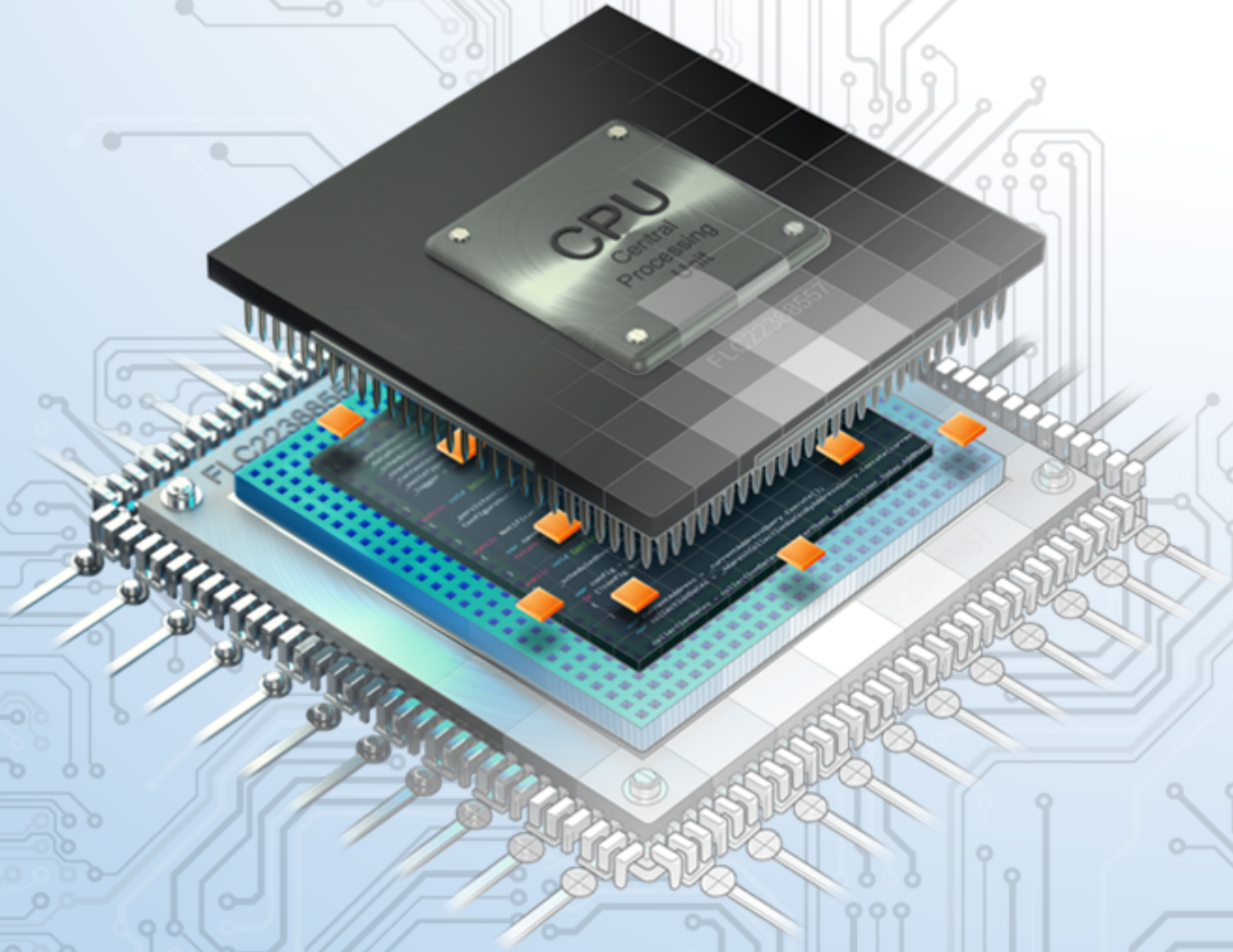
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



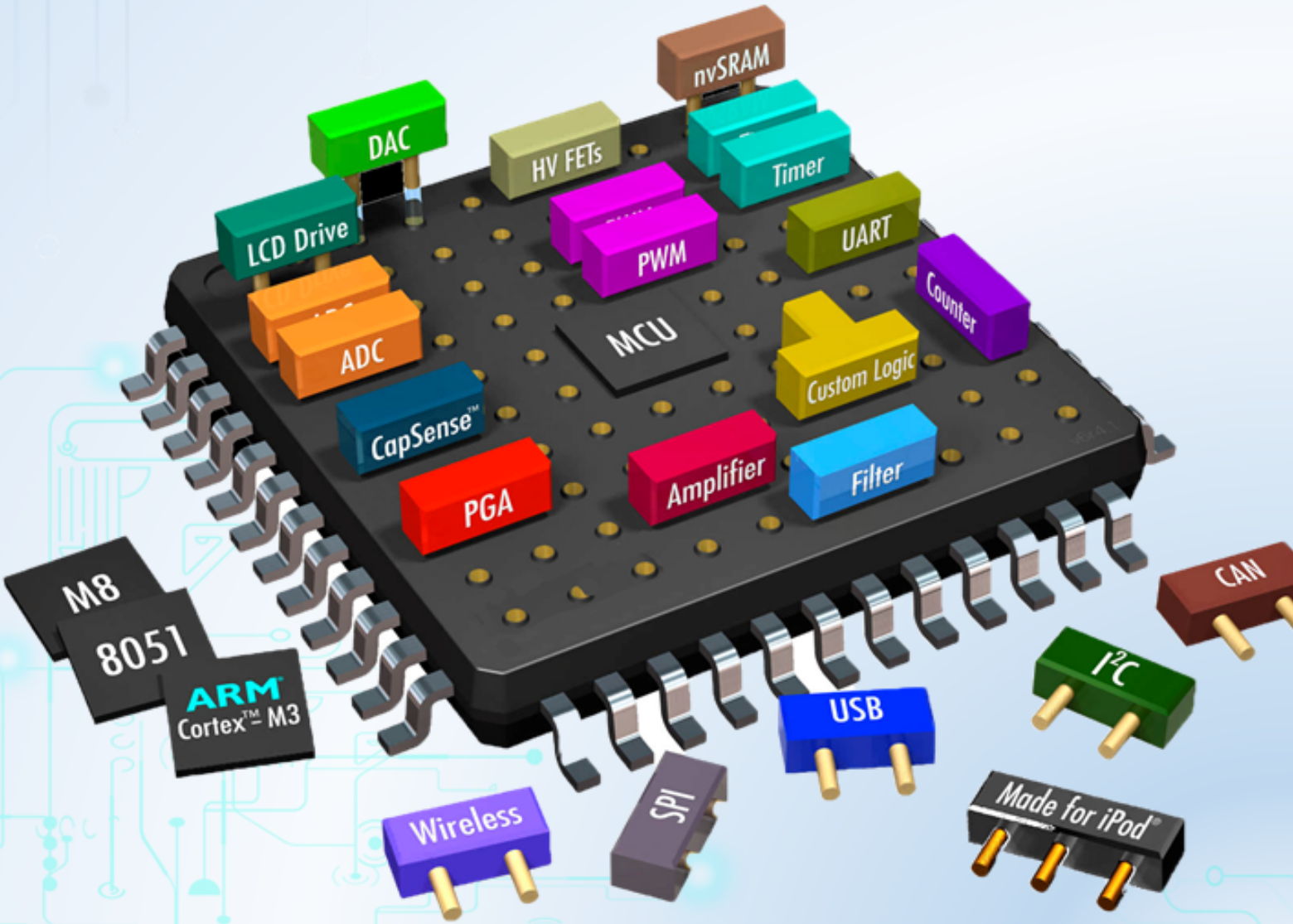
Dr. Le Trong Nhan

Contents

Chapter 1. Timer Interrupt and LED Scanning **7**
1 Introduction 8
2 Exercise and Report 9
2.1 Exercise 1 9
2.2 Exercise 2 11
2.3 Exercise 3 13
2.4 Exercise 4 16
2.5 Exercise 5 16
2.6 Exercise 6 18
2.7 Exercise 7 19
2.8 Exercise 8 20
2.9 Exercise 9: 21
2.10 Exercise 10: 24

CHAPTER 1

Timer Interrupt and LED Scanning



1 Introduction

Timers are one of the most important features in modern micro-controllers. They allow us to measure how long something takes to execute, create non-blocking code, precisely control pin timing, and even run operating systems.

The STM32 line of micro-controllers from ST-Microelectronics are no exception: each controller offers a full suite of timers for us to use. In this manual, how to configure a timer using STM32CubeIDE is presented how to use them to flash an LED. Finally, students are proposed to finalize 10 exercises using timer interrupt for applications based LED Scanning.

2 Exercise and Report

2.1 Exercise 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 segment 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:

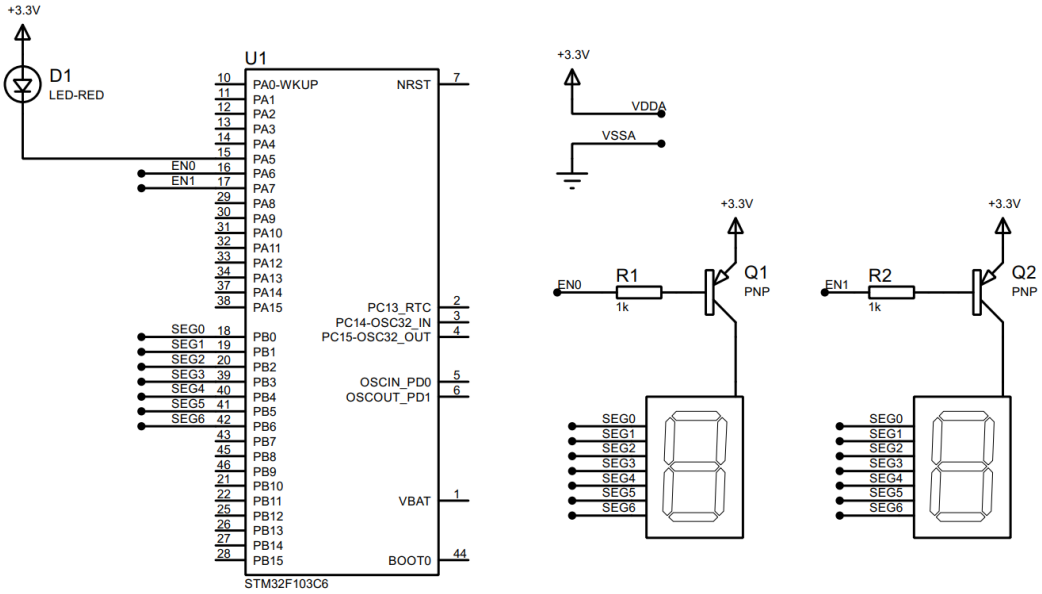


Figure 1.1: Simulation schematic in Proteus

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: Capture your schematic from Proteus and show in the report.

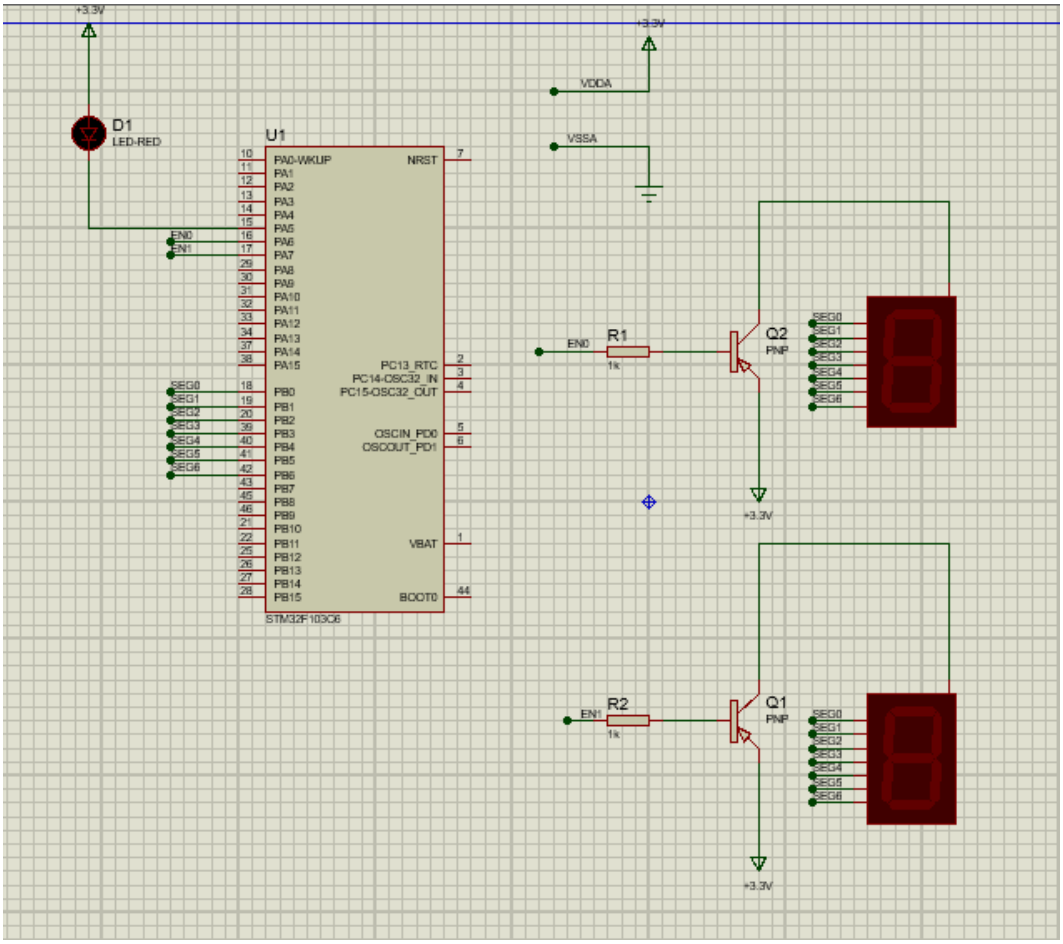


Figure 1.2: Schematic in Proteus

Report 2: Present your source code in the
Ideas:: Create a global variable is num to select the first or seconds 7SEGMENT LEDS will turn on. Beside that , because the timer of **HAL_TIM_PeriodElapsedCallback** is 10ms, it's mean after 10ms function will be called.So to guaranti the timer between 2 LEDS is hafl of second (500ms), we need to set a mount of call funtion after first LEDS turn on is 50.So in each of loop function **HAL_TIM_PeriodElapsedCallback**, counter was decrease 1 value, when counter equals 0 we turn on first or seconds LED and set counter variable is 50. We can reuse funtion **display7SEG** in previous lab to display 2 LEDS. **HAL_TIM_PeriodElapsedCallback** function.

```
int counter  = 100;
int num = 1;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
   {
  counter--;
  if(counter <= 0){
    switch(num){
```

```
        case 1:
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
          display7SEG(1);
          num = 2;
          break;
        case 2:
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
          display7SEG(2);
          num = 1;
          break;
        }
      counter = 50;
    }
}
```

Program 1.1: Source code

Short question: What is the frequency of the scanning process?
Answer: The timer of the scanning process is 1s (timer 1 leds turn on is 0,5s, we have 2 ledsleds) so the Frequency of the scanning process is 1/1 = 1Hz.

2.2 Exercise 2

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

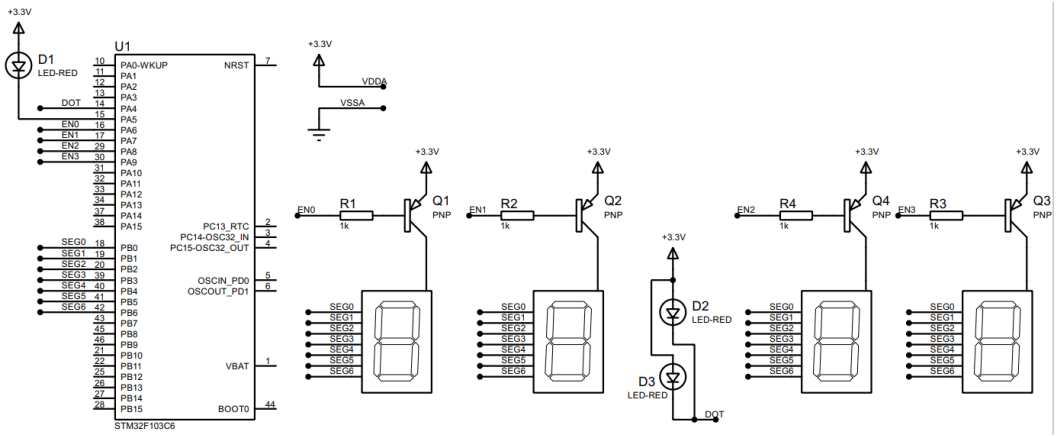


Figure 1.3: Simulation schematic in Proteus

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: Capture your schematic from Proteus and show in the report.
Report 2: Present your source code in the **HAL_TIM_PeriodElapsedCallback** function.
Ideas: Followings the thread,4 LEDS respectively display from left to right vaslue is

1;2;3;0

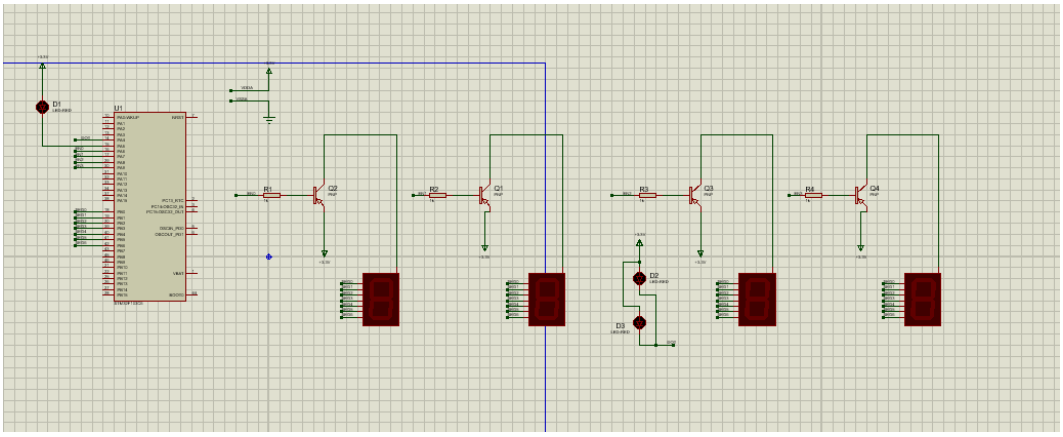


Figure 1.4: Schematic in Proteus

. Idead is simalar to previous exercise, create 2 global variables: counter with value is 50 (guaranties timer between LEDS is 500ms), index (selection position LEDs),but instead of 2 as exercise 1 we set 4 to display 4 LEDS. Moreover, to guaranty 2 leds connected to PA4, blinking every second, create a count_dot with value is 100 and decrease coun_dot 1 value **HAL_TIM_PeriodElapsedCallback**.

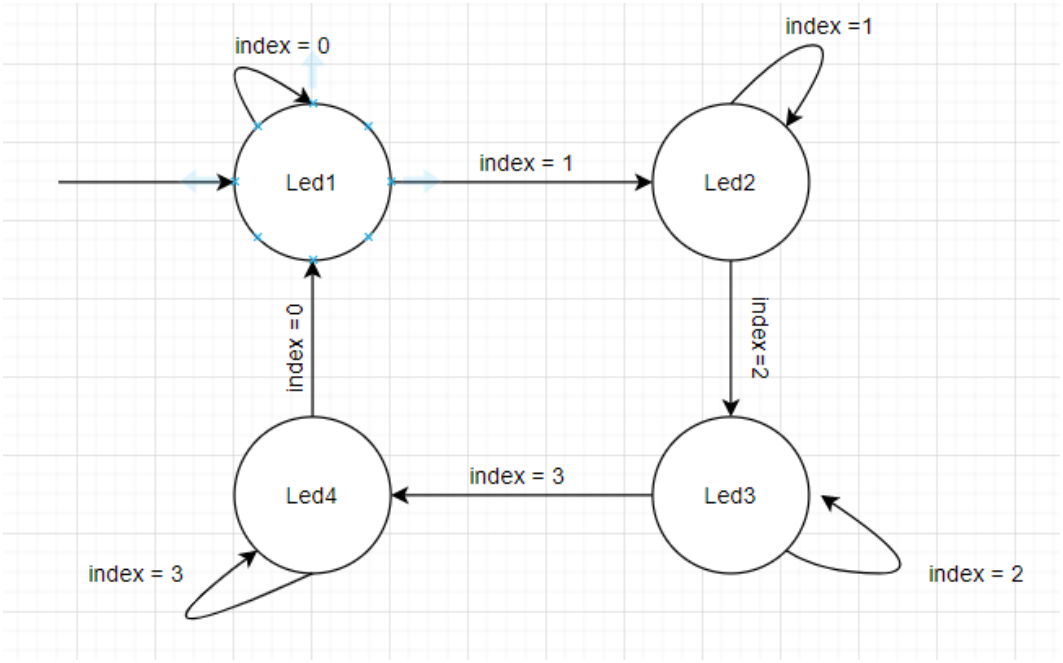


Figure 1.5: State of four LedsLeds

```
int counter  = 50;
int index = 0; // is check number position of leds
int count_dot = 100; // to discriminate  timer of dot and
   leds
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
   {
  counter--;
  count_dot --;
```

```
  if(counter <= 0){
    switch(index){
      case 0:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7
            |GPIO_PIN_8|GPIO_PIN_9, SET);
        display7SEG(1);
        index = 2;
        break;
      case 1:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_6
            |GPIO_PIN_8|GPIO_PIN_9, SET);
        display7SEG(2);
        index = 3;
        break;
      case 2:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, RESET);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7
            |GPIO_PIN_6|GPIO_PIN_9, SET);
        display7SEG(3);
        index = 4;
        break;
      case 3:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, RESET);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7
            |GPIO_PIN_8|GPIO_PIN_66, SET);
        display7SEG(0);
        index = 1;
        break;
        }
    counter = 50;
  }
  if(count_dot <= 0){
    HAL_GPIO_Toggle(GPIOA,GPIO_PIN_5);
    count_dot = 100;
  }
```

Program 1.2: Source code

Short question: What is the frequency of the scanning process?
Answer: The timer of the scanning process is 1s (timer 1 leds turn on is 0,5s, we have 4 ledsleds) so the Frequency of the scanning process is 1/2 = 0.55Hz.

2.3 Exercise 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 7SEG with led_buffer[0]
            break;
        case 1:
            //Display the second 7SEG with led_buffer[1]
            break;
        case 2:
            //Display the third 7SEG with led_buffer[2]
            break;
        case 3:
            //Display the forth 7SEG with led_buffer[3]
            break;
        default:
            break;
    }
}
```

Program 1.3: 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.

Report 1: Present the source code of the update7SEG function.
Ideas : In each case, we turn on one LED and turn off all LEDs that remain. After that, call display7SEG and display the value of position's led_buffer corresponding.

```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:
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7|GPIO_PIN_8|
                   GPIO_PIN_9, SET);
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
      break;
    case 1:
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6|GPIO_PIN_8|
                   GPIO_PIN_9, SET);
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
      break;
    case 2:
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7|GPIO_PIN_6|
                   GPIO_PIN_9, SET);
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, RESET);
```

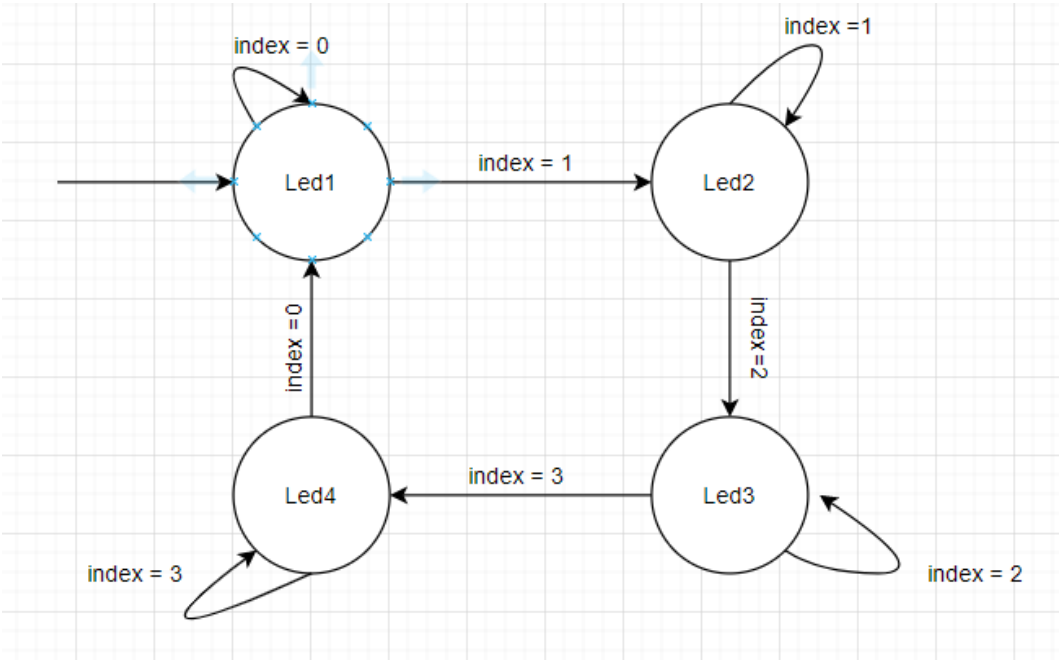


Figure 1.6: State of four Leds.

```
      break;
    case 3:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7|GPIO_PIN_8|
                        GPIO_PIN_6, SET);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, RESET);
      break;
    }
  display7SEG(led_buffer[index]);
}
```

Program 1.4: Source code of update7SEG

Report 2: Present the source code in the HAL_TIM_PeriodElapsedCallback.
Ideas: CCreate 2 global variables: counter with value is 50 (guaranties timer between LEDS is 500ms), index (selection position LEDs),but instead of 2 as exercise 1 we set 4 to display 4 LEDS. Moreover, to guaranty 2 leds connected to PA4, blinking every second, create a count_dot with value is 100 and decrease coun_dot 1 value **HAL_TIM_PeriodElapsedCallback**.

```
int counter  = 50;
int count_led = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
   {
  counter--;
  counter_led--;
  if(counter <= 0){
    if(index_led >= MAX_LED) index_led = 0;
    update7SEG(index_led++);
    counter = 50;
  }
  if(count_led <= 0){
```

```
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
    count_led = 100;
  }
}
```

Program 1.5: Source code of HAL_TIM_PeriodElapsedCallback

Students are proposed to change the values in the **led_buffer** array for unit test this function, which is used afterward.
int led_buffer [4] = {1,2,0,0};

2.4 Exercise 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
Ideas: Because the frequenccy of 4 seven segment LEDs to is 1Hz so timer between turn on and off of Led is 0.25s.Similar to code in Exercise 3, but when seven segment LED turn on, set counter = 25.

```
int counter  = 25;
int count_led = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
   {
  counter--;
  counter_led--;
  if(counter <= 0){
    if(index_led >= MAX_LED) index_led = 0;
    update7SEG(index_led++);
    counter = 25;
  }
  if(count_led <= 0){
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
    count_led = 100;
  }
}
```

Program 1.6: An example for your code

Report 1: Present the source code in the **HAL_TIM_PeriodElapsedCallback**.

2.5 Exercise 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:

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

```
        minute = 0;
        hour++;
    }
    if(hour >= 24){
        hour = 0;
    }
    updateClockBuffer();
    HAL_Delay(1000);
}
```

Program 1.7: An example for your 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: Present the source code in the **updateClockBuffer** function.
Ideas: led_buffer[0],led_buffer[1] hold tens, unit of hour, led_buffer[2],led_buffer[3] hold tens, unit of minute.

```
int hour = 15, minute = 8, second = 50;
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4];
void updateClockBuffer(){
  led_buffer[0] = hour / 10;
  led_buffer[1] = hour % 10;
  led_buffer[2] = minute / 10;
  led_buffer[3] = minute % 10;
}
```

Program 1.8: Source code of updateClockBuffer

```
int counter  = 25;
int count_dot = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
   {
  counter--;
  count_dot--;
  if(counter <= 0){
    if(index_led >= MAX_LED) index_led = 0;
    update7SEG(index_led++);
    counter = 25;
  }
  if(count_dot <= 0){
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
    count_dot= 100;
  }
}
```

Program 1.9: Source HAP_TIM_PeriodElapsedCallBack

2.6 Exercise 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 setTimer0 ( 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;
 }
}
```

Program 1.10: Software timer based timer interrupt

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
}
```

Program 1.11: 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) ;
 }
```

```
}
```

Program 1.12: Software timer based timer interrupt

Report 1: if in line 1 of the code above is miss, what happens after that and why?
If in line 1 of the code above is miss,led_red won't be turned off.Because after 10ms **timer_run** in **HAL_TIM_PeriodElapsedCallback** will be called, !(timer0_counter > 0) so timer0_flag = 0 then the Led can't be turn off in while.
Report 2: if in line 1 of the code above is changed to setTimer0(1), what happens after that and why?
If in line 1 of the code above is changed to setTimer0(1),timer0_counter = 1/10 = 0 (timer0_counter is type int). Because timer0_counter = 0 before first times call timer_run() so timer0_flag can't have value 1. Then, the Led won't be turned off.
Report 3: if in line 1 of the code above is changed to setTimer0(10), what is changed compared to 2 first questions and why?
If in line 1 of the code above is changed to setTimer0(10),timer0_counter = 1,If in line 1 of the code above is changed to setTimer0(10),timer0_counter = 1, so after the first times call function timer_run(), the timer0_flag will be set 1. Then, when we debug the led will be turned off after 10ms

2.7 Exercise 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: Present your source code in the while loop on main function. **Ideas:** Look like step 1-2 in exercise 6, we created a software timer.Use the timer in the main function by invoked setTimer0 function, then check for its flag (timer0_flag). When timer0_flag = 1, second will increase 1 and setTimer0(1000) to guaranty after 1s second will increase 1. Beside that, in front of while loop we should setTimer0(10) to guaranty timer0_flag = 1 in first loop.

```
setTimer0 (10) ;
  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(GPIOA, GPIO_PIN_4);
```

```
      setTimer0(1000);
    }
}
```

Program 1.13: source code

```
int counter = 25;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
   htim )
{
    timer_run ();
    if(counter <= 0){
    if(index_led >= MAX_LED) index_led = 0;
    update7SEG(index_led++);
    counter = 25;
  }
// YOUR OTHER CODE
}
```

Program 1.14: Software timer based timer interrupt

2.8 Exercise 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: Present your source code in the the main function. In the case more extra functions are used (e.g. the second software timer), present them in the report as well.

Ideas: We will be create 2 value timer1_counter = 0 and timer1_flag = 0 , then create a function **setTimer1(int duration)** with implementaition function similar **setTimer0**. Addition, when timer1_flag = 1, setTimer1(250) to guaranty the timer between 2 Leds is 0.25s. Beside that, in front of while loop we should setTimer1(10) to guaranty timer1_flag = 1 in first loop.

```
int timer0_counter = 0;
int timer0_flag = 0;
int timer1_counter = 0;
int timer1_flag = 0;
int TIMER_CYCLE = 10;
void setTimer0 ( int duration ){
  timer0_counter = duration / TIMER_CYCLE ;
  timer0_flag = 0;
}
void setTimer1 ( int duration ){
  timer1_counter = duration / TIMER_CYCLE ;
  timer1_flag = 0;
}
void timer_run (){
  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;
  }
}
```

Program 1.15: Software timer based timer interrupt

```
  setTimer0 (10);
  setTimer1(10);
  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(GPIOA, GPIO_PIN_4);
              setTimer0(1000);
            }
            if(timer1_flag == 1){
              if(index_led >= MAX_LED)index_led = 0;
              update7SEG(index_led++);
              setTimer1(250);
            }
    }
```

Program 1.16: Source code

2.9 Exercise 9:

Report 1: Present the schematic of your system by capturing the screen in Proteus.

Report 2: Implement the function, updateLEDMatrix(int index), which is similarly to 4 seven led segments.
Ideas: We must know Matrix Led first, each columns we can consider as a 7 Segment Led. For example, when we turn on COL, the state of each row Leds in this col will be depend on data input of ROW (0: turn on; 1: turn off).
In this exercise,each matrix_buffer will sequence performance state of each MATRIX_LED

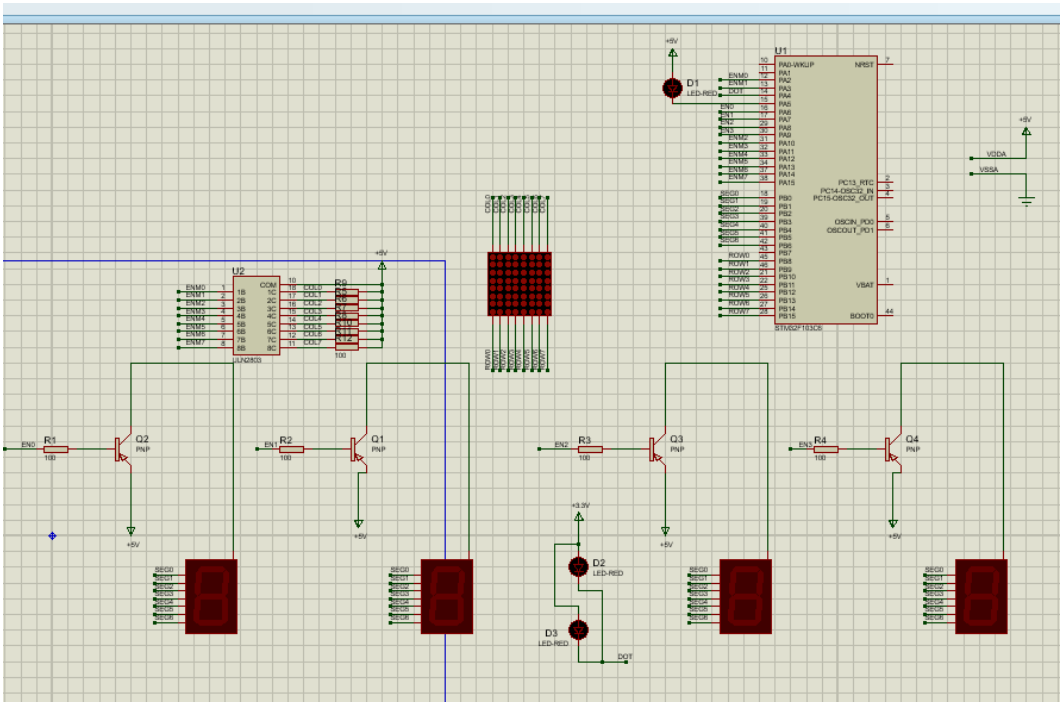


Figure 1.7: *Schematic.*

collums.**updateLEDMatrix** will control position collum will be turn on. We must to create a function **LEDMatrix(int index)**, to show the state of each row.

```
1  void LEDMatrix(int index){
2  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8,!((matrix_bufferB[
    index]>>0)&0x01));
3  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9,!((matrix_bufferB[
    index]>>1)&0x01));
4  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10,!((matrix_bufferB[
    index]>>2)&0x01));
5  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11,!((matrix_bufferB[
    index]>>3)&0x01));
6  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,!((matrix_bufferB[
    index]>>4)&0x01));
7  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13,!((matrix_bufferB[
    index]>>5)&0x01));
8  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14,!((matrix_bufferB[
    index]>>6)&0x01));
9  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15,!((matrix_bufferB[
    index]>>7)&0x01));
10 }
```

Program 1.17: source code of LEDMatrix

We can explaint **LEDMatrix**: the value of matrix_buffer is hex (8 bit) so corresponding each bit we have the input data of each Port

Beside that, we will to create a function **clearLEDMatrix** to guaranty when we call **updateLEDMatrix** just one collum turn on.
Corresponding index, we will turn off all LED and show the state Led in updateLEDMatix.

input	0x	0	0	0	0	0	0	0	1	0x01
		PB15	PB14	PB13	PB12	PB11	PB10	PB9	PB8	

Figure 1.8: Explaint for LEDMatrix function.

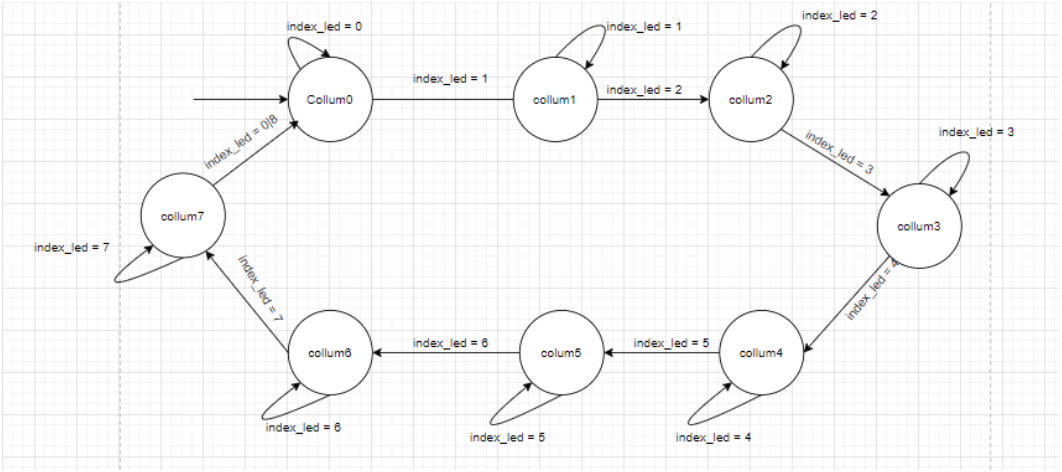


Figure 1.9: Schematic.

```
  const int MAX_LED_MATRIX = 8;
 int index_led_matrix = 0;
 uint8_t matrix_buffer [8] = {0x01 , 0x02 , 0x03 , 0x04 , 0
   x05 , 0
x06 , 0x07 , 0 x08 };
 void updateLEDMatrix (int index ){
  clearLEDMatrix();
 switch(index){
    case 0:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_2,RESET);
      break;
    case 1:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_3,RESET);
      break;
    case 2:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_10,RESET);
      break;
    case 3:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_11,RESET);
      break;
    case 4:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_12,RESET);
      break;
    case 5:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,RESET);
      break;
    case 6:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_14,RESET);
```

```
      break;
    case 7:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,RESET);
      break;
  }
  LEDMatrix(index);
}
```

Program 1.18: Function to display data on LED Matrix

```
while(1){
    if(index_led_matrix >= 8)index_led_matrix = 0;
    updateLEDMatrix(index_led_matrix++);
    HAL_Delay(1);
}
```

To display A we must know state off LEDs:

	PA2	PA3	PA10	PA11	PA12	PA13	PA14	PA15
PB8	0	0	0	1	1	0	0	0
PB9	0	0	1	0	0	1	0	0
PB10	0	0	1	0	0	1	0	0
PB11	0	0	1	0	0	1	0	0
PB12	0	0	1	1	1	1	0	0
PB13	0	0	1	0	0	1	0	0
PB14	0	0	1	0	0	1	0	0
PB15	0	0	1	0	0	1	0	0
HEX	0x00	0x00	0xFE	0x11	0x11	0xFE	0x00	0x00

Figure 1.10: Simulation the display "A".

Following the figure 1.9, we can change the value of matrix_buffer. uint8_t matrix_buffer [8] = {0x00 , 0x00 , 0xFE , 0x11 , 0x11 , 0xFE , 0x00 , 0x00 };

2.10 Exercise 10:

Create an animation on LED matrix, for example, the character is shifted to the left.
Report 1: Briefly describe your solution and present your source code in the report. **Ideas:** Move the position of each matrix_buffer's value 1 unit to simulation the animation on LED matrix (shifted to the left).
We create a function **animationLED(int index)** with index is a collumn we display, create a global variable count with value is 0 to shift left, create a global variable timer is 16.

```
int count  = 0;
int timer = 16; //timer shift left 1 value is 16ms
void animationLED(int index){
  if(count >= 8) count = 0;
  if(index + count >= 8){
    LEDMatrix(index+ count - 8);
  }
```

count = 0	0x00	0x00	0xFE	0x11	0x11	0xFE	0x00	0x00
count = 1	0x00	0xFF	0x11	0x11	0xFF	0x00	0x00	0x00
count = 2	0xFF	0x11	0x11	0xFF	0x00	0x00	0x00	0x00
count = 3	0x11	0x11	0xFF	0x00	0x00	0x00	0x00	0xFF
count = 4	0x11	0xFF	0x00	0x00	0x00	0x00	0xFF	0x11
count = 5	0xFF	0x00	0x00	0x00	0x00	0xFF	0x11	0x11
count = 6	0x00	0x00	0x00	0x00	0xFF	0x11	0x11	0xFF
count = 7	0x00	0x00	0x00	0xFF	0x11	0x11	0xFF	0x00
count = 0	0x00	0x00	0xFE	0x11	0x11	0xFE	0x00	0x00

Figure 1.11: Simulation animation the display "A".

```
  else {
    LEDMatrix(index +count);
  }
  if(timer == 0){
    timer = 16;
    count++;
  }
  timer--;
}
```

Program 1.19: source code

After that, replace LedMatrix(index) in updateLEDMatrix by animationLED(index).

```
void updateLEDMatrix (int index ){
  clearLEDMatrix();
  switch(index){
    case 0:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_2,RESET);
      break;
    case 1:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_3,RESET);
      break;
    case 2:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_10,RESET);
      break;
    case 3:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_11,RESET);
      break;
    case 4:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_12,RESET);
      break;
    case 5:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_13,RESET);
      break;
    case 6:
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_14,RESET);
      break;
    case 7:
```

```
26          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,RESET);
27          break;
28      }
29      //LEDMatrix(index);
30      animationLED(index);
31  }
```

Program 1.20: source code