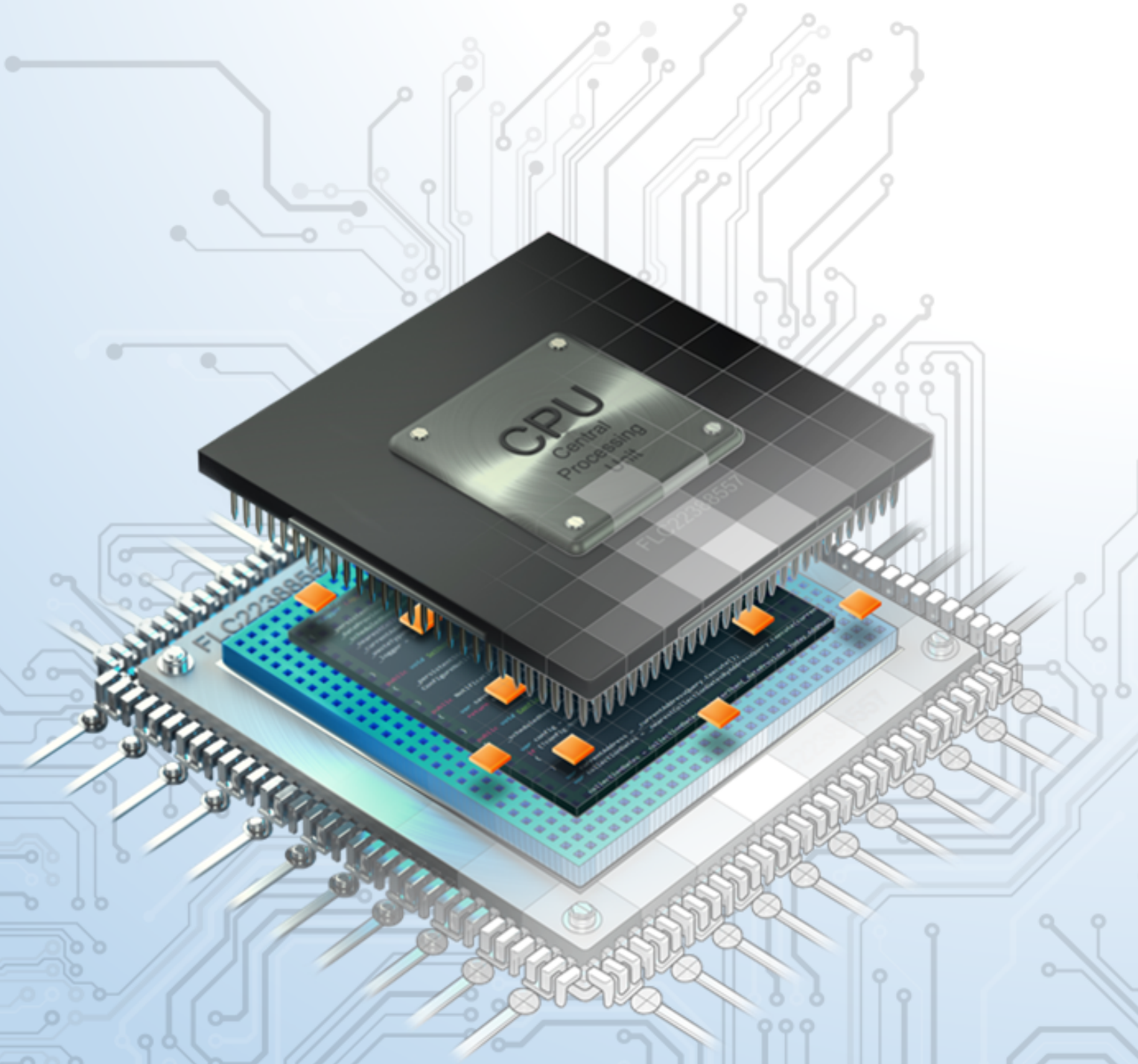
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

# Microcontroller

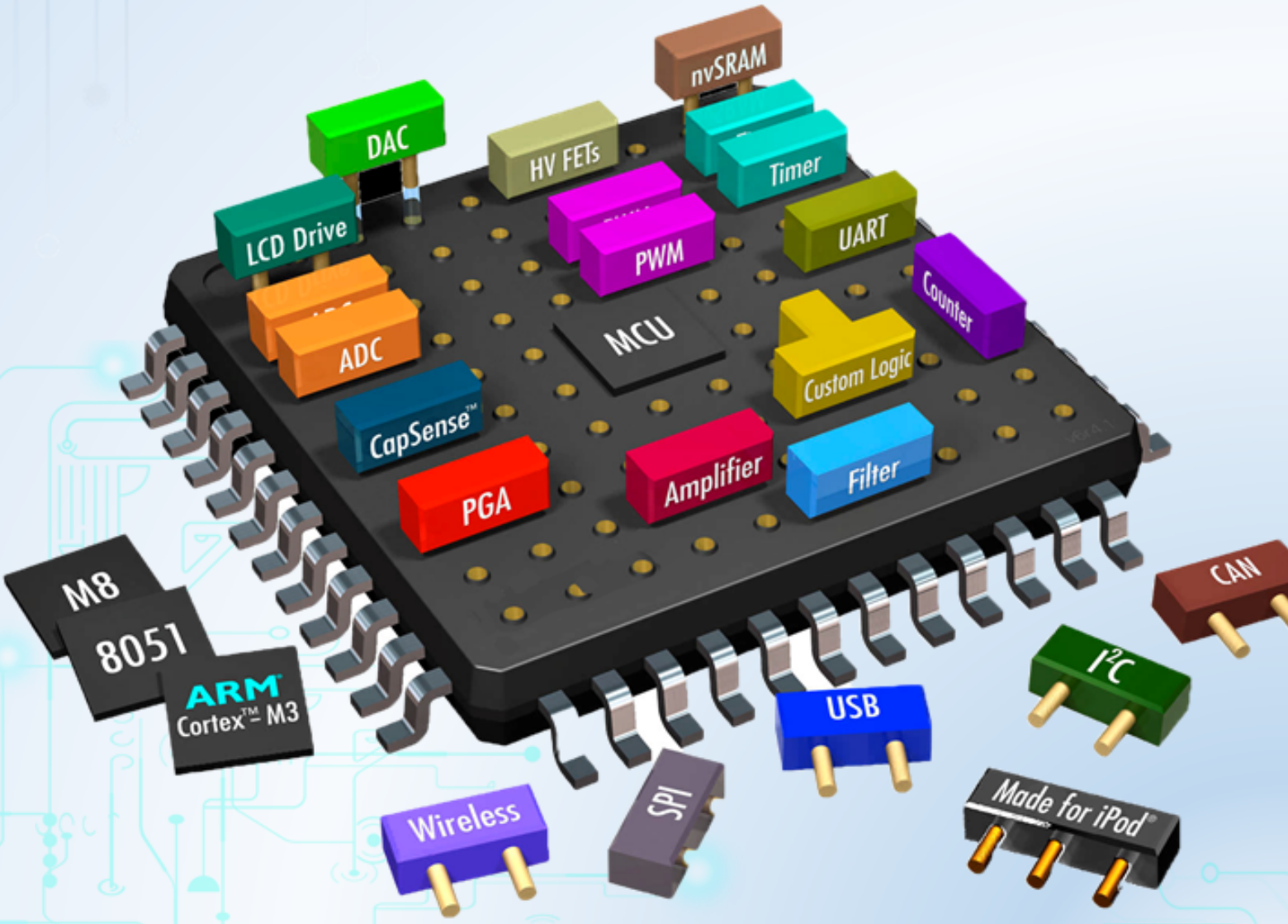



**T**utor: Dr. Le Trong Nhan
**S**tudent: Trần Cao Duy Trường
**2052299**

# Mục lục

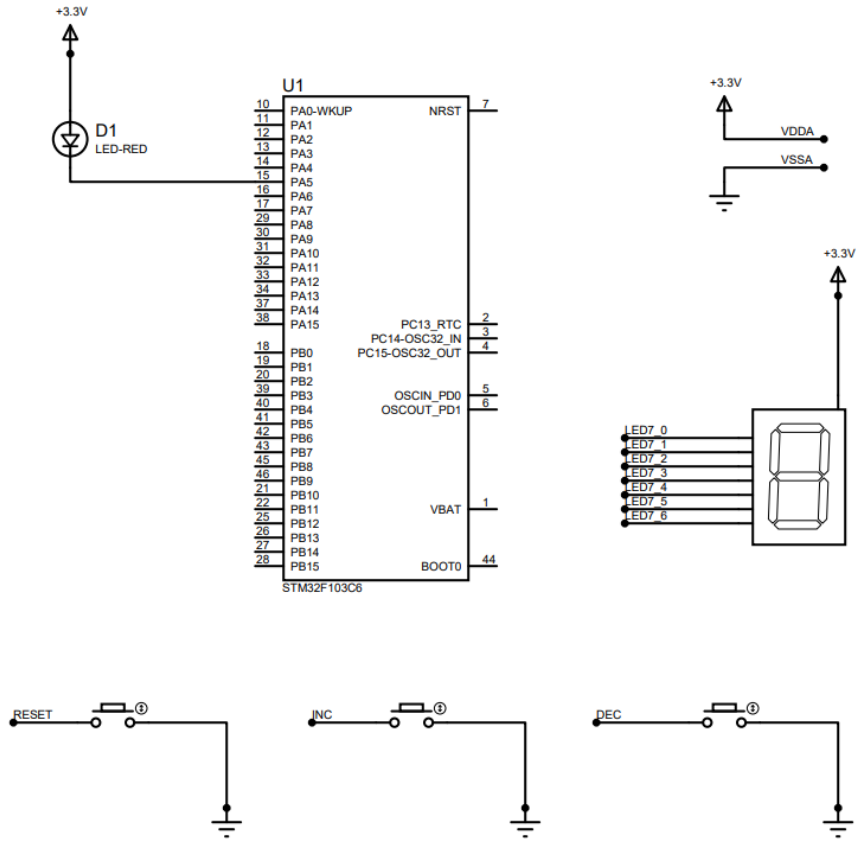
# CHƯƠNG 1

## MIDTERM 2022



# 1 Introduction

In this midterm project, a count-down system is designed and implemented in Proteus simulation. As it can be seen from Fig. 1.1, main components used in this project are the STM32F103C6, one LED, one LED7 segment and 3 different buttons.




*Hình 1.1*: *Proteus schematic for count-down system*

The main functions of the system are listed bellow:

- LED7 segment is used to display a counter ranging from 0 to 9.
- The **RESET** button is used to reset the counter value to 0. Meanwhile, the **INC** and **DEC** buttons are used to increase and decrease the counter value, respectively. There are two events need to handle for these buttons, including the normal-press and long-press.
- The D1 LED is blinking every second, which is normally used to monitor the execution of the system.

Students are supposed to following the section bellow, to finalize the project and fill in reports for their implementations. Some important notes for your midterm are listed bellow:

- The timer interrupt is 10ms. The value for counter is 9 (10 is also acceptable) when the pre-scaller is 7999.

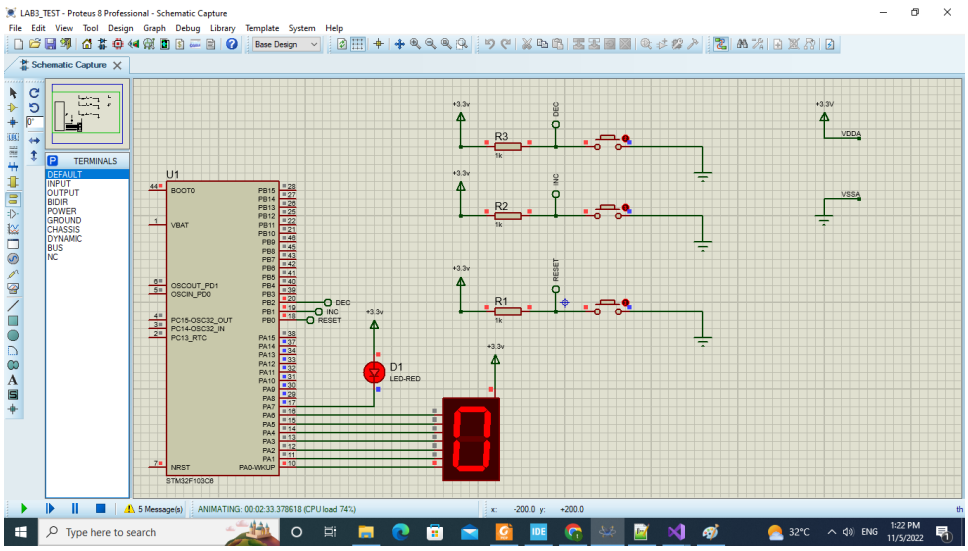
- All the buttons must be DEBOUNCING by using a timer interrupt service routing. A timeout for long press event is 3 seconds.
- There is no HAL_Delay() function in your source code. All the delay behavior must be based on a software timer.
- This report must be submitted with your answer.
- GitHub link for the source code and demo video link must be public access.

# 2 Implement and Report

## 2.1 Proteus schematic - 1 point

In this part, students propose the connection of the LED7 segment and 3 buttons to the STM32F103C6.

**Your report:** The schematic of your system is presented here. The screen can be captured and present in this part.




*Hình 1.2: Proteus design for this project*

## 2.2 State machine Step 1 - 2 points

A state machine is required in this step to perform just only the normal-press (or a button push) behavior of three buttons:

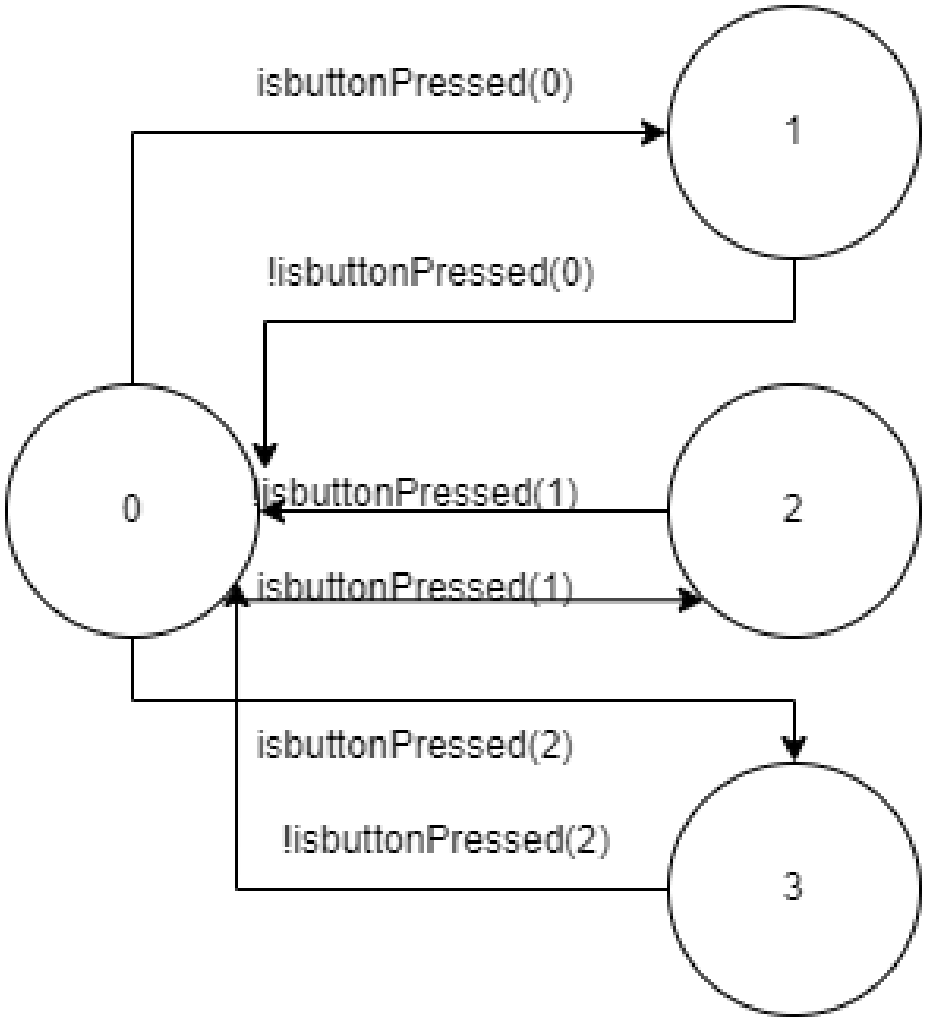
- Whenever the RESET is pressed, the counter value is 0.
- When INC is pressed, the counter is increased by 1. When counter is 9, it comes back to 0.

- When DEC is pressed, the counter is decreased by 1. When counter is 0, it rolls back to 9.

The value of the counter is displayed on the LED7 Segment.

**Your report:** Present your state machine in this part.

- Let the initial state of buttons be state 0 .
- When RST is pressed (button 0) we move to state 1 to process the counter reset value
- When INC is pressed (button 1) we move to state 2 to process the counter increment
- When DEC is pressed (button 2) we move to state 3 to process the counter decrement
- When buttons are not pressed we back to state 0 as initial state and wait for new signal




*Hình 1.3*: *Simple FSM machine*

**Your report:** Present a main function, which is used to implement the state machine. This function should be invoked in main().

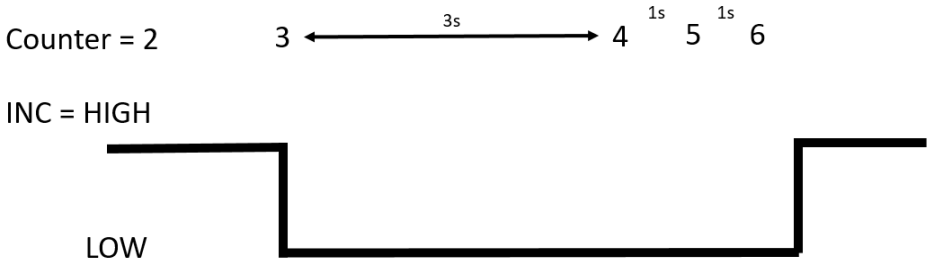
```
void fsm_simple_buttons_run(){
      for(int i=0;i<3;i++)
  {
    switch(i){
       case 0:
          if(isButtonPressed(0)){
          ledIdx=0;
          display7SEG(0);
          }
          break;
       case 1:
          if(isButtonPressed(1)){
          ledIdx++;
                    display7SEG(ledIdx);
          }
          break;
       case 2:
      if(isButtonPressed(2)){
        ledIdx--;
          display7SEG(ledIdx);
        }
      break;
    }

  }
}
```

Program 1.1: Implementation of the state machine

## 2.3 State machine Step 2 - 2 points

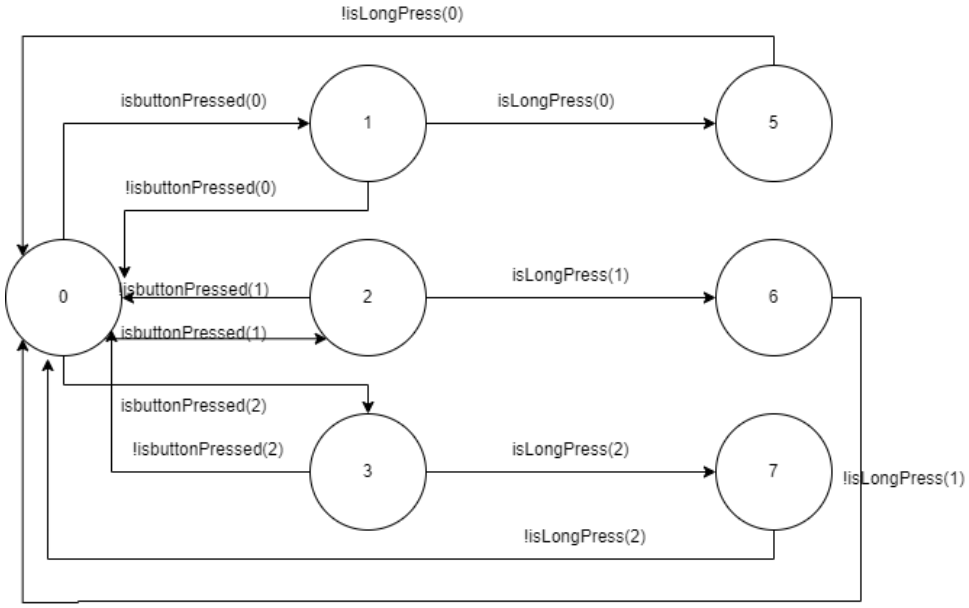
In this part, long-press events for INC and DEC buttons are added to the project. For a button, this event is raised after 3 seconds keep pressing the button.

When a long-press event is detected, the value of counter keeps changing every 1 second until the button is released. For example, the current value of counter is 2 and the INC button is pressed. The value of counter immediately increased by 1, or counter = 3. The INC button keeps pressing for 3 seconds, then the value of counter is 4. As long as the INC button is pressed, the value continues increasing **every 1 second**. This behavior is illustrated in the Figure bellow:




*Long press behavior for INC button*
The behaviors of the DEC button are reversed to the INC button. The value of counter is also roll back if it reaches 0 or 9.

**Your report:** Present your whole state machine when the long press events are added.




*Hình 1.4: FSM machine with LongPress*

- After first press if LongPress is detected it will move to corresponding state ( 5 for longRST , 6 for longINC, 7 for longDEC) to auto functioning.
- When buttons are not pressed anymore we back to state 0 as initial state and wait for new signal

**Your report:** Present a main function, which is used to implement additional states. Minor changes in the previous source code are note required to present here.

```
int TimeOutForKeyPress[NUM_OF_BUTTON] =  {100,100,100};
int button_flag[NUM_OF_BUTTON] = {0, 0, 0};
int detectLong[NUM_OF_BUTTON]={0,0,0};
```

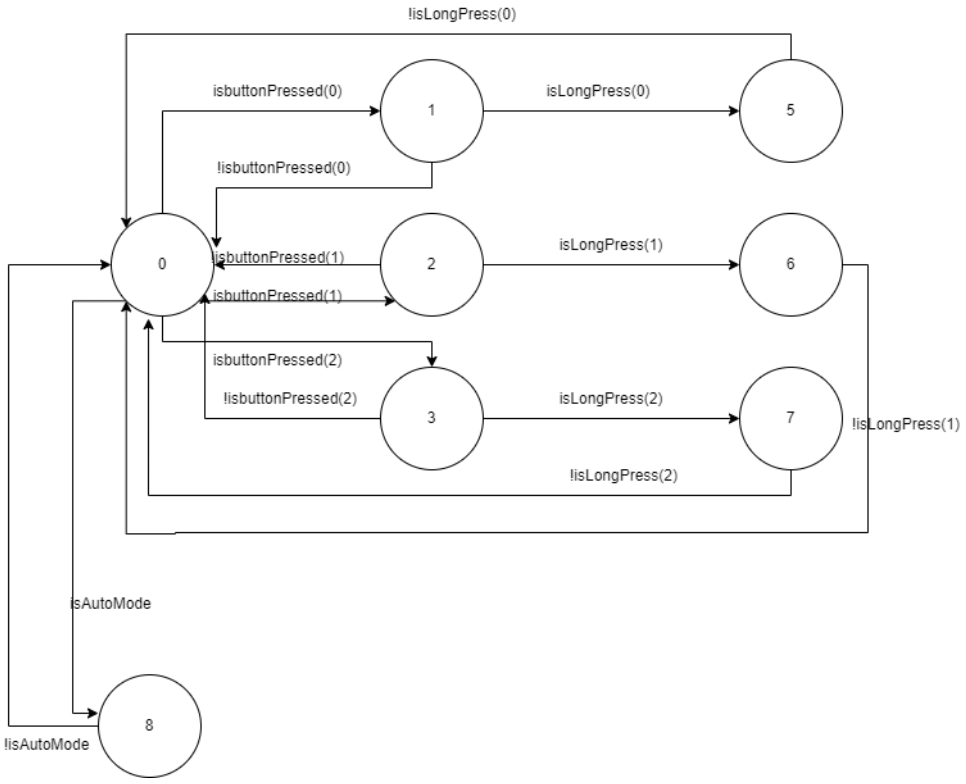
Program 1.2: code change for new FSM

To provide long press function, we set the value for TimeOutForKeyPress to 100 and make this value minus by 1 every timer run (10 ms) and raise the flag to enable buttons to be process again.
For the case of detecting first 3s . The first press appear to be single press in second 0 and long press only work from 3s which make the detectLong value be the conditional for enabling the function (only work with value 0 and >3). Otherwise long press wont be detected and detectLong value will be reset to 0.

## 2.4 State machine Step 3 - 2 points

Finally, where there is no button event after 10 seconds, the value of of counter is counted down and stopped at 0. If the INC or DEC are pressed again, the status of the system comes back to previous state, which is designed in Subsection 2 or 3.

**Your report:** Present your whole state machine for the 10s time-out event.




*Complete FSM machine*

- A state of auto countdown is added ( state 8). This state is enable only if no signal is send within 10s and will be disabled if press signal is detected from the button. This auto state will return to initial state if auto countdown is disabled

**Your report:** Present a main function, which is used to implement additional states. Minor changes in the previous source code are note required to present here.

```
void automode(){
  timeout10s--;
  if(timeout10s<0){
    ledIdx--;
    if(ledIdx<0){
      ledIdx=0;
      return;
    }
    display7SEG(ledIdx);
  }
}
```

Program 1.3: code change for complete FSM

**t**imeout10s is added and will be set to 10 if any button is pressed otherwise it will countdown every 1s and lower the ledIdx if timeout10s is <0.

## 2.5 Led Blinky for Debugging - 1 point

Finally, for many projects based on microcontroller, there is an LED keeps blinking every second. In this project, the LED connected to PA5 is used to perform this feature.

**Your report:** Present your solution and the source code for this feature. It can be very simple source code or a new state machine for this LED. If a state machine is used, please present it in the report.

```
int ledPin=OUT7_Pin;
void blinkLed(){
  HAL_GPIO_TogglePin(GPIOA,ledPin);

}
```

Program 1.4: code change for complete FSM

**T**he ledPin is assign and toggle function is blinked(). This fucntion will be called every 1s in the software timer.

## 2.6 Github and Demo

A link to your github presented the last commit of your project is provided in this section. This link contains all files in your STMCube project (configurations, header and source files)

https://github.com/DuyTruong123456/midtermmicro$_2$052299

The video is also in github or this is the drive link :

https://drive.google.com/file/d/1DsNY3ELuDtXd96pWM0rA6OeTO2GZMlVB/view?usp=sha

# 3 Extra exercise - Engineer mindset -1 point

In this course, we encourage you to obtain an innovative mindset to solve daily problem. In this question, we would expect you to write a C program to solve the following problem.

Suffix with Unit

EXample:

1 suffixWithUnit(123) => 123

2 suffixWithUnit(1234) => 1.234 Kilo

3 suffixWithUnit(12345) => 12.345 Kilo

4 suffixWithUnit(1234567) => 1.234567 Mega

5 suffixWithUnit(12345678) => 12.345678 Mega

Prototype

```
string suffixWithUnit(double number) {
}
```

How would you solve them? Please share your thinking to solve this problem and provide your answer.
Idea :
+Create an array with given suffix so that we can add on later.
+Create a MAX,MIN limit so that we dont exceed the array
+Set condition for number so that it cannot be smaller than 0
+Main logic of this is to set condition for number/1000 or number*1000 in order to increase or decrease the current index to change the suffix. If number/1000 is >1 this mean we can still change the suffix because this number is too big. Same apply to number*1000>1000 because this case number is to small

```
string suffixArr[]={ "yotto","zepto","atto","femto","pico",
   "nano","micro","mili", "" ,"kilo","mega","giga","tera","
   peta","exa"};
int idx=8;
int MIN=0;
int MAX=sizeof(suffixArr) / sizeof(int);
string suffixWithUnit (double number ) {
```

```
6       if(number<0)return "invalid";
7       else if(number/1000>=1&&idx<MAX){
8           number=number/1000;
9           idx++;
10      }
11      else if(number*1000<=1000&&idx>MIN){
12          number=number*1000;
13          idx--;
14      }
15      return to_string(number)+suffixArr[idx];
16  }
```