實驗二 實驗結報

0316213 蒲郁文 & 0316323 薛世恩

實驗名稱

Lab4: STM32 GPIO System

實驗目的

熟悉基本 GPIO 以及 麵包板
學習如何 debounce

實驗步驟

PART1.

```
        .syntax unified
        .cpu cortex-m4
        .thumb

.data
        leds: .byte 0
        .align

.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

@ r0 gpio output destination
@ r8 left or right direction flag
@ r9 light or close leds

main:
        bl      gpio_init
        mov     r9, 0b00001100
```

```
        ldrb r8, =leds
        strb r9, [r8]
        mov    r8, 0b0
        b       loop

gpio_init:
        movs r0, 0b10
        ldr    r1, =RCC_AHB2ENR
        str    r0, [r1]

        movs r0, 0b01010101000000
        ldr    r1, =GPIOB_MODER
        ldr    r2, [r1]
        and    r2, 0xFFFFC03F
        orrs r2, r0
        str    r2, [r1]

        movs r0, 0b10101010000000
        ldr    r1, =GPIOB_OSPEEDR
        strh r0, [r1]

        ldr    r0, =GPIOB_ODR
        bx      lr

loop:
        bl      display_led

        ldr    r5, =4000000
        movs r5, r5
        bl      delay

        cmp    r8, 0b0
        it      eq
        bleq move_left

        cmp    r8, 0b1
        it      eq
        bleq move_right
```

```
        cmp    r9, 0b11000000
        it     eq
        bleq set_right_flag

        cmp    r9, 0b00001100
        it     eq
        bleq set_left_flag

        b      loop

display_led:
        eor    r5, r9, 0xFFFFFFFF
        strh r5, [r0]
        bx     lr

move_left:
        lsl    r9, r9, 0b1
        bx     lr

move_right:
        lsr    r9, r9, 0b1
        bx     lr

set_right_flag:
        mov    r8, 0b1
        bx     lr

set_left_flag:
        mov    r8, 0b0
        bx     lr

delay:
        beq    delay_end
        subs r5, 4
        b      delay

delay_end:
```

```
    bx      lr
```

PART2

```
    .syntax unified
    .cpu cortex-m4
    .thumb

.data
    leds: .byte 0
    .align

.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

@ r0 gpio led output [init in gpio_init]
@ r7 running or stopped [init in main]
@ r8 left or right direction flag [init in main]
@ r9 turn on or off leds [init in main]
@ r10 gpio button input [init in gpio_init]
@ (r11, r12): (latest, confirmed) button value [init in main]
@ (1, 1) -> (0, 1) -> (0, 0) -> (1, 0) -> (1, 1)

main:
    bl      gpio_init
    mov    r9, 0b00001100
    ldrb r8, =leds
    strb r9, [r8]
    mov    r7, 0b0
    mov    r8, 0b0
```

```
        mov    r11, 0b1
        mov    r12, 0b1
        b      loop

gpio_init:
        movs r0, 0b110
        ldr    r1, =RCC_AHB2ENR
        str    r0, [r1]

        movs r0, 0b01010101000000
        ldr    r1, =GPIOB_MODER
        ldr    r2, [r1]
        and    r2, 0xFFFFC03F
        orrs r2, r0
        str    r2, [r1]

        ldr    r1, =GPIOC_MODER
        ldr    r0, [r1]
        ldr    r2, =0xF3FFFFFF
        and    r0, r2
        str    r0, [r1]

        movs r0, 0b10101010000000
        ldr    r1, =GPIOB_OSPEEDR
        strh r0, [r1]

        ldr    r0, =GPIOB_ODR
        ldr    r10, =GPIOC_IDR
        bx     lr

loop:
        bl     display_led

        ldr    r5, =4000000 @ cpu is 4mhz
        movs r5, r5
        bl     delay

        cmp    r7, 0b1
```

```
        beq    loop

        cmp    r8, 0b0
        it     eq
        bleq move_left

        cmp    r8, 0b1
        it     eq
        bleq move_right

        cmp    r9, 0b11000000
        it     eq
        bleq set_right_flag

        cmp    r9, 0b00001100
        it     eq
        bleq set_left_flag

        b      loop

display_led:
        eor    r5, r9, 0xFFFFFFFF
        strh r5, [r0] @ r0 output data reg
        bx     lr

move_left:
        lsl    r9, r9, 0b1
        bx     lr

move_right:
        lsr    r9, r9, 0b1
        bx     lr

set_right_flag:
        mov    r8, 0b1
        bx     lr

set_left_flag:
```

```
        mov    r8, 0b0
        bx     lr

delay:
        beq    delay_end
        ldr    r1, =0b1111111111111111
        ands r1, r5, r1
        beq    check_button @ branch every 32.768 ms
        subs r5, 8
        b      delay

check_button:
        ldrh r1, [r10] @ r10 input data reg
        lsr    r1, 13
        mov    r2, 1
        and    r1, r2
        cmp    r1, r11
        mov    r11, r1
        beq    button_confirmed
        subs r5, 8
        b      delay

button_confirmed:
        subs r1, r11, r12
        cmp    r1, 1
        mov    r12, r11
        beq    switch
        subs r5, 8
        b      delay

switch:
        eor    r7, 0b1
        subs r5, 8
        b      delay

delay_end:
        bx     lr
```

PART3

```
        .syntax unified
        .cpu cortex-m4
        .thumb

.data
        password: .byte 0b1100
        .align

.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

main:
        b       init_gpio

init_gpio:
        mov    r0, 0b110
        ldr    r1, =RCC_AHB2ENR
        str    r0, [r1]

        mov    r0, 0b01010101000000
        ldr    r1, =GPIOB_MODER
        ldr    r2, [r1]
        and    r2, 0xFFFFC03F
        orr    r2, r0
        str    r2, [r1]

        ldr    r1, =GPIOC_MODER
```

```
        ldr    r0, [r1]
        ldr    r2, =0xF3FFFF00
        and    r0, r2
        str    r0, [r1]

        mov    r0, 0b10101010000000
        ldr    r1, =GPIOB_OSPEEDR
        str    r0, [r1]

        ldr    r1, =GPIOC_PUPDR
        ldr    r0, [r1]
        ldr    r2, =0b01010101
        and    r0, 0xFFFFFF00
        orr    r0, r2
        str    r0, [r1]

        ldr    r10, =GPIOB_ODR @ leds
        ldr    r11, =GPIOC_IDR @ user button
        ldr    r12, =GPIOC_IDR @ dip switch

        mov    r0, 0b11111111
        strh r0, [r10]
        b      poll_button_init

poll_button_init:
        mov    r8, 1
        mov    r9, 1
        movs r0, 0
        b      poll_button

poll_button:
        beq    poll_button_restart
        ldr    r1, =0b1111111111111111
        ands r1, r0, r1
        beq    poll_button_check
        subs r0, 8
        b      poll_button
```

```
poll_button_check:
        ldrh r1, [r11]
        lsr    r1, 13
        mov    r2, 1
        and    r1, r2
        cmp    r1, r8
        mov    r8, r1
        beq    poll_button_confirm
        subs r0, 8
        b       poll_button

poll_button_confirm:
        sub    r1, r8, r9
        cmp    r1, 1
        mov    r9, r8
        beq    read_switch
        subs r0, 8
        b       poll_button

poll_button_restart:
        ldr    r0, =4000000
        movs r0, r0
        b       poll_button

read_switch:
        ldrh r1, [r12]
        and    r1, 0b1111
        eor    r1, 0b1111

        ldr    r0, =password
        ldrb r0, [r0]

        cmp    r0, r1
        mov    r1, 0b11111111
        mov    r2, 0b0
        beq    leds_3x
        b       leds_1x
```

```
leds_3x:
        eor    r1, 0b11111111
        strh r1, [r10]
        add    r2, 0b1

        cmp    r2, 6
        beq    poll_button_init

        ldr    r0, =2000000
        movs r0, r0
        b      leds_3x_blink

leds_3x_blink:
        beq    leds_3x_again
        subs r0, 4
        b      leds_3x_blink

leds_3x_again:
        b      leds_3x

leds_1x:
        eor    r1, 0b11111111
        strh r1, [r10]
        add    r2, 0b1

        cmp    r2, 2
        beq    poll_button_init

        ldr    r0, =2000000
        movs r0, r0
        b      leds_1x_blink

leds_1x_blink:
        beq    leds_1x_again
        subs r0, 4
        b      leds_1x_blink
leds_1x_again:
        b      leds_1x
```

實驗結果與問題回答

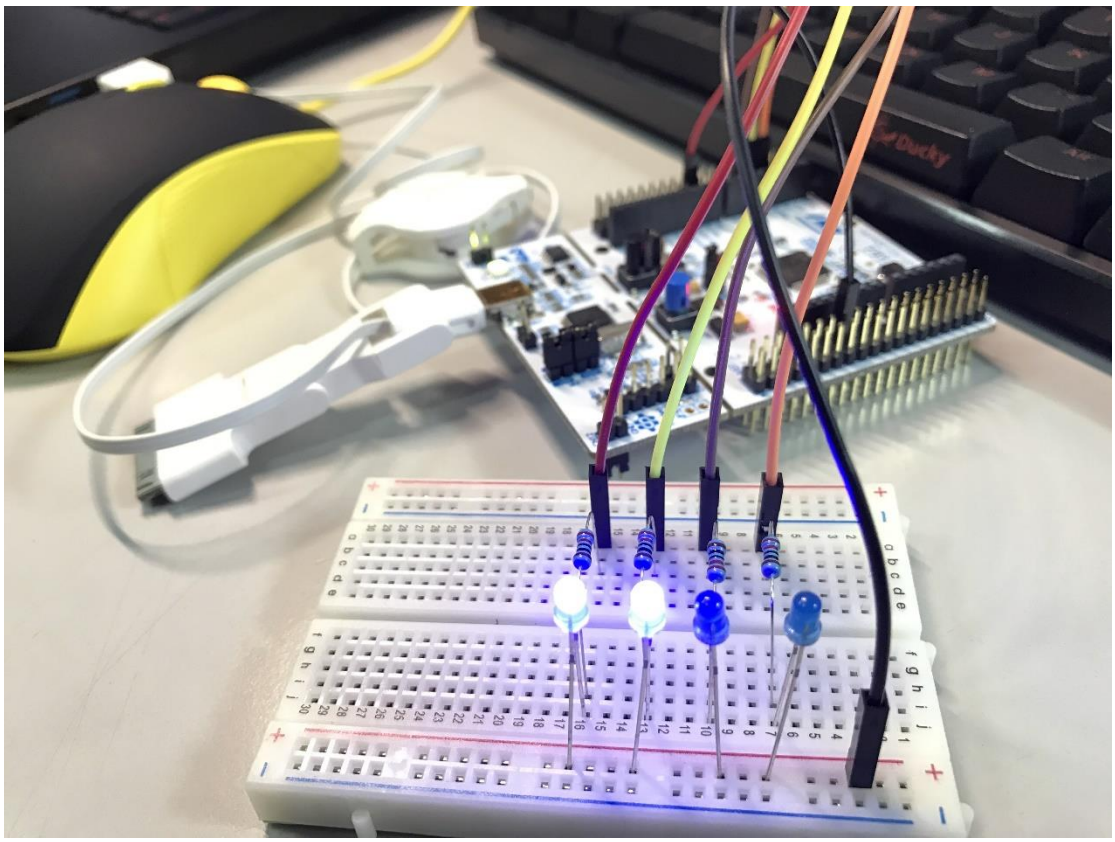
PART1.：首先設定記憶體位置，首先先將其初始化，然後在loop一跑，當我們要做delay1秒時，首先先算出指令要多少時間，然後再調整數字，每一次跑要減去多少，然後當他減到0時就beq跳出，於是就會造成一秒delay。然後儲存LED現在是往左跑還是往右跑，當碰觸到左右邊界時，就改變其方向。

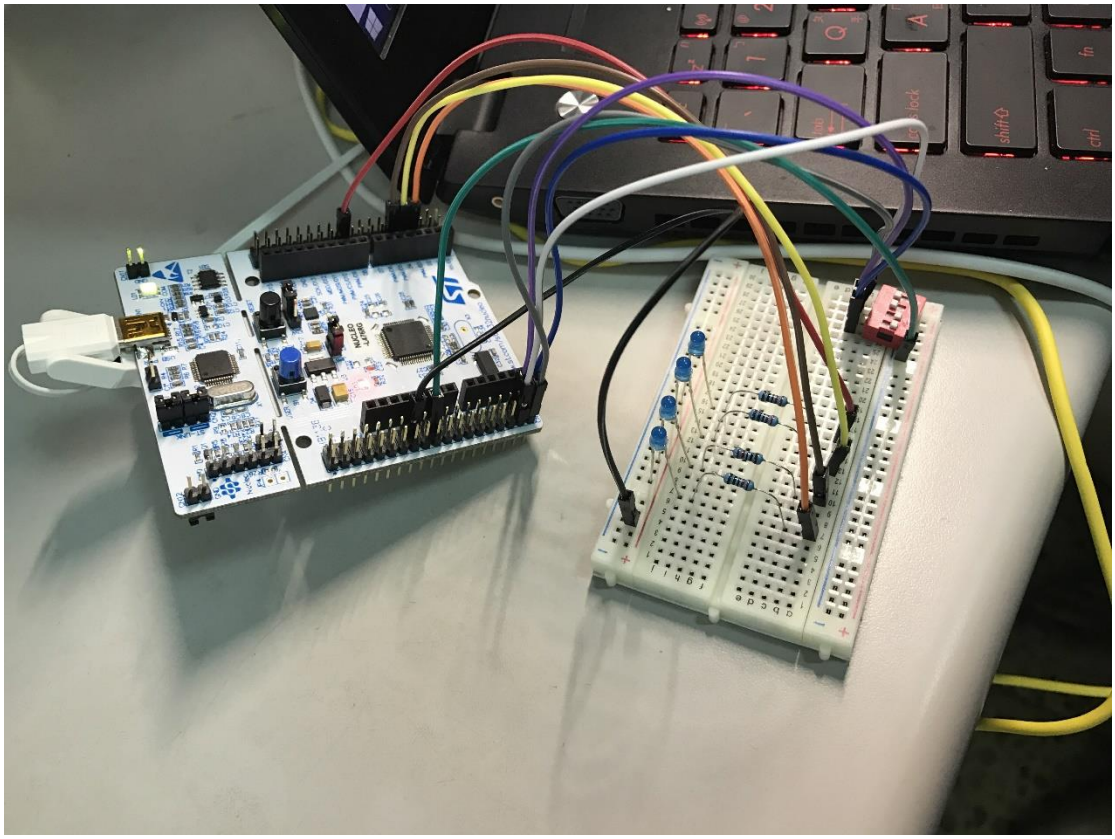
PART2：

基本上由第一題改寫，主要就是debounce，計算出一個數字(也就是debounce抓取數據的區間)，我們抓大概32ms，於是當每32ms我們就抓取，然後判斷新舊值是0還是1，依此計算有沒有按下按鈕，當按下按鈕時，就進去讓他停止不動的區塊，因此可以造成，按下去就停住的效果，當再次感應到按鈕被按下去，就繼續開始跑。(r7是我們的run or stop的flag)

PART3：

設定CPIOC為switch輸入，然後GPIOB為LED輸出，也是由第一題跟第二題改寫，當botton沒有被按下去的時候就一直跑自己的loop，當按鈕被按下去時就向下執行READ PID，然後比較他跟密碼是否相同，如果相同就跳進leds_3x否則就跳進leds_1x，兩個基本上只有閃爍次數不同，由counter來實作。然後做完在跳回botton還沒被按的狀態，繼續等待botton被按。





心得討論與應用聯想

這真的是我做過微處理機最累的一次，首先是接電路完全跳脫我學電子電路時的知識，如何在線材不夠時還要接出相同效果之電路。
也學習到如何應付 active low 的條件，輸入時碰到 portA 的 2 跟 3 對於輸入沒有反應，但是將其改成 portC 時即可，這個錯誤修正就弄了兩個小時。
PBIO 控制真的有點複雜，每個東西都要經過一大堆計算，跟一大堆設計好的數字，才能完美的組合出結果，看到成果時真的覺得很感動。
研究各種電路元件要怎麼使用，其基本性質以及結構。