控制其他常見的裝置，被嵌入的系統通常包含數位硬體與機械部件等，可以智慧的處理各式各樣的運算狀況；物聯網是在電腦網路的基礎上，利用藍芽、Wi-Fi 等技術，創造一個覆蓋世界上萬物的“Internet of Thing”。

利用這兩項技術的優點，我們可以讓智慧時鐘有更多應用的可能，像是蒐集天氣資訊、量測環境溫溼度、蒐集手機的通知訊息等，這些應用可以使我們的生活更方便更美好。

本專題基於此一原理，設計出可以使用具有 Wi-Fi 模組的開發板 NodeMCU 和嵌入式開發版 STM32 結合，使它有基本的時間自動校正功能與網路路由功能，讓任何有 Wi-Fi 功能的設備可以加入此智慧時鐘底下做設定，之後再以這些為基礎做出更多方便的應用。

第 2 章　原理與方法

2.1　網路時間協定

網路時間協定(Network Time Protocol, NTP)是由電腦系統之間通過封包交換進行時鐘同步的一個網路協議。NTP 最終目的是希望所有參與的電腦跟協調世界時(UTC)時間同步的誤差到幾毫秒以內。

該協定通常描述為一種主從式架構，但它也可以用在對等式網路中，對等體雙方認定另一方為潛在的時間源。傳送與接收採用用戶資料通報協定(UDP)的通訊埠 123 實現。這可以使用廣播或者多播，其中用戶端可以在往返交換後被動地監聽時間更新。

NTP 伺服器以階層式架構形成時間同步體系，如*圖 2.2* 所示，位於第一階層的伺服器會同步到國家標準時間，第二階伺服器則透過第一階伺服器間接同步到國家標準時間，每台伺服器以本身時鐘來維持某精度時間，並自行式適當校時週期主動向上一層發出請求。為了時間同步的穩健性，最好有三條以上的同步路徑，如*圖 2.1* 所示，每條路徑的時鐘過濾器會從最近幾次的時鐘偏差值中挑選出最佳數值作為輸出。

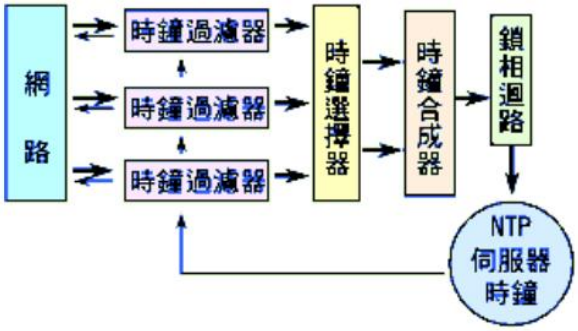

圖 2.1　NTP 伺服器時鐘校正原理示意圖

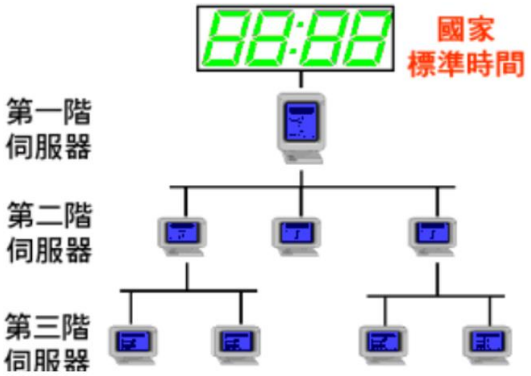

圖 2.2　NTP 階層架構示意圖

2.2 翻頁鐘

翻頁鐘是一種利用翻轉數字板來顯示時刻的「機械式數位時鐘」。數位並不是指裡面是由 IC 控制，而是指顯示的方式是由數字方式來表現；以時針、分針顯示時間的，則被稱為「類比時鐘」。

翻頁鐘的構造有兩顆馬達控制兩個葉片，葉片裡有六十小頁顯示時間，左邊葉片 24 小時轉一圈(0 到 11 小時有兩小片相同，12 到 23 小時有三小片相同)，右邊葉片 60 分鐘轉一圈。當馬達發出一個正電跟負電的訊號就會驅動葉片落下，基於以上原理就可以使用 STM32 來模擬出一個時鐘。

圖 2.3 實驗用翻頁鐘圖

2.3 UART 介面

UART (Universal Asynchronous Receiver / Transmitter)是通用的非同步收發器(非同步串列通信口)，它包括了 RS232、RS449 和 RS485 等介面標準規範和匯流排標準規範。

UART 的工作就是從 CPU 一次接收 8 bits 資料，然後將資料 1 次 1 bit 的送往周邊設備；同時，UART 還可以接收周邊設備傳送過來的資料，當組成 8 bits 時，再將資料送往 CPU，資料格式如*圖 2.4* 所示：

圖 2.4 UART Frame 格式圖

第 3 章　硬體架構

3.1　架構圖

本章節會介紹智慧時鐘架構圖，如圖 3.1 所示，以下藍色線為 NodeMCU 和 STM32 接線方式稱為內網，黑色線為往 Internet 的線稱為外網。首先，我們須在 NodeMCU 讓 ESP8266 可以連到我們 Router 並成功的存取 Internet 資源；接下來，NodeMCU 需發出自己的 Wi-Fi 訊號讓任何 Device 可以加入智慧時鐘底下，控制時鐘或者讀取其他 Sensor 的資料。

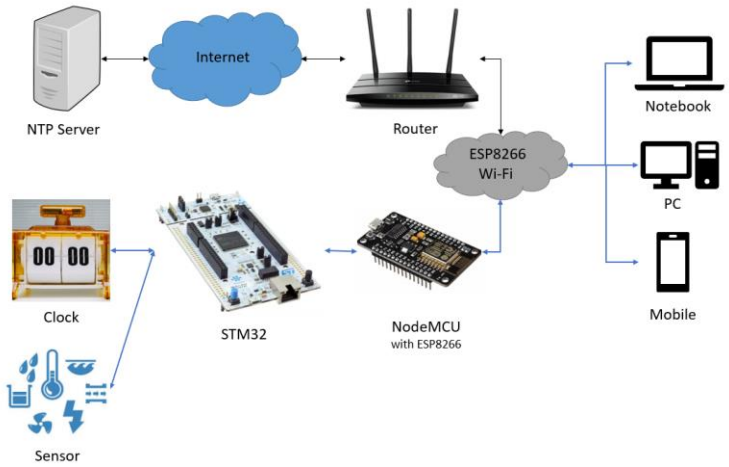

圖 3.1　智慧時鐘架構圖

3.2　微控制器

採用 NodeMCU 開發板是由安信可科技(Ai-Thinker)公司基於樂鑫(Eressif)公司的 ESP8266 模組做出的開發板，CPU 使用 32-bit 的 Tensilica Xtensa LX3，有 64K Boot ROM、64K Instruction RAM 以及 96K Data RAM，也具有 Wi-Fi 8011 b/g/n 2.4GHz 可以設定為 AP、Station 或 Station + AP 等各種網路應用模式，另外它是一個開源的物聯網平台，使用 Lua 腳本語言撰寫，因此具有強大功能。

其他硬體周邊提供 Timers、Deep sleep mode、JTAG debugging，GPIO Pins 最多有 13 支接點，提供了 PWM、I2C 和 ADC 等介面，因此這顆板子雖然小旦功能非常強大，能夠提供開發者快速的弄出物聯網平台。

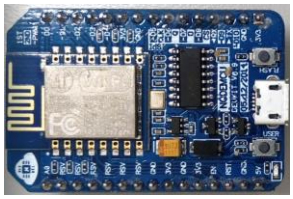

圖 3.2　NodeMCU 微控制器圖

3.3 Wi-Fi 模組

許多開發板都沒有 Wi-Fi 功能，因此需要具有 Wi-Fi 的模組與網路連線，再透過 ADC、PWN 等腳位與開發板溝通。本專題使用 ESP8266 模組，如*圖 3.3* 所示，具有易開發又價錢便宜等優勢，例如可以扮演運作 Web Server 角色隨時記錄感測器的數值，也可以扮演運作 Web Client 角色蒐集網路的資料再傳回開發板。

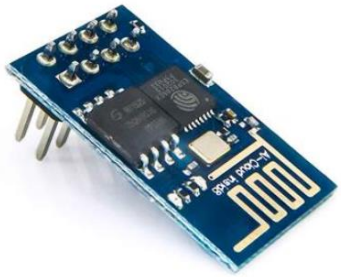

圖 3.3　ESP8266 無線 Wi-Fi 模組

3.4 磁簧繼電器

磁簧繼電器是以線圈產生磁場將磁簧管作動之繼電器，為一種線圈感測裝置，如*圖 3.4* 所示。磁簧繼電器具有小型尺寸、輕量、反應速度快等優點，所以本專題使用 SUN HOLD 所開發的 MD-5 磁簧繼電器，如*圖 3.5* 所示，來控制智慧時鐘的馬達。

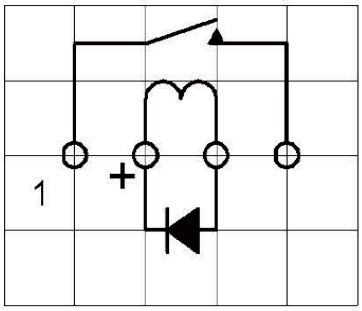

圖 3.4　磁簧繼電器原理圖

圖 3.5　MD-5 磁簧繼電器圖

第 4 章　系統設計

4.1　Grafcet (STM32)

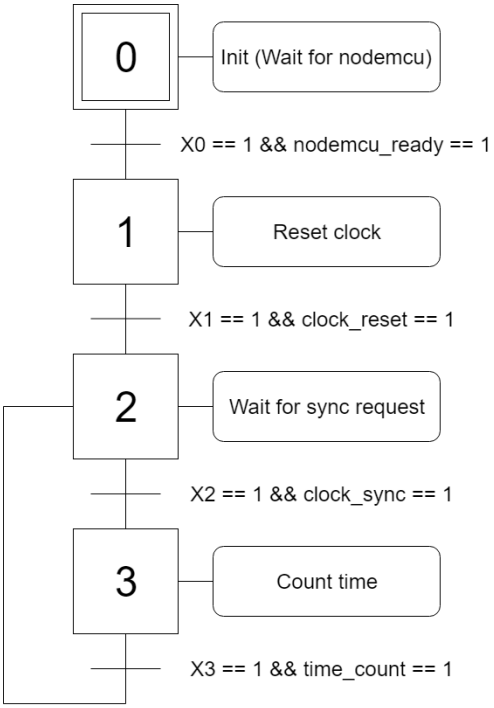

圖 4.1　STM32 Grafcet 流程圖

4.2　STM32 核心程式碼

Source Code: https://github.com/wy56/Smart-Clock/blob/master/Keil/main.cpp

```
#include "mbed.h"
#include "pinmap.h"

AnalogIn pinHourIn(A1);
AnalogIn pinMinIn(A2);
DigitalOut out_min(PA_4);
DigitalOut out_hour(PA_5);
Serial pc(USBTX, USBRX);
Serial nodemcu(PC_6,PC_7);

int i, v_out1, v_out2;
int X0, X1, X2, X3;
int timeout_c = 0;
int now_hour = 0, now_min = 0;
```

```
int gra_nodemcu_ready, gra_clock_reset, gra_clock_sync, gra_clock_count;
float v_hour, v_min;
double now_sec = 0;
char buffer[64];

void grafcet();
void action();
void wait_clock(double);
void check_clock();
void hour_click(void);
void min_click(void);
void reset_clock(void);
void set_clock(int, int, int);

int main()
{
    nodemcu.baud(9600);
    pc.printf("Start Smart Clock!\n");
    X0=1;
    while(1){
        grafcet();
    }
}

void grafcet(){
    if(X0==1 && gra_nodemcu_ready == 1)X0=gra_nodemcu_ready=0,X1=1;
    else if(X1==1 && gra_clock_reset == 1)X1=gra_clock_reset=0,X2=1;
    else if(X2==1 && gra_clock_sync == 1)X2=gra_clock_sync=0,X3=1;
    else if(X3==1 && gra_clock_count == 1)X3=gra_clock_count=0,X2=1;
    pc.printf("X0 = %d, X1 = %d, X2 = %d, X3 = %d\r\n",X0,X1,X2,X3);
    action();
}

void action(){
    if(X0) {
        if (nodemcu.readable()) {
            nodemcu.scanf("%s", buffer);
            pc.printf("%s\n\r", buffer);
```

```
        if(strcmp(buffer,"192.168.4.1")==0) {
            pc.printf("gra_nodemcu_ready\n\r");
            gra_nodemcu_ready = 1;
        }
    }
}
else if(X1) {
    reset_clock();
    pc.printf("gra_clock_reset\n\r");
    gra_clock_reset=1;
}
else if(X2 ){
    int time_out_max = 10000;
    if (nodemcu.readable()) {
        int h;
        int m;
        int s;
        nodemcu.scanf("%s", buffer);
        pc.printf("%s\n\r", buffer);
        if(buffer[2]==':'&&buffer[5]==':'){
            pc.printf("Time:%s\n\r", buffer);
            sscanf(buffer, "%d:%d:%d", &h,&m,&s);
            pc.printf("Time2:%d %d %d\n\r", h,m,s);
        }
        if(buffer[3]==':'&&buffer[6]==':'){
            pc.printf("Time:%s\n\r", buffer);
            sscanf(buffer, "%d:%d:%d", &h,&m,&s);
            h=h%100;
            pc.printf("Time2:%d %d %d\n\r", h,m,s);
            if(h<24&&m<60&&s<60){
                set_clock(h, m, s);
            }
        }
    }
    pc.printf("gra_clock_sync\n\r");
    gra_clock_sync = 1;
}
else if(X3) {
```

```
        now_sec += 1;
        check_clock();
        pc.printf("gra_clock_count\n\r");
        gra_clock_count=1;
    }
}

void wait_clock(double sec) {
    wait(sec);
    now_sec += sec;
    if(now_sec >= 60.f) {
        now_sec -= 60;
        now_min += 1;
        min_click();
        if(now_min >= 60) {
            hour_click();
            hour_click();
            now_min -= 60;
            now_hour = (now_hour + 1) % 24;
            if(now_hour >= 12) {
                hour_click();
            }
        }
    }
}

void check_clock(){
    if(now_sec >= 60.f) {
        now_sec -= 60;
        now_min += 1;
        min_click();
        if(now_min >= 60) {
            hour_click();
            hour_click();
            now_min -= 60;
            now_hour = (now_hour + 1) % 24;
            if(now_hour >= 12) {
                hour_click();
```

```
            }
        }
    }
}

void hour_click() {
    out_hour = 1;
    wait_clock(0.05);
    out_hour = 0;
    wait_clock(0.1);
}

void min_click() {
    out_min = 1;
    wait_clock(0.05);
    out_min = 0;
    wait_clock(0.1);
}

void reset_clock() {
    while(v_hour<30) {
        v_hour = pinHourIn  * 33.f;
        hour_click();
    }

    while(v_hour>30) {
        hour_click();
        v_hour = pinHourIn  * 33.f;
    }

    while(v_min<30) {
        v_min = pinMinIn  * 33.f;
        min_click();
    }

    while(v_min>30) {
        min_click();
        v_min = pinMinIn  * 33.f;
```

```
    }
}

void set_clock(int input_hour, int input_min, int input_sec) {
    int old_hour = now_hour;
    int old_min = now_min;
    int old_sec = now_sec;
    now_hour = input_hour;
    now_min = input_min;
    now_sec = input_sec;
    int d_hour = now_hour - old_hour;
    int d_input = now_min - old_min;
    int d_sec = now_sec - old_sec;
    if(d_hour<0||d_input<0){
        reset_clock();
        old_hour = 0;
        old_min = 0;
    }

    for(i=old_hour; i<now_hour; i++) {
        if(i >= 12) {
            hour_click();
        }
        hour_click();
        hour_click();
    }
    for(i=old_min; i<now_min; i++) {
        min_click();
    }
}
```

4.3 NodeMCU 核心程式碼

Source Code: https://github.com/wy56/Smart-Clock/blob/master/NodeMCU/ESP8266.lua

```
--Settings
WIFI_STA_SSID = "WiFi_SSID"
WIFI_STA_PW = "WiFI_Password"
WIFI_AP_SSID = "SmartClock-11597"
WIFI_AP_PW = ""
NTP_SERVER_IP = "59.124.29.241"
NTP_SYNC_FEQ = 5
NTP_GMT = 8

--UART Setup
uart.setup(1, 9600, 8, uart.PARITY_NONE, uart.STOPBITS_1, 1) --UART For STM32
--Wi-Fi Setup
wifi_sta_setup(WIFI_STA_SSID, WIFI_STA_PW)
wifi_ap_setup(WIFI_AP_SSID, WIFI_AP_PW)
ip=wifi.ap.getip()
print(ip)
-- Create a server
-- and set 30s time out for a inactive client
srv = net.createServer(net.TCP, 30)
-- Server listen on 80
srv:listen(80, function(conn)
  conn:on("receive", receiver)
end)
--Automatic Sync NTP Time
mytimer2 = tmr.create()
mytimer2:register(30000, tmr.ALARM_SEMI , ntp_sync() ) --For 30 Sec
mytimer2:start()
----------------------------------------------------------------
function wifi_sta_setup(ssid, pwd)
    wifi.setmode(3)
    station_cfg={}
    station_cfg.ssid=ssid
    station_cfg.pwd=pwd
    station_cfg.save=false
    wifi.sta.config(station_cfg)
```

```
end

function wifi_ap_setup(ssid, pwd)
    wifi.setmode(3)
    ap_cfg={}
    ap_cfg.ssid=ssid
    if pwd ~= "" then
        ap_cfg.pwd=pwd
    end
    ap_cfg.save=true
    wifi.ap.config(ap_cfg)
end

function ntp_sync()
    if wifi.sta.getip() == nil then
        return false
    end
    sntp.sync(NTP_SERVER_IP,
        function(sec, usec, server, info)
            rtctime.set(sec+NTP_GMT*3600, usec)
            tm = rtctime.epoch2cal(rtctime.get())
            uart.write(1,string.format("%02d:%02d:%02d\n",tm["hour"],
tm["min"], tm["sec"]))
            return true
        end,
        function()
            --print('failed!')
            return false
        end
    )
end

function receiver(sck, data)
--Parse HTTP Request
  local response = {}
  local _, _, method, path, vars = string.find(data, "([A-Z]+) (.+)?(.+)
HTTP")
  if method == nil then
```

```
    _, _, method, path = string.find(data, "([A-Z]+) (.+) HTTP")
  end
  --Parse GET Query String
  local _GET = {}
  if vars ~= nil then
    for id, value in string.gmatch(vars, "([%w_]+)=([%w._]+)&*") do
        _GET[id] = value
    end
  end

--Route
  if path == "/" then
    response[#response + 1] = "hello"
  elseif path == "/wifi_sta_config" then
    response[#response + 1] = "<h1>Config Succeed!</h1>"
    response[#response + 1] = "<p>wifi_ssid: "..._GET["ssid"].."</P>"
    response[#response + 1] = "<p>wifi_password  :"..._GET["password"].."</P>"
    WIFI_STA_SSID = _GET["ssid"]
    WIFI_STA_PW = _GET["password"]
    wifi_sta_setup(WIFI_STA_SSID, WIFI_STA_PW)
  elseif path == "/wifi_ap_config" then
    response[#response + 1] = "<h1>Config Succeed!</h1>"
    response[#response + 1] = "<p>wifi_ssid: "..._GET["ssid"].."</P>"
    response[#response + 1] = "<p>wifi_password  :"..._GET["password"].."</P>"
    WIFI_AP_SSID = _GET["ssid"]
    WIFI_AP_PW = _GET["password"]
    wifi_ap_setup(WIFI_AP_SSID, WIFI_AP_PW)
  elseif path == "/ntp_config" then
    response[#response + 1] = "<h1>Config Succeed!</h1>"
    response[#response + 1] = "<p>server_ip : "..._GET["server_ip"].."</P>"
    response[#response + 1] = "<p>gmt :"..._GET["gmt"].."</P>"
    response[#response + 1] = "<p>sync_feq :"..._GET["sync_feq"].."</P>"
    NTP_SERVER_IP = _GET["server_ip"]
    NTP_SYNC_FEQ = _GET["update_feq"]
    NTP_GMT = _GET["gmt"]
  elseif path == "/get_ntp_config" then
    response[#response + 1] = "{"
    response[#response + 1] = "server_ip:\'"..NTP_SERVER_IP.."\',"
```

```
    response[#response + 1] = "gmt:\'"..NTP_GMT.."\',"
    response[#response + 1] = "sync_feq:"..NTP_SYNC_FEQ.."}"
  elseif path == "/get_wifi_sta_config" then
    response[#response + 1] = "{"
    response[#response + 1] = "ssid:\'"..WIFI_STA_SSID.."\',"
    response[#response + 1] = "rssi:"..wifi.sta.getrssi()..","
    response[#response + 1] = "ip:\'"..wifi.sta.getip().."\',"
    response[#response + 1] = "mac:\'"..wifi.sta.getmac().."\'}"
  elseif path == "/get_wifi_ap_config" then
    response[#response + 1] = "{"
    response[#response + 1] = "ssid:\'"..WIFI_AP_SSID.."\',"
    response[#response + 1] = "ip:\'"..wifi.ap.getip().."\',"
    response[#response + 1] = "mac:\'"..wifi.ap.getmac().."\'}"
  elseif path == "/ntp_sync" then
    ntp_sync()
    response[#response + 1] = "sync"
  else
    response[#response + 1] = "404 not found"
  end

  -- Sends and removes the first element from the 'response' table
  local function send(localSocket)
    if #response > 0 then
      localSocket:send(table.remove(response, 1))
    else
      localSocket:close()
      response = nil
    end
  end
  -- Triggers the send() function again once the first chunk of data was sent
  sck:on("sent", send)
  send(sck)
end
```

第 5 章　系統整合驗證

DEMO 影片 : https://youtu.be/XV4XlSmcJhs

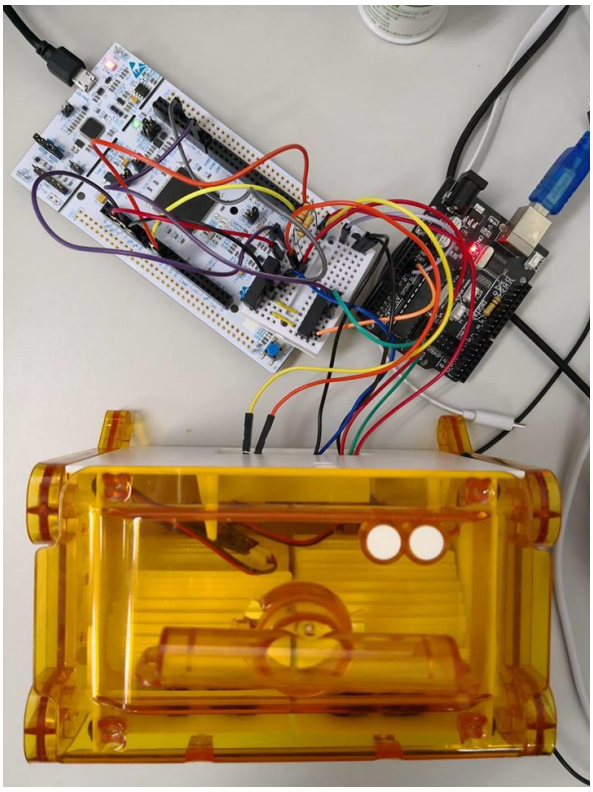

圖 5.1　Smart Clock 接線圖

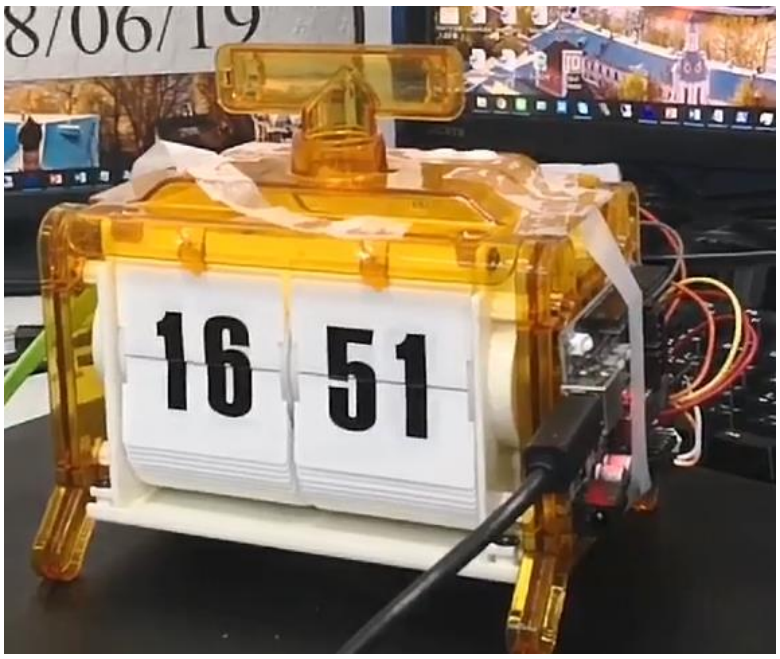

圖 5.2　Smart Clock 產品完成圖

第 6 章　結論與心得

這次嵌入式系統設計期末專題在實作智慧時鐘的時間有點倉促，因此有些在前言所提及的應用還無法達成，但架構上已經完成，在日後可以新增感測器就可以新增更多功能，缺點除了時間上會誤差一至兩秒，因為之間透過 NodeMCU 與 STM32 連線和驅動馬達的所需耗時，所以在本次改進版本中已經把降低誤差值在一秒以內。

這次期末專題結合了我們研究領域所學，並把本堂課所學習的嵌入式系統相關知識整合起來，未來方面想要做具有智慧手錶功能的智慧時鐘，在目前市場上來說 IoT 得應用具有強大的潛力，在日常辦公或在生活上面假設可以透過 IoT 傳達訊息就可以減少對手機的依賴，若日後有機會研究，下一階段預計朝向與手機連線把手機資訊傳回到時鐘上，甚至可以把智慧時鐘透過網路把資訊傳回手機裡，達到萬物聯網的終極目標。

第 7 章　Reference

- https://www.techbang.com/posts/46225-smart-watch-enough-looking-intelligent-glance-clock-clock-helps-you-master-the-big-little-thing
- http://wiki.mbalib.com/zh-tw/%E7%89%A9%E8%81%94%E7%BD%91
- https://zh.wikipedia.org/wiki/%E5%B5%8C%E5%85%A5%E5%BC%8F%E7%B3%BB%E7%BB%9F
- http://www.stdtime.gov.tw/chinese/index_b/b_4.htm
- https://zh.wikipedia.org/wiki/%E7%B6%B2%E8%B7%AF%E6%99%82%E9%96%93%E5%8D%94%E5%AE%9A
- https://www.getit01.com/p20180114124489479/
- http://albert-oma.blogspot.com/2012/03/uart.html
- https://github.com/nodemcu/nodemcu-firmware
- http://nodemcu.com/index_en.html
- http://iot.ttu.edu.tw/wp-content/uploads/2016/06/10_NodeMCU_and_Zumo_IoT.pdf
- https://makerpro.cc/2015/08/esp8266-guideline/
- https://www.edntaiwan.com/news/article/20170802TA01-Circuit-Relay
- https://tw.answers.yahoo.com/question/index?qid=20100727000016KK02009