**VI XỬ LÝ – VI ĐIỀU KHIỂN**

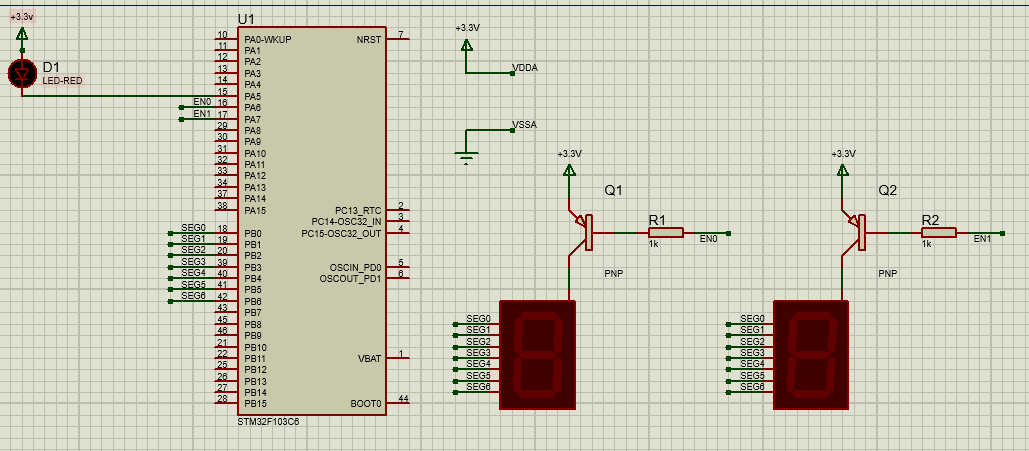
**LAB 2 - Lớp: L03**

**Họ và tên:** Nguyễn Đình Đạt

**Mssv:** 1811869

**Exercise 1:**

**Report 1**: Schematic from Proteus.



[Figure 1.4: Simulation schematic in Proteus](https://github.com/ngddat2706/Vxl_Vdk_lab/blob/12d7a86a15814c096ac64aaa447665af648d65e8/exercise1.pdsprj)

**Report 2**: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function.

/\* USER CODE BEGIN 4 \*/

**void** **display7SEG**(**int** counter){

**switch**(counter){

**case** 0:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5, 0);

**break**;

**case** 1:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 2:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_6, 0);

**break**;

**case** 3:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_6, 0);

**break**;

**case** 4:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 5:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 6:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 7:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 8:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 9:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

}

}

**int** counter = 100;

**void** **HAL\_TIM\_PeriodElapsedCallback**( TIM\_HandleTypeDef \*htim){

counter --;

**if**(counter == 50){

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 0);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, 0);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, 1);

display7SEG(1);

}

**if**(counter <= 0){

counter = 100;

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, 0);

display7SEG(2);

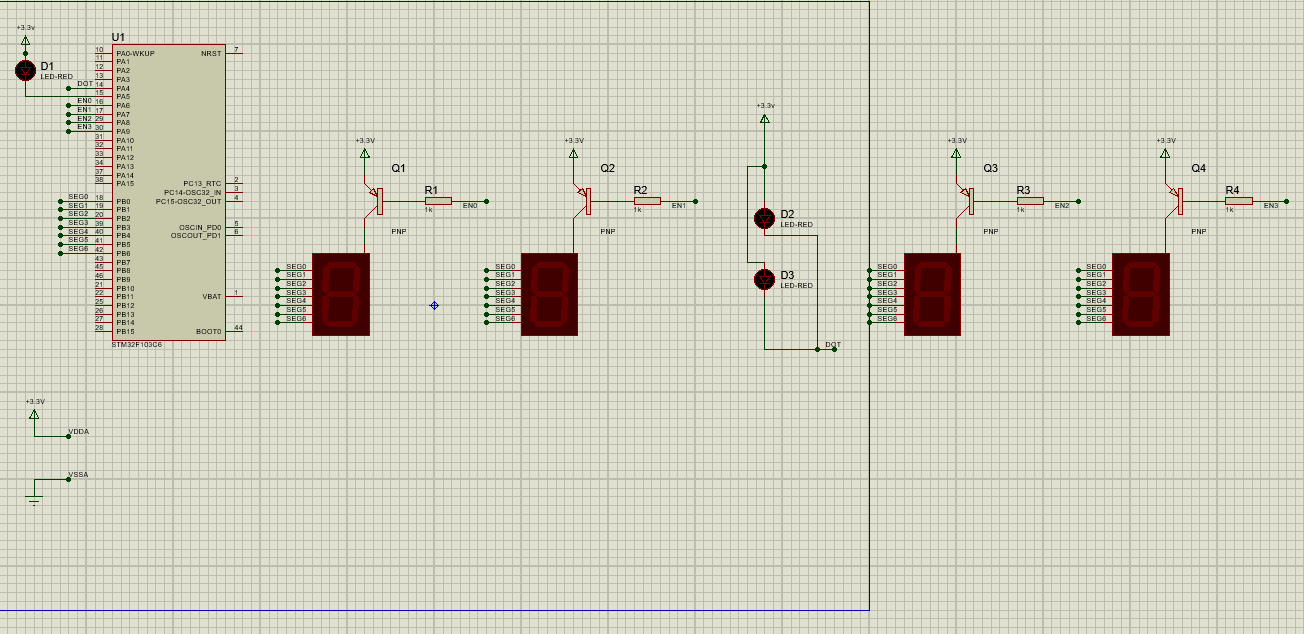
}

}

/\* USER CODE END 4 \*/

**Exercise 2:**

**Report 1**: Schematic from Proteus.



[Figure 1.5: Simulation schematic in Proteus](https://github.com/ngddat2706/Vxl_Vdk_lab/blob/12d7a86a15814c096ac64aaa447665af648d65e8/exercise2.pdsprj)

**Report 2**: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function.

/\* USER CODE BEGIN 4 \*/

**void** **display7SEG**(**int** counter){

**switch**(counter){

**case** 0:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5, 0);

**break**;

**case** 1:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 2:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_6, 0);

**break**;

**case** 3:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_6, 0);

**break**;

**case** 4:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 5:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 6:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 7:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 8:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 9:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

}

}

**void** **clearAllClock**(){

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6| GPIO\_PIN\_7| GPIO\_PIN\_8| GPIO\_PIN\_9, 1);

}

**void** **setNumberOnClock**(**int** num){

**switch**(num){

**case** 0: HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, 0);

**break**;

**case** 1: HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, 0);

**break**;

**case** 2: HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, 0);

**break**;

**case** 3: HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_9, 0);

**break**;

}

}

**int** counter = 200;

**int** counter\_led = 50;

**void** **HAL\_TIM\_PeriodElapsedCallback** ( TIM\_HandleTypeDef \* htim ){

counter--;

counter\_led--;

**if**(counter\_led <= 0){

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

counter\_led = 50;

}

**if**(counter >= 150){

clearAllClock();

setNumberOnClock(0);

display7SEG(1);

}

**else if**(counter >= 100){

clearAllClock();

setNumberOnClock(1);

display7SEG(2);

}

**else if**(counter >= 50 ){

clearAllClock();

setNumberOnClock(2);

display7SEG(3);

}

**else if**(counter >= 0){

clearAllClock();

setNumberOnClock(3);

display7SEG(0);

}

**else** {

counter= 200;

}

}

/\* USER CODE END 4 \*/

**Exercise 3:**

**Report 1:** Source code of the **update7SEG** function.

/\* USER CODE BEGIN 4 \*/

**void** **display7SEG**(**int** counter){

**switch**(counter){

**case** 0:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5, 0);

**break**;

**case** 1:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 2:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_6, 0);

**break**;

**case** 3:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_6, 0);

**break**;

**case** 4:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 5:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 6:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 7:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 8:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 9:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

}

}

**void** **clearAllClock**(){

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6| GPIO\_PIN\_7| GPIO\_PIN\_8| GPIO\_PIN\_9, 1);

}

**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]

clearAllClock();

display7SEG(led\_buffer[0]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, 0);

**break** ;

**case** 1:

// Display the second 7 SEG with led\_buffer [1]

clearAllClock();

display7SEG(led\_buffer[1]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, 0);

**break** ;

**case** 2:

// Display the third 7 SEG with led\_buffer [2]

clearAllClock();

display7SEG(led\_buffer[2]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, 0);

**break** ;

**case** 3:

// Display the forth 7 SEG with led\_buffer [3]

clearAllClock();

display7SEG(led\_buffer[3]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_9, 0);

**break** ;

**default** :

**break** ;

}

}

**Report 2:** Source code in the **HAL\_TIM\_PeriodElapsedCallback**.

**int** counter = 50;

**void** **HAL\_TIM\_PeriodElapsedCallback** ( TIM\_HandleTypeDef \* htim ){

counter--;

**if**(counter <= 0){

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

update7SEG(index\_led);

index\_led++;

**if**(index\_led == MAX\_LED){

index\_led =0;

}

Counter = 50;

}

}

**Exercise 4:**

**Report 1:** Source code in the **HAL\_TIM\_PeriodElapsedCallbac**k.

**int** counter = 25;

**int** counter\_led = 50;

**void** **HAL\_TIM\_PeriodElapsedCallback** ( TIM\_HandleTypeDef \* htim ){

counter--;

counter\_led--;

**if**(counter\_led <= 0){

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

counter\_led =50;

}

**if**(counter == 0){

update7SEG(index\_led);

index\_led++;

**if**(index\_led == MAX\_LED){

index\_led =0;

}

counter= 25;

}

}

**Exercise 5:**

**Report 1:** Source code in the updateClockBuffer function.

**int** hour = 15 , minute = 8 , second = 50;

**int** led\_buffer [4] = {1 , 2 , 3 , 4};

**void** **updateClockBuffer** (){

led\_buffer[0] = hour/10;

led\_buffer[1] = hour%10;

led\_buffer[2] = minute/10;

led\_buffer[3] = minute%10;

}

**Exercise 6:**

**Report 1:** What happens after that and why?

* LED RED không thay đổi\\

-> Nếu bỏ qua dòng 1: setTimer0(1000) thì giá trị timer0\_counter luôn bằng 0, dẫn đến timer0\_flag luôn bằng 0 nên trạng thái Led RED không thay đổi.\\

**Report 2:** If in line 1 of the code above is changed to setTimer0(1), what happens after that and why?

* LED RED không thay đổi

Sau khi thực hiện setTimer0(1) thì timer0\_counter vẫn bằng 0, dẫn đến timer0\_flag luôn bằng 0 nên trạng thái Led RED không thay đổi.

**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?

* Led Red nhấp nháy

Sau khi thực hiện setTimer0(10) thì timer0\_counter = 1 > 0, nên sau khi thực hiện time\_run() thì timer0\_flag = 1, dẫn đến hàm if trong vòng while được thực hiện, nên trạng thái của Led Red được đảo. Tiếp tục thực hiện hàm setTimer0(2000) thì timer0\_counter = 200 > 0, sau khi timer0\_counter giảm đến 0 thì timer0\_flag = 1, dẫn đến hàm if trong vòng while được thực hiện, nên trạng thái của Led Red được đảo.

**Exercise 7:**

**int** timer0\_counter = 0; // Time to update 7SEG

**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;

}

}

setTimer0(1000);

**while** (1)

{

/\* USER CODE END WHILE \*/

**if**(timer0\_flag == 1){

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

second ++;

**if** ( second >= 60) {

second = 0;

minute ++;

}

**if**( minute >= 60) {

minute = 0;

hour ++;

}

**if**( hour >=24) {

hour = 0;

}

updateClockBuffer () ;

setTimer0(1000);

}

/\* USER CODE BEGIN 3 \*/

}

**Exercise 8:**

**int** hour = 15 , minute = 8 , second = 57;

**int** led\_buffer [4] = {1 , 2 , 3 , 4};

// Hàm updateClockBuffer sẽ tạo ra các giá trị cho mảng led\_buffer theo các giá trị của giờ và phút.

**void** **updateClockBuffer** (){

led\_buffer[0] = hour/10;

led\_buffer[1] = hour%10;

led\_buffer[2] = minute/10;

led\_buffer[3] = minute%10;

}

// Exercise 6

**int** timer0\_counter = 0; // Time to update 7SEG

**int** timer0\_flag = 0;

**int** timer1\_counter = 0; // Time to update 7SEG

**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;

}

}

**void** **display7SEG**(**int** counter){

**switch**(counter){

**case** 0:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5, 0);

**break**;

**case** 1:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 2:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_6, 0);

**break**;

**case** 3:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4|GPIO\_PIN\_5, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_6, 0);

**break**;

**case** 4:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_3|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 5:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1|GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 6:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 7:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2, 0);

**break**;

**case** 8:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

**case** 9:

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_5|GPIO\_PIN\_6, 0);

**break**;

}

}

**void** **clearAllClock**(){

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6| GPIO\_PIN\_7| GPIO\_PIN\_8| GPIO\_PIN\_9, 1);

}

**const** **int** MAX\_LED = 4;

**int** index\_led = 0;

**void** **update7SEG** ( **int** index ) {

**switch** ( index ) {

**case** 0:

// Display the first 7 SEG with led\_buffer [0]

clearAllClock();

display7SEG(led\_buffer[0]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, 0);

**break** ;

**case** 1:

// Display the second 7 SEG with led\_buffer [1]

clearAllClock();

display7SEG(led\_buffer[1]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, 0);

**break** ;

**case** 2:

// Display the third 7 SEG with led\_buffer [2]

clearAllClock();

display7SEG(led\_buffer[2]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, 0);

**break** ;

**case** 3:

// Display the forth 7 SEG with led\_buffer [3]

clearAllClock();

display7SEG(led\_buffer[3]);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_9, 0);

**break** ;

**default** :

**break** ;

}

}

Trong ham main:

setTimer1(250);

setTimer0(10);

**while** (1)

{

**if**(timer0\_flag == 1){

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_5);

setTimer0(2000);

}

**if**(timer1\_flag == 1){

update7SEG(index\_led);

index\_led++;

**if**(index\_led == MAX\_LED){

index\_led =0;

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

second ++;

**if** ( second >= 60) {

second = 0;

minute ++;

}

**if**( minute >= 60) {

minute = 0;

hour ++;

}

**if**( hour >=24) {

hour = 0;

}

}

updateClockBuffer () ;

setTimer1(250);

}

}

**void** **HAL\_TIM\_PeriodElapsedCallback** ( TIM\_HandleTypeDef \* htim ){

timer\_run(); // function for update 7 SEG

}

**Exercise 10:**

**//** Hàm updateLEDMatrix gồm hai thông số truyền vào

// Một là chỉ số cột hiện tại

// Hai là số lần dịch sang trái

void updateLEDMatrix(int index, int index\_run){

// Chỉ số matrix\_buffer của chỉ số cột hiện tại = chỉ số cột hiện tại + số lần dịch sang trái

// Nếu chỉ số matrix\_buffer lớn hơn MAX\_LED\_MATRIX thì trừ đi MAX\_LED\_MATRIX

GPIOB->BRR = matrix\_buffer[index + index\_run] << 8 ;

updateLEDMatrix(index\_led\_matrix, index\_led\_run);

index\_led\_matrix++;

if(index\_led\_matrix >= MAX\_LED\_MATRIX){

index\_led\_matrix = 0;

// Sau khi hiển thị hết 8 cột thì tăng số lần dịch lên

index\_led\_run ++;

// Nếu số lần dịch vượt quá MAX\_LED\_MATRIX thì set về 0

if(index\_led\_run >= MAX\_LED\_MATRIX){

index\_led\_run = 0;

}

}