ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

THÍ NGHIỆM

VI XỬ LÍ - VI ĐIỀU KHIỂN (CO 3010)

BÁO CÁO LAB 2

Giảng viên: PHAN VĂN SỸ

Họ và tên	Mã số sinh viên
Nguyễn Văn Sang	2212912

TP.Hồ Chí Minh, 08/10/2025

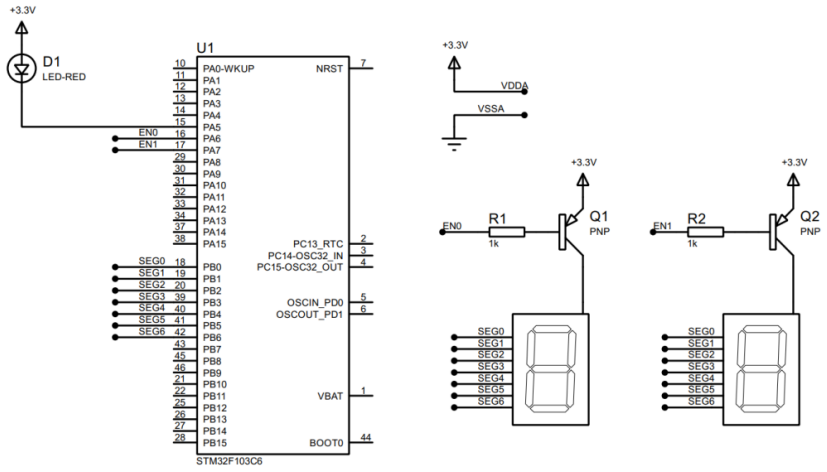
Mục lục

1 Bài 1

The first exercise show how to interface for multiple seven segment LEDs to STM32F103C6 micro-controller (MCU). Seven segment displays are common anode type, meaning that the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins.

In order to save the resource of the MCU, individual cathode pins from all the seven seg ment LEDs are connected together, and connect to 7 pins of the MCU. These pins are popular known as the signal pins. Meanwhile, the anode pin of each seven segment LEDs are controlled under a power enabling circuit, for instance, an PNP transistor. At a given time, only one seven segment LED is turned on. However, if the delay is small enough, it seems that all LEDs are enabling.

Implement the circuit simulation in Proteus with two 7-SEGMENT LEDs as following:



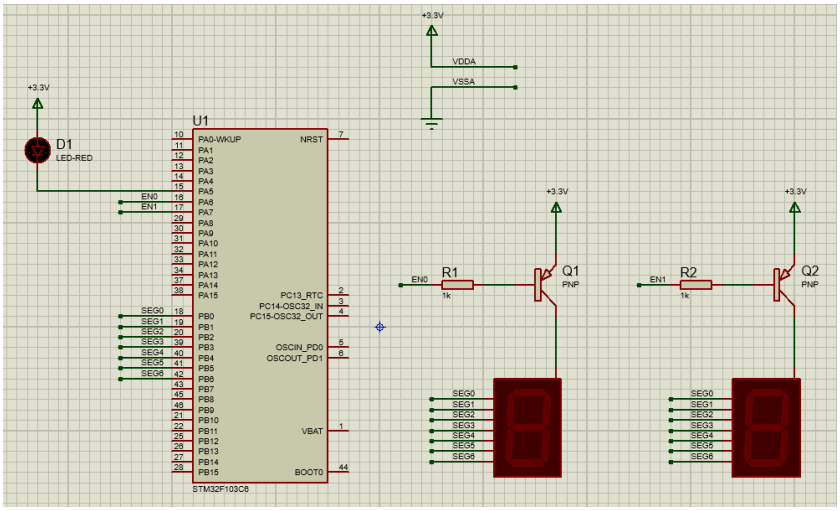
Hình 1: Simulation schematic in Proteus

Components used in the schematic are listed bellow:

- 7SEG-COM-ANODE (connected from PB0 to PB6)
- LED-RED
- PNP
- RES
- STM32F103C6

Students are proposed to use the function display7SEG(int num) in the Lab 1 in this exercise. Implement the source code in the interrupt callback function to display number "1" on the first seven segment and number "2" for second one. The switching time between 2 LEDs is half of second.

Report 1:



Hình 2: Schematic of exercise 1

Report 2:

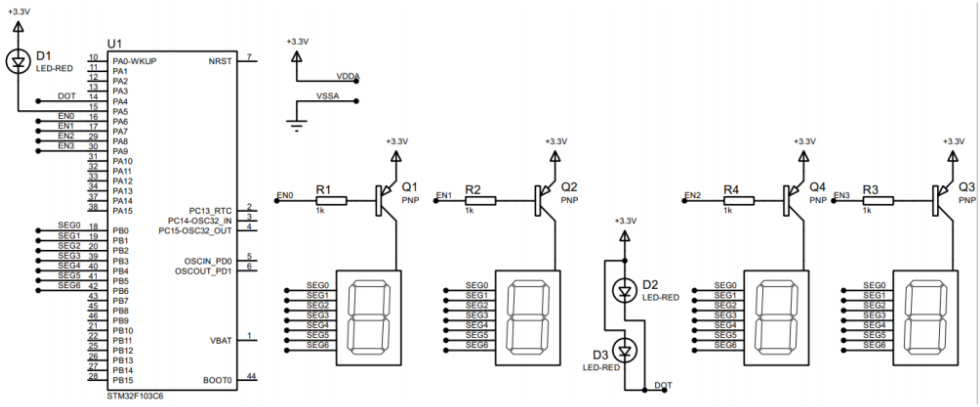
```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
  if (htim == &htim2) {
    static uint8_t toggle = 0;
    static uint16_t count = 0;
    static uint16_t led_count = 0;
    // 7-segment LED changes every 0.5s
    count++;
    if (count >= 50) {
      count = 0;
      toggle = !toggle;
    }

    if (toggle) {
      display7SEG(2);
      HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_RESET);
    } else {
      display7SEG(1);
      HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_SET);
    }
    // Led_Do blinking
    led_count++;
    if (led_count >= 100) {
      led_count = 0;
      #ifdef Led_Do_Pin
        HAL_GPIO_TogglePin(GPIOA, Led_Do_Pin);
      #else
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
      #endif
    }
  }
}
```

Short question: The frequency of the scanning process is: $f = \dfrac{1}{0.5 + 0.5} = 1(Hz)$

Link Github bài 1: `https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B1`

2 Bài 2

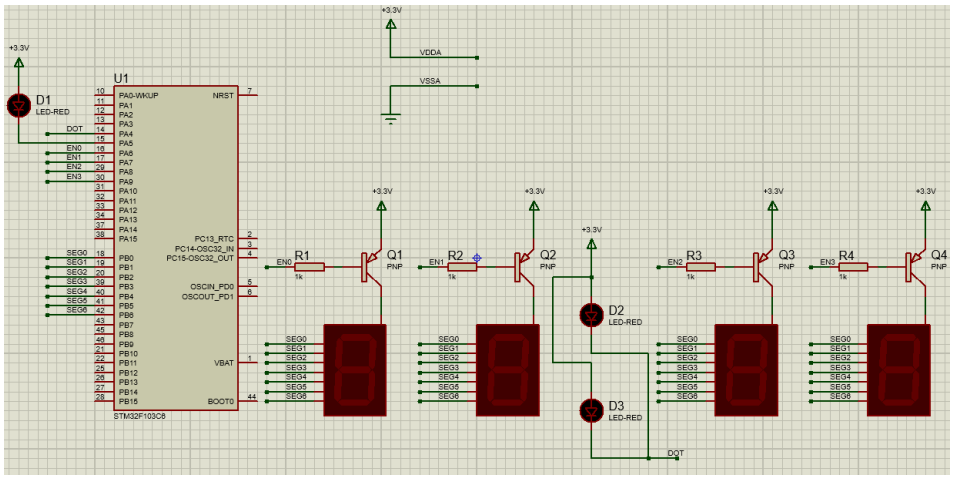
Extend to 4 seven segment LEDs and two LEDs (connected to PA4, labeled as DOT) in the middle as following:



Hình 3: Simulation schematic in Proteus

Blink the two LEDs every second. Meanwhile, number 3 is displayed on the third seven segment and number 0 is displayed on the last one (to present 12 hour and a half). The switching time for each seven segment LED is also a half of second (500ms). Implement your code in the timer interrupt function.

Report 1:



Hình 4: Schematic of exercise 2

Report 2:

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        static uint32_t counter = 0;
        counter++;
        // Handle 7-segment display
        HAL_GPIO_WritePin(GPIOA, EN0_Pin | EN1_Pin | EN2_Pin | EN3_Pin,
            GPIO_PIN_SET);
        uint32_t index = phase % 4;
        uint8_t digits[4] = {1, 2, 3, 0};
        uint8_t pat = patterns[digits[index]];
        HAL_GPIO_WritePin(GPIOB, SEG0_Pin, (pat & 0x01) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG1_Pin, (pat & 0x02) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG2_Pin, (pat & 0x04) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG3_Pin, (pat & 0x08) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG4_Pin, (pat & 0x10) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG5_Pin, (pat & 0x20) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, SEG6_Pin, (pat & 0x40) ? GPIO_PIN_RESET :
            GPIO_PIN_SET);
        uint16_t en_pins[4] = {EN0_Pin, EN1_Pin, EN2_Pin, EN3_Pin};
        HAL_GPIO_WritePin(GPIOA, en_pins[index], GPIO_PIN_RESET);
        // Handle LEDs
        if (counter % 50 == 0) {
          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
          GPIO_PinState state = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_5);
          HAL_GPIO_WritePin(GPIOA, DOT_Pin, state);
          HAL_GPIO_WritePin(GPIOA, Led_Do_Pin, state);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, state);
        }
    }
}
```

Short question: The frequency of the scanning process is: $f = \dfrac{1}{0.5 + 0.5 + 0.5 + 0.5} = 0.5(Hz)$

Link Github bài 2: `https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B2`

3 Bài 3

Implement a function named update7SEG(int index). An array of 4 integer numbers are declared in this case. The code skeleton in this exercise is presented as following:

```c
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1, 2, 3, 4};
void update7SEG(int index) {
    switch (index) {
        case 0:
            // Display the first 7 SEG with led_buffer [0]
            break;
        case 1:
            // Display the first 7 SEG with led_buffer [1]
            break;
        case 2:
            // Display the first 7 SEG with led_buffer [2]
```

```
            break;
        case 3:
            // Display the first 7 SEG with led_buffer [3]
            break;
        default:
            break;
    }
}
```

Listing 1: An example for your source code

This function should be invoked in the timer interrupt, e.g update7SEG(index_led++). The variable index_led is updated to stay in a valid range, which is from 0 to 3. Students are proposed to change the values in the led_buffer array for unit test this funtion, which is used afterward.

Report 1:

```
void update7SEG(int index) {
    // Turn off all enable pins
    HAL_GPIO_WritePin(GPIOA, EN0_Pin | EN1_Pin | EN2_Pin | EN3_Pin,
        GPIO_PIN_SET);
    // Get digits from buffer
    int digit = led_buffer[index];
    if (digit < 0 || digit > 9) digit = 0;
    // Set segment pins based on pattern
    uint8_t pattern = seven_seg_patterns[digit];
    for (int i = 0; i < 7; i++) {
        GPIO_PinState state = (pattern & (1 << i)) ? GPIO_PIN_RESET :
            GPIO_PIN_SET;
        HAL_GPIO_WritePin(GPIOB, seg_pins[i], state);
    }
    // Turn on the corresponding enable pin
    switch (index) {
        case 0:
            HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_RESET);
            break;
        case 1:
            HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_RESET);
            break;
        case 2:
            HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_RESET);
            break;
        case 3:
            HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_RESET);
            break;
        default:
            break;
    }
}
```

Report 2:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    static int counter = 0;
    if (htim == &htim2) {
        // Update 7-segment every 50 interrupts
        if (counter % 50 == 0) {
            update7SEG(index_led);
            index_led = (index_led + 1) % MAX_LED;
        }
```

```
        // Update LED_RED and DOT pin every 100 interrupts
        if (counter % 100 == 0) {
            HAL_GPIO_TogglePin(GPIOA, Led_Do_Pin | DOT_Pin);
        }
        counter++;
    }
}
```

Link Github bài 3: https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B3

4 Bài 4

Change the period of invoking update7SEG function in order to set the frequency of 4 seven segment LEDs to 1Hz. The DOT is still blinking every second.

Report 1:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    static int counter = 0;
    if (htim == &htim2) {
        // Update 7-segment every 25 interrupts
        if (counter % 25 == 0) {
            update7SEG(index_led);
            index_led = (index_led + 1) % MAX_LED;
        }
        // Update LED_RED and DOT pin every 100 interrupts
        if (counter % 100 == 0) {
            HAL_GPIO_TogglePin(GPIOA, Led_Do_Pin | DOT_Pin);
        }
        counter++;
    }
}
```

Link Github bài 4: https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B4

5 Bài 5

Implement a digital clock with hour and minute information displayed by 2 seven segment LEDs. The code skeleton in the main function is presented as follows:

```
int hour = 15, minute = 8, second = 50;
while (1) {
    second++;
    if (second >= 60) {
        second = 0;
        minute++;
    }
    if (minute >= 60) {
        minute = 0;
        hour++;
    }
    if (hour >= 24) {
        hour = 0;
    }
    updateClockBuffer();
    HAL_Delay(1000);
}
```

Listing 2: An example for your source code

The function updateClockBuffer will generate values for the array led_buffer according to the values of hour and minute. In the case these values are 1 digit number, digit 0 is added.

Report 1:

```
void updateClockBuffer(void) {
    led_buffer[0] = hour / 10;
    led_buffer[1] = hour % 10;
    led_buffer[2] = minute / 10;
    led_buffer[3] = minute % 10;
}
```

6 Bài 6

The main target from this exercise to reduce the complexity (or reduce code processing) in the timer interrupt. The time consumed in the interrupt can lead to the nested interrupt issue, which can crash the whole system. A simple solution can disable the timer whenever the interrupt occurs, the enable it again. However, the real-time processing is not guaranteed anymore.

In this exercise, a software timer is created and its counter is count down every timer interrupt is raised (every 10ms). By using this timer, the Hal_Delay(1000) in the main function is removed. In a MCU system, non-blocking delay is better than blocking delay. The details to create a software timer are presented bellow. The source code is added to your current program, do not delete the source code you have on Exercise 5.

Step 1: Declare variables and functions for a software timer, as following:

```
/* USER CODE BEGIN 0 */
int timer0_counter = 0;
int timer0_flag = 0;
int TIMER_CYCLE = 10;
void set
Timer0(int duration) {
    timer0_counter = duration / TIMER_CYCLE;
    timer0_flag = 0;
}
void timer_run() {
    if (timer0_counter > 0) {
        timer0_counter --;
    if (timer0_counter == 0) timer0_flag = 1;
    }
}
/* USER CODE END 0 */
```

Listing 3: Software timer based timer interrupt

Please change the TIMER_CYCLE to your timer interrupt period. In the manual code above, it is 10ms.

Step 2: The timer_run() is invoked in the timer interrupt as following:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim)  {
    timer_run();
    // YOUR OTHER CODE
}
```

Listing 4: Software timer based timer interrupt

Step 3: Use the timer in the main function by invoked setTimer0 function, then check for its flag (timer0_flag). An example to blink an LED connected to PA5 using software timer is shown as follows:

```
setTimer0(1000);
while (1) {
    if(timer0_flag == 1) {
        HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
        setTimer0(2000);
    }
}
```

Listing 5: Software timer is used in main fuction to blink the LED

Report 1: Nếu không gọi setTimer0(1000) trước vòng while, thì timer0_counter vẫn là giá trị ban đầu (thường 0) và timer0_flag = 0.

Vòng while sẽ không bao giờ vào nhánh if (vì timer0_flag không bao giờ được thiết lập thành 1) tiếp đó là đèn không nhấp nháy vì setTimer0() phải khởi tạo counter để bắt đầu đếm ngược.

Tóm lại là nếu thiếu dòng khởi tạo thì sẽ không có lần nhấp nháy đầu tiên.

Report 2: setTimer0(1) gán timer0_counter = 1 / 10 = 0 (do chia nguyên).

Sau gọi này, timer0_counter = 0 và timer0_flag được đặt = 0 tiếp đến timer_run() không bao giờ thấy counter > 0, do đó timer0_flag sẽ không bao giờ chuyển thành 1.

Kết quả: vòng while cũng không thực hiện nhánh if do đó không nhấp nháy vì đây là hậu quả của chia nguyên khi duration < TIMER_CYCLE.

Report 3: setTimer0(10) gán timer0_counter = 10 / 10 = 1.

Khi ngắt timer gọi timer_run() một lần (10 ms), timer0_counter từ 1 → 0 và timer0_flag được đặt = 1 thì vòng while sẽ kích hoạt if sau 10 ms vì vậy khác biệt so với 2 câu trước:

- so với thiếu dòng: có nhấp nháy (nhanh) vì counter được khởi tạo.
- so với setTimer0(1): khác vì 1 chia cho 10 bằng 0 và 10 chia cho 10 bằng 1.

Tóm lại setTimer0(10) sẽ tạo ra một khoảng trễ 1 × TIMER_CYCLE = 10 ms trước khi cờ được set.

7 Bài 7

Upgrade the source code in Exercise 5 (update values for hour, minute and second) by using the software timer and remove the HAL_Delay function at the end. Moreover, the DOT (connected to PA4) of the digital clock is also moved to main function.

Report 1:

```
while (1) {
    if (timer1_flag == 1) {
        if (index_led == 0) {
            second++;
            if (second >= 60) {
                second = 0;
                minute++;
            }
            if (minute >= 60) {
```

```
                minute = 0;
                hour++;
            }
            if (hour >= 24) {
                hour = 0;
            }
            HAL_GPIO_TogglePin(GPIOA, Led_Do_Pin | DOT_Pin);
            updateClockBuffer();
        }
        update7SEG(index_led);
        index_led = (index_led + 1) % 4;
        setTimer1(250);
    }
}
```

8 Bài 8

Move also the update7SEG() function from the interrupt timer to the main. Finally, the timer interrupt only used to handle software timers. All processing (or complex computations) is move to an infinite loop on the main function, optimizing the complexity of the interrupt handler function.

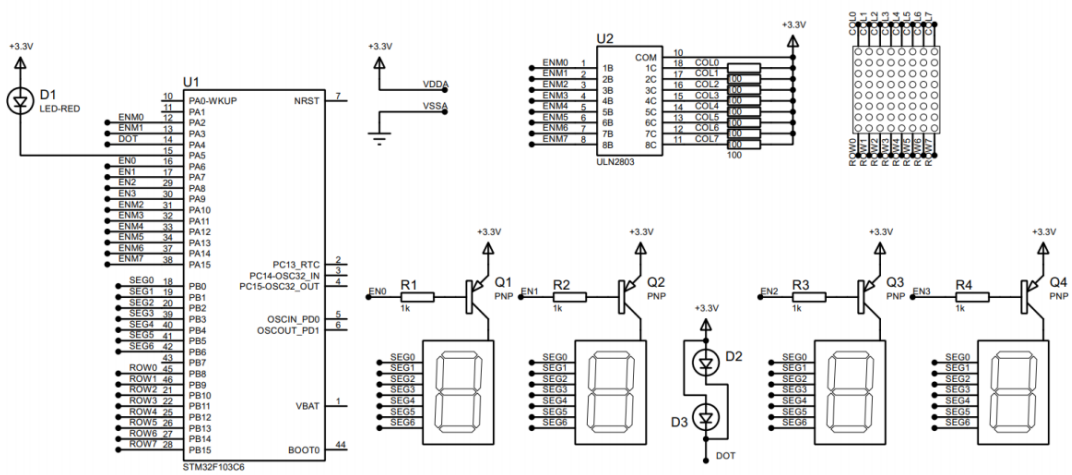
Report 1:

```
//----- main.c -----//
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        timer_run();
    }
}

//----- software_timer.h -----//
void timer_run(void);

//----- software_timer.c -----//
void timer_run(void) {
    if (timer0_counter > 0) {
        timer0_counter--;
    if (timer0_counter == 0) timer0_flag = 1;
    }
    if (timer1_counter > 0) {
        timer1_counter--;
    if (timer1_counter == 0) timer1_flag = 1;
    }
}
```

Link Github bài 5, 6, 7 và 8: `https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B5-8`

9 Bài 9

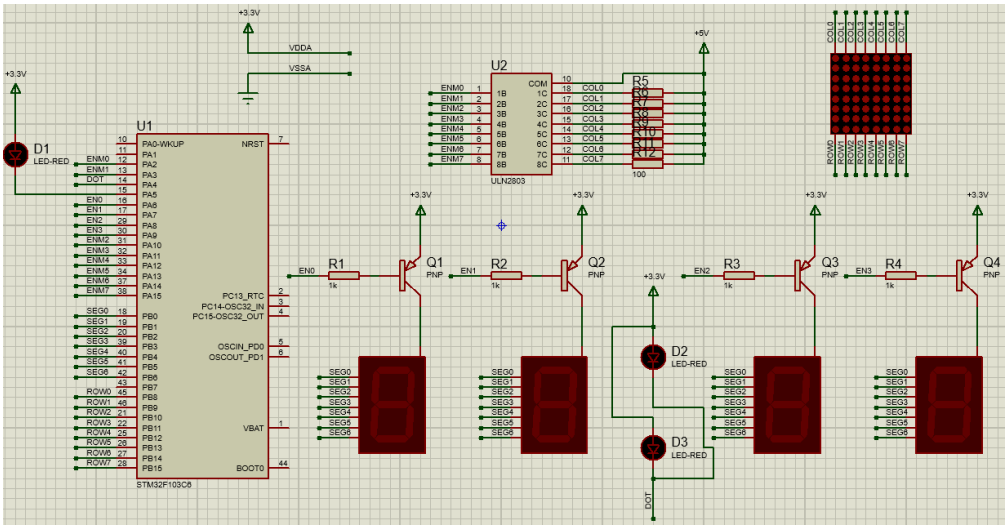
This is an extra works for this lab. A LED Matrix is added to the system. A reference design is shown in figure bellow:



Hình 5: LED matrix is added to the simulation

In this schematic, two new components are added, including the MATRIX-8X8-RED and ULN2803, which is an NPN transistor array to enable the power supply for a column of the LED matrix. Students can change the enable signal (from ENM0 to ENM7) if needed. Finally, the data signal (from ROW0 to ROW7) is connected to PB8 to PB15.

Report 1:



Hình 6: Schematic of exercise 9

Report 2:

```
1  void updateLEDMatrix(int index) {
2      HAL_GPIO_WritePin(GPIOA, ENM0_Pin|ENM1_Pin|ENM2_Pin|ENM3_Pin|ENM4_Pin|
           ENM5_Pin|ENM6_Pin|ENM7_Pin, GPIO_PIN_SET);
3
```

```
    switch (index) {
        case 0: HAL_GPIO_WritePin(GPIOA, ENM0_Pin, GPIO_PIN_RESET);
            break;
        case 1: HAL_GPIO_WritePin(GPIOA, ENM1_Pin, GPIO_PIN_RESET);
            break;
        case 2: HAL_GPIO_WritePin(GPIOA, ENM2_Pin, GPIO_PIN_RESET);
            break;
        case 3: HAL_GPIO_WritePin(GPIOA, ENM3_Pin, GPIO_PIN_RESET);
            break;
        case 4: HAL_GPIO_WritePin(GPIOA, ENM4_Pin, GPIO_PIN_RESET);
            break;
        case 5: HAL_GPIO_WritePin(GPIOA, ENM5_Pin, GPIO_PIN_RESET);
            break;
        case 6: HAL_GPIO_WritePin(GPIOA, ENM6_Pin, GPIO_PIN_RESET);
            break;
        case 7: HAL_GPIO_WritePin(GPIOA, ENM7_Pin, GPIO_PIN_RESET);
            break;
        default:
            break;
    }
    for (int i = 0; i < LED_MATRIX_SIZE; i++) {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8 << i, (matrix_buffer[index] >> i)
            & 1);
    }
}
```

Link Github bài 9: `https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B9`

10 Bài 10

Create an animation on LED matrix, for example, the character is shifted to the left.

Report 1:

Khởi tạo dữ liệu ban đầu:

- Mảng uint8_t matrix_buffer[8]=0xFF, 0xFF, 0x83, 0xED, 0xED, 0x83, 0xFF, 0xFF.
- Mỗi phần tử đại diện cho một cột của matrix.

Hàm thực hiện dịch chuyển moveMatrix():

- Lặp qua từng cột i từ 0 đến 7.
- Dịch chuyển từng byte sang phải 1 bit: matrix_buffer[i] >> 1.
- Di chuyển bit cuối cùng lên đầu: ((matrix_buffer[i] & 0x01) << 7).

Kích hoạt định kỳ qua timer:

- Trong vòng lặp while (1) của main(), kiểm tra t4_flag == 1.
- Gọi moveMatrix() để cập nhật dữ liệu.
- Đặt lại setTimer4(25), đảm bảo hiệu ứng mượt mà.

Hiển thị hiệu ứng:

- Hàm updateLEDMatrix(int index) cập nhật một cột cụ thể bằng cách điều khiển chân GPIO (ENM cho chọn cột, ROW cho chọn hàng).

- setTimer3(1) quét qua các cột là từ 0 đến 7, modulo 8) để làm mới lại toàn bộ LED ma trận nhanh chóng, tạo cảm giác chuyển động liên tục.

```
// The function that performs the shift
void moveMatrix(void) {
    for (int i = 0; i < LED_MATRIX_SIZE; i++) {
        matrix_buffer[i] = (matrix_buffer[i] >> 1) | ((matrix_buffer[i] & 0x01
            ) << 7);
    }
}
// Trigger periodically via setTimer3
if (t4_flag == 1) {
    moveMatrix();
    setTimer4(25);
}
```

Link Github bài 10: https://github.com/SangNguyen-232/VXLVDK-LAB2/tree/main/B10

11 Link Github tổng hợp LAB 2

https://github.com/SangNguyen-232/VXLVDK-LAB2