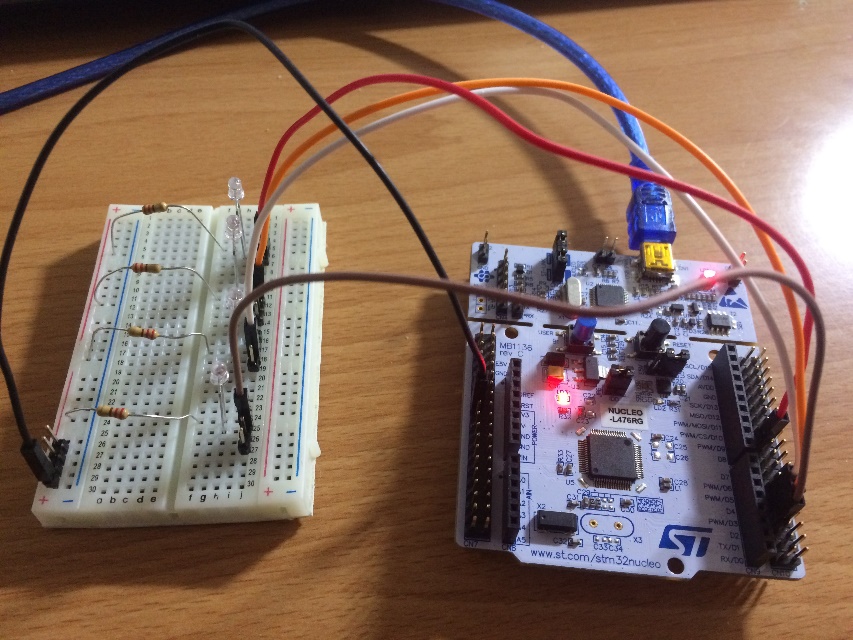
Lab3

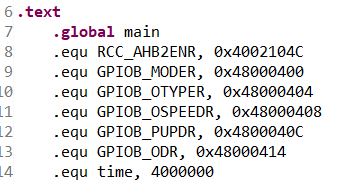
0616018 林哲宇

一、Lab3.1 : LED pattern displayer

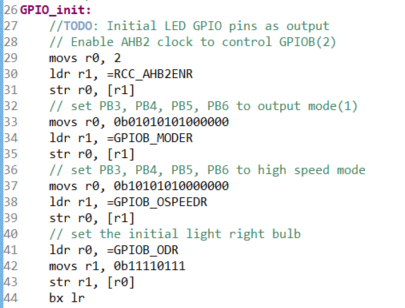
這一題的接法如下圖。黑色那一條用來接向正極，透過電阻接到燈泡同一直排中，燈泡另一個腳則接到spec建議的PB3, PB4, PB5, PB6，分別對應咖啡色、橘色、白色、紅色，對應的方式是看講義第五頁。



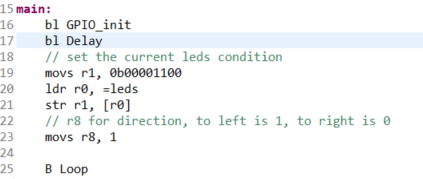
在下圖程式碼中，先設定RCC\_AHB2ENR之後可以設定啟動clock。還有GPIOB的各項資訊，需要設定的只有MODER, OSPEEDR, ODR，另外兩個則使用預設值。Time則是等等做delay要用到的常數，1 sec = 10^6 us ; 1 cycle = 0.25 us，所以換算過來，一秒總共有4000000個cycles。



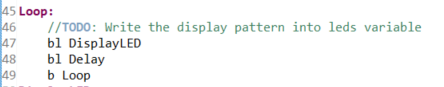
在下圖程式碼中，我初始化GPIOB的各項數值，其中因為只需要使用到B PORT，所以RCC\_AHB2ENR要設為2。PB3～PB6的MODER要設為output mode，也就是1，因為是3～6，所以右邊要空2\*3 bits。再來設定OSPEEDR，我不太懂設定speed的用意，但是看講義這樣做，我就這樣做了，看名字應該是設定反應速度吧。最後是ODR，一開始只有最右邊的燈泡是亮的，所以就讓第四個bits是0。



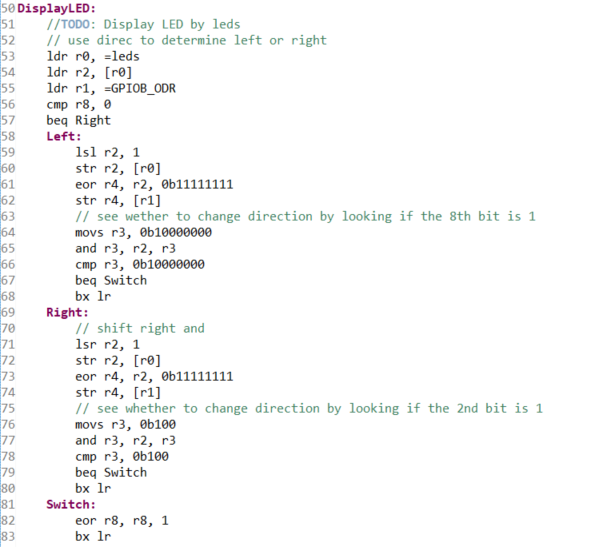
main中我有設定一些變數，一個是spec給的leds，還有用來代表方向的r8。leds表示哪幾個燈泡是亮的，亮的則為1，因為等等shift之後補位會變成0。之所以要讓第三個bit也設為1是因為等等向左移動或向右移動時，會有兩個燈泡一起亮著，所以先把那個bit設好比較方便。接著是r8，r8用來當作方向，向左的話是1，向右則為0，之後程式碼會用r8來做一些判斷。



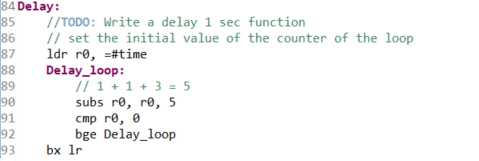
程式碼流程是這樣：先顯示LED燈的變化，再來要停1秒，之後反覆執行。



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。



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。

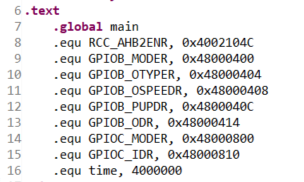


心得感想：第一題花爆多時間的，主要是因為我對硬體一點概念都沒有，只好穩穩地從頭開始看講義，因此花了非常多的時間。一開始連線怎麼街都不知道，更不用說哪個port有什麼功效，直到來回看了五次以上的講義才大概明白。

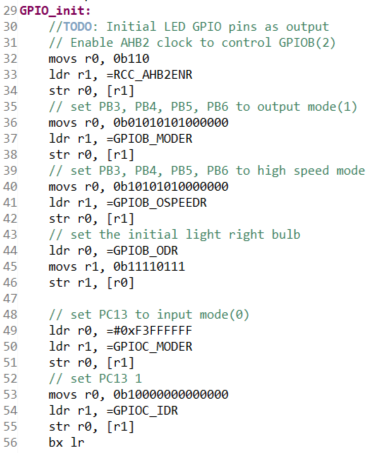
二、Lab3.2 : Push button

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。



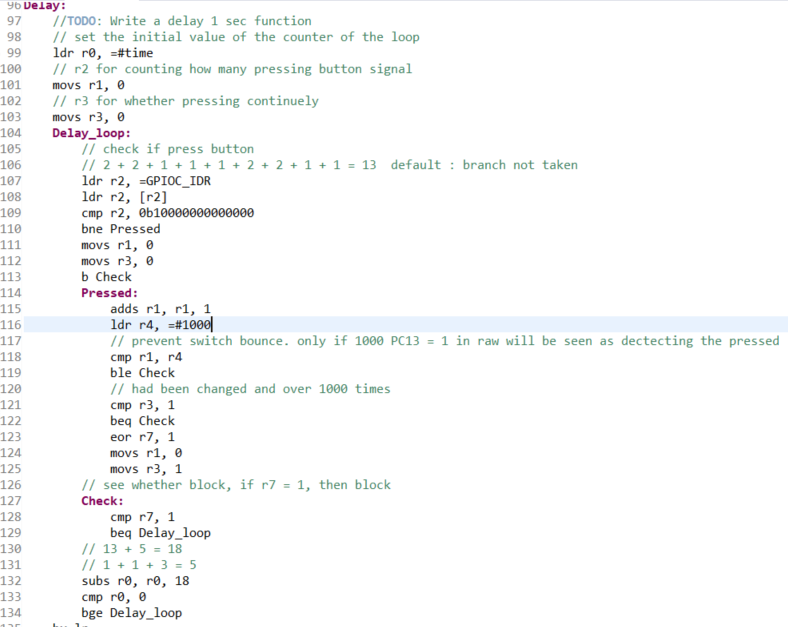
這邊跟上一題不一樣的地方就是RCC\_AHB2ENR要設為0b110，因為需要用到B, C PORT。要把PC13的MODE設為input mode，即是0。然後PC13的IDR一開始要設為1，按下去則變成0。



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

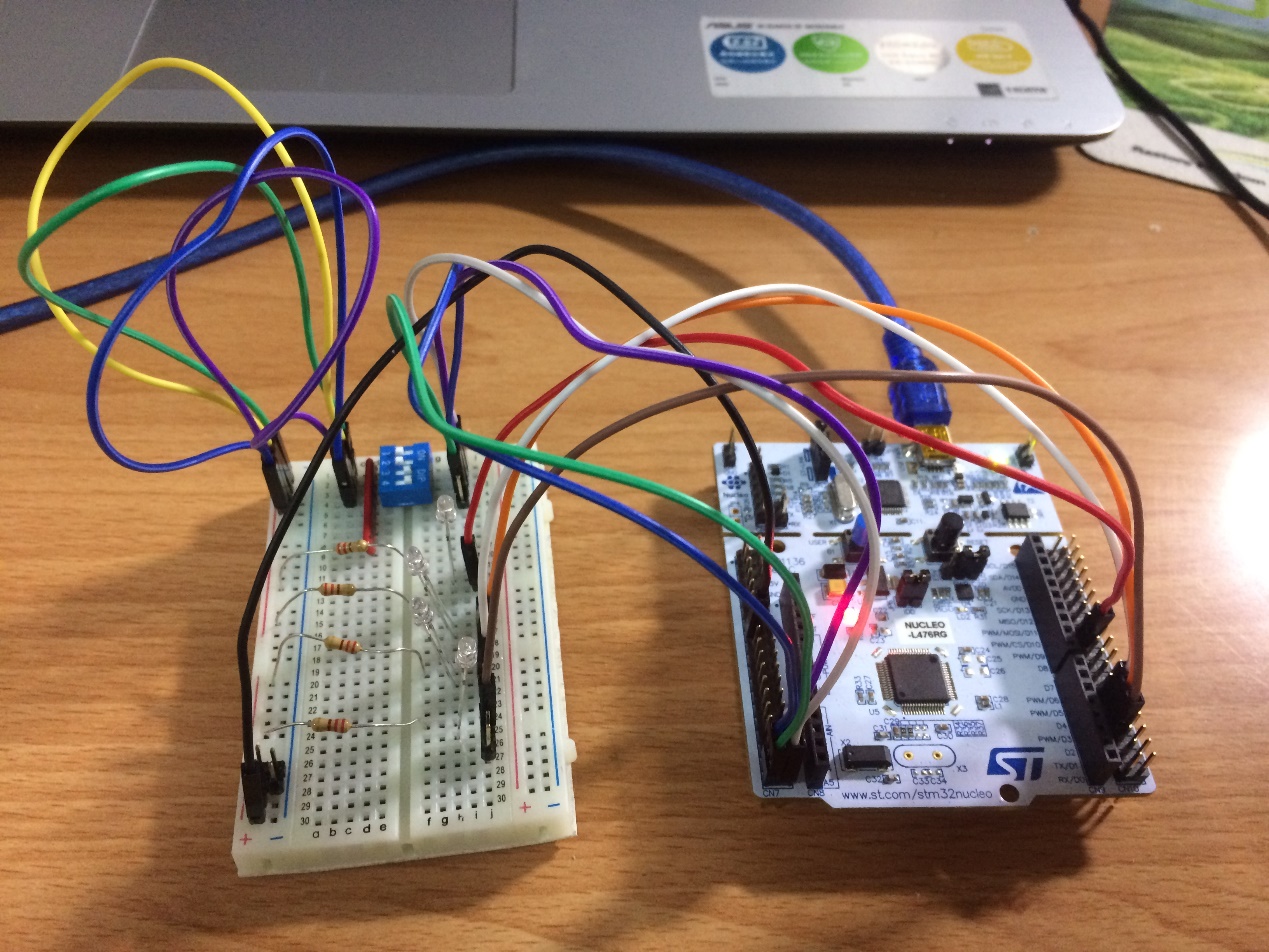


心得感想：

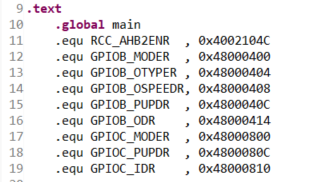
第二題花的時間大概是第一題的四分之一吧。有了第一題的基礎，第二題好做許多。基本上沒有遇到什麼瓶頸，唯一困惑的部分是：按鈕的部分不知道怎麼Debug，因為和燈泡不一樣，按下去之後繼續跑，memory似乎也沒有改變。所以這題是先寫完直接肉眼Debug，還好一次就對了。

三、Lab3.3 : 密碼鎖

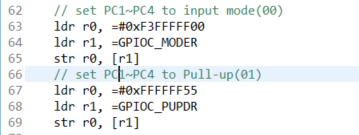
這題也是延續第一題和第二題的，就是多了密碼鎖還有燈泡亮的方式要改變，所以我多設定GPIOC的部分，將PC0～PC3分別接給密碼鎖的0123，另外因為電阻不夠用，所以用線接了四條出來用，並放上一排電阻接通。其餘都和第一題一樣。



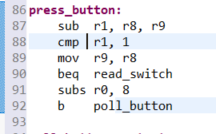
這題我繼續用GPIOC，多使用了 PUPDR，因為需要先設定成Pull-up(01)。



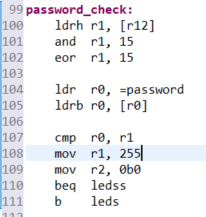
GPIO\_init比之前多的部分就只是多設定PC0～PC3的input mode和Pull-up而已。



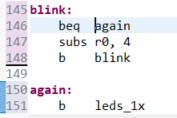
其中一個重點就是要偵測是否有按下按鈕，如果有的話則準備要判斷password是否正確。這會放在下面截圖。



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。



如果判斷完，不管是否正確都會需要blink。



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。