

**TRƯỜNG ĐẠI HỌC VINH**

**VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

-----🙞🙜🕮🙞🙜-----



**TIỂU LUẬN**

**Môn:**

**THIẾT KẾ HỆ THỐNG NHÚNG**

Sinh viên thực hiện: Thịnh Kiên Cường

Lớp 58K KTĐK&TĐH

MSSV 1755252021600040

Giảng viên hướng dẫn: ThS. Mai Thế Anh

ThS. Lê Văn Chương

***Nghệ An, 2021***

NỘI DUNG CÁC CÂU HỎI

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;

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

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;

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).

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.

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

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, tắt BUTTON tắt 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.

Câu 5. Hãy trình bày:

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

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.

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

**a) Hiển thị số 00 lên 2 LED 7 thanh nối vào cổng P2 theo phương pháp quét LED;**

**+ Code viết bằng Mikro C**

#include <REG8253.H>

sbit L7S1 = P3^0;

sbit L7S2 = P3^1;

#define LED7SEG P2;

int main()

{

P2= 0xC0;

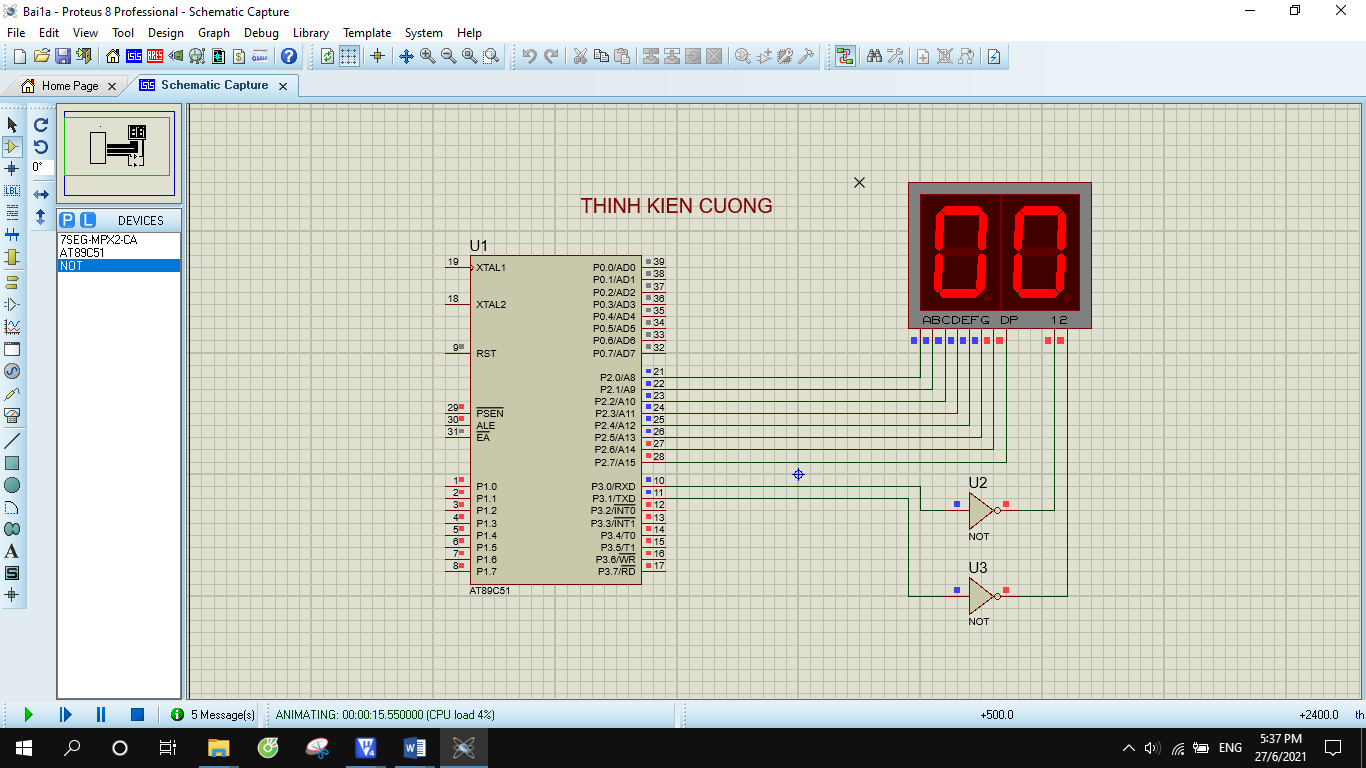
L7S1=0;

L7S2=0;

while(1) ;

}

**+ Mô phỏng bằng proteus**



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

\* SBD: 4

→ Số đếm = 4 + 20 = 24

**+ Code viết bằng Mikro C**

#include <REGX51.H>

sbit L7S1 = P3^0;

sbit L7S2 = P3^1;

char LED7SEG[11] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

void Fn\_Delay (unsigned int \_vrui\_Time)

{

while(\_vrui\_Time--);

}

void tangsbd()

{

char i;

int j;

for(i = 0; i < 25; i++)

{

for (j = 0; j < 30; j++)

{

P2 = LED7SEG[i/10];

L7S1 = 0;

Fn\_Delay(500);

L7S1 = 1;

P2 = LED7SEG[i%10];

L7S2 = 0;

Fn\_Delay(500);

L7S2 = 1;

}

}

}

int main ()

{

L7S1 = 0;

L7S2 = 0;

while(1)

{

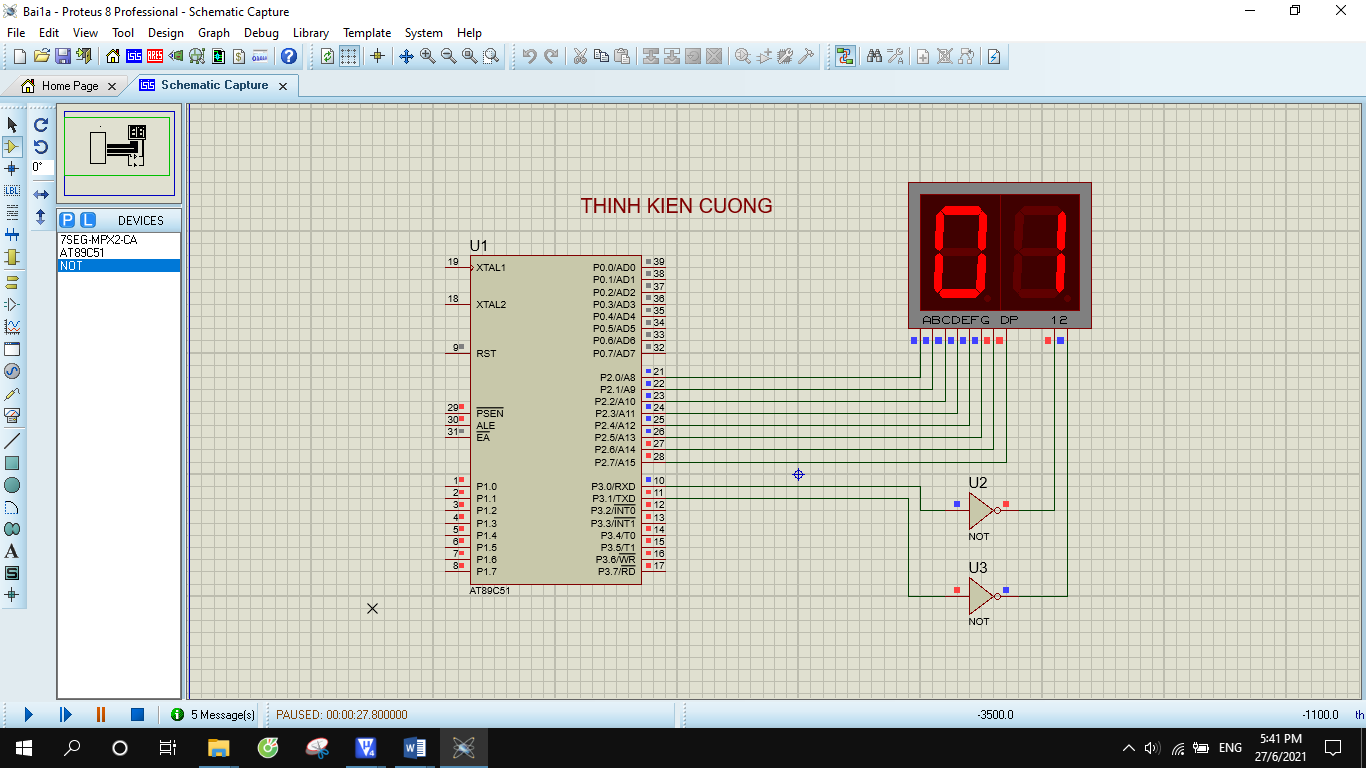
tangsbd();

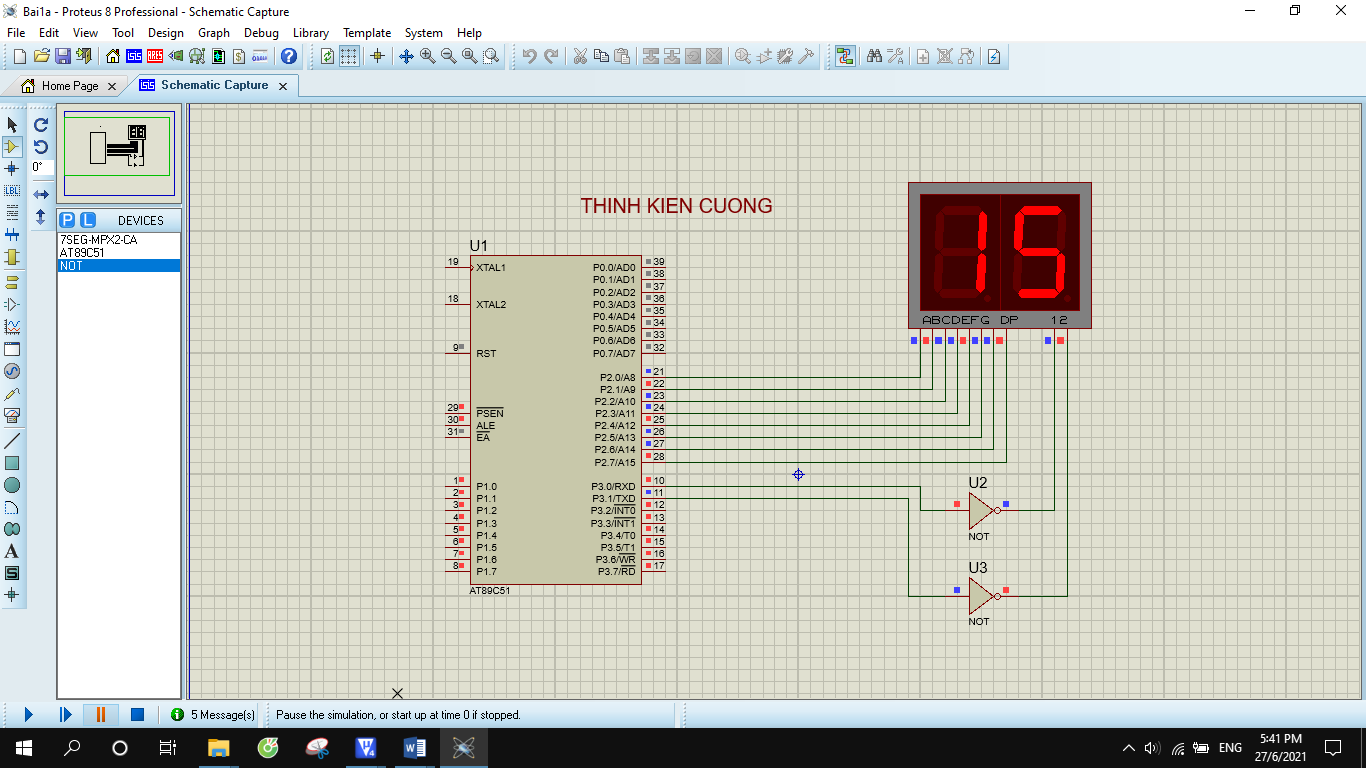
}

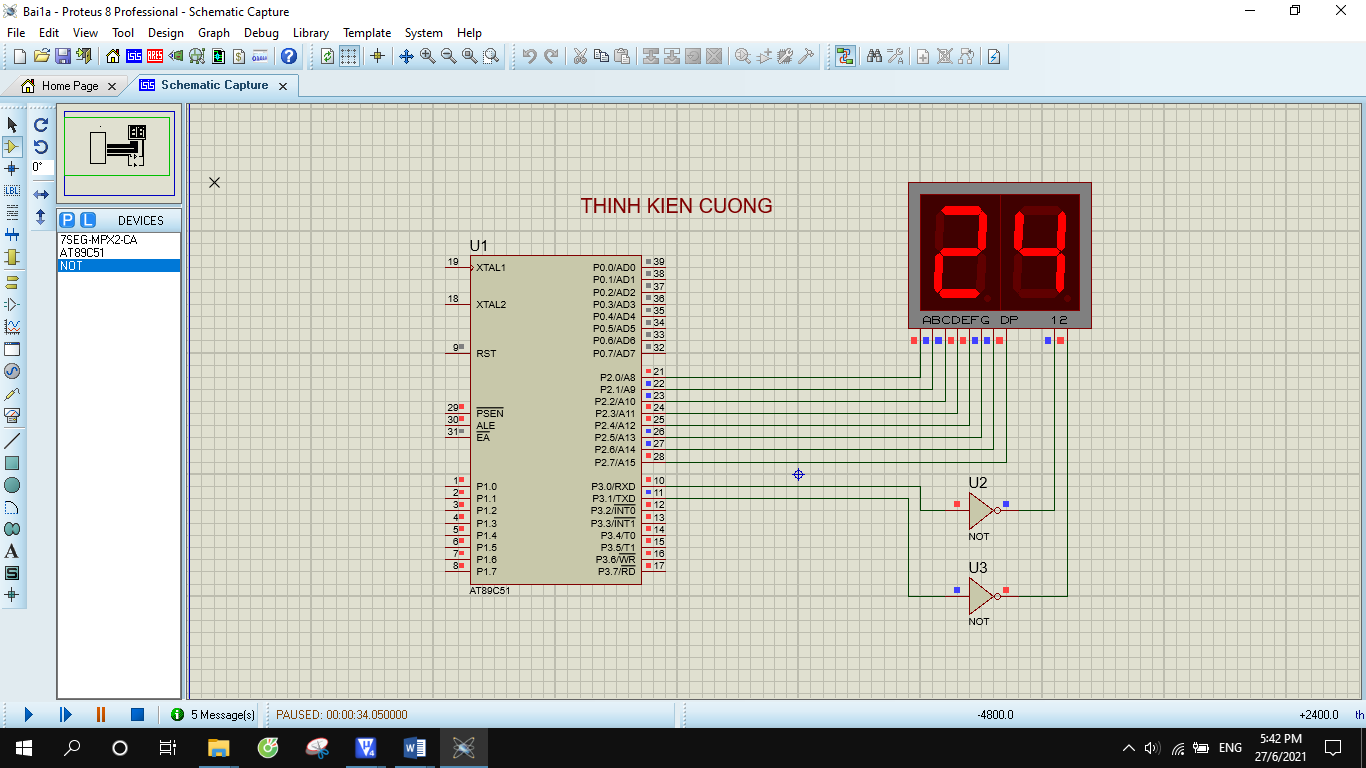
return 0;

}

**+ Mô phỏng bằng Proteus**







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

**a) Cấu hình ngắt ngoài INT0 ở chế độ ngắt sườn xuống;**

**+ Code viết bằng Mikro C**

#include <REGX51.H>

void delay\_ms(int time)

{

while(time--)

{

TMOD = 0x01;

TH0 = 0xfc;

TL0 = 0x18;

TR0 = 1;

while(!TF0);

TF0 = 0;

TR0 = 0;

}

}

void main()

{

EX0 = 1;

IT0 = 1;

EA = 1;

while(1)

{

P2 = 0;

delay\_ms(1000);

P2 = 0xff;

delay\_ms(1000);

}

}

void nutan(void) interrupt 0

{

long a=50000;

P1\_3=0;

while(a--){};

P1\_3=1;

}

**b) 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).**

\*SBD: 4

→ Số lần bấm = 4 + 10 = 14 lần

**+ Code viết bằng Mikro C**

#include <REGX51.H>

#define DOUT P2 //Chan xuat data led 7

#define D1 P3\_0 //Quet LED so 1

#define D2 P3\_1 //Quet LED so 2

#define D3 P3\_4 //Quet LED so 3

#define D4 P3\_5 //Quet LED so 4

#define UP P3\_2 //Nut nhan UP

unsigned char Mang[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90}; //Mang LED 7

char a,b;

unsigned char dem;

void delay(unsigned int t) //Ham tao tre

{

unsigned int i,j;

for(i=0;i<t;i++)for(j=0;j<100;j++);

}

void quet\_led() //Ham quet 4 LED 7

{

D1=0; //Led so 1 sang

DOUT=Mang[a/10]; //Xuat data gtri hang chuc cua bien a

delay(1) ;//Tao tre

DOUT=0xff; //Ngat data (tat bong mo voi led don xep thanh led 7)

D1=1;D2=0; //Led so 1 tat, led 2 sang

DOUT=Mang[a%10]; //Xuat data gtri hang don vi cua bien a

delay(1);

DOUT=0xff;

D2=1;D3=0; //Led so 2 tat, led 3 sang

DOUT=Mang[b/10]; //Xuat data gtri hang chuc cua bien b

delay(1);

DOUT=0xff;

D3=1;D4=0; //Led so 3 tat, led 4 sang

DOUT=Mang[b%10]; //Xuat data gtri hang don vi cua bien b

delay(1);

DOUT=0xff;

D4=1; //Tat Led so 4

}

void nut\_nhan() //Ham scan nut nhan

{

if(!UP) //Nut UP dc nhan

{

a++; //Tang gia tri

if(a>14)a=0; //Set ve 0 khi a>15

while(!UP); //Cho nha phim

}

}

void main() //Ctr chinh

{

delay(500);

while(1) //Vong lap ctr

{

nut\_nhan(); //Scan nut nhan

quet\_led(); //Scan LED 7

dem++; //Dem tu tang

if(dem>1)

{

b++; //b tang

dem=0;

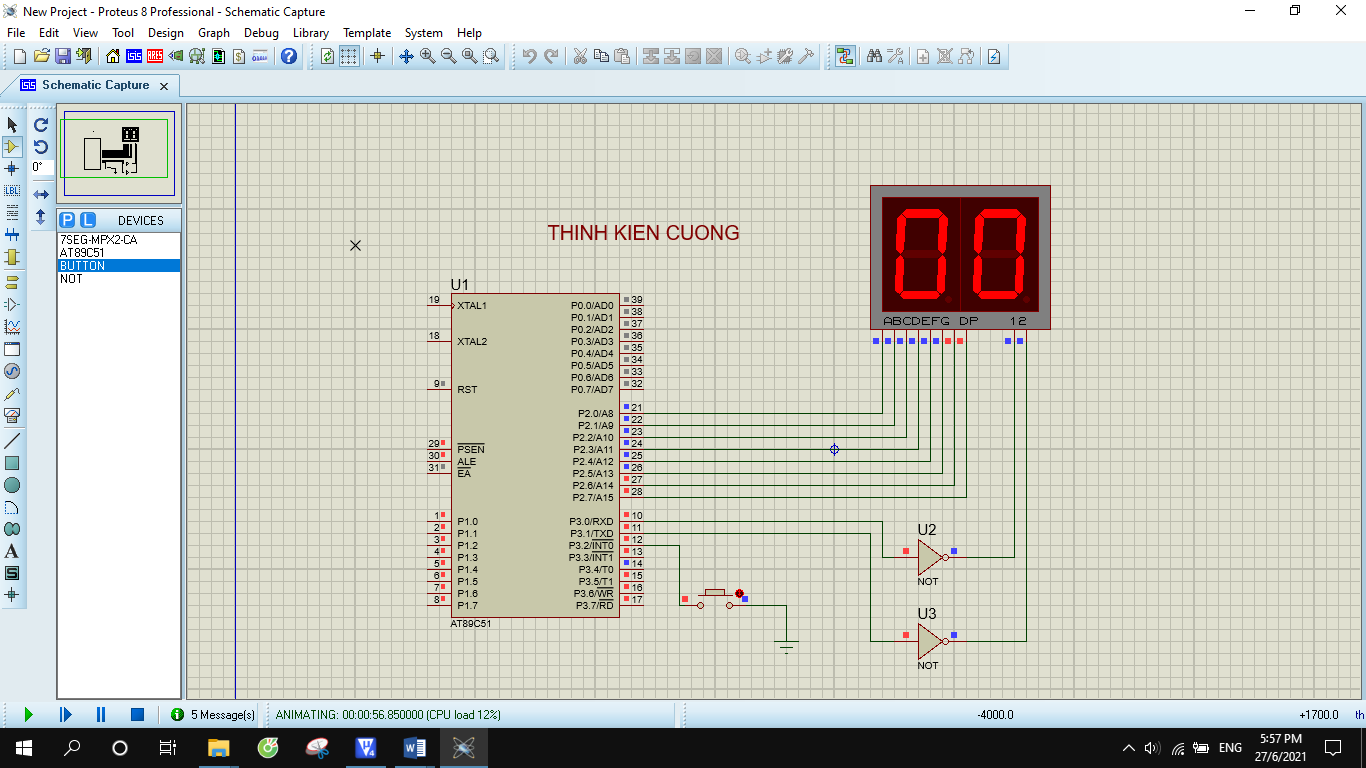
if(b>14)b=0; //Set lai gia tri

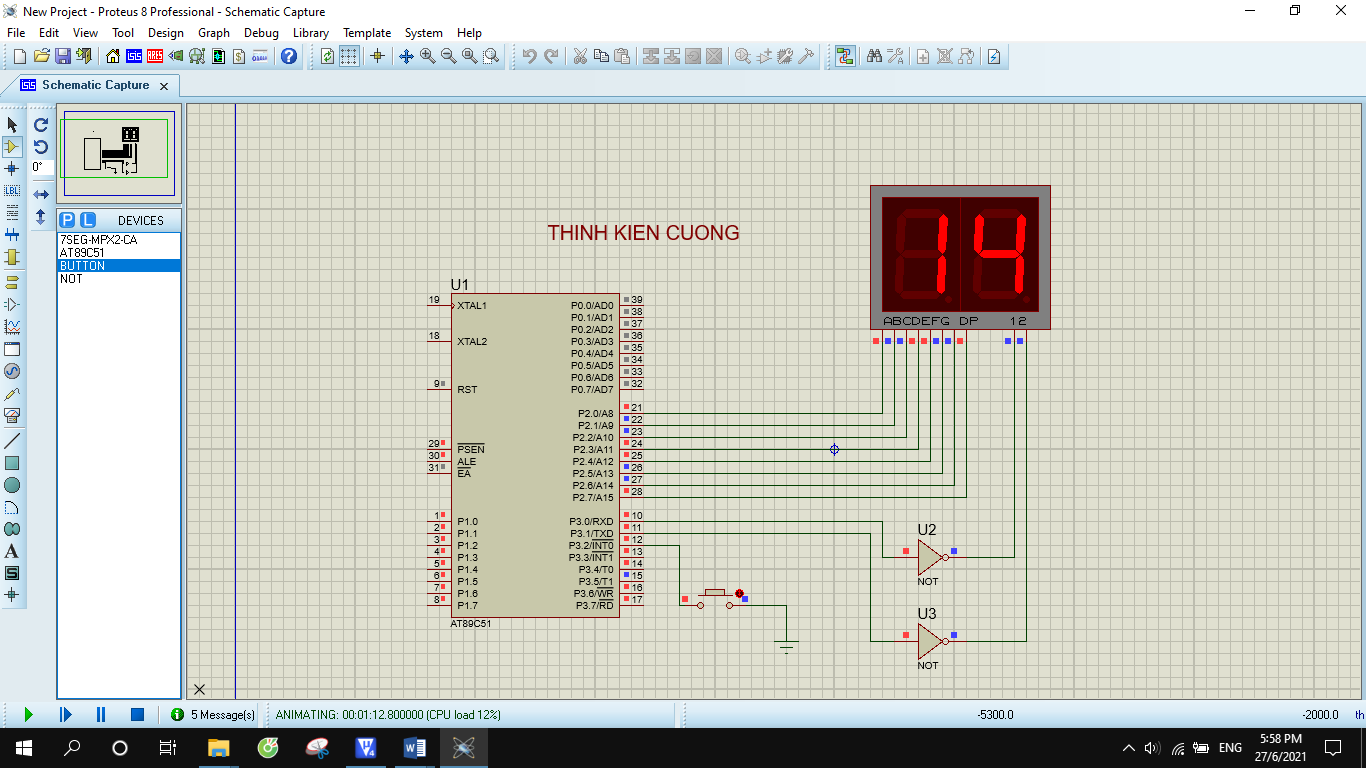
}

}

}

**+ Mô phỏng bằng Proteus**





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

**a) 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 viết bằng Mikro C**

#include <REGX51.H>

/\*\*\*\*\*\*\*\*Khai bao chan giao tiep\*\*\*\*\*\*\*\*\*\*\*\*/

#define LCD\_RS P0\_0

#define LCD\_RW P0\_1

#define LCD\_EN P0\_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++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ctr 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();

}

void LCD\_Clear() // Ham Xoa Man Hinh LCD

{

LCD\_SendCommand (0x01);

delay\_us (10);

}

// 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++;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ctr chinh\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

LCD\_Init();//Khoi tao LCD

delay\_ms(10);

LCD\_Puts("THINH KIEN CUONG");//Gui chuoi len LCD

delay\_ms(1000);

LCD\_Gotoxy(0,1) ;//Tro toi vi tri

LCD\_Puts("1755252021600040");

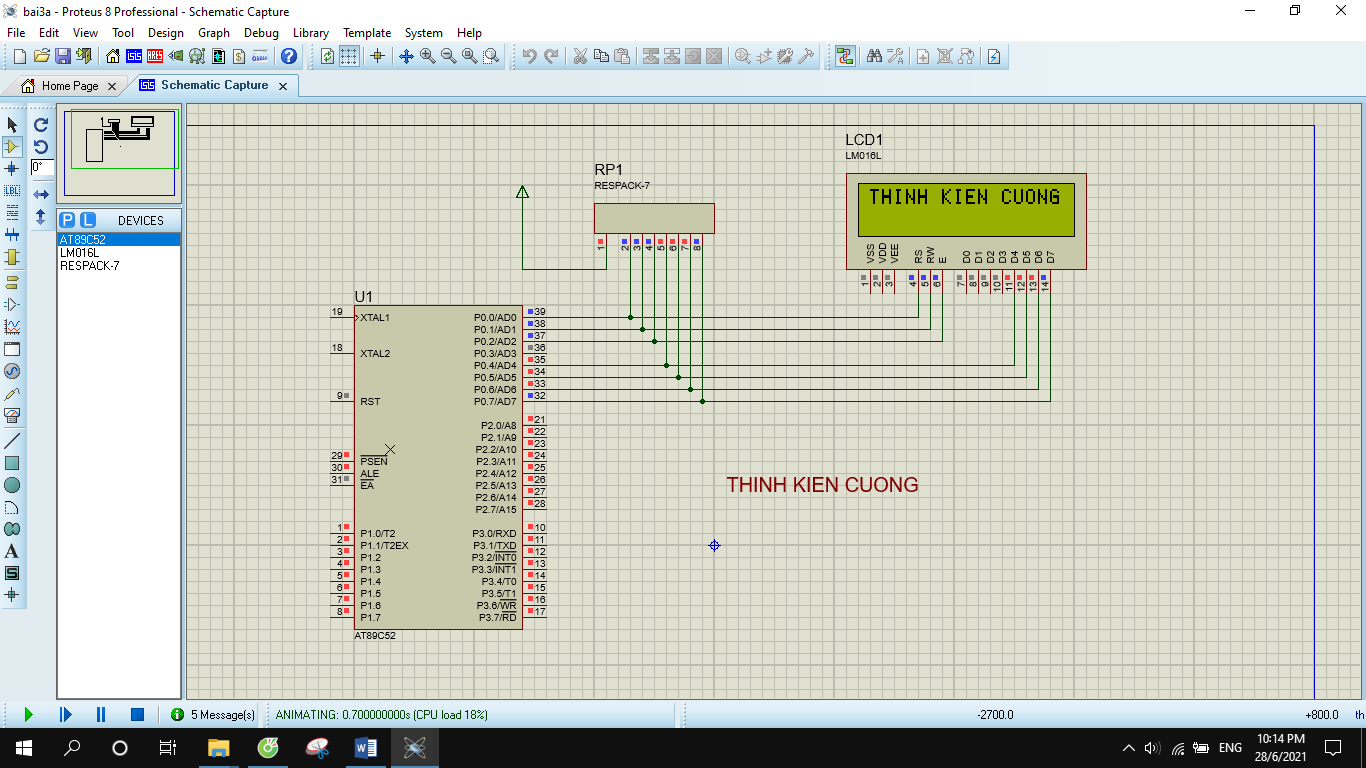
delay\_ms(1000000);

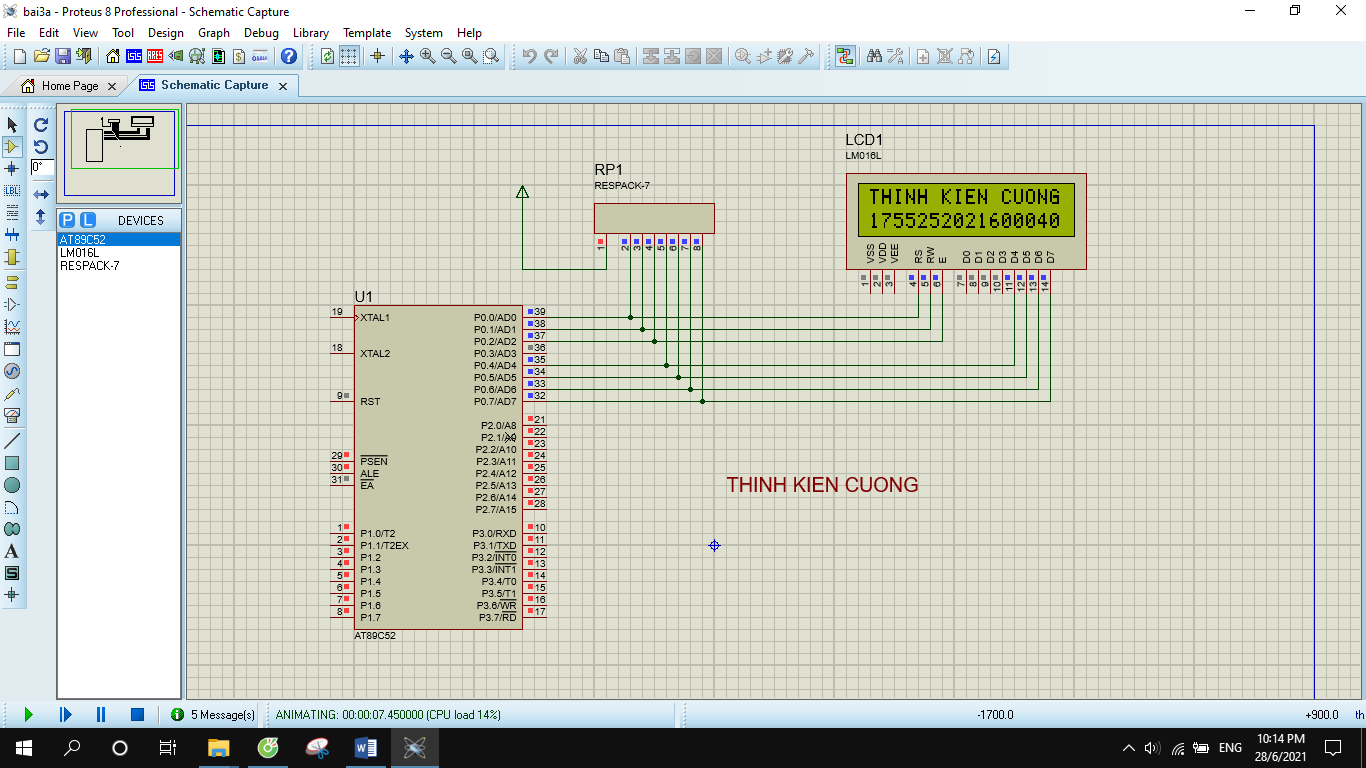
LCD\_Clear(); //Xoa man hinh

while(1);

}

**+ Mô phỏng proteus**





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

**+ Code viết bằng Mikro C**

#include <REGX51.H>

#include <string.h>

void delay (unsigned int t) //Ctr delay 50ms dung timer0

{

unsigned int i;

for(i=0;i<t;i++)

{

TH0=0x3c; //-50000us

TL0=0xb0;

TR0=1;

while(!TF0); //cho timer0 tran

TF0=TR0=0;

}

}

void send(unsigned char \*s) //Ham gui chuoi ki tu qua UART

{

unsigned char n,i;

n=strlen(s); //Dem xem co bao nhieu ky tu

for(i=0;i<n;i++){//Vong lap gui tung ky tu 1

SBUF=s[i];//Gui 1 byte

while(!TI); TI=0;//xoa co truyen

}

}

void ngat\_uart()interrupt 4 //Ngat nhan du lieu tu uart

{

if(RI)

{

P1=SBUF; //Xuat du lieu ra Post 1

}

RI = 0; //Xoa co nhan

}

void main (void)

{

TMOD=0x21; //Timer 1 che do 8bit nap lai tu dong, timer0 cho delay che do 16bit

SCON=0x50;//01010000 che do 1, cho phep nhan

TH1=TL1=0xFD;//Nap 253 tao baud 9600 ko nhan doi baud

TR1=1; //khoi dong timer1

ES=1; //Ngat UART

EA=1; //Cho phep ngat

delay (20);

send ("THINH KIEN CUONG \r");

delay (20);

send ("175525201600040 \r");

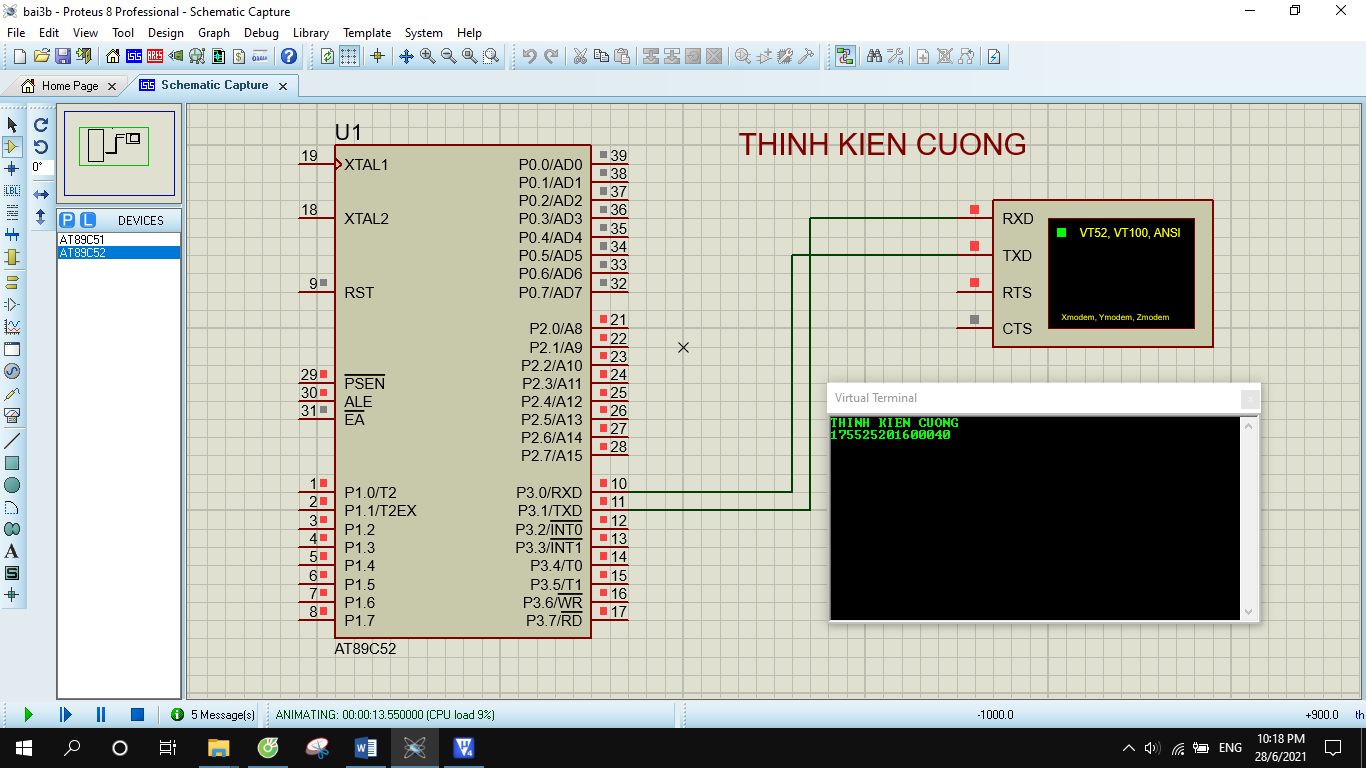
while(1)

{

}

}

**+ Mô phỏng bằng Proteus**



**Bài 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, tắt BUTTON tắt 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 viết bằng Mikro C**

**- Khởi tạo thư viện RTX51TNY.H**

#ifndef \_\_RTX51TNY\_H\_\_

#define \_\_RTX51TNY\_H\_\_

os\_send\_signal(Task\_id);

isr\_send\_signal(Task\_id);

os\_clear\_signal(Task\_id)

os\_wait2(K\_SIG, 50);

#endif

**- Code chương trình**

#include <REGX51.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)

}

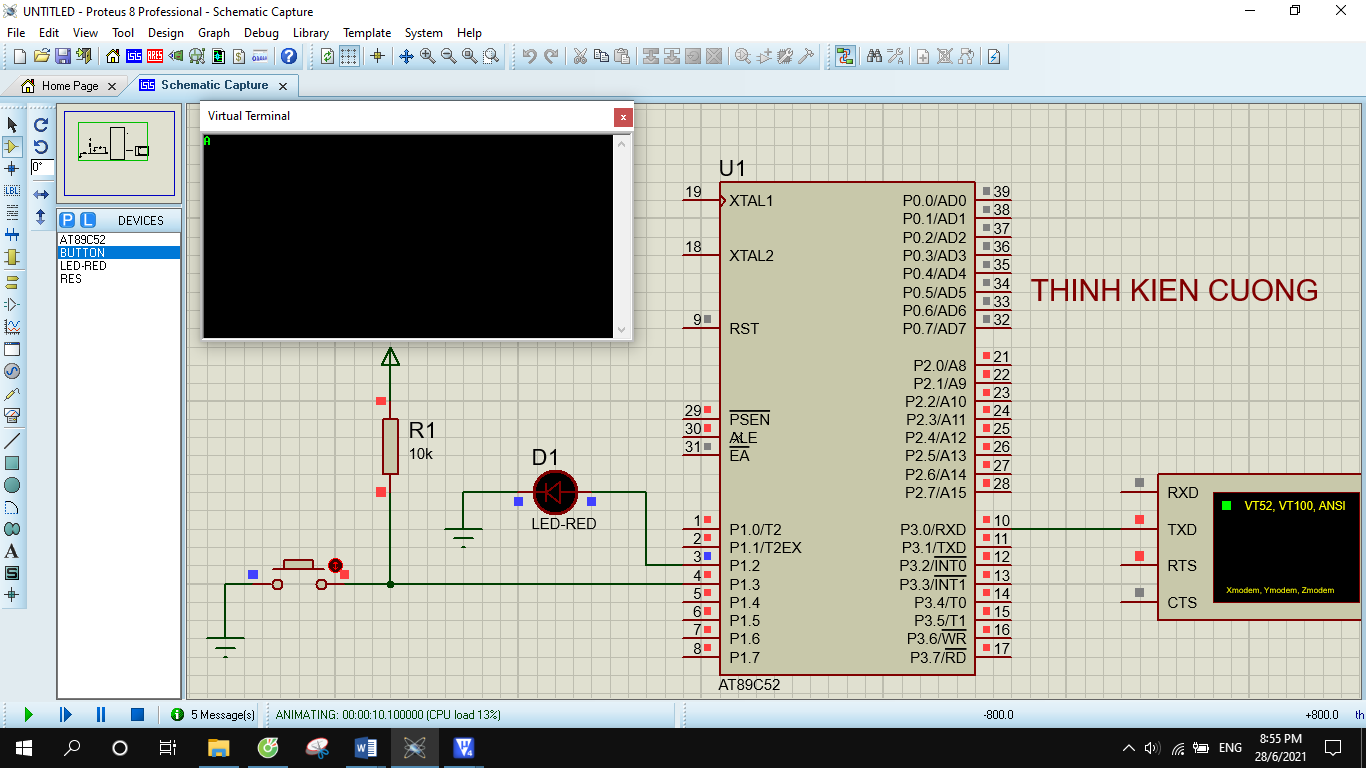
os\_wait2(K\_TMO, 10); //Cho 10 ticks = 100ms

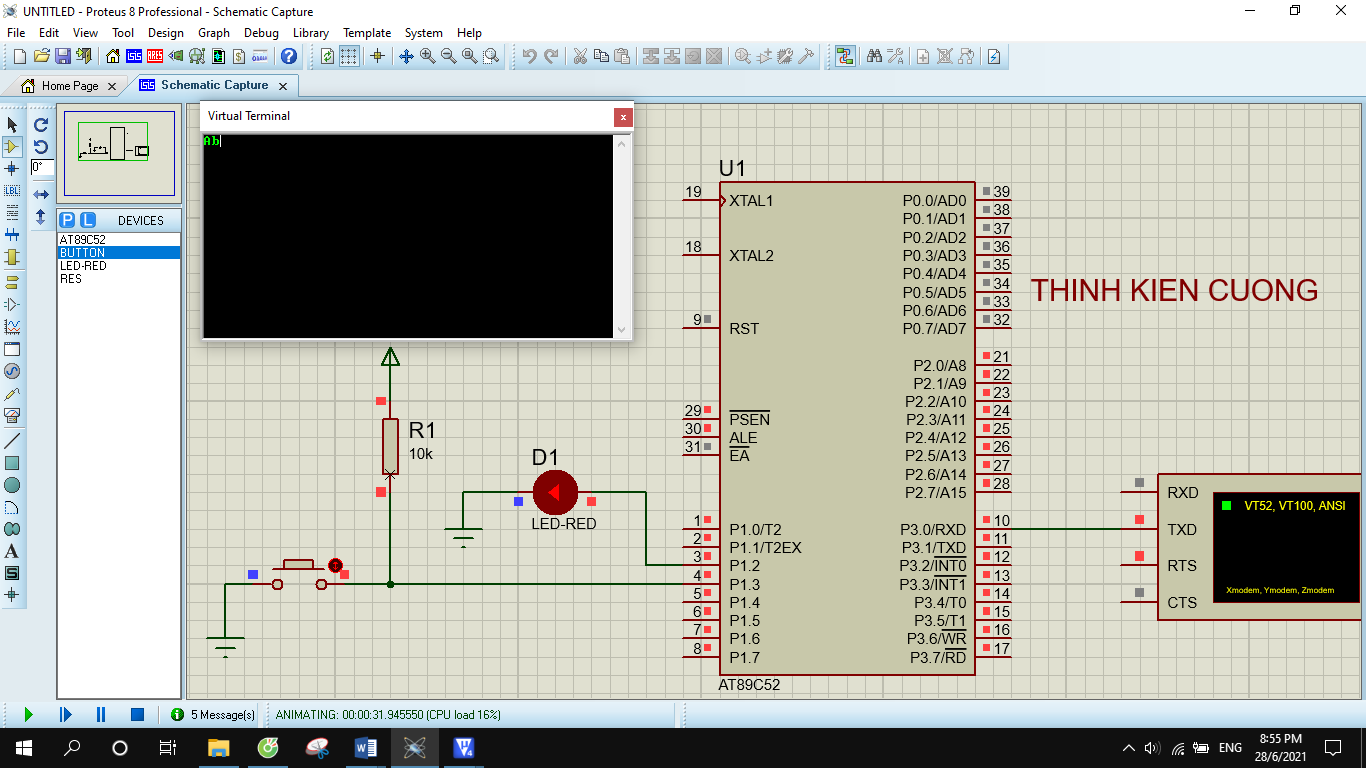
}

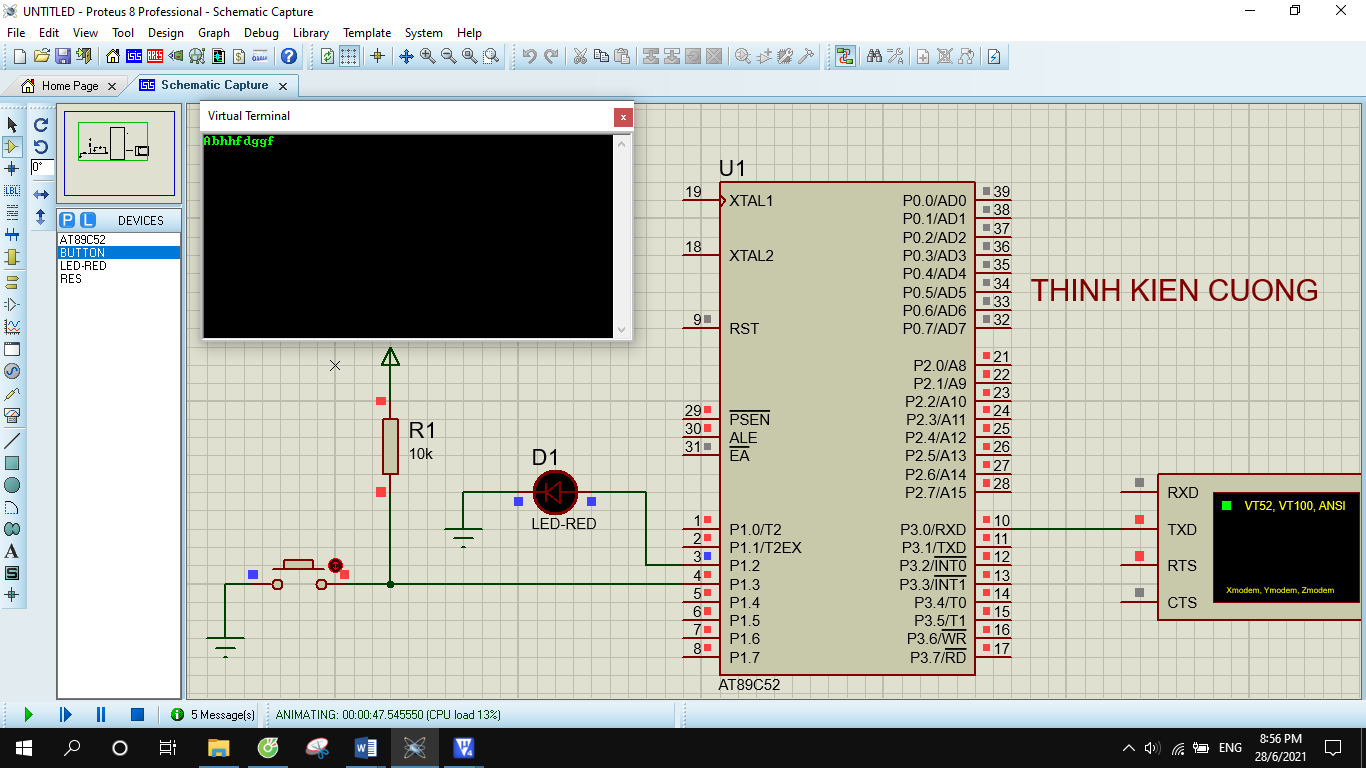
}

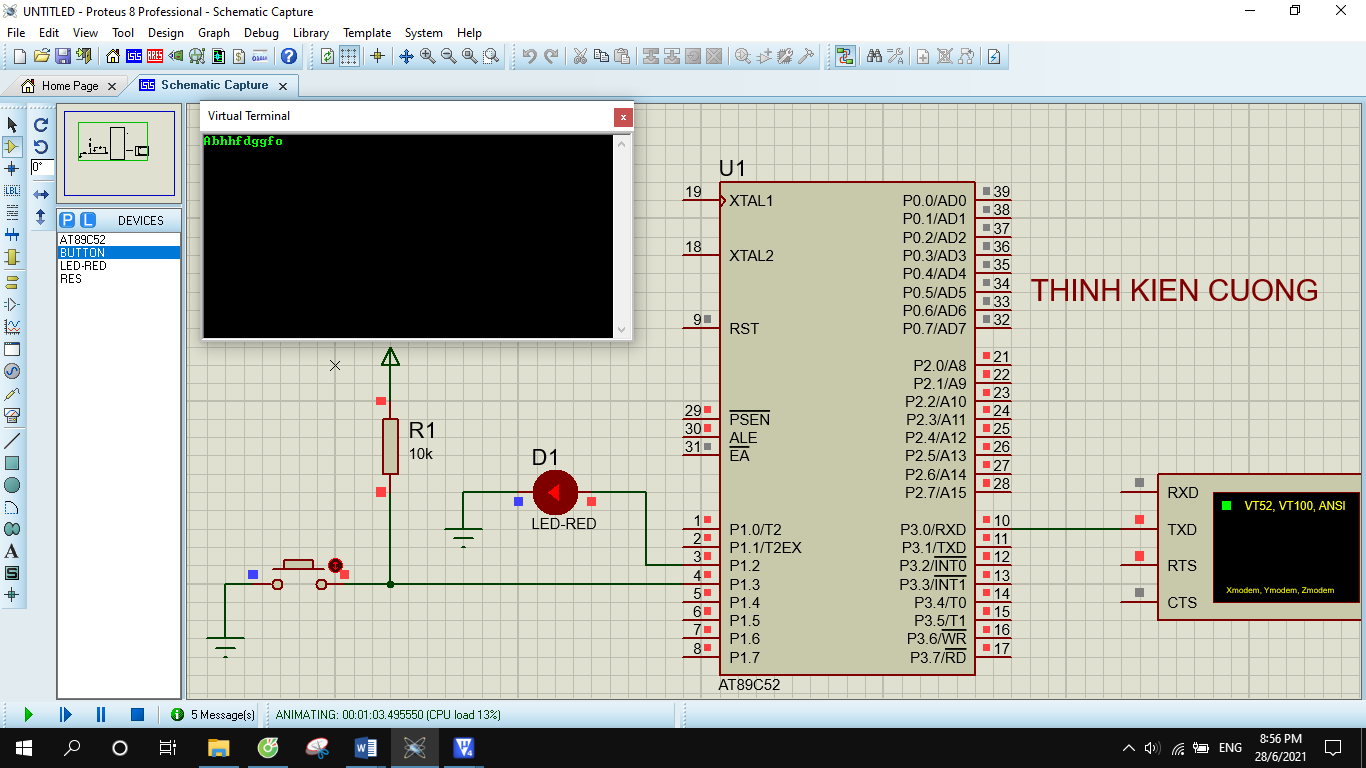
**+ Mô phỏng bằng Proteus**

Khi ta ấn phím bất kỳ hiển thị trên hộp thoại Virtual Terminal thì LED sẽ chuyển trạng thái đóng ngắt liên tục.









Bài 5. Hãy trình bày

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

 a) Thiết kế phần cứng hệ thống nhúng:

- Bộ xử lý chức năng đơn chuyên dụng

- Bộ xử lý chức năng đơn tiêu chuẩn – ngoại vi

- Bộ nhớ

- Giao diện

- Công nghệ IC

b) Lập trình cho hệ thống nhúng:

- Các phần mềm lập trình

- Kiểu Biến

- Từ khóa

- Hằng số

- Các Toán tử

- Các cấu trúc điều khiển

- Hàm

- Tiền xử lý và thư viện

c) Thiết kế hệ thống nhúng:

Việc thiết kế hệ thống nhúng khá phức tạp và đòi hỏi người kỹ sư thiết kế phải có nhiều kinh nghiệm. Hầu hết các mô hình được sử dụng trong việc thiết kế hệ thống nhúng đều dựa trên một hoặc sự kết hợp nhiều mô hình trong các các mô hình phát triển sau:

+ Mô hình big-bang: Không hề có một kế hoạch cụ thể trước và trong suốt quá trình phát triển hệ thống

Mô hình code-and-fix: Là một mô hình khá đơn giản, chỉ thích hợp cho các chương trình nhỏ (không đòi hỏi việc bảo trì), không thích hợp với các hệ thống lớn, bao gồm 2 bước:

- Viết code

- Fix các vấn đề phát sinh

+ Mô hình waterfall: Trong mô hình này, quá trình phát triển hệ thống được xây dựng theo từng bước, các kết quả của một bước sẽ được sử dụng cho bước kế tiếp.

+ Mô hình spiral: Quá trình phát triển hệ thống được chia thành nhiều giai đoạn. Dựa trên sự phản hồi từ các giai đoạn, kết hợp trở lại vào quá trình để lên kế hoạch cho việc thực hiện giai đoạn tiếp theo.

**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**

**2.1. Hệ điều hành thời gian thực (RTOS)**

Là hệ điều hành nhằm phục vụ ứng dụng thời gian thực xử lý dữ liệu khi nó được đưa vào, hầu như không có độ trễ bộ đệm. Dạng đầy đủ của RTOS là Hệ điều hành thời gian thực.

Trong RTOS, yêu cầu về thời gian xử lý được tính bằng phần gia số phần mười giây của thời gian. Đó là hệ thống ràng buộc thời gian có thể được định nghĩa là các ràng buộc thời gian cố định. Trong loại hệ thống này, quá trình xử lý phải được thực hiện bên trong các ràng buộc được chỉ định. Nếu không, hệ thống sẽ bị lỗi.

**2.2. Ưu điểm, nhược điểm hệ điều hành thời gian thực**

\* Ưu điểm

+ Ít thời gian ngừng hoạt động:

Trong khi giữ tất cả các thiết bị ở trạng thái hoạt động, RTOS đảm bảo rằng hệ thống tiêu thụ nhiều tài nguyên hơn. Do đó, một hệ thống sử dụng RTOS trải qua rất ít thời gian chết. Ngoài ra, các công ty lưu trữ cũng thể hiện kết quả tối đa khi sử dụng RTOS.

+ Quản lý công việc:

Hệ điều hành thời gian thực thường mất ít thời gian hơn để chuyển từ tác vụ này sang tác vụ khác. Thông thường, phải mất 3 micro giây hoặc ít hơn để chuyển nhiệm vụ. Kiểu quản lý tác vụ nhanh hơn này đảm bảo các quy trình quan trọng có thể đạt được đúng thời gian.

+ Hiệu quả:

Hệ điều hành thời gian thực tập trung vào một ứng dụng tại một thời điểm nhất định. Thông thường, ứng dụng này sẽ là ứng dụng đã chạy. Tất cả những người khác trong hàng đợi sẽ được giữ trong giai đoạn chờ đợi. Do đó, các nhiệm vụ quan trọng có thể được xử lý đúng thời hạn trong thời hạn nhất định để đạt được kết quả chính xác cần thiết.

+ Tính khả dụng:

Do RTOS thể hiện kết quả tối đa, nó là một hệ thống hoạt động 24/7. Do đó, nó phù hợp nhất cho các ứng dụng luôn hoạt động. Bên cạnh đó, một hệ thống RTOS có khả năng hỗ trợ các nền tảng MCU khác nhau.

+ Độ tin cậy

Hệ điều hành thời gian thực, đặc biệt là những hệ điều hành RTOS cứng hoàn toàn không có lỗi. Nó đảm bảo một cách tốt hơn để xử lý lỗi. Hơn nữa, hệ điều hành gặp phải một vấn đề được gọi là jitter, trong đó số lượng lỗi giữa các vòng lặp tiếp theo được đo. Nếu được lập trình chính xác, RTOS có thể được tối ưu hóa theo cách mà nó ít bị rung hơn.

\* Nhược điểm

+ Hệ thống RTOS có thể chạy các tác vụ tối thiểu cùng nhau và nó chỉ tập trung vào những ứng dụng có lỗi để có thể tránh chúng.

+ Là hệ thống tập trung vào một số tác vụ. Do đó, rất khó để các hệ thống này thực hiện đa tác vụ.

+ Các trình điều khiển cụ thể được yêu cầu cho RTOS để nó có thể cung cấp thời gian phản hồi nhanh để ngắt tín hiệu, giúp duy trì tốc độ của nó.

+ Rất nhiều tài nguyên được sử dụng bởi RTOS, điều này làm cho hệ thống này trở nên đắt đỏ.

+ Các tác vụ có mức độ ưu tiên thấp cần phải đợi một thời gian dài vì RTOS duy trì tính chính xác của chương trình đang được thực thi.

+ Việc chuyển đổi tối thiểu các tác vụ được thực hiện trong hệ điều hành Thời gian thực.

+ Nó sử dụng các thuật toán phức tạp khó hiểu.

+ RTOS sử dụng nhiều tài nguyên, đôi khi không phù hợp với hệ thống.

**2.3. Ứng dụng của hệ điều hành thời gian thực trong thiết kế các hệ thống nhúng**

+ Hệ thống đặt chỗ của các hãng hàng không.

+ Hệ thống kiểm soát không lưu.

+ Hệ thống cung cấp cập nhật ngay lập tức.

+ Được sử dụng trong bất kỳ hệ thống nào cung cấp thông tin cập nhật từng phút về giá cổ phiếu.

+ Hệ thống ứng dụng phòng thủ như RADAR.

+ Hệ thống đa phương tiện nối mạng

+ Hệ thống điều khiển lệnh

+ Điện thoại Internet

+ Hệ thống phanh chống bó cứng

+ Máy tạo nhịp tim