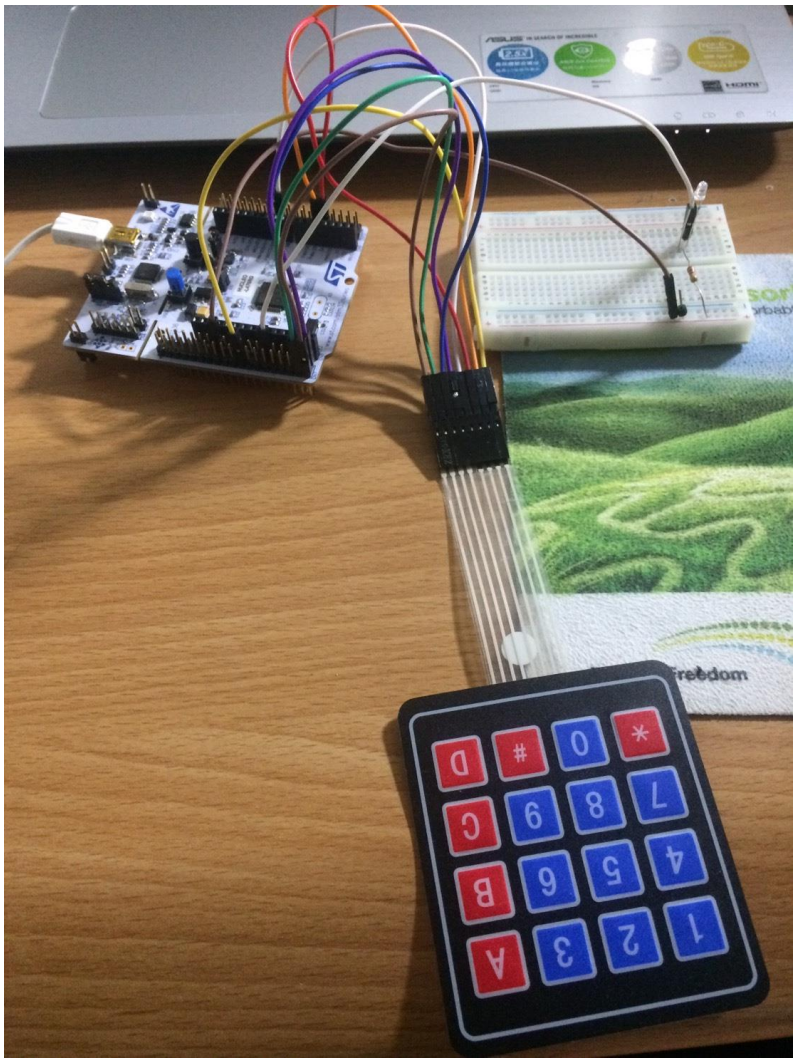
NCTU 2019 微處理機 Lab7

學號 :0616018
姓名 : 林哲宇

Lab7.1: SysTick timer interrupt setting

這題要利用 Systick timer 來製造 interrupt，所以跟上週的 timer 不一樣。

首先，第一題只需要用到一個燈泡，所以我將這個燈泡接在 GPIOA，而 keypad 部分是第二題的，線路如下：



再來是程式碼的部分，main 就只是將 GPIO 初始化，接著進入這次作業的重點 – SystemClock_Config，最後就是一個無窮迴圈。

```
32 int main(){
33     GPIO_init();
34     SystemClock_Config();
35
36     while (1);
37
38 }
39
```

在 SystemClock_Config 中，由於有要求使用 HSI16，所以要將 RCC->CR 和 RCC->CFGR 設置成 0x100 和 1。接著將 prescaler 設成 2，可以讓每 1000000 cycles 為一秒。下面設置 SysTick->CTRL |= 2 是讓秒數數到零時觸發 interrupt，讓程式可以跳到 SysTick_Handler。SysTick->LOAD = 3*1000000 是三秒。Systick->VAL 是初始值，數到零時就會觸發 interrupt，而每次中斷完畢，又會 reload 成 SysTick->LOAD 重新倒數。最後 SysTick->CTRL 是 enable，讓 Systick 開始倒數。當中斷發生時，就會進入 SysTick_Handler()變動 GPIOA->ODR，讓燈泡亮或暗。

```
13 void SystemClock_Config() {
14     //turn on HSI16
15     RCC->CR |= 0x100;
16     RCC->CFGR |= 1;
17     //for prescaler 2
18     RCC->CFGR |= 0x80;
19
20     //external clock source
21     // tickint
22     SysTick->CTRL |= 2;
23     // 3 second
24     SysTick->LOAD = 3 * 1000000;
25     SysTick->VAL = 0;
26     //enable
27     SysTick->CTRL |= 1;
28 }
29 void SysTick_Handler(void) {
30     GPIOA->ODR ^= 1;
31 }
```

Lad7.2: Multiple External Interrupt setting

第二題是要將 keypad 變為中斷源，所以這次要使用 EXTI->IMR，判斷是否需要 interrupt。線路和第一題一模一樣，keypad 的 output 是接在 GPIOC0,1,2,3，input 是 GPIOB4,5,6,7。

在 main 中，SysTick 改成每 0.1 秒就 interrupt 一次，進入 SysTick_Handler()。後面 gpio_init, led_init, keypad_init 都和之前的 Lab 一樣。exti_init()是設置 syscfg, exit 的函式。

```c
int main()
{
    SysTick_UserConfig();
    gpio_init();
    led_init();
    keypad_init();
    exti_init();
    while (1)
    {
        prev = YPORT->IDR;
    }
}

void SysTick_UserConfig()
{
    // 第一個 enable, 第二個 是否數到零發生意外 Tickint, 第三個選擇時鐘 clksource
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = 400000; // 0.1 second
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}
```

每 0.1 秒就會進入這個 SysTick_Handler()，利用 keypad 的 output 一個一個掃，觸發中斷。

```c
void SysTick_Handler()
{
    EXTI->IMR1 = 0;
    scan_state = (scan_state + 1) % 4;
    int i;
    for(i=0; i<4; i++){
        XPORT->BRR = x_pin[i];
        if(i == scan_state) XPORT->BSRR = x_pin[i];
    }
    // IMR 啟動中斷
    EXTI->IMR1 |= EXTI_IMR1_IM4 | EXTI_IMR1_IM5 | EXTI_IMR1_IM6 | EXTI_IMR1_IM7;
}
```

EXTI_IMR1_IM4 會觸發 EXTI4_IRQHandler()，EXTI_IMR1_IM5, EXTI_IMR1_IM6, EXTI_IMR1_IM7 則會觸發 EXTI9_5_IRQHandler()。

在這兩個 interrupt 函式中都會呼叫 scan()，並且最後設置 EXTI->PR1 清除中斷。

```
58 void EXTI4_IRQHandler()
59 {
60     uint32_t *ptr;
61     ptr = (uint32_t *) NVIC_ICPR;
62     ptr[0] = 0x00000400;
63     scan();
64     EXTI->PR1 |= EXTI_PR1_PIF4;
65 }
66
67 void EXTI9_5_IRQHandler()
68 {
69     uint32_t *ptr;
70     ptr = (uint32_t *) NVIC_ICPR;
71     ptr[0] = 0x00800000;
72     scan();
73     // PR 是做清除動作
74     EXTI->PR1 |= EXTI_PR1_PIF5 | EXTI_PR1_PIF6 | EXTI_PR1_PIF7;
75 }
76
```

最後是 scan()，就是掃 keypad 的 input，看是否有被按下，接著讓燈泡閃爍相對應次數，結束後再亮起來。

```
77 void scan(){
78     int now = YPORT->IDR, i, j;
79     for(i=0; i<4; i++)
80         for(j=0; j<4; j++)
81             if((prev & y_pin[j]) && !(now & y_pin[j]) && scan_state == i) key_value = table[i][j];
82
83     for(int i=0;i<2*key_value;i++){
84         if((GPIOA->ODR&1)==1)
85             GPIOA->ODR&=0xFFFFFFFE;
86         else
87             GPIOA->ODR|=0b1;
88         delay_1s();
89     }
90     GPIOA->ODR&=0xFFFFFFFE;
91 }
```

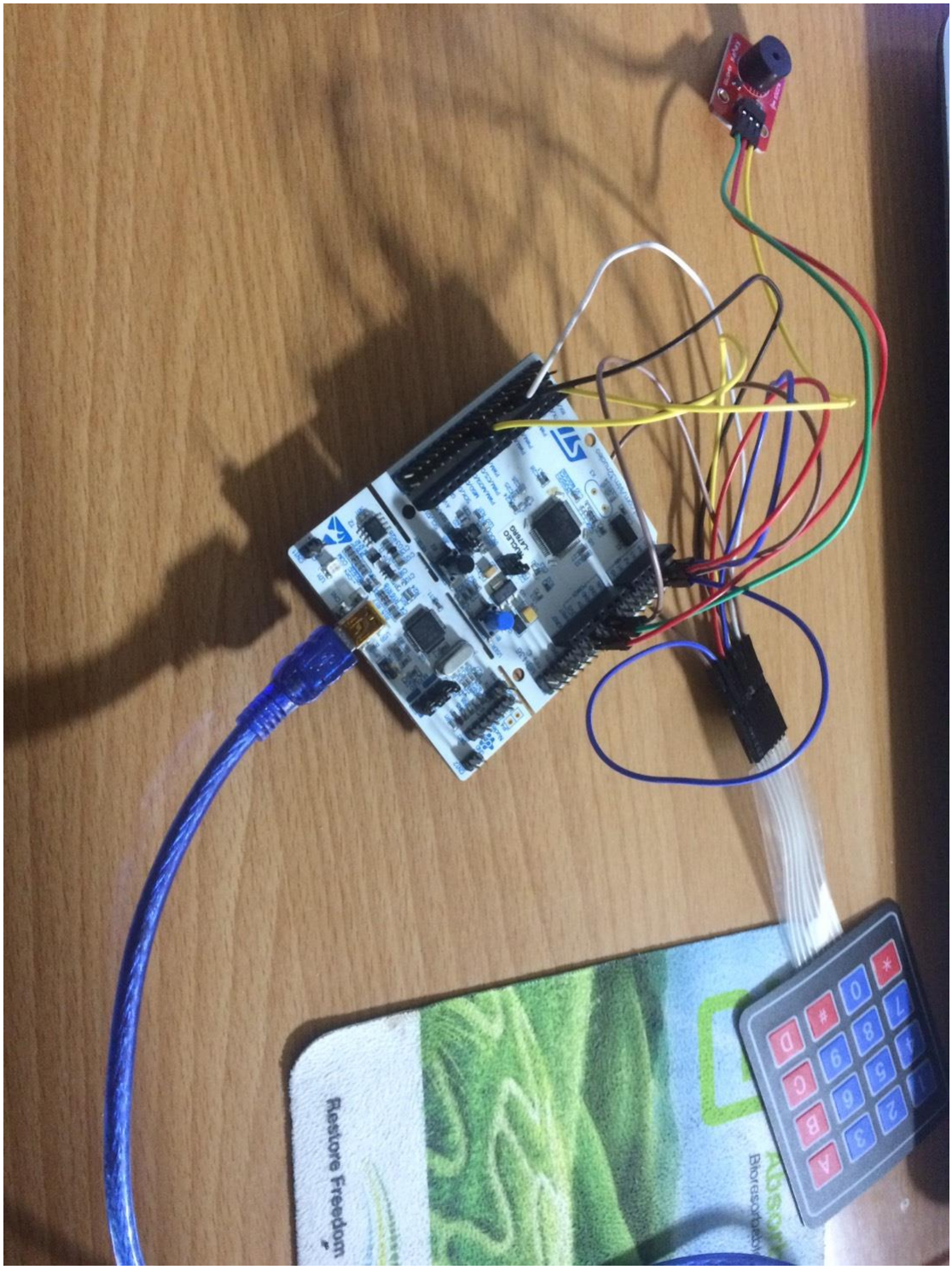
不過這題做完有一點怪怪的，有時候 keypad 按下去沒反應，有試著調整過仍然無法。

Lab7.3: 簡單鬧鐘

這題需要使用到 SysTick timer、User button、keypad 和蜂鳴器，也就是上個 Lab 和前兩題的綜合體。

首先是線路，keypad 的 output 是接在 GPIOC0,1,2,3，input 是

GPIOB4,5,6,7，蜂鳴器分別接電源、接地、GPIOB3。圖片如下：



在 main 中，跟之前一樣去初始化各項零件，接著有一個 while 迴圈，裡面去掃又有個 while 迴圈在接收 keypad 的輸入。如果沒有

輸入，就會一直在 while 迴圈中，注意 keypad_disable，這是為了保證在倒數時，不會又被使用者按 keypad 而導致錯誤而設置的保護。有輸入時，就會離開迴圈設定 keypad_disable 和 SysTick，設置的秒數(SysTick->LOAD)就是 keypad 輸入的值的秒數，接著 enable SysTick。

```
int main()
{
    fpu_enable();
    gpio_init();
    keypad_init();
    SystemClock_Config();

    while(1){
        while(1){
            curr = keypad_scan();
            if(curr == -1 || keypad_disable)continue;
            else break;
        }
        keypad_disable = 1;
        // 3 second
        if(curr)SysTick->LOAD = curr * 1000000;
        else SysTick->LOAD = 10;
        SysTick->VAL = 0;
        //enable
        SysTick->CTRL |= 1;
    }
}

```

當倒數完畢時，進入 SysTick_Handler，執行讓蜂鳴器發出聲音的函式，直到使用者按下 user button 為止，也就是檢查 GPIOC 的第十三個 pin 是否回零，如果是的話，就會將變數設回初始值，並且離開迴圈。

```
78 void SysTick_Handler() {
79     SysTick->CTRL &= 0xfffffffe;
80     while (1)
81     {
82         if(curr != -10){
83             freq = DO;
84             TIM2->PSC = (uint32_t) (4000000 / freq / 100);
85             // prescaler value
86             TIM2->CCR2 = duty_cycle;
87             // compare 2 preload value
88             TIM2->CR1 |= TIM_CR1_CEN;
89         }
90         if(!(GPIOC->IDR & 0b10000000000000)){
91             TIM2->CR1 &= ~TIM_CR1_CEN;
92             freq = -1;
93             curr = -1;
94             keypad_disable = 0;
95             break;
96         }
97     }
98 }
```

心得：

這次 Lab 學到了如何製造 interrupt，也大概能理解概念，學著如何控制程式流來達到要做的中段效果。個人認為這次 Lab 比較困難的點就是在於初始化那些值，如果有足夠的範例程式應該可以更輕鬆寫完，不過不管是網路上資料，甚至是講義，都有些艱澀難懂。