微處理機系統實習　Lab4

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?

本次實驗包含兩部分：

Lab 4.1 — 隨機數與模數運算顯示：

透過按下 Keypad 上的 R 鍵 產生一個亂數（00～99），並在七段顯示器右側顯示該亂數。使用不同的鍵（%2、%3、%5）可進行相應的模數運算，並在最左側顯示餘數。按下 C 鍵 可清除顯示並重新開始。

此設計練習了 亂數產生（seed 設定）、多重輸入判斷（key mapping） 以及 多位數七段顯示控制 的概念。

Lab 4.2 — 紅綠燈模擬系統：

以三顆 LED 燈模擬紅綠燈運作，週期為：綠燈 8 秒 → 黃燈 5 秒 → 紅燈 13 秒循環。

同時利用七段顯示器顯示倒數秒數，並可藉由按下 G、Y、R 鍵手動切換燈號；按下 + 鍵可增加 5 秒倒數時間。

程式中以 enum 與 struct 結構化交通燈邏輯，搭配函式 TrafficSignal_initialize() 與 TrafficSignal_countDown() 提升可讀性與維護性。

二、【遭遇的問題】：

What problems you faced during design and implementation?

1. 在 Lab4.1 中，亂數若使用 srand(time(NULL)) 無法編譯，顯示 time() 未定義。
2. 七段顯示器刷新速度若太慢，畫面會閃爍或殘影。
3. 在 Lab4.2 紅綠燈模擬時，若按鍵輸入過快，有時會導致狀態切換異常。
4. 紅綠燈倒數的時間與實際 1 秒不同步，顯示更新略為不穩定。

三、【解決方法】：

How did you solve the problems?

1. 將 srand(time(NULL)) 改為以 計數器變數 seedCounter 產生種子，使每次按下 R 鍵時都能生成不同亂數。
2. 將七段顯示的延遲時間設為約 5 ms（CLK_SysTickDelay(5000)），確保多工掃描（multiplexing）下畫面穩定且不閃爍。
3. 增加變數 lastKey 判斷按鍵是否為「新按下」狀態，以避免重複觸發事件。
4. 在 Lab4.2 中透過 tick 計數器模擬約 1 秒延遲（每 900 次迴圈觸發一次倒數），使秒數顯示與 LED 狀態轉換更接近實際時間。
5. 將三顆 LED（PA12、PA13、PA14）以 Show_LED() 函式統一控制，分別代表綠、黃、紅燈，確保燈號顯示清晰且邏輯簡潔。

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?

1. 七段顯示器在切換燈號時偶爾仍會出現極短暫閃爍，可能因為多工掃描與倒數更新在同一主迴圈中競爭資源所致。
2. 按鍵防彈跳（debounce）僅以軟體延遲實現，若連續快速輸入仍有誤判情況，未實作中斷式或硬體防彈跳機制。
3. 倒數時間仍非精準 1 秒，因主迴圈執行時間受顯示刷新延遲影響，需進一步透過 Timer Interrupt 方式改善。

五、【程式碼】 ：

Lab 4.1:

```c
#include <stdio.h>
#include <stdlib.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "Seven_Segment.h"
#include "Scankey.h"

void Display_Lab4_1(int remainder, int randomNum)
{
    uint8_t d0, d1, d3;
    d1 = (randomNum / 10) % 10;
    d0 = randomNum % 10;
    d3 = remainder % 10;

    CloseSevenSegment();
    ShowSevenSegment(3, d3);
    CLK_SysTickDelay(5000);

    CloseSevenSegment();
    CLK_SysTickDelay(5000);

    CloseSevenSegment();
    ShowSevenSegment(1, d1);
    CLK_SysTickDelay(5000);

    CloseSevenSegment();
    ShowSevenSegment(0, d0);
    CLK_SysTickDelay(5000);
}
```

```c
int main(void)
{
    int key = 0;
    int lastKey = 0;
    int randomNum = 0;
    int remainder = 0;
    int seedCounter = 0;
    int clearFlag = 1;

    SYS_Init();
    OpenSevenSegment();
    OpenKeyPad();
    CloseSevenSegment();

    while (1)
    {
        seedCounter++;
        key = ScanKey();

        if (key == 0 && lastKey == 3)
        {
            srand(seedCounter);
            randomNum = rand() % 100;
            clearFlag = 0;
        }

        if (key != 0)
        {
            if (key == 1)
                remainder = randomNum % 2;
            else if (key == 4)
                remainder = randomNum % 3;
            else if (key == 7)
                remainder = randomNum % 5;
            else if (key == 9)
            {
                randomNum = 0;
                remainder = 0;
                clearFlag = 1;
                CloseSevenSegment();
            }
        }

        lastKey = key;

        if (!clearFlag)
            Display_Lab4_1(remainder, randomNum);
        else
        {
            CloseSevenSegment();
            CLK_SysTickDelay(20000);
        }
    }
}
```

Lab 4.2:

```c
#include <stdio.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "Seven_Segment.h"
#include "Scankey.h"

void Init_GPIO(void)
{
    GPIO_SetMode(PA, BIT12, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PA, BIT13, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PA, BIT14, GPIO_MODE_OUTPUT);
    PA12 = 1;
    PA13 = 1;
    PA14 = 1;
}

void Display_7seg(uint16_t value)
{
    uint8_t digit;

    digit = value / 1000;
    CloseSevenSegment();
    ShowSevenSegment(3, digit);
    CLK_SysTickDelay(200);

    value %= 1000;
    digit = value / 100;
    CloseSevenSegment();
    ShowSevenSegment(2, digit);
    CLK_SysTickDelay(200);

    value %= 100;
    digit = value / 10;
    CloseSevenSegment();
    ShowSevenSegment(1, digit);
    CLK_SysTickDelay(200);

    digit = value % 10;
    CloseSevenSegment();
    ShowSevenSegment(0, digit);
    CLK_SysTickDelay(200);
}

enum State { GREEN, YELLOW, RED };
```

```c
struct TrafficSignal {
    enum State state;
    int greenDuration, yellowDuration, redDuration;
    int timer;
};

void TrafficSignal_initialize(struct TrafficSignal *ts, int g, int y, int r)
{
    ts->state = GREEN;
    ts->greenDuration = g;
    ts->yellowDuration = y;
    ts->redDuration = r;
    ts->timer = g;
}

void TrafficSignal_countDown(struct TrafficSignal *ts)
{
    ts->timer--;
    if (ts->timer <= 0)
    {
        switch (ts->state)
        {
        case GREEN:
            ts->state = YELLOW;
            ts->timer = ts->yellowDuration;
            break;
        case YELLOW:
            ts->state = RED;
            ts->timer = ts->redDuration;
            break;
        case RED:
        default:
            ts->state = GREEN;
            ts->timer = ts->greenDuration;
            break;
        }
    }
}
```

```
86  void Show_LED(enum State s)
87  {
88      switch (s)
89      {
90      case GREEN:
91          PA12 = 1;
92          PA13 = 0;
93          PA14 = 1;
94          break;
95      case YELLOW:
96          PA12 = 1;
97          PA13 = 0;
98          PA14 = 0;
99          break;
100     case RED:
101         PA12 = 1;
102         PA13 = 1;
103         PA14 = 0;
104         break;
105     }
106 }
```

```c
int main(void)
{
    struct TrafficSignal ts;
    int key = 0, last_key = 0;
    int tick = 0;

    SYS_Init();
    Init_GPIO();
    OpenSevenSegment();
    OpenKeyPad();

    TrafficSignal_initialize(&ts, 8, 5, 13);

    while (1)
    {
        key = ScanKey();
        if (key != 0 && key != last_key)
        {
            last_key = key;
            switch (key)
            {
            case 1:
                ts.state = GREEN;
                ts.timer = ts.greenDuration;
                break;
            case 2:
                ts.state = YELLOW;
                ts.timer = ts.yellowDuration;
                break;
            case 3:
                ts.state = RED;
                ts.timer = ts.redDuration;
                break;
            case 9:
                ts.timer += 5;
                if (ts.timer > 99) ts.timer = 99;
                break;
            default:
                break;
            }
        }
        else if (key == 0)
        {
            last_key = 0;
        }

        Display_7seg(ts.timer);
        Show_LED(ts.state);

        tick++;
        if (tick >= 900)
        {
            TrafficSignal_countDown(&ts);
            tick = 0;
        }
    }
}
```