TRƯỜNG ĐẠI HỌC VINH

VIỆN KỸ THUẬT CÔNG NGHỆ

=====***=====

BÀI TIỂU LUẬN MÔN:

THIẾT KẾ HỆ THỐNG NHÚNG

Giản viên: TS. Mai Thế Anh

ThS.NCS. Lê Văn Chương

Họ và tên: Nguyễn Tuấn Tú

SBD: 46

Ngày sinh: 24/04/1999

MSSV: 1755252021600001

Lớp: 58K – Kỹ thuật ĐK & TĐH

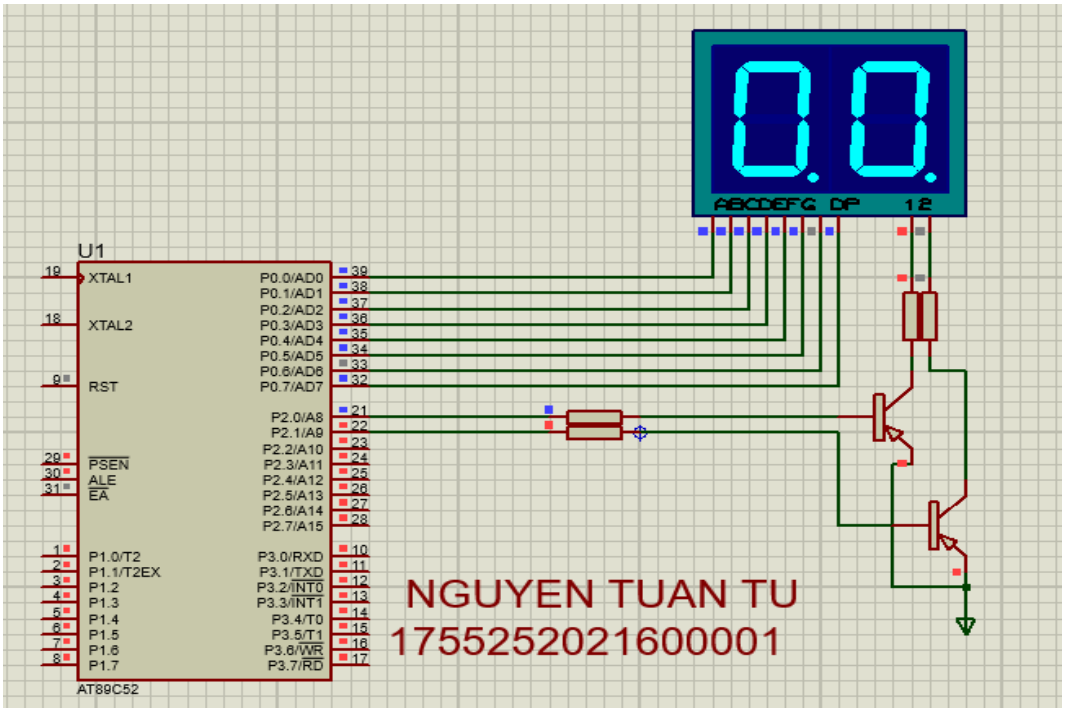
Nghệ An, 2021

Câu 1: Sử dụng vi điều khiển AT89C52, mô phỏng trên phần mềm Proteus và lập trình thực hiện các nhiệm vụ sau:

1. Hiển thị số 00 lên 2 LED 7 thanh nối vào cổng P2 theo phương pháp quét LED.
   - CODE

```
//NGUYEN TUAN TU
#include <REGX52.H>

charso[]=
{0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10,0x89,0x06,0xC7,0x40};

#define led1 P2_0
#define led2 P2_1
#define sang 0
#define tat 1

char i;

void delay(int time){
	while(time--);
}

void main(){
	led1 = led2 = tat;
	while (1){
		led1 = sang;
		P0 = so[0];
		delay(100);
		led1 = tat;

		led2 = sang;
		P0 = so[0];
		delay(100);
		led2 = tat;

	}
}
```

- MÔ PHỎNG PROTEUS

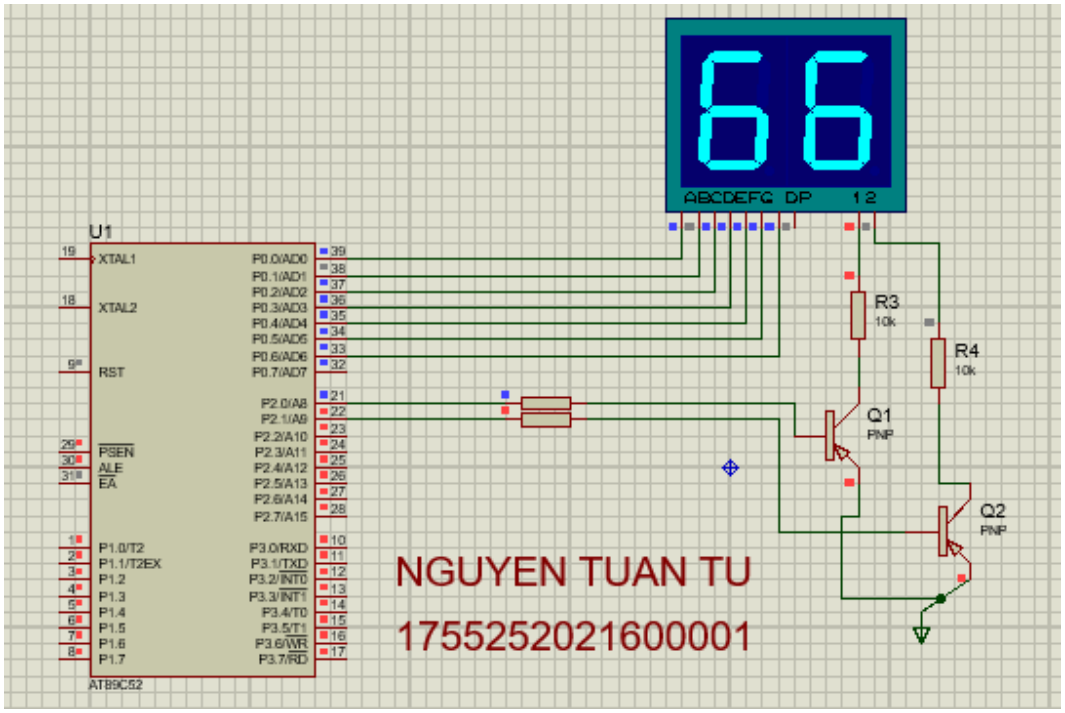


2. Tăng số đếm sau mỗi 500ms, nếu số đếm bằng "SBD+20" thì dừng lại (sử dụng timer để định thời gian)

```
//NGUYEN TUAN TU
#include <REGX52.H>

char so[] =
{0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10,0x89,0x06,0xC7,0x40};

char i;
int dem;
unsigned char chuc, donvi;
#define led1 P2_0
#define led2 P2_1
#define sang 0
#define tat 1


void delay(int time){
    while(time--);
}

void main(){
    led1 = led2 = tat;
    while (1){
            for (dem=0;dem<=66;dem++)
            {
```

```
                chuc = dem/10;
                donvi = dem%10;

                for (i=0; i<=10; i++)
                        {
                        led1 = sang;
                        P0 = so[chuc];
                        delay(1000);
                        led1 = tat;

                        led2 = sang;
                        P0 = so[donvi];
                        delay(1000);
                        led2 = tat;
                        }
                }
        }
}
```

- MÔ PHỎNG PROTEUS

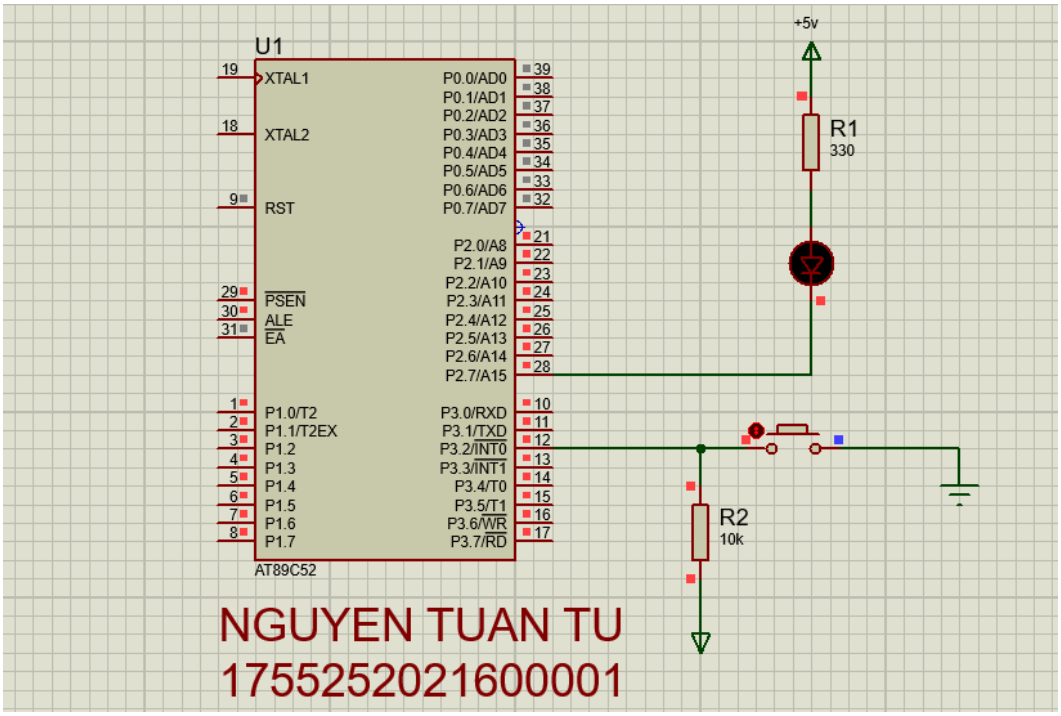


Câu 2: Sử dụng vi điều khiển AT89C52, mô phỏng trên phần mềm Proteus và lập trình thực hiện các nhiệm vụ sau:

1. Cấu hình ngắt ngoài INT0 ở chế độ ngắt sườn xuống;
   - CODE

```
//NGUYEN TUAN TU
#include<REGX52.H>
sbit BUTTON = P3^2;
sbit LED    = P2^7;
void Delay_ms(unsigned int t)
{
unsigned int x,y;
for(x=0;x<t;x++)
{
for(y=0;y<123;y++);
}
}
void main (void)
{
 while(1)
 {
        if(BUTTON == 0)
        {
                Delay_ms(100);
                if(BUTTON == 0)
                {
                        while(BUTTON == 0) { }
                        LED = !LED;
                }
        }
 }
}
```

- MÔ PHỎNG PROTEUS



2. Đếm số lần nút bấm nút nối vào chân INT0 được bấm, hiển thị kết quả lên 2 LED 7 thanh nối vào cổng P2 theo phương pháp quét LED (nếu số lần bấm bằng "SBD+10" thì quay về 0).

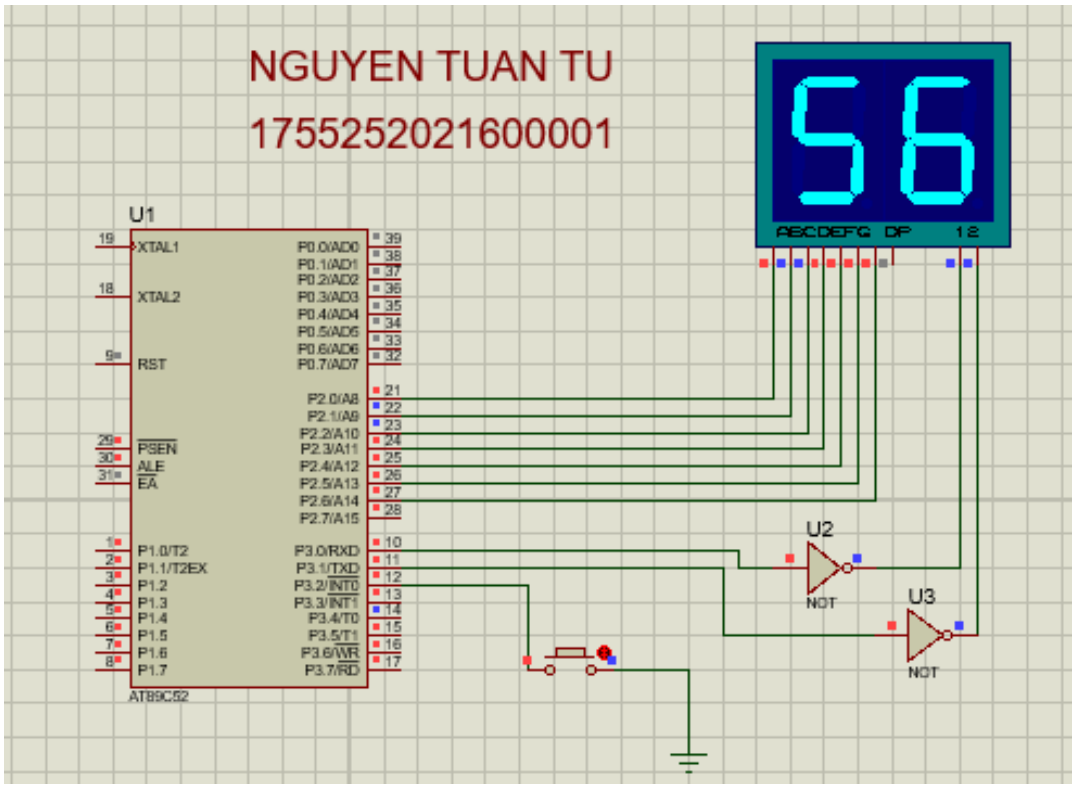
- CODE

```
//NGUYEN TUAN TU
#include <REGX52.H>

#define OUT    P2
#define D1              P3_0
#define D2              P3_1
#define D3              P3_4
#define D4              P3_5
#define BUT             P3_2

unsigned char
ma7thanh[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

char a,b;

unsigned char dem;

void delay(unsigned int t)
{
    unsigned int i,j;
    for(i=0;i<t;i++)
```

```
    for(j=0;j<100;j++);
}

void quet_led()
{
    D1 = 0;
    OUT = ma7thanh[a/10];
    delay(1);
    OUT = 0xff;
    D1 = 1;
    D2 = 0;
    OUT = ma7thanh[a%10];
    delay(1);
    OUT = 0xff;
    D2 = 1;
    D3 = 0;
    OUT = ma7thanh[b/10];
    delay(1);
    OUT = 0xff;
    D3 = 1;
    D4 = 0;
    OUT = ma7thanh[b%10];
    delay(1);
    OUT = 0xff;
    D4 = 1;
}
void nut_nhan()
{
    if(!BUT){
            a++;
            if(a >56)a = 0;
            while(!BUT);

    }
}

void main()
{
    delay(500);
    while(1)
            {
                    nut_nhan();
                    quet_led();
                    dem++;
                    if(dem > 1)
```

```
                    {
                        b++;
                        dem = 0;
                        if(b > 38)b = 0;
                    }
                }
    }
```

- MÔ PHỎNG PROTEUS



Câu 3: Sử dụng vi điều khiển AT89C52, thực hiện các nhiệm vụ sau:

1. Vẽ sơ đồ mô phỏng trên Proteus ghép nối với LCD theo chế độ 4bit, hiển thị họ tên, mã số sinh viên lên LCD.

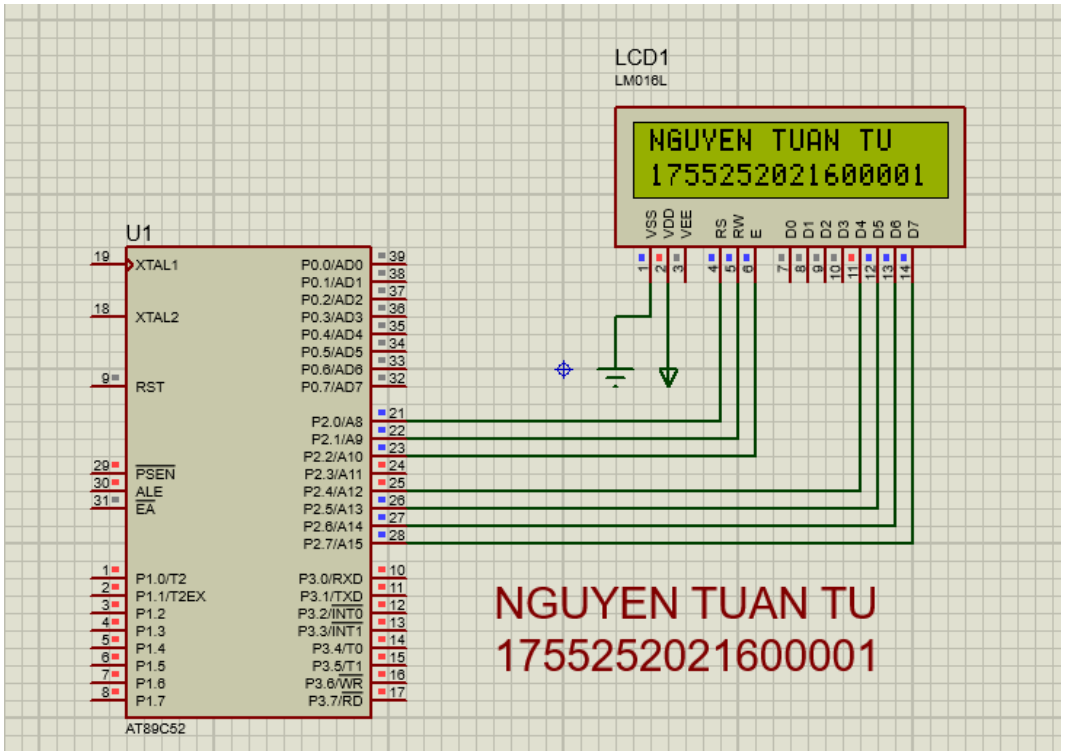
- CODE

```
//NGUEN TUAN TU
#include <REGX52.H>
#define LCD_RS P2_0
#define LCD_RW P2_1
#define LCD_EN P2_2
#define LCD_D4 P2_4
#define LCD_D5 P2_5
#define LCD_D6 P2_6
#define LCD_D7 P2_7

void delay_us(unsigned int t){
    unsigned int i;
```

```
	for(i=0;i<t;i++);
}
void delay_ms(unsigned int t){
	unsigned int i,j;
	for(i=0;i<t;i++)
	for(j=0;j<125;j++);
}
//Chuong trinh giao tiep LCD 16x2 4bit
void LCD_Enable(void){
	LCD_EN =1;
	delay_us(3);
	LCD_EN=0;
	delay_us(50);
}
//Ham Gui 4 Bit Du Lieu Ra LCD
void LCD_Send4Bit(unsigned char Data){
	LCD_D4=Data & 0x01;
	LCD_D5=(Data>>1)&1;
	LCD_D6=(Data>>2)&1;
	LCD_D7=(Data>>3)&1;
}
// Ham Gui 1 Lenh Cho LCD
void LCD_SendCommand(unsigned char command){
	LCD_Send4Bit(command >>4);// Gui 4 bit cao
	LCD_Enable();
	LCD_Send4Bit(command); //Gui 4 bit thap
	LCD_Enable();
}


// Ham Khoi Tao LCD
void LCD_Init(){
	LCD_Send4Bit(0x00);
	delay_ms(20);
	LCD_RS=0;
	LCD_RW=0;
	LCD_Send4Bit(0x03);
	LCD_Enable();
	delay_ms(5);
	LCD_Enable();
	delay_us(100);
	LCD_Enable();
	LCD_Send4Bit(0x02);
	LCD_Enable();
	LCD_SendCommand( 0x28 ); // giao thuc 4 bit, hien thi 2 hang, ki tu 5x8
```

```
	LCD_SendCommand( 0x0c); // cho phep hien thi man hinh
	LCD_SendCommand( 0x06 ); // tang ID, khong dich khung hinh
	LCD_SendCommand(0x01); // xoa toan bo khung hinh
}
void LCD_Gotoxy(unsigned char x, unsigned char y){
	unsigned char address;
	if(!y)address=(0x80+x);
	else address=(0xc0+x);
	delay_us(1000);
	LCD_SendCommand(address);
	delay_us(50);
}
void LCD_PutChar(unsigned char Data){//Ham Gui 1 Ki Tu
	LCD_RS=1;
	LCD_SendCommand(Data);
	LCD_RS=0 ;
}
void LCD_Puts (char *s){//Ham gui 1 chuoi ky tu
	while (*s){
		LCD_PutChar(*s);
		s++;
	}
}
void main(){
	LCD_Init();//Khoi tao LCD
	LCD_Gotoxy(0,0);//Tro toi vi tri
	LCD_Puts("NGUYEN TUAN TU");

	LCD_Gotoxy(0,1);//Tro toi vi tri
	LCD_Puts("1755252021600001");
	while(1);
}
```

- MÔ PHỎNG PROTEUS



2. Vẽ sơ đồ mô phỏng trên Proteus, lập trình hiển thị “Họ tên và mã số sinh viên” qua chuẩn truyền thông UART.

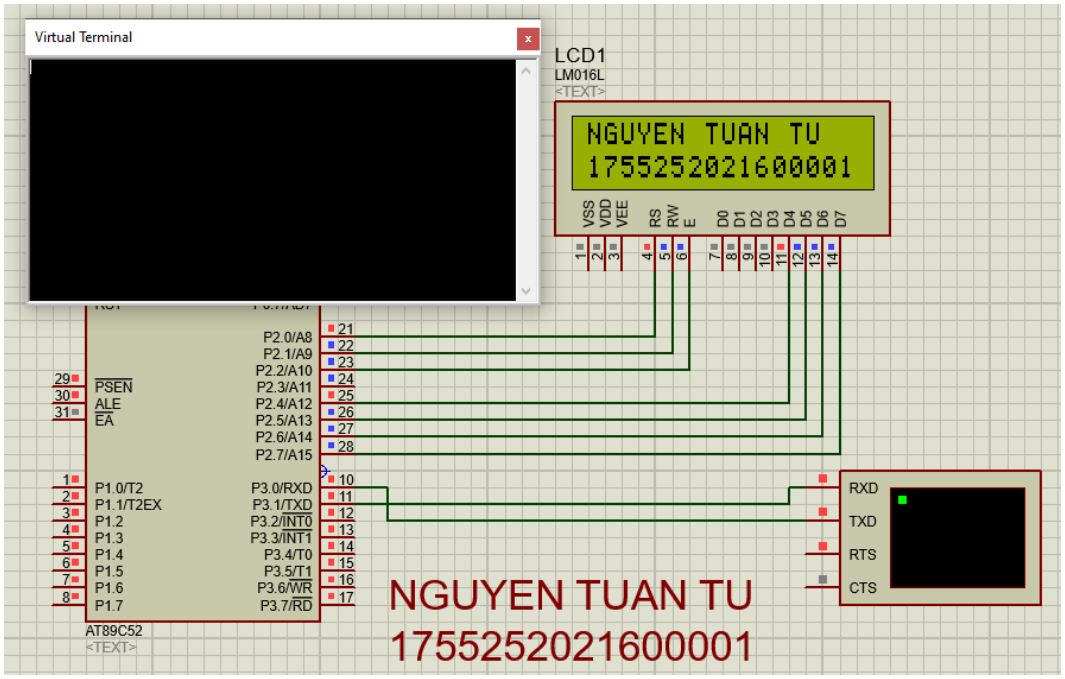
- CODE

```
//NGUYEN TUAN TU
#include <REGX52.H>

sbit rs = P2^0;
sbit rw = P2^1;
sbit e = P2^2;

void delay(unsigned int t)
{
	unsigned int i, j;
	e = 1;
	for(i = 0; i < t; i++)
	for(j = 0; j < 1275; j++);
	e = 0;
}

void cmd1(unsigned char ch)
{
	P2 = ch;
	rs = 0;
	rw = 0;
```

```
        delay(10);
}

void dat1(unsigned char ch)
{
        P2 = ch;
        rs = 1;
        rw = 0;
        delay(10);
}

void cmd(unsigned char a)
{
        unsigned char x;
        x = a & 0xF0;
        cmd1(x);
        x = (a << 4) & 0xF0;
        cmd1(x);
}

void dat(unsigned char a)
{
        unsigned char x;
        x = a & 0xF0;
        dat1(x);
        x = (a << 4) & 0xF0;
        dat1(x);
}

void main(void)
{
        unsigned char mybyte;
        cmd(0x28);
        cmd(0x01);
        cmd(0x0C);
        cmd(0x80);
        cmd(0x06);
        TMOD = 0x20;
        TH1 = 0xFD;
        SCON = 0x50;
        TR1 = 1;
        while(1)
        {
                while(RI == 0);
                mybyte = SBUF;
```

```
                    dat(mybyte);
                    RI = 0;
                    if(mybyte == 0x08)
                        cmd(0x01);
                    if(mybyte == 0x0D)
                        cmd(0xC0);
                }
            }
```

- MÔ PHỎNG PROTEUS



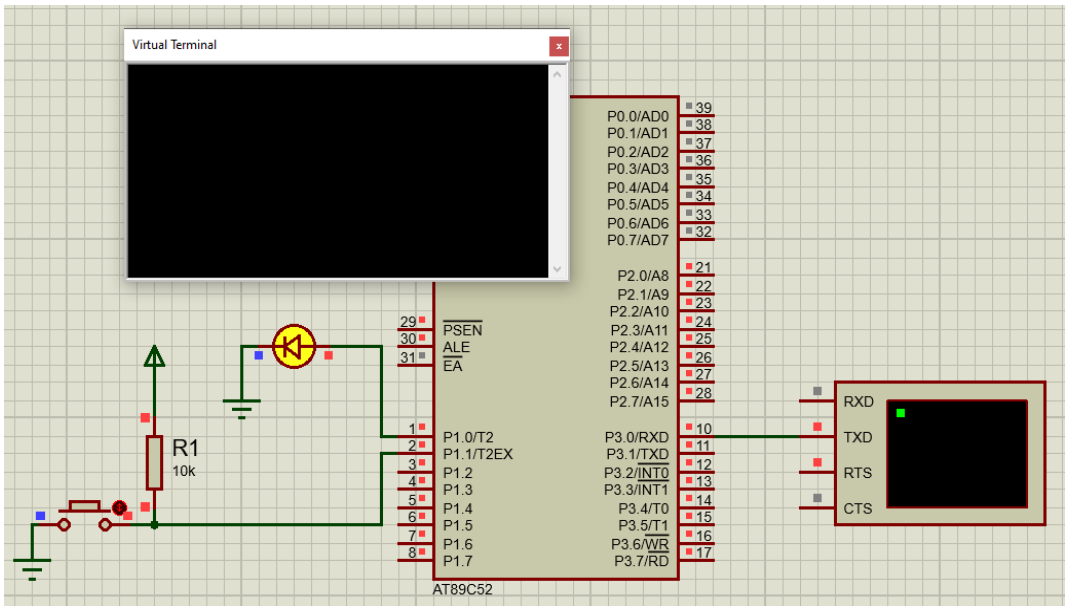
Câu 4: Sử dụng vi điều khiển AT89C52, vẽ sơ đồ mô phỏng trên Proteus ghép nối với Led D1 qua cổng P1.2, BUTTON B1 qua cổng P1.3. Sử dụng hệ điều hành RTX51 lập trình ngắt USART, tast BUTTON, tast LED. Thực hiện gửi "signal" từ ngắt USART và task BUTTON đến tast LED. Task LED thực hiện đảo trạng thái của Led D1 khi nhận được tín hiệu task khác gửi tới.

- CODE

```
//NGUYEN TUAN TU
#include <REGX52.H>
#include <RTX51TNY.H>     //Su dung thu vien RTX51 Tiny Real-Time
#define INIT 0              //Dinh nghia INIT = 0
#define DO 1                //Dinh nghia DO = 1
#define BUTT 2              //Dinh nghia BUTTTON = 2
sbit LED_DO = P1^0;      //Dinh nghia chan LED_DO
sbit BUTTON = P1^1;    //Dinh nghia chan BUTTON
void USART(void) interrupt 4 //Ngat nhan USART
{
 if(RI)//Flag nhan duoc ki tu
 { //Clear flag
 RI=0; //Nhan ki tu
 isr_send_signal(DO);       //Gui signal cho task DO
 }
}
//================================
void Startup(void) _task_ INIT
{
 SCON=0x52;                //USART che do 1
 TMOD=0x21;                //Timer 1 mode 2
 TH1=TL1=-3;               //baudrate 9600
 TR1=1;
 IE=0x90;                  //Ngat USART
 os_create_task (DO);       //Tao Task_Led_Do
 os_create_task (BUTT);      //Tao Task BUTTON
 os_delete_task (INIT);      //Xoa Task hien tai (Task 0)
}
void Task_Led_Do(void) _task_ DO
{
 while(1)
 {
 os_wait2(K_SIG ,50);       //Cho signal voi time out 50 ticks
 LED_DO ^= 1;               //Tao trang thai Led Do
 }
}
void Task_BUTTON(void) _task_ BUTT
{
 while(1)
 {
 if(BUTTON == 0)            //Nhan nut nhan = 0
 {
 os_send_signal(DO);        //Gui signal cho task DO
```

```
        while(BUTTON==0);          //Cho nut nhan = 1(Chong nhieu)
        }
        os_wait2(K_TMO, 10);       //Cho 10 ticks = 100ms
        }
      }
```

- MÔ PHỎNG PROTEUS



Câu 5:

1. Trình bày quy trình thiết kế hệ thống nhúng sử dụng vi điều khiển.

Trong luồng thiết kế từ trên xuống, chúng ta bắt đầu với việc đưa ra các yêu cầu thiết kế (Requirements) của hệ thống. Trong bước tiếp theo, đặc tả kỹ thuật (specification), chúng ta tạo ra một bản mô tả chi tiết hơn về những gì chúng ta muốn thiết kế. Tuy nhiên, bản đặc tả chỉ nêu ra cách hệ thống hoạt động mà không phải cách nó được xây dựng thế nào. Các chi tiết bên trong của hệ thống bắt đầu hình thành khi chúng ta phát triển kiến trúc – một sơ đồ khối chỉ ra cấu trúc của hệ thống dưới dạng các khối chức năng. Khi chúng ta xác định các khối chức năng chính cấu thành lên hệ thống chúng ta có thể thiết kế các khối chức năng đó, bao gồm cả các khối chức năng phần mềm và bất kỳ khối chức năng phần cứng chuyên dụng nào mà chúng ta cần. Dựa trên các khối chức năng đó, cuối cùng chúng ta có thể tích hợp chúng lại với nhau để xây dựng một hệ thống hoàn chỉnh.

2. Hệ điều hành thời gian thực (RTOS). Ưu điểm, nhược điểm và ứng dụng của hệ điều hành thời gian thực trong thiết kế các hệ thống nhúng.

Hệ điều hành thời gian thực – RealTime Operating Systems(RTOS), là phần mềm điều khiển chuyên dụng thường được dùng trong những ứng dụng điện toán nhúng có tài nguyên bộ nhớ hạn chế và yêu cầu ngặt nghèo về thời gian đáp ứng tức thời, tính sẵn sàng cao và khả năng tự kiểm soát một cách chính xác. RTOS là một kiến trúc tốt hơn các hệ điều hành nhúng khác vì nó có thể đáp ứng các yêu

cầu ràng buộc về thời gian (ràng buộc thời gian cứng – không cho phép thời gian xử lý chậm và ràng buộc thời gian mềm – cho phép xử lý chậm trong một lân cận nào đó), chịu lỗi cao và cho phép xử lý đa nhiệm (định danh tiến trình và độ ưu tiên - thông qua một số phương thức: semaphores, mailbox, queue...).

- Ưu điểm:

- Ưu điểm lớn của RTOS là xử lý nhanh chóng vì thế nó sẽ dành cho các thiết bị đòi hỏi khả năng xử lý có độ trễ thấp nhất có thể. Lợi ích nó đem lại bao gồm đa nhiệm tốt, ưu tiên các nhiệm vụ và quản lý chia sẻ các tài nguyên. Ngoài ra nó cũng không đòi hỏi nhiều về tài nguyên hay bộ nhớ RAM quá lớn

- Thay đổi bất kỳ tác vụ nào trong Round Robin hoặc lập lịch theo hàng đợi đều có một nhược điểm là ảnh hưởng đến tổng thể toàn bộ các tác vụ.

- Thay đổi tới tác vụ có độ ưu tiên thấp hơn trong RTOS không ảnh hưởng đến thời gian đáp ứng của các tác vụ có độ ưu tiên cao hơn. - RTOS được sử dụng rộng rãi và là giải pháp thật sự cần thiết cho các hệ thống yêu cầu ràng buộc về thời gian đáp ứng (ràng buộc thời gian cứng, mềm)

- Nhược điểm:

- RTOS cần thêm một số thời gian để xử lý các thông tin về tác vụ trước và sau khi đưa nó vào xử lý trong CPU nên hiệu suất sử dụng bị ảnh hưởng (do yêu cầu tính ràng buộc về thời gian nên độ phức tạp cao và xử lý cần đảm bảo độ an toàn). Chi phí cao khi mua các sản phẩm thương mại.

- Ứng dụng:

- Hệ điều hành thời gian thực dành cho các hệ thống nhúng đòi hỏi khả năng xử ý dữ liệu có độ trể thấp, những lợi ích mà nó mang lại bao gồm khả năng đa nhiệm, ưu tiên các nhiệm vụ và quản lý việc chia sẽ tài nguyên giữa các tác vụ phúc tạp.

- Hệ điều hành này được sử dụng phổ biến rộng rải trong ngành hàng không, nhiều ngành công nghiệp và các thiết bị chăm sóc sức khỏe IoT.

- một số ứng dụng cử thể sau đây:

+ xử lý ngắt

+ bộ lập lịch

+ dịch vụ thời gian

+ dịch vụ quản lý thiết bị

+ dịch vụ quản lý bộ nhớ

+ dịch vụ quản lý kết nối