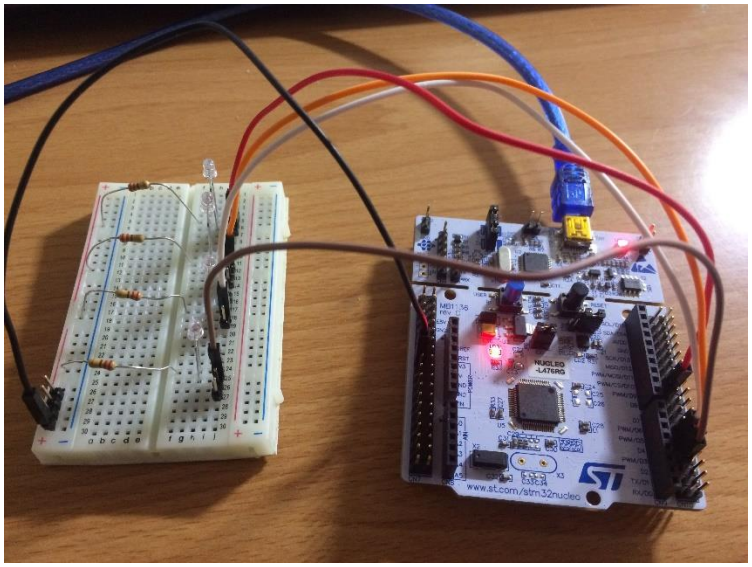
Lab3

0616018 林哲宇

一、Lab3.1 : LED pattern displayer

這一題的接法如下圖。黑色那一條用來接向正極，透過電阻接到燈泡同一直排中，燈泡另一個腳則接到 spec 建議的 PB3, PB4, PB5, PB6，分別對應咖啡色、橘色、白色、紅色，對應的方式是看講義第五頁。



在下圖程式碼中，先設定 RCC_AHB2ENR 之後可以設定啟動 clock。還有 GPIOB 的各項資訊，需要設定的只有 MODER, OSPEEDR, ODR，另外兩個則使用預設值。Time 則是等等做 delay 要用到的常數，1 sec = 10^6 us ; 1 cycle = 0.25 us，所以換算過來，一秒總共有 4000000 個 cycles。

```
.text
    .global main
    .equ RCC_AHB2ENR, 0x4002104C
    .equ GPIOB_MODER, 0x48000400
    .equ GPIOB_OTYPER, 0x48000404
    .equ GPIOB_OSPEEDR, 0x48000408
    .equ GPIOB_PUPDR, 0x4800040C
    .equ GPIOB_ODR, 0x48000414
    .equ time, 4000000
```

在下圖程式碼中，我初始化 GPIOB 的各項數值，其中因為只需要使用到 B PORT，所以 RCC_AHB2ENR 要設為 2。PB3～PB6 的 MODER 要設為 output mode，也就是 1，因為是 3～6，所以右邊要空 2*3 bits。再來設定 OSPEEDR，我不太懂設定 speed 的用意，但是看講義這樣做，我就這樣做了，看名字應該是設定反應速度吧。最後是 ODR，一開始只有最右邊的燈泡是亮的，所以就讓第四個 bits 是 0。

```
26 GPIO_init:
27     //TODO: Initial LED GPIO pins as output
28     // Enable AHB2 clock to control GPIOB(2)
29     movs r0, 2
30     ldr r1, =RCC_AHB2ENR
31     str r0, [r1]
32     // set PB3, PB4, PB5, PB6 to output mode(1)
33     movs r0, 0b01010101000000
34     ldr r1, =GPIOB_MODER
35     str r0, [r1]
36     // set PB3, PB4, PB5, PB6 to high speed mode
37     movs r0, 0b10101010000000
38     ldr r1, =GPIOB_OSPEEDR
39     str r0, [r1]
40     // set the initial light right bulb
41     ldr r0, =GPIOB_ODR
42     movs r1, 0b11110111
43     str r1, [r0]
44     bx lr
```

main 中我有設定一些變數，一個是 spec 給的 leds，還有用來代表方向的 r8。leds 表示哪幾個燈泡是亮的，亮的則為 1，因為等等 shift 之後補位會變成 0。之所以要讓第三個 bit 也設為 1 是因為等等向左移動或向右移動時，會有兩個燈泡一起亮著，所以先把那個 bit 設好比較方便。接著是 r8，r8 用來當作方向，向左的話是 1，向右則為 0，之後程式碼會用 r8 來做一些判斷。

```
15 main:
16     bl GPIO_init
17     bl Delay
18     // set the current leds condition
19     movs r1, 0b00001100
20     ldr r0, =leds
21     str r1, [r0]
22     // r8 for direction, to left is 1, to right is 0
23     movs r8, 1
24
25     B Loop
```

程式碼流程是這樣：先顯示 LED 燈的變化，再來要停 1 秒，之後反覆執行。

```
45 Loop:
46     //TODO: Write the display pattern into leds variable
47     bl DisplayLED
48     bl Delay
49     b Loop
```

顯示 LED 燈的變化要考慮一些事情，首先要先判斷現在是要向左還是向右，這點可以根據之前設定好的 r8 判斷。然後將存有目前狀態的 leds 向左或向右移動一個 bit，並且檢查是否碰到底要轉方向了。若要轉方向則會跳到 Switch 中把 r8 和 1 做 xor，也就是把 r8 反轉。

```
50 DisplayLED:
51     //TODO: Display LED by leds
52     // use direc to determine left or right
53     ldr r0, =leds
54     ldr r2, [r0]
55     ldr r1, =GPIOB_ODR
56     cmp r8, 0
57     beq Right
58     Left:
59         lsl r2, 1
60         str r2, [r0]
61         eor r4, r2, 0b11111111
62         str r4, [r1]
63         // see wether to change direction by looking if the 8th bit is 1
64         movs r3, 0b10000000
65         and r3, r2, r3
66         cmp r3, 0b10000000
67         beq Switch
68         bx lr
69     Right:
70         // shift right and
71         lsr r2, 1
72         str r2, [r0]
73         eor r4, r2, 0b11111111
74         str r4, [r1]
75         // see whether to change direction by looking if the 2nd bit is 1
76         movs r3, 0b100
77         and r3, r2, r3
78         cmp r3, 0b100
79         beq Switch
80         bx lr
81     Switch:
82         eor r8, r8, 1
83         bx lr
```

最後是 Delay 的部分，總之就是跑一個垃圾迴圈，什麼也不做，浪費掉一秒。先把在 main 中設好的 time 拿到暫存器 r0，所以現在 r0=4000000，這代表 4000000 個 cycles。在 Delay_loop 要去判斷是否已經一秒，是的話就要離開迴圈。所以每次要減掉一定的數值，直到 r0 比 0 小就離開迴圈。那每次要減少的 cycle 數就是 subs + cmp + bge(taken) = 1 + 1 + 3 = 5。

```
84 Delay:
85     //TODO: Write a delay 1 sec function
86     // set the initial value of the counter of the loop
87     ldr r0, =#time
88     Delay_loop:
89         // 1 + 1 + 3 = 5
90         subs r0, r0, 5
91         cmp r0, 0
92         bge Delay_loop
93     bx lr
```

心得感想：第一題花爆多時間的，主要是因為我對硬體一點概念都沒有，只好穩穩地從頭開始看講義，因此花了非常多的時間。一開始連線怎麼街都不知道，更不用說哪個 port 有什麼功效，直到來回看了五次以上的講義才大概明白。

二、Lab3.2：Push button

這題線路圖和第一題一樣，因為這題只需要多一個功能：利用 User button 控制第一題的燈停下或繼續。
和上一題不同的地方就是多了 GPIOC_MODER, GPIOC_IDR。因為 User button 是放在 PC13，所以需要多設定一個 PORT。

```
.text
    .global main
    .equ RCC_AHB2ENR, 0x4002104C
    .equ GPIOB_MODER, 0x48000400
    .equ GPIOB_OTYPER, 0x48000404
    .equ GPIOB_OSPEEDR, 0x48000408
    .equ GPIOB_PUPDR, 0x4800040C
    .equ GPIOB_ODR, 0x48000414
    .equ GPIOC_MODER, 0x48000800
    .equ GPIOC_IDR, 0x48000810
    .equ time, 4000000
```

這邊跟上一題不一樣的地方就是 RCC_AHB2ENR 要設為 0b110，因為需要用到 B, C PORT。要把 PC13 的 MODE 設為 input mode，即是 0。然後 PC13 的 IDR 一開始要設為 1，按下去則變成 0。

```
GPIO_init:
    //TODO: Initial LED GPIO pins as output
    // Enable AHB2 clock to control GPIOB(2)
    movs r0, 0b110
    ldr r1, =RCC_AHB2ENR
    str r0, [r1]
    // set PB3, PB4, PB5, PB6 to output mode(1)
    movs r0, 0b01010101000000
    ldr r1, =GPIOB_MODER
    str r0, [r1]
    // set PB3, PB4, PB5, PB6 to high speed mode
    movs r0, 0b10101010000000
    ldr r1, =GPIOB_OSPEEDR
    str r0, [r1]
    // set the initial light right bulb
    ldr r0, =GPIOB_ODR
    movs r1, 0b11110111
    str r1, [r0]

    // set PC13 to input mode(0)
    ldr r0, =#0xF3FFFFFF
    ldr r1, =GPIOC_MODER
    str r0, [r1]
    // set PC13 1
    movs r0, 0b10000000000000
    ldr r1, =GPIOC_IDR
    str r0, [r1]
    bx lr
```

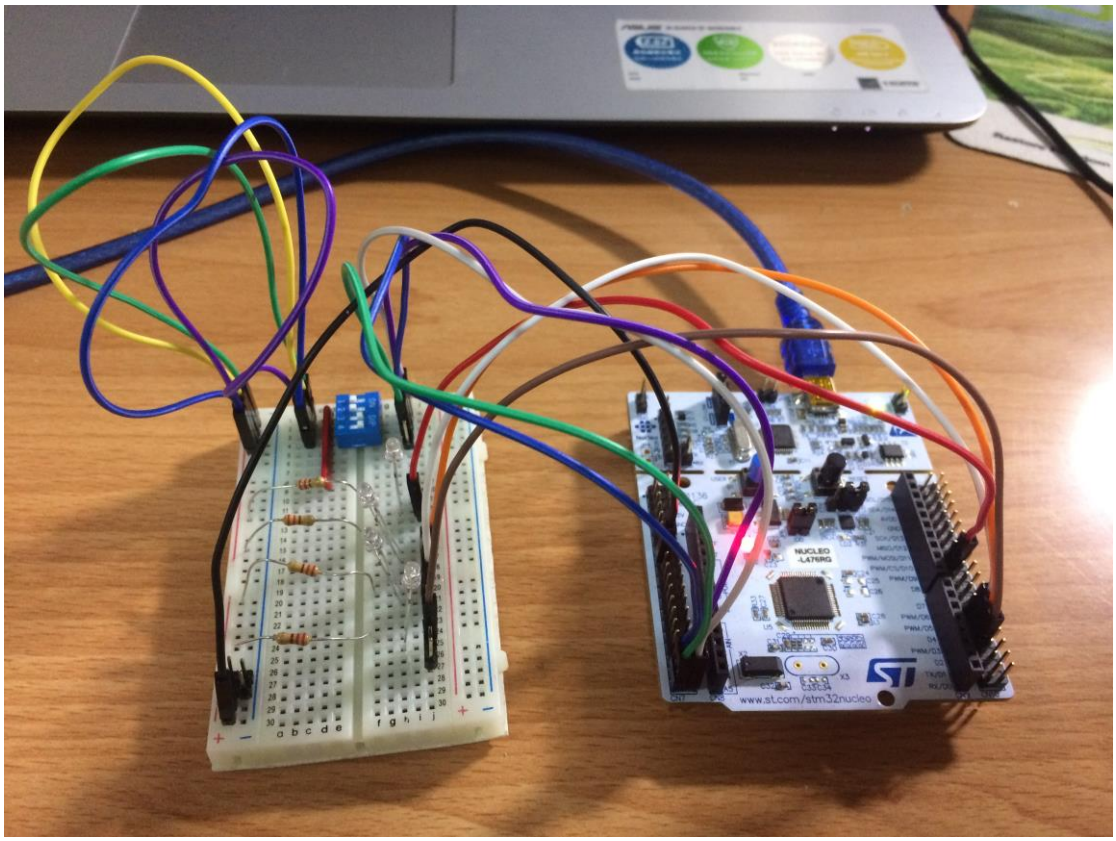
DisplayLED 跟上一題一樣，而最不同的是下圖 Delay。因為會有 Switch bounce，因此我的解決方法是，所果有超過連續一千個 PC13 為 0 的情形發生，我就判斷 User button 有被壓下去了。如果 PC13 是 0，則會 branch 到 Pressed，在這裡會先將 r1 加一，再判斷 r1 有沒有大於 1000。如果大於 1000，就會將 r7 反轉，並將 r1 設回為 0，之後就跑到 check，去判斷 r7 是 1 或 0，1 就 block，0 就繼續。其中，我設定了 r3，因為我擔心如果按鈕壓太久，會出現第二個 1000，這樣又會將 r7 反轉，這不是我們預期的結果。最後是整個 Delay_loop 的 cycle 數，因為預設是沒有按按鈕的，所以會假設 bne Pressed 的 branch 不被 taken；相反的，b check 會假設 taken。
因此最後的算法是：
ldr + ldr + cmp + bne Pressed(not taken) + movs + movs + b Check(taken) + cmp + beq Delay_loop(not taken) + subs + cmp + bge = 2 + 2 + 1 + 1 + 1 + 2 + 2 + 1 + 1 + 1 + 1 + 3 = 18

```
96 Delay:
97     //TODO: Write a delay 1 sec function
98     // set the initial value of the counter of the loop
99     ldr r0, =#time
100    // r2 for counting how many pressing button signal
101    movs r1, 0
102    // r3 for whether pressing continuely
103    movs r3, 0
104    Delay_loop:
105        // check if press button
106        // 2 + 2 + 1 + 1 + 1 + 2 + 2 + 1 + 1 = 13  default : branch not taken
107        ldr r2, =GPIOC_IDR
108        ldr r2, [r2]
109        cmp r2, 0b10000000000000
110        bne Pressed
111        movs r1, 0
112        movs r3, 0
113        b Check
114        Pressed:
115            adds r1, r1, 1
116            ldr r4, =#1000
117            // prevent switch bounce. only if 1000 PC13 = 1 in raw will be seen as dectecting the pressed
118            cmp r1, r4
119            ble Check
120            // had been changed and over 1000 times
121            cmp r3, 1
122            beq Check
123            eor r7, 1
124            movs r1, 0
125            movs r3, 1
126        // see whether block, if r7 = 1, then block
127        Check:
128            cmp r7, 1
129            beq Delay_loop
130        // 13 + 5 = 18
131        // 1 + 1 + 3 = 5
132        subs r0, r0, 18
133        cmp r0, 0
134        bge Delay_loop
```

心得感想：
第二題花的時間大概是第一題的四分之一吧。有了第一題的基礎，第二題好做許多。基本上沒有遇到什麼瓶頸，唯一困惑的部分是：按鈕的部分不知道怎麼 Debug，因為和燈泡不一樣，按下去之後繼續跑，memory 似乎也沒有改變。所以這題是先寫完直接肉眼 Debug，還好一次就對了。

三、Lab3.3： 密碼鎖

這題也是延續第一題和第二題的，就是多了密碼鎖還有燈泡亮的方式要改變，所以我多設定 GPIOC 的部分，將 PC0～PC3 分別接給密碼鎖的 0123，另外因為電阻不夠用，所以用線接了四條出來用，並放上一排電阻接通。其餘都和第一題一樣。



這題我繼續用 GPIOC，多使用了 PUPDR，因為需要先設定成 Pull-up(01)。

```
.text
    .global main
    .equ RCC_AHB2ENR   , 0x4002104C
    .equ GPIOB_MODER   , 0x48000400
    .equ GPIOB_OTYPER  , 0x48000404
    .equ GPIOB_OSPEEDR, 0x48000408
    .equ GPIOB_PUPDR   , 0x4800040C
    .equ GPIOB_ODR     , 0x48000414
    .equ GPIOC_MODER   , 0x48000800
    .equ GPIOC_PUPDR   , 0x4800080C
    .equ GPIOC_IDR     , 0x48000810
```

GPIO_init 比之前多的部分就只是多設定 PC0～PC3 的 input mode 和 Pull-up 而已。

```
62    // set PC1~PC4 to input mode(00)
63    ldr r0, =#0xF3FFFFF00
64    ldr r1, =GPIOC_MODER
65    str r0, [r1]
66    // set PC1~PC4 to Pull-up(01)
67    ldr r0, =#0xFFFFFF55
68    ldr r1, =GPIOC_PUPDR
69    str r0, [r1]
```

其中一個重點就是要偵測是否有按下按鈕，如果有的話則準備要判斷 password 是否正確。這會放在下面截圖。

```
86 press_button:
87    sub  r1, r8, r9
88    cmp  r1, 1
89    mov  r9, r8
90    beq  read_switch
91    subs r0, 8
92    b    poll_button
93
```

驗證密碼，如果正確則會呼叫 ledss，否則呼叫 leds。在 ledss 就是閃三下，leds 則是閃兩下，delay 的程式碼和前兩題差不多。

```
 99 password_check:
100    ldrh r1, [r12]
101    and  r1, 15
102    eor  r1, 15
103
104    ldr  r0, =password
105    ldrb r0, [r0]
106
107    cmp  r0, r1
108    mov  r1, 255
109    mov  r2, 0b0
110    beq  ledss
111    b    leds
```

如果判斷完，不管是否正確都會需要 blink。

```
145 blink:
146    beq  again
147    subs r0, 4
148    b    blink
149
150 again:
151    b    leds_1x
```

心得感想：

第三題一開始測試都會怪怪的，本來使用 PA0～PA3，結果出來的值都不對，直到試過 PC0～PC3 才正確，不太確定原因，可能是接錯了？接完之後寫程式的部分基本上沒什麼問題，果然只要第一題會了，後面也水到渠成。這三份作業

比想像中有趣，雖然一開始接近崩潰，後來挖糞塗牆才好不容易掌握到了要領，可喜可賀。