**综合实验三 双音多频拨号识别系统设计**

**3130103382 包烨帆**

**负责部分：**

**基础实验部分：**

**1对于单个语音信号通过DFT和FFT两种方式进行检测。**

**拓展实验部分：**

**拓展实验部分：**

1. **与廖伟涵同学合作进行录音信号和实时声音收集进行解码分析VI搭建，以备孙涛同学进行声控门锁设计。**

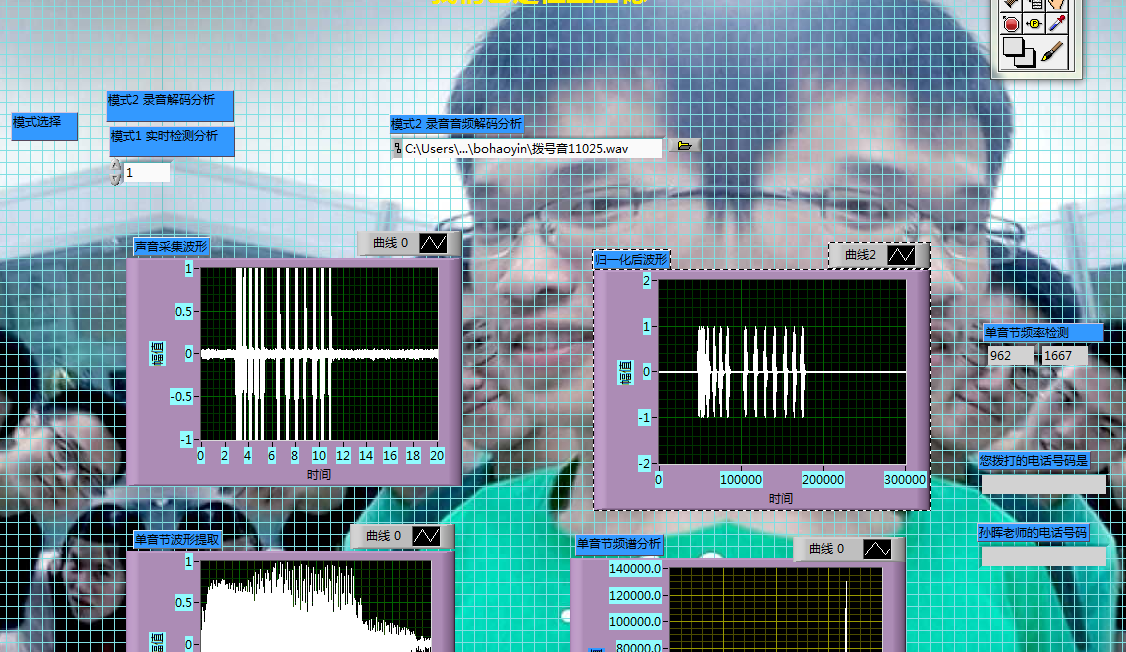
**一．实验原理及目的。**

**声音信号实时处理分析：**

**我们在基础实验部分对自己产生的DTMF信号进行解码分析，在这一点上有两点不足，首先我们没有进行噪音分析，分析的是纯净的DTMF信号，没有模拟空气中的环境，如果仅仅对由程序产生的DTMF信号进行检测，那么实际意义就小的多，或者说如基础实验部分一样认为加入白噪声，对于真实情况没有进行分析。其次我们针对一段连续的音节进行处理，而不是单个字符，如果仅仅对单个字符进行处理。那么我们进行声控门锁的识别应用性就小了很多，那我们最多只能进行16个音的识别，根据字符串或者是ASCII码数组进行后续处理。那么应用性就会广泛很多。下面就具体介绍实时处理部分的过程。**

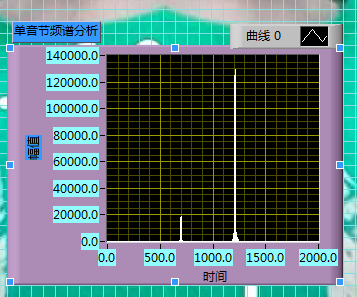
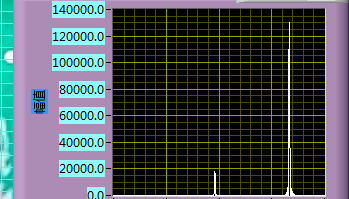
**二．实验过程记录。**

**总体界面设计：**

** 声音采集部分：**

**我们利用了Au软件进行录音收集，在录音过程中，我们发现了一点与文献报告中不同的地方，对于每一个DTMF信号的长度，文献资料中普遍定义为周期100ms，有效时间为45ms到55ms，而在实际录制中，信号长度达到160—170ms之间，这一点的发现也使我们对后面的分帧处理有了比较好的参考，一切均以实际情况为准。在录制完声音之后我们将文件分别保存为拨号音11025,，拨号音22000以及拨号音8000，这一点主要是针对不同的采样率进行分析，想对这一点分析也是因为在文献阅读中遇到的采样为什么均选用8000Hz，从实现功能考虑DTMF信号频率段在700Hz-1700Hz之间，已经完全满足奈奎斯特频率要求，只需要检测对应的8个频率，精度要求相对较低，能满足识别的要求，虽然这可能是众多研究者考虑的因素之一，但我觉得既然要实时检测，声音环境会复杂的多，那么我们就有必要考虑到底选择多少作为采样频率。我自身考虑的因素之一是，如果采样率更高，那么所有声音信号都会被还原地很准确，一般噪音信号的频率在100HZ到1000HZ之间那么噪音信号也会被相对完好的记录，可能会对我们的破译过程造成影响。当然我们也做了实验分析。**

**通过8000采样率获取的处理后的频谱波形如下图所示**

****

**图表 2 11025采样频率音节2频谱**

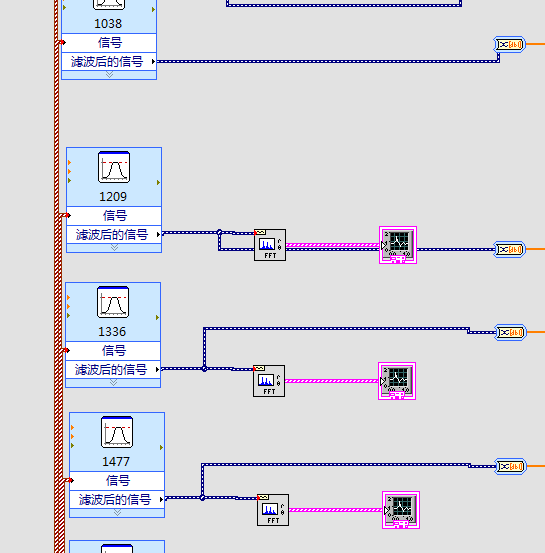
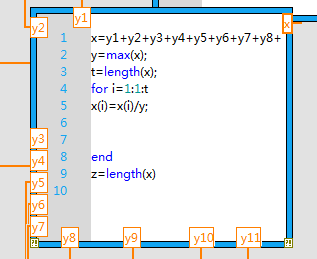
**图表 1 8000采样频率音节1频谱**

**左图为8000采样率下的解析结果，右图为11025采样率下的结果，如果运用更高的采样率效果会更明显，信号被还原的更加真实，有更加分明的毛刺和棱角分布，除此之外效果时近似的，这也要印证了我们最初的想法，我们并不是做音频处理，我们不需要更清晰更还原，我们需要的只是更准确地检测。那么相对而言8000采样率是毫无疑问的最佳选择。我们最终的处理都是基于这一标准，在保存过程中因为labview的导入格式问题，我们选择保存为wav 采样率8000模式，在保存中通过选项调整。（补充，在AU中可以直接对一段一个DTMF的信号进行窗口——频率分析，找出两个特征频率）。**

**对于第二种实时采样信号，我们直接采用的是信号处理中的声音采集功能，配置时间8000，持续时间20s，以便我们拥有足够的时间输入一段DTMF信号。**

**我们进入程序框图设计部分：**

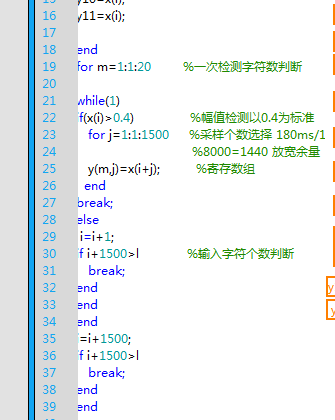
**第一部分：对于收录的信号，我们要先进行预处理，这其中一部分功能沿用了基础实验的部分滤波器组思想，但这合成信号中，由于之前廖伟涵同学利用的是合成信号，其实只是把所有信号叠加，而不是把信号相加处理，所以在导入后面的框图中出现了问题，我们原本以为是动态数据转换成数组过程中使用方法不当，进行了许多尝试，但都无法恢复原信号，直至我们咨询了班里对VI比较熟悉的同学，问题才得以解决。通过滤波器组部分，我们对于处理后的信号进行相加以及归一化处理，以幅度最高的时域信号设定为阈值1，其余声音信号幅度均在1以下。通过设定一定标准来找出鉴别DTMF信号的幅度，作为判断标准。**



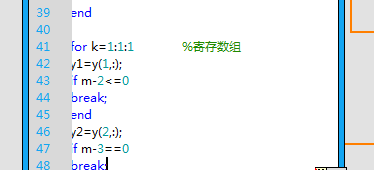
**图表 3 滤波器组模块**

**图表 4 幅值归一化**

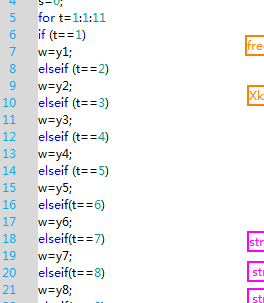
**第二部分：对预处理后的信号进行分帧处理，提取有效信息并分离，由于进入程序框图的声音信息已经失去了它的时间信息，我们首先要对其时间信息进行恢复，这边针对8000采样率的信号，我们计算得dtmf信号的时间，截取的长度大约为1500个采样点，对应180ms左右。其次是对界定DTMF信号的幅值判断，这一过程主要是通过模拟结果得到，在实验中我们一次输入0到9以及ABCD\*#的信号，尝试在同一台电脑，电脑之间，手机与电脑之间测试，电脑之间我分别尝试了廖伟涵和梁志坚同学的电脑，大多数情况，声音的幅度都近似相等，只要在与廖伟涵同学电脑中发出拨号音，接收时，拨号音最高0的幅度较拨号音最低3的幅度大近2倍，并且每一个行频之间都存在一定幅度差，后来考虑声音输出过程中其电脑已经添加了滤波器，综合考虑之下，我们还是最终选择0.4为相对幅度。在完成这一过程后，我们通过数组寄存多维数组中寄存的数据，输出至解码模块算法进行处理。**

****

**图表 5 分帧处理模块**



**图表 6 寄存模块**

**第三部分 解码部分。我这边主要采用了FFT算法进行识别。**

**图表 7 确定字符串位置**

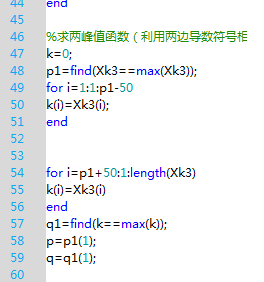
**1.首先我们要对字符的位数进行判断，判断位输入字符串的第几位。**

1. **利用fft算法进行频域分析，对于每一个音节识别率进行测定**

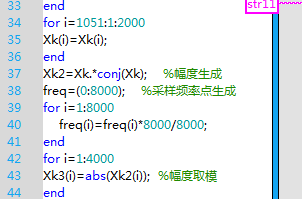
**FFT解码误差分析表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 标定频率/Hz | 实际频率/Hz | 差值/Hz | 百分误差 |
| 697 | 687.5 | 9.5 | 0.01363 |
| 770 | 781.25 | 11.25 | 0.01461 |
| 852 | 843.75 | 8.25 | 0.009683 |
| 947 | 937.5 | 9.5 | 0.010032 |
| 1209 | 1218.75 | 9.75 | 0.008065 |
| 1336 | 1343.75 | 7.75 | 0.005801 |
| 1477 | 1468.75 | 8.25 | 0.005586 |
| 1633 | 1625 | 8 | 0.004899 |

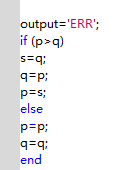
**在基础实验部分孙涛同学已经制作出戈泽尔算法以及自编改进戈泽尔算法，从识别度上讲两者的误差度相近，戈泽尔算法只关心信号幅度谱的分量，所需内存空间较少，且灵活性更强，计算量也更小，但本质上的核心仍然是FFT算法。在这边DFT,FFT,以及戈泽尔算法，以及自编戈泽尔算法四种解码方式均可采用。具体方式在基础实验报告上已有详细说明，不再赘述。**

****

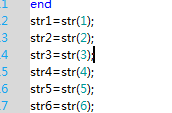
**图表 9 特征频率提取模块**

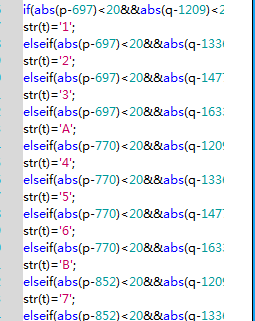
****

**图表 8 fft解码模块**

**3．分析频谱获取两个峰值频率，在理想DTMF信号解码得到的频谱中，频谱是较为规律的，在两个特征频率两侧分别连续增减，所以我们直接采用函数IndMax=find(diff(sign(diff(Xk3)))<0)+1得到两个特征频率值，但是我们也可以从图一与图二中看出，频谱幅度的变化往往不是那么规律的，有许多毛刺及不规律变化出现。我们提出了两种方案，一种是设定一个赋度值，对其以上导数变化进行记录，但这一种方案很快被否定了，因为频谱分析中每个音节的幅度难以界定。我们采用了直接求取最大值的方法，在取得第一个最大频率幅度点后，排除周围100个频率点，再次求最大值，获得两个频率点。这样操作可能会造成行频与列频的错位，需要再加上一个判断交换。**

**图表 10 行列频率判断**

**4.最后一部分，根据两个特征频率判断拨号音，连接成字符串。**

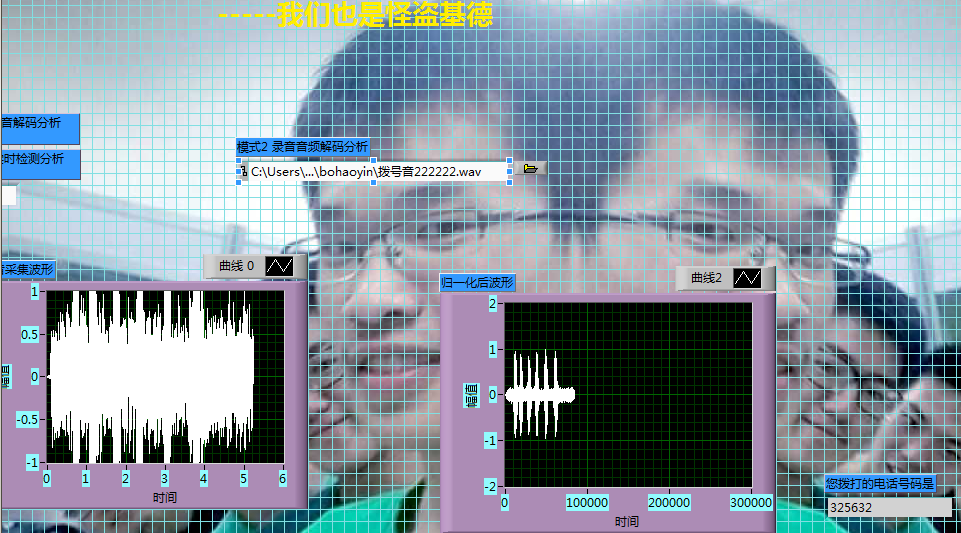


**图表1 2 字符串分离以备拼接**

**图表 11 特征频率匹配拨号音**

**三．实验演示。**

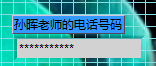
**1.录音模式事先输入一段录制好的拨号音，保存为wav 8000采样率格式导入，右下角检测出拨打的按键音。**

****

**图表 13 录音部分功能展示**

**2.实时检测部分展示。**

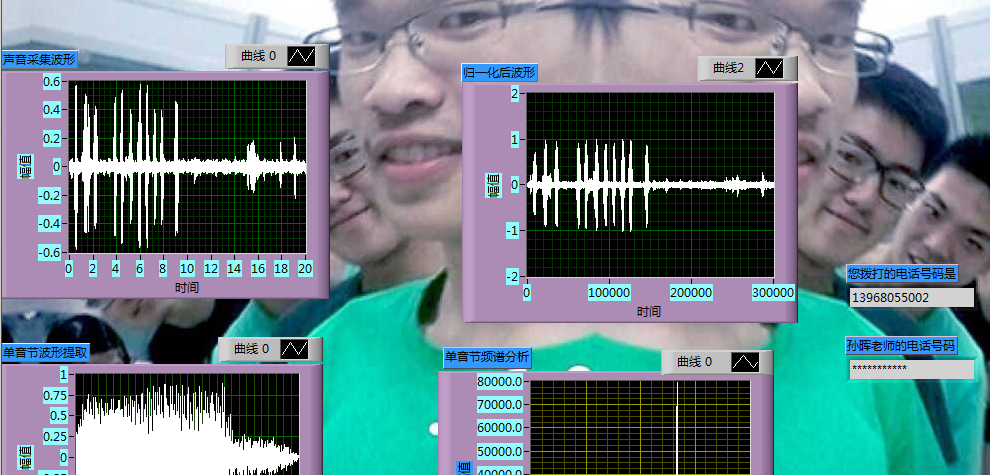
**以孙晖老师的电话号码作为距离，输入孙晖老师电话号码作为密码。**

** 为直观演示转换为显示模式。**

****

**图表 14 密码设置**

**此时我们将模式选择调节至1，通过另一台电脑发出拨号音。**

****

图表 5 实时检测部分演示

**从上图中可以看出，左上图为声音采集的波形，包含噪声较多，右上为进行滤波器组以及归一化处理后的波形，dtmf信号音幅度保持较为一致，噪音分量明显减小，左下为截取的某一单一音节的截取波形，我们设置的截取点数相对这一拨号音是稍长的，因为拨号音的长短有一定范围，需要留有一定裕量，最后为最后一个音节的频谱分析。**

1. **如果字符串匹配成功，那么音频就会解锁，播放巴赫，哥德堡变奏曲。**

**四．实验心得及需要改进之处。**

**1.从准确性上讲，这个系统不存在什么误差，但操作较为麻烦，如果不写很好的说明，可能很难入手。**

**2.如果完全不输入拨号音，也会检测这些频率的幅度值，如果存在对应点附近的频率，可能会被检测到，其实我们针对不同方式声音传输，最后确定的相对幅度值0.4是偏小的，一般情况下0.7左右就完全足够了。这也是为了当场演示完全正确组内决定的一个改动。**

**3.我最初考虑用寄存器做完全实时的检测，但后来想没有意义，因为你无法判断起始位，检测到的是一段源源不断的数字，如果有噪音也会被收录，通过我这种方式检测当有一个正确的DTMF信号时，匹配频率的噪音是不会被收录进来。**

**4.在VI中应当加一个指示灯，表示每次检测开始时的信号，现在仅能通过波形变换。**

**五．个人总结。**

**我做的这一部分，主要是对实际情况进行分析，其实对于实时收录信号的处理方式有很多种，我可以说我这种检测方式对于声音处理的应用性很小，但如果仅仅对于检测，个人认为已经是相当好的一种算法，对于学弟学妹们的建议，对于实时系统的完善可能是一方面，因为开始时这一部分由两个人完成，后期由我独自完善，可能一个人的想法有很多不足之处。另外可以考虑一下如果存在回删拨号键，如果实现密码匹配。**

**包烨帆 3130103382**

**2015年6月15日午后**

附录：源代码。

%预处理：

**x=y1+y2+y3+y4+y5+y6+y7+y8+y9+y10+y11;**

**y=max(x);**

**t=length(x);**

**for i=1:1:t**

**x(i)=x(i)/y;**

**end**

**z=length(x)**

**%分桢，提取有效信号**

**=1;**

**l=length(x);**

**y=0;**

**m=1;**

**for i=1:1:1500 %对数组赋初值**

**y1=x(i);**

**y2=x(i);**

**y3=x(i);**

**y4=x(i);**

**y5=x(i);**

**y6=x(i);**

**y7=x(i);**

**y8=x(i);**

**y9=x(i);**

**y10=x(i);**

**y11=x(i);**

**end**

**for m=1:1:20 %一次检测字符数判断**

**while(1)**

**if(x(i)>0.4) %幅值检测以0.4为标准**

**for j=1:1:1500 %采样个数选择 180ms/1000ms\***

**%8000=1440 放宽余量**

**y(m,j)=x(i+j); %寄存数组**

**end**

**break;**

**else**

**i=i+1;**

**if i+1500>l %输入字符个数判断**

**break;**

**end**

**end**

**for k=1:1:1 %寄存数组**

**y1=y(1,:);**

**if m-2<=0**

**break;**

**end**

**y2=y(2,:);**

**if m-3==0**

**break;**

**end**

**y3=y(3,:);**

**if m-4==0**

**break;**

**end**

**y4=y(4,:);**

**if m-5==0**

**break;**

**end**

**y5=y(5,:);**

**if m-6==0**

**break;**

**end**

**y6=y(6,:);**

**if m-7==0**

**break;**

**end**

**y7=y(7,:);**

**if m-8==0**

**break;**

**end**

**y8=y(8,:);**

**if m-9==0**

**break;**

**end**

**y9=y(9,:);**

**if m-10==0**

**break;**

**end**

**y10=y(10,:);**

**if m-11==0**

**break;**

**end**

**y11=y(11,:);**

**if m-12==0**

**break;**

**end**

**y12=y(12,:);**

**if m-13==0**

**break;**

**end**

**y13=y(13,:);**

**if m-14==0**

**break;**

**end**

**y14=y(14,:);**

**if m-15==0**

**break;**

**end**

**y15=y(15,:);**

**if m-16==0**

**break;**

**end**

**y16=y(16,:);**

**if m-17==0**

**break;**

**end**

**y17=y(17,:);**

**if m-18==0**

**break;**

**end**

**y18=y(18,:);**

**if m-19==0**

**break;**

**end**

**y19=y(19,:);**

**if m-20==0**

**break;**

**end**

**end**

**%FFT算法**

**str=' ';**

**s=0;**

**for t=1:1:11**

**if (t==1)**

**w=y1;**

**elseif (t==2)**

**w=y2;**

**elseif (t==3)**

**w=y3;**

**elseif (t==4)**

**w=y4;**

**elseif (t==5)**

**w=y5;**

**elseif(t==6)**

**w=y6;**

**elseif(t==7)**

**w=y7;**

**elseif(t==8)**

**w=y8;**

**elseif(t==9)**

**w=y9;**

**elseif(t==10)**

**w=y10;**

**elseif(t==11)**

**w=y11;**

**end**

**%sound(y10)**

**Xk=fft(w,8000);**

**for i=1:1:1050**

**Xk(i)=Xk(i);**

**end**

**for i=1051:1:2000**

**Xk(i)=Xk(i);**

**end**

**Xk2=Xk.\*conj(Xk); %幅度生成**

**freq=(0:8000); %采样频率点生成**

**for i=1:8000**

**freq(i)=freq(i)\*8000/8000;**

**end**

**for i=1:4000**

**Xk3(i)=abs(Xk2(i)); %幅度取模**

**end**

**%求两峰值函数（利用两边导数符号相异）**

**k=0;**

**p1=find(Xk3==max(Xk3));**

**for i=1:1:p1-50**

**k(i)=Xk3(i);**

**end**

**for i=p1+50:1:length(Xk3)**

**k(i)=Xk3(i)**

**end**

**q1=find(k==max(k));**

**p=p1(1);**

**q=q1(1);**

**%字符串输出显示**

**%误差取20**

**output='ERR';**

**if (p>q)**

**s=q;**

**q=p;**

**p=s;**

**else**

**p=p;**

**q=q;**

**end**

**if(abs(p-697)<20&&abs(q-1209)<20)**

**str(t)='1';**

**elseif(abs(p-697)<20&&abs(q-1336)<20)**

**str(t)='2';**

**elseif(abs(p-697)<20&&abs(q-1477)<20)**

**str(t)='3';**

**elseif(abs(p-697)<20&&abs(q-1633)<20)**

**str(t)='A';**

**elseif(abs(p-770)<20&&abs(q-1209)<20)**

**str(t)='4';**

**elseif(abs(p-770)<20&&abs(q-1336)<20)**

**str(t)='5';**

**elseif(abs(p-770)<20&&abs(q-1477)<20)**

**str(t)='6';**

**elseif(abs(p-770)<20&&abs(q-1633)<20)**

**str(t)='B';**

**elseif(abs(p-852)<20&&abs(q-1209)<20)**

**str(t)='7';**

**elseif(abs(p-852)<20&&abs(q-1336)<20)**

**str(t)='8';**

**elseif(abs(p-852)<20&&abs(q-1477)<20)**

**str(t)='9';**

**elseif(abs(p-852)<20&&abs(q-1633)<20)**

**str(t)='C';**

**elseif(abs(p-941)<20&&abs(q-1209)<20)**

**str(t)='\*';**

**elseif(abs(p-941)<20&&abs(q-1336)<20)**

**str(t)='0';**

**elseif(abs(p-941)<20&&abs(q-1477)<20)**

**str(t)='#';**

**elseif(abs(p-941)<20&&abs(q-1633)<20)**

**str(t)='D';**

**else**

**str(t)=' ';**

**end**

**end**

**str1=str(1);**

**str2=str(2);**

**str3=str(3);**

**str4=str(4);**

**str5=str(5);**

**str6=str(6);**

**str7=str(7);**

**str8=str(8);**

**str9=str(9);**

**str10=str(10);**

**str11=str(11);**