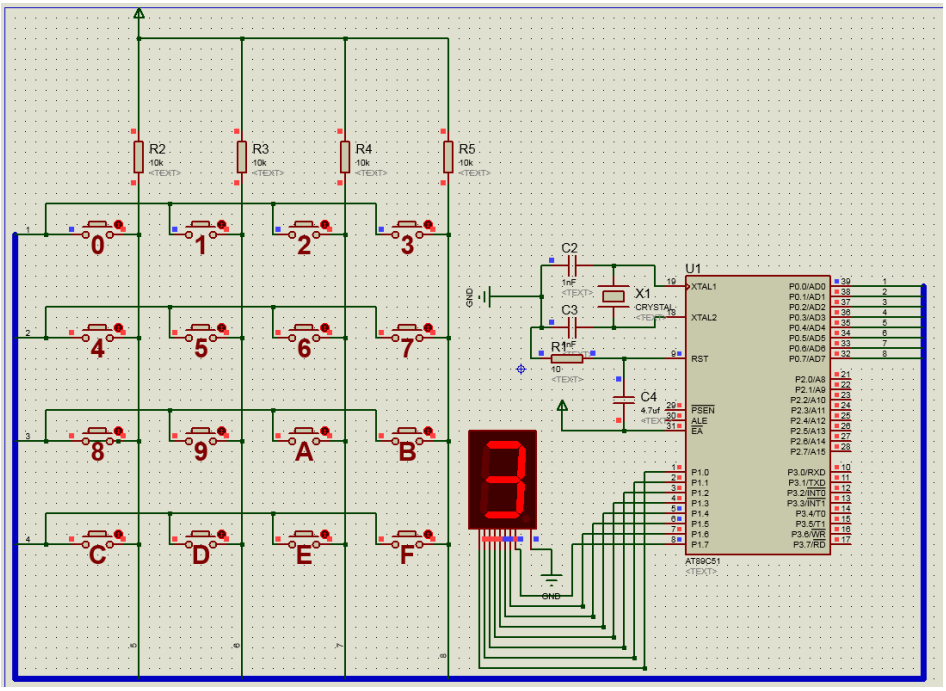
9.1　简述矩阵式键盘的工作原理。

答：矩阵式键盘中，行、列线分别连接到按键开关的两端，行线通过上拉电阻接到+5V 上。当无键按下时，行线处于高电平状态；当有键按下时，行、列线将导通，此时，行线电平将由与此行线相连的列线电平决定。这一点是识别矩阵按键是否被按下的关键。矩阵键盘中的行线、列线和多个键相连，各按键按下与否均影响该键所在行线和列线的电平，各按键间将相互影响，因此，必须将行线、列线信号配合起来作适当处理，才能确定闭合键的位置。

9.2　设计一个 4×4 按键电路和一位数码管显示电路，当按下某个键时，数码管显示该按键对应的编号，16 个按键的编号为 0~15，其中 0-9 编号直接显示数字，10-15 显示为字母 a~f，要求采用中断定时方式实现按键软件消抖（参照教材例 9.1 给出的定时中断处理消抖延时方法），请在 Proteus 中画出仿真电路，并编写程序进行测试验证。

解答：教材例 9-3 的电路如下，采用定时中断实现消抖延时。



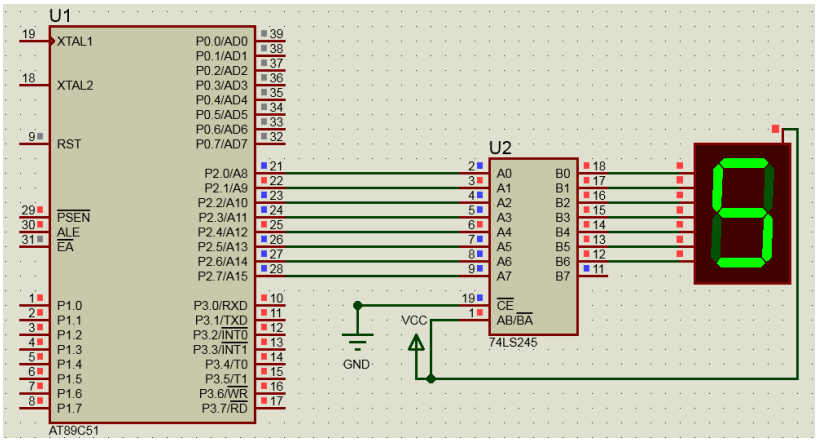

```
#include <reg51.h>
unsigned  char  KeyTime;
unsigned  char  RowFlag=0;  //用于标注消抖判断的行
bit KeyFlag=0;   //用于判断是否需要进行消抖处理
void InitTimer0() //初始化定时器
{
    TMOD = 0x01;//8 bit ->GATE C/T M1 M0  GATE C/T M1 M0
    TH0 = 0xF8;      //22.1184M,计数值 1843
    TL0 = 0xCD;      //设定值=65536-1843=63693=F8CDH
    ET0 = 1;     //允许定时器/计数器 0 的溢出中断
    TR0 = 1;     //启动定时器 0
    EA=1;
}//void InitTimer0 ()

void Timer0IRQ() interrupt 1 //定时器 0 中断，1 号中断
{
    TH0 = 0xF8;              //22.1184M,计数值 1843,1ms 中断一次
    TL0 = 0xCD;              //设定值=65536-1843=63693=F8CDH
    if(KeyTime >0)           //按键消抖计时参数
        KeyTime --;
```

```
}//void Timer0IRQ() interrupt 1

unsigned char KeyScan()
{
    unsigned char KeyData;
    unsigned char KeyNum=0;
    if(KeyFlag==0)   //说明还没有发生第一次按键
    {
        P0=0xfe;     //检测第一行
        KeyData =P0;
        KeyData = KeyData &0xf0;
        if(KeyData!=0xf0)
        {
            KeyFlag=1;   //第一次按键发生，需要进行消抖处理
            KeyTime=50;
            RowFlag=1;       //当前按键发生在第一行；
        }
        else
        {
            P0=0xfd;     //检测第二行
            KeyData =P0;
            KeyData = KeyData &0xf0;
            if(KeyData!=0xf0)
            {
                KeyFlag=1;  //第一次按键发生，需要进行消抖处理
                KeyTime=50;
                RowFlag=2;   //当前按键发生在第 2 行；
            }
            else
            {
                P0=0xfb;//判断第三行
                KeyData =P0;
                KeyData = KeyData &0xf0;
                if(KeyData!=0xf0)
                {
                    KeyFlag=1;  //第一次按键发生，需要进行消抖处理
                    KeyTime=50;
                    RowFlag=3;   //当前按键发生在第 3 行；
                }
                else
                {
                    P0=0xf7;//判断第四行
                    KeyData =P0;
                    KeyData = KeyData &0xf0;
                    if(KeyData!=0xf0)
                    {
                        KeyFlag=1;  //第一次按键发生，需要进行消抖处理
                        KeyTime=50;
                        RowFlag=4;   //当前按键发生在第 4 行；
                    }
                }
            }
        }
        return 0x55;
    }
```

```
else
{
    if(KeyTime==0)  //消抖时间完成，需要再次判断是否有键按下
    {
        KeyFlag=0;  //清除标志
        KeyData =P0;
        KeyData = KeyData &0xf0;
        if(KeyData!=0xf0)
        {
             switch(RowFlag)
             {
                case 1:   //第 1 行
                switch(KeyData)
                {
                    case 0xe0: KeyNum =0;break;
                    case 0xd0: KeyNum =1; break;
                    case 0xb0: KeyNum =2; break;
                    case 0x70: KeyNum =3; break;
                }
                break;
                case 2://第 2 行
                    switch(KeyData)
                    {
                         case 0xe0: KeyNum =4; break;
                         case 0xd0: KeyNum =5; break;
                         case 0xb0: KeyNum =6; break;
                         case 0x70: KeyNum =7; break;
                    }
                break;
                case 3://第 3 行
                switch(KeyData)
                 {
                     case 0xe0: KeyNum =8;  break;
                     case 0xd0: KeyNum =9; break;
                     case 0xb0: KeyNum =10; break;
                     case 0x70: KeyNum =11; break;
                 }
                break;
                case 4:  //第 4 行
                    switch(KeyData)
                    {
                         case 0xe0: KeyNum =12; break;
                         case 0xd0: KeyNum =13; break;
                         case 0xb0: KeyNum =14; break;
                         case 0x70: KeyNum =15; break;
                    }
                break;
                default:
                break;
             }
             return KeyNum;
        }
        else  //表示没有发生按键
        {
           RowFlag=0;   //清除按键行信息；
```

```
                    return 0x55;
                }
            }
        }
    }

    unsigned char
    CC[16]={0x3f,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6f,0x77,0x7c,0x39
    ,0x5e,0x79,0x71};//  0-9,A-f
    void main()
    {
        unsigned char KeyD;
        InitTimer0(); //初始化定时器
        while(1)
        {
            KeyD=KeyScan();
            if(KeyD<16)
                P1=CC[KeyD];  //送到数码管显示
        }
    }
```

9.3　设单片机的时钟频率为 12MHz，在 P2 口连接一个共阳极数码管，要求编程使之循环显示 0～9 数字，相邻数字显示的间隔为 1000ms。




```
#include <reg51.h>
//unsigned char CC[10]={0x3f,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6f};
//共阴极
unsigned char CC[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x90};
//共阳极
unsigned  int  LEDTime;
void InitTimer0() //初始化定时器
{
    TMOD = 0x01;//8 bit ->GATE C/T M1 M0  GATE C/T M1 M0
    TH0 = 0xFC;  //12M,计数值 1000,1ms 中断一次
    TL0 = 0x18;  //设定值=65536-1000=FC18H
    ET0 = 1;     //允许定时器/计数器 0 的溢出中断
    EA=1;
    TR0 = 1;     //启动定时器 0
}//void InitTimer0 ()

void Timer0IRQ() interrupt 1 //定时器 0 中断，1 号中断
{
```

```
    TH0 = 0xFC;      //12M,计数值1000,1ms中断一次
    TL0 = 0x18;      //设定值=65536-1000=FC18H
    if(LEDTime >0)   //按键消抖计时参数
        LEDTime --;
}//void Timer0IRQ() interrupt 1

void main()
{
    unsigned  char i=0;
    LEDTime=0;   //1000ms
    InitTimer0();
    while(1)
    {
        if(LEDTime==0)
        {
            P2= CC[i];
            i++;
            if(i==10)
                i=0;
            LEDTime=1000; //1000ms
        }
    }
}
```

9.4　结合工业测控、智慧城市等领域的应用需求，阐述人机交互在这些领域中的作用及设计思路。

参考答案：人机交互在智慧城市中城市内部按照人的活动参与多少分为两类系统：一种是自封闭工程性支撑系统，基于信息技术，实现完全自动化。工程性支撑系统由于本身能够精准量化、统一规制，并且自封闭循环，可以通过信息化手段，实现数据精细采集、精准监控和实时分析、反馈、调节，即“智慧”全自动化。另一种是复杂城市运行系统，基于大数据为支撑的集总效应判断，实现人机交互下的决策支撑。对城市中的众多复杂系统来说，以异构融合的多维时空数据为纽带，构筑跨学科的大数据运算平台，来支撑尽可能多的点状应用，实现对系统间集总效应的耦合分析，进而进行情景模拟，为治理决策提供支撑，这是其实现“智慧”的正确路径。

思路：1.实时监测评估。

基于采集、反馈的数据信息，对比社会治理的普遍底线（如公共安全、环境保护、违法犯罪等），一旦突破底线，实时提示报警。由此，实现对于影响社会治理公共底线行为的实时监测评估和预警分析。

例如：现在我国政府机关严查酒驾，倡导大家安全出行，但需要消耗较大的人力物力。目前的智能汽车已经初具市场，但在预防酒驾、智能化检测酒驾等方面尚未发现智能产品。为了方便驾驶员自主自查预防酒驾，节省社会资源，通过研究智能防酒驾系统以方便大家自查自测是否处于安全驾驶状态，从而减少安全事故的发生。在现有的酒精含量探测器的基础上，我们国家已经开始采用信息化技术手段，将酒精探测器、智能锁车、数据传输等结合在一起，实现及时检测，及时锁车，及时进行数据传输，从而降低酒驾事故的发生率，做到平安出行。

智能防酒驾系统主要由 STC89C52 单片机控制模块、呼气式酒精浓度传感器（NQ-3）检测模块、A/D 转换模块、继电器控制模块、语音报警模块、LCD 液晶显示模块和短信通知模块等组成。该系统是集成在仪表盘上的单片机，酒精传感器分别安装在前后排座位前方、上方以及左右两侧，以求达到更好检测酒精浓度的效果，单片机根据处理检测到的数据控制后续的汽车熄火系统、光电报警系统、语音报警系统、短信通知系统。

2. 情景模拟。城市以及社会发展一定是非线性或者超线性的，背后有一系列的复杂模型支撑。社会发展的情景模拟可以通过计算机媒体直观展示，保证所有内行、外行都可以参与评价。

例如：随着电子感应技术的进步，门禁系统得到了飞速的发展。现在全国逐步地正在由原来传统的机

械锁转变为具有智能化的感应式门禁系统。那我们就可以以单片机为核心控制器，研究刷卡和密码结合的双重门禁电路。

基于单片机的双重门禁电路设计，电路主要由主控电路、密码锁电路、刷卡电路、显示电路和继电器电路这五个部分组成。使用非接触式 IC 卡和 4×4 矩阵键盘作为输入，用直流继电器来控制电子锁的开启和关闭功能，同时在液晶显示屏上显示其状态。