微處理機期末專題報告

電腦架

電機三甲 10928102 林芊蓉 、 電機三甲 1928131 黃芷宸

摘要

現今長期使用電腦，已經成為人們生活的一部分，人們選擇電腦架主要為三個原因，分別為增加筆記型電腦散熱空間、增加桌面利用空間與減少姿勢不良困擾，先來說第一點，長期使用電腦，會使得電腦溫度升高，如果沒有適當的散熱方式，當 CPU 溫度太高，將會觸發降低效能以避免損壞處理器的機制，使用電腦架能始的電腦更容易散熱。再來，由於使用了電腦架，桌子的使用空間會比把電腦放在桌上使用時的空間大，我們可以更有效的運用桌子去做其他事情。最後，長時間坐在電腦桌前用筆記型電腦工作，肩頸會不自覺的就開始痠痛，其實是因為螢幕高度與正常平視的角度不同，在使用筆記型電腦時，脖子都會有稍微需要低下來的動作，長期下來便導致肩頸方面的困擾。因此有一個適合自己且方便的電腦架是很重要的。

1. 前言

現代大多數人都需要長期使用電腦，長期都用一個不正確的姿勢去使用電腦，可能造成肌肉及筋膜機械張力增加、肌筋膜張力失衡，進而導致肌筋膜疼痛，讓我們的頸椎容易受傷。因此我們藉由這次的期末專題，來製作一個兩段變換高度的電腦支架。再來因為長期使用電腦，會使得電腦溫度升高，如果沒有適當的散熱方式，當 CPU 溫度太高，將會觸發降低效能以避免損壞處理器的機制，電腦的溫度正常範圍為 30 度到 60 度，所以我們在電腦支架上加裝的散熱葉片，當偵測到的溫度超過 60 度時，葉片會自動啟動。

2. 工作原理

我們用群勝 HT32F52352 開發板所用的 Keil uVision5 去編譯程式。我們所用到的程式有以下五個:SPI、GPIO、PWM、EXTI、USART。

2.1 程式介紹

如圖一所示，我們宣告了我們所需用到的程式。

```
#include "ht32.h"
#include "ht32_board.h"
#include "ht32_board_config.h"

#define BufferSize         4
#define BEE_NBR            4                       /*!< Number of bee after reset
#define BEE_FREQ           50                      /*!< Frequency of bee
#define BEE_ACTIVE_TIME    (BEE_FREQ * 5) / 100  /*!< Active 50mS per bee cycle
#define BEE_INACTIVE_TIME  (BEE_FREQ * 5) / 100  /*!< Inactive 50mS per bee cycle

float input;
void SPI_Configuration(void);
void SPI_Loopback(void);
void EXTI_Configuration(void);
void Key_Process(void);
void P(void);
void f(void);

vu32 guKeyState[3];
u8 SPI0_Buffer_Tx[BufferSize] = {0};
u8 SPI1_Buffer_Tx[BufferSize] = {0};
u16 SPI0_Buffer_Rx[1] = {0};
u8 SPI1_Buffer_Rx[BufferSize] = {0};
vu8 SPI0_Tx_Index = 0, SPI1_Tx_Index = 0, SPI0_Rx_Index = 0, SPI1_Rx_Index = 0;

__IO uint32_t wMctmUpdateDownCounter = 0;   /*!< Used to save the numbers of Updat
```

圖一 程式宣告

如圖二所示，是我們的主函式，裡面呼叫副函式使其運作外，在無限迴圈中，我們計算溫度並顯示出當前溫度，當溫度超過一定值時，GPIO 啟動，反之則不啟動。

```
int main(void)
{
  float temp =0;

  RETARGET_Configuration();
  /* Initialize LED1 & LED2 on HT32 board
  HT32F_DVB_LEDInit(HT_LED1);
  HT32F_DVB_LEDInit(HT_LED2);
  SPI_Configuration();


  while (1){
    temp = (SPI0_Buffer_Rx[0] >> 7) + 0.0625*((SPI0_Buffer_Rx[0]>>3)&0x000f);
    printf("Temp = %.5f\r\n", temp);
    if(temp > 25){
    GPIO_WriteOutBits(HT_GPIOC, GPIO_PIN_14, RESET);
    }else{
    GPIO_WriteOutBits(HT_GPIOC, GPIO_PIN_14, SET);
      }
    P();
      }
    }
```

圖二 主函式

如圖三、圖四、圖五所示，其副函式為 SPI 的程式，設定 SPI 初始化設定:分別為 MASTER 的 MISO、MOSI、SCLK、CS 四個角位、系統時鐘的配置。

```
void SPI_Configuration(void)
{
  SPI_InitTypeDef SPI_InitStructure;

  CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
  /* Enable Px, Master, Slave  & AFIO clock
  HTCFG_SPI_MASTER_SEL_GPIO_CLOCK(CKCUClock)         = 1;
  HTCFG_SPI_SLAVE_CLOCK(CKCUClock)                   = 1;
  HTCFG_SPI_MASTER_CLOCK(CKCUClock)                  = 1;
  CKCUClock.Bit.AFIO                                 = 1;
  CKCU_PeripClockConfig(CKCUClock, ENABLE);

  /* MASTER_SEL idle state is HIGH
  GPIO_PullResistorConfig(HTCFG_SPI_MASTER_SEL_GPIO_ID, HTCFG_SPI_MASTER_SEL_AFIO_PIN, GPIO_PR_UP);

  /* Configure related IO to Master mode
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SEL_AFIO_PORT, HTCFG_SPI_MASTER_SEL_AFIO_PIN, AFIO_FUN_SPI); //B2
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SCK_AFIO_PORT, HTCFG_SPI_MASTER_SCK_AFIO_PIN, AFIO_FUN_SPI); //B3
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MOSI_AFIO_PORT, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI); //B4
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_AFIO_PORT, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI); //B5
```

圖三 SPI 程式碼-時間配置/MASTER

```
SPI_InitStructure.SPI_Mode = SPI_MASTER;
SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_16;
SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_ClockPrescaler = 2400;
SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);

/* SPI1 configuration: Slave mode
SPI_InitStructure.SPI_Mode = SPI_SLAVE;
SPI_Init(HTCFG_SPI_SLAVE, &SPI_InitStructure);

/* Set SEL as output mode for slave select
SPI_SELOutputCmd(HTCFG_SPI_MASTER, ENABLE);

/* Enable SPI1 TXBE & RXBNE interrupt
SPI_IntConfig(HTCFG_SPI_SLAVE, SPI_INT_TXBE | SPI_INT_RXBNE, ENABLE);
```

圖四 SPI 程式碼-資料長度設為 16bits / Clock 設 2400

```
SPI_IntConfig(HTCFG_SPI_MASTER, SPI_INT_TXBE | SPI_INT_RXBNE, ENABLE);

/* Configure and enable master interrupt
NVIC_SetPriority(HTCFG_SPI_MASTER_IRQn, 1);
NVIC_EnableIRQ(HTCFG_SPI_MASTER_IRQn);

/* Configure and enable slave interrupt
NVIC_SetPriority(HTCFG_SPI_SLAVE_IRQn, 0);
NVIC_EnableIRQ(HTCFG_SPI_SLAVE_IRQn);

/* Enable slave first
SPI_Cmd(HTCFG_SPI_SLAVE, ENABLE);
/* Enable master later
SPI_Cmd(HTCFG_SPI_MASTER, ENABLE);
}
```

圖五 SPI 程式碼

如圖六所示，為按鈕的設定，會呼叫兩個副函式執行。

```
void P (void)
{
  EXTI_Configuration();
  Key_Process();
}
```

圖六 按鈕設定

如圖七所示，其為 PWM 的角度設定，我們設定了兩個 guKeyState 作為兩個不同的角度。

```
void Key_Process(void)
{
  if (guKeyState[1] == TRUE)
  {
    guKeyState[1] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED2);
    input=0.92;
    f();
  }
  if (guKeyState[2] == TRUE)
  {
    guKeyState[2] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED3);
    input=0.94;
    f();
  }
}
```

圖七 PWM 角度設定

如圖八、圖九所示，此為 EXTI 中斷的設定，設定 EXTI 使得按鈕能在 SPI 運作下使用。

```
void EXTI_Configuration(void)
{
  { /* Enable peripheral clock
    CKCU_PeripClockConfig_TypeDef CKCUClock = {{ 0 }};
    CKCUClock.Bit.AFIO = 1;
    CKCUClock.Bit.EXTI = 1;
    CKCUClock.Bit.PC   = 1;
    CKCU_PeripClockConfig(CKCUClock, ENABLE);
  }

  /* Configure AFIO mode of input pins
  AFIO_GPxConfig(GPIO_PC, AFIO_PIN_10, AFIO_FUN_GPIO);
  AFIO_GPxConfig(GPIO_PC, AFIO_PIN_11, AFIO_FUN_GPIO);

  /* Enable GPIO Input Function
  GPIO_InputConfig(HT_GPIOC, GPIO_PIN_10, ENABLE);
  GPIO_InputConfig(HT_GPIOC, GPIO_PIN_11, ENABLE);

  /* Configure GPIO pull resistor of input pins
  GPIO_PullResistorConfig(HT_GPIOC, GPIO_PIN_10, GPIO_PR_DISABLE);
  GPIO_PullResistorConfig(HT_GPIOC, GPIO_PIN_11, GPIO_PR_DISABLE);
```

圖八 配置系統時鐘/配置 AFIO/EXTI 來源引角

```
AFIO_EXTISourceConfig(AFIO_EXTI_CH_11, AFIO_ESS_PC);
AFIO_EXTISourceConfig(AFIO_EXTI_CH_10, AFIO_ESS_PC);

{ /* Configure EXTI Channel n as rising edge trigger
  EXTI_InitTypeDef EXTI_InitStruct;
  EXTI_InitStruct.EXTI_Channel = EXTI_CHANNEL_11;
  EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
  EXTI_InitStruct.EXTI_DebounceCnt = 0;
  EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
  EXTI_Init(&EXTI_InitStruct);
  EXTI_IntConfig(EXTI_CHANNEL_11, ENABLE);
  NVIC_EnableIRQ(EXTI11_IRQn);

  EXTI_InitStruct.EXTI_Channel = EXTI_CHANNEL_10;
  EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
  EXTI_InitStruct.EXTI_DebounceCnt = 0;
  EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
  EXTI_Init(&EXTI_InitStruct);
  EXTI_IntConfig(EXTI_CHANNEL_10, ENABLE);
  NVIC_EnableIRQ(EXTI10_IRQn);
}
```

圖九 EXTI 暫存器

如圖十、圖十一、圖十二所示，這些為 PWM 的設定。

```
void f(void)
{
  CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
  TM_TimeBaseInitTypeDef TM_TimeBaseInitStructure;
  TM_OutputInitTypeDef TM_OutputInitStructure;
  uint32_t wCRR = 0, wPSCR = 0;

  /* Enable PCLK of BUZZER and AFIO
  BUZZER_TM_CLK(CKCUClock) = 1;
  CKCUClock.Bit.AFIO       = 1;
  CKCU_PeripClockConfig(CKCUClock, ENABLE);

  /* Configure the BUZZER_GPIO_PIN as TM channel output AFIO function
  HT32F_DVB_GPxConfig(GPIO_PA, AFIO_PIN_10, AFIO_FUN_MCTM_GPTM);

  /* Compute CRR and PSCR value
  wCRR = (SystemCoreClock / BEE_FREQ) - 1;
  while ((wCRR / (wPSCR + 1)) > 0xFFFF)
  {
    wPSCR++;
  }
  wCRR = wCRR / (wPSCR + 1);
```

圖十 PWM 程式碼

```
  /* Init BUZZER TM time-base       */
  RETARGET_Configuration();
  // printf("wCRR = %d, wPSCR = %d \r\n", wCRR, wPSCR);
  TM_TimeBaseInitStructure.CounterReload = wCRR;
  TM_TimeBaseInitStructure.Prescaler = wPSCR;
  TM_TimeBaseInitStructure.RepetitionCounter = 0;
  TM_TimeBaseInitStructure.CounterMode = TM_CNT_MODE_UP;
  TM_TimeBaseInitStructure.PSCReloadTime = TM_PSC_RLD_IMMEDIATE;
  TM_TimeBaseInit(BUZZER_TM, &TM_TimeBaseInitStructure);

  /* Clear Update Event Interrupt flag
  TM_ClearFlag(BUZZER_TM, TM_FLAG_UEV);

  /* Enable interrupt of BUZZER TM update event
  NVIC_EnableIRQ(BUZZER_IRQn);
  TM_IntConfig(BUZZER_TM, TM_INT_UEV, ENABLE);
#if defined(USE_HT32F52352_SK)
  /* BUZZER TM Channel Main Output enable
  MCTM_CHMOECmd(BUZZER_TM, ENABLE);
#endif
  /* BUZZER TM counter enable
  TM_Cmd(BUZZER_TM, ENABLE);
```

圖十一 PWM 程式碼

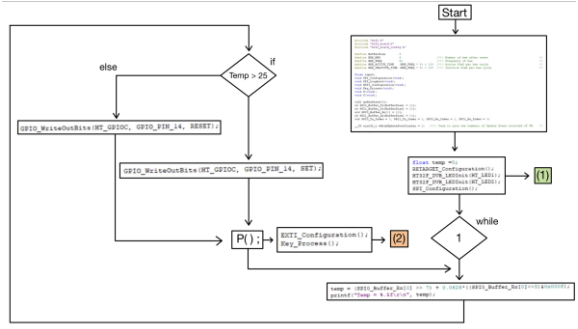
```
{
  TM_OutputInitStructure.Channel = BUZZER_TM_CHANNEL;
  TM_OutputInitStructure.OutputMode = TM_OM_PWM2;
  TM_OutputInitStructure.Control = TM_CHCTL_ENABLE;
  TM_OutputInitStructure.ControlN = TM_CHCTL_DISABLE;//TM_CHCTL_ENABLE;
  TM_OutputInitStructure.Polarity = TM_CHP_NONINVERTED;
  TM_OutputInitStructure.PolarityN = TM_CHP_NONINVERTED;
  TM_OutputInitStructure.IdleState = MCTM_OIS_LOW;
  TM_OutputInitStructure.IdleStateN = MCTM_OIS_HIGH;
  TM_OutputInitStructure.Compare = (wCRR + 1) * input;
  TM_OutputInit(BUZZER_TM, &TM_OutputInitStructure);
 };
}

#if (HT32_LIB_DEBUG == 1)
#endif
```

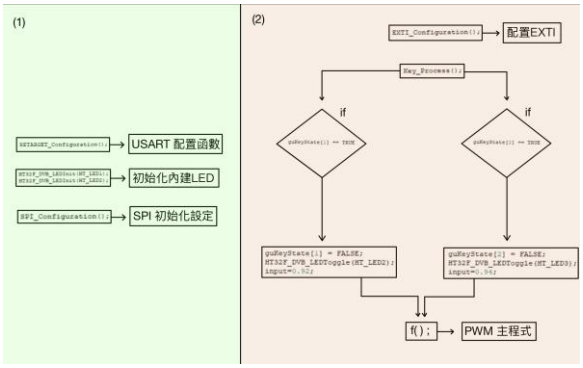
圖十二 PWM 程式碼

2.2 程式流程圖

以下是我們用流程圖，介紹我們這次專題所有的程式。



圖十三 流程圖一



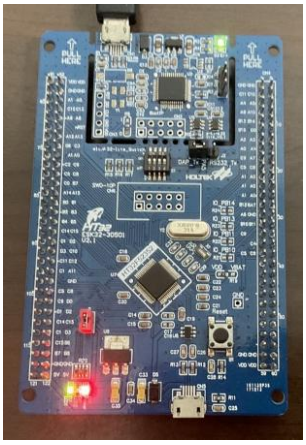
圖十四 流程圖二

3. 作品結構-硬體架構介紹

我們支架結合了一個勝群的 HT32 開發版、一個伺服馬達、一個溫度感測、一個直流馬達、兩個按鈕控制兩個不同的角度，利用兩個按鈕控制伺服馬達的角度，並用溫度感測去偵測溫度，當溫度高於一定的值時，啟動馬達散熱，當溫度降回正常值時，馬達自動停止。以下將分別介紹各裝置及其功能用途:

3.1 盛群 HT32F52352 開發版

HOLTEK HT32F52342/52352 器件是基於 Arm Cortex-M0+ 處理器內核的高性能、低功耗 32 位微控制器。



圖十五 盛群 HT32F52352 開發版

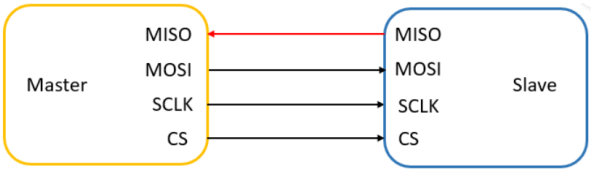
3.2 伺服馬達(PWM)

伺服馬達就是依照指示命令動作所構成的控制裝置，應用於電機的伺服控制，將感測器裝在電機與控制對象機器上，偵測結果會返回伺服放大器與指令值做比較。伺服馬達的接線分別為:棕色為接地 、紅色為 5V 、橘色為訊號線接 PWM。

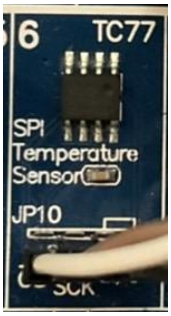
圖十六 伺服馬達

3.3 溫度感測(SPI)

主要應用在 EEPROM、快閃記憶體、AD 轉換、部分 LCD 上都會使用，SPI 屬於高速、全雙工且具同步的通訊網路，連結外部設備時只需要四條線就可以完成通訊，同時節省了空間。其優點為僅需 4 條線有效減少空間、全雙工通訊、資料傳送快。



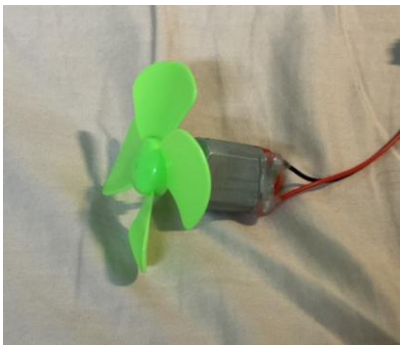
圖十七 SPI 一對一接法



圖十八　SPI Temperature Sensor

3.4 直流馬達(GPIO)

最早發明能將電力轉換為機械功率的電動機，雖然交流伺服馬達應用於精密的定位控制已是未來的發展趨勢，但直流馬達有良好的線性特性，具有簡單易於控制的優點，仍是目前最常應用於變速控制的馬達，同時瞭解直流馬達的特性與控制也是進入交流伺服控制的必要途徑。



圖十九　直流馬達與扇葉

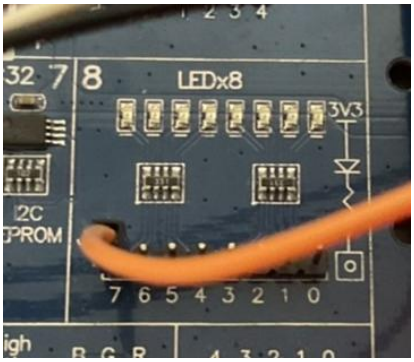
3.5 按鈕

是一種電閘（switch，或稱開關），用來控制機械或程式的某些功能。

圖二十　按鈕

3.6 LED

是一種半導體光源，當電流通過它時會發光；即一種電致發光的半導體電子元件，其內電子與電子空穴複合，以光子的形式釋放能量。



圖二十一　按鈕

3.7 外接電源

我們用板子所提供的電源，可能因為皆了很多東西，導致電壓不足，伺服馬達不穩定，因此我們外接了一個電源。

圖二十二　外接電源

3.8 架子

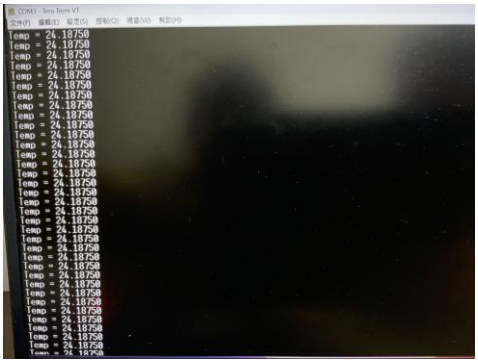
我們製作了一個架子用來演示我們所做出來的專題。

圖二十三　架子

4. 測試方法

4.1 溫度以及 Tera Term 的數據顯示

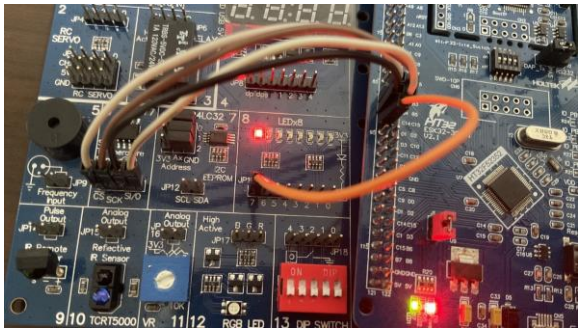
我們用 Tera Term 顯示溫度，我們在影片中所演示的溫度為 25 度，若是以我們所想，電腦過熱溫度為 60 度，但溫度過高，我們不好演示，故用 25 度設為過高溫度。



圖二十四　Tera Term 顯示

4.2 溫度大於 25 度時 LED 是否會亮

原本我們是要使用直流馬達並安裝上扇葉，但因為直流馬達會把所有的電都搶走，因此我們用 LED 燈代替運作。



圖二十五　LED 運作

4.3 伺服馬達與按鈕

我們用伺服馬達設定了兩個角度，利用按鈕控制，分別為圖二十四和圖二十五。

圖二十六　角架角度一

圖二十七　角架角度二

5. 結論

本作品使用盛群 HT32F52352 開發板做為電腦架的核心，接收溫度感測，並在達到一定溫度時，開啟馬達，再來運用按鈕控制伺服馬達，使其有不同的角度，讓使用者調整至方便使用的角度。

使用電腦架避免了，長期使用電腦，會使得電腦 CPU 溫度太高，會觸發降低效能以避免損壞處理器的機制。同時也解省了桌面空間，讓我們能有更多桌面空間做其他事情。最後，因為使用電腦架，使得螢幕高度與正常平視的角度一致，解決了長時間坐在電腦桌前用筆記型電腦工作，肩頸會不自覺痠痛的問題。

這次我們使用了五種不同的函式，原本沒有要加上 EXTI 的中斷函式，但是因為覺得利用在電腦輸入想要的角度有點太奇怪了，再加上 Tera Term 同時還要顯示目前偵測到的溫度值，會讓兩者之間輸出會衝突，所以後來才加上中斷函式，利用按鈕直接控制角度。當然，這樣改還是有一定的缺點，就是不能微調整想要的角度，只能夠調整已經固定的角度而已，如果想要利用按鈕調整多個角度的話，就必須要加入多個中斷的腳位。

此次專題最難的部分是在合併 SPI 函式以及 PWM 函式，如果單純地把兩個函式合併在一起時會有衝突，沒辦法同時進行，所以我們最終會選擇多加入中斷也有一部分的原因是為了讓這兩者不會產生衝突，此專題是將 PWM 函式加入到 EXTI 函式裡，才成功地讓程式運行，但如果在 SPI 函式裡加入中斷函式其實也能成功地讓這兩個函式同時運作。

6. 實驗影片

https://youtu.be/QdxCRR4R02M

7. 參考資料

https://zh.wikipedia.org/zh-tw/%E4%BC%BA%E6%9C%8D%E9%A6%AC%E9%81%94

STM32-10 SPI 介紹 - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)

產品細節 - Holtek

直流馬達 - 超專業 der 馬達介紹 (google.com)

【2021 筆電架推薦】精選 10 個輕巧的筆電增高架 | 質感極簡款 | AY's Simple Life (aaay.website)

https://ithelp.ithome.com.tw/articles/10284265?sc=rss.qu