VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Microprocessor - Microcontroller

Lab Report - CO3010

Lab 2

Advisor(s): Phan Văn Sỹ
Student(s): Chu Le Hoang 2352346

HO CHI MINH CITY, SEPTEMBER 2025

Contents

1 Overall

Lab schematics and the source codes are submitted via GitHub link: https://github.com/KennyLe298/MPU-MCU

2 Exercise 1

2.1 Report 1

Schematic of exercise 1:

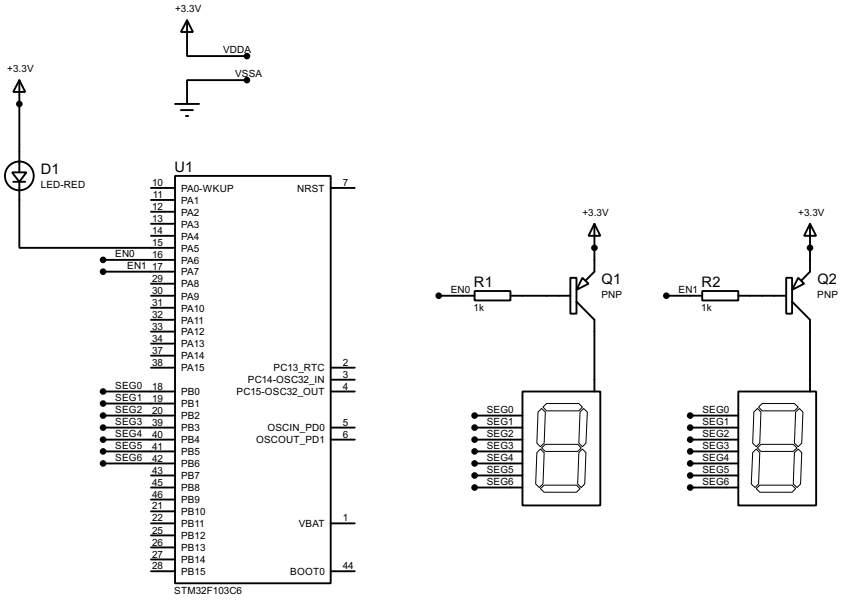

Figure 2.1: Schematic of exercise 1

2.2 Report 2

Source code of `HAL TIM PeriodElapsedCallback` function and its supporting functions :

```c
void init(){
   display7SEG(1);
   HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
   HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
}
//initial stage of the leds

int counter1 = 50; //50*10ms = 500ms (half a second)
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim){
    counter1--;
    if (counter1 == 0){
        if (HAL_GPIO_ReadPin(EN0_GPIO_Port, EN0_Pin) == 0){
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
            display7SEG(2);
        }
        else {
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
            display7SEG(1);
        }
        counter1 = 50;
    }
```

Short question: What is the frequency of the scanning process?

Frequency is calculated by

$$f = \frac{1}{T}$$

so we have:

$$f = \frac{1}{0.5(s) + 0.5(s)} = 1(Hz)$$

3 Exercise 2

3.1 Report 1

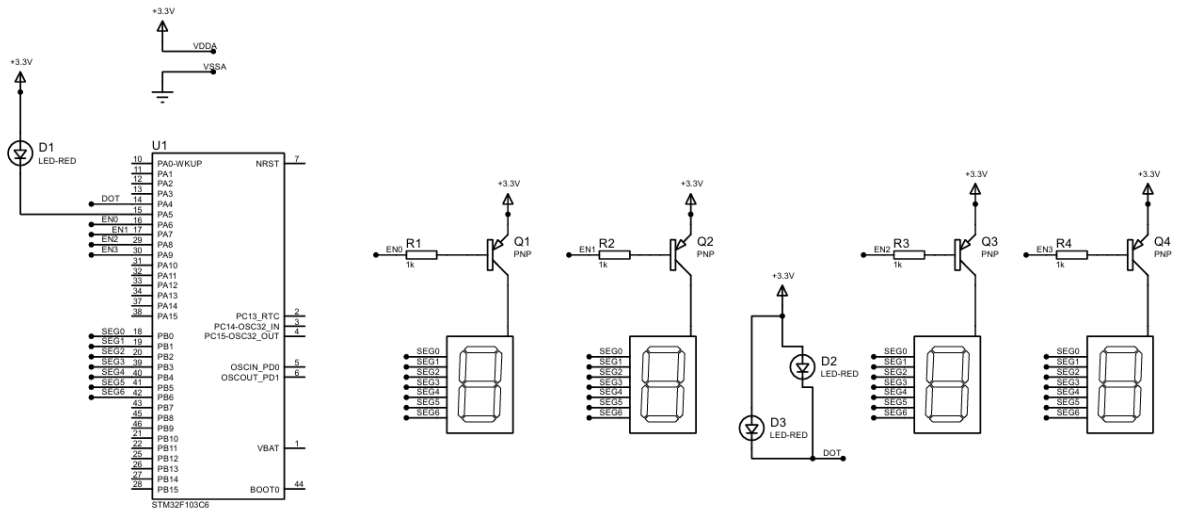

Figure 3.1: Schematic for exercise 2

3.2 Report 2

Source code of `HAL TIM PeriodElapsedCallback` function and its supporting functions:

```
int clock_1 = 100;
int clock7Seg = 50;
int stage = 0; //initial stage
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
clock_1--;
if (clock_1 <= 0) {
    HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
    clock_1 = 100;
}
clock7Seg--;
if (clock7Seg <= 0){
    switch (stage){
        case 0:
        {
```

```
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
            display7SEG(1);
            stage = 1;
            break;
        }
        case 1:
        {
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
            display7SEG(2);
            stage = 2;
            break;
        }
        case 2:
        {
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
            display7SEG(3);
            stage = 3;
            break;
        }
        case 3:
        {
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
            display7SEG(0);
            stage = 0;
            break;
        }
```

```
    }
    clock7Seg = 50;
   }
}
```

Short question: What is the frequency of the scanning process?

Frequency is calculated by

$$f = \frac{1}{T}$$

so we have:

$$f = \frac{1}{0.5(s) + 0.5(s) + 0.5(s) + 0.5(s)} = 0.5(Hz)$$

4 Exercise 3

4.1 Report 1

Source code of the update7SEG function:

```
void update7SEG(int index){
    if(index_led >= MAX_LED){
        index_led = 0;
    }
    index = index%4;
    switch(index){
    case 0:
    {
        selectEn(index);
        display7SEG(led_buffer[index]);
        break;
    }
    case 1:
    {
        selectEn(index);
        display7SEG(led_buffer[index]);
        break;
    }
    case 2:
    {
```

```
        selectEn(index);
        display7SEG(led_buffer[index]);
        break;
    }
    case 3:
    {
        selectEn(index);
        display7SEG(led_buffer[index]);
        break;
    }
    default:
    {
        selectEn(-1);
        break;
    }
    }
}
```

4.2 Report 2

Source code of in the `HAL_TIM_PeriodElapsedCallback`

```
int clock_1 = 50;
int clockBlink = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
    clock_1--;
    clockBlink--;
    if(clock_1 <= 0){
        update7SEG(index_led);
        index_led++;
        clock_1 =50;
    }
    if(clockBlink <= 0){
        HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
        clockBlink = 100;
    }
}
```

5 Exercise 4

5.1 Report 1

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.

```
int clock_1 = 25; //changed frequency of each led
int clockBlink = 100; //keep the dot blinking every sec
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
    clock_1--;
    clockBlink--;
    if(clock_1 <= 0){
        update7SEG(index_led);
        index_led++;
        clock_1 =25;
    }
    if(clockBlink <= 0){
        HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
        clockBlink = 100;
    }
}
```

To change the frequency to 1Hz, we simply change the counter to be 25 because:

$$f = \frac{1}{T} = \frac{1}{0.25(s) + 0.25(s) + 0.25(s) + 0.25(s)} = 1(Hz)$$

6 Exercise 5

6.1 Report 1

Source code of `void updateClockBuffer()`:

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

7 Exercise 6

7.1 Report 1

If in line 1 of the code above is miss, what happens after that and why?

If we don't set Timer0 function, the timer0 counter will not be calculated and the flag is not set to 0 after each iteration. Therefore, the LED will not blink due to the flag is not triggered and the clock will not be set back to it's default value.

7.2 Report 2

If in line 1 of the code above is changed to setTimer0(1), what happens after that and why?

If the input duration is 1, then the counter will be calculated by duration divided by cycle, $\frac{1}{10}$, which results in 0 for the counter. So if the duration is less than 10, the result will always be 0, which will cause the counter to not run at all. This is equivalent to not having the function and the LED will not blink.

7.3 Report 3

If in line 1 of the code above is changed to setTimer0(10), what is changed compared to 2 first question?

Then the counter will equal to $\frac{10}{10} = 1$. This means the counter will have 2 stage 0 and 1, this causes the LED to toggle every 2 seconds while running the loop.

7.4 Example run case

```
1 //Using the software timer functions from the LAB_Timer file
2   setTimer0(1000);
3   while (1)
4   {
5       if(timer0_flag == 1){
6               HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
7               // Blink RED LED
8               setTimer0(1000);
9           }
10  }
```

8 Exercise 7

8.1 Report 1

Source code in the while loop on main function:

```
int hour = 15, minute = 8, second = 50;
setTimer0(1000); //1s
while (1){
    if(timer0_flag == 1){
        second++;
        if (second >= 60){
            second = 0;
            minute++;
        }
        if (minute >= 60){
            minute = 0;
            hour++;
        }
        if (hour >= 24){
            hour = 0;
        }

        updateClockBuffer();
        HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin); // DOT blink
        setTimer0(1000); // reset
    }
}
```

9 Exercise 8

9.1 Report 1

Source code in the main function:

```
int hour = 15, minute = 8, second = 50;
int index_led = 0;
setTimer0(1000); //1s in real time
setTimer1(250);
while (1){
    if(timer0_flag == 1){
      second++;
      if(second >= 60){
          second = 0;
          minute++;
      }
      if(minute >= 60){
          minute = 0;
          hour++;
      }
      if(hour >= 24){
          hour = 0;
      }
      HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin); // DOT blink
      setTimer0(1000); // restart 1s
  }

  if(timer1_flag == 1){
      update7SEG(index_led++);
      if(index_led >= 4) index_led = 0;
      setTimer1(250); // restart scan blink
  }
  updateClockBuffer();
}
```

10 Exercise 9

10.1 Report 1

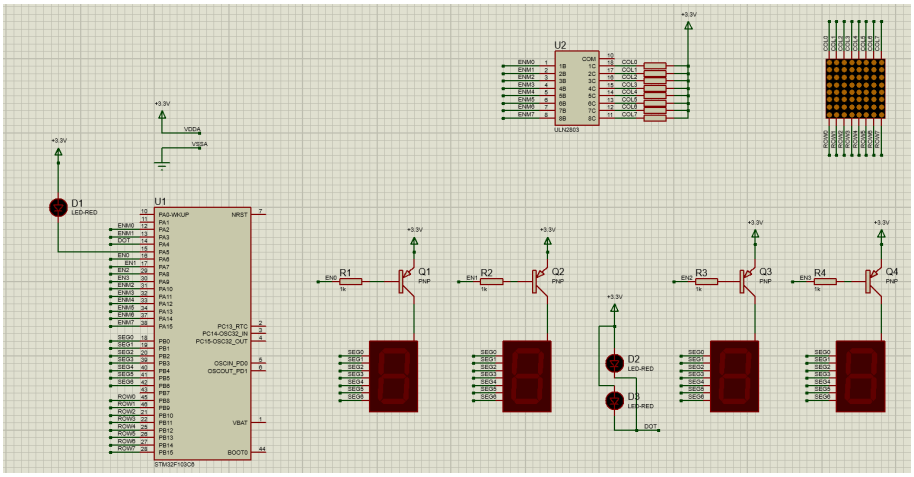

Figure 10.1: Exercise 9 and 10 schematics

10.2 Report 2

Source code to display 'A' word on the Matrix 8x8 LED:

```
//Timer2
int timer2_counter = 0;
int timer2_flag = 0;
void setTimer2(int duration){ timer2_counter = duration /
   TIMER_CYCLE; timer2_flag = 0; }
void timer_run(){
    //Timer0 and Timer 1 here
   //for ex9
    if(timer2_counter > 0){
        timer2_counter--;
        if(timer2_counter == 0) timer2_flag = 1;
    }
}
const int MAX_LED_MATRIX = 8;
int index_led_matrix = 0;

uint8_t matrix_buffer[8] =
```

```
{
    0b00011000,
    0b00100100,
    0b01000010,
    0b01000010,
    0b01111110,
    0b01000010,
    0b01000010,
    0b01000010
};

void selectRow(int index){
    HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin, (index == 0) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin, (index == 1) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin, (index == 2) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin, (index == 3) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin, (index == 4) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin, (index == 5) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin, (index == 6) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin, (index == 7) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
}

void writeCols(uint8_t data){
    HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, (data & 0x80) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, (data & 0x40) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, (data & 0x20) ?
       GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, (data & 0x10) ?
```

```
        GPIO_PIN_SET : GPIO_PIN_RESET);
44      HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, (data & 0x08) ?
            GPIO_PIN_SET : GPIO_PIN_RESET);
45      HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, (data & 0x04) ?
            GPIO_PIN_SET : GPIO_PIN_RESET);
46      HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, (data & 0x02) ?
            GPIO_PIN_SET : GPIO_PIN_RESET);
47      HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, (data & 0x01) ?
            GPIO_PIN_SET : GPIO_PIN_RESET);
48  }
49  void led_matrix_init(){
50      HAL_GPIO_WritePin(GPIOA, ENM0_Pin|ENM1_Pin|ENM2_Pin|ENM3_Pin|
            ENM4_Pin|ENM5_Pin|ENM6_Pin|ENM7_Pin, GPIO_PIN_RESET);
51      HAL_GPIO_WritePin(GPIOB, ROW0_Pin|ROW1_Pin|ROW2_Pin|ROW3_Pin|
            ROW4_Pin|ROW5_Pin|ROW6_Pin|ROW7_Pin, GPIO_PIN_RESET);
52  }
53  void updateLEDMatrix(int index){
54      index = index % MAX_LED_MATRIX;
55      HAL_GPIO_WritePin(GPIOA, ENM0_Pin|ENM1_Pin|ENM2_Pin|ENM3_Pin|
            ENM4_Pin|ENM5_Pin|ENM6_Pin|ENM7_Pin, GPIO_PIN_RESET);
56      writeCols(matrix_buffer[index]);
57      selectRow(index);
58  }
```

11 Exercise 10

To implement the shifting of the character, we add these functions:

```
void timer_run(){
    //timer0, timer1 and timer2 here
    //for ex10
    if(timer3_counter > 0){
        timer3_counter--;
        if(timer3_counter == 0) timer3_flag = 1;
    }
}
//Shift Left
void shiftCharacterLeft(void){
    for(int i = 0; i < MAX_LED_MATRIX; i++){
        uint8_t leftmost_bit = (matrix_buffer[i] & 0b10000000) >>
           7;
        matrix_buffer[i] = matrix_buffer[i] << 1;
        matrix_buffer[i] = matrix_buffer[i] | leftmost_bit;
    }
}
//Shift Right
void shiftCharacterRight(void){
    for(int i = 0; i < MAX_LED_MATRIX; i++){
        uint8_t rightmost_bit = (matrix_buffer[i] & 0b00000001) <<
            7;
        matrix_buffer[i] = matrix_buffer[i] >> 1;
        matrix_buffer[i] = matrix_buffer[i] | rightmost_bit;
    }
}
//Shift Down
void shiftCharacterDown(void){
    uint8_t last_row = matrix_buffer[MAX_LED_MATRIX - 1];
    for(int i = MAX_LED_MATRIX - 2; i >= 0; i--){
        matrix_buffer[i+1] = matrix_buffer[i];
    }
    matrix_buffer[0] = last_row;
}
```