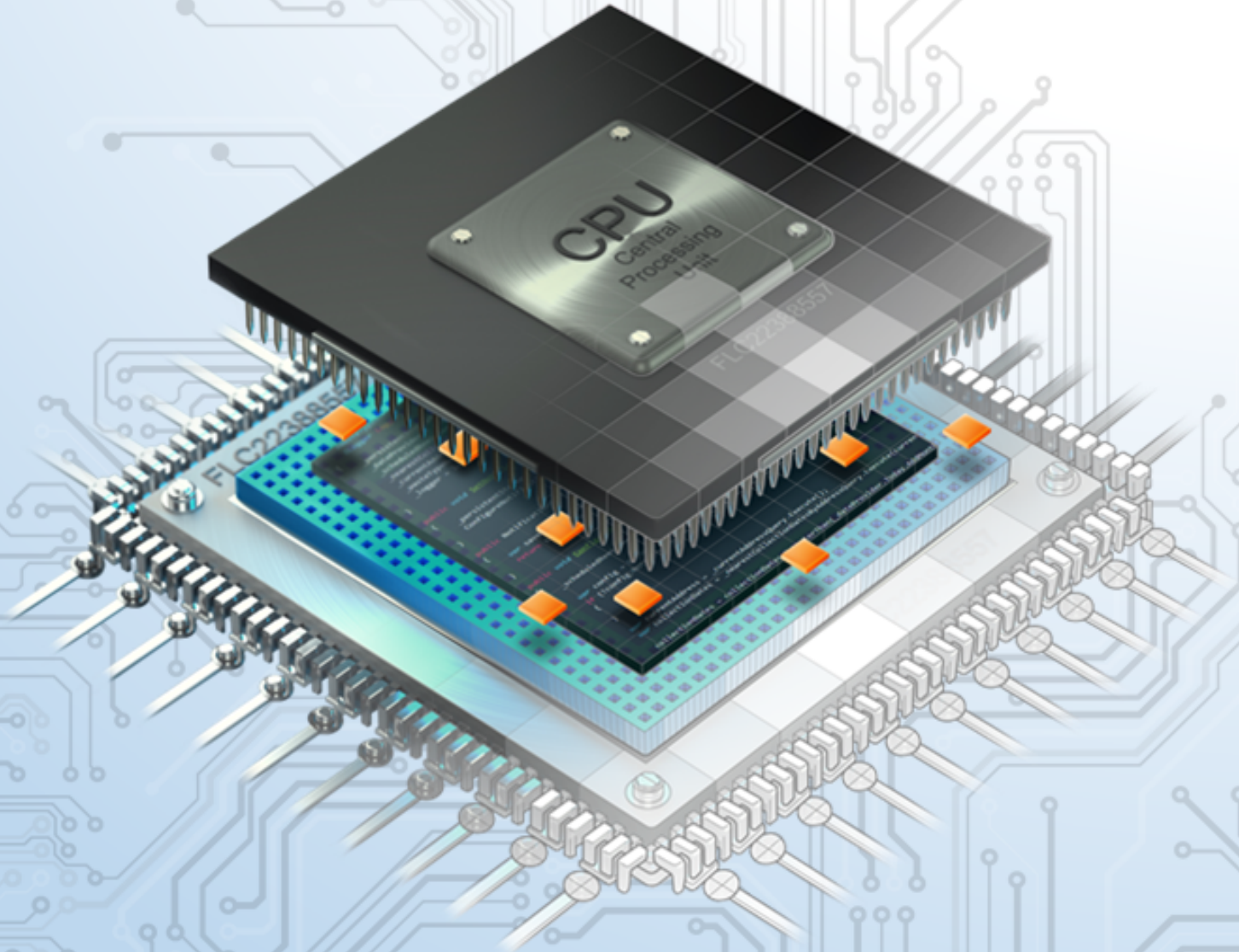
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

Sinh viên thực hiện: Nguyễn Thanh Hiền (MSSV: 2111203)
Lớp: L03 - HK231

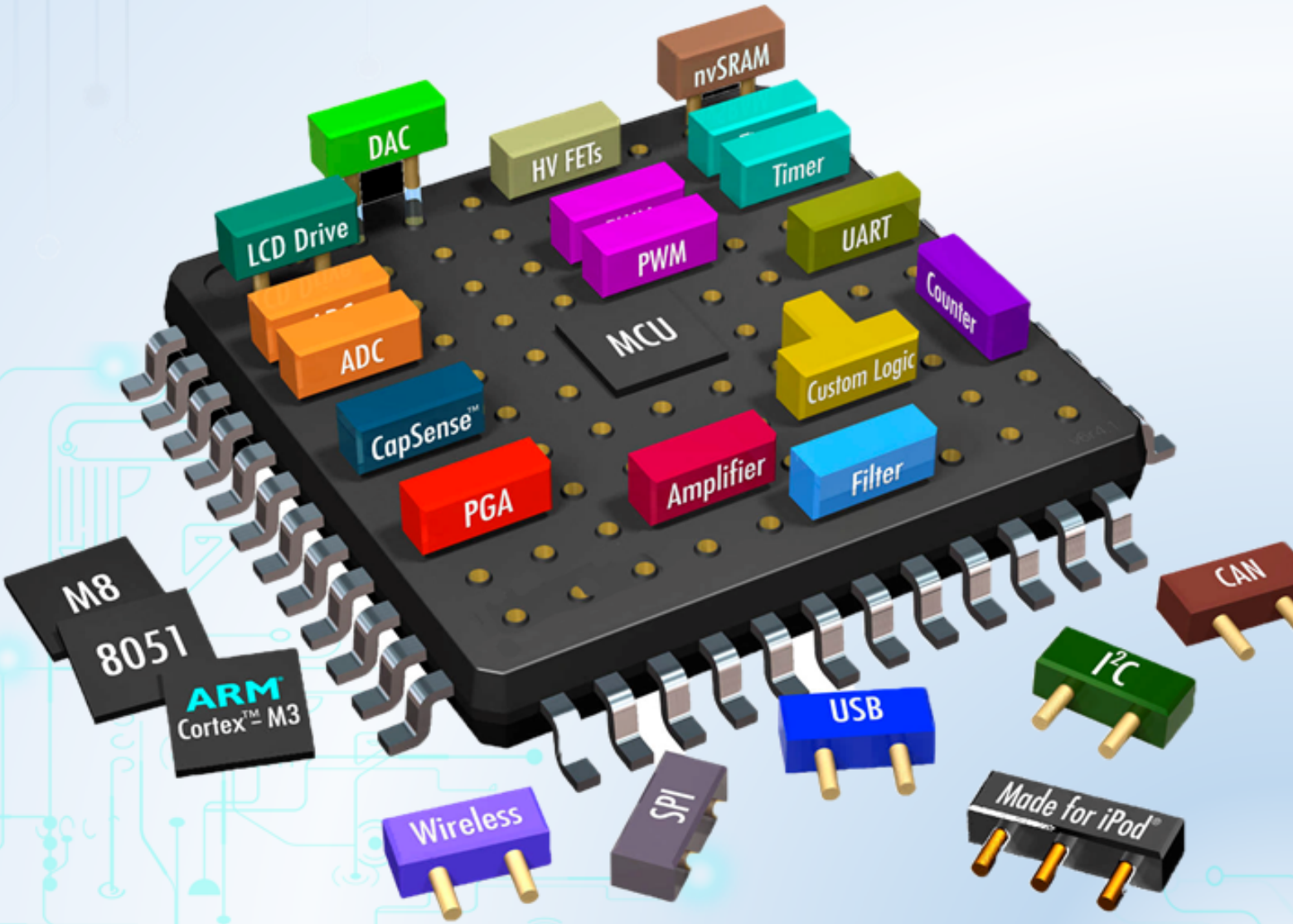


Dr. Le Trong Nhan

Mục lục

CHƯƠNG 1

LED Animations

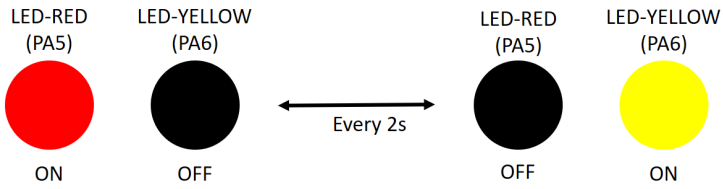


Github: https://github.com/amyranotamirror/CO3009-Lab1

1 Exercise 1

1.1 Yêu cầu

Đèn LED đỏ và vàng luân phiên nhau chớp tắt sau mỗi 2s như hình bên dưới.



Hình 1.1: State transitions for 2 LEDs

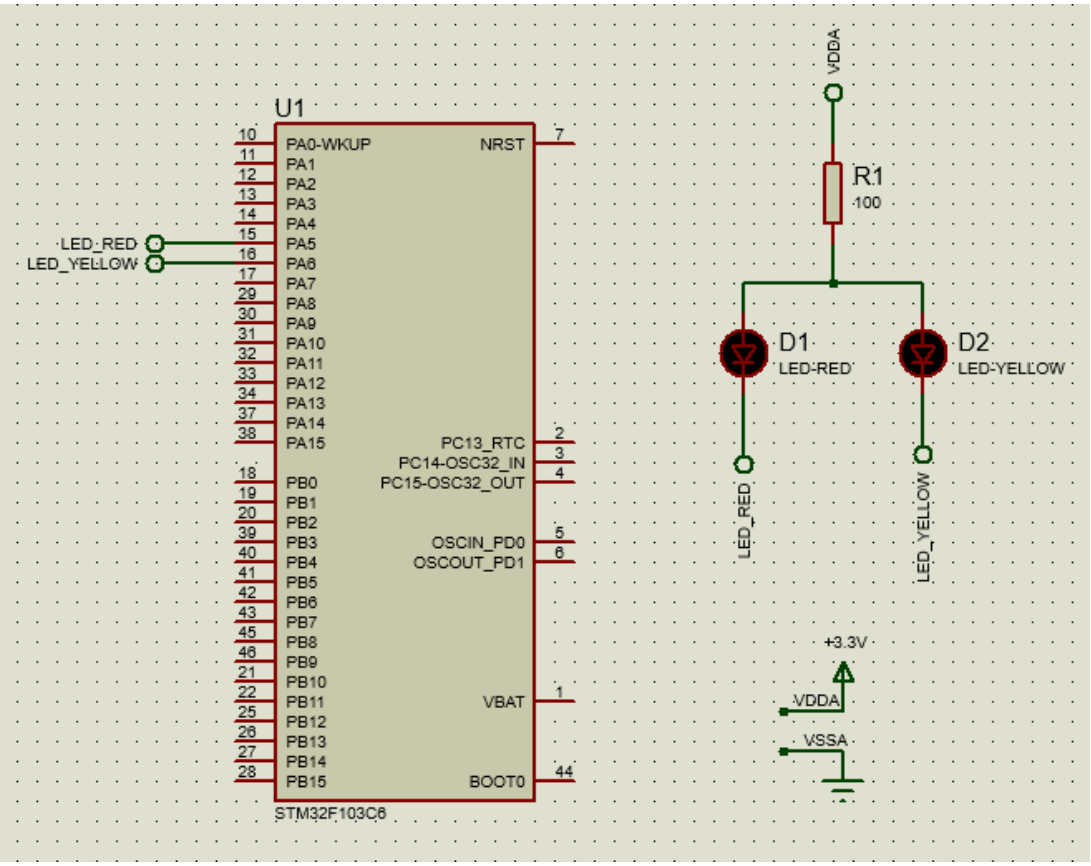
1.2 Bài làm

Kết nối pin **PA5**, **PA6** của STM32 lần lượt tới đầu cathode của đèn LED đỏ và vàng. Ta sử dụng tín hiệu tích cực mức **thấp** để bật đèn, ngược lại, sử dụng tín hiệu tích cực mức **cao** để tắt đèn.

```
HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin,
   GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED_YELLOW_GPIO_Port, LED_YELLOW_Pin,
   GPIO_PIN_SET);
int counter = 0;

while (1){
    if(counter <= 2){
          counter = 0;
          HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
          HAL_GPIO_TogglePin(LED_YELLOW_GPIO_Port,
   LED_YELLOW_Pin);
      }
      counter++;
      HAL_Delay(1000);
}
```

Program 1.1: Đèn LED luân phiên nhấp nháy mỗi 2s



Hình 1.2: Sơ đồ nguyên lý mô phỏng đèn LED luân phiên nhấp nháy

2 Exercise 2

2.1 Yêu cầu

Mở rộng thêm từ bài thứ nhất để mô phỏng hành vi của đèn giao thông, một đèn LED thứ ba màu xanh được thêm vào. Thời lượng đèn đỏ là 5s, đèn vàng là 2s và đèn xanh là 3s. Trình bày schematic và source code.

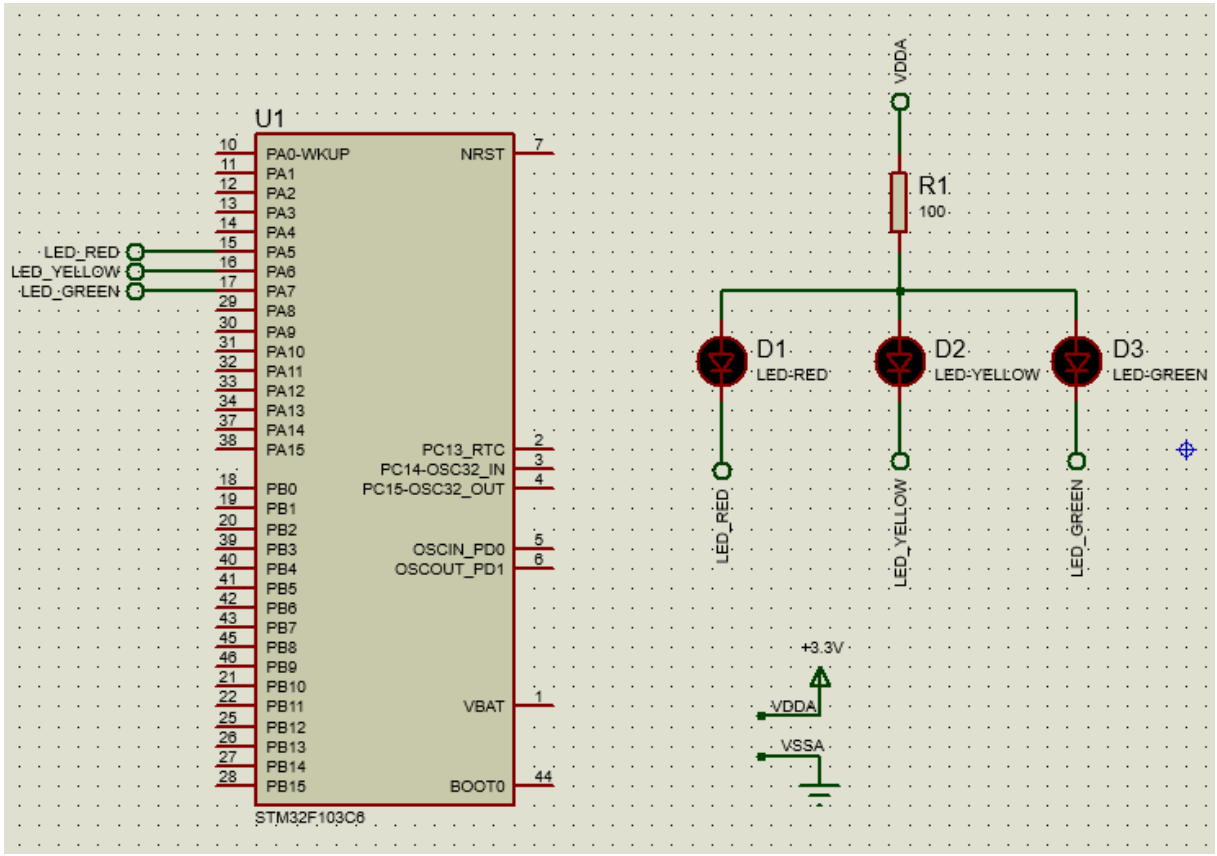
2.2 Bài làm

Kết nối pin **PA7** của STM21 tới đàu cathode của đèn LED xanh, tương tự, sử dụng sử dụng tín hiệu tích cực mức **thấp** để bật đèn.

Đầu tiên, thiết lập các hàm để ghi tín hiệu mức thấp/cao lên các pin. Thông qua đó, ta chỉ cần cài đặt tên port, pin của các đèn 1 lần và sử dụng hàm để phát tín hiệu phù hợp.

```
#define NUM_COLOR 3

typedef struct{
    GPIO_TypeDef* port;
    uint16_t pin;
```



Hình 1.3: Sơ đồ nguyên lý mô phỏng đèn giao thông 1 phía

```
} GPIO_config;
GPIO_config light_color[NUM_COLOR];
void set_GPIO_off(GPIO_config *destination);
void set_GPIO_on(GPIO_config *destination);
```

Program 1.2: Hàm hỗ trợ ghi tín hiệu trong file global.h

Tiếp theo, bổ sung các hàm hỗ trợ điều khiển các đèn LED theo cơ chế của đèn giao thông:

- `void init_traffic()`: Thiết lập trạng thái ban đầu của đèn giao thông với các port, pin tương ứng.
- `void turn_light_on(int color)`, `void turn_light_off(int color)`: Điều khiển bật/tắt 1 đèn màu `color`.
- `void turn_all_light_off()`: Tắt hết 3 đèn, sử dụng cho trường hợp khởi tạo hoặc lỗi.
- `void run_traffic_light)(int color)`: Bật duy nhất 1 đèn tương ứng với màu `color` và tắt 2 đèn còn lại.

```
#define RED_LIGHT 0
#define YELLOW_LIGHT 1
#define GREEN_LIGHT 2

```

```c
#define GREEN_TIME 3
#define YELLOW_TIME 2
#define RED_TIME (GREEN_TIME + YELLOW_TIME)

void init_traffic();
void turn_light_off(int color);
void turn_light_on(int color);
void turn_all_light_off();
void run_traffic_light(int color);
```

Program 1.3: Hàm điều khiển đèn LEDs trong file traffic_light.h

Cuối cùng, ta khởi tạo trạng thái ban đầu là đèn màu **ĐỎ** và luân phiên chớp tắt đèn LEDs theo cơ chế đèn giao thông: đèn đỏ 5s, đèn xanh 3s và đèn vàng 2s.

```c
init_traffic();
int traffic_color = RED_LIGHT;
int traffic_timer = RED_TIME - 1;

while (1)
{
  run_traffic_light(traffic_color);
  switch(traffic_color){
  case RED_LIGHT:
      if(traffic_timer <= 0){
          traffic_color = GREEN_LIGHT;
          traffic_timer = GREEN_TIME;
      }
      break;
  case GREEN_LIGHT:
      if(traffic_timer <= 0){
          traffic_color = YELLOW_LIGHT;
          traffic_timer = YELLOW_TIME;
      }
      break;
  case YELLOW_LIGHT:
      if(traffic_timer <= 0){
          traffic_color = RED_LIGHT;
          traffic_timer = RED_TIME;
      }
      break;
  default:
      turn_all_light_off();
      break;
  }
  traffic_timer--;
  HAL_Delay(1000);
}
```

Program 1.4: Vòng while của file main.c

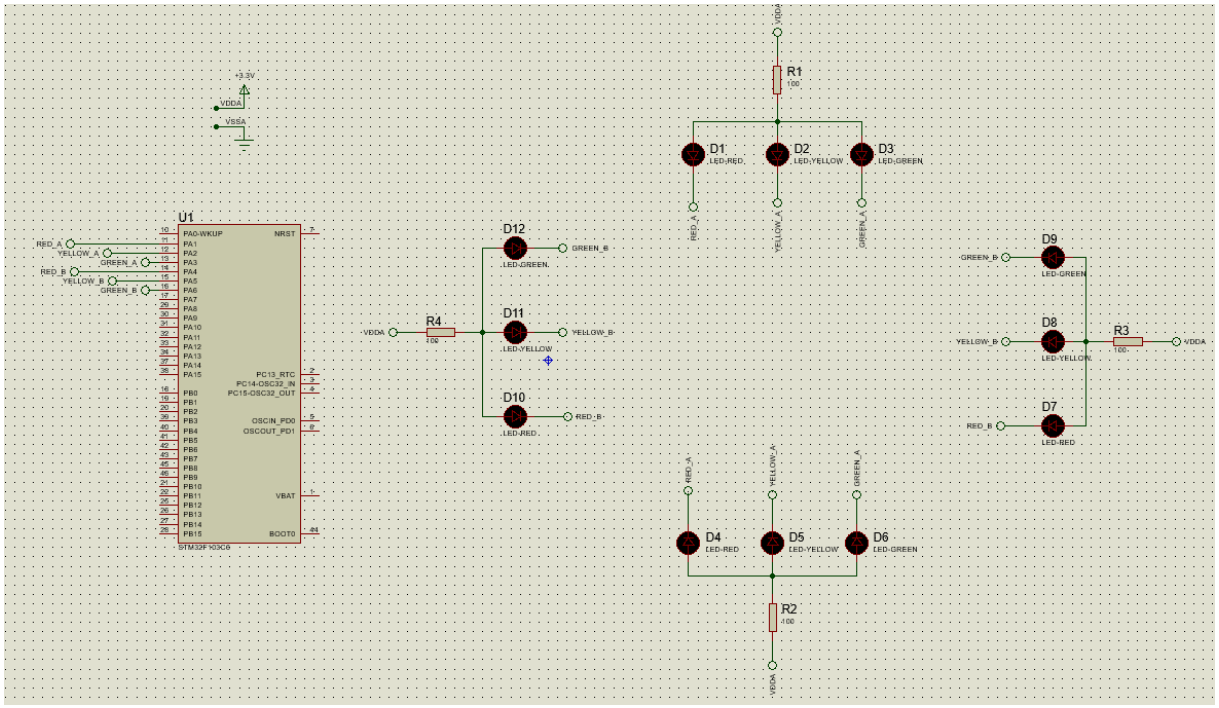
3 Exercise 3

3.1 Yêu cầu

Mở rộng trở thành đèn giao thông tại ngã tư.

3.2 Bài làm

Bổ sung 1 khối đèn giao thông ở 3 phía còn lại, trong đó, mỗi phía đối diện nhau sẽ cùng được điều khiển chung.



Hình 1.4: Sơ đồ nguyên lý đèn giao thông tại ngã tư

Đầu tiên, ta bổ sung thêm định nghĩa về các màu và đồng hồ đếm ngược tại mỗi phía.

```
#define NUM_SIDE 2
int traffic_color[NUM_SIDE];
int traffic_timer[NUM_SIDE];
```

Program 1.5: Phần được bổ sung trong file global.h

Tiếp theo, bổ sung và thay đổi hàm để chuyển trạng thái giữa các phía đèn giao thông:

- `void run_traffic_light(int side, int color)`: Đối với 1 phía cụ thể, hàm sẽ chỉ bật duy nhất 1 LED phù hợp, các LED còn lại sẽ tắt
- `void change_traffic_light(int side, int color)`: Thực hiện việc chuyển màu đèn tại 1 phía, đồng thời, thiết lập đồng hồ đếm ngược phù hợp.

```
#define SIDE_A 0
#define SIDE_B 1

void init_traffic_light();
void turn_light_off(int side, int color);
void turn_light_on(int side, int color);
void turn_all_light_off();
void run_traffic_light(int side, int color);
void change_traffic_light(int side, int color);
```

Program 1.6: Hàm điều khiển đèn LEDs trong file traffic_light.h

Cuối cùng, thay đổi code trong hàm điều khiển chỉnh để hiện thực cơ chế đèn đỏ tại ngã tư.

```
while (1)
{
  run_traffic_light(SIDE_A, traffic_color[SIDE_A]);
  run_traffic_light(SIDE_B, traffic_color[SIDE_B]);
  switch(traffic_color[SIDE_A]){
  case RED_LIGHT:
      if(traffic_timer[SIDE_A] <= 0 && traffic_timer[SIDE_B]
   <= 0){
          // SIDE_A: turn to green, SIDE_B: turn to red
          change_traffic_light(SIDE_A, GREEN_LIGHT);
          change_traffic_light(SIDE_B, RED_LIGHT);
          break;
      }
      if(traffic_timer[SIDE_B] <= 0){
          // SIDE_A: remain red, SIDE_B: turn yellow
          change_traffic_light(SIDE_B, YELLOW_LIGHT);
          break;
      }
      break;
  case GREEN_LIGHT:
      if(traffic_timer[SIDE_A] <= 0){
          // SIDE_A: turn yellow, SIDE_B: remain red
          change_traffic_light(SIDE_A, YELLOW_LIGHT);
          break;
      }
      break;
  case YELLOW_LIGHT:
      if(traffic_timer[SIDE_A] <= 0 && traffic_timer[SIDE_B]
   <= 0){
          // SIDE_A: turn to red, SIDE_B: turn to green
          change_traffic_light(SIDE_A, RED_LIGHT);
          change_traffic_light(SIDE_B, GREEN_LIGHT);
          break;
      }
      break;
  default:
```

```
35          turn_all_light_off();
36          break;
37      }
38      traffic_timer[SIDE_A]--;
39      traffic_timer[SIDE_B]--;
40      HAL_Delay(1000);
41
42  }
```

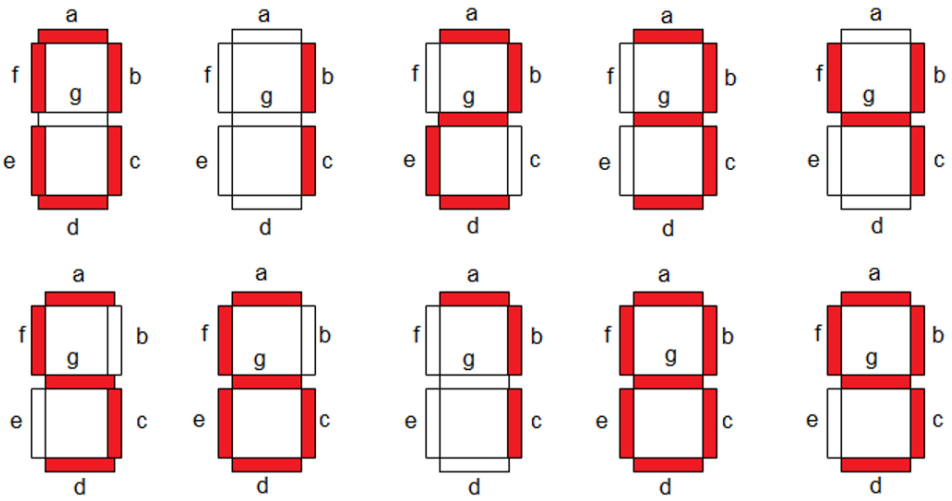
Program 1.7: Nội dung vòng while của file main.h

4 Exercise 4

4.1 Yêu cầu

Thêm **duy nhất 1 LED 7 đoạn** dương chung vào sơ đồ nguyên lý exercise 3. Với loại LED này, chân dương chung cần được kết nối lên nguồn và các chân còn lại được sử dụng để kết nối với các chân từ **PB0** đến **PB6**. Vì vậy, để bật đèn LED thì pin của STM32 phải tích cực tại mức logic 0.

Hiện thực hàm `display7SEG(int num)` để mô phỏng hiển thị số từ 0 đến 9 trên đèn LED 7 đoạn.

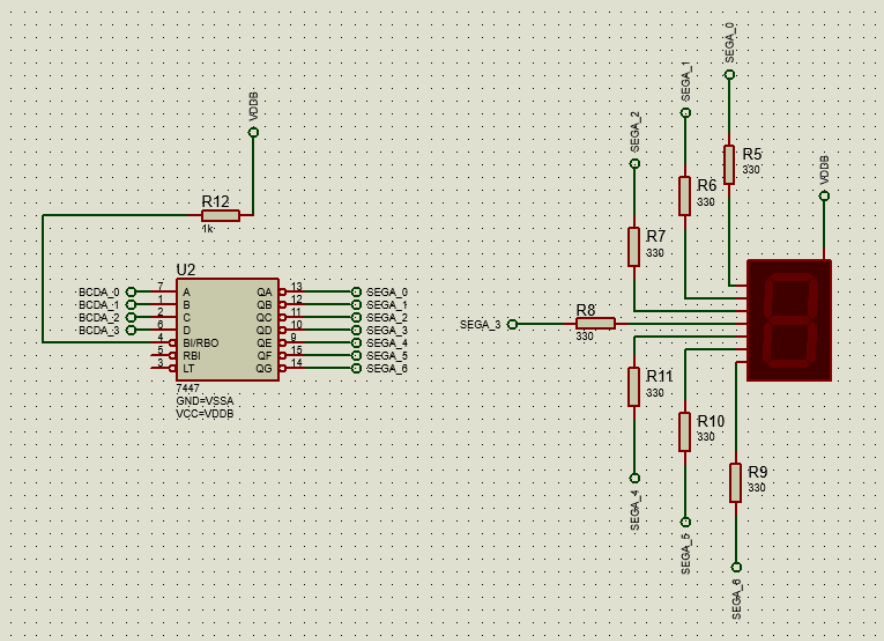


Hình 1.5: Display a number on 7 segment LED

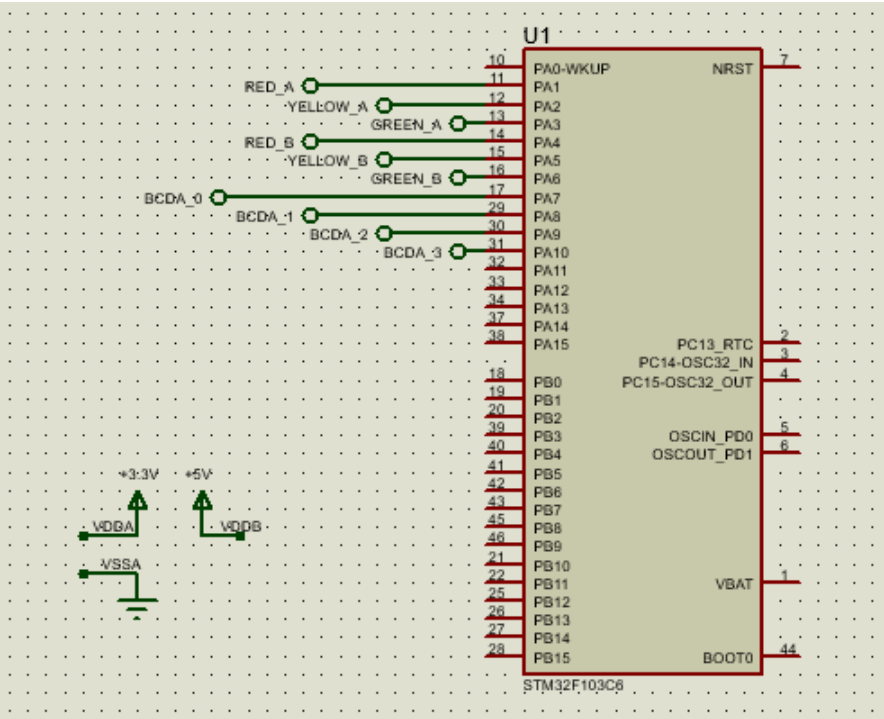
4.2 Bài làm

Cách tiếp cận thứ nhất

Thông qua IC7447 sử dụng đầu vào là số BCD từ STM32 để điều khiển 7 chân dữ liệu của đèn LED. Tuy nhiên, IC này cũng sẽ đảo tín hiệu đầu ra của các chân nối với LED, nên ta sẽ sử dụng tín hiệu mức cao để điều khiển đèn LED (tương ứng với các số BCD).



Hình 1.6: Sơ đồ nguyên lý kết nối LED 7 đoạn với IC7447



Hình 1.7: Sơ đồ nguyên lý kết nối các chân của STM32

Đầu tiên, ta sẽ bổ sung các hàm hỗ trợ hiển thị LED 7 đoạn các số được điều khiển bằng mã BCD qua các chân **PA7** đến **PA10**:

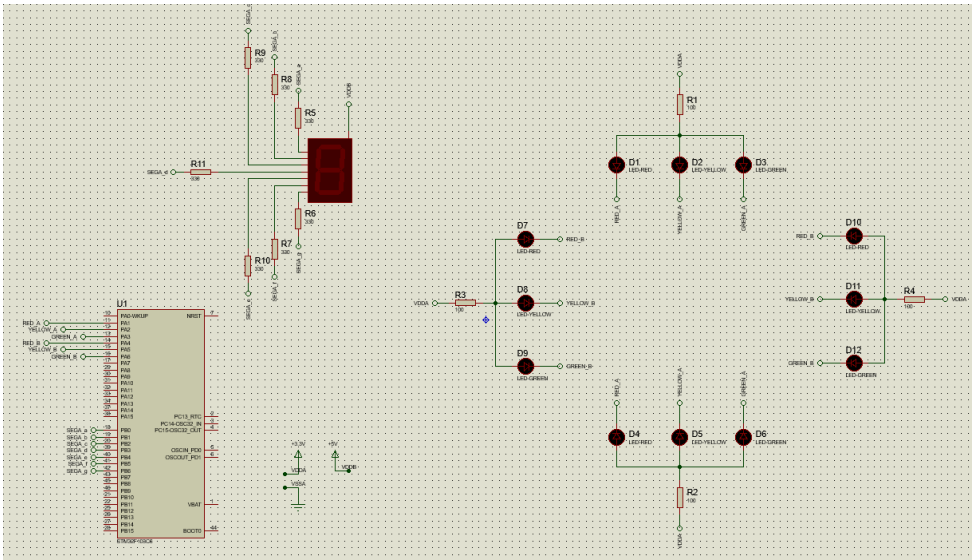
- `void init_7_seg()`: Thiết lập trạng thái ban đầu của LED 7 đoạn với các port, pin tương ứng.
- `void turn_seg_on(int BCD_position)`, `void turn_seg_off(int BCD_position)`: Điều khiển mức tín hiệu mức cao/thấp của mỗi bit BCD.
- `void turn_all_seg_off()`: Reset đèn LED 7 đoạn thành số 0 (BCD = 0000)
- `void display7SEG(int num)`: Điều khiển tín hiệu BCD của 4 chân tín hiệu, dùng làm đầu vào cho IC7447 để điều khiển LED 7 đoạn.

```
void init_7_seg();
void turn_seg_on(int BCD_position);
void turn_seg_off(int BCD_position);
void turn_all_seg_off();
void display7SEG(int num){
    if(num >= MIN_BCD && num <= MAX_BCD){
        for(int pos = 0; pos < NUM_BITS; pos++){
            if(num % 2 == 0) turn_seg_off(pos);
            if(num % 2 == 1) turn_seg_on(pos);
            num = num / 2;
        }
    }
}
```

Program 1.8: Hàm điều khiển LED 7 đoạn trong file led_seg.h và led_seg.c

Cách tiếp cận thứ hai

Hiện thực tương tự như đề bài yêu cầu, ta kết nối trực tiếp 7 chân của đèn LED 7 đoạn vào STM32.



Hình 1.8: Sơ đồ nguyên lý kết nối LED 7 đoạn trực tiếp tới STM32

Kết nối 7 chân của đèn LED với Port B, trực tiếp ghi tín hiệu vào port với các các pin được masked để đánh dấu trạng thái bật/tắt mỗi đoạn.

```
void display7SEG(int num){
    switch(num){
    case 0:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
  SEG_c_Pin|SEG_d_Pin|SEG_e_Pin|SEG_f_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_g_Pin, GPIO_PIN_SET);
        break;
    case 1:
        HAL_GPIO_WritePin(GPIOB, SEG_b_Pin|SEG_c_Pin,
  GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_d_Pin|
  SEG_e_Pin|SEG_f_Pin|SEG_g_Pin, GPIO_PIN_SET);
        break;
    case 2:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
  SEG_d_Pin|SEG_e_Pin|SEG_g_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_c_Pin|SEG_f_Pin,
  GPIO_PIN_SET);
        break;
    case 3:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
  SEG_c_Pin|SEG_d_Pin|SEG_g_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_e_Pin|SEG_f_Pin,
  GPIO_PIN_SET);
        break;
    case 4:
        HAL_GPIO_WritePin(GPIOB, SEG_b_Pin|SEG_c_Pin|
  SEG_f_Pin|SEG_g_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_d_Pin|
  SEG_e_Pin, GPIO_PIN_SET);
        break;
    case 5:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_c_Pin|
  SEG_d_Pin|SEG_f_Pin|SEG_g_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,SEG_b_Pin|SEG_e_Pin,
  GPIO_PIN_SET);
        break;
    case 6:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_c_Pin|
  SEG_d_Pin|SEG_e_Pin|SEG_f_Pin|SEG_g_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,SEG_b_Pin, GPIO_PIN_SET);
        break;
    case 7:
        HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
  SEG_c_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, SEG_d_Pin|SEG_e_Pin|
  SEG_f_Pin|SEG_g_Pin, GPIO_PIN_SET);
```

```
34          break;
35      case 8:
36          HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
    SEG_c_Pin|SEG_d_Pin|SEG_e_Pin|SEG_f_Pin|SEG_g_Pin,
    GPIO_PIN_RESET);
37          break;
38      case 9:
39          HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
    SEG_c_Pin|SEG_d_Pin|SEG_f_Pin|SEG_g_Pin, GPIO_PIN_RESET);
40          HAL_GPIO_WritePin(GPIOB, SEG_e_Pin, GPIO_PIN_SET);
41          break;
42      default:
43          HAL_GPIO_WritePin(GPIOB, SEG_a_Pin|SEG_b_Pin|
    SEG_c_Pin|SEG_d_Pin|SEG_e_Pin|SEG_f_Pin|SEG_g_Pin,
    GPIO_PIN_SET);
44          break;
45      }
46  }
```

Program 1.9: Hàm display7SEG(int num)

5 Exercise 5

5.1 Yêu cầu

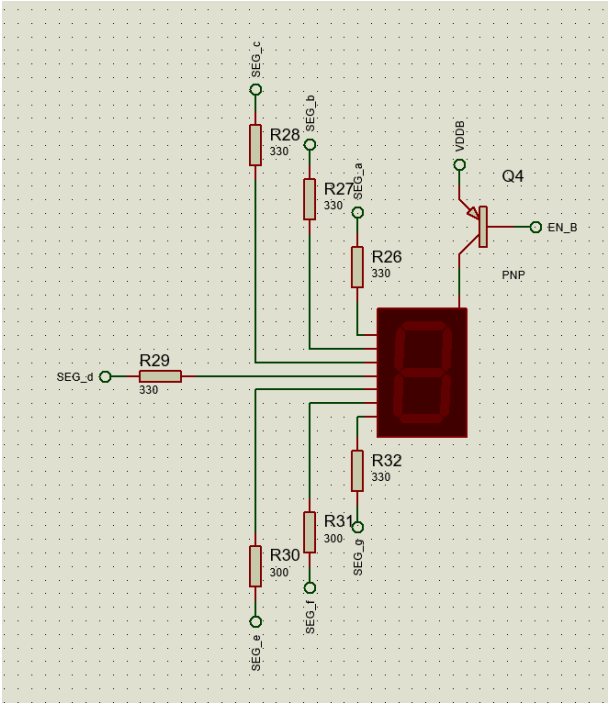
Tích hợp LED 7 đoạn vào hệ thống đèn giao thông. Trong trường hợp này, đèn LED phải hiển thị giá trị đếm ngược. Đối với bài này, sinh viên chỉ cần cung cấp source code

5.2 Bài làm

Trong bài lần này, em bổ sung thêm 2 chân `EN_A` và `EN_B` nhằm lựa chọn cặp LED nào sẽ hiển thị tại mỗi mốc thời gian nhất định (cơ chế quét LED đơn giản). Đồng thời, hàm điều khiển LED cũng cần bổ sung biến đầu vào để nhận biết cặp LED nào đang được hiển thị.

```
1 HAL_GPIO_WritePin(EN_A_GPIO_Port, EN_A_Pin, (GPIO_PinState)(
    side != SIDE_A));
2 HAL_GPIO_WritePin(EN_B_GPIO_Port, EN_B_Pin, (GPIO_PinState)(
    side != SIDE_B));
```

Program 1.10: Bổ sung vào display7SEG(int side int num)



Hình 1.9: Bổ sung chân Enable để thực hiện cơ chế quét LED đơn giản

```
while (1)
{
  run_traffic_light(SIDE_A, traffic_color[SIDE_A]);
  run_traffic_light(SIDE_B, traffic_color[SIDE_B]);
  period--;
  if(period <= 0){
      traffic_timer[SIDE_A]--;
      traffic_timer[SIDE_B]--;
      period = 10;
  }
  if(period % 2 == 0)
    display7SEG(SIDE_A, traffic_timer[SIDE_A]);
  if(period % 2 == 1)
    display7SEG(SIDE_B, traffic_timer[SIDE_B]);
  switch(traffic_color[SIDE_A]){
  case RED_LIGHT:
      if(traffic_timer[SIDE_A] <= 0 && traffic_timer[SIDE_B]
   <= 0){
          // SIDE_A: turn to green, SIDE_B: turn to red
          change_traffic_light(SIDE_A, GREEN_LIGHT);
          change_traffic_light(SIDE_B, RED_LIGHT);
      }
      if(traffic_timer[SIDE_B] <= 0){
          // SIDE_A: remain red, SIDE_B: turn yellow
          change_traffic_light(SIDE_B, YELLOW_LIGHT);
      }
      break;
  case GREEN_LIGHT:
```

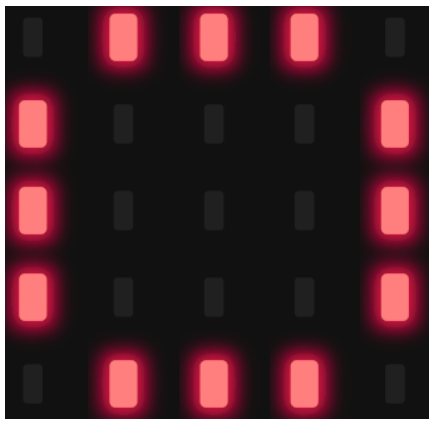
```
28        if(traffic_timer[SIDE_A] <= 0){
29            // SIDE_A: turn yellow, SIDE_B: remain red
30            change_traffic_light(SIDE_A, YELLOW_LIGHT);
31        }
32        break;
33    case YELLOW_LIGHT:
34        if(traffic_timer[SIDE_A] <= 0 && traffic_timer[SIDE_B]
     <= 0){
35            // SIDE_A: turn to red, SIDE_B: turn to green
36            change_traffic_light(SIDE_A, RED_LIGHT);
37            change_traffic_light(SIDE_B, GREEN_LIGHT);
38        }
39        break;
40    default:
41        turn_all_light_off();
42        break;
43    }
44    HAL_Delay(100);
45 }
```

Program 1.11: Vòng lặp while với delay

6 Exercise 6

6.1 Yêu cầu

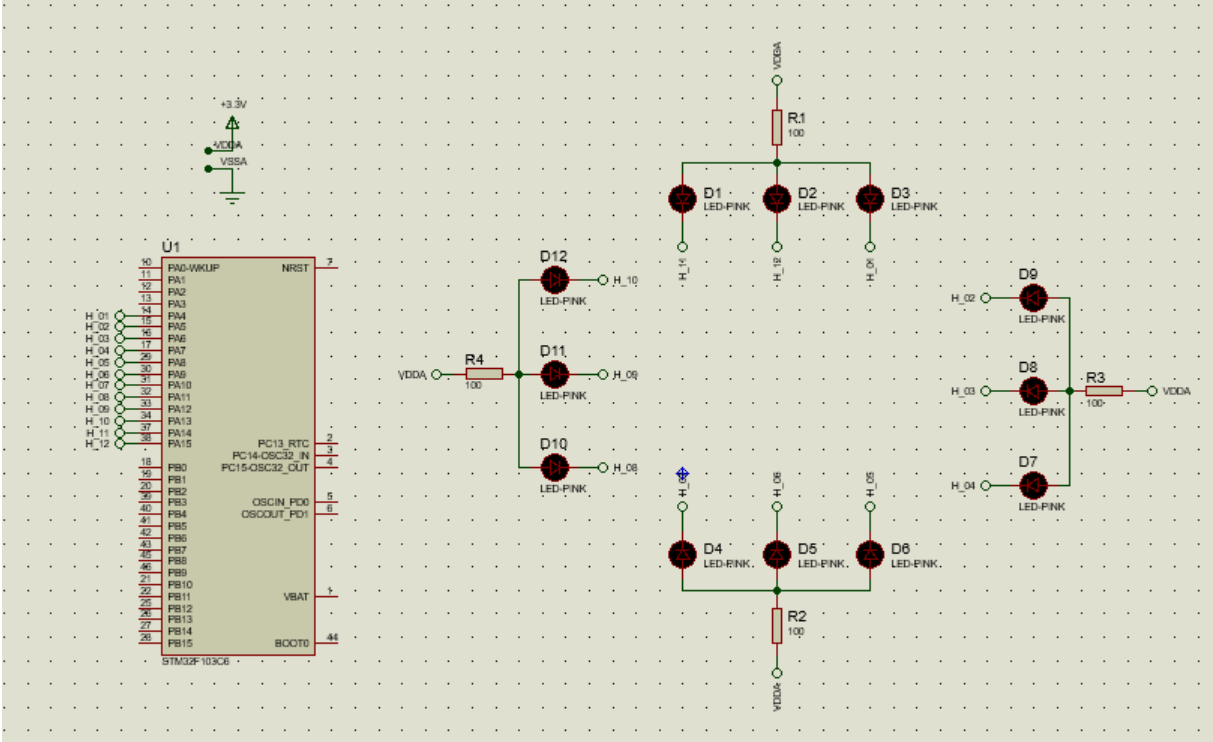
Mô phỏng đồ hồ analog với 12 số khác nhau, lần lượt được kết nối từ **PA4** đến **PA15** theo hình sau:



Hình 1.10: 12 LEDs for an analog clock

Trình bày schematic và hiện thực chương trình đơn giản kết nối tới tất cả các bóng đè và tuần tự tắt mở LED.

6.2 Bài làm



Hình 1.11: Sơ đồ nguyên lý đồng hồ analog

```
void init_analog_clock(){
    analog_led[0] = (GPIO_config){H_01_GPIO_Port, H_01_Pin};
    analog_led[1] = (GPIO_config){H_02_GPIO_Port, H_02_Pin};
    analog_led[2] = (GPIO_config){H_03_GPIO_Port, H_03_Pin};

    analog_led[3] = (GPIO_config){H_04_GPIO_Port, H_04_Pin};
    analog_led[4] = (GPIO_config){H_05_GPIO_Port, H_05_Pin};
    analog_led[5] = (GPIO_config){H_06_GPIO_Port, H_06_Pin};

    analog_led[6] = (GPIO_config){H_07_GPIO_Port, H_07_Pin};
    analog_led[7] = (GPIO_config){H_08_GPIO_Port, H_08_Pin};
    analog_led[8] = (GPIO_config){H_09_GPIO_Port, H_09_Pin};

    analog_led[9] = (GPIO_config){H_10_GPIO_Port, H_10_Pin};
    analog_led[10] = (GPIO_config){H_11_GPIO_Port, H_11_Pin};
    analog_led[11] = (GPIO_config){H_12_GPIO_Port, H_12_Pin};
    void clearAllClock();
}
```

Program 1.12: Hàm thiết lập trạng thái ban đầu và cài đặt port/pin cho LEDs

```
void set_GPIO_on(GPIO_config* destination){
  HAL_GPIO_WritePin(destination->port, destination->pin,
   GPIO_PIN_SET);
}
```

```
void set_GPIO_off(GPIO_config* destination){
  HAL_GPIO_WritePin(destination->port, destination->pin,
   GPIO_PIN_RESET);
}
```

Program 1.13: Hàm hỗ trợ trong global.c để điều khiển tín hiệu tại các pin

```
void test_clock(int position){
    turn_analog_led_on(position);
    if(position == 0)
        turn_analog_led_off(NUM_LEDS - 1);
    if(position != 0)
        turn_analog_led_off(position - 1);
}
```

Program 1.14: Hàm test_clock(int position) hỗ trợ bật/tắt đèn tại vị trí xác định

```
while (1)
{
  test_clock(test_counter);
  test_counter = (test_counter + 1) % NUM_LEDS;
  HAL_Delay(1000);
}
```

Program 1.15: Vòng lặp while(1) để lần lượt chớp tắt đèn LED

7 Exercise 7

7.1 Yêu cầu

Hiện thực hàm `clearAllClock()` để tắt tất cả 12 đèn LEDs, trình bày source code cho hàm này.

7.2 Bài làm

```
void clearAllClock(){
    for(int position = 0; position < NUM_LEDS; position++)
        turn_analog_led_off(position);
}
```

Program 1.16: Hàm clearAllClock()

8 Exercise 8

8.1 Yêu cầu

Hiện thực hàm `setNumberOnClock(int num)`. Dữ liệu đầu vào của hàm là giá trị từ 0 tới 11 thì một đèn LED phù hợp sẽ bật. Trình bày source code của hàm trên.

8.2 Bài làm

```
void setNumberOnClock(int num){
    set_GPIO_off(&analog_led[num]);
}
```

Program 1.17: Hàm setNumberOnClock(int num)

9 Exercise 9

9.1 Yêu cầu

Hiện thực hàm `clearNumberOnClock(int num)`. Dữ liệu đầu vào của hàm là giá trị từ 0 tới 11 thì một đèn LED phù hợp sẽ tắt. Trình bày source code của hàm trên.

9.2 Bài làm

```
void clearNumberOnClock(int num){
    set_GPIO_on(&analog_led[num]);
}
```

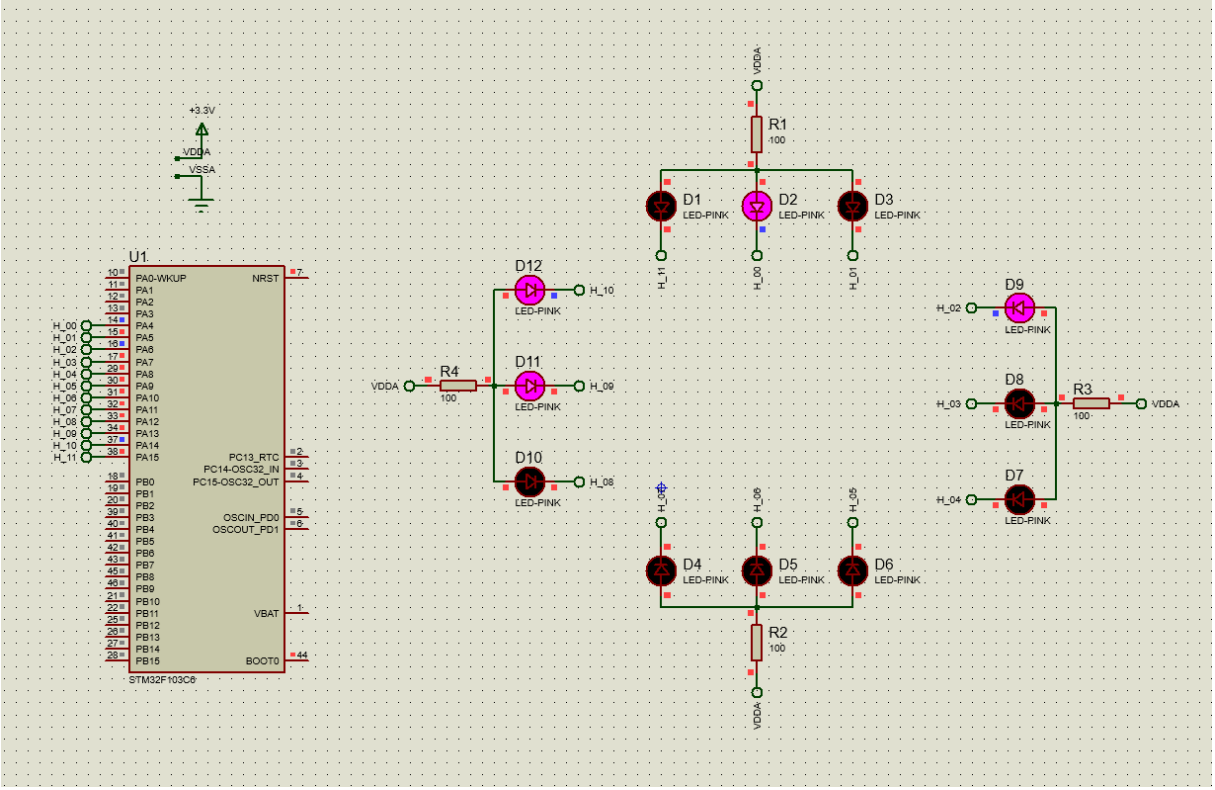
Program 1.18: Hàm clearNumberOnClock(int num)

10 Exercise 10

10.1 Yêu cầu

Tích hợp toàn bộ hệ thống sử dụng 12 đèn LEDs để hiển thị đồng hồ. Tại mọi thời điểm, chỉ có 3 đèn LED được sáng đại diện cho giờ, phút và giây.

10.2 Bài làm



Hình 1.12: Mô phỏng đồng hồ analog

Hiện thực hàm nhận thông số đầu vào là tổng số giây để tính ra thời gian theo giờ, phút, giây tương ứng cần hiển thị.

```
void run_timer(uint32_t sum_seconds){
    int hour = sum_seconds / (60*60),
        minute = (uint32_t)(sum_seconds / 60) % 60,
        second = sum_seconds % 60;
    clearAllClock();
    setNumberOnClock(hour % 12);//Display hour
    setNumberOnClock(minute / 5);//Display minute
    setNumberOnClock(second / 5);//Display second
}
```

Program 1.19: Hàm rum_timer(uint32_t sum_seconds) điều khiển đồng hồ

Trong vòng lặp `while(1)`, ta sử dụng biến đếm `test_counter` để mô phỏng thời gian chạy, với delay 20.

```
while (1)
{
  run_timer(test_counter);
  test_counter = (test_counter + 1) % (24*60*60);
  HAL_Delay(20);
}
```