**TRƯỜNG ĐẠI HỌC VINH**

VIỆN KỸ THUẬT CÔNG NGHỆ

**=====\*\*\*====**



BÁO CÁO TIỂU LUẬN

***THIẾT KẾ HỆ THỐNG NHÚNG***

Giáo viên hướng dẫn: TS MAI THẾ ANH

Th.S LÊ VĂN CHƯƠNG

Họ & tên: Nguyễn Tiến Đạt

MSSV: 1755252021600009

Lớp: 58K – Kỹ thuật ĐK & TĐH

**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, 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.

**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.

**NỘI DUNG BÀI LÀM**

**Câu 1.**

* 1. **Chương trình điều khiển**

#include <REGX52.H>

char ma7thanh[] = {0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10, 0x89, 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 = ma7thanh[0];

delay(1000);

LED1 = tat;

LED2 = sang;

P0 = ma7thanh[0];

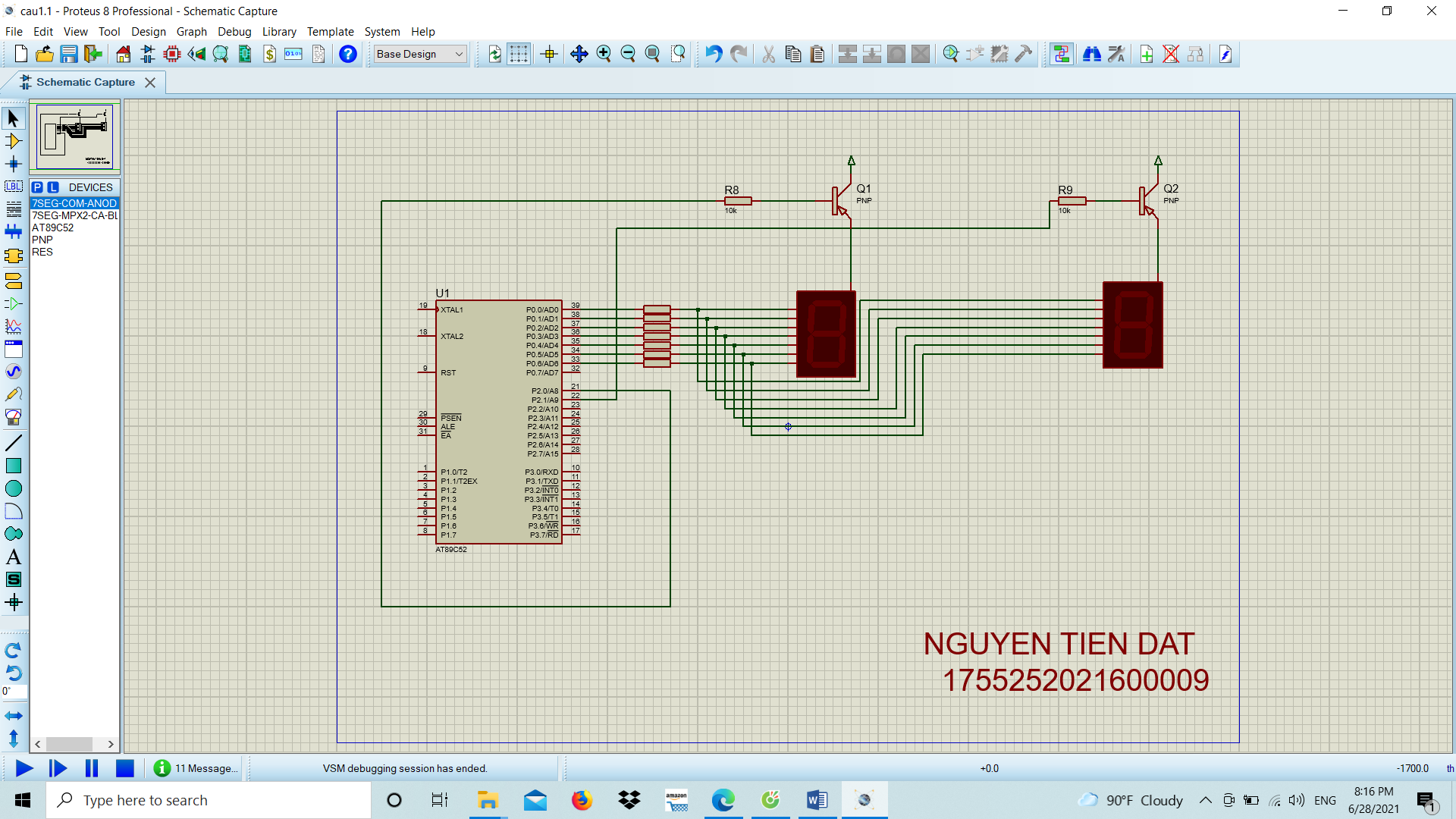
delay(1000);

LED2 = tat;

}

}

* 1. **Mô phỏng Proteus**



* 1. **Chương trình điều khiển**

#include <REGX52.H>

char ma7thanh[] = {0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10, 0x89, 0xC7, 0x40};

char i;

int dem;

unsigned char chuc, donvi;

#define LED1 P2\_0

#define LED2 P2\_1

#define sang 0

#define tat 1

char i;

void delay()

{

TMOD = 0x01;

TH0 = 0xD8;

TL0 = 0xF0;

TR0 = 1;

while(!TF0);

TF0 = 0;

TR0 = 0;

}

void main()

{

LED1 = LED2 = tat;

while(1)

{

for (dem = 0; dem <= 45; dem++)

{

chuc = dem/10;

donvi = dem%10;

for (i = 0; i <= 20; i++)

{

LED1 = sang;

P0 = ma7thanh[chuc];

delay();

LED1 = tat;

LED2 = sang;

P0 = ma7thanh[donvi];

delay();

LED2 = tat;

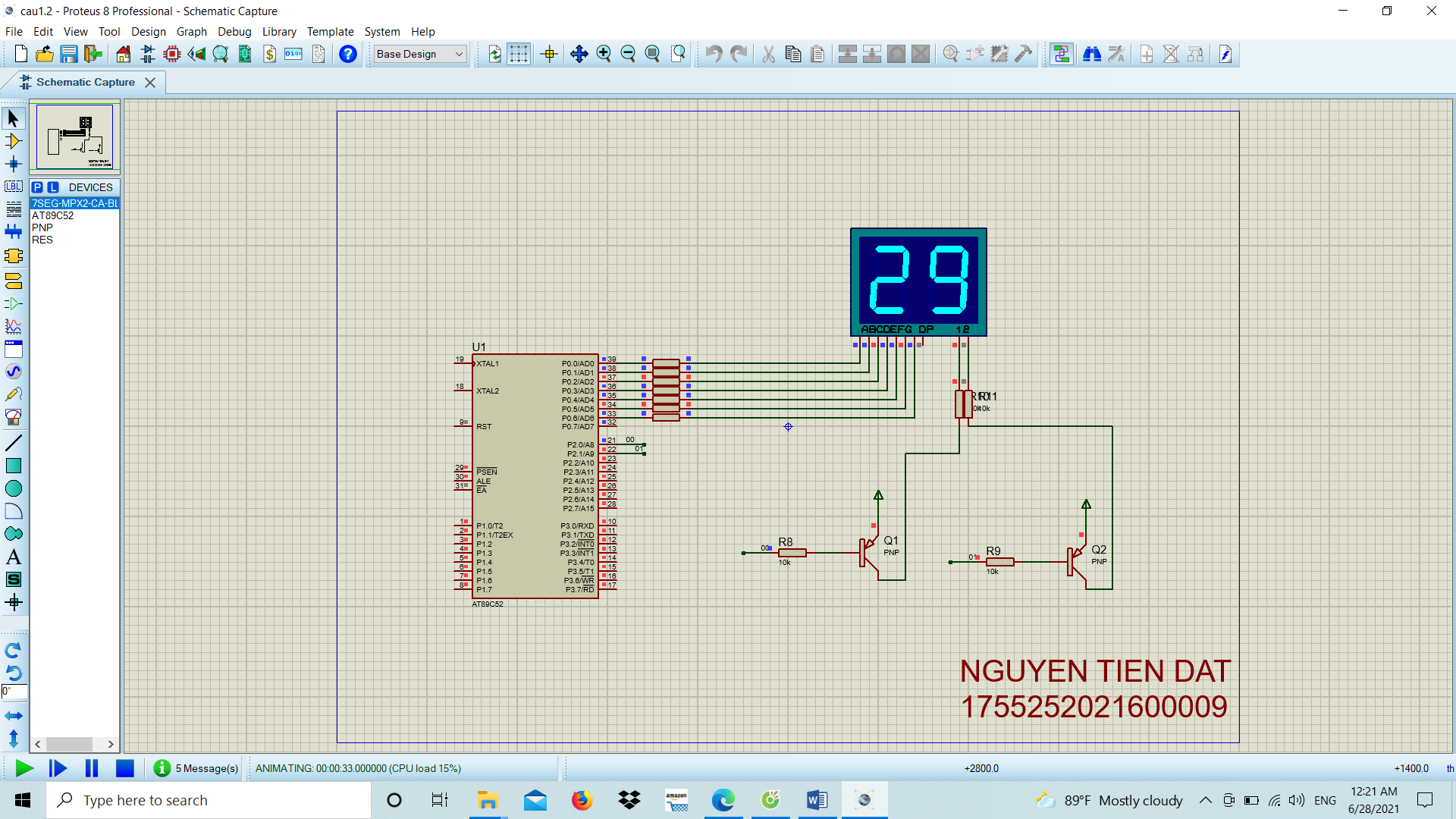
}

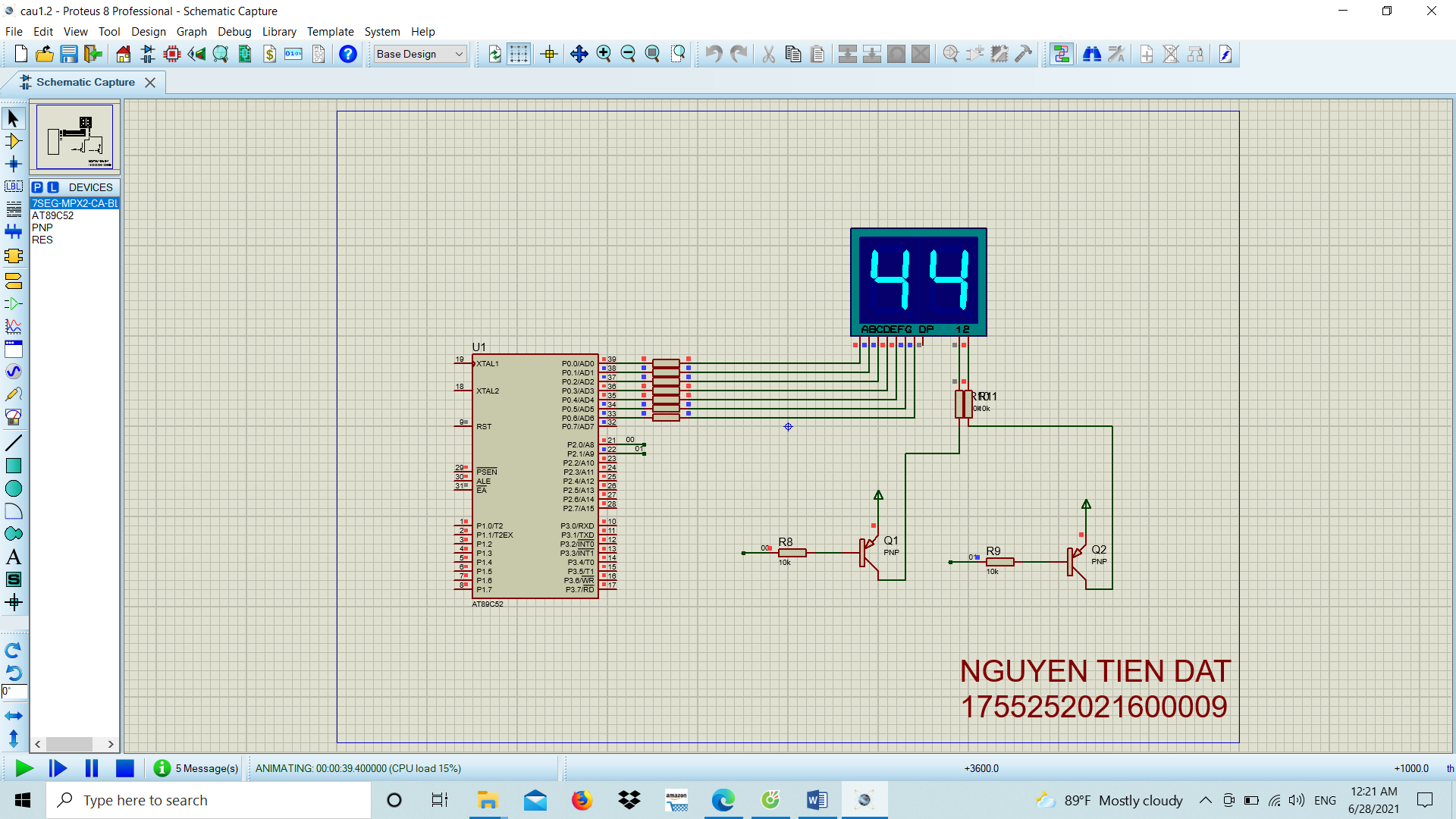
}

}

}

* 1. ***Mô phỏng Proteus***





**Câu 2**.

* 1. **Chương trình điều khiển**

#include <REGX52.H>

void delay\_ms(int ms)

{

while(ms--)

{

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 ngat() interrupt 0

{

long a = 50000;

P1\_3 = 0;

while(a--)

{

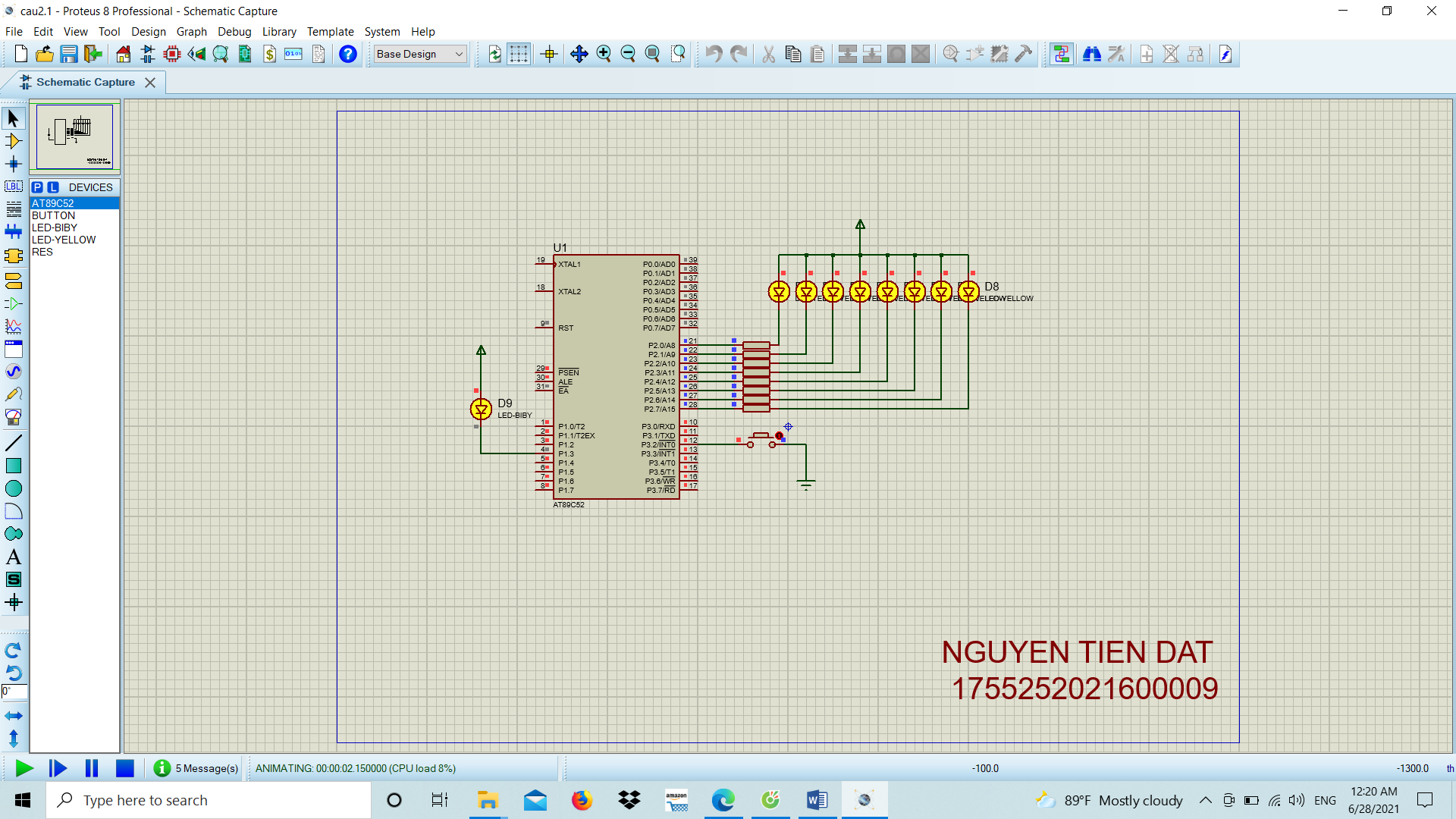
;

}

P1\_3 = 1;

}

* 1. **Mô phỏng Protues**



* 1. **Chương trình điều khiển**

#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 > 35)a = 0;

while(!BUT);

}

}

void main()

{

delay(500);

while(1)

{

nut\_nhan();

quet\_led();

dem++;

if(dem > 1)

{

b++;

dem = 0;

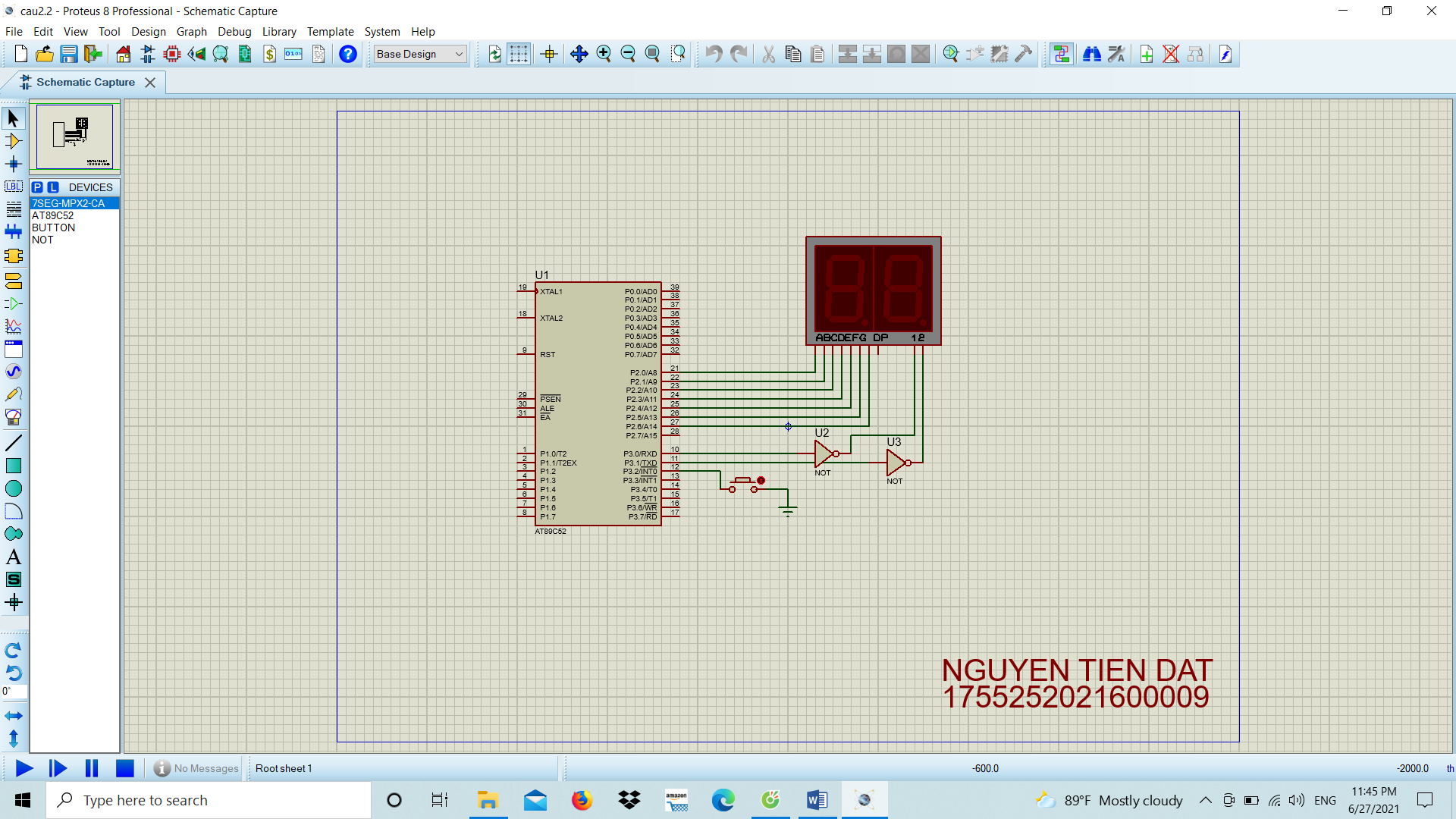
if(b > 35) b = 0;

}

}

}

* 1. **Mô phỏng Proteus**



**Câu 3.**

* 1. **Chương trình điều khiển**

#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\_4bit(unsigned char Data)

{

LCD\_D4 = Data & 0x01;

LCD\_D5 = (Data >> 1) & 1;

LCD\_D6 = (Data >> 2) & 1;

LCD\_D7 = (Data >> 3) & 1;

}

void LCD\_Cmd(unsigned char cmd)

{

LCD\_4bit(cmd >> 4);

LCD\_Enable();

LCD\_4bit(cmd);

LCD\_Enable();

}

void LCD\_Clear()

{

LCD\_Cmd(0x01);

delay\_us(10);

}

void LCD\_Init()

{

LCD\_4bit(0x00);

delay\_ms(20);

LCD\_RS = 0;

LCD\_RW = 0;

LCD\_4bit(0x03);

LCD\_Enable();

delay\_ms(5);

LCD\_Enable();

delay\_us(100);

LCD\_Enable();

LCD\_4bit(0x02);

LCD\_Enable();

LCD\_Cmd(0x28);

LCD\_Cmd(0x0C);

LCD\_Cmd(0x06);

LCD\_Cmd(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\_Cmd(address);

delay\_us(50);

}

void LCD\_PutChar(unsigned char Data)

{

LCD\_RS = 1;

LCD\_Cmd(Data);

LCD\_RS = 0;

}

void LCD\_Puts(char \*s)

{

while(\*s)

{

LCD\_PutChar(\*s);

s++;

}

}

void main()

{

LCD\_Init();

LCD\_Gotoxy(0, 0);

LCD\_Puts("NGUYEN TIEN DAT");

delay\_ms(1000);

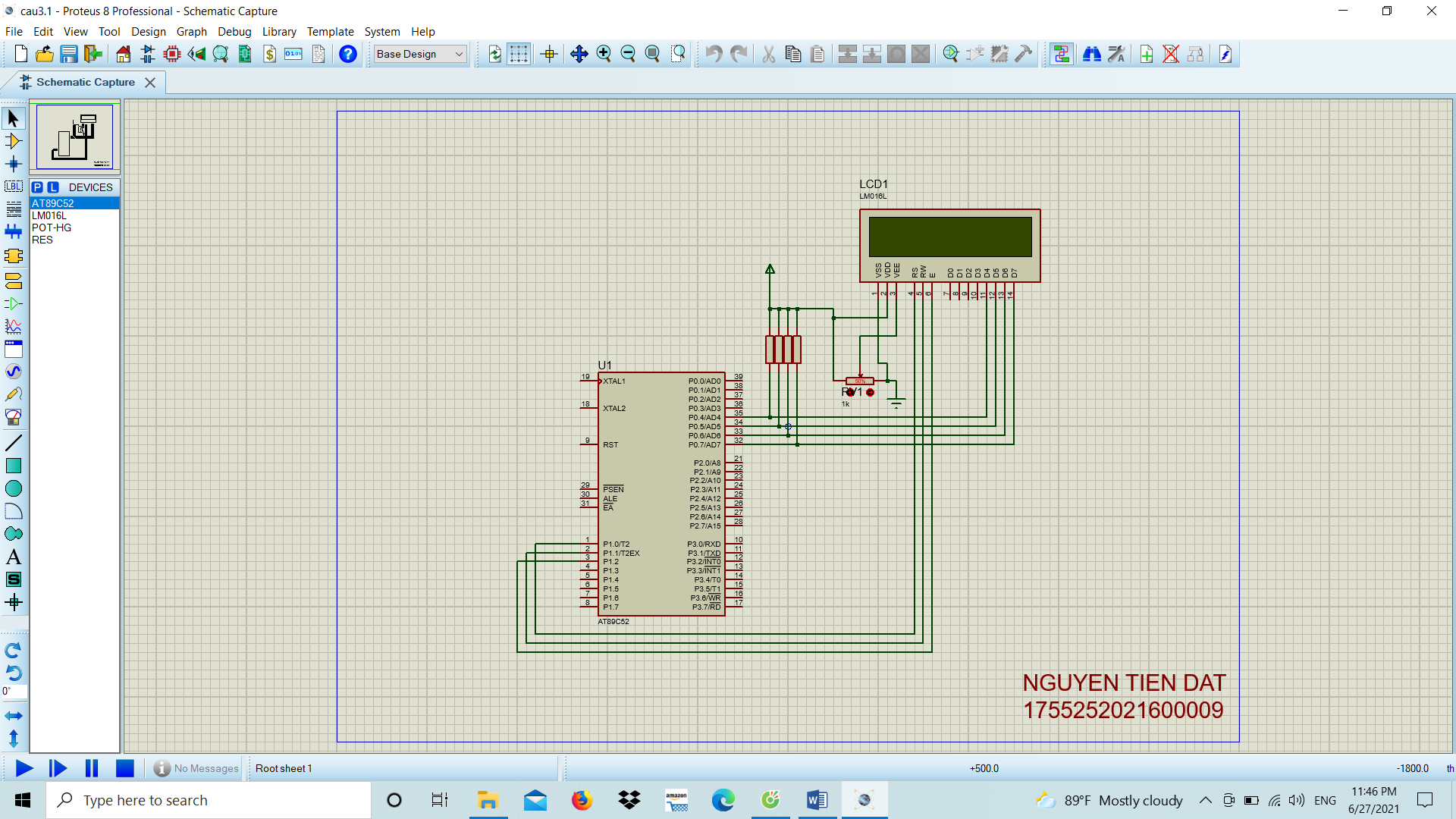
LCD\_Gotoxy(0, 1);

LCD\_Puts("1755252021600009");

while(1);

}

* 1. **Mô phỏng Proteus**



* 1. **Chương trình điều khiển**

#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);

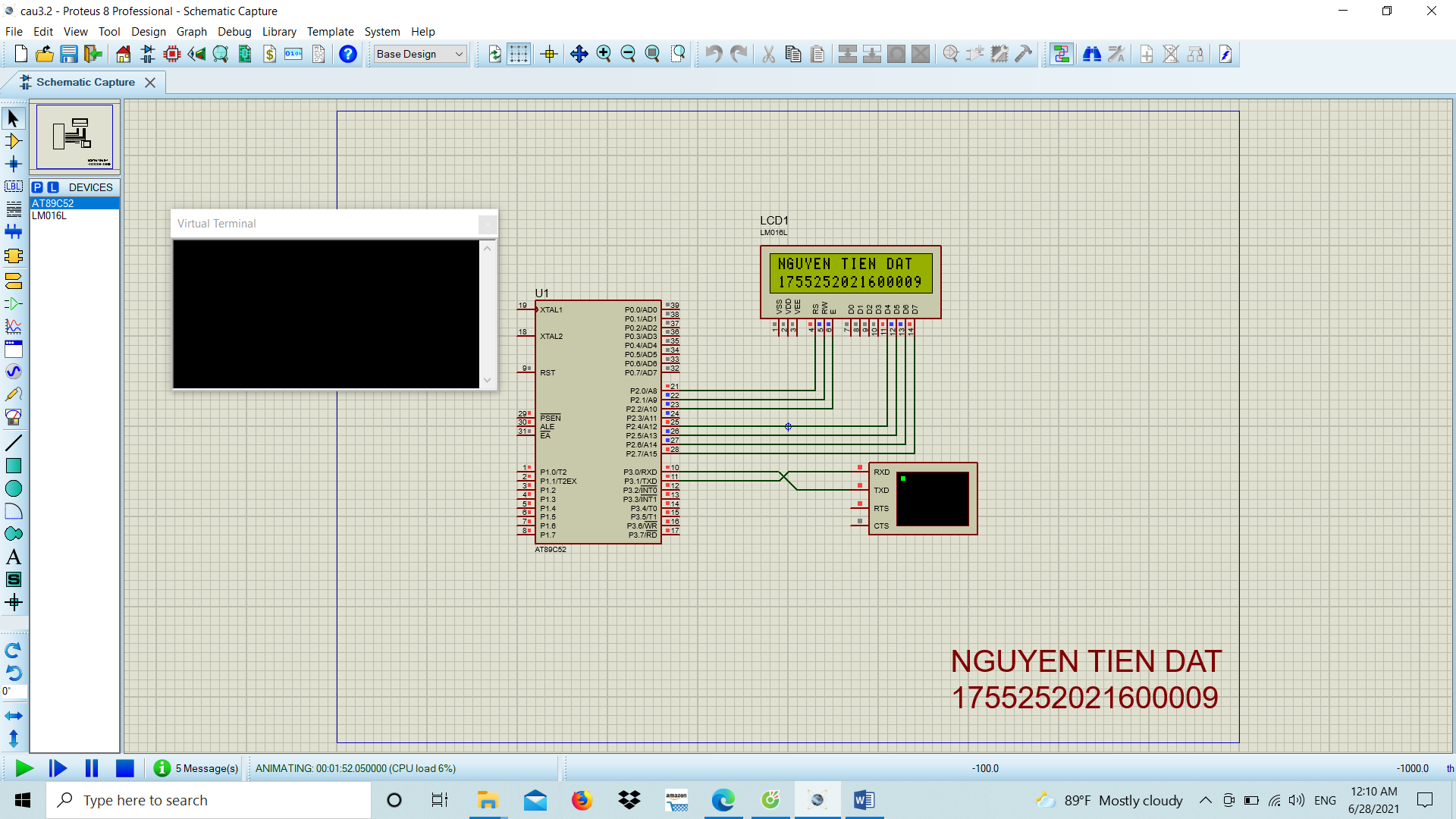
if(mybyte == 0x0D)

cmd(0xC0);

}

}

* 1. **Mô phỏng Proteus**



**Câu 4**.

**4.1. Chương trình điều khiển**

#include <rtx51tny.h>

#include <REGX51.H>

#define INIT 0

#define DO 1

#define BUTT 2

sbit LED = P1^2;

sbit BUTTON = P1^3;

void USART(void) interrupt 4

{

if(RI)

{

RI = 0;

isr\_send\_signal(DO);

}

}

void Startup(void) \_task\_ INIT

{

SCON = 0x52;

TMOD = 0x21;

TH1 = TL1 = -3;

TR1= 1;

IE = 0x90;

os\_create\_task(DO);

os\_create\_task(BUTT);

os\_delete\_task(INIT);

}

void LED\_DO(void) \_task\_ DO

{

while(1)

{

os\_wait2(K\_SIG, 50);

LED ^= 1;

}

}

void Button(void) \_task\_ BUTT

{

while(1)

{

if (BUTTON == 0)

{

os\_send\_signal(DO);

while(BUTTON == 0);

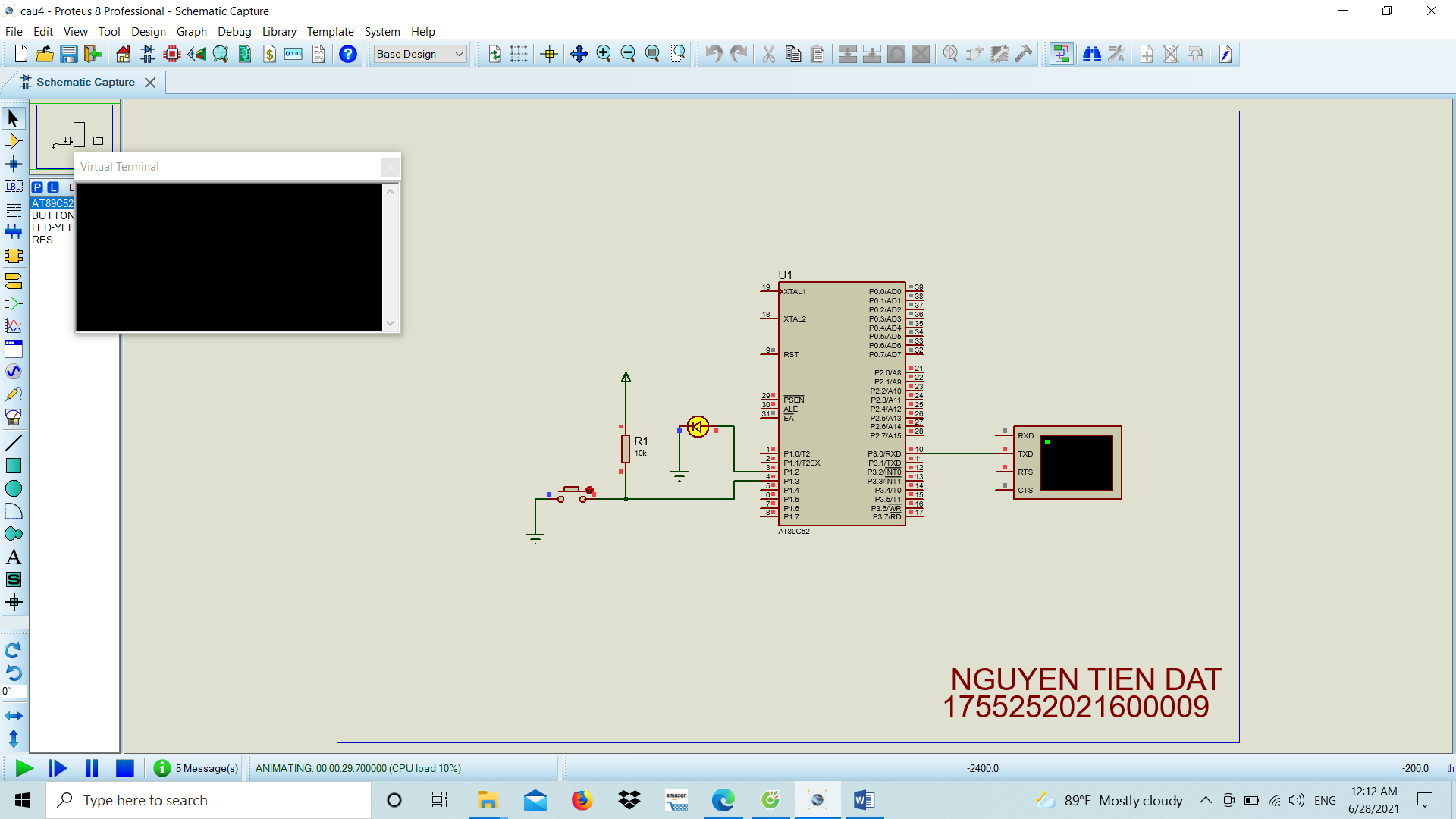
}

os\_wait2(K\_TMO, 10);

}

}

**4.2. Mô phỏng Proteus**



**Câu 5**. Hãy trình bày:

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.

1. **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,...