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

Vi xử lý - Vi điều khiển (TN) (CO3010)

LAP 2

Microprocessor- Microcontroller

GVHD: TÔN HUỲNH LONG
SV thực hiện: BÙI NGUYỄN NHẬT TIẾN 2213444

TP Hồ Chí Minh, Tháng 10 Năm 2025

Mục lục

1 Exercise **2**
1.1 Exercise 1 . . . 3
1.1.1 Report 1 . . . 3
1.1.2 Report 2 . . . 3
1.2 Exercise 2 . . . 3
1.2.1 Report 1 . . . 3
1.2.2 Report 2 . . . 4
1.3 Exercise 3 . . . 4
1.3.1 Report 1 — Function `update7SEG` . . . 4
1.3.2 Report 2 — Timer Callback . . . 5
1.4 Exercise 4 . . . 6
1.4.1 Report 1 . . . 6
1.5 Exercise 5 . . . 6
1.5.1 Report . . . 6
1.6 Exercise 6 . . . 6
1.6.1 Report 1 . . . 6
1.6.2 Report 2 . . . 7
1.6.3 Report 3 . . . 7
1.7 Exercise 7 . . . 7
1.7.1 Report . . . 7
1.8 Exercise 8 . . . 8
1.8.1 Report . . . 8
1.9 Exercise 9 . . . 10
1.9.1 Report 1 . . . 10
1.9.2 Report 2 . . . 10

2 Exercise 10 **11**

1 Exercise

Liên kết GitHub chứa sơ đồ và mã nguồn cho các bài thực hành: https://github.com/NhatTiens/vxl_lap2/tree/main/Lab_2

Dưới đây là sơ đồ mạch cho các bài tập:

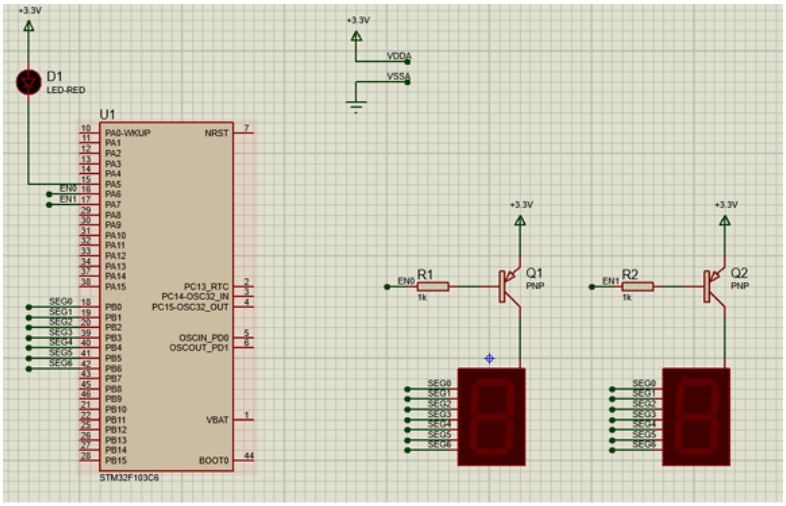

Figure 1: Figure 1: EX1

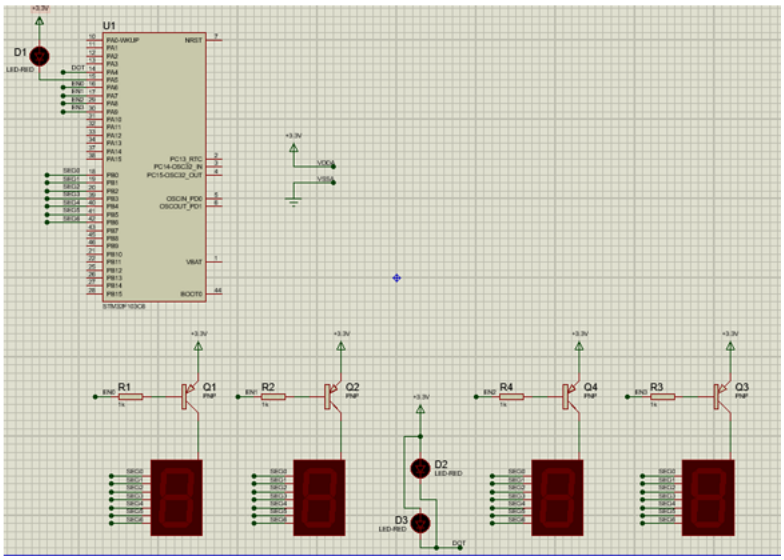

Figure 2: Figure 2: EX2

1.1 Exercise 1

1.1.1 Report 1

Nội dung nằm trong phần EX1.

1.1.2 Report 2

```
int counter = 100;

void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
{
    if (counter == 100){
        display7SEG(1);
        HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
        HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
    } else if (counter == 50){
        display7SEG(2);
        HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
        HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
    }
    counter = (counter > 0) ? counter - 1 : 100;
}
```

Giải thích: Dựa trên mô tả, việc chuyển đổi giữa hai LED hiển thị được thực hiện sao cho **2 LED hoạt động luân phiên trong 0,5 giây**.
Do đó, **mỗi LED được quét một lần trong nửa khoảng thời gian này**.

Tính tần số:

- Tổng thời gian cho 2 LED: **0,5 giây**
- Thời gian cho mỗi LED: **0,5 / 2 = 0,25 giây (250 ms)**
- Tần số f là nghịch đảo của chu kỳ T:

$$f = \frac{1}{T} = \frac{1}{0.25} = 4\,\text{Hz}$$

1.2 Exercise 2

1.2.1 Report 1

Nội dung tham khảo từ phần 2 của báo cáo gốc.

1.2.2 Report 2

```
const unsigned int Default = 200;
int counter = Default;
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
{
    if (counter == Default){
        display7SEG(1);
        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);
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
    } else if (counter == (Default *3/4)){
        display7SEG(2);
        HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
        HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
    } else if (counter == (Default *2/4)){
        display7SEG(3);
        HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, RESET);
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
    } else if (counter == (Default *1/4)){
        display7SEG(0);
        HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, RESET);
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
    }
    counter = (counter > 0) ? counter - 1 : Default;

    if (dot_counter >= Default) {
        dot_counter = 0;
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
    }
    dot_counter++;
}
```

Giải thích: Theo mô tả, **thời gian quét cho mỗi LED 7 đoạn là 0,5 giây (500 ms)**. Vì có **4 LED hiển thị**, nên tổng thời gian để quét qua tất cả là:

$$T = 4 \times 0.5\,\text{giây} = 2\,\text{giây}$$

Tần số quét f là nghịch đảo của chu kỳ T:

$$f = \frac{1}{T} = \frac{1}{2} = 0.5\,\text{Hz}$$

1.3 Exercise 3

1.3.1 Report 1 — Function `update7SEG`

```
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:
            display7SEG(led_buffer[0]);
            HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
            break;
        case 1:
            display7SEG(led_buffer[1]);
            HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
            break;
        case 2:
            display7SEG(led_buffer[2]);
            HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, RESET);
            break;
        case 3:
            display7SEG(led_buffer[3]);
            HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, RESET);
            break;
        default:
            break;
    }
}
```

1.3.2 Report 2 — Timer Callback

```
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
{
    if (counter == Default){
        update7SEG(index_led++);
    } else if (counter == (Default*3/4)){
        update7SEG(index_led++);
    } else if (counter == (Default*2/4)){
        update7SEG(index_led++);
    } else if (counter == (Default*1/4)){
        update7SEG(index_led++);
    }

    index_led = (index_led < MAX_LED) ? index_led : 0;
    counter = (counter > 0) ? counter - 1 : Default;

    if (dot_counter >= Default) {
        dot_counter = 0;
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
    }
    dot_counter++;
```

```
}
```

1.4 Exercise 4

1.4.1 Report 1

```
const unsigned int Default = 100;
int counter = Default;
int dot_counter = 0;

void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
{
    if (counter == Default) {
        update7SEG(index_led++);
        index_led = (index_led < MAX_LED) ? index_led : 0;
    } else if (counter == (Default * 3 / 4)) {
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
    } else if (counter == (Default * 1 / 4)) {
        HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
    }

    // Reduce counter
    counter = (counter > 0) ? counter - 1 : Default;

    if (dot_counter >= Default) {
        dot_counter = 0;
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
    }
    dot_counter++;
}
```

1.5 Exercise 5

1.5.1 Report

Code trong hàm `updateClockBuffer`:

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

1.6 Exercise 6

1.6.1 Report 1

Nếu dòng lệnh `setTimer0(1000)` bị thiếu, bộ định thời (timer) sẽ không được khởi tạo và giá trị của `timer0_counter` sẽ giữ nguyên ở 0 (hoặc giá trị trước đó nếu có).

Điều này có nghĩa là `timer0_flag` sẽ không bao giờ được đặt lại để điều khiển khoảng thời gian giữa các lần LED nhấp nháy.
Kết quả là LED sẽ không nhấp nháy vì điều kiện `if(timer0_flag == 1)` sẽ không bao giờ đúng, do `timer0_flag` không được kích hoạt thông qua việc đếm ngược của bộ định thời.

1.6.2 Report 2

Nếu dùng `setTimer0(1)` thay vì `setTimer0(1000)`, bộ định thời phần mềm sẽ được thiết lập với độ trễ rất ngắn, chỉ chờ **1 ms** (thay vì **1000 ms**) trước khi đặt `timer0_flag = 1`.
Do đó, LED sẽ nhấp nháy nhanh hơn rất nhiều, gần như ngay lập tức sau mỗi vòng lặp.
Về cơ bản, LED sẽ nhấp nháy ở **tần số rất cao**, khiến mắt người không thể phân biệt được từng lần nhấp nháy, và LED sẽ **trông như sáng liên tục**.

1.6.3 Report 3

Khi sử dụng `setTimer0(10)` thay vì `setTimer0(1000)`:

1. **So với `setTimer0(1000)`:** Bộ định thời sẽ được đặt ở 10 ms thay vì 1000 ms.
   Điều này có nghĩa là LED sẽ nhấp nháy nhanh hơn rất nhiều.
   Thay vì chờ 1 giây (1000 ms) mới nhấp nháy, giờ chỉ cần 10 ms.
   Kết quả là LED sẽ nhấp nháy nhanh hơn khoảng **100 lần**, có thể quá nhanh để mắt người nhận ra.

2. **So với `setTimer0(1)`:** Sự khác biệt ít đáng kể hơn.
   Thay vì chỉ 1 ms, bây giờ LED sẽ đợi 10 ms.
   LED vẫn nhấp nháy nhanh, nhưng chậm hơn khoảng **10 lần** so với trường hợp 1 ms.
   Mặc dù vậy, tốc độ này vẫn quá nhanh để người dùng có thể thấy rõ từng lần nhấp nháy riêng lẻ.
   Trong cả hai trường hợp, đèn LED sẽ nhấp nháy nhanh hơn nhiều so với độ trễ ban đầu 1000ms, nhưng setTimer0(10) cung cấp tốc độ vừa phải hơn so với setTimer0(1).

1.7 Exercise 7

1.7.1 Report

```
HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
int hour = 15, minute = 59, second = 59;
setTimer0(1000);
int duration = 1000;
updateClockBuffer(hour, minute);
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if (timer0_flag == 1){
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
        setTimer0(2000);
        second++;
```

```
        if (second >= 60) {
            second = 0;
            minute++;
        }
        if (minute >= 60) {
            minute = 0;
            hour++;
        }
        if (hour >= 24){
            hour = 0;
        }
        updateClockBuffer(hour, minute);
        setTimer0(duration);
    }
}
```

Giải thích: Đoạn mã trên mô phỏng hoạt động của một đồng hồ điện tử hiển thị giờ và phút bằng 4 LED 7 đoạn.
- Các biến `hour`, `minute`, và `second` lưu trữ giá trị thời gian hiện tại. - Bộ định thời `Timer0` được khởi tạo với chu kỳ 1 giây (`setTimer0(1000)`). - Mỗi khi `timer0_flag == 1`, chương trình sẽ:

- Đảo trạng thái LED (bật/tắt) để tạo hiệu ứng nhấp nháy.
- Cập nhật thời gian (giây, phút, giờ).
- Gọi hàm `updateClockBuffer()` để hiển thị thời gian mới lên LED 7 đoạn.

Cấu trúc này giúp LED hiển thị đồng hồ chạy chính xác theo thời gian thực, và dấu chấm giữa LED sẽ nhấp nháy định kỳ để biểu thị giây.

1.8 Exercise 8

1.8.1 Report

Code of two software timers:

```
int timer0_counter = 0;
int timer0_flag = 1;
int TIMER_CYCLE = 10;
int timer1_counter = 0;
int timer1_flag = 1;

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;
    }
}
```

Main loop:

```
HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
int hour = 15, minute = 59, second = 59;
setTimer0(1000);
int duration = 1000;
updateClockBuffer(hour, minute);
while (1)
{
    if (timer0_flag == 1){
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
        second++;
        if (second >= 60) { second = 0; minute++; }
        if (minute >= 60) { minute = 0; hour++; }
        if (hour >= 24) { hour = 0; }
        updateClockBuffer(hour, minute);
        setTimer0(duration);
    }
    if (timer1_flag == 1){
        update7SEG(index_led++);
        index_led = (index_led < MAX_LED) ? index_led : 0;
        setTimer1(duration / MAX_LED);
    }
}
```

Giải thích: Trong bài này, chương trình sử dụng **hai bộ định thời phần mềm (software timers)** để xử lý hai tác vụ song song:

- **Timer0:** Điều khiển việc cập nhật thời gian (giờ, phút, giây) và nhấp nháy LED.
- **Timer1:** Điều khiển quét hiển thị cho 4 LED 7 đoạn.

Các bước hoạt động:

1. `setTimer0(1000)` khởi tạo Timer0 với chu kỳ 1 giây. Khi hết thời gian, cờ `timer0_flag` được đặt lên 1.
2. Mỗi khi `timer0_flag == 1`, chương trình sẽ:
    - Đảo trạng thái LED (hiệu ứng nhấp nháy).
    - Cập nhật thời gian và hiển thị lại bằng `updateClockBuffer()`.

3. `Timer1` chịu trách nhiệm quét lần lượt từng LED 7 đoạn trong mảng hiển thị, giúp các LED sáng liên tục nhờ tốc độ quét nhanh.

Việc tách riêng hai bộ định thời giúp chương trình xử lý **nhiều tác vụ song song một cách hiệu quả**, đảm bảo LED hiển thị mượt mà mà không ảnh hưởng đến việc cập nhật thời gian.

1.9 Exercise 9

1.9.1 Report 1

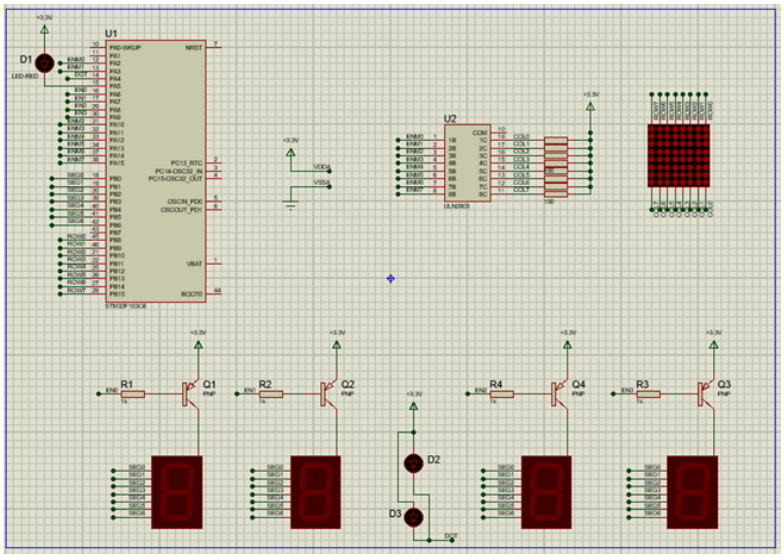

Figure 3: Exercise 9 circuit diagram

1.9.2 Report 2

Code in the `updateLEDMatrix` function:

```
void updateLEDMatrix(int index){
    switch (index){
        case 0:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 1:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 2:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 3:
```

```
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 4:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 5:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 6:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        case 7:
            MatrixColLed(index);
            displayLEDMatrix(index);
            break;
        default:
            break;
    }
}
```

Giải thích: Trong bài này, chương trình thực hiện việc **điều khiển ma trận LED 8x8** bằng phương pháp **quét cột (column scanning)**. Mỗi cột của ma trận LED được kích hoạt lần lượt thông qua hàm `updateLEDMatrix()`, giúp hiển thị dữ liệu từ mảng đệm hiển thị `matrix_buffer[]`.

Cách hoạt động:

- Ma trận LED gồm 8 cột và 8 hàng, tương ứng 64 điểm sáng (LED).
- Hàm `MatrixColLed(index)` bật cột LED tương ứng với giá trị `index`.
- Hàm `displayLEDMatrix(index)` quyết định hàng nào trong cột được bật để tạo hình ảnh mong muốn.
- Quá trình quét diễn ra rất nhanh (thường vài ms mỗi cột), nhờ đó mắt người thấy toàn bộ hình ảnh hiển thị ổn định.

Kết luận: Kỹ thuật quét LED ma trận giúp giảm số lượng chân điều khiển cần thiết và cho phép hiển thị nhiều mẫu hoặc ký tự khác nhau một cách linh hoạt. Bằng cách thay đổi nội dung mảng `matrix_buffer`, ta có thể tạo ra hiệu ứng cuộn chữ, hiển thị giờ hoặc các hình động trên ma trận LED.

2 Exercise 10

1. Timer variable setup:

```
int timer0_counter = 0;
int timer0_flag = 1;
```

```
int timer1_counter = 0;
int timer1_flag = 1;
int timer2_counter = 0;
int timer2_flag = 1;
int TIMER_CYCLE = 10;
```

Giải thích: Các biến này được sử dụng để quản lý chức năng định thời của chương trình. Mỗi bộ định thời (timer) có:

- Một bộ đếm (`counter`) để theo dõi thời gian đã trôi qua.
- Một cờ (`flag`) để xác định khi nào bộ định thời hết hạn.

TIMER_CYCLE: Là hằng số xác định khoảng thời gian của mỗi chu kỳ đếm (tính bằng mili giây). Giá trị được đặt là 10 ms, nghĩa là mỗi bộ định thời sẽ đếm giảm dần theo bước 10 ms.

2. Timer setup function:

```
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 setTimer2(int duration) {
    timer2_counter = duration / TIMER_CYCLE;
    timer2_flag = 0;
}
```

Giải thích: Các hàm này được sử dụng để khởi tạo bộ định thời với một khoảng thời gian cụ thể. Mỗi hàm nhận tham số là `duration` (tính bằng mili giây), sau đó tính toán số chu kỳ tương ứng với `TIMER_CYCLE`. Hàm đặt giá trị đếm cho bộ định thời và đặt lại cờ báo hiệu, chỉ ra rằng bộ định thời đang hoạt động.

3. Timer running:

```
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;
    }
    if (timer2_counter > 0) {
```

```
        timer2_counter--;
        if (timer2_counter == 0) timer2_flag = 1;
    }
}
```

Giải thích: Hàm này được gọi định kỳ để giảm giá trị đếm của từng bộ định thời. Nếu giá trị `counter` của bộ định thời lớn hơn 0, nó sẽ giảm dần. Khi `counter` giảm về 0, cờ tương ứng (`flag`) được đặt thành 1 để báo hiệu bộ định thời đã hết hạn — cho phép chương trình thực hiện hành động tiếp theo.

4. LED Matrix array:

```
const uint8_t MatrixCol[8] = {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f};
GPIO_TypeDef * RowPort[8] = {Row0_GPIO_Port, Row1_GPIO_Port, Row2_GPIO_Port,
                             Row3_GPIO_Port, Row4_GPIO_Port, Row5_GPIO_Port,
                             Row6_GPIO_Port, Row7_GPIO_Port};
const uint16_t RowPin[8] = {Row0_Pin, Row1_Pin, Row2_Pin, Row3_Pin, Row4_Pin,
                            Row5_Pin, Row6_Pin, Row7_Pin};
GPIO_TypeDef * ColPort[8] = {Enm0_GPIO_Port, Enm1_GPIO_Port, Enm2_GPIO_Port,
                             Enm3_GPIO_Port, Enm4_GPIO_Port, Enm5_GPIO_Port,
                             Enm6_GPIO_Port, Enm7_GPIO_Port};
const uint16_t ColPin[8] = {Enm0_Pin, Enm1_Pin, Enm2_Pin, Enm3_Pin, Enm4_Pin,
                            Enm5_Pin, Enm6_Pin, Enm7_Pin};
const int MAX_LED_MATRIX = 8;
```

Giải thích:

- `MatrixCol[]` xác định trạng thái cho từng cột trong ma trận LED (mỗi phần tử điều khiển một cột).
- `RowPort[]` và `ColPort[]` ánh xạ tới các cổng GPIO tương ứng cho hàng và cột.
- `RowPin[]` và `ColPin[]` chứa các chân điều khiển cụ thể cho từng hàng và cột.
- `MAX_LED_MATRIX` đại diện cho số lượng tối đa các LED (hoặc cột) trong ma trận.

5. Update LED Matrix:

```
void displayLEDMatrix(int num) {
    for (int i = 0; i < MAX_LED_MATRIX; i++) {
        HAL_GPIO_WritePin(RowPort[i], RowPin[i], (matrix_buffer[num] >> i) & 0x01);
    }
}
```

Giải thích: Hàm này chịu trách nhiệm cập nhật ma trận LED để hiển thị dữ liệu hiện tại dựa trên chỉ số `num`. Mỗi bit trong `matrix_buffer[num]` quyết định trạng thái của LED ở hàng tương ứng (bật hoặc tắt).

6. Update Matrix buffer:

```
void updateMatrixBuffer() {
    matrix_buffer[7] = matrix_buffer[0];
    for (int i = 0; i < 7; i++) {
        matrix_buffer[i] = matrix_buffer[i + 1];
    }
}
```

Giải thích: Phần tử cuối cùng của `matrix_buffer` được gán bằng phần tử đầu tiên, sau đó mỗi phần tử được dịch sang trái một vị trí. Điều này tạo ra hiệu ứng ký tự hiển thị di chuyển sang trái trên ma trận LED.

7. Main loop:

```
while (1) {
    // USER CODE END WHILE
    // USER CODE BEGIN 3
    if (timer0_flag == 1) {
        updateMatrixBuffer();
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
        second++;
        if (second >= 60) { second = 0; minute++; }
        if (minute >= 60) { minute = 0; hour++; }
        if (hour >= 24) { hour = 0; }
        updateClockBuffer(hour, minute);
        updateMatrixBuffer();
        setTimer0(1000);
    }

    if (timer1_flag == 1) {
        update7SEG(index_led++);
        updateMatrixBuffer();
        index_led = (index_led < MAX_LED) ? index_led : 0;
        setTimer1(250);
    }

    if (timer2_flag == 1) {
        setTimer2(10);
        if (index_led_matrix >= 8) index_led_matrix = 0;
        updateLEDMatrix(index_led_matrix++);
    }
}
```

Giải thích: Vòng lặp chính kiểm tra các cờ của từng bộ định thời và thực hiện tác vụ tương ứng:

- **Timer0:** Khi `timer0_flag` được kích hoạt, chương trình cập nhật `matrix_buffer` để tạo hiệu ứng dịch chuyển, nhấp nháy LED, cập nhật thời gian và khởi tạo lại bộ định thời 1 giây.

- **Timer1:** Khi `timer1_flag` được kích hoạt, chương trình quét và hiển thị các LED 7 đoạn, đồng thời đặt lại bộ định thời 250 ms.
- **Timer2:** Khi `timer2_flag` được kích hoạt, chương trình hiển thị từng cột trong ma trận LED và khởi tạo lại chu kỳ quét tiếp theo (10 ms).