微處理機系統實習　　Lab5

班級：資訊三乙 學號：D1210799 姓名：王建葦

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

◆Lab5.1 — 四位數字電子鎖系統

- 利用 Keypad、LCD、七段顯示器、蜂鳴器及 RGB LED 實作簡易電子鎖。
- 按鍵 1~6 用於密碼輸入；
- R 鍵隨機產生四位數密碼並於七段顯示器上顯示；
- C 鍵清除 LCD 畫面（不清除密碼）；
- 鍵確認輸入結果，判斷是否開鎖。
- 若輸入正確，LCD 顯示「PASS」，LED 執行跑馬燈動畫；若錯誤，LCD 顯示「ERROR」並鳴響蜂鳴器；若未輸入，顯示「NULL」。
- 目的在於練習 多輸入控制、LCD 輸出管理、隨機數生成 與 多模組協同控制。

◆Lab5.2 — 二進位數顯示器

- 按下 R 鍵後，LCD 第一行顯示 8 位亂數（格式：0bxxxxxxxx），可透過不同方向鍵改變數值：
  - 「1>」與「<1」分別表示向右／左推入 1；
  - 「0>」與「<0」表示向右／左推入 0；
  - 鍵清除為全 0；
  - 「U」、「S」、「X」三鍵切換顯示模式（無號、有號、十六進位）。
- 七段顯示器會即時顯示結果，不可閃爍。此實驗練習 位元運算、格式化顯示 與 多模式資料呈現。

二、【遭遇的問題】：

What problems you faced during design and implementation?

- Lab5.1 中若使用 srand(time(NULL)) 初始化亂數會報錯，顯示 time 未定義。
- LCD 顯示超出四行後無法正常輸出新資料。
- 七段顯示器顯示密碼或二進位數時，若刷新延遲太短會造成閃爍。
- 電子鎖密碼比對邏輯初期出現錯誤，輸入正確仍顯示「ERROR」。
- 在 Lab5.2 的有號顯示模式中，負號與數字對齊方式不易處理。

三、【解決方法】：

How did you solve the problems?

- 改以內部變數 count_seed 為亂數種子，結合 SysTick->VAL 與 GPIO PIN 資訊增加隨機性。
- 透過 lcd_line_now 控制 LCD 當前行數，避免超出範圍時繼續輸出，並在 do_clear() 中重設。
- 在七段顯示部分加入 CLK_SysTickDelay(3000) 控制掃描延遲，使畫面穩定不閃爍。
- 改進密碼比對流程，先確認輸入長度為 4，再逐位比對陣列 inbuf[] 與 secret[]。
- 有號數模式中以 MINUS_CODE=16 代表負號，在 make_digits_S() 中根據數字位數自動

調整負號顯示位置。

- 實作函式 right_led_chaser() 與 LED_Running()，以多層次跑馬燈呈現「PASS」動畫；錯誤則以蜂鳴器 beep(120) 提示。

在找尋這些問題的解決方法與問題點時，我有使用 ChatGPT 協助我找尋與解決問題。包含實驗結報的內容修改與潤飾都有使用 ChatGPT 協助。

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

- 若長時間執行後產生亂數仍有機率重複，推測因 count_seed 遞增過快導致分佈不均。
- 在 LCD 顯示同時進行七段刷新時，偶爾仍會出現短暫閃爍，需進一步以中斷或多執行緒改善。
- 由於鍵盤掃描採輪詢方式，長按按鍵仍會重複觸發，未使用中斷式防彈跳（debounce）技術。

五、【程式碼】：

Lab 5.1:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "NUC100Series.h"
#include "GPIO.h"
#include "SYS.h"
#include "SYS_init.h"
#include "LCD.h"
#include "Scankey.h"
#include "Seven_Segment.h"

#define PIN_LED_R   PA12
#define PIN_LED_G   PA13
#define PIN_LED_B   PA14
#define PIN_BUZZER  PB11

#define LED_ACTIVE_LOW 1
#if LED_ACTIVE_LOW
  #define LED_ON(p)   ((p)=0)
  #define LED_OFF(p)  ((p)=1)
#else
  #define LED_ON(p)   ((p)=1)
  #define LED_OFF(p)  ((p)=0)
#endif

#define K1 1
#define K2 2
#define K3 3
#define K4 4
#define K5 5
#define K6 6
#define KR 7
#define KC 8
#define KO 9
#define RLED_ACTIVE_LOW 1
static uint8_t secret[4] = {1,2,3,4};
static uint8_t inbuf[4];
static uint8_t inlen = 0;
static uint32_t count_seed = 0;
```

```
40  static uint8_t lcd_line_now = 0;
41  static const uint16_t RLED_MASK =
42      (BIT8 | BIT9 | BIT10 | /* skip BIT11 */ BIT12 | BIT13 | BIT14 | BIT15);
43
44  static void right_leds_init(void){
45    GPIO_SetMode(PB, RLED_MASK, GPIO_MODE_OUTPUT);
46  #if RLED_ACTIVE_LOW
47    /* all OFF = high */
48    if (RLED_MASK & BIT8)  PB8  = 1;
49    if (RLED_MASK & BIT9)  PB9  = 1;
50    if (RLED_MASK & BIT10) PB10 = 1;
51    if (RLED_MASK & BIT12) PB12 = 1;
52    if (RLED_MASK & BIT13) PB13 = 1;
53    if (RLED_MASK & BIT14) PB14 = 1;
54    if (RLED_MASK & BIT15) PB15 = 1;
55  #else
56    if (RLED_MASK & BIT8)  PB8  = 0;
57    if (RLED_MASK & BIT9)  PB9  = 0;
58    if (RLED_MASK & BIT10) PB10 = 0;
59    if (RLED_MASK & BIT12) PB12 = 0;
60    if (RLED_MASK & BIT13) PB13 = 0;
61    if (RLED_MASK & BIT14) PB14 = 0;
62    if (RLED_MASK & BIT15) PB15 = 0;
63  #endif
64  }
65
66  static void right_leds_all_off(void){
67  #if RLED_ACTIVE_LOW
68    if (RLED_MASK & BIT8)  PB8  = 1;
69    if (RLED_MASK & BIT9)  PB9  = 1;
70    if (RLED_MASK & BIT10) PB10 = 1;
71    if (RLED_MASK & BIT12) PB12 = 1;
72    if (RLED_MASK & BIT13) PB13 = 1;
73    if (RLED_MASK & BIT14) PB14 = 1;
74    if (RLED_MASK & BIT15) PB15 = 1;
75  #else
76    if (RLED_MASK & BIT8)  PB8  = 0;
77    if (RLED_MASK & BIT9)  PB9  = 0;
78    if (RLED_MASK & BIT10) PB10 = 0;
79    if (RLED_MASK & BIT12) PB12 = 0;
80    if (RLED_MASK & BIT13) PB13 = 0;
81    if (RLED_MASK & BIT14) PB14 = 0;
82    if (RLED_MASK & BIT15) PB15 = 0;
```

```
83    #endif
84    }
85
86    static void right_led_on_bit(uint16_t bit){
87      right_leds_all_off();
88    #if RLED_ACTIVE_LOW
89      if (bit == BIT8)  PB8  = 0;
90      if (bit == BIT9)  PB9  = 0;
91      if (bit == BIT10) PB10 = 0;
92      if (bit == BIT12) PB12 = 0;
93      if (bit == BIT13) PB13 = 0;
94      if (bit == BIT14) PB14 = 0;
95      if (bit == BIT15) PB15 = 0;
96    #else
97      if (bit == BIT8)  PB8  = 1;
98      if (bit == BIT9)  PB9  = 1;
99      if (bit == BIT10) PB10 = 1;
100     if (bit == BIT12) PB12 = 1;
101     if (bit == BIT13) PB13 = 1;
102     if (bit == BIT14) PB14 = 1;
103     if (bit == BIT15) PB15 = 1;
104   #endif
105   }
106
107   /* ???:? PB8?PB10?PB12?PB13?PB14?PB15(?? PB11)???? */
108   static void right_led_chaser(void){
109     uint16_t seq[7];
110     int n = 0, i;
111
112     if (RLED_MASK & BIT8)  seq[n++] = BIT8;
113     if (RLED_MASK & BIT9)  seq[n++] = BIT9;
114     if (RLED_MASK & BIT10) seq[n++] = BIT10;
115     if (RLED_MASK & BIT12) seq[n++] = BIT12;
116     if (RLED_MASK & BIT13) seq[n++] = BIT13;
117     if (RLED_MASK & BIT14) seq[n++] = BIT14;
118     if (RLED_MASK & BIT15) seq[n++] = BIT15;
119
120     for (i = 0; i < n; i++){          /* ??? */
121       right_led_on_bit(seq[i]);
122       CLK_SysTickDelay(80000);
123     }
124     for (i = n - 2; i >= 1; i--){     /* ???(?????) */
125       right_led_on_bit(seq[i]);
```

```
126         CLK_SysTickDelay(80000);
127       }
128       right_leds_all_off();
129     }
130     static void leds_off(void){
131       GPIO_SetMode(PA, BIT12|BIT13|BIT14, GPIO_MODE_OUTPUT);
132       LED_OFF(PIN_LED_R); LED_OFF(PIN_LED_G); LED_OFF(PIN_LED_B);
133     }
134     static void beep(uint32_t ms){
135       GPIO_SetMode(PB, BIT11, GPIO_MODE_OUTPUT);
136       PIN_BUZZER = 0; CLK_SysTickDelay(ms*1000); PIN_BUZZER = 1;
137     }
138     static void led_pass_chaser(void){
139       int i;
140       for (i=0; i<3; i++){
141         LED_ON(PIN_LED_R); CLK_SysTickDelay(120000); LED_OFF(PIN_LED_R);
142         LED_ON(PIN_LED_G); CLK_SysTickDelay(120000); LED_OFF(PIN_LED_G);
143         LED_ON(PIN_LED_B); CLK_SysTickDelay(120000); LED_OFF(PIN_LED_B);
144       }
145     }
146
147     static void seg_show4_digits(const uint8_t d[4]){
148       CloseSevenSegment(); ShowSevenSegment(3, d[0]); CLK_SysTickDelay(3000);
149       CloseSevenSegment(); ShowSevenSegment(2, d[1]); CLK_SysTickDelay(3000);
150       CloseSevenSegment(); ShowSevenSegment(1, d[2]); CLK_SysTickDelay(3000);
151       CloseSevenSegment(); ShowSevenSegment(0, d[3]); CLK_SysTickDelay(3000);
152     }
153     static void lcd_write_line16(uint32_t line, const char* s){
154       char buf[17];
155       int n = 0, i = 0;
156       while (s[n] && n < 16) { buf[n] = s[n]; n++; }
157       for (i = n; i < 16; i++) buf[i] = ' ';
158       buf[16] = '\0';
159       print_Line(line, buf);
160     }
161     static void lcd_show_input(void){
162       char s[17];
163       uint8_t i;
164       for (i=0; i<inlen && i<16; i++) s[i] = (char)('0' + inbuf[i]);
165       s[i] = '\0';
166       print_Line(lcd_line_now, s);
167     }
168     static void lcd_show_status(const char* msg){
```

```
169       print_Line(2, (char*)msg);
170     }
171     static void lcd_clear_status(void){
172       print_Line(2, "");
173     }
174
175     static void new_secret_and_show(void){
176       int i, t;
177       unsigned seed;
178       seed = (count_seed * 1664525u) + 1013904223u;
179       seed ^= (unsigned)SysTick->VAL;
180       seed ^= ((unsigned)PE->PIN << 8) | (unsigned)PC->PIN;
181       srand(seed);
182       for (i=0; i<4; i++) secret[i] = (rand() % 6) + 1;
183       for (t=0; t<80; t++) seg_show4_digits(secret);
184     }
185
186     static void accept_digit(uint8_t d){
187       if (inlen < 4) {
188         inbuf[inlen++] = d;
189         lcd_show_input();
190         lcd_clear_status();
191       }
192     }
193     static void lcd_show_status_append(const char* status){
194       char s[17];
195       uint8_t i;
196       uint8_t pos;
197
198       for (i=0; i<inlen && i<16; i++) s[i] = (char)('0' + inbuf[i]);
199       pos = i;
200       while (*status && pos < 16) { s[pos++] = *status++; }
201       s[pos] = '\0';
202
203       lcd_write_line16(1, s);
204     }
205
206     static void lcd_show_result_line(const char* digits, const char* status)
207     {
208       char line[17];
209       int i = 0, pos = 0;
210
211       if (lcd_line_now >= 4) return;
```

```
213       for (i = 0; i < 16; i++) line[i] = ' ';
214       line[16] = '\0';
215
216
217       while (digits[pos] && pos < 11) {
218         line[pos] = digits[pos];
219         pos++;
220       }
221
222
223       i = 11;
224       while (*status && i < 16) line[i++] = *status++;
225
226       print_Line(lcd_line_now, line);
227       lcd_line_now++;
228     }
229
230     void LED_Running(void)
231     {
232         int i, bit;
233         int dir = 1;  // ??:1 = ???,-1 = ???
234         int start = 12;
235         int end = 15;
236         int current;
237
238         for (i = 0; i < 6; i++)  // ????? 3 ? (6 ?????)
239         {
240             if (dir == 1)  // ???
241             {
242                 for (current = start; current <= end; current++)
243                 {
244                     PC->DOUT &= ~(1 << current);   // LED ?
245                     CLK_SysTickDelay(100000);
246                     PC->DOUT |= (1 << current);    // LED ?
247                 }
248             }
249             else  // ???
250             {
251                 for (current = end; current >= start; current--)
252                 {
253                     PC->DOUT &= ~(1 << current);   // LED ?
254                     CLK_SysTickDelay(100000);
255                     PC->DOUT |= (1 << current);    // LED ?
```

```
256                 }
257             }
258             dir = -dir;  // ????
259         }
260     }
261
262
263
264     static void do_open(void)
265     {
266         uint8_t pass = 1;
267         uint8_t i;
268         char digits[8];
269
270         if (lcd_line_now >= 4) return;
271
272         if (inlen == 0) {
273             lcd_show_result_line("", "NULL");
274             return;
275         }
276
277         for (i=0; i<inlen && i<4; i++) digits[i] = '0' + inbuf[i];
278         digits[i] = '\0';
279
280         if (inlen != 4) pass = 0;
281         else {
282             for (i=0; i<4; i++) {
283                 if (inbuf[i] != secret[i]) { pass = 0; break; }
284             }
285         }
286
287         if (pass) {
288
289             lcd_show_result_line(digits, "PASS");
290             right_led_chaser();
291                     LED_Running();
292
293         } else {
294             lcd_show_result_line(digits, "ERROR");
295             beep(120);
296         }
297
298         inlen = 0;
```

```
299     }
300
301
302
303     static void do_clear(void){
304       clear_LCD();
305       inlen = 0;
306       lcd_line_now = 0;
307     }
308
309
310
311
312
313
314
315
316     int main(void)
317     {
318       uint8_t key;
319
320       SYS_Init();
321
322     SYS->GPA_MFP &= ~(SYS_GPA_MFP_PA14_Msk | SYS_GPA_MFP_PA15_Msk);
323     SYS->ALT_MFP &= ~(SYS_ALT_MFP_PA14_Msk | SYS_ALT_MFP_PA15_Msk);
324     SYS->ALT_MFP1 &= ~(SYS_ALT_MFP1_PA14_Msk | SYS_ALT_MFP1_PA15_Msk);
325
326
327
328       init_LCD(); clear_LCD();
329       OpenKeyPad();
330       OpenSevenSegment();
331       leds_off();
332       GPIO_SetMode(PB, BIT11, GPIO_MODE_OUTPUT); PIN_BUZZER = 1;
333     right_leds_init();
334       new_secret_and_show();
335
336       while(1){
337         seg_show4_digits(secret);
338         count_seed++;
339
340
341         key = ScanKey();
```

```
342         if (key==0) continue;
343 
344         switch (key){
345           case K1: accept_digit(1); break;
346           case K2: accept_digit(2); break;
347           case K3: accept_digit(3); break;
348           case K4: accept_digit(4); break;
349           case K5: accept_digit(5); break;
350           case K6: accept_digit(6); break;
351           case KR: new_secret_and_show(); break;
352           case KC: do_clear(); break;
353           case KO: do_open(); break;
354           default: break;
355         }
356 
357         CLK_SysTickDelay(120000);
358         while (ScanKey()!=0);
359       }
360     }
```

Lab 5.2:

```c
#include <stdio.h>
#include <stdlib.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "LCD.h"
#include "Scankey.h"
#include "Seven_Segment.h"
#define MINUS_CODE 16
#define SEG_BLANK 0xFF

static void bin_str(char* out, unsigned char v){
  int i;
  out[0]='0'; out[1]='b';
  for(i=0;i<8;i++) out[2+i] = (v & (1<<(7-i)))? '1':'0';
  out[10]='\0';
}

static void lcd_show_bin(unsigned char N){
  char line[17];
  int i;
  char b[11];
  for(i=0;i<16;i++) line[i]=' ';
  line[16]='\0';
  bin_str(b,N);
  for(i=0;i<10;i++) line[3+i]=b[i];
  print_Line(1, line);
}

static unsigned char dig[4]={0,0,0,0};

static unsigned char mask[4]={0,0,0,0};

static void make_digits_U(unsigned int v){
    int i;
    for(i=0;i<4;i++){ dig[i]=0; mask[i]=0; }

    if(v == 0){
        dig[0]  = 0;
        mask[0] = 1;
        return;
```

```
42          }
43
44          dig[0]=(unsigned char)(v%10); mask[0]=1; v/=10;
45          if(v>0){ dig[1]=(unsigned char)(v%10); mask[1]=1; v/=10; }
46          if(v>0){ dig[2]=(unsigned char)(v%10); mask[2]=1; v/=10; }
47          if(v>0){ dig[3]=(unsigned char)(v%10); mask[3]=1; }
48      }
49
50      // ---------- ???? (S) ----------
51      static void make_digits_S(int val){
52          int neg = (val < 0);
53          unsigned int a = (unsigned int)(neg ? -val : val);
54          int i, idx;
55
56          for(i=0;i<4;i++){ dig[i]=0; mask[i]=0; }
57
58          if(!neg){
59              if(a == 0){
60                  dig[0]  = 0;
61                  mask[0] = 1;
62                  return;
63              }
64              idx = 0;
65              while(a>0 && idx<4){
66                  dig[idx] = a % 10;
67                  mask[idx]=1;
68                  a/=10;
69                  idx++;
70              }
71          }else{
72              if(a >= 100){
73                  dig[3]=MINUS_CODE; mask[3]=1;
74                  dig[2]=(a/100)%10; mask[2]=1;
75                  dig[1]=(a/10)%10;  mask[1]=1;
76                  dig[0]=a%10;       mask[0]=1;
77              }else if(a >= 10){
78                  dig[2]=MINUS_CODE; mask[2]=1;
79                  dig[1]=a/10;       mask[1]=1;
80                  dig[0]=a%10;       mask[0]=1;
81              }else{
82                  dig[1]=MINUS_CODE; mask[1]=1;
83                  dig[0]=a;          mask[0]=1;
84              }
```

```
85          }
86      }
87
88      static void make_digits_X(unsigned char v){
89        int i;
90        for(i=0;i<4;i++){ dig[i]=0; mask[i]=0; }
91        dig[0]=(unsigned char)(v & 0x0F);
92        dig[1]=(unsigned char)((v>>4) & 0x0F);
93        mask[0]=1; mask[1]=1;
94      }
95
96      static void refresh_once(void){
97        int i;
98        for(i=0;i<4;i++){
99          CloseSevenSegment();
100         if(mask[i]){
101           ShowSevenSegment(i, dig[i]);
102         }else{
103           PE->DOUT = SEG_BLANK;
104         }
105         CLK_SysTickDelay(1000);
106       }
107     }
108
109     int main(void){
110       unsigned char N=0, mode=0;
111       unsigned char k=0, prev_k=0;
112       unsigned char prev_mode=255, prev_N=255;
113       SYS_Init();
114       init_LCD();
115       clear_LCD();
116       OpenKeyPad();
117       CloseSevenSegment();
118       lcd_show_bin(N);
119       make_digits_U(N);
120       while(1){
121         k = ScanKey();
122
123         if(prev_k==2 && k==0){
124           N=(unsigned char)(rand()&0xFF);
125         }
126
127         if(prev_k==0 && k!=0){
```

```
128           if(k==1) N=(unsigned char)((N>>1)|0x80); // 1>
129           else if(k==3) N=(unsigned char)((N<<1)|0x01); // <1
130           else if(k==4) N=(unsigned char)(N>>1); // 0>
131           else if(k==6) N=(unsigned char)(N<<1); // <0
132           else if(k==5) N=0; // 0???
133           else if(k==7) mode=0; // U
134           else if(k==8) mode=1; // S
135           else if(k==9) mode=2; // X
136         }
137
138         if(N!=prev_N || mode!=prev_mode){
139           lcd_show_bin(N);
140           if(mode==0) make_digits_U(N);
141           else if(mode==1) make_digits_S((signed char)N);
142           else make_digits_X(N);
143           prev_N=N; prev_mode=mode;
144         }
145         refresh_once();
146         prev_k=k;
147       }
148     }
```