بسم الله الرحمن الرحیم

گزارش پروژه چهارم سیستم های نهفته

پاییز ۹۷

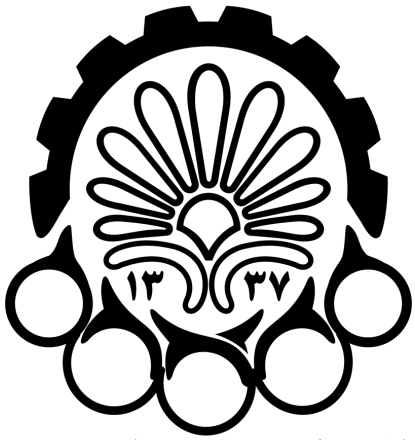
امیرحسین عباسی

سید محمدرضا حسینی

۹۳۳۱۰۷۰- ۹۳۳۱۰۰۸

دانشگاه امیرکبیر

دانشکده کامپیوتر و فناوری اطلاعات



دانشگاه صنعتی امیرکبیر

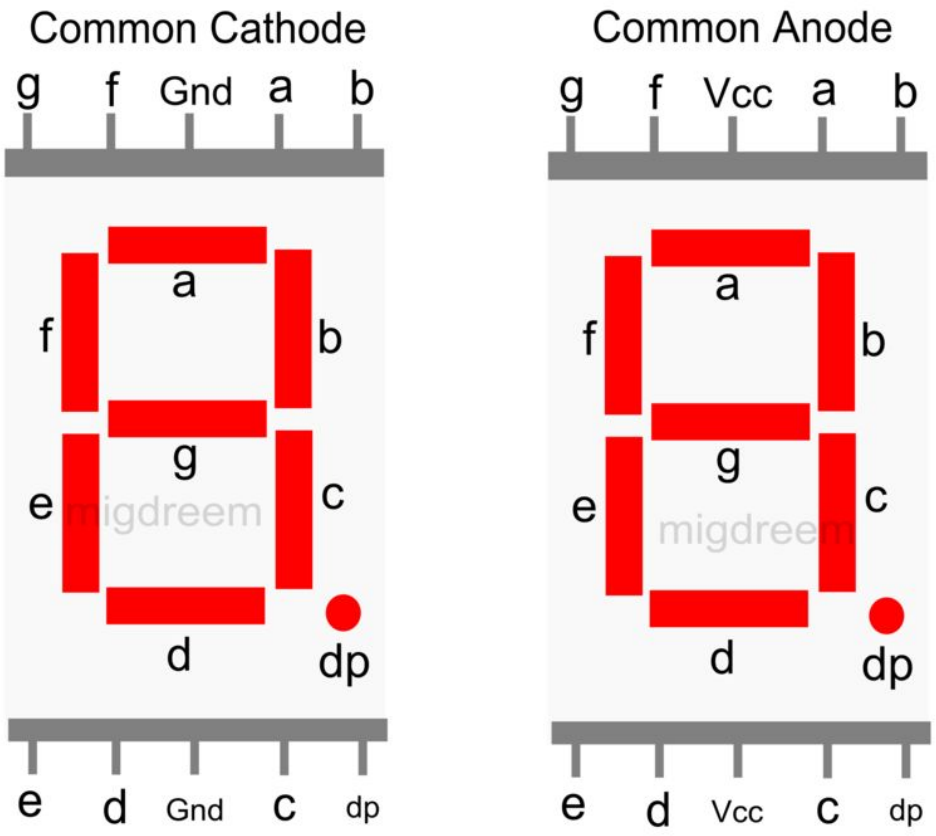
(پلی تکنیک تهران)

هدف این پروژه این است که ما بتوانیم یک scheduler بنویسیم که بتواند تسک های مختلف با اولویت های مختلف را مدیریت کند.

قسمت اول : قسمت RTC و Non Preemptive

```
BusOut display(PTA1,PTA2,PTD4,PTA12,PTA4,PTA5,PTC8,PTC9);
DigitalOut redled(PTE4);
DigitalOut greenled(PTE5);
InterruptIn switch1(PTD2);
InterruptIn switch2(PTD1);
InterruptIn switch3(PTA13);
```

ورودی های سیستم و خروجی ها و پورت آن ها را مشخص میکنیم. سه سوییچ و دو LED داریم.
همچنین برای 7 segment هم باید ۷ پورت در نظر بگیریم تا بتوانیم با کمک روشن کردن خط های a,b,c,d,e,f,g اعداد ۱ تا ۹ را نمایش دهیم.



نحوه اتصال 7segment به FRDM KL25Z به صورت تصویر بالا است.

پیاده سازی کلی قسمت non preemptive به این صورت است که ما دو آرایه داریم با نام های taskarr و ready2runArr.

```
int TASK=1;
int taskarr[6]={0,4,4,3,2,1};
int ready2runArr[6]={0,1,1,0,0,0};
```

آرایه ی taskarr : این آرایه یک آرایه static میباشد که برای ذخیره کردن اولویت تسک ها به ترتیب شماره آن ها استفاده میشود. به طوری که taskarr[3] اولویت تسک ۳ را به ما بدهد.

آرایه ی ready2runArr : این آرایه یک آرایه ی dynamic است که در زمان ران تایم تغییر میکند و تسک هایی که درخواست اجرای آن ها آمده است (به صورت دیفالت همه تسک ها با آمدن درخواست اجرا آماده به اجرا هستند) را نشان میدهد. این آرایه به این صورت است که ready2runArr[3]=1 نشان میدهد که درخواست تسک ۳ آمده است.
با زدن هر سوییچ در حین اجرای برنامه،روتین اینتراپت آن انجام میشود و در آن تسک متناظر با آن سوییچ در ready2runArr برابر با ۱ میشود تا بعد از کامل شدن تسکی که در لحظه در حال انجام است بتوانیم به درخواست ها پاسخ دهیم. با آمدن درخواست برای هر تسک مقدار متناظر با آن در این آرایه را ++ میکنیم

```
void intrupt1(){
   ready2runArr[3]++;
   }
void intrupt2(){
   ready2runArr[4]++;
    }
void intrupt3(){
   ready2runArr[5]++;
     }
```

همچنین یک متغیر با نام TASK داریم که تسکی که باید اجرا شود را نگه میدارد. در واقع هدف scheduler این است که این متغیر را با توجه به اولویت تسک ها به درستی مقداردهی کند.

برنامه main به این صورت است که در یک while (true) مرتبا متغیر TASK را چک میکند و تسک متناظر با آن را انجام میدهد.

```
int main() {
    switch1.rise(&intrupt1);
    switch2.rise(&intrupt2);
    switch3.rise(&intrupt3);
    while(1) {
        if(TASK==1){
          task1();
          }
        else if(TASK==2){
          task2();
          }
        else if(TASK==3){
          task3();
          }
        else if(TASK==4){
          task4();
          }
        else if(TASK==5){
          task5();
          }
    }
}
```

در صورت زدن سوییچ به روتین اینتراپت میرود و آرایه ready2runArr را آپدیت میکند. در پایان انجام هر تسک هم scheduler صدا میشود تا TASK درست با توجه به درخواست ها و اولویت ها انتخاب میشود.

پیاده سازی scheduler به این صورت است که به دنبال تسکی میگردد که هم درخواست آن آمده است (یعنی ready2run آن بالاتر از ۱ است) و هم اولویت بیشتری نسبت به سایر تسک های درخواست داده شده را دارد. این کار را با زدن یک for و گشتن به دنبال تسک با کمترین عدد اولویت (بالاترین اولویت) میگردیم و در صورتی که درخواست آن رسیده باشد TASK را مساوی آن قرار میدهیم.

```
void scheduler(){
   for(int i=0;i<6;i++){
      if(taskarr[TASK]>taskarr[i]){
         if(ready2runArr[i]==1){
            TASK=i;
            }
         }
      }
   }
```

پیاده سازی تسک ها :

تسک اول و دوم :

```
      redled = 1;
      wait(0.04);
      redled = 0;
      wait(0.04);
      scheduler();
```

با فرکانس ۰.۲۵ هرتز LED ها را روشن و خاموش میکنیم. در پایان آن تابع scheduler را صدا میکنیم تا تسک بعدی برای انجام را محاسبه کند.

تسک سوم :

```
      redled = 1;
      greenled = 0;
      wait(0.04);
      redled=0;
      greenled=1;
      wait(0.04);
      ready2runArr[3]=0;
      scheduler();
```

مانند همان تسک اول و دوم میباشد با این تفاوت که باید LEDها را یکی درمیان تاگل کنیم. تفاوت دیگر آن این است که در پایان آن قبل از صدا کردن scheduler باید آرایه ready2run را هم آپدیت کنیم و تعداد درخواست های تسک انجام شده را یکی کم کنیم.

تسک چهارم:

```
redled = 1;
 greenled = 1;
 wait(0.04);
 redled=0;
 greenled=0;
 wait(0.04);
 ready2runArr[4]=0;
 scheduler();
```

هر دو لامپ با هم روشن و خاموش میشوند. در اینجا هم باید ready2run را اپدیت کنیم

تسک پنجم:

```
  unsigned char i;
  unsigned char
arr[10]={0x40,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
  for (i=0; i<10; i++){
      display=arr[i];
      wait(1);
    }
    ready2runArr[5]=0;
    scheduler();
```

در اینجا یک آرایه را با اعداد هگز برای نشان دادن روی ۷ سگمنت ذخیره میکنیم. به اندازه ی یک ثانیه ویت میکنیم و یکی از اعداد را روی ۷ سگمنت display میکنیم.
برای RTC بودن تسک ها، تابع scheduler فقط در آخر تسک ها صدا میشود. در این صورت همه تسک ها تا پایان انجام میشوند و سپس تسک بعدی محاسبه میشود.

Number	g f e d c b a	Hexadecimal
0	0 1 1 1 1 1 1	3F
1	0 0 0 0 1 1 0	06
2	1 0 1 1 0 1 1	5B
3	1 0 0 1 1 1 1	4F
4	1 1 0 0 1 1 0	66
5	1 1 0 1 1 0 1	6D
6	1 1 1 1 1 0 1	7D
7	0 0 0 0 1 1 1	07
8	1 1 1 1 1 1 1	7F
9	1 1 0 1 1 1 1	6F

قسمت دوم : preemptive scheduling

برای preemptive بودن کافی است بعد از زده شدن هر سوییچ در روتین اینتراپت دوباره scheduler صدا زده شود تا در صورت بالاتر بودن اولویت تسک خواسته شده متغیر TASK آپدیت شود و در تابع main در حلقه بعدی وارد انجام شدن آن تسک شود.

```
void intrupt1(){
   ready2runArr[3]=1;
   scheduler();
   }
```