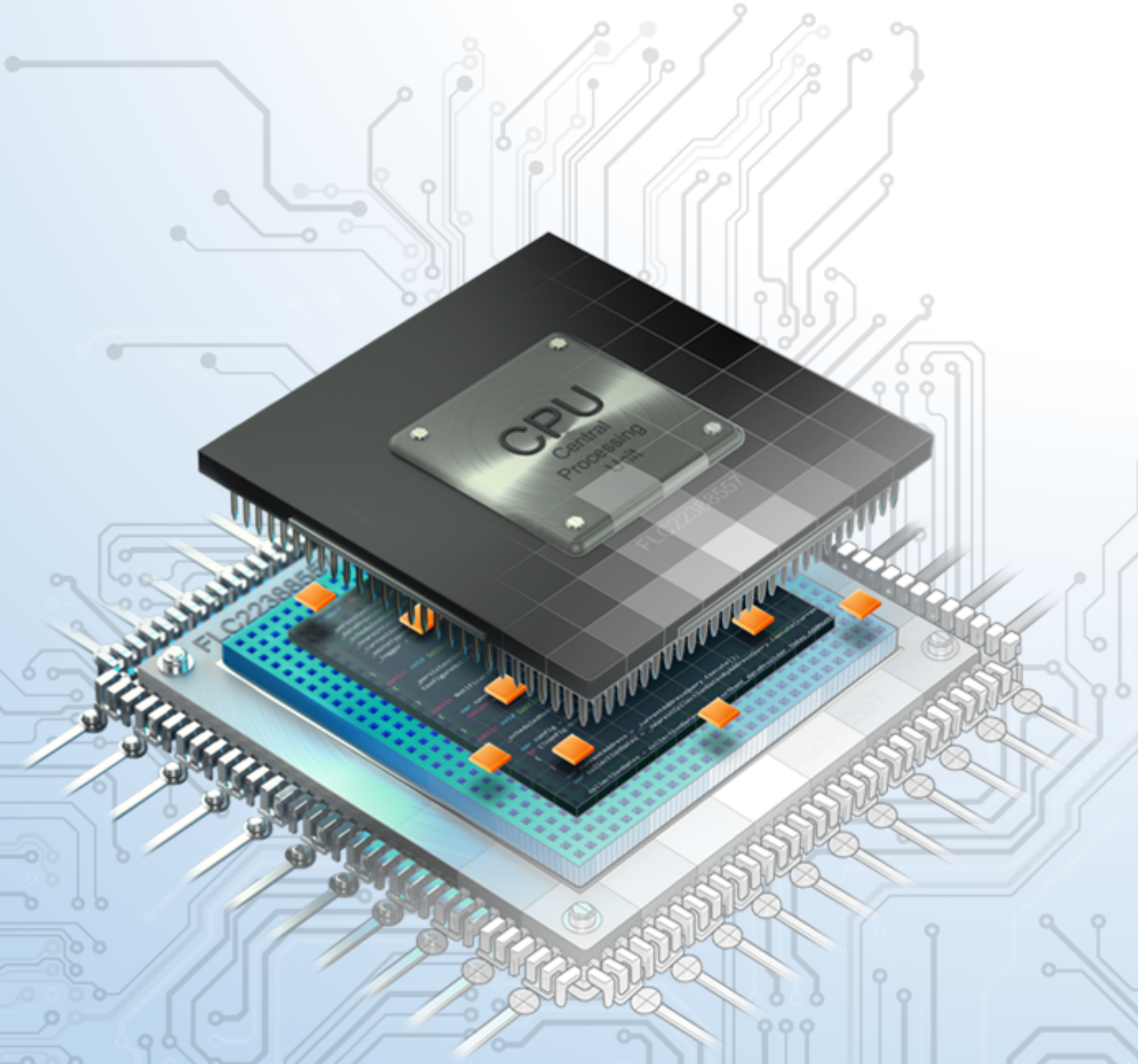
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

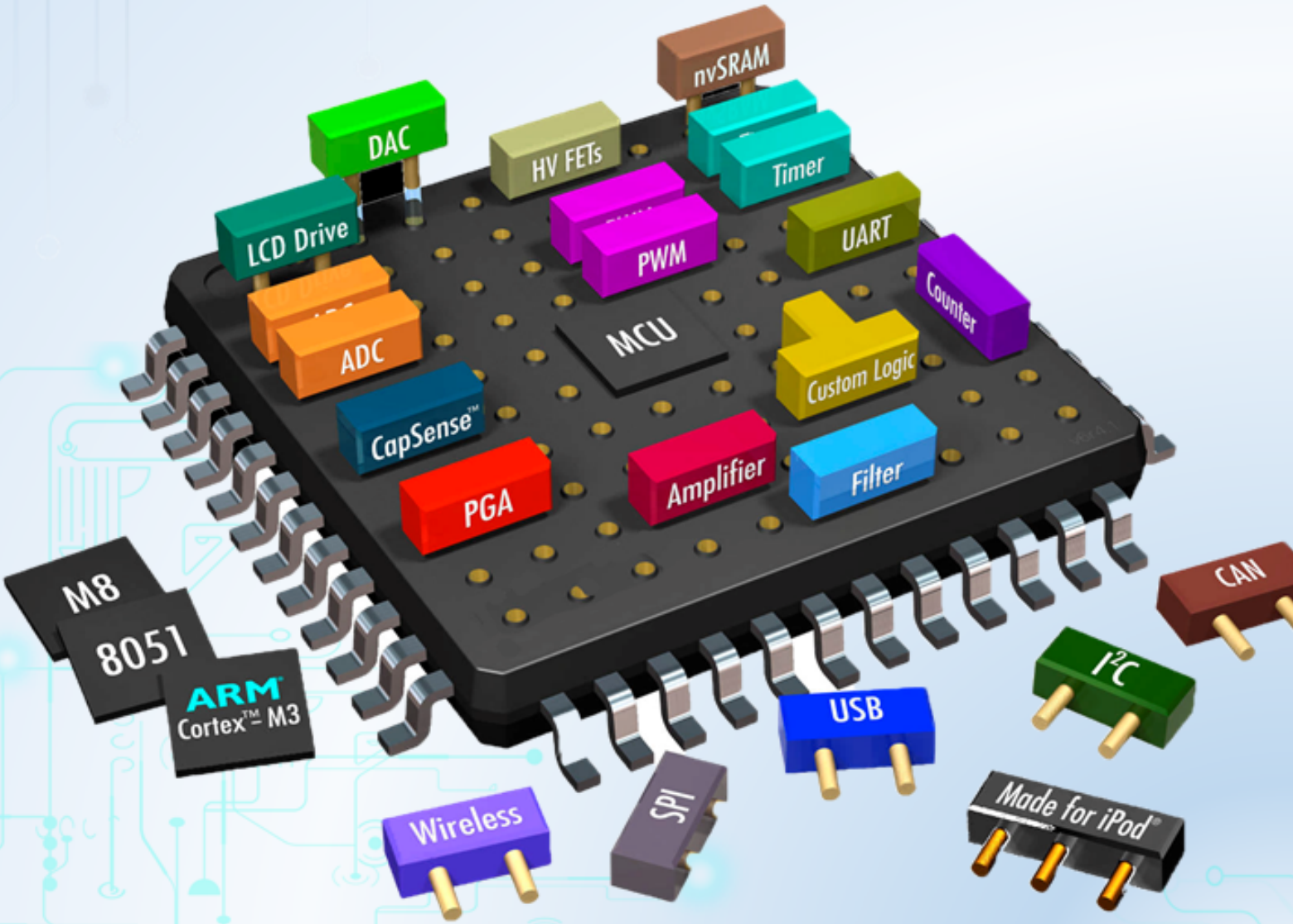


Dr. Le Trong Nhan

Mục lục

CHƯƠNG 1

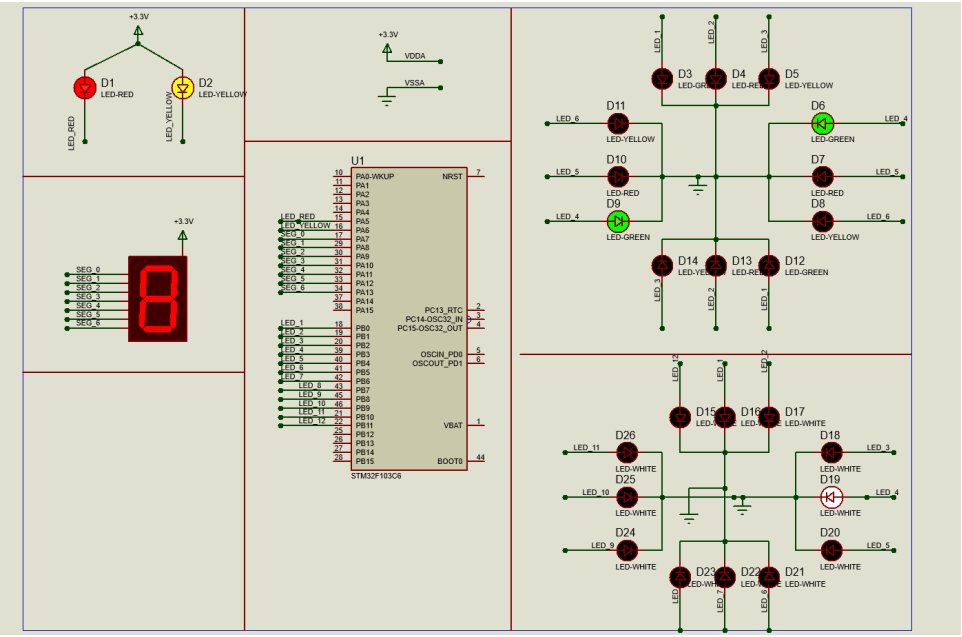
LED Animations



Program 1.1: First blinky LED project

1 Exercise and Report

I use one schematic for all Exercises below:



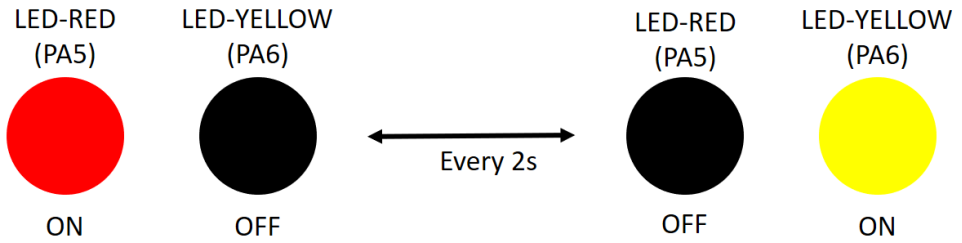
Hình 1.1: Schematic for all Exercises in lab 1

1.1 Exercise 1

From the simulation on Proteus, one more LED is connected to pin **PA6** of the STM32 (negative pin of the LED is connected to PA6). The component suggested in this exercise is **LED-YELLOW**, which can be found from the device list.

In this exercise, the status of two LEDs are switched every 2 seconds, as demonstrated in the figure bellow.

link Proteus Project: click here



Hình 1.2: *State transitions for 2 LEDs*

```
1    void create_state(){
2  HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, SET);
```

```
  HAL_GPIO_WritePin(LED_YELLOW_GPIO_Port, LED_YELLOW_Pin,
   RESET);
}
void q1_run(){
  if(count_q1>=2){
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port,LED_RED_Pin );
    HAL_GPIO_TogglePin(LED_YELLOW_GPIO_Port, LED_YELLOW_Pin
   );
  }
  count_q1++;
}
```

Program 1.2: This is supported code in Q1.h

```
void create_state_Q2(){
  LED_GREEN_off();
  LED_YELLOW_off();
  LED_RED_on();
  state=RED_ON;
}
```

```
create_state();
while (1)
{
q1_run}();
HAL_Delay(2000);
}
```

Program 1.3: This code in main (ex1)

1.2 Exercise 2

Extend the first exercise to simulate the behavior of a traffic light. A third LED, named **LED-GREEN** is added to the system, which is connected to **PA7**. A cycle in this traffic light is 5 seconds for the RED, 2 seconds for the YELLOW and 3 seconds for the GREEN. The LED-GREEN is also controlled by its negative pin.

```
#define RED_ON 0
#define YELLOW_ON 1
#define GREEN_ON 2
#define RED_TIME 5
#define YELLOW_TIME 2
#define GREEN_TIME 3
int state=0;
int count_q2=0;
```

Program 1.4: This is used for defining some variablesvariables in Q2.h

```
void create_state_Q2(){
  LED_GREEN_off();
  LED_YELLOW_off();
  LED_RED_on();
  state=RED_ON;
}
```

Program 1.5: Create$_s tate for EX2$

```

void Q2_run(){
  switch (state) {
    case RED_ON:
      if(count_q2<=0){
        LED_RED_off();
        LED_GREEN_on();
      }
      break;

    case YELLOW_ON:
      if(count_q2<=0){
        LED_YELLOW_off();
        LED_RED_on();
      }
      break;
    case GREEN_ON:
      if(count_q2<=0){
        LED_GREEN_off();
        LED_YELLOW_on();
      }
      break;
    default:
      break;
```

```
  }
  count_q2--;
}
```

Program 1.6: The code is used for describing the activation of leds

```
create_state_Q2();
while (1)
{
Q2_run();
HAL_Delay(1000);
}
```

Program 1.7: This code in main (ex2)

1.3 Exercise 3

Extend to the 4-way traffic light. Arrange 12 LEDs in a nice shape to simulate the behaviors of a traffic light. A reference design can be found in the figure bellow.

Such as the Exercise 2, the Exercise 3 has two lanes.

```
#define RED_ON 0
#define YELLOW_ON 1
#define GREEN_ON 2
#define RED_TIME 5
#define YELLOW_TIME 2
#define GREEN_TIME 3

int state1=0;
int count_q3_1=0;
int state2=0;
int count_q3_2=0;
```

Program 1.8: This is used for defining some variables in Q3.h

```
void create_state_q3(){
  LED_RED_1_on();
  LED_GREEN_1_off();
  LED_YELLOW_1_off();

  LED_RED_2_off();
  LED_GREEN_2_on();
  LED_YELLOW_2_off();
}
```

Program 1.9: Create state for EX3

```
void lane1(){
  count_q3_1--;
  switch (state1) {
    case RED_ON:
      if(count_q3_1<=0){
        LED_RED_1_off();
        LED_GREEN_1_on();
      }
      break;
    case YELLOW_ON:
      if(count_q3_1<=0){
        LED_YELLOW_1_off();
        LED_RED_1_on();
      }
      break;
    case GREEN_ON:
      if(count_q3_1<=0){
        LED_GREEN_1_off();
        LED_YELLOW_1_on();
```

```
      }
      break;
    default:
      break;
  }
}
void lane2(){
  count_q3_2--;
  switch (state2) {
    case RED_ON:
      if(count_q3_2<=0){
        LED_RED_2_off();
        LED_GREEN_2_on();
      }
      break;
    case YELLOW_ON:
      if(count_q3_2<=0){
        LED_YELLOW_2_off();
        LED_RED_2_on();
      }
      break;
    case GREEN_ON:
      if(count_q3_2<=0){
        LED_GREEN_2_off();
        LED_YELLOW_2_on();
      }
      break;
    default:
      break;
  }
}
void q3_run(){

  lane1();
  lane2();
}
}
```

Program 1.10: The code is used for describing the activation of leds

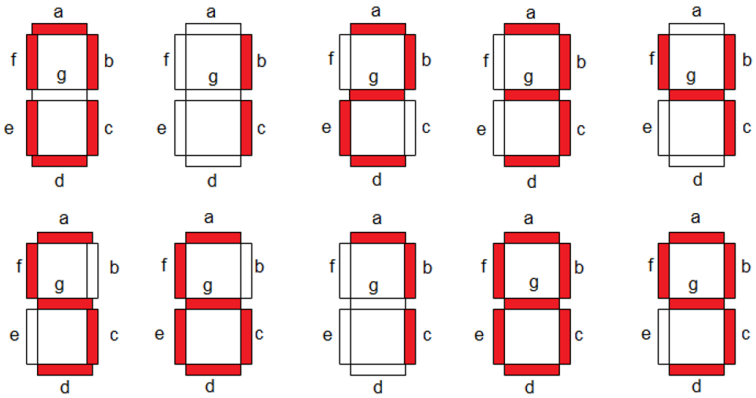
```
      create_state_q3();
  while (1)
  {
    q3_run();
    HAL_Delay(1000);
  }
```

Program 1.11: code in while loop

1.4 Exercise 4

Add **only one 7 led segment** to the schematic in Exercise 3. This component can be found in Proteus by the keyword **7SEG-COM-ANODE**. For this device, the common pin should be connected to the power supply and other pins are supposed to connected to PB0 to PB6. Therefore, to turn-on a segment in this 7SEG, the STM32 pin should be in logic 0 (0V).

Implement a function named **display7SEG(int num)**. The input for this function is from 0 to 9 and the outputs are listed as following:



Hình 1.3: Display a number on 7 segment LED

This function is invoked in the while loop for testing as following:

```
int counter = 0;
while (1){
    if(counter >= 10) counter = 0;
    display7SEG(counter++);
    HAL_Delay(1000);

}
```

Program 1.12: An example for your source code

```
    const int16_t array_seg_pin[]={
    SEG_0_Pin,
    SEG_1_Pin,
    SEG_2_Pin,
    SEG_3_Pin,
    SEG_4_Pin,
    SEG_5_Pin,
    SEG_6_Pin,
};


void display7SEG ( int number){
  switch (number) {
    case 0:
```

```
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], SET);
      break;
    case 1:
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], SET);
      break;
    case 2:
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
      break;
    case 3:
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
      break;
    case 4:
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
      break;
    case 5:
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], SET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
      HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
```

```
64        HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
65        HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
66        HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
67        break;
68      case 6:
69        HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
70        HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], SET);
71        HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
72        HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
73        HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], RESET);
74        HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
75        HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
76        break;
77      case 7:
78        HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
79        HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
80        HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
81        HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], SET);
82        HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
83        HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], SET);
84        HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], SET);
85        break;
86      case 8:
87        HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
88        HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
89        HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
90        HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
91        HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], RESET);
92        HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
93        HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
94        break;
95      case 9:
96        HAL_GPIO_WritePin(GPIOA, array_seg_pin[0], RESET);
97        HAL_GPIO_WritePin(GPIOA, array_seg_pin[1], RESET);
98        HAL_GPIO_WritePin(GPIOA, array_seg_pin[2], RESET);
99        HAL_GPIO_WritePin(GPIOA, array_seg_pin[3], RESET);
100       HAL_GPIO_WritePin(GPIOA, array_seg_pin[4], SET);
101       HAL_GPIO_WritePin(GPIOA, array_seg_pin[5], RESET);
102       HAL_GPIO_WritePin(GPIOA, array_seg_pin[6], RESET);
103       break;
104
105     default:
106       break;
107   }
108 }
```

1.5 Exercise 5

Integrate the 7SEG-LED to the 4 way traffic light. In this case, the 7SEG-LED is used to display countdown value.

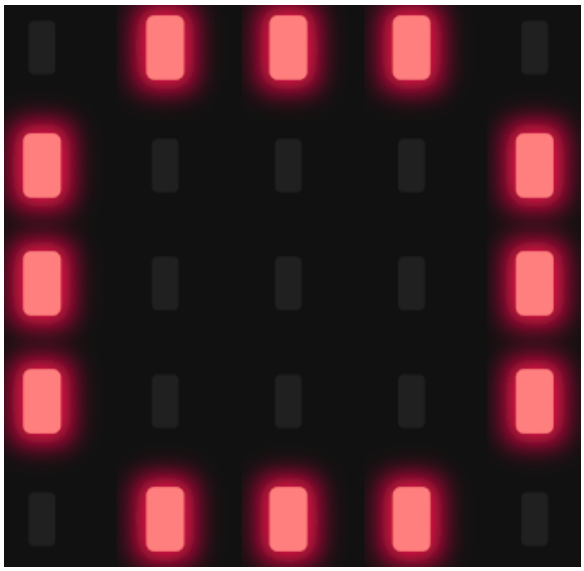
In this exercise, only source code is required to present. The function display7SEG in previous exercise can be re-used.

I use code from EX3 and EX4 to make EX5. This is my while loop in main.

```
create_state_q3();
while (1)
  {
    display7SEG(count_q3_1); // display the 7segment-led
    HAL_Delay(time_delay);
    q3_run(); // ex5= ex3+ex4
}
```

1.6 Exercise 6

In this exercise, a new Proteus schematic is designed to simulate an analog clock, with 12 different number. The connections for 12 LEDs are supposed from PA4 to PA15 of the STM32. The arrangement of 12 LEDs is depicted as follows.



Hình 1.4: 12 LEDs for an analog clock

```
    void test_sequence(){
  for(int i=0; i<12; i++){
    HAL_GPIO_WritePin(GPIOB,arr_led_pin[i], SET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(GPIOB, arr_led_pin[i], RESET);
  }
}
```

Program 1.13: the test led function

```
    while(1){
    test_sequence();
    }
```

1.7 Exercise 7

Implement a function named **clearAllClock()** to turn off all 12 LEDs. Present the source code of this function.

```
void clearAllClock(){
  for(int i=0; i<12; i++){
    HAL_GPIO_WritePin(GPIOB,arr_led_pin[i], RESET);
  }
```

```
}
```

Program 1.14: Implementation of function clearAllClock()

1.8 Exercise 8

Implement a function named **setNumberOnClock(int num)**. The input for this function is from **0 to 11** and an appropriate LED is turn on. Present the source code of this function.

```
void setNumberOnClock(int num){
  if(num<0|| num>11) return;
  HAL_GPIO_WritePin(GPIOB, arr_led_pin[num], SET);
}
```

Program 1.15: Implementation of function setNumberOnClock

1.9 Exercise 9

Implement a function named **clearNumberOnClock(int num)**. The input for this function is from **0 to 11** and an appropriate LED is turn off.

```
    void clearNumberOnClock(int num){
  if(num<0|| num>11) return;
  HAL_GPIO_WritePin(GPIOB, arr_led_pin[num], RESET);
}
```

1.10 Exercise 10

Integrate the whole system and use 12 LEDs to display a clock. At a given time, there are only 3 LEDs are turn on for hour, minute and second information.

```
int second=0;
int minute=0;
int hour=0;

int second_on_screen=0;
int minute_on_screen=0;
int hour_on_screen=0;
```

Program 1.16: define variables for ex10

```
void create_state(){
  second_on_screen=second/5;
  minute_on_screen=minute/5;
  hour_on_screen=hour;
  setNumberOnClock(second_on_screen);
  setNumberOnClock(minute_on_screen);
  setNumberOnClock(hour_on_screen);
```

```
}
```

Program 1.17: create the first state for ex10

```
    void update_clock(){
  if(floor(second/5) != second_on_screen ){
    clearNumberOnClock(second_on_screen);
    second_on_screen=second/5;
  }
  if(floor(minute/5) != minute_on_screen){
    clearNumberOnClock(minute_on_screen);
    minute_on_screen=minute/5;
  }
  if(hour!=hour_on_screen){
    clearNumberOnClock(hour_on_screen);
    hour_on_screen=hour;
  }
  setNumberOnClock(second_on_screen);
  setNumberOnClock(minute_on_screen);
  setNumberOnClock(hour_on_screen);


}
void ex10(){
  second++;
  if(second >=60){

    second=0;
    minute++;
  }
  if(minute>=60){
    minute=0;
    hour++;
  }
  if(hour>=12){
    hour=0;
  }
  update_clock();
}
```

Program 1.18: update clock and how ex10 runs

```
#define time_delay 1000
create_state();
  while (1)
  {
    //test_sequence();
   ex10();
   HAL_Delay(time_delay);
    /* USER CODE END WHILE */

```

```
10      /* USER CODE BEGIN 3 */
11    }
```

Program 1.19: While in main

If we want to see the behavior of the clock, we just change the time_delay from 1000 to less number such as 50ms or 100ms.

github: STM32_LAB1_Tang_Phon_Thinh