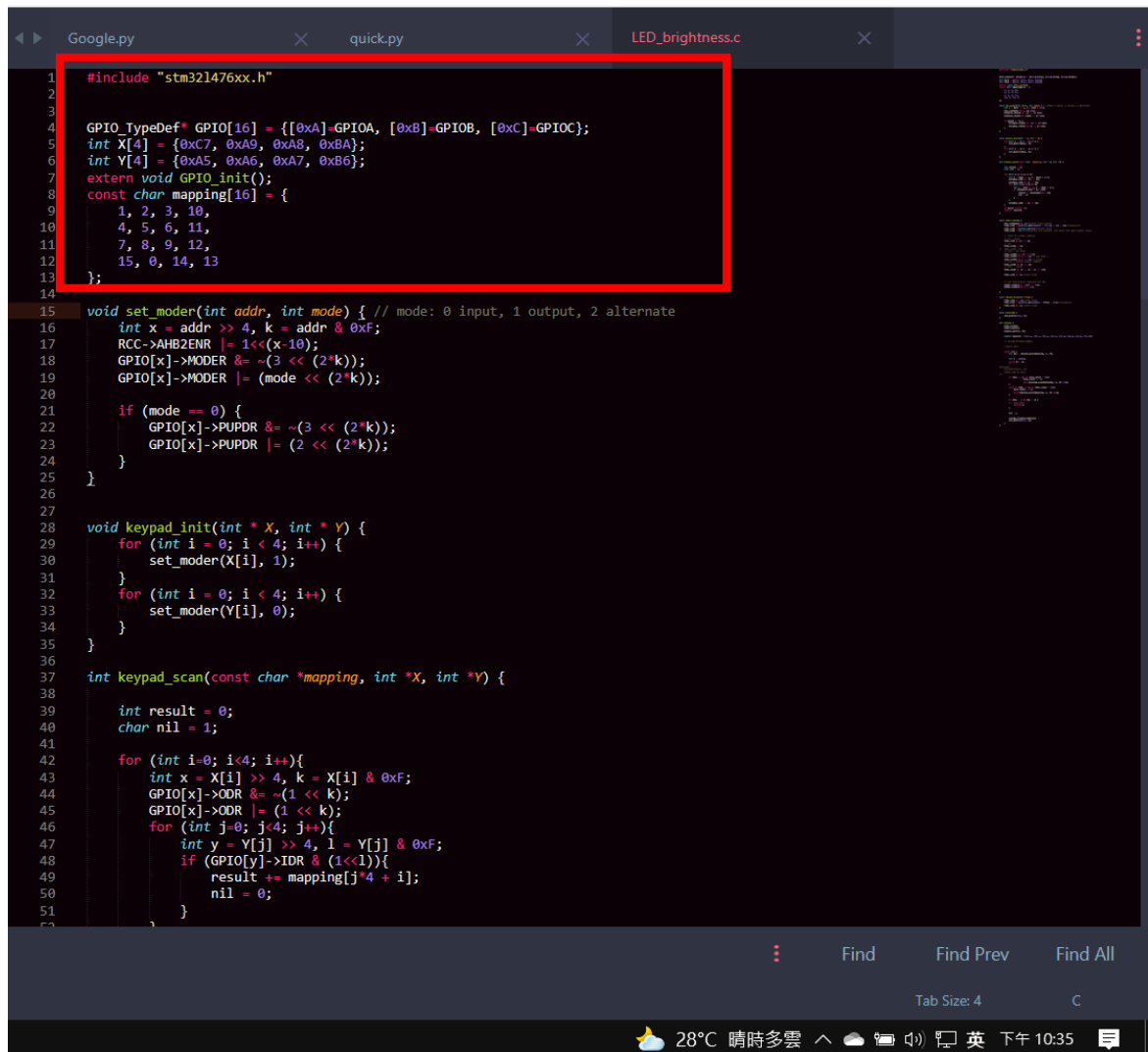


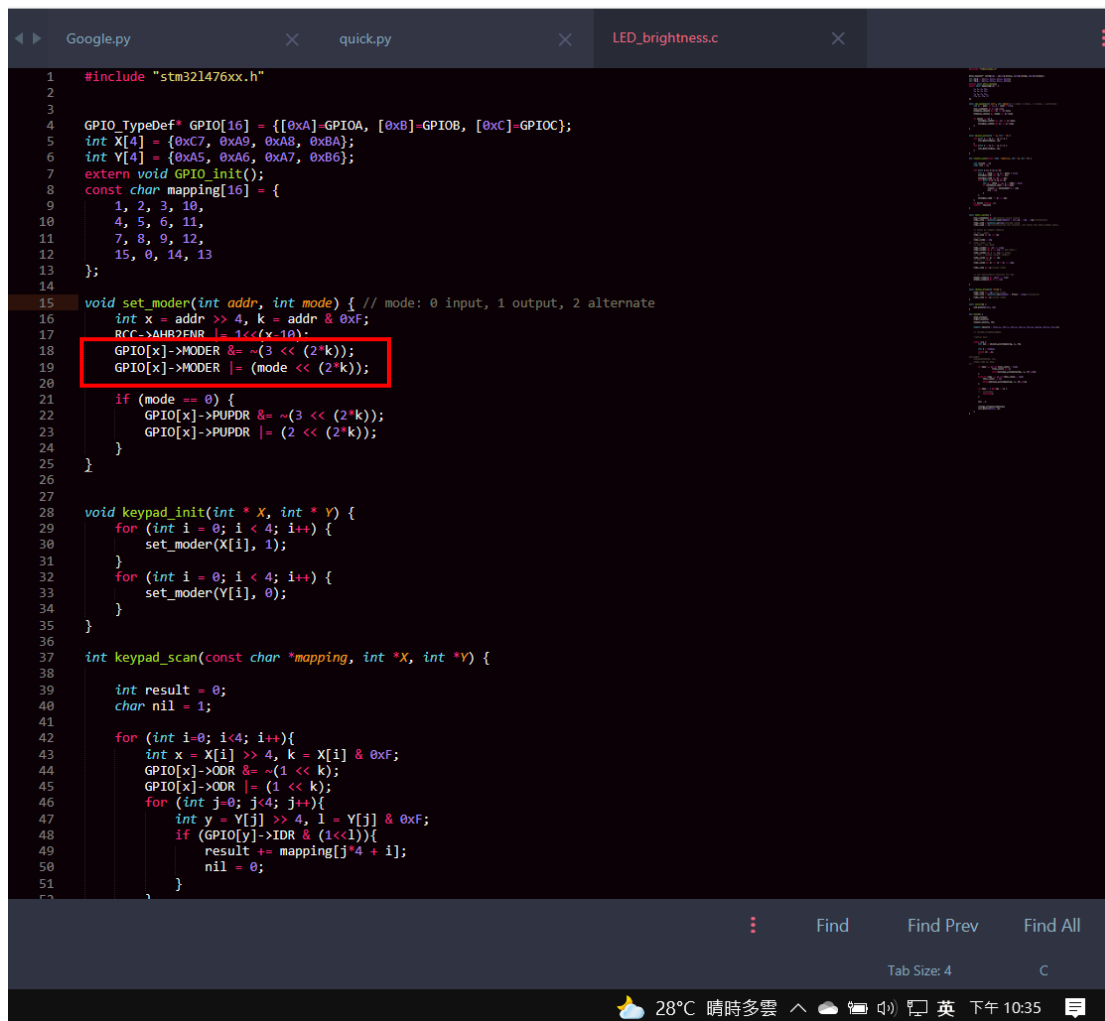
從 LAB6 開始，有一個別人寫的 func，可以簡化所有程式



```
1  #include "stm321476xx.h"
2
3
4  GPIO_TypeDef* GPIO[16] = {[0xA]=GPIOA, [0xB]=GPIOB, [0xC]=GPIOC};
5  int X[4] = {0xC7, 0xA9, 0xA8, 0xBA};
6  int Y[4] = {0xA5, 0xA6, 0xA7, 0xB6};
7  extern void GPIO_init();
8  const char mapping[16] = {
9      1, 2, 3, 10,
10     4, 5, 6, 11,
11     7, 8, 9, 12,
12     15, 0, 14, 13
13 };
14
15 void set_mode(int addr, int mode) { // mode: 0 input, 1 output, 2 alternate
16     int x = addr >> 4, k = addr & 0xF;
17     RCC->AHB2ENR |= 1<<(x-10);
18     GPIO[x]->MODER |= ~(3 << (2*k));
19     GPIO[x]->MODER |= (mode << (2*k));
20
21     if (mode == 0) {
22         GPIO[x]->PUPDR |= ~(3 << (2*k));
23         GPIO[x]->PUPDR |= (2 << (2*k));
24     }
25 }
26
27 void keypad_init(int *X, int *Y) {
28     for (int i = 0; i < 4; i++) {
29         set_mode(X[i], 1);
30     }
31     for (int i = 0; i < 4; i++) {
32         set_mode(Y[i], 0);
33     }
34 }
35
36 int keypad_scan(const char *mapping, int *X, int *Y) {
37     int result = 0;
38     char nil = 1;
39
40     for (int i=0; i<4; i++){
41         int x = X[i] >> 4, k = X[i] & 0xF;
42         GPIO[x]->ODR |= ~(1 << k);
43         GPIO[x]->ODR |= (1 << k);
44         for (int j=0; j<4; j++){
45             int y = Y[j] >> 4, l = Y[j] & 0xF;
46             if (GPIO[y]->IDR & (1<<l)){
47                 result += mapping[j*4 + i];
48                 nil = 0;
49             }
50         }
51     }
52 }
```

這個 用法先不用太特別理會他，前幾次 LAB 先用比較笨的方法寫 GPIO (因為才可以真正了解內容)。

用法如下：



```
1  #include "stm321476xx.h"
2
3
4  GPIO_TypeDef* GPIO[16] = {[0xA]-GPIOA, [0xB]-GPIOB, [0xC]-GPIOC};
5  int X[4] = {0xC7, 0xA9, 0xA8, 0xBA};
6  int Y[4] = {0xA5, 0xA6, 0xA7, 0xB6};
7  extern void GPIO_init();
8  const char mapping[16] = {
9      1, 2, 3, 10,
10     4, 5, 6, 11,
11     7, 8, 9, 12,
12     15, 0, 14, 13
13 };
14
15 void set_mode(int addr, int mode) { // mode: 0 input, 1 output, 2 alternate
16     int x = addr >> 4, k = addr & 0xF;
17     RCC->AHB2ENR |= 1<<(x-10);
18     GPIO[x]->MODER |= ~(3 << (2*k));
19     GPIO[x]->MODER |= (mode << (2*k));
20
21     if (mode == 0) {
22         GPIO[x]->PUPDR |= ~(3 << (2*k));
23         GPIO[x]->PUPDR |= (2 << (2*k));
24     }
25 }
26
27
28 void keypad_init(int *X, int *Y) {
29     for (int i = 0; i < 4; i++) {
30         set_mode(X[i], 1);
31     }
32     for (int i = 0; i < 4; i++) {
33         set_mode(Y[i], 0);
34     }
35 }
36
37 int keypad_scan(const char *mapping, int *X, int *Y) {
38     int result = 0;
39     char nil = 1;
40
41     for (int i=0; i<4; i++){
42         int x = X[i] >> 4, k = X[i] & 0xF;
43         GPIO[x]->ODR |= ~(1 << k);
44         GPIO[x]->ODR |= (1 << k);
45         for (int j=0; j<4; j++){
46             int y = Y[j] >> 4, l = Y[j] & 0xF;
47             if (GPIO[y]->IDR & (1<<l)){
48                 result += mapping[j*4 + i];
49                 nil = 0;
50             }
51         }
52     }
53 }
```

相信各位聰明的大大自己研究 **code** 可以了解其設計的原理

```
1 #include "stm321476xx.h"
2
3 GPIO_TypeDef* GPIO[16] = {[0xA]-GPIOA, [0xB]-GPIOB, [0xC]-GPIOC};
4
5 int X[4] = {0xC7, 0xA9, 0xA8, 0xBA};
6 int Y[4] = {0xA5, 0xA6, 0xA7, 0xB6};
7 extern void GPIO_init();
8 const char mapping[16] = {
9     1, 2, 3, 10,
10    4, 5, 6, 11,
11    7, 8, 9, 12,
12    15, 0, 14, 13
13 };
14
15 void set_mode(int addr, int mode) { // mode: 0 input, 1 output, 2 alternate
16     int x = addr >> 4, k = addr & 0xF;
17     RCC->AHB2ENR |= 1<<(x-10);
18     GPIO[x]->MODER |= ~3 << (2*k);
19     GPIO[x]->MODER |= (mode << (2*k));
20
21     if (mode == 0) {
22         GPIO[x]->PUPDR &= ~3 << (2*k);
23         GPIO[x]->PUPDR |= (2 << (2*k));
24     }
25 }
26
27 void keypad_init(int *X, int *Y) {
28     for (int i = 0; i < 4; i++) {
29         set_mode(X[i], 1);
30     }
31     for (int i = 0; i < 4; i++) {
32         set_mode(Y[i], 0);
33     }
34 }
35
36 int keypad_scan(const char *mapping, int *X, int *Y) {
37     int result = 0;
38     char nil = 1;
39
40     for (int i=0; i<4; i++){
41         int x = X[i] >> 4, k = X[i] & 0xF;
42         GPIO[x]->ODR &= ~(1 << k);
43         GPIO[x]->ODR |= (1 << k);
44         for (int j=0; j<4; j++){
45             int y = Y[j] >> 4, l = Y[j] & 0xF;
46             if (GPIO[y]->IDR & (1<<l)){
47                 result += mapping[j*4 + i];
48                 nil = 0;
49             }
50         }
51     }
52 }
```

上面是 Keypad 的簡化用法，是我參考網路上的用法。

用一個陣列來表示可以省很多草作上的麻煩~

(同前面，如果你是初學者，先不要理會這種操作，先用老師教的方法慢慢設定 GPIO 的 bit，到後期 GPIO 使用熟了<再用這些方法節省寫 code 的時間

Keypad init

是每次是用前必須先初始化

Keypad_scan 是一個我自己寫好的 func，因為太多 LAB 要用它了先寫好比較方便

最後，資料夾內有一檔案為 ReadMe.txt 內有

Spec 網址，可以參考一下