Lab3 STM32 GPIO System
實驗三 STM32 GPIO System

1. Lab objectives 實驗目的

- Understand the principle of using STM32 basic I/O port.
- Design a simple LED marquee program.
- Understand the use of buttons and DIP switches.
- Understand the use of color code resistor and network resistor.
- 了解 STM32 基本輸出入 I/O port 使用原理。
- 設計簡易 LED 跑馬燈程式
- 了解按鈕與指撥開關使用原理
- 了解色碼電阻與排阻使用原理

2. Lab principle 實驗原理

Please refer to the course GPIO lecture slide, STM32L4x6 Reference manual and the lab note.

請參考上課 GPIO 講義, STM32L4x6 Reference manual 及 lab note。

3. Steps 實驗步驟

3.1. LED scroller, LED 跑馬燈

Requirement: Please construct a circuit containing 4 active low LEDs. That is, the LED will be turned off when the corresponding GPIO pin outputs high potential (VDD), and it will be turned on when the low potential (VSS) is received. Then, refer to the lecture slide, finish the initialization of peripheral bus (AHB2) and GPIO pins.
Then, complete the program below and use the variable "leds" to record the LEDs' states. Using the function "DisplayLED" to set LEDs to the pattern corresponding to the variable .

請構建一個包含 4 個低態有效 LED 的電路。 也就是說，當相應的 GPIO 引腳輸出高電位（VDD）時，LED 將關閉，而當接收到低電位（VSS）時，LED 將被打開。
接著，參考章節投影片，完成週邊裝置匯流排（AHB2）及 GPIO 接腳的初始化。接著，完成以下程式碼。利用 "leds" 這個變數紀錄目前位移數值，並使用 "DisplayLED" 函式將 LED 設置為與變數對應的圖案。

```
.data
  leds: .byte 0
.text
   .global main

main:
  BL   GPIO_init
  MOVS R1, #1
  LDR  R0, =leds
  STRB R1, [R0]

Loop:
   /* TODO: Write the display pattern into leds variable */
  BL   DisplayLED
  BL   Delay
  B    Loop

GPIO_init:
  /* TODO: Initialize LED GPIO pins as output */
  // Enable AHB2 clock
  // Set pins (Ex. PB3-6) as output mode
  // Keep PUPDR as the default value(pull-up).
  // Set output speed register
  BX LR

DisplayLED:
   /* TODO: Display LED by leds */
   BX LR

Delay:
  /* TODO: Write a delay 1 sec function */
  // You can implement this part by busy waiting.
  // Timer and Interrupt will be introduced in later lectures.
```

Waring: Please do not use PA13 and PA14 as I/O pins or you will find your debugger is malfunctioning. By default they are shared with SWD signals connected to ST-LINK/V2-1. You can connect your LEDs to PB3, PB4, PB5, PB6 on board.

請不要將 PA13 和 PA14 用作 I/O 引腳，否則會發現除錯工具發生故障。默認情況下，它們與連接到 ST-LINK/V2-1 的 SWD 信號共享。你可以連接你的 LDEs 至實驗板上的 PB3, PB4, PB5, PB6。

Pattern requirement:

In the beginning, the rightmost LED is on. Later, The lights shift left in order every one second until reaching the left end and then change the shifting direction to the right. During the process, there should be two LEDs light up, except for when they reach the left end and the right end. See the figure below.

在一開始，最右邊的 LED 亮起。 接著，燈光每秒鐘按順序向左移動直到抵達左端，然後將改變移動方向為右。 在此過程中，除了到達左端和右端之外應該有兩個 LED 點亮。見下圖。

	LED0	LED1	LED2	LED3
t = 0s	0	0	0	1
t = 1s	0	0	1	1
t = 2s	0	1	1	0
t = 3s	1	1	0	0
t = 4s	1	0	0	0
t = 5s	1	1	0	0
t = 6s	0	1	1	0
t = 7s	0	0	1	1

Repeat (t = 0s) to (t = 8s)

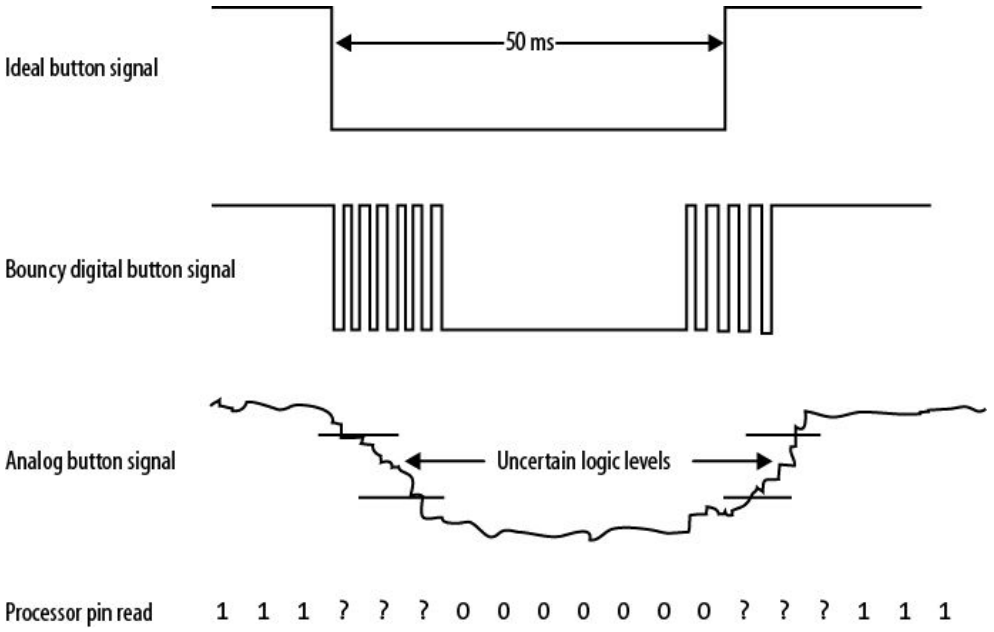
"1" represents that LED is light up, and "0" represents LED is blind.

"1" 代表 LED 亮，"0" 代表 LED 暗

3.2. Push button switch 按鈕開關

The button is a mechanical device. After pressing, the internal knot will bounce back and forth for a few milliseconds before it stabilizes. During the time when the bounce is removed, both low and high signals are detected, causing false positives. This phenomenon is called "mechanical bounce" or "switch bounce".

按鍵開關是一種機械裝置。按壓後，在穩定之前內部連結會在幾毫秒間來回彈跳。在彈跳解除前的這段時間裡，low 和 high 的訊號都會偵測到，並造成誤判。這種現象稱作「機械彈跳」或「按鍵彈跳」。



Requirement: Our development board provides a built-in blue user button which is connected to the I/O PC13 of the STM32 microcontroller. Please initialize GPIO PC13 as pull-up input and solve the mechanical bounce problem using software debounce tricks.
Then, design a polling program to read the state of the user button on board, and use the button to control the scrolling of the LEDs (Require 2-1). Once we press the button, the LEDs will pause scrolling and it should resume scrolling when we press the button again.

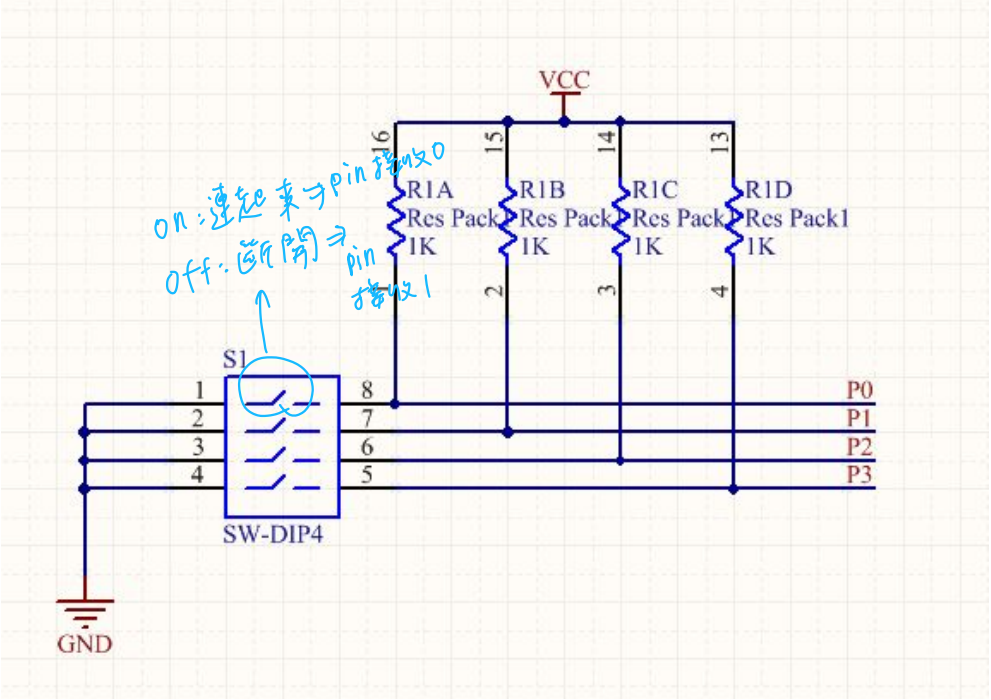
我們的開發板提供了一個內建的藍色使用者按鈕，該按鈕連接到 STM32 微控制器的 I/O PC13。 請初始化 GPIO PC13 作為上拉輸入，並使用軟體解彈跳技巧來解決機械彈跳問題。
接著，設計一個輪詢程式讀取使用者按鈕的狀態。然後利用這個按鈕控制 LED 的滾動（Require 2-1）。 按下按鈕後，LED 將暫停滾動，當再次按下按鈕時，它將從繼續滾動。

```
.data
  leds: .byte // (or leds: .word)

.text
   .global main

main:
  BL   GPIO_init
  //(option) Test! Turn on all LEDs

Loop:
   /* TODO: Check the button status to determine whether to
            pause updating the LED pattern*/
  BL CheckPress
  BL DisplayLED
  BL   Delay
  B    Loop

CheckPress:
  /* TODO: Do debounce and check button state */

DisplayLED :
   /* TODO: Display LED by leds */
   BX LR

GPIO_init:
  /* TODO: Initialize LED, button GPIO pins */
  // Enable AHB2 clock

  /* Set LED gpio output */
  // Set gpio pins as output mode
  // Keep PUPDR as the default value(pull-up)
  // Set output speed register

  /* Set user button(pc13) as gpio input */
  // set PC13 as input mode
  // Set PC13 as Pull-up

  BX LR

Delay:
  /* TODO: Write a delay 1 sec function */
  // You can implement this part by busy waiting.
  // Timer and Interrupt will be introduced in later lectures.
  BX LR
```

3.3. combination lock 密碼鎖

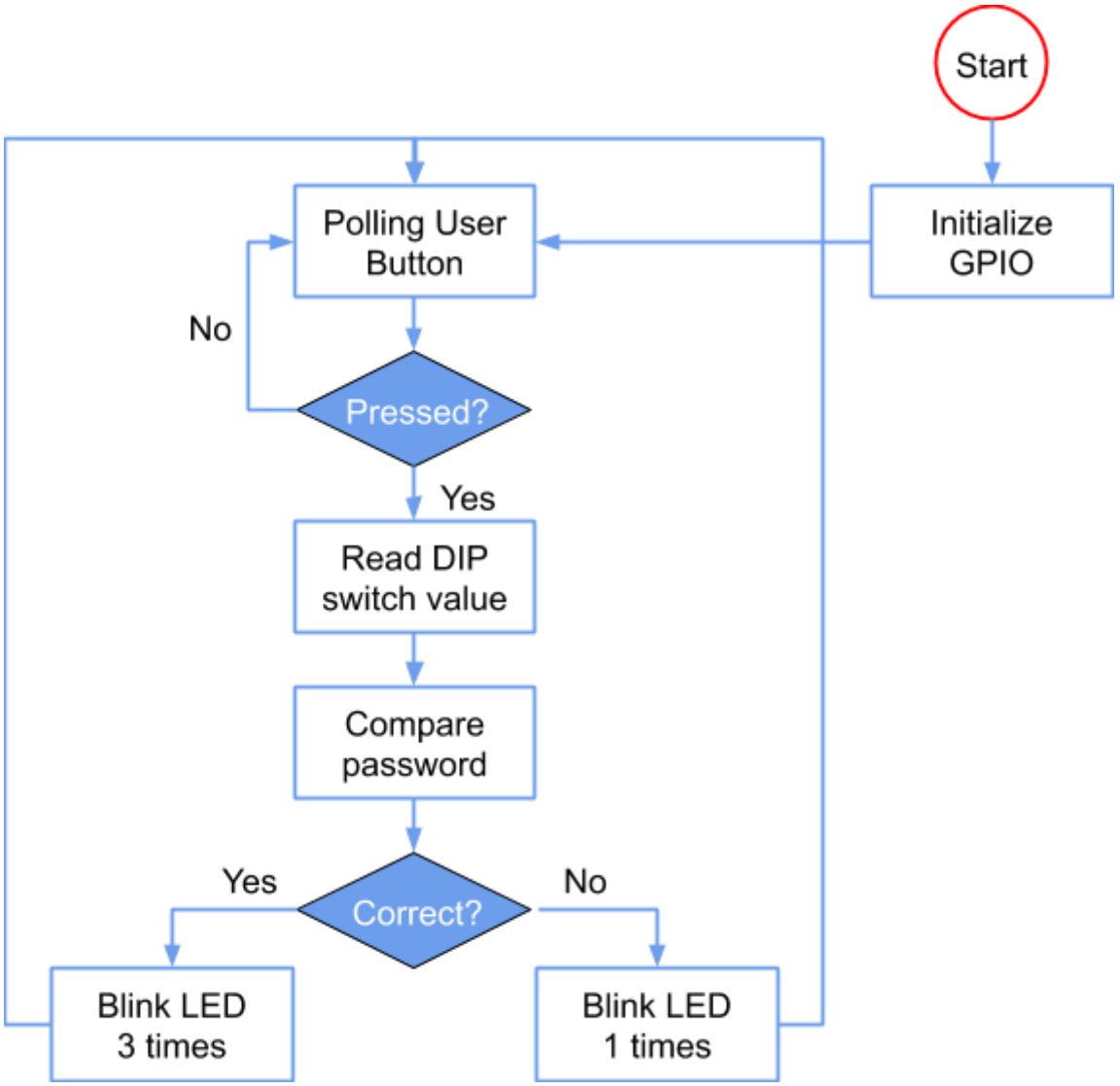
Requirement: Please use the breadboard to construct an active low DIP switch circuit and connect P0~P3 to GPIO pins on board (You could choose the pins by yourselves). Therefore, when we turn on the switch the GPIO pins will receive the low potential. For instance, if we short pin-1 and pin-8 of the DIP switch on the following figure, then, P0 will receive the low potential.
Then, declare a 1-byte global variable "password" and implement a simple 4 bits combination lock. When we press the user button, it will read the password from the DIP switch and check the correctness, and then show the result by blinking the LED. If the password is correct, blink the LED three times and one time for incorrect. (You can use the on-board user LED which is connected to the I/O PA5).

請使用麵包板構建一個低態有效的指撥開關電路，並將 P0～P3 連接至開發板上的 GPIO 引腳（您可以自行選擇引腳）。 則當我們打開開關時，GPIO 引腳將接收到低電位。 例如，如果我們將下圖中指撥開關的 pin-1 和 pin-8 短路，則 P0 將接收到低電位。
接著宣告一個單字節的全局變數 "password"，並實現簡單的 4 位元密碼鎖。當我們按下使用者按鈕時，它將從指撥開關中讀取密碼並檢查正確性，然後通過閃爍 LED 來顯示結果。正確時閃三次，錯誤時閃一次。（您可以使用板子上的使用者LED它被連接到PA5）。



Note: We did not provide any 1KΩ resistor, instead we provided and A102J network resistor. Therefore, you must use it to implement your circuit.

我們並未提供任何 1KΩ 的電阻。 因此，必須使用 A102J 網絡電阻器來實現電路。



Note: We define DIP switch ON as "1", OFF as "0". Thus, when users input "ON ON OFF OFF", it will be encoded as "1 1 0 0". Please set the blink frequency to about 0.5s.

Note: DIP switch ON 代表 "1", OFF 代表 "0", 若使用者輸入 "ON ON OFF OFF" 則它會被編碼成 "1 1 0 0"。請將閃爍間隔設定爲 0.5 秒。

3.4. Question 實驗課問題

Question 1-1: What is the memory-mapped I/O (MMIO)? What are its pros and cons?

什麼是 memory mapped I/O (MMIO)？它的優缺點是什麼？

I/O 和 memory 共用記憶體空間，I/O 的 port 會映射到 memory address 上（像我們用的這個 0x48000000 是 GPIOB）優：可直接做記憶體讀寫來作為 I/O 存取 缺：被 mapped 到的地址就不能當成記憶體用

Question 1-2: What is the port-mapped I/O (PMIO)? What are its pros and cons?

什麼是 port-mapped I/O (PMIO)？它的優缺點是什麼？

I/O 和 memory 各自獨立擁有自己的記憶體空間，優：不用考慮記憶體空間被 I/O 佔用 缺：需要額外的指令處理 I/O 存取

Question 1-3: When we set GPIO pin to the input mode, we also need to config the GPIOx_PUPDR register to pull-up, pull-down or floating (non-pull-up, non-pull-down). What's the effect of these settings?

當我們將 GPIO pin 設成 input mode 的時候，還需要將 GPIOx_PUPDR 設成上拉，下拉或浮動（非上拉、非下拉）。這些設置有什麼作用？

一般 output 會設成 no pull

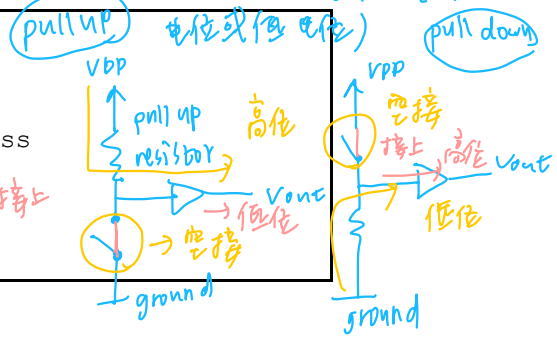
GPIO pin 設成 input mode 時，當沒有 input 訊號進來的時候，是處於高阻抗（high impedance）狀態，相當於 pin 腳所接的外界 input 沒有連到任何電路（或是想成是串接一個超大電阻），那此時如果我們去讀 pin 的值，pin 值常受環境雜訊影響（例：磁場變化→電流），而電路中的 1 或 0 是以電壓的範圍來區分的（像是可能 0~1.5 之間，我就判斷為 0，所在 3~5 我就判斷為 1），所以在 pin 空接時，讀到的電位值浮浮動動的話，那判斷 1 或 0 就可能判斷錯；需要 pull up/pull down resistor 來讓 pin 在空接時給一個穩定電位（明確的高電位或低電位）

3.5. Reference & Hint 參考資料與提示

Hint 2-1: To configure system registers such as setting GPIO mode for some pins, we may use "orr" instruction to set specific bits to 1 . Nevertheless, if we only do the "orr" instruction, it may lead to some errors. Refer to the code below, show the case that will lead to an error and patch this error.

要配置系統暫存器，例如為某些引腳設置 GPIO 模式，我們可以使用 "orr" 指令將特定位設置為 1。但是，如果僅執行 "orr" 指令，則可能會導致一些錯誤。請參考下面的程式，展示會導致錯誤的情況並修補此錯誤。

```
set_PB4_to_AF_mode:
  mov r0, #2           //GPIO AF mode
  lsl r0, r0, #8       //MODER_4 offset
  ldr r1, =0x48000400 //GPIOB_MODER base address
  ldr r2, [r1]         //load register value
  orr r2, r2, r0       //set PB4 moder
  str r0, [r1]         //store register value
```



output 的話影響的是 push-pull 和 open-drain

output 在 push-pull 下，輸出 0 表示低電位，輸出 1 表示高電位

output 在 open-drain 下，輸出 0 表示低電位，輸出 1 表示高阻抗，電位無法確定

Hint 3-1: When we need to distinguish long-presses(Ex. Press for 1 second) and short-presses (Ex. Press for 0.5 seconds) or distinguish double-click (Ex. click twice in a short time) and single-click, what are the problems to the implementation of Require 3-1-1? What modification should we make?

大部分都是用 push pull 居多

當我們需要區分長按（Ex. 按 1 秒）和短按（Ex. 按 0.5 秒）或區分雙擊（Ex. 在短時間內點擊兩次）和單擊時， Require 3-1-1 的實做面臨哪些問題？我們應該做哪些修改？

Hint 4-1: How to use A102J network resistor to generate 2kΩ equivalent circuit? How to use A102J network resistor to generate 0.5kΩ equivalent circuit?

如何使用 A102J 網絡電阻產生 2kΩ 的等效電路？如何使用 A102J 網絡電阻器產生 0.5kΩ 的等效電路？