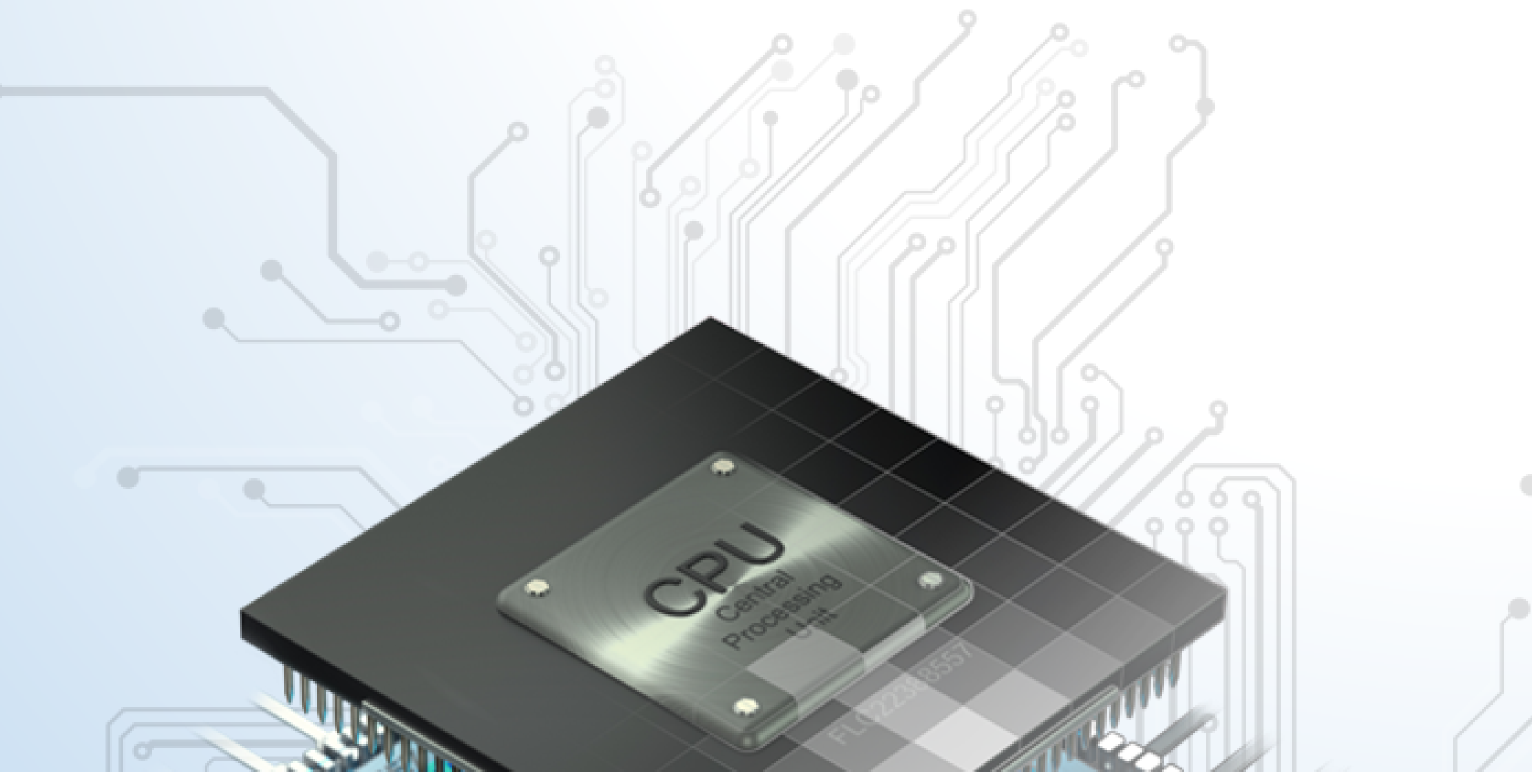
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

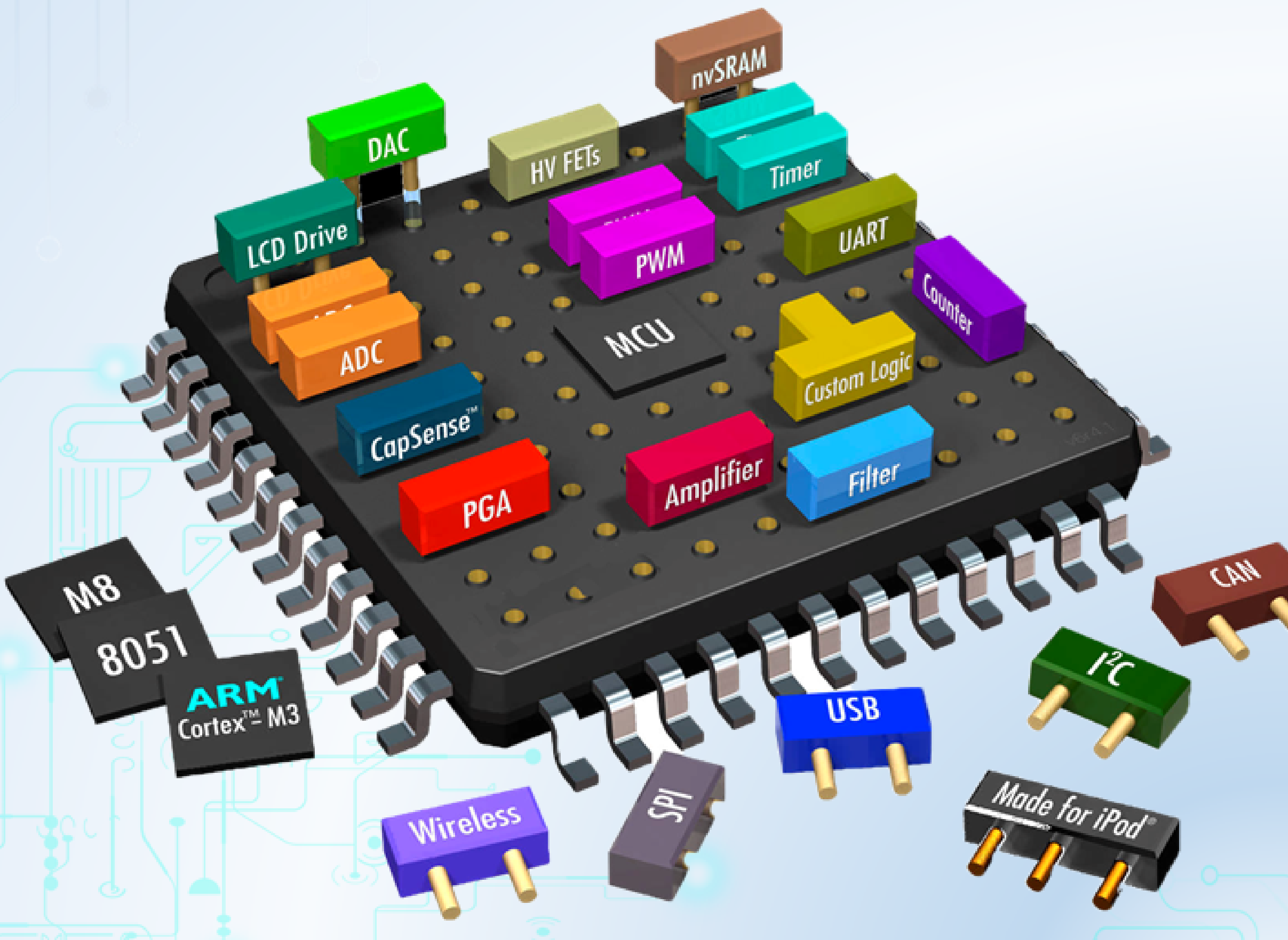


Dr. Le Trong Nhan

Mục lục

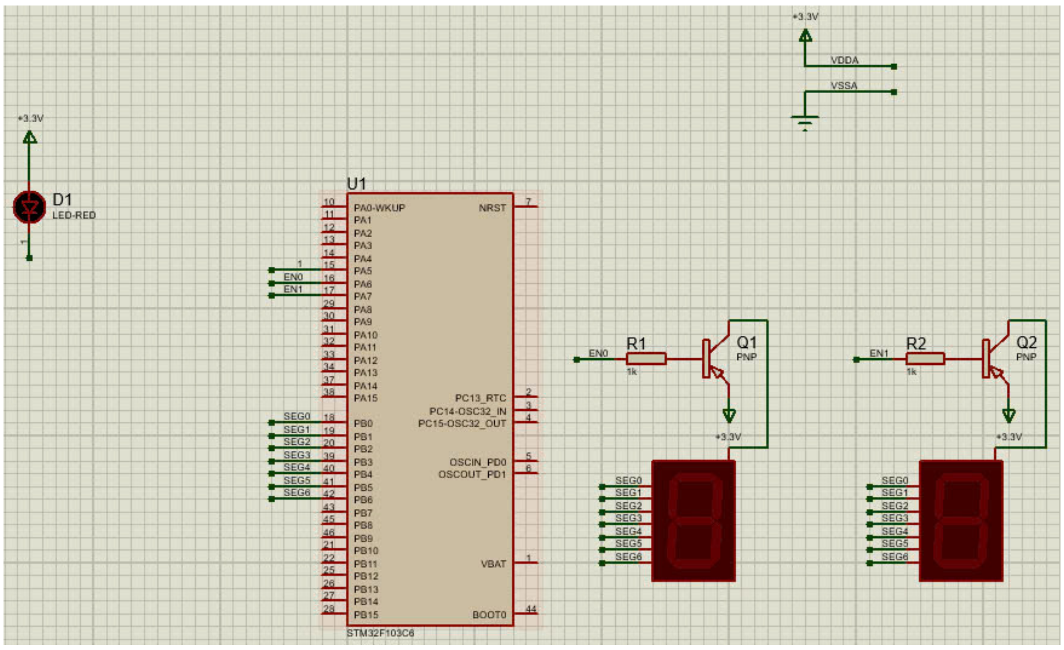
CHƯƠNG 1

Timer Interrupt and LED Scanning



1 Exercise and Report

1.1 Exercise 1



Hình 1.1: Schematic in Proteus

```c
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 500);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
      setTimer(0, 500);
      Ex1_run();
    }
    /* USER CODE END WHILE */

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

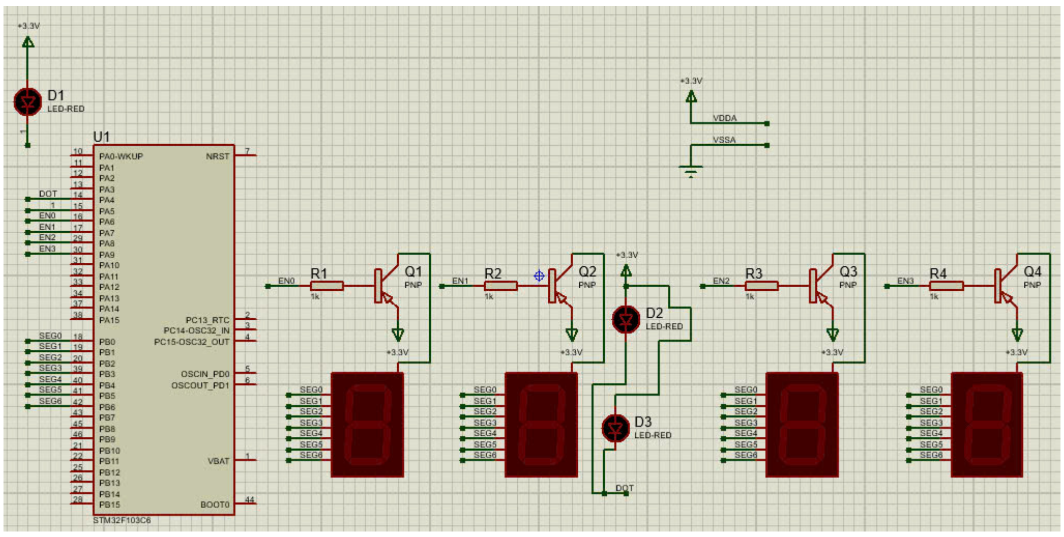
Program 1.1: main.c

```c
#include "software_timer1.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10

```

```c
int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
int index_led=0;

void display7SEG(int num) {
      const uint8_t segmentMap[10] = {
          0b11111100,
          0b01100000,
          0b11011010,
          0b11110010,
          0b01100110,
          0b10110110,
          0b10111110,
          0b11100000,
          0b11111110,
          0b11110110
      };
      HAL_GPIO_WritePin(SEG0_GPIO_Port, SEG0_Pin, (segmentMap[num] & 0b10000000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG1_Pin, (segmentMap[num] & 0b01000000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG2_GPIO_Port, SEG2_Pin, (segmentMap[num] & 0b00100000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG3_GPIO_Port, SEG3_Pin, (segmentMap[num] & 0b00010000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG4_GPIO_Port, SEG4_Pin, (segmentMap[num] & 0b00001000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG5_GPIO_Port, SEG5_Pin, (segmentMap[num] & 0b00000100) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG6_GPIO_Port, SEG6_Pin, (segmentMap[num] & 0b00000010) ? GPIO_PIN_RESET : GPIO_PIN_SET);
  }

void Ex1_run(){
  if(index_led>=2) index_led=0;
  index_led++;
  if(index_led<=1){
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    display7SEG(1);
  }
  if(index_led>=2){
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
    display7SEG(2);
  }
}

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}

int isTimerExpired(int index){
  if(timer_flag[index]==1){
    timer_flag[index]=0;
    return 1;
  }
  return 0;
}

void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
      if(timer_counter[i]>0){
      timer_counter[i]--;
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
    }
}
```
Program 1.2: software_timer1.c

Short question: Tần số của quá trình quét là $\frac{1}{2 \times 0.5} = 1$ Hz.

1.2 Exercise 2



Hình 1.2: Schematic in Proteus

```
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 500);
  setTimer(1, 1000);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
        setTimer(0, 500);
        Ex2_run();
      }
    if(isTimerExpired(1)==1){
      setTimer(1, 1000);
      led_blinky();
    }
    /* USER CODE END WHILE */

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

Program 1.3: main.c

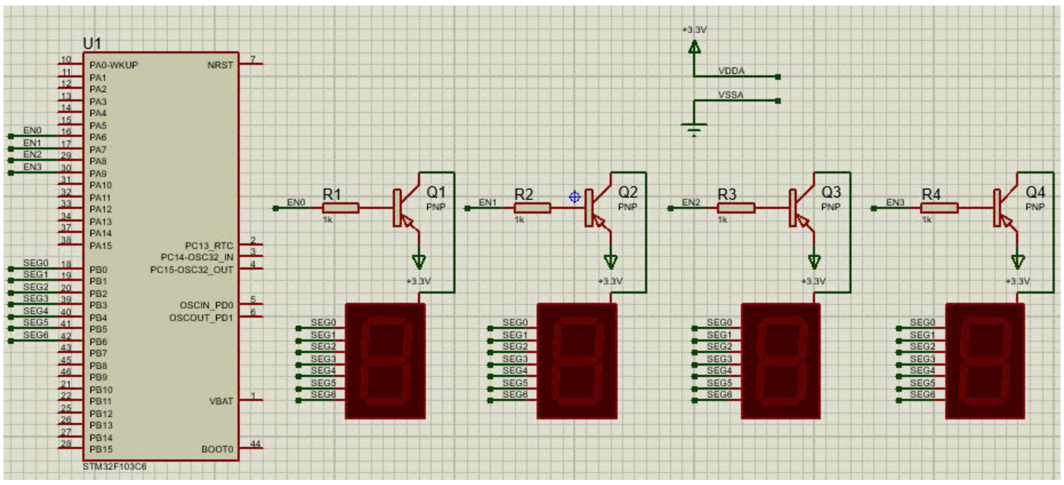
```
#include "software_timer2.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10

int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
int index_led=0;
int counter=0;

void display7SEG(int num) {
      const uint8_t segmentMap[10] = {
          0b11111100,
          0b01100000,
          0b11011010,
          0b11110010,
          0b01100110,
          0b10110110,
          0b10111110,
          0b11100000,
          0b11111110,
          0b11110110
      };
      HAL_GPIO_WritePin(SEG0_GPIO_Port, SEG0_Pin, (segmentMap[num] & 0b10000000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG1_Pin, (segmentMap[num] & 0b01000000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG2_GPIO_Port, SEG2_Pin, (segmentMap[num] & 0b00100000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG3_GPIO_Port, SEG3_Pin, (segmentMap[num] & 0b00010000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG4_GPIO_Port, SEG4_Pin, (segmentMap[num] & 0b00001000) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG5_GPIO_Port, SEG5_Pin, (segmentMap[num] & 0b00000100) ? GPIO_PIN_RESET : GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG6_GPIO_Port, SEG6_Pin, (segmentMap[num] & 0b00000010) ? GPIO_PIN_RESET : GPIO_PIN_SET);
  }

void Ex2_run(){
  if(index_led>=4) index_led=0;
  index_led++;
  if(index_led<=1){
      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);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
      display7SEG(1);
  }
  if(index_led>=2&&index_led<3){
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
        display7SEG(2);
  }
  if(index_led>=3&&index_led<4){
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
      display7SEG(3);
    }
  if(index_led>=4){
      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);
    }
}

void led_blinky(){
  if(counter>=2) counter=0;
  counter++;
  if(counter<=1) HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin ,
   RESET);
  else HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin , SET);
}

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}

int isTimerExpired(int index){
  if(timer_flag[index]==1){
    timer_flag[index]=0;
    return 1;
  }
  return 0;
}

void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
    if(timer_counter[i]>0){
      timer_counter[i]--;
```

```c
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
  }
}
```

Program 1.4: software_timer2.c

Short question: Tần số của quá trình quét là $\frac{1}{4\times 0.5} = 0.5$ Hz.

1.3 Exercise 3, 8



Hình 1.3: Schematic in Proteus

```c
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 500);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
        setTimer(0, 500);
        Ex3_run();
    }
    /* USER CODE END WHILE */

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

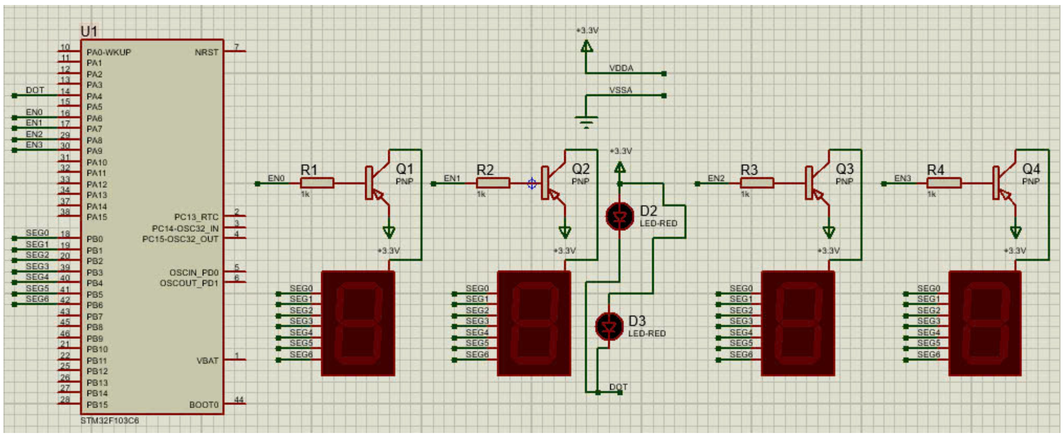
Program 1.5: main.c

```c
#include "software_timer3.h"

#define MAX_COUNTER 10
```

```c
#define TIMER_TICK 10

int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
const int MAX_LED = 4;
int index_led = 0;
int led_buffer [4] = {1 , 2 , 3 , 4};

void display7SEG(int num) {
       const uint8_t segmentMap[10] = {
           0b11111100,
           0b01100000,
           0b11011010,
           0b11110010,
           0b01100110,
           0b10110110,
           0b10111110,
           0b11100000,
           0b11111110,
           0b11110110
       };
       HAL_GPIO_WritePin(SEG0_GPIO_Port, SEG0_Pin, (
   segmentMap[num] & 0b10000000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG1_Pin, (
   segmentMap[num] & 0b01000000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG2_GPIO_Port, SEG2_Pin, (
   segmentMap[num] & 0b00100000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG3_GPIO_Port, SEG3_Pin, (
   segmentMap[num] & 0b00010000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG4_GPIO_Port, SEG4_Pin, (
   segmentMap[num] & 0b00001000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG5_GPIO_Port, SEG5_Pin, (
   segmentMap[num] & 0b00000100) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG6_GPIO_Port, SEG6_Pin, (
   segmentMap[num] & 0b00000010) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
   }

void update7SEG ( int index ) {
   switch ( index ) {
   case 0:
   // Display the first 7 SEG with led_buffer [0]
     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);
     HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
     display7SEG(led_buffer[index]);
   break ;
   case 1:
   // Display the second 7 SEG with led_buffer [1]
     HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
     HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
     HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
     HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
     display7SEG(led_buffer[index]);
   break ;
   case 2:
   // Display the third 7 SEG with led_buffer [2]
     HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
     HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
     HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
     HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
     display7SEG(led_buffer[index]);
   break ;
   case 3:
   // Display the forth 7 SEG with led_buffer [3]
     HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
     HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
     HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
     HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
     display7SEG(led_buffer[index]);
   break ;
   default :
     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);
     HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
   break ;
   }
 }

void Ex3_run(){
  if(index_led>=MAX_LED) index_led=0;
    update7SEG(index_led++);
}

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}

int isTimerExpired(int index){
  if(timer_flag[index]==1){
    timer_flag[index]=0;
    return 1;
  }
  return 0;
}

void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
    if(timer_counter[i]>0){
      timer_counter[i]--;
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
  }
}
```

Program 1.6: software_timer3.c

1.4 Exercise 4



Hình 1.4: Schematic in Proteus

```c
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 250);
  setTimer(1, 1000);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
        setTimer(0, 250);
        Ex4_run();
   }
```

```c
    if(isTimerExpired(1)==1){
        setTimer(1, 1000);
        led_blinky();
   }
    /* USER CODE END WHILE */

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

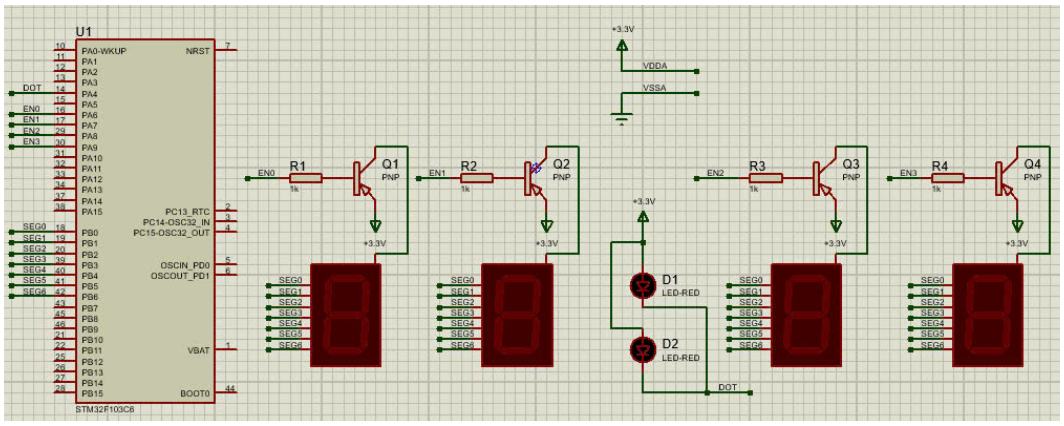
Program 1.7: main.c

```c
#include "software_timer4.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10

int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
int counter=0;
const int MAX_LED = 4;
int index_led = 0;
int led_buffer [4] = {1 , 2 , 3 , 4};

void display7SEG(int num) {
       const uint8_t segmentMap[10] = {
           0b11111100,
           0b01100000,
           0b11011010,
           0b11110010,
           0b01100110,
           0b10110110,
           0b10111110,
           0b11100000,
           0b11111110,
           0b11110110
       };
       HAL_GPIO_WritePin(SEG0_GPIO_Port, SEG0_Pin, (
   segmentMap[num] & 0b10000000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG1_Pin, (
   segmentMap[num] & 0b01000000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG2_GPIO_Port, SEG2_Pin, (
   segmentMap[num] & 0b00100000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG3_GPIO_Port, SEG3_Pin, (
   segmentMap[num] & 0b00010000) ? GPIO_PIN_RESET :
   GPIO_PIN_SET);
       HAL_GPIO_WritePin(SEG4_GPIO_Port, SEG4_Pin, (
   segmentMap[num] & 0b00001000) ? GPIO_PIN_RESET :
```

```
GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG5_GPIO_Port, SEG5_Pin, (
segmentMap[num] & 0b00000100) ? GPIO_PIN_RESET :
GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG6_GPIO_Port, SEG6_Pin, (
segmentMap[num] & 0b00000010) ? GPIO_PIN_RESET :
GPIO_PIN_SET);
    }

void update7SEG ( int index ) {
    switch ( index ) {
    case 0:
    // Display the first 7 SEG with led_buffer [0]
      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);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
      display7SEG(led_buffer[index]);
    break ;
    case 1:
    // Display the second 7 SEG with led_buffer [1]
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
      display7SEG(led_buffer[index]);
    break ;
    case 2:
    // Display the third 7 SEG with led_buffer [2]
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
      display7SEG(led_buffer[index]);
    break ;
    case 3:
    // Display the forth 7 SEG with led_buffer [3]
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
      display7SEG(led_buffer[index]);
    break ;
    default :
      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);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    break ;
    }
  }

void led_blinky(){
  if(counter>=2) counter=0;
  counter++;
  if(counter<=1) HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin ,
   RESET);
  else HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin , SET);
 }

void Ex4_run(){
  if(index_led>=MAX_LED) index_led=0;
    update7SEG(index_led++);
}

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}

int isTimerExpired(int index){
  if(timer_flag[index]==1){
    timer_flag[index]=0;
    return 1;
  }
  return 0;
}

void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
    if(timer_counter[i]>0){
      timer_counter[i]--;
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
  }
}
```

Program 1.8: software_timer4.c

1.5 Exercise 5, 7



Hình 1.5: Schematic in Proteus

```c
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 500);
  setTimer(1, 1000);
  setTimer(2, 1000);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
          setTimer(0, 500);
          scan_7LED();
    }
    if(isTimerExpired(1)==1){
          setTimer(1, 1000);
          Ex5_run();
    }
    if(isTimerExpired(2)==1){
          setTimer(2, 1000);
          led_blinky();
    }
    /* USER CODE END WHILE */

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

Program 1.9: main.c

```c
#include "software_timer5.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10
```

```

int hour = 0 , minute = 0 , second = 0;
int counter=0;
int index_led = 0;
int led_buffer [4] = {0,0,0,0};
int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];

void led_blinky(){
  if(counter>=2) counter=0;
  counter++;
  if(counter<=1) HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin ,
   RESET);
  else HAL_GPIO_WritePin(DOT_GPIO_Port,DOT_Pin , SET);
}

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

void Ex5_run(){
  second ++;
   if ( second >= 60) {
     second = 0;
     minute ++;
  }
  if( minute >= 60) {
     minute = 0;
     hour ++;
  }
  if( hour >=24) {
     hour = 0;
  }
  updateClockBuffer () ;
}

void display7SEG(int num) {
       const uint8_t segmentMap[10] = {
           0b11111100,
           0b01100000,
           0b11011010,
           0b11110010,
           0b01100110,
           0b10110110,
           0b10111110,
           0b11100000,
          0b11111110,
          0b11110110
      };
      HAL_GPIO_WritePin(SEG0_GPIO_Port, SEG0_Pin, (
  segmentMap[num] & 0b10000000) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG1_Pin, (
  segmentMap[num] & 0b01000000) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG2_GPIO_Port, SEG2_Pin, (
  segmentMap[num] & 0b00100000) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG3_GPIO_Port, SEG3_Pin, (
  segmentMap[num] & 0b00010000) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG4_GPIO_Port, SEG4_Pin, (
  segmentMap[num] & 0b00001000) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG5_GPIO_Port, SEG5_Pin, (
  segmentMap[num] & 0b00000100) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
      HAL_GPIO_WritePin(SEG6_GPIO_Port, SEG6_Pin, (
  segmentMap[num] & 0b00000010) ? GPIO_PIN_RESET :
  GPIO_PIN_SET);
  }
void update7SEG ( int index ) {
  switch ( index ) {
  case 0:
  // Display the first 7 SEG with led_buffer [0]
    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);
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    display7SEG(led_buffer[index]);
  break ;
  case 1:
  // Display the second 7 SEG with led_buffer [1]
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
    display7SEG(led_buffer[index]);
  break ;
  case 2:
  // Display the third 7 SEG with led_buffer [2]
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
```

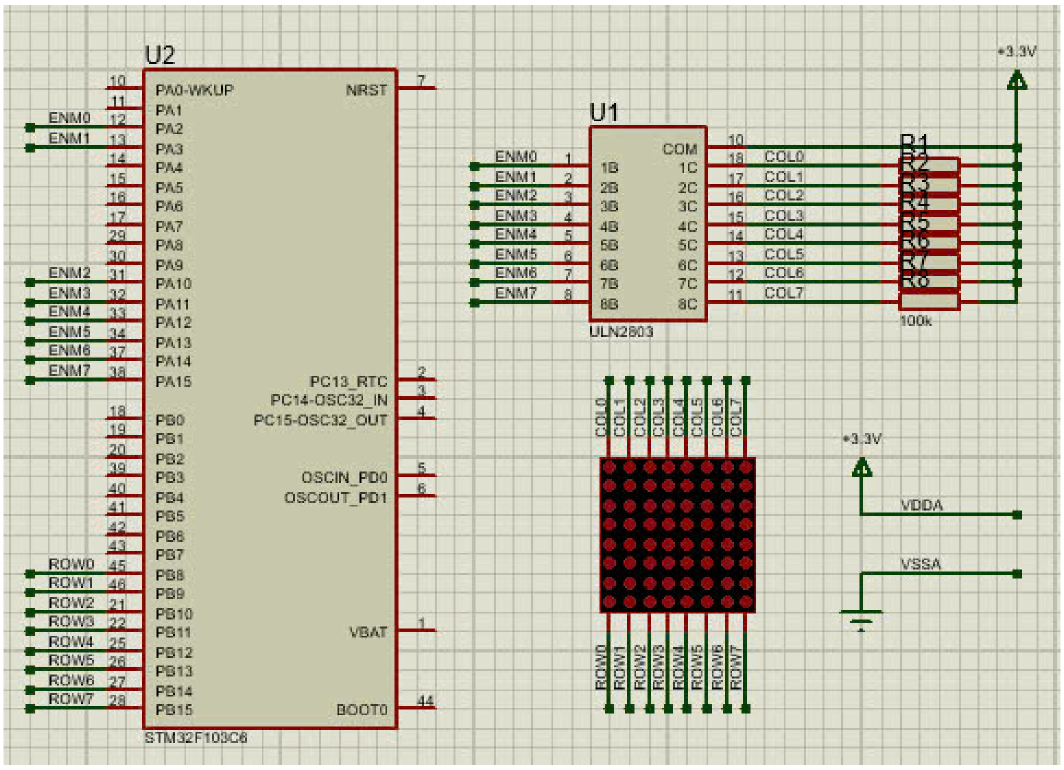
```c
      display7SEG(led_buffer[index]);
    break ;
    case 3:
    // Display the forth 7 SEG with led_buffer [3]
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
      display7SEG(led_buffer[index]);
    break ;
    default :
      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);
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    break ;
    }
 }

void scan_7LED(){
  if(index_led>=4) index_led=0;
      update7SEG(index_led++);
}

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}

int isTimerExpired(int index){
  if(timer_flag[index]==1){
        timer_flag[index]=0;
        return 1;
        }
  return 0;
}

void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
    if(timer_counter[i]>0){
      timer_counter[i]--;
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
  }
}
```

Program 1.10: software_timer5.c

1.6 Exercise 6

Nếu dòng 1 của đoạn mã trên bị miss thì chỉ còn 99 lần gọi hàm ngắt mỗi 10 ms do lần đầu chạy quá nhanh nó xảy ra ngắt liền, sai số về thời gian lúc này chỉ là $\frac{1}{100}$ = 1%.
Nếu dòng 1 của đoạn mã trên bị thay đổi thành setTimer0(1), thì vòng lặp while không gọi hàm ngắt mỗi 10 ms được vì sai số về thời gian lúc này lên đến là $\frac{1}{0.1}$ = 1000%.
Nếu dòng 1 của đoạn mã trên bị thay đổi thành setTimer0(10), thì vòng lặp while không gọi hàm ngắt mỗi 10 ms được vì sai số về thời gian lúc này lên đến là $\frac{1}{1}$ = 100% dẫn đến đèn led không chớp tắt được.

1.7 Exercise 9



Hình 1.6: LED matrix is added to the simulation

```
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 10);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
```

```c
    if(isTimerExpired(0)==1){
      setTimer(0, 10);
      Ex9_run();
    }
    /* USER CODE END WHILE */

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

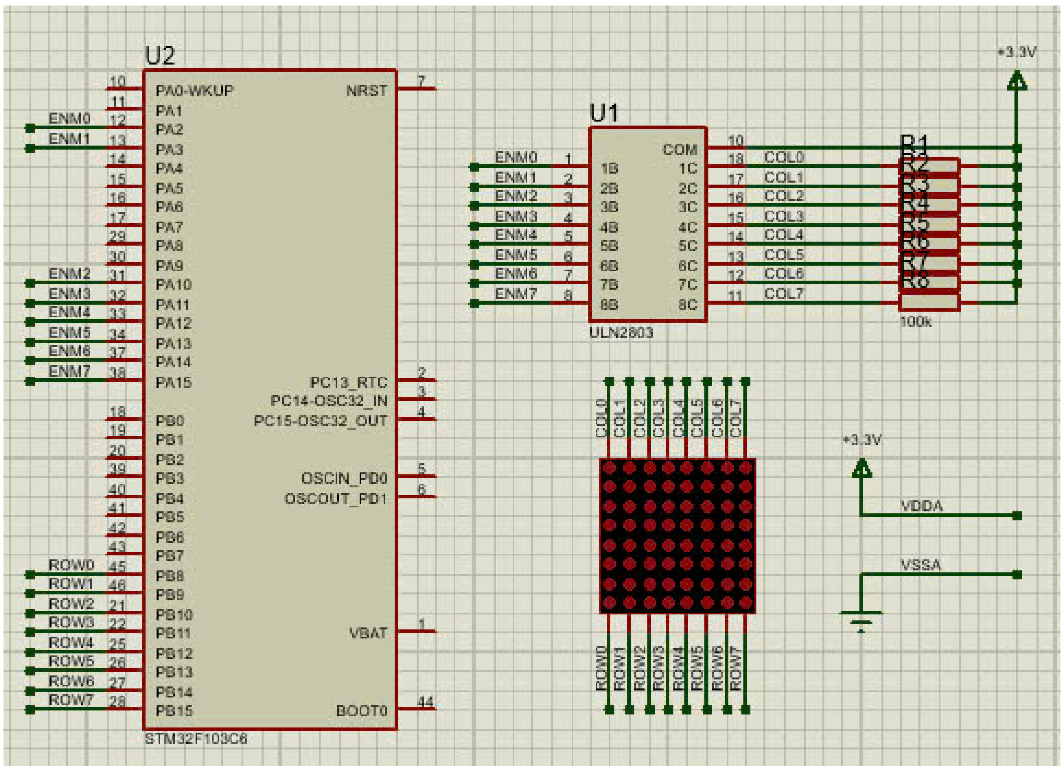
Program 1.11: main.c

```c
#include "software_timer9.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10

int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
const int MAX_LED_MATRIX = 8;
int index_led_matrix = 0;
uint8_t matrix_buffer [8] = {0xFF , 0xC0 , 0x80 , 0x33 , 0
   x33 , 0x80 , 0xC0 , 0xFF };
uint16_t segmentPins[8] = {ROW0_Pin,ROW1_Pin,ROW2_Pin,
   ROW3_Pin,ROW4_Pin,ROW5_Pin,ROW6_Pin,ROW7_Pin};

void displayMatrix(uint8_t num){
   for(int i=0;i<MAX_LED_MATRIX;i++){
     HAL_GPIO_WritePin(GPIOB, segmentPins[i], (num&(0x80>>i
  ))?SET:RESET);
   }
 }

void Ex9_run(){
   if(index_led_matrix>=8) index_led_matrix=0;
   updateLEDMatrix(index_led_matrix++);
 }

void updateLEDMatrix (int index ) {
   HAL_GPIO_WritePin(GPIOA, LED_Pins, SET);
     switch ( index ) {
     case 0:
       HAL_GPIO_WritePin(GPIOA, ENM0_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 1:
       HAL_GPIO_WritePin(GPIOA, ENM1_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 2:
       HAL_GPIO_WritePin(GPIOA, ENM2_Pin, RESET);
```

```
      displayMatrix(matrix_buffer[index]);
    break ;
    case 3:
      HAL_GPIO_WritePin(GPIOA, ENM3_Pin, RESET);
      displayMatrix(matrix_buffer[index]);
    break ;
    case 4:
      HAL_GPIO_WritePin(GPIOA, ENM4_Pin, RESET);
      displayMatrix(matrix_buffer[index]);
    break ;
    case 5:
      HAL_GPIO_WritePin(GPIOA, ENM5_Pin, RESET);
      displayMatrix(matrix_buffer[index]);
    break ;
    case 6:
      HAL_GPIO_WritePin(GPIOA, ENM6_Pin, RESET);
      displayMatrix(matrix_buffer[index]);
    break ;
    case 7:
      HAL_GPIO_WritePin(GPIOA, ENM7_Pin, RESET);
      displayMatrix(matrix_buffer[index]);
    break ;
    default :
      HAL_GPIO_WritePin(GPIOA, LED_Pins , SET);
    break ;
  }
 }

void setTimer(int index, int value){
  timer_counter[index]=value/TIMER_TICK;
  timer_flag[index]=0;
}
int isTimerExpired(int index){
  if(timer_flag[index]==1){
    timer_flag[index]=0;
    return 1;
  }
  return 0;
}
void timerRun(){
  for(int i=0;i<MAX_COUNTER;i++){
    if(timer_counter[i]>0){
      timer_counter[i]--;
      if(timer_counter[i]<=0) timer_flag[i]=1;
    }
  }
}
```

Program 1.12: software_timer9.c

1.8 Exercise 10



Hình 1.7: LED matrix is added to the simulation

```c
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  setTimer(0, 10);
  setTimer(1, 80);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(isTimerExpired(0)==1){
      setTimer(0, 10);
      Ex10_run();
    }
    if(isTimerExpired(1)==1){
        setTimer(1, 80);
//        shiftColLeft();
//        shiftColRight();
//        shiftRowUp();
//        shiftRowDown();
        }
    /* USER CODE END WHILE */

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

Program 1.13: main.c

```
#include "software_timer10.h"

#define MAX_COUNTER 10
#define TIMER_TICK 10

int timer_counter[MAX_COUNTER];
int timer_flag[MAX_COUNTER];
const int MAX_LED_MATRIX = 8;
int index_led_matrix = 0;
uint8_t matrix_buffer [8] = {0xFF , 0xC0 , 0x80 , 0x33 , 0
   x33 , 0x80 , 0xC0 , 0xFF };
uint16_t segmentPins[8] = {ROW0_Pin,ROW1_Pin,ROW2_Pin,
   ROW3_Pin,ROW4_Pin,ROW5_Pin,ROW6_Pin,ROW7_Pin};

void displayMatrix(uint8_t num){
   for(int i=0;i<MAX_LED_MATRIX;i++){
     HAL_GPIO_WritePin(GPIOB, segmentPins[i], (num&(0x80>>i
   ))?SET:RESET);
   }
 }

void shiftColLeft(){
   for(int i=0;i<MAX_LED_MATRIX-1;i++){
     matrix_buffer[i]=matrix_buffer[i+1];
   }
   matrix_buffer[7]=matrix_buffer[0];
 }

void shiftColRight(){
   for(int i=MAX_LED_MATRIX-1;i>0;i--){
     matrix_buffer[i]=matrix_buffer[i-1];
   }
   matrix_buffer[0]=matrix_buffer[7];
  }

void shiftRowUp(){
   for (int i = 0; i < MAX_LED_MATRIX; i++) {
           uint8_t temp = (matrix_buffer[i] & 0x80) ? 1 :
   0;
           matrix_buffer[i] <<= 1;
           matrix_buffer[i] |= temp;
       }
   }

void shiftRowDown(){
   for (int i = 0; i < MAX_LED_MATRIX; i++) {
    uint8_t temp = (matrix_buffer[i] & 0x01) ? 0x80 : 0;
           matrix_buffer[i] >>= 1;
           matrix_buffer[i] |= temp;
       }
    }

void Ex10_run(){
   if(index_led_matrix>=8) index_led_matrix=0;
   updateLEDMatrix(index_led_matrix++);
 }

void updateLEDMatrix (int index ) {
   HAL_GPIO_WritePin(GPIOA, LED_Pins, SET);
     switch ( index ) {
     case 0:
       HAL_GPIO_WritePin(GPIOA, ENM0_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 1:
       HAL_GPIO_WritePin(GPIOA, ENM1_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 2:
       HAL_GPIO_WritePin(GPIOA, ENM2_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 3:
       HAL_GPIO_WritePin(GPIOA, ENM3_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 4:
       HAL_GPIO_WritePin(GPIOA, ENM4_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 5:
       HAL_GPIO_WritePin(GPIOA, ENM5_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 6:
       HAL_GPIO_WritePin(GPIOA, ENM6_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     case 7:
       HAL_GPIO_WritePin(GPIOA, ENM7_Pin, RESET);
       displayMatrix(matrix_buffer[index]);
     break ;
     default :
       HAL_GPIO_WritePin(GPIOA, LED_Pins , SET);
     break ;
    }
  }

void setTimer(int index, int value){
    timer_counter[index]=value/TIMER_TICK;
    timer_flag[index]=0;
}

int isTimerExpired(int index){
    if(timer_flag[index]==1){
        timer_flag[index]=0;
        return 1;
    }
    return 0;
}

void timerRun(){
    for(int i=0;i<MAX_COUNTER;i++){
        if(timer_counter[i]>0){
            timer_counter[i]--;
            if(timer_counter[i]<=0) timer_flag[i]=1;
        }
    }
}
```

Program 1.14: software_timer10.c

Trong bài này các hàm trong vòng lặp while lần lượt dùng để dịch ký tự "A" sang trái, phải so với cột ma trận, dịch lên, xuống so với hàng ma trận, cứ mỗi lần tất cả các cột đều được quét qua thì lại cập nhập giá trị mới của mỗi cột và cứ lặp lại như thế .