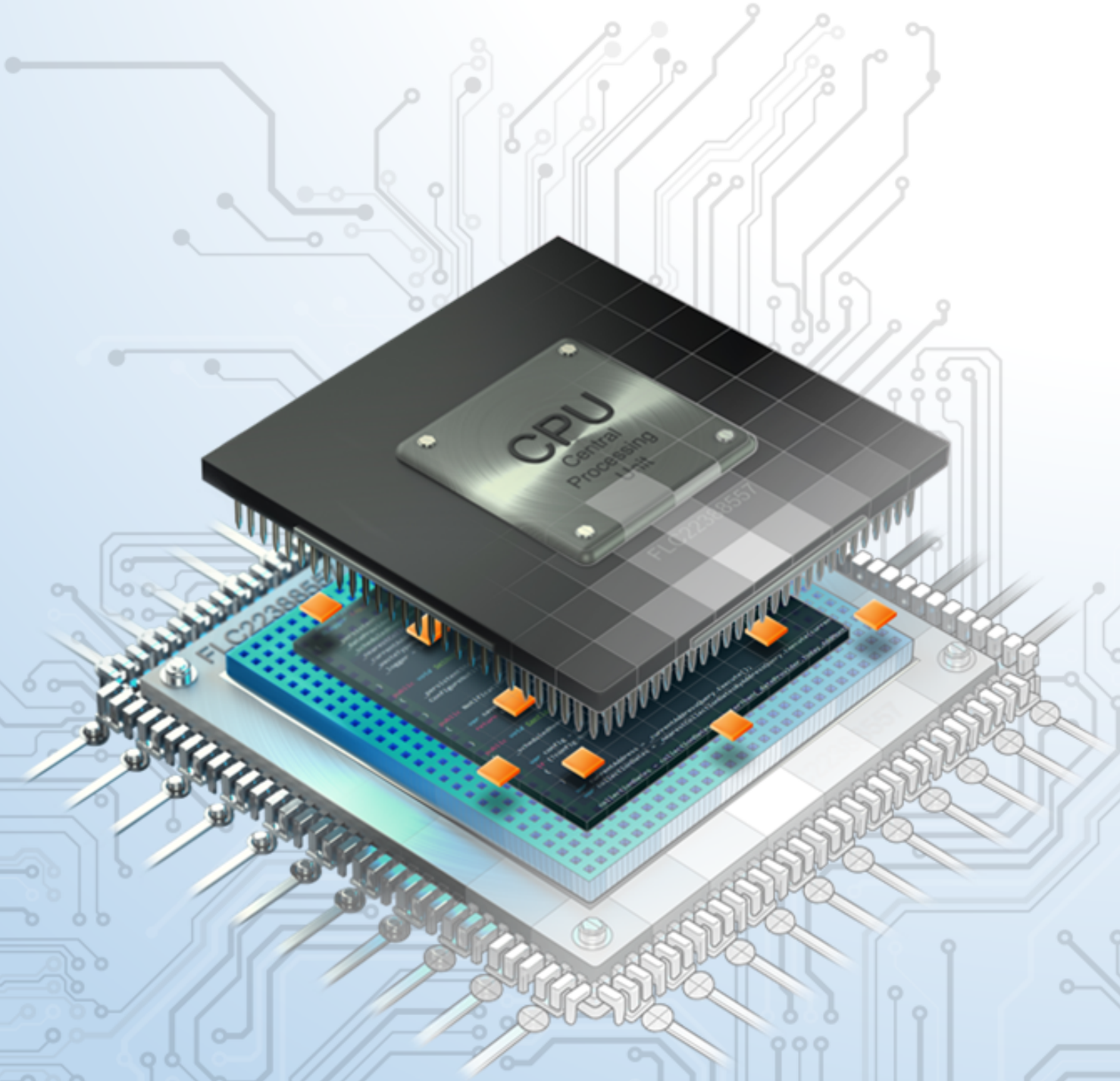
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



Dr. Le Trong Nhan

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

REPORT
MICROCONTROLLER (CO3010)

Midterm

Lecturer: Le Trong Nhan
Huynh Phuc Nghi
Class: L05

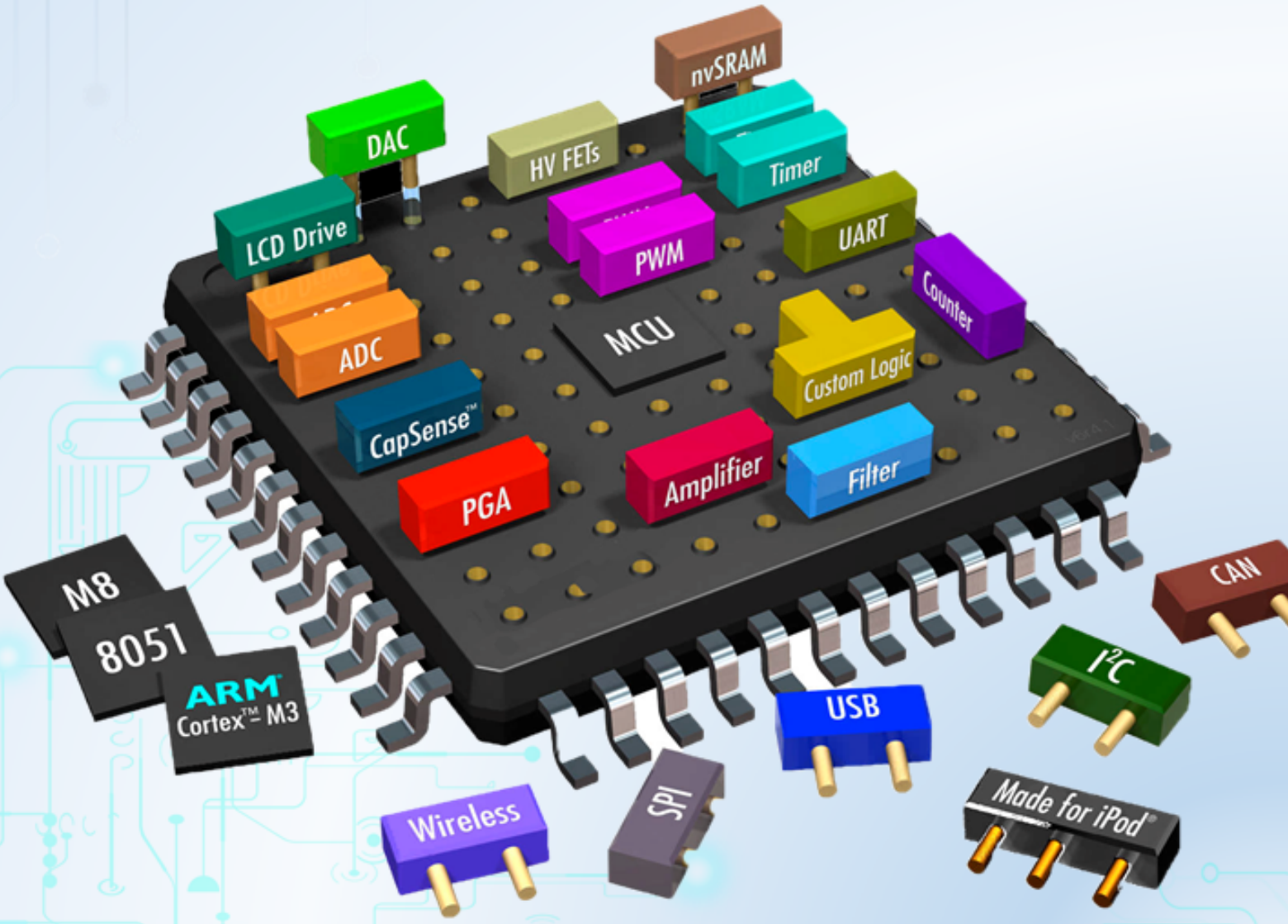
Name	StudentID
Nguyen Cong Thanh	1915144

Ho Chi Minh City, November 2021

Mục lục

CHƯƠNG 1

Bài Tập Giữa Kì



1 Giới thiệu

Trong yêu cầu của bài giữa kì, một đồng hồ analog với 12 bóng đèn hiển thị, trượng trưng cho 12 số trên đồng hồ. Bên cạnh đó, sẽ có 3 nút nhấn: nút MENU, INC và DEC (để tăng và giảm thông tin)

Để tiện lợi trong quá trình demo, đồng hồ sẽ có 12 giờ. Tuy nhiên, mỗi phút sẽ chỉ có 12 giây, và mỗi giờ cũng sẽ có 12 phút.
Đồng hồ có 3 chế độ hoạt động như sau:

- Mode 0: Đây là chế độ mặt định mỗi khi bật nguồn hoặc khởi động lại hệ thống. Thông tin giờ phút và giây sẽ được hiển thị trên màn hình. Tại một thời điểm, chỉ có tối đa 3 đèn được hiển thị. Nếu 2 thông tin (hoặc 3 thông tin) trùng nhau, thì chỉ 1 đèn sẽ được hiển thị. Khi đang ở chế độ này, thông tin về giờ phút giây sẽ được cập nhật theo đúng quy luật của đồng hồ.
- Mode 1: Khi đang ở Mode 0 và nhấn vào nút MENU, đồng hồ sẽ chuyển sang chế độ này để chỉnh giờ. **Thông tin về giờ phút giây sẽ ngừng cập nhật để người dùng chỉnh giờ**. Chỉ một thông tin về giờ hiện tại sẽ được hiển thị. Khi nhấn nút INC và DEC, thông tin giờ sẽ được cộng thêm, hoặc trừ đi. Đồng thời, đèn hiển thị cũng sẽ được cập nhật trên mặt của đồng hồ. Lưu ý rằng, khi đang ở vị trí số 1, và nhấn nút trừ (DEC), thì thông tin giờ sẽ đếm vòng sang 12. Mỗi lần nhấn nút INC hoặc DEC, thông tin mới về giờ sẽ được lưu lại ngay lập tức.
- Mode 2: Khi đang ở Mode 1 và nhấn vào nút MENU, đồng hồ sẽ chuyển sang chế độ chỉnh phút. Chỉ một thông tin về phút hiện tại của được hiển thị trên màn hình. Việc chỉnh phút cũng được thực hiện tương tự như chỉnh giờ.

Khi đang ở Mode 1 hoặc Mode 2, người dùng không tương tác vào bất cứ nút nào (MENU, INC hay DEC) trong vòng 5 giây, hệ thống tự động chuyển về Mode 0.

Hệ thống sẽ được hiện thực trên phần mềm mô phỏng Proteus. Vi điều khiển STM32103C6 sẽ được sử dụng. Sinh viên không bắt buộc phải sử dụng ngắt timer. Nút nhấn chỉ tích cực mỗi khi nhấn xuống và **không kiểm tra trường hợp nhấn đè**. Ví dụ đang ở trạng thái chỉnh giờ, nút INC được nhấn xuống, giờ ngay lập tức sẽ được tăng lên 1 đơn vị. Tuy nhiên nếu cứ giữ đè, thì giờ vẫn không tăng. Nút nhấn được chống rung bằng cách đọc 2 lần liên tiếp giống nhau, mỗi lần cách nhau 10ms.

2 Nộp bài

Sinh viên hoàn thiện file report này, với các nội dung được yêu cầu bên dưới, file main.c ,tất cả các file .h và .c sinh viên hiện thực thêm sẽ được nén lại với MSSV và nộp lên hệ thống.

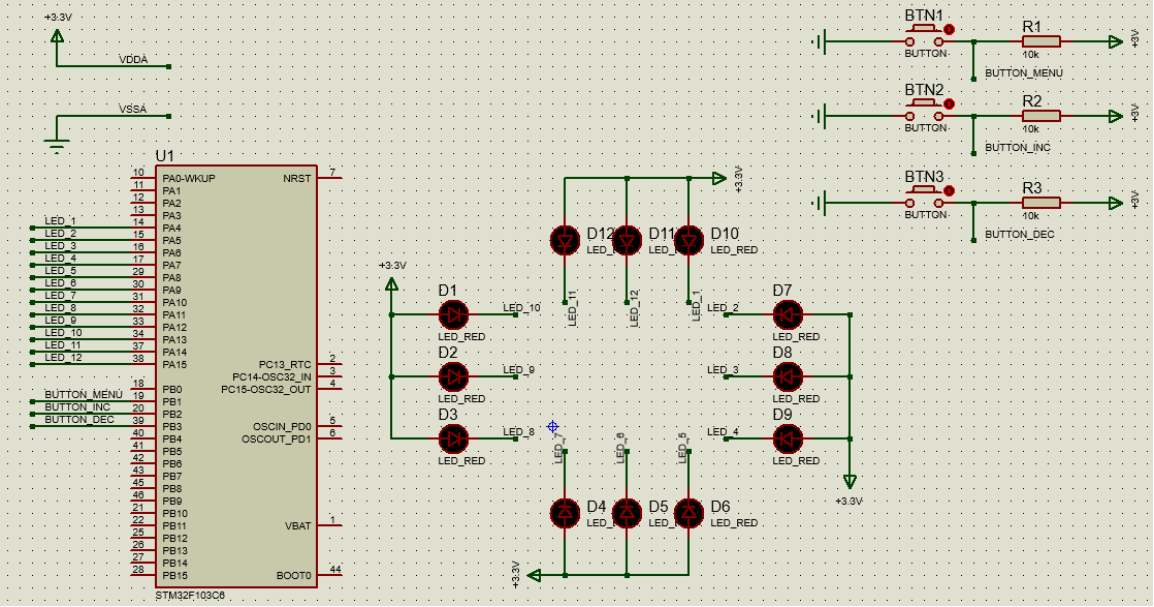
Sinh viên quay màn hình phần demo trên Proteus và tải lên Drive của mình (ở chế độ public). Link của video demo sẽ được trình bày ở phần cuối của Report.

3 Report

3.1 Mô phỏng trên Proteus

Thiết kế sơ đồ mạch điện trên Proteus, bao gồm 12 chân đèn LED và 3 nút nhấn. Để đơn giản, sinh viên có thể lược bỏ điện trở hạn dòng cho đèn LED.

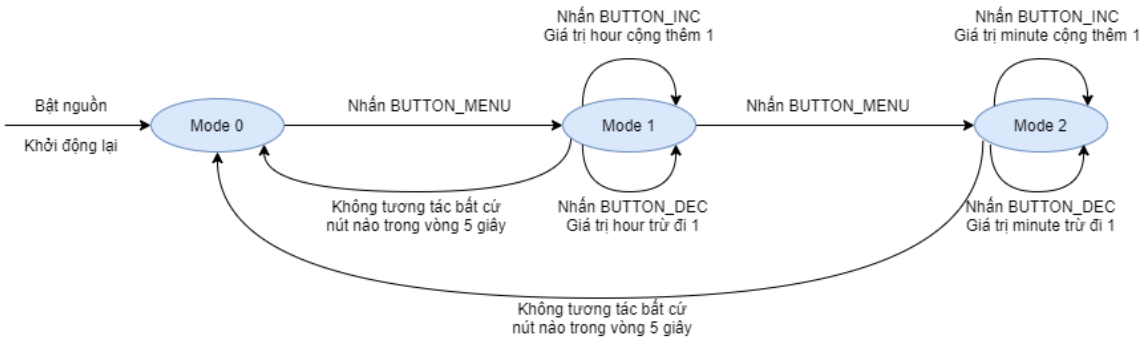
Sinh viên chụp hình màn hình mô phỏng Proteus và dán vào phần report này.



Hình 1.1: *Lược đồ mô phỏng Proteus*

3.2 Thiết kế máy trạng thái

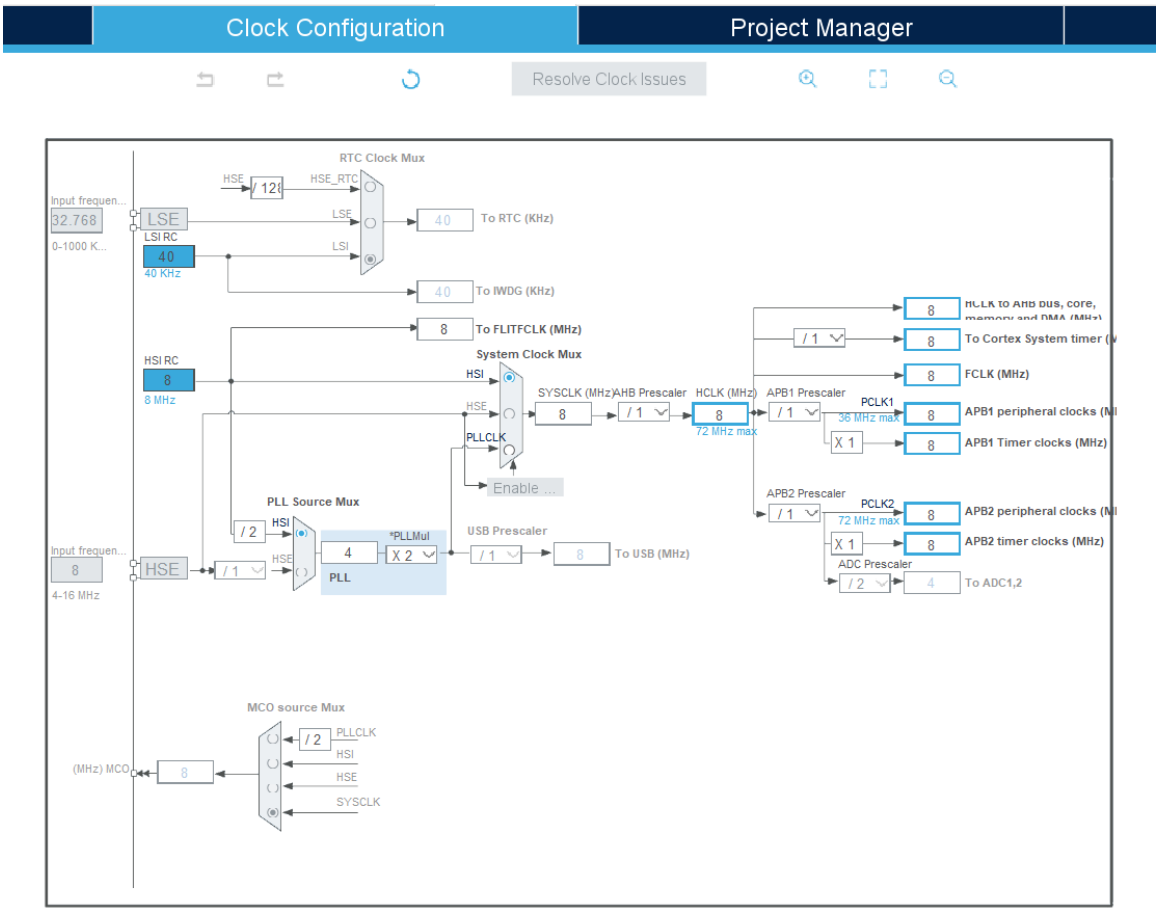
Sinh viên thiết kế máy trạng thái cho hệ thống và trình bày ở phần này. Sinh viên được khuyến khích giải thích thêm cho máy trạng thái của mình để tiện lợi cho giảng viên khi đánh giá.



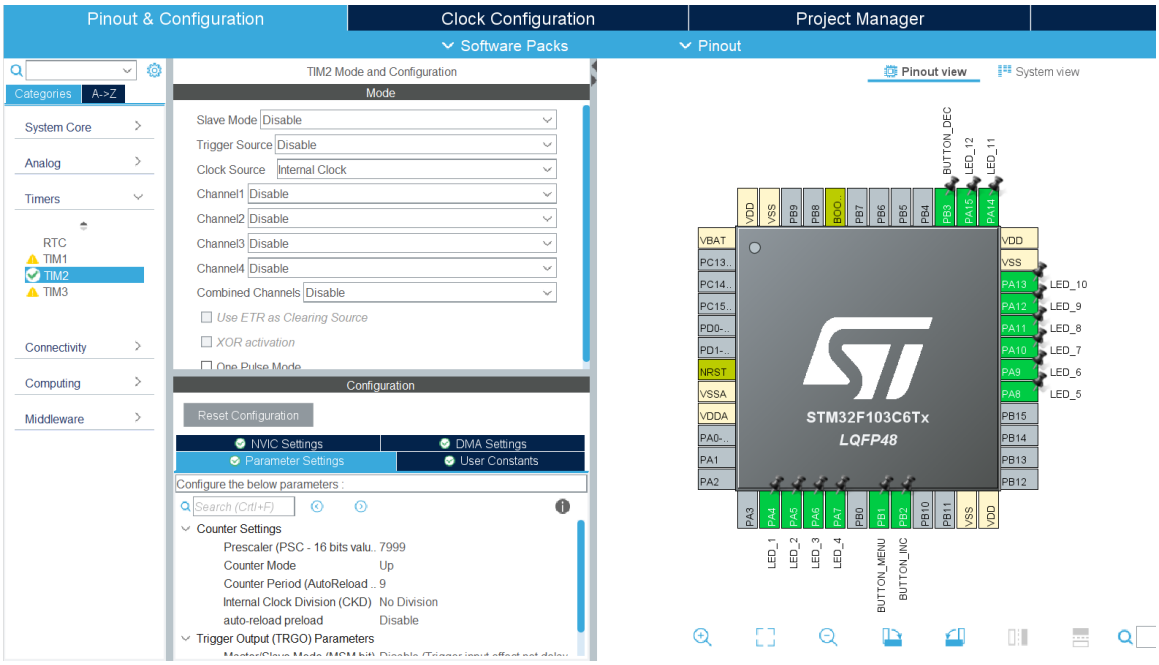
Hình 1.2: *Máy trạng thái FSM*

3.3 Lập trình trên STM32Cube

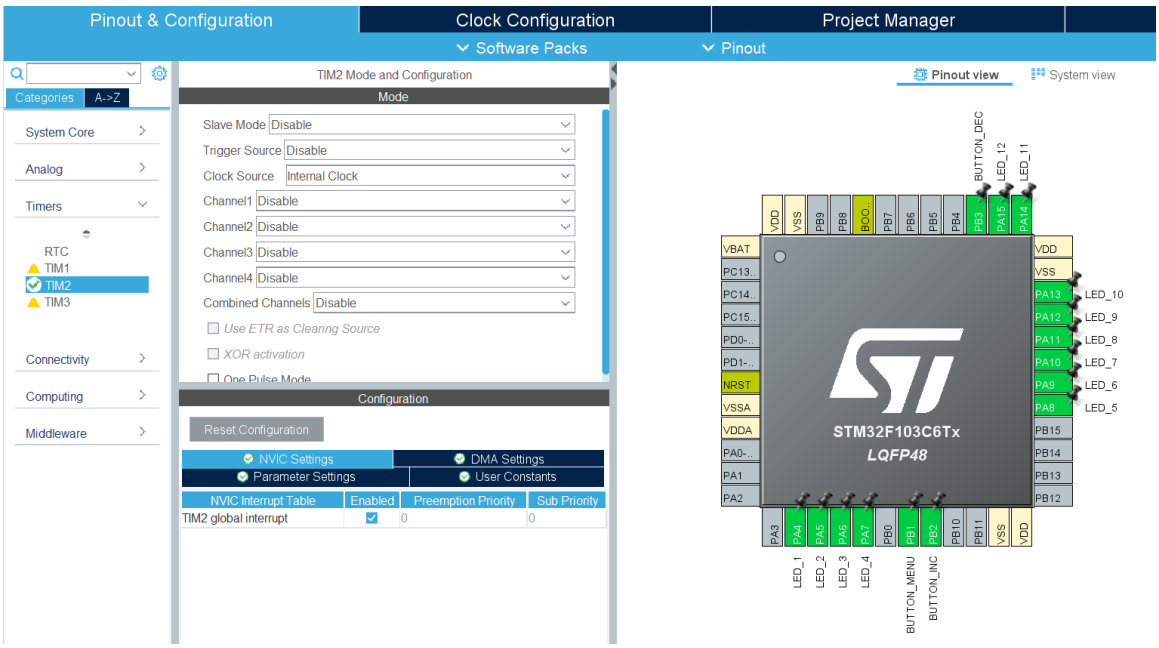
Cấu hình ngắt timer trên STM32Cube với cấu hình là ngắt 10ms:



Hình 1.3: Cấu hình ngắt timer trên STM32Cube



Hình 1.4: Cấu hình ngắt timer trên STM32Cube



Hình 1.5: Cấu hình ngắt timer trên STM32Cube

Cấu trúc chương trình trên main khi sử dụng ngắt timer:

```c
int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (&htim2);
  /* USER CODE END 2 */
  clear_button();
  initialize_variable();
  clearAllClock();
  setTimer0(10);
  while (1)
  {
    if(timer0_flag == 1){
        clock_fsm();
        setTimer0(10);
    }
  }
}
```

Program 1.1: Cấu trúc chương trình trên main

3.4 Module Timer

Đặc tả ngắn gọn về module này

3.4.1 File global.h

File **global.h** chứa các biến được khai báo trong chương trình.

```c
#ifndef INC_GLOBAL_H_
#define INC_GLOBAL_H_

// Declare variables for clock
int hour, minute, second;
int count_second;
// Declare counter for timer 1s in mode 1
int counter;
// Declare counter_flag to return mode 1
int counter_flag_5s;
void initialize_variable();
#endif /* INC_GLOBAL_H_ */
```

Program 1.2: Mã nguồn global .h

3.4.2 File global.c

File **global.c** chứa hàm khởi tạo các biến được dùng trong chương trình với các ý nghĩa ở mục comment.

```
#include "global.h"
void initialize_variable(){
  // Declare variables for clock
  hour = 3;
  minute = 6;
  second = 0;
  // Declare counter for timer 1s in mode 1
  counter = 100;
  // Declare counter_flag to return mode 1
  counter_flag_5s = 500;
}
```

Program 1.3: Mã nguồn global.c

3.4.3 File timer.h

File **timer.h** chứa các biến và định nghĩa các hàm ngắt timer.

```
#ifndef INC_TIMER_H_
#define INC_TIMER_H_
extern int timer0_flag;

void setTimer0(int duration);
void timer_run();

#endif /* INC_TIMER_H_ */
```

Program 1.4: Mã nguồn timer .h

3.4.4 File timer.c

File **timer.c** khởi tạo các biến và hiện thực các hàm ngắt timer.

```
#include "timer.h"
#include "main.h"
#include "input_reading.h"

int timer0_counter = 0;
int timer0_flag = 0;
int TIMER_CYCLE = 10;

void setTimer0(int duration){
  timer0_counter = duration / TIMER_CYCLE;
  timer0_flag = 0;
}

```

```c
void timer_run(){
  if(timer0_counter > 0){
    timer0_counter --;
    if(timer0_counter <= 0) timer0_flag = 1;
  }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim->Instance == TIM2)
    button_scan();
  timer_run();
}
```

Program 1.5: Mã nguồn timer .c

3.4.5 File led_display.h

File **led_display.h** dùng để định nghĩa các hàm hiển thị 12 led red đơn.

```c
#ifndef INC_LED_DISPLAY_H_
#define INC_LED_DISPLAY_H_

void clearAllClock();
void setNumberOnClock(int num);
void clearNumberOnClock(int num);

#endif /* INC_LED_DISPLAY_H_ */
```

Program 1.6: Mã nguồn led_display.h

3.4.6 File led_display.c

File **led_display.c** dùng để hiện thực các hàm hiển thị các chế độ của 12 led red đơn.

```c
#include "main.h"
#include "led_display.h"
void clearAllClock(){
  HAL_GPIO_WritePin(GPIOA, LED_1_Pin|LED_2_Pin|LED_3_Pin|
   LED_4_Pin|LED_5_Pin|LED_6_Pin|LED_7_Pin|LED_8_Pin|
   LED_9_Pin|LED_10_Pin|LED_11_Pin|LED_12_Pin, 1);
}
void setNumberOnClock(int num){
  switch(num){
  case 0:
      HAL_GPIO_WritePin(GPIOA, LED_12_Pin, 0);
      break;
  case 1:
      HAL_GPIO_WritePin(GPIOA, LED_1_Pin, 0);
      break;
```

```
  case 2:
      HAL_GPIO_WritePin(GPIOA, LED_2_Pin, 0);
      break;
  case 3:
      HAL_GPIO_WritePin(GPIOA, LED_3_Pin, 0);
      break;
  case 4:
      HAL_GPIO_WritePin(GPIOA, LED_4_Pin, 0);
      break;
  case 5:
      HAL_GPIO_WritePin(GPIOA, LED_5_Pin, 0);
      break;
  case 6:
      HAL_GPIO_WritePin(GPIOA, LED_6_Pin, 0);
      break;
  case 7:
      HAL_GPIO_WritePin(GPIOA, LED_7_Pin, 0);
      break;
  case 8:
      HAL_GPIO_WritePin(GPIOA, LED_8_Pin, 0);
      break;
  case 9:
      HAL_GPIO_WritePin(GPIOA, LED_9_Pin, 0);
      break;
  case 10:
      HAL_GPIO_WritePin(GPIOA, LED_10_Pin, 0);
      break;
  case 11:
      HAL_GPIO_WritePin(GPIOA, LED_11_Pin, 0);
      break;
  }
}
void clearNumberOnClock(int num){
  switch(num){
    case 0:
        HAL_GPIO_WritePin(GPIOA, LED_12_Pin, 1);
        break;
    case 1:
        HAL_GPIO_WritePin(GPIOA, LED_1_Pin, 1);
        break;
    case 2:
        HAL_GPIO_WritePin(GPIOA, LED_2_Pin, 1);
        break;
    case 3:
        HAL_GPIO_WritePin(GPIOA, LED_3_Pin, 1);
        break;
    case 4:
        HAL_GPIO_WritePin(GPIOA, LED_4_Pin, 1);
        break;
```

```
63      case 5:
64          HAL_GPIO_WritePin(GPIOA, LED_5_Pin, 1);
65          break;
66      case 6:
67          HAL_GPIO_WritePin(GPIOA, LED_6_Pin, 1);
68          break;
69      case 7:
70          HAL_GPIO_WritePin(GPIOA, LED_7_Pin, 1);
71          break;
72      case 8:
73          HAL_GPIO_WritePin(GPIOA, LED_8_Pin, 1);
74          break;
75      case 9:
76          HAL_GPIO_WritePin(GPIOA, LED_9_Pin, 1);
77          break;
78      case 10:
79          HAL_GPIO_WritePin(GPIOA, LED_10_Pin, 1);
80          break;
81      case 11:
82          HAL_GPIO_WritePin(GPIOA, LED_11_Pin, 1);
83          break;
84      }
85  }
```

Program 1.7: Mã nguồn led_display.c

3.4.7 File input_reading.h

File **input_reading.h** dùng để định nghĩa các hàm xử lý button.
Hàm **clear_button();** dùng để reset các giá trị button.
Hàm **buton_scan(void);** dùng để ghi nhận các trạng thái hiện tại của mỗi button.
Hàm **is_button_pressed(unsigned char index);** dùng để kiểm tra xem button có được nhấn hay không.

```
1  #ifndef INC_INPUT_READING_H_
2  #define INC_INPUT_READING_H_
3  void button_scan(void);
4  void clear_button();
5  unsigned char is_button_pressed(unsigned char index);
6  #endif /* INC_INPUT_READING_H_ */
```

Program 1.8: Mã nguồn input_reading.h

3.4.8 File input_reading.c

File **input_reading.c** dùng để định nghĩa và hiện thực các biến, các hàm đã khởi tạo ở **input_reading.h**.

```
1  #include "main.h"
2  #include "input_reading.h"
3  //we aim to work with more than one buttons
```

```c
#define NO_OF_BUTTONS                    3
#define DURATION_FOR_AUTO_INCREASING    100
// i = 0 => BUTTON_MENU; i = 1 => BUTTON_INC; i = 2 => BUTTON_DEC;
#define BUTTON_IS_PRESSED                GPIO_PIN_RESET
#define BUTTON_IS_RELEASED               GPIO_PIN_SET
//the buffer that the final result is stored after debouncing
static GPIO_PinState buttonBuffer[NO_OF_BUTTONS];
//we define two buffers for debouncing
static GPIO_PinState debounceButtonBuffer1[NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer2[NO_OF_BUTTONS];
//we define counter for checking the button is pressed
static uint16_t counterForButtonPress[NO_OF_BUTTONS];
void clear_button(){
  for(int i = 0; i < NO_OF_BUTTONS; i ++){
    debounceButtonBuffer1[i] = GPIO_PIN_SET; //No press
    debounceButtonBuffer2[i] = GPIO_PIN_SET; //No press
    buttonBuffer[i] = GPIO_PIN_SET; //No press
  }
  HAL_GPIO_WritePin(BUTTON_MENU_GPIO_Port, BUTTON_MENU_Pin, 1);
  HAL_GPIO_WritePin(BUTTON_INC_GPIO_Port, BUTTON_INC_Pin, 1);
  HAL_GPIO_WritePin(BUTTON_DEC_GPIO_Port, BUTTON_DEC_Pin, 1);
}
void button_scan(void){
  for(int i = 0; i < NO_OF_BUTTONS; i ++){
    debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
    //Catch signal every time the corresponding button is pressed
    if(i == 0)
      debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(BUTTON_MENU_GPIO_Port, BUTTON_MENU_Pin);
    else if (i == 1)
      debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(BUTTON_INC_GPIO_Port, BUTTON_INC_Pin);
    else if (i == 2)
      debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(BUTTON_DEC_GPIO_Port, BUTTON_DEC_Pin);
    if(debounceButtonBuffer1[i] == debounceButtonBuffer2[i]){
      buttonBuffer[i] = debounceButtonBuffer1[i];
      if(buttonBuffer[i] == BUTTON_IS_PRESSED){
        //If a button is pressed, we start counting
        if(counterForButtonPress[i] < DURATION_FOR_AUTO_INCREASING){
          counterForButtonPress[i]++;
```

```
        }
      } else {
        counterForButtonPress[i] = 0;
      }
    }
  }
}

unsigned char is_button_pressed(uint8_t index){
  if(index >= NO_OF_BUTTONS) return 0;
  if (buttonBuffer[index] == BUTTON_IS_PRESSED)
  {
    return counterForButtonPress[index];
  }
  return 0;
}
```

Program 1.9: Mã nguồn input_reading.c

3.4.9 File input_processing.h

File **input_processing.h** dùng để định nghĩa hàm clock_fsm(void) cho máy trạng thái của chương trình.

```
#ifndef INC_INPUT_PROCESSING_H_
#define INC_INPUT_PROCESSING_H_

void clock_fsm(void);

#endif /* INC_INPUT_PROCESSING_H_ */
```

Program 1.10: Mã nguồn input_processing.h

3.4.10 File input_processing.c

File **input_processing.c** dùng để hiện thực hàm clock_fsm(void) cho máy trạng thái của chương trình. Với các mode 0, mode 1, mode 2 được đặc tả như yêu cầu ban đầu.

```
#include "main.h"
#include "input_reading.h"
#include "input_processing.h"
#include "led_display.h"
#include "global.h"
enum ButtonState{MODE_0=0, MODE_1=1, MODE_2=2};
enum ButtonState buttonState = MODE_0;
enum ButtonIndex{PRESS_BUTTON_MENU = 0, PRESS_BUTTON_INC=1,
    PRESS_BUTTON_DEC=2};

void clock_fsm(void){
  switch(buttonState){
```

```
12      case MODE_0:
13        // Display the clock is running normally
14        if(counter == 0){
15          second ++;
16          clearNumberOnClock((second == 0) ? 11 : (second-1))
    ;
17          if(second > 11){
18            second = 0;
19            clearNumberOnClock(minute);
20            minute ++;
21            if(minute > 11){
22              minute = 0;
23              clearNumberOnClock(hour);
24              hour ++;
25              if(hour > 11)
26                hour = 0;
27            }
28          }
29          counter = 100;
30        }
31        counter --;
32        setNumberOnClock(hour);
33        setNumberOnClock(minute);
34        setNumberOnClock(second);
35        // When BUTTON_MENU is pressed, go to the next state
36        if(is_button_pressed(PRESS_BUTTON_MENU) == 1){
37          clearAllClock();
38          buttonState = MODE_1;
39        }
40        break;
41      case MODE_1:
42        setNumberOnClock(hour);
43        // When BUTTON_INC  is pressed, hour value increased
    by 1
44        if(is_button_pressed(PRESS_BUTTON_INC) == 1){
45          counter_flag_5s = 500;
46          clearNumberOnClock(hour);
47          hour += 1;
48          if(hour > 11)
49            hour = 0;
50        }
51        // When BUTTON_DEC is pressed, hour value decreased
    by 1
52        if(is_button_pressed(PRESS_BUTTON_DEC) == 1){
53          counter_flag_5s = 500;
54          clearNumberOnClock(hour);
55          if(hour == 0)
56            hour = 11;
57          else hour -= 1;
```

```
58        }
59        // When BUTTON_MENU is pressed, go to the next state
60        if(is_button_pressed(PRESS_BUTTON_MENU) == 1){
61          counter_flag_5s = 500;
62          clearAllClock();
63          buttonState = MODE_2;
64        }
65        // Check that no button is pressed within 5s
66        counter_flag_5s --;
67        if(counter_flag_5s == 0){
68          buttonState = MODE_0;
69        }
70        break;
71      case MODE_2:
72        setNumberOnClock(minute);
73        // When BUTTON_INC  is pressed, hour value increased
    by 1
74        if(is_button_pressed(PRESS_BUTTON_INC) == 1){
75          counter_flag_5s = 500;
76          clearNumberOnClock(minute);
77          minute += 1;
78          if(minute > 11){
79            minute = 0;
80          }
81        }
82        // When BUTTON_DEC is pressed, hour value decreased
    by 1
83        if(is_button_pressed(PRESS_BUTTON_DEC) == 1){
84          counter_flag_5s = 500;
85          clearNumberOnClock(minute);
86          if(minute == 0)
87            minute = 11;
88          else minute -= 1;
89        }
90        // Check that no button is pressed within 5s
91        counter_flag_5s --;
92        if(counter_flag_5s == 0){
93          buttonState = MODE_0;
94        }
95        break;
96      default:
97        break;
98    }
99  }
```

Program 1.11: Mã nguồn input_processing.c

4 Video demo

Sinh viên cần quay 1 đoạn demo ngắn trên Proteus để minh họa việc chỉnh giờ, dừng 5 giây không chỉnh phút để thoát ra chế độ hoạt động bình thường. Sau đó nhấn vào MENU để chỉnh giờ, rồi nhấn tiếp MENU để chỉnh phút. Sau khi chỉnh phút xong (không tương tác trong 5s), hệ thống sẽ quay lại chế độ hoạt động bình thường.

Đường link cho video demo như sau:

Link drive: https://bit.ly/1915144-Midterm
Link dự phòng (Youtube): https://youtu.be/lNcV0GSsW$_Y$