BÁO CÁO TIỂU LUẬN

Tên học phần: Thiết kế hệ thống nhúng

Họ và tên: Nguyễn Như Truyền

Mssv: 1755252021600017

Lớp: 58K KTĐK&TĐH

Sbd: 44

Bài làm

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

1. Hiện thị số 00 lên 2 LED nối vào cổng P2 theo phương pháp quét LED

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

Giải

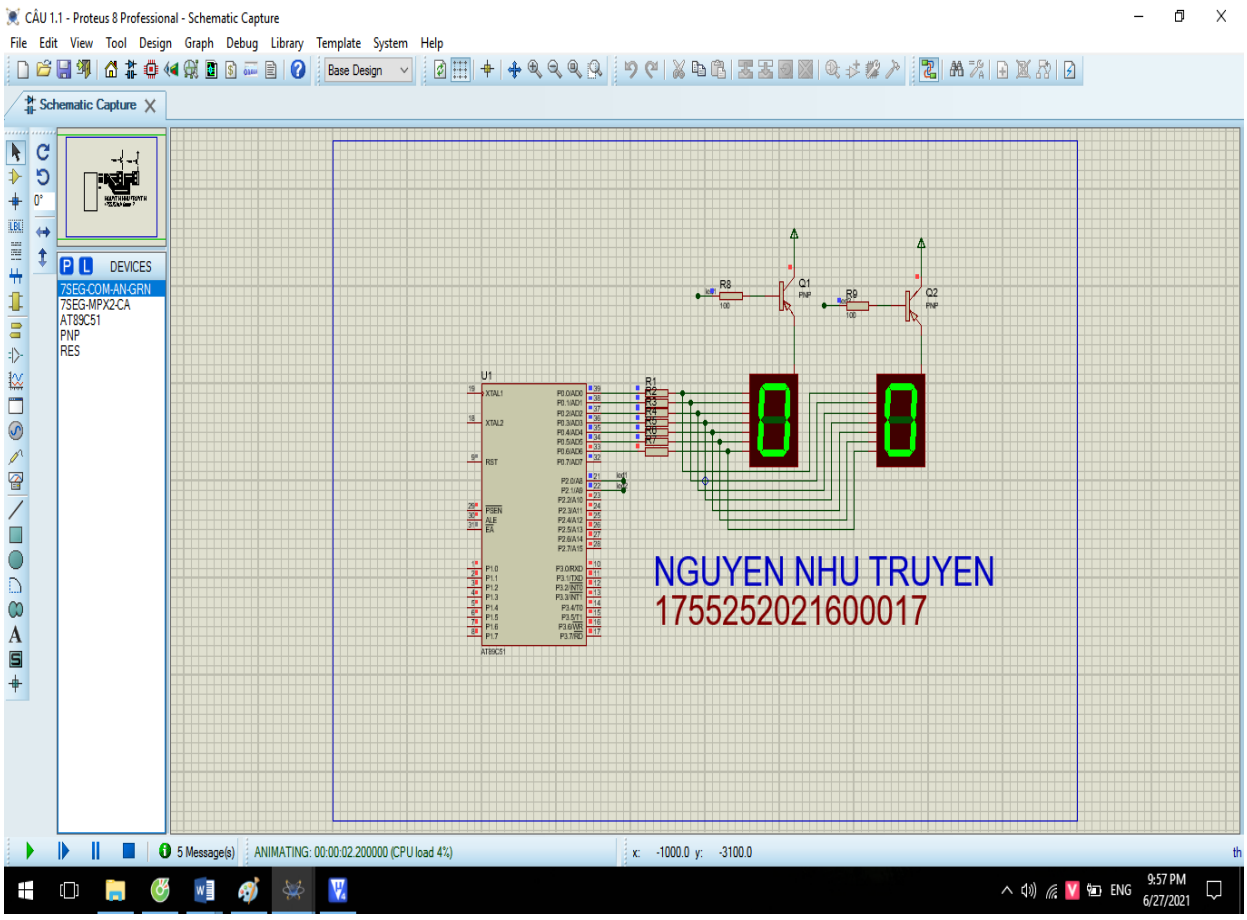
Câu 1.1

Viết code trên KeilC:

```
#include <REGX51.H>

Char so[]={0x40,0x79,0x24,0x30,0x19,0x12,0x89,0x06,0xC7,0x40};

//

#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(300000);
led1 = tat;
led2 tat;
P0 = so[0];
delay(300000);
led2 = sang;
                }
}
```

Mô phỏng trên phần mềm Proteus



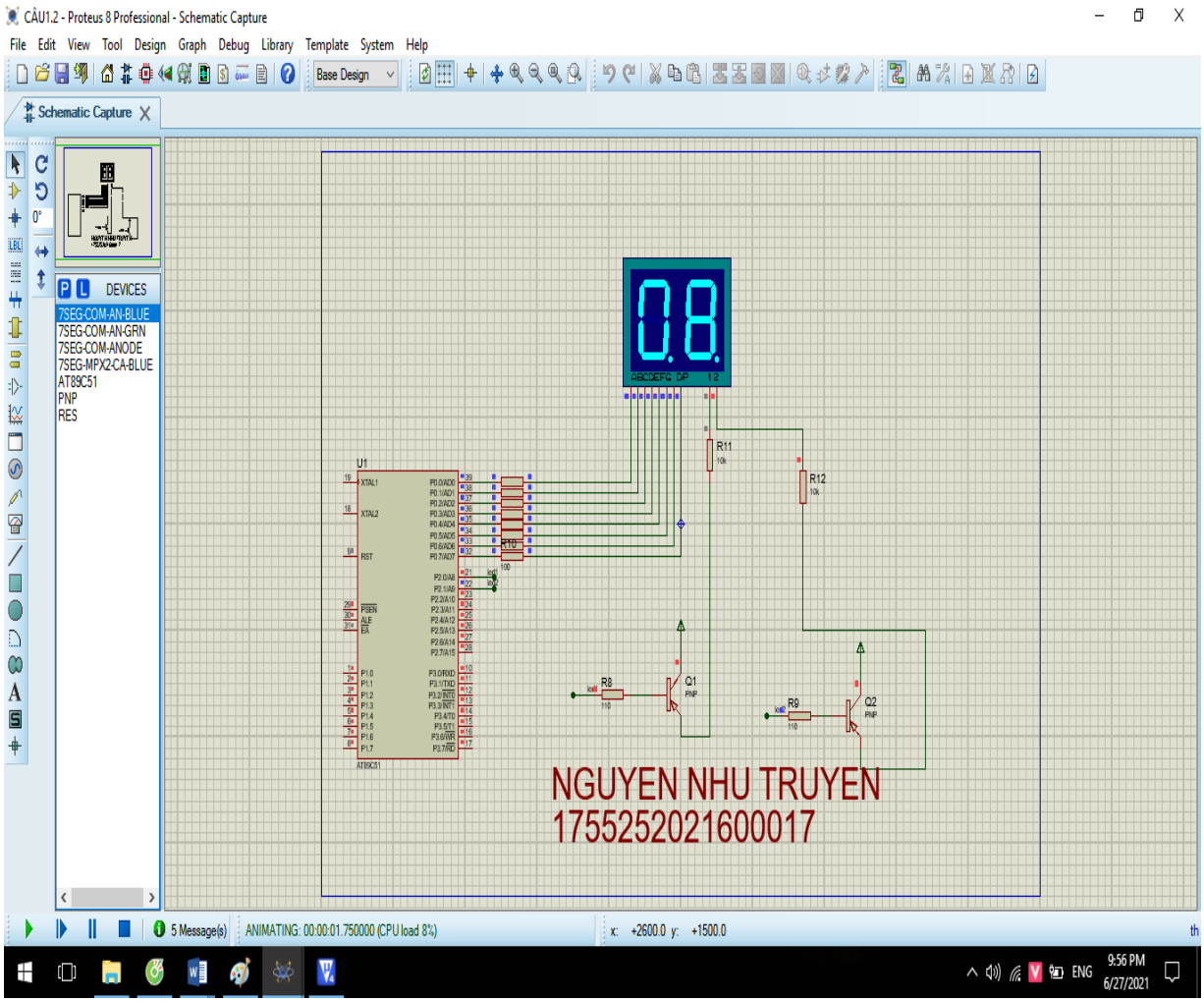
Câu 1.2

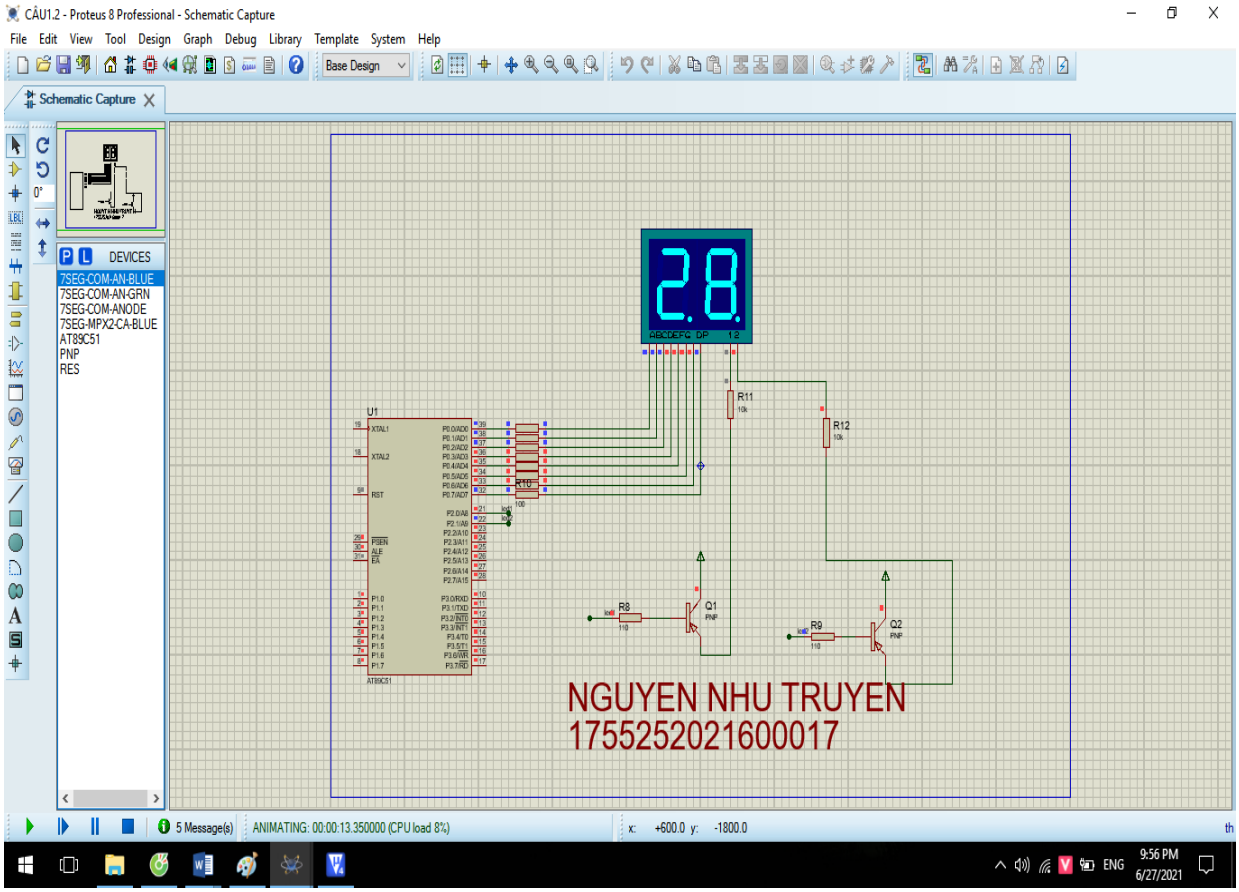
Viết code trên keilC

```
#include <REGX51.H>

Char so[]={0x40,0x79,0x24,0x30,0x19,0x12,0x89,0x06,0xC7,0x40};

//

char i;

int dem; j

unsigned 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<40;dem++){

//tach chu so

chuc = dem/10;

donVi = dem%10;

for (i = 0 ; i<=10;i++){

led1 = sang;

P0   = so[chuc];

delay(1000);
```

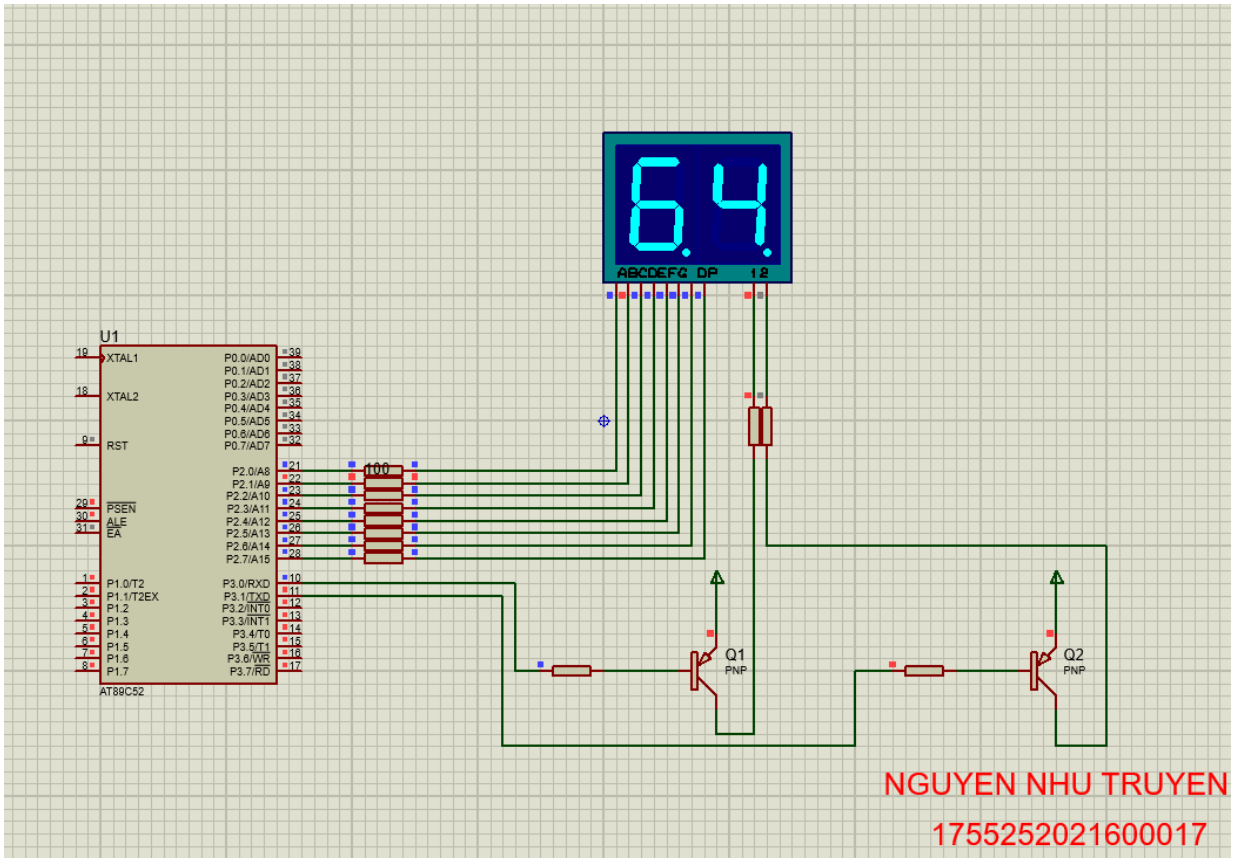
```
led1 = tat;
led2 = sang;
P0 = so[donVi];
delay(1000);
led2 = tat;
}
}
}
}
```

Mô phỏng trên phần mềm Proteus:





Đến SBD+20 thì dừng lại (SBD của em là 44)



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

1. Cấu hình ngắt nút ngoài INT0 ở chế độ ngắt sườn xuống.

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

Giải

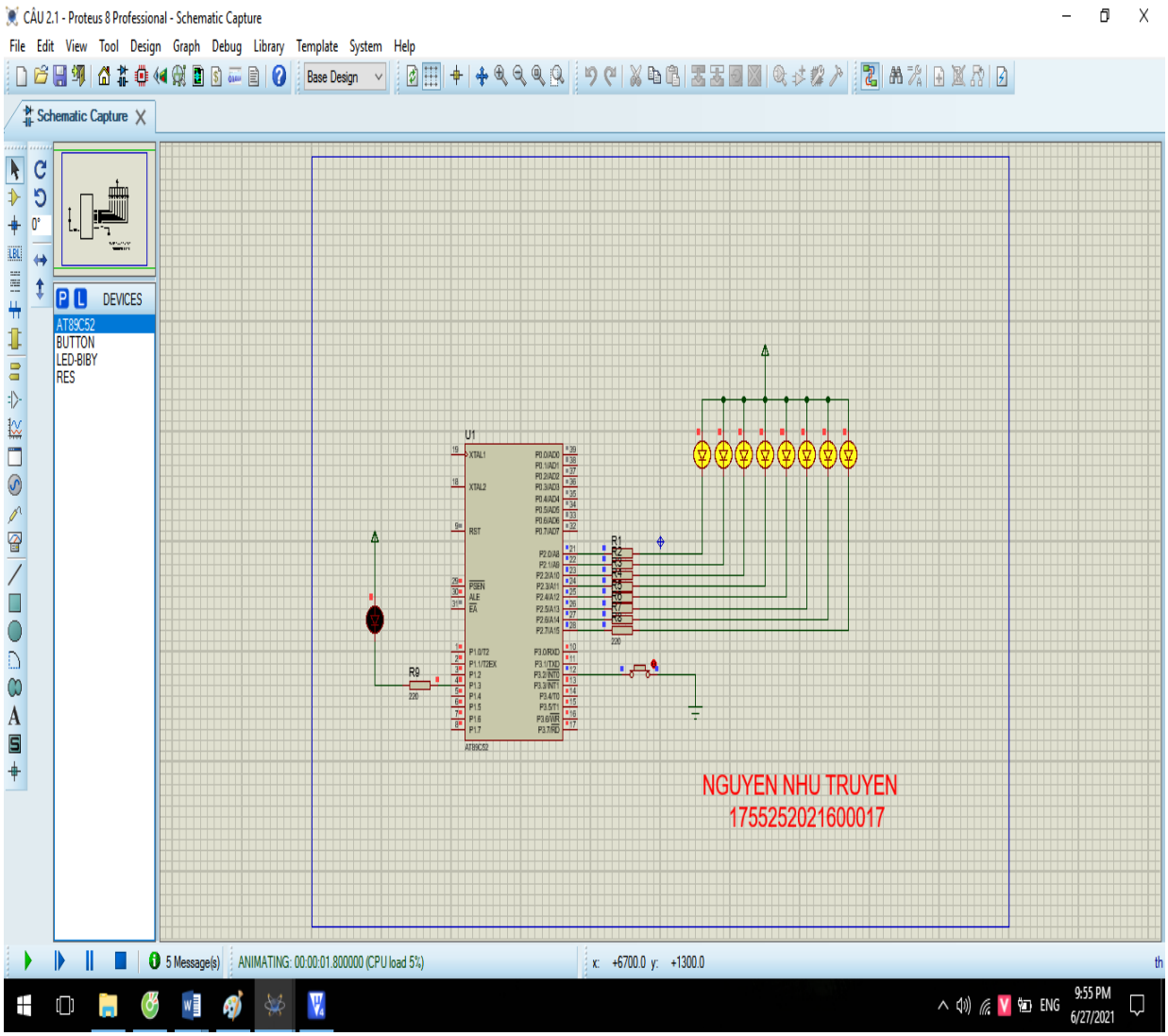
Câu 2.1:

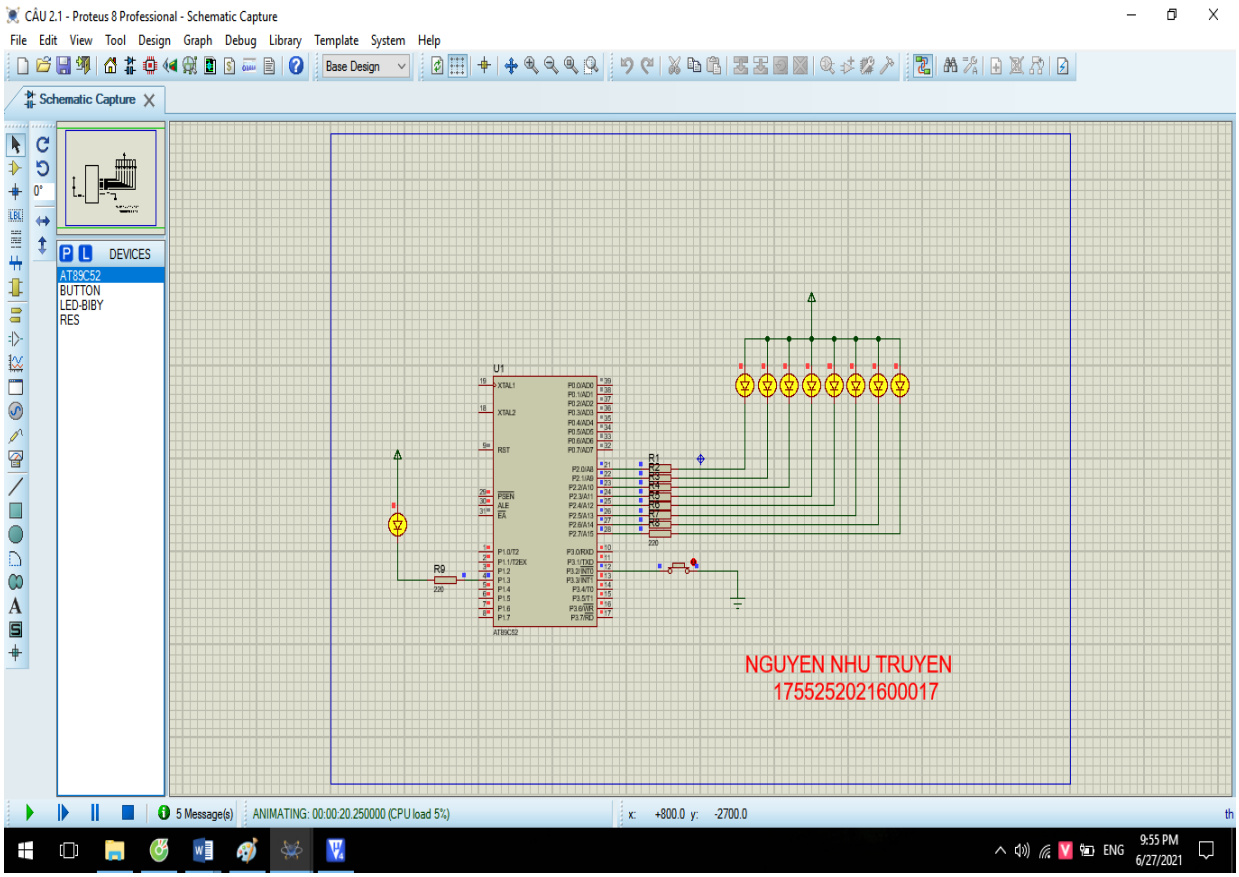
Viết code trên KeilC

```
#include <REGX52.H>

void delay_ms(int ms){while(ms--){

TH0 = 0Xfc;

 TL0 = 0x18;

R0 = 1;

while(!TF0);

TF0 = 0;

TR0 = 0;

 }

}

void main(){

EX0 = 1; //cho phep ngat ngoai 0

IT0 = 1; //chon kieu ngat theo suon

EA  = 1; //cho phep ngat ngoai cuc

while(1){

P2 = 0;

delay_ms(1000);

P2 = 0xff;
```

```
delay_ms(1000);
}
}
void ngat() interrupt 0{
long a = 50000;
P1_3 = 0;
while(a--){};
P1_3 = 1;
}
```

Mô phỏng trên phần mềm Proteus





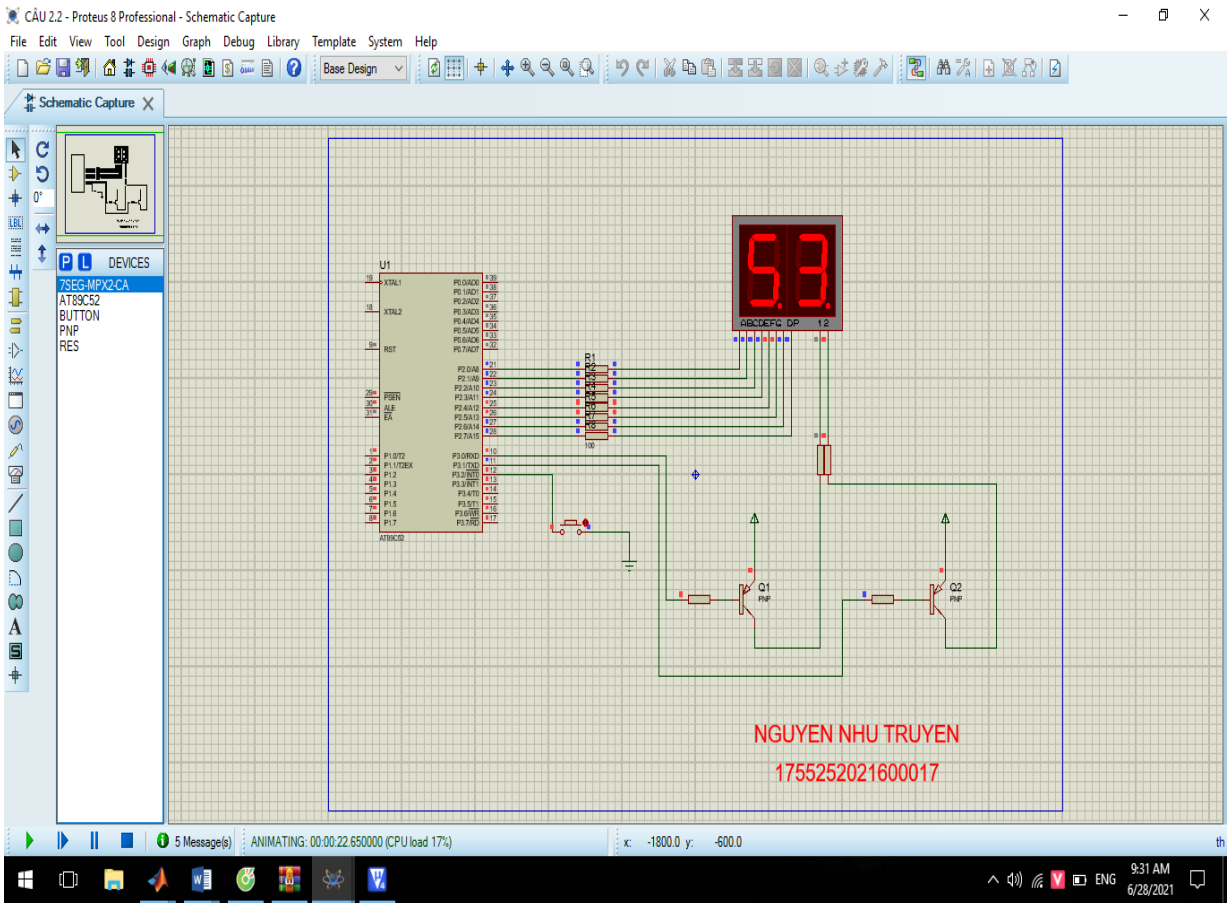
Câu 2.2

Code viết trên KeilC

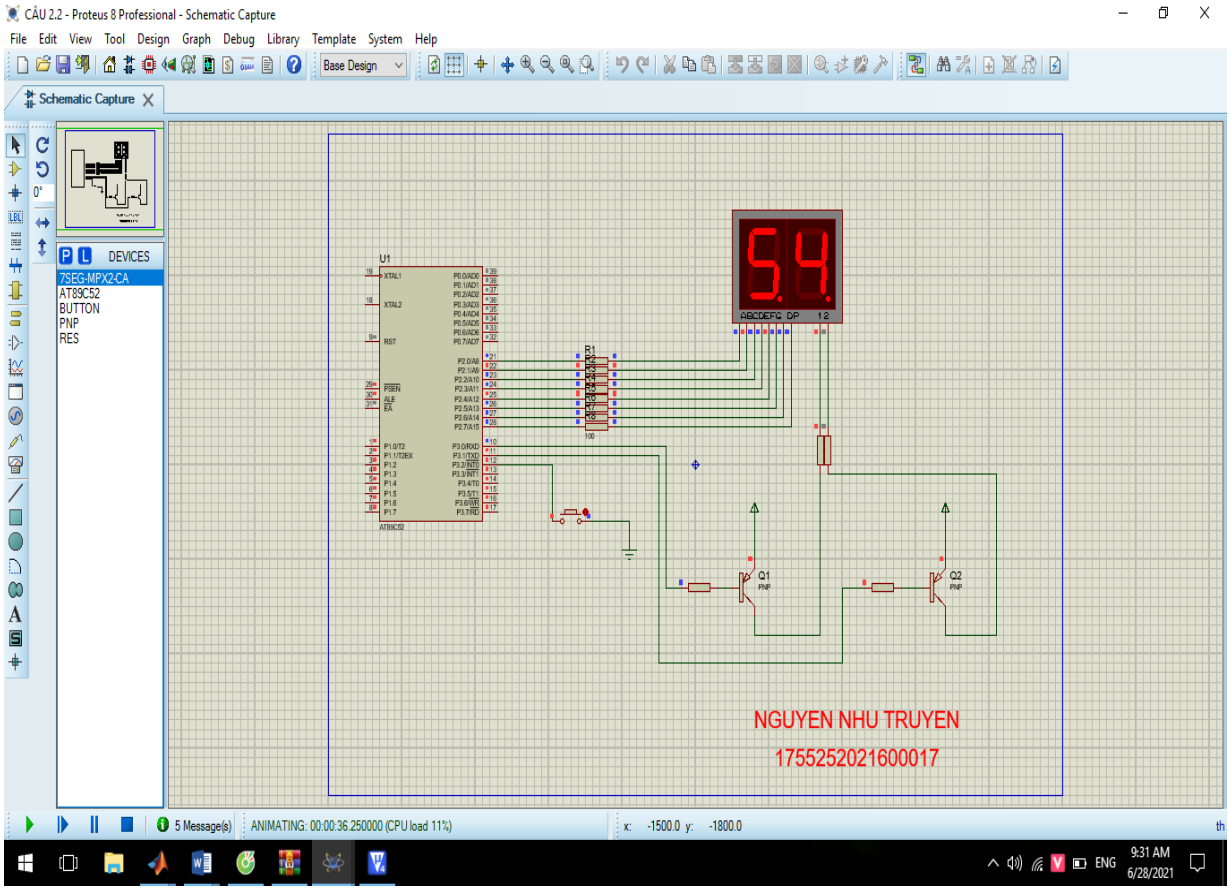
```
#include <REGX52.H>

#define led1 P3_0

#define led2 P3_1

#define sang 0

#define tat 1

sbit up=P3^2;

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

char count;

unsigned char chuc,donvi;

void delay_ms(int time){

while(time--){

TMOD=0x01;
```

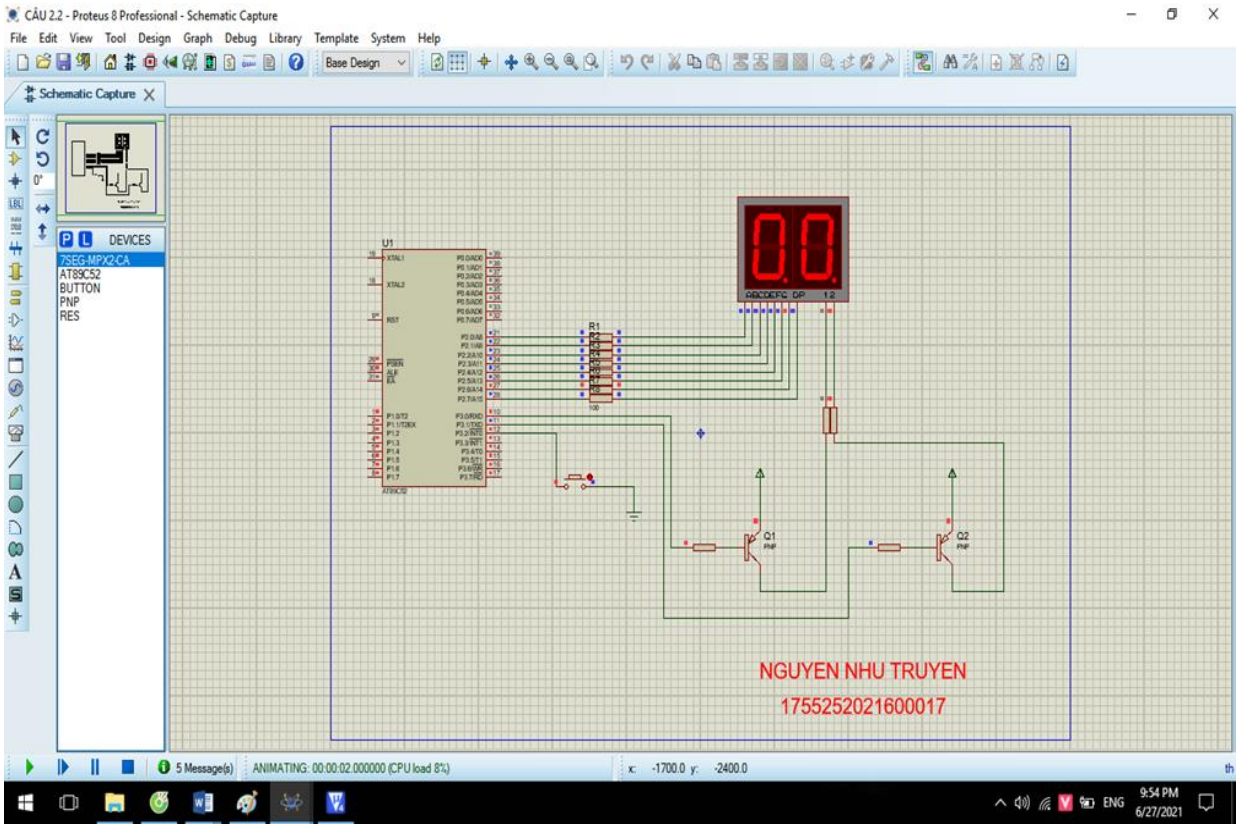
```
 TH0=0xfc;
TL0=0x18;
TR0=1;
while(!TF0);
TF0=0;
TR0=0;
            }
 }
void tang()interrupt 0{
count++;
if(count>30) count=0;
}
void main (){
EA=1;
EX0=1;
IT0=1;
EX1=1;
IT1=1;
while(1){
chuc=count/10
donvi=count%10;
led1 =sang;    P2=so[chuc];   delay_ms(10);   led1 =tat;
led2 =sang;    P2=so[donvi];  delay_ms(10);   led2 =tat;
            }
}
```

Mô phỏng trên phần mềm Proteus



SBD + 10 (SBD của em là 44)





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 4 bít, hiển thị họ tên mã số sinh viên lên LCD.

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

Giải:

Câu 3.1

Viết code trên KeilC

```
#include <REGX52.H>

#define LCD_RS P1_0

#define LCD_RW P1_1

#define LCD_EN P1_2

#define LCD_D4 P0_4

#define LCD_D5 P0_5
```

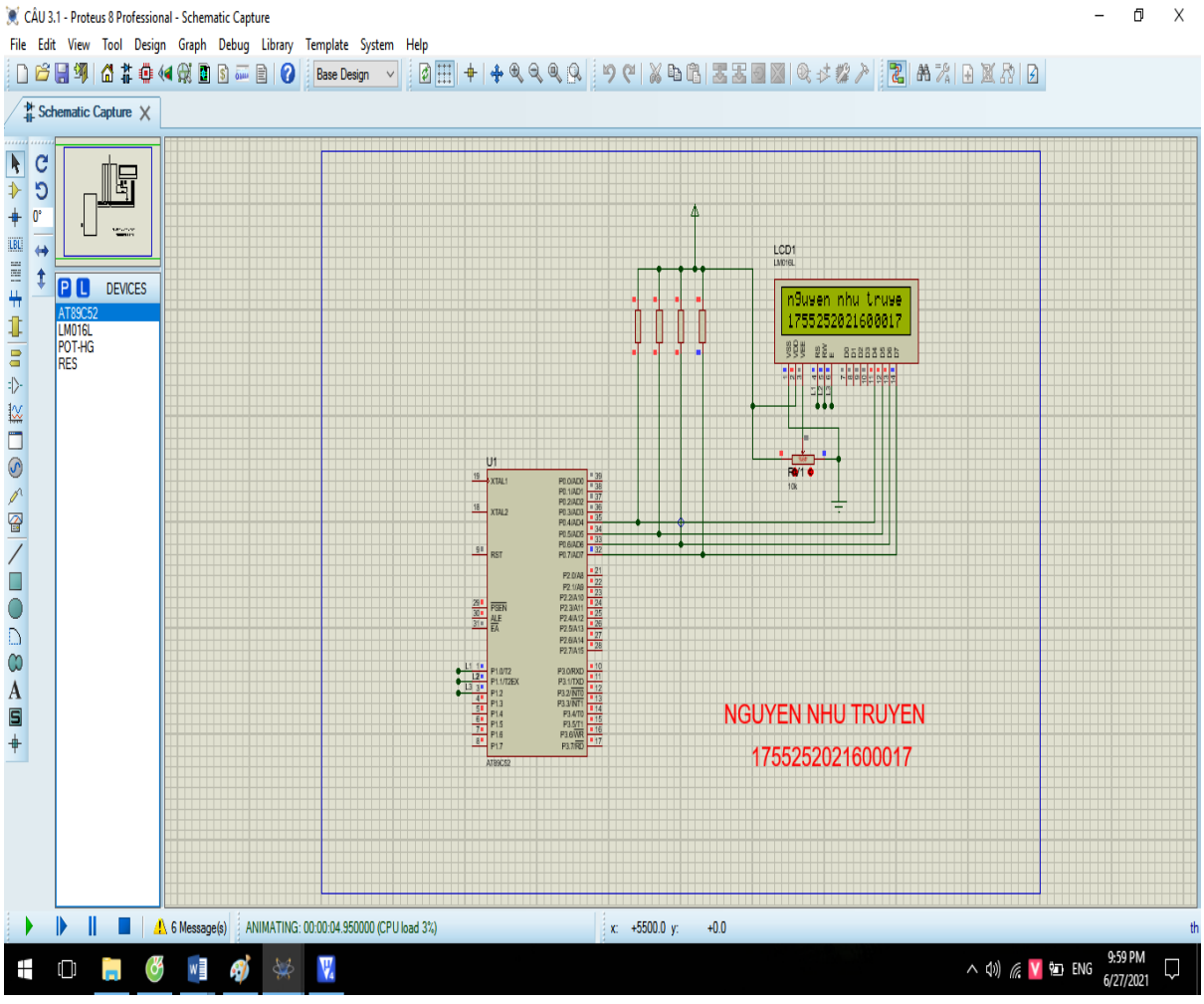
```
#define LCD_D6 P0_6
#define LCD_D7 P0_7
void delay_us(unsigned int t){
      unsigned int i;
           for(i=0;i<t;i++);
}
void delay_ms(unsigned int t){
      unsigned int i,j;
           for(i=0;i<t;i++)
           for(j=0;j<125;j++);
}
void delay(long time)
{
   while (time--);
}
void LCD_Enable(void){
       LCD_EN =1;
                  delay_us(3);
                  LCD_EN =0;
                  delay_us(50);
}
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;
}
void LCD_SendCommand(unsigned char command){
    LCD_Send4Bit(command >>4);
        LCD_Enable();
        LCD_Send4Bit(command);
        LCD_Enable();
}
void LCD_Clear(){
    LCD_SendCommand(0x01);
        delay_us(10);
}
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);
        LCD_SendCommand(0x0c);
        LCD_SendCommand(0x06);
        LCD_SendCommand(0x01);
}
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){
    LCD_RS=1;
        LCD_SendCommand(Data);
        LCD_RS=0;
}
void LCD_Puts(char*s){
    while (*s){
            LCD_PutChar(*s);
                s++;
        }
}
void main(){
```

```
    LCD_Init();
        LCD_Puts(" hello");
        delay_ms(1000);
        LCD_Clear();
        LCD_Gotoxy(0,0);
        LCD_Puts("nguyennhutruyen");
        delay_ms(1000);
        LCD_Gotoxy(0,1);
        LCD_Puts("1755252021600017");
        while(1);
}
```

Mô phỏng trên phần mềm Proteus



Câu 3.2

Viết code trên KeilC

```
#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)
```

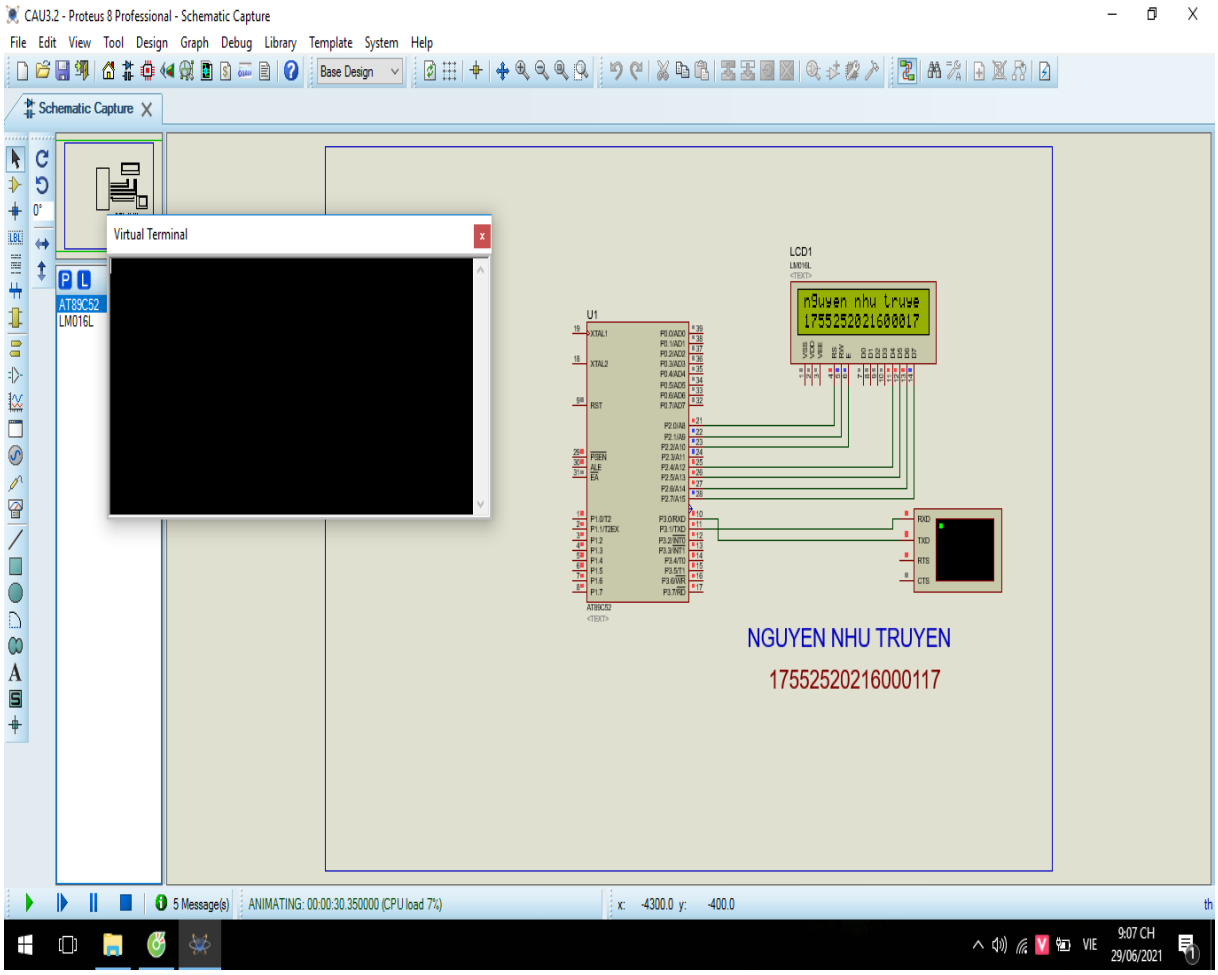
```
{
unsigned char x;
x = a & 0xF0;
cmd1(x); x = (a << 4) & 0xF0;
cmd1(x);
}
void dat(unsigned char a) {
unsigned char x;
x = a & 0xF0;
dat1(x); x = (a << 4) & 0xF0;
dat1(x); }
void main(void) {
unsigned char mybyte;
cmd(0x28);
cmd(0x01);
cmd(0x0C);
cmd(0x80);
cmd(0x06);
TMOD = 0x20; TH1 = 0xFD; SCON = 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 trên phần mềm 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 trình ngắt qua USART, task BUTTON, task LED. Thực hiện gửi " signal" từ ngắt USART và BUTTON đến tast LED. Task LED thực hiện trạng đảo trạng thái của LED D1 khi nhận được tín hiệu task khác gửi tới.

Giải

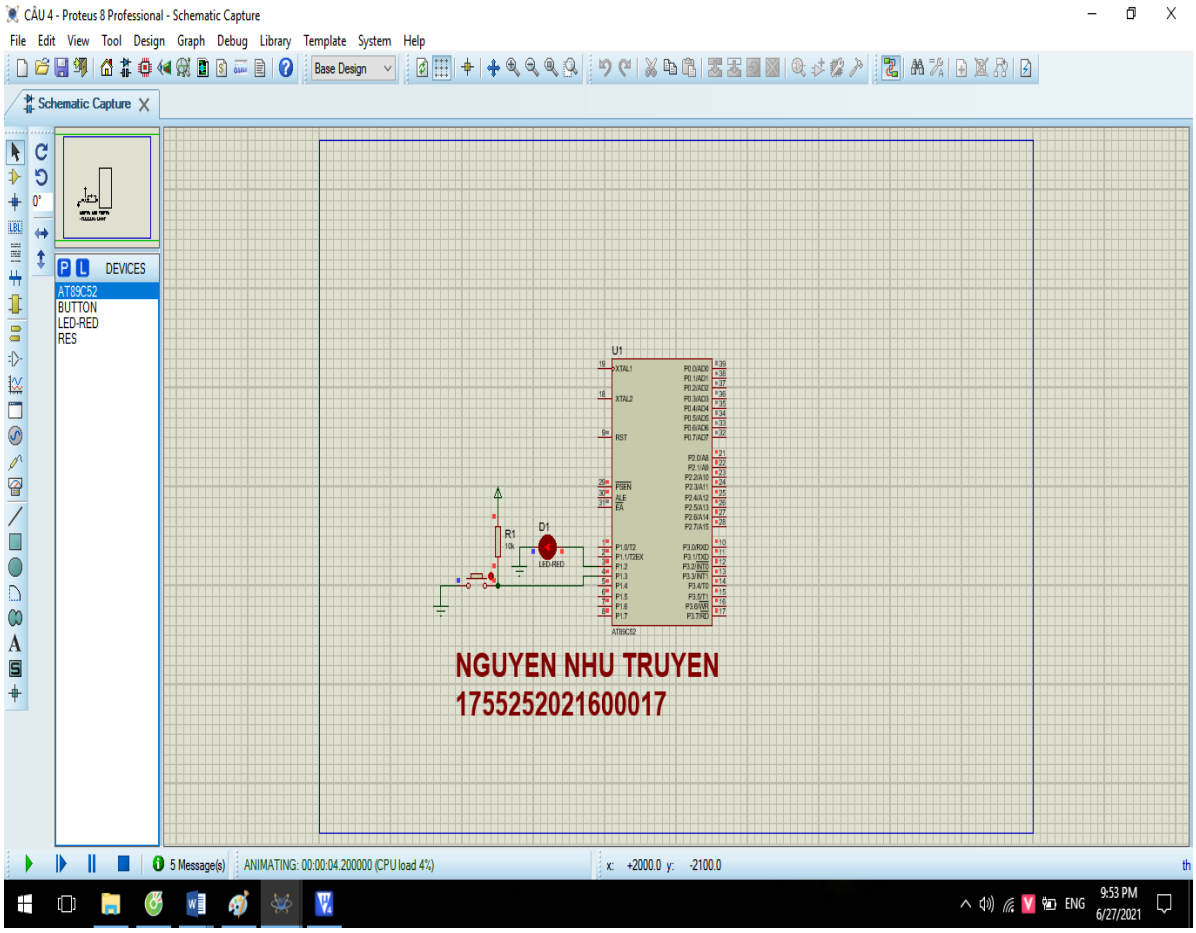
Viết code trên KeilC

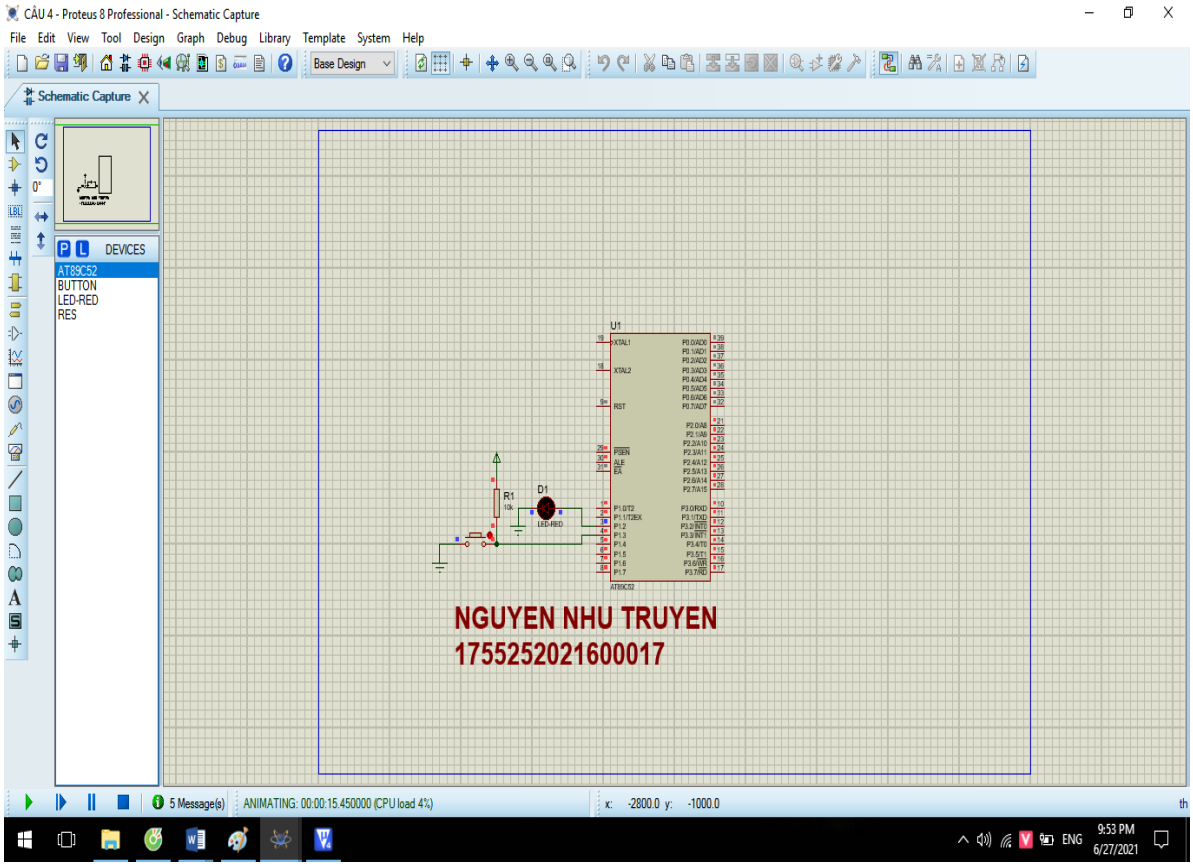
#include <REGX52.H>

```
#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 = P1^2;         //Dinh nghia chan LED_DO
sbit BUTTON = P1^3;         //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
 }
}
//=Ham Start up=
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;                //Dao 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 trên phần mềm Proteus


CÂU 4 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
AT89C52
BUTTON
LED-RED
RES
U1
R1
D1
LED-RED
NGUYEN NHU TRUYEN
1755252021600017
5 Message(s)
ANIMATING: 00:00:04.200000 (CPU load 4%)
x: +2000.0 y: -2100.0
ENG
9:53 PM
6/27/2021


CÂU 4 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
AT89C52
BUTTON
LED-RED
RES
U1
R1
D1
LED-RED
NGUYEN NHU TRUYEN
1755252021600017
5 Message(s)
ANIMATING: 00:00:15.450000 (CPU load 4%)
x: -2800.0 y: -1000.0
ENG
9:53 PM
6/27/2021

Câu 5. Hãy trình bày:

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

Các kỹ sư xây dựng hệ thống nhúng luôn phải đối mặt với nhiều khó khăn trong quá trình thiết kế hệ thống. Từ việc xác định phân bổ giữa phần cứng và phần mềm cho đến việc tính toán để thiết kế đạt những mục tiêu về hiệu năng và chi phí. Việc xây dựng hệ thống nhúng ngày càng trở lên phức tạp vì yêu cầu ngày càng cao. Họ phải xây dựng những hệ thống ngày càng thông minh hơn, có nhiều chức năng hơn nhưng lại phải được gói gọn trong một không gian nhỏ hơn, ít tiêu thụ điện năng hơn, thời gian sản xuất nhanh hơn và chi phí cho hệ thống giảm. Có lẽ bởi vì liên quan đến sự kết hợp của ít nhất hai nguyên tắc phức tạp và sáng tạo: kỹ nghệ phần mềm và thiết kế logic. Vấn đề này thường liên quan đến xây dựng một cái mới mà mọi người chưa từng xây dựng bao giờ hay đơn giản là có quá nhiều sự lựa chọn như: sử dụng bộ vi xử lý nào cho phù hợp, cách sắp xếp các bus, triển khai bằng ngôn ngữ lập trình nào, có sử dụng hệ điều hành hay không, môi trường phát triển …làm cho người phát triển nhiều lúc không biết bắt đầu từ đâu. Không giống với việc thiết kế các ứng dụng phần mềm trên máy tính, việc thiết kế một hệ thống nhúng phải thực hiện thiết kế cả phần cứng và phần mềm một cách song song. Mặc dù không phải lúc nào cũng vậy nhưng thực tế cho thấy đây là cách thức tiếp cận việc thiết kế hệ thống nhúng một cách hiệu quả và ảnh hưởng sâu sắc đến việc xây dựng hệ thống. Quy trình thiết kế hệ thống nhúng bao gồm các bước sau:

- Xác định yêu cầu sản phẩm
- Phân bổ phần cứng phần mềm
- Thiết kế phần cứng/phần mềm nhúng
- Tích hợp phần cứng/phần mềm
- Kiểm tra
- Bảo trì và nâng cấp

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.

5.2. 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: - 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 lý dữ liệu có độ trễ thấp, những lợi ích mà nó mà lại bao gồm khả năng đa nhiệm, ưu tiên các nhiệm vụ và quản lý việc chia sẻ tài nguyên giữa các tác vụ phức tạp. Hệ điều hành này được sử dụng phổ biến rộng rãi trong ngành hàng không, nhiều ngành công nghiệp và các thiết bị chăm sóc sức khỏe IoT với một số dịch vụ mà nó mang lại:

- Xử lý ngắt
- Dịch vụ quản lý thời gian
- Dịch vụ quản lý thiết bị
- Dịch vụ quản lý bộ nhớ
- Dịch vụ quản lý các kết nối vào – ra

Ví dụ: RIOT OS, VxWorks, Google Brillo, ARM Mbed OS, Hệ điều hành nhúng của Appe, Nucleus RTOS,...