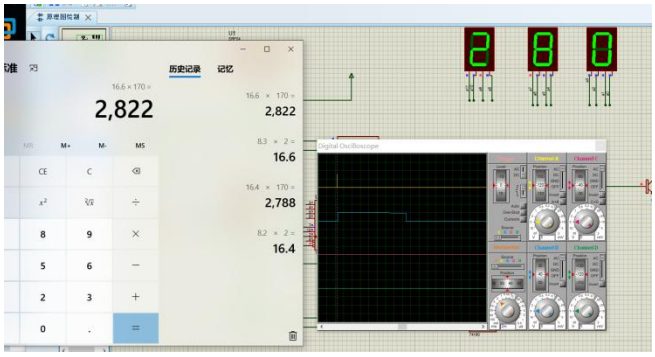
综合实验 1

B 级任务（90%）

阅读教材，观察传感器实物、两组特定距离的 trig 和 echo 的示波器捕获波形，认识超声波测距传感器工作原理，在 Proteus 上用 SRF04 传感器、蜂鸣器 BUZZER 和数字电路芯片设计一个能单次启动测距系统电路。参考基本电路图如下，请在此电路图上继续完成所要求的功能：

1，了解整个超声波测距工程的演示与测量过程，对超声波测距传感器的工作原理进行整体认知；
实验开始之前，通过单次发射，button 开关，会产生一个 trig 信号持续时间大于 10us，此时超声波模块自动发送很多个连续的方波信号，然后检测是否有回响信号，通过回响信号高电平持续时间来计算距离，并根据此距离来选择相应的频率使得蜂鸣器发出不同的声音

2、仔细看 trig 和 echo 信号的示波器波形，对其进行分析，了解其计算过程并说明；

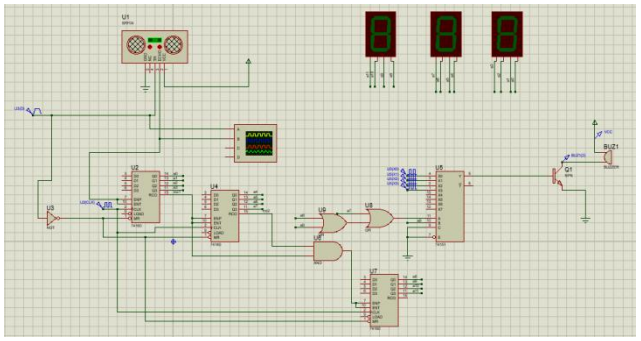


按道理来讲，持续时间 t*340/2 等于距离，我们需要根据 1cm 来设置相应的始终频率，所以 0.01=t*170,t=1/1700,然后换成频率就是 17000hz，所以频率应该设置成这个值，但设置这个值得到的结果会存在误差，后来通过多次不断实验计算对数据进行处理，最后的 echo 计数器的时钟频率设置成 17.25khz

发现 echo 回响信号高电平持续时间是差不多 8.2 或者 8.3 个格子，每个格子通过示波器右下角可知是 2ms，那么通过计算 8.3*2*340*/2/10=282.2，与 280cm 相近，验证结果正确

二、在 Proteus 上用 SRF04 传感器、蜂鸣器 BUZZER 和和数字电路芯片设计一个能单次启动测距系统的电路。

总体电路：

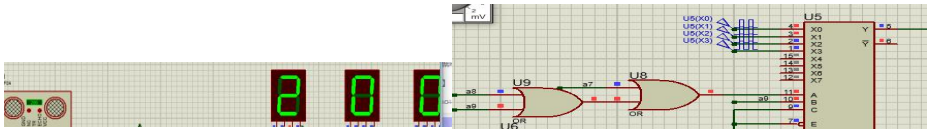


右下角是蜂鸣器模块，通过将 a7 和 a8,a9 一起或门相连接上 74151 选择端，以及 a9 单独接到 74151 选择端可以对三种频率进行选择（虽然接了四种频率）由此可以使得蜂鸣器对不同距离进行抱警划分的范围是 0~79 80~199 200~399，因为本实验的距离范围是 8 位，所以距离最大 255，在所表示的范围之内。

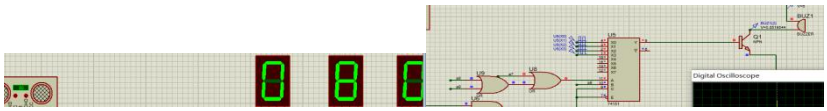
左边超声波模块接受自己设置的 trig 信号会返回一个 echo 回响信号，并将信号传给了示波器，可以在示波器中观察到相应的波形，左下方是三个 74160 十进制计数器级联，来记录 echo 回响信号的持续时间，通过计数次数来计时，所以需要自己设置好时钟频率，根据实验多次测量 17.25khz 是最佳时钟频率（上面已经分析了），右上方是 7 段码 bcd 数码管显示，接上相应的四位，a3~a0 是低位（个位），a7~a4 是十位，a11~a8 是百位，连接上对应

的数码管显示相应的距离（由于是十进制计数器所以 8 位 2 进制的距离需要 3 个十进制表示的数码管来显示），右下方是蜂鸣器选择模块（已做分析不做赘述）

结果验证

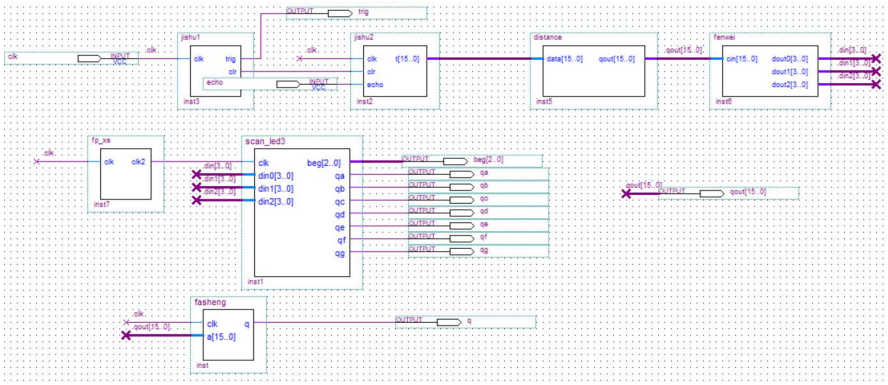


距离为 200 的时候选择位电平都为高所以选择第 11 路的频率



距离为 8 的时候 a7 为高电平，所以选择 01 路的频率

综合实验 2



要求：

①以24MHz作为输入时钟源，规划好时序分配并记录。

时序应包括：

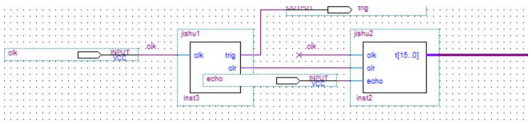
超声波模块，用于产生Trig启动信号的脉冲的时钟源和用于echo脉宽计数器的时钟信号。

状态机，用于切换每1秒开始新一轮测量的时钟源。

蜂鸣器，用于产生上周内容中单音频率和间隔的时钟源。

数码管，用于产生多位数码管扫描频率和显示数据刷新时间的时钟源。

超声波模块



此模块内容包括了状态机模块每一秒开始新一轮测量时钟源的功能

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity jishul is
    port(clk:in std_logic;
         trig,clr:out std_logic);
end entity jishul;
architecture arch of jishul is
signal j:std_logic_vector(15 downto 0):="1111111111111111";
begin
    process(clk)
    begin
    if(j="1111111111111111") then
        trig<='1'; clr<='1'; j<="0000000000000000";
    elsif(clk'event and clk='1') then
        j<=j+1; clr<='0';
        if(j="0000000000011111") then
            trig<='0';      --trig
        end if;
    end if;
end process;
end arch;
```

Trig 信号设置为 32 次计数后就会归 0，然后本次实验要求时钟频率设置为 24mhz，差不多 0.5us 为了便于验证波形设置成 500ns，代码中可看出一共计数 32 次 trig，然后才会让 trig 归 0，那么我们的 trig 会在第 32 次的时钟上升沿变为 0，32*500*10^-3 就是 16us，然后超声波模块会给我们的电路在 8 个时钟周期后发回 echo 信号，我们假设 echo 信号是 32 个时钟周期那么距离就应该是 32*170*500/10000=272cm

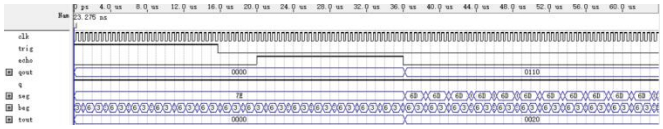
距离计算模块

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity jishu2 is
    port(clk,clr,echo:in std_logic;
         t:out std_logic_vector(15 downto 0));
end jishu2;

architecture arch of jishu2 is
signal j:std_logic_vector(15 downto 0):="0000000000000000";

begin
    process(clk,clr,echo)
    begin
    if(clr='1') then
        j<="0000000000000000";
    elsif(j="1111111111111111") then
        j<="0000000000000000";
    elsif(clk'event and clk='1') then
        if(echo='1') then
            j<=j+1;
        else  t<=j;
        end if;
    end if;
end process;
end arch;
```

echo 回响计数

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
entity distance is
port(data:in std_logic_vector(15 downto 0);
     qout:buffer std_logic_vector(15 downto 0));
end distance;
architecture rtl of distance is
signal tn:integer;
signal ans:integer;
signal ansl:integer;
begin
process(tn,ans,qout)
    begin
    tn<= conv_integer(data);
    ans<=85*(tn)/10;
    qout<=conv_std_logic_vector(ans,16);
end process;
end rtl;
```

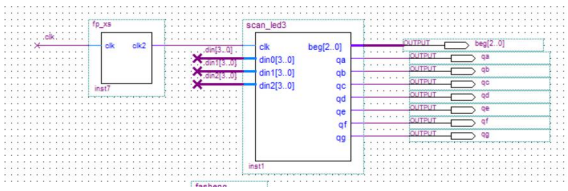
针对 echo 持续时间和上一个模块传过来的计数器的次数来计算距离，由于是 500ns Count*500*340*10^-6/2=85*count/10



与理论预算结果符合十六进制 qout 就是我们的距离 0110 转换成十进制是 272，且 echo 计数器十六进制 20 表示的十进制数就是 32 也与我们设定的 echo 输入相符合

数码管

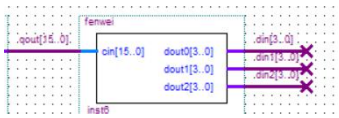
scan_led3



```
library ieee;
use ieee.std_logic_1164.all;
entity fp_xs is
port (clk:in std_logic;
      clk2:out std_logic);
      end fp_xs;
architecture rtl of fp_xs is
signal count0:integer;
signal out1:std_logic;
begin
    process(clk)
        begin
            if clk'event and clk='0' then
                if count0=2500 then
                    count0<=0;
                    out1<=not out1;
                elsif count0<2500 then
                    count0<=count0+1;
                else
                    count0<=0;
                end if;
            end if;
    clk2<=out1;
    end process;
end rtl;
```

数码管显示电路，左边连接了 fp_xs 通过修改原始时钟信号，通过调节频率的方式（扫描频率太低数码管会出现闪烁的现象，频率太高则亮度不够甚至无法看清，所以一般扫描间隔多为几毫秒，这里设置时钟频率为 1ms 左右）保证能够在数码管上看清数值。

fenwei

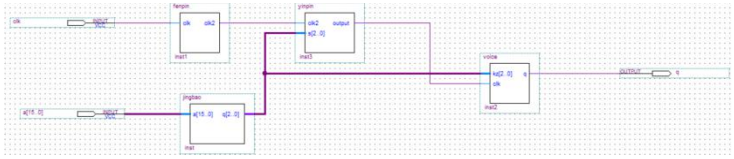


```
begin
    process(cin_Reg)
    begin
        if(cin>"0000001111100111")then
            cin_Reg<="0000001111100111";
        else
            cin_Reg<=cin;
        end if;
    end process;
    dout_Reg<=conv_integer(cin_Reg);
    bb<=dout_Reg/100;
    ss<=(dout_Reg-bb*100)/10;
    gg<=(dout_Reg-bb*100-ss*10);
    b_Reg<=conv_std_logic_vector(bb,4);
    s_Reg<=conv_std_logic_vector(ss,4);
    g_Reg<=conv_std_logic_vector(gg,4);
    dout0<=b_Reg;
    dout1<=s_Reg;
    dout2<=g_Reg;
```

将输出距离划分成三段可用十进制表示

蜂鸣器

顶层图



fenpin

```
library ieee;
use ieee.std_logic_1164.all;
entity fenpin is
port(clk:in std_logic;
     clk2:out std_logic);
end entity;
architecture one of fenpin is
signal count0:integer:=0;
signal out1:std_logic:='0';
begin
    process(clk,count0)
        begin
        if(clk'event and clk='0')the
            if(count0=50)then
                count0<=0;
                out1<=NOT out1;
            else
                count0<=count0+1;
            end if;
        end if;
    end process;
    clk2<=out1;
end one;
```

这一模块首先连接在时钟源上，保证进行第一次分频，保证信号是占空比为百分之五十的标准信号

yinpin

```
        end if;
    end process;
    process(clk2,count2)
        begin
        if(clk2'event and clk2='1')then
            if(count2=5000)then
                count2<=0;
            else
                if(count2<2000)then
                    out2<=NOT out2;
                else
                    out2<='0';
                end if;
                count2<=count2+1;
            end if;
        end if;
    end process;
    process(clk2,count3)
        begin
        if(clk2'event and clk2='1')then
            if(count3=10000)then
                count3<=0;
            else
                if(count3<2000)then
                    out3<=NOT out3;
                else
                    out3<='0';
                end if;
                count3<=count3+1;
            end if;
```

仅展示部分代码，通过改变占空比，来使得蜂鸣器的报警声音改变，同时该模块还有一个选择端口，根据 jingbao 模块根据距离选择的距离传来的距离选择信号选择相应的某一种声音的频率来输出。

Jingbao

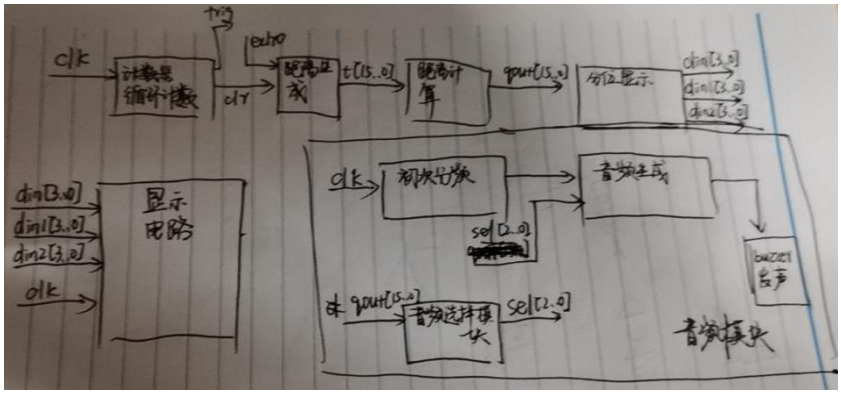
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity jingbao is
    port(a:in integer range 0 to 65527;
         q:out std_logic_vector(2 downto 0));
end jingbao;

architecture cc of jingbao is
begin
process(a)
    begin
        if(a>0) and (a<=20) then q<="100";
        elsif(a>20) and (a<=60) then q<="011";
        elsif(a>60) and (a<=100) then q<="010";
        elsif(a>100) and (a<=150) then q<="001";
        else q<="000";
        end if;
    end process;
end cc;
```

根据不同距离来输出不同的音频选择信号给 yinpin

voice

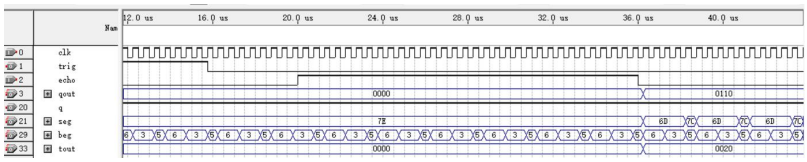
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity voice is
port(kz:in std_logic_vector(2 downto 0);
     clk:in std_logic;
     q:out std_logic:='1');
end voice;
architecture yl15 of voice is
signal count:std_logic_vector(2 downto 0):="000";
begin
process(kz,clk,count)
    begin
    if(clk'event and clk='1')then
        count<=count+1;
        if(kz="000"and count="100")then
            q<='0';
        elsif(kz="001"and count="011")then
            q<='0';
        elsif(kz="010"and count="010")then
            q<='0';
        elsif(kz="011"and count="001")then
            q<='0';
        elsif(kz="100"and count="000")then
            q<='0';
        end if;
    end if;
end process;
end yl15;
```

根据相应的距离产生的音频选择信号和输出的相应音频来进行最后的声音的输出。



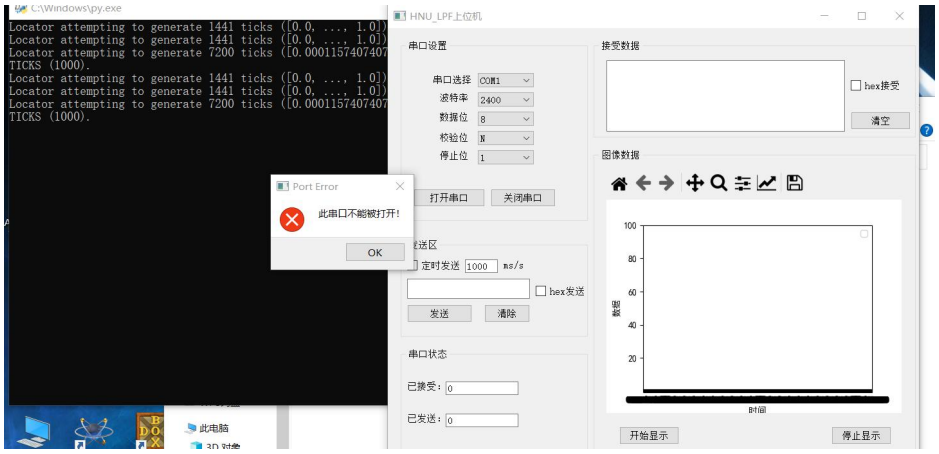
信号名	位宽	方向	释义
Clk	1	各初始需要时钟大模块	最开始的 24MHZ 频率的时钟信号给各大模块提供最开始的时钟源
Trig	1	传送给超声波测距器	由超声波模块产生，持续时间大于 10us 才能触发接下来的工作，满足条件后产生后续的 echo 信号进行距离的计算
Echo	1	传送给距离测算模块	echo 是超声波来回行走的持续时间，可根据 echo 来计算距离
Tout（t）	16	传送给 distance 模块	根据 echo 作为使能端的计数器计数次数，通过特定的计算公式来得到最后的距离结果
Qout	16	传送给 fenwei 模块，距离选择模块	最终的距离
Din 系列	4	传送给 Scan_led 进行数码管显示	将最终距离划分成可在数码管上表示的三位十进制数，分别代表个位十位百位
Sel[2..0]	3	传送给音频生成模块	根据距离产生相应范围内的距离选择信号传送给音频模块选择相应的音频
q	1	对最初的时钟不断分频，并根据距离选择合适占空比的频率	Buzzer 发声的最终时钟源

实验结果验证



echo 信号从 20us 开始计数，到 36us 停下，一共 16us，由于是每 0.5us 记一次数所以 tout 一共计数 32 次，转换成十六进制是 0010，然后 qout 是距离计算就是 16*340*10^-3/2 换算一下 cm 数就是 16*17,对应 16 进制就是 0110.实验结果验证正确。

综合任务 3：



通过查阅资料编写绘图代码 mplCanvasWrapper.py，界面控制代码 UI.py，储存模块公用变量代码 BL.py。
Main.py 控制数据接收和发送等逻辑，在 main 中进行串口的打开关闭还有清零，定时发送等逻辑操作。
最后实验结果：
不清楚为什么串口总是不能正常打开，可能是因为少下载 serial，也可能是其他的一些原因

```
C:\Users\86178>pip install pyserial
Requirement already satisfied: pyserial in d:\lib\site-packages (3.5)
```

查看网上资料，好像是说没有找到什么路径，由于对 python 的了解还是有限所以最终综合实验 3 并没有完成的很理想，串口没有打开导致后续的验证也难以展开。
总结
人文：
本次实验非常需要老师和同学的帮助，在同学的帮助下避免许多易错的问题，并且可以在别的同学的帮助下和同学达到同一起跑线一起克服新的难题，老师总能在许多关键点地方指导出我的错误，让我能够在很多时候突然想明白许多事，解决很多疑惑。但最终 s 级实验任务，由于知识能力的有限最终没有全部完成也感到可惜，不过在以后的学习生活中也给了我不断去挑战高难度问题的信心。
知识：
本次实验让我明白了超声波测距是如何实现，也明白蜂鸣器的报警声音也就是和音频频率有关，通过许多不同的组合，可以产生不同的音调，同时本次实验又用到了显示模块，也明白了通过不同的方式显示模块也能有不同的用法，比如转换成 3 位数码管的输出，同时也间接学会了 quartus 数据类型转换和 gernic 语法，也加深了对 proteus 软件的使用，对示波器也有了一定的了解，并在对 s 级任务的实验过程中了解更多 python 的知识和对自己电脑上 cmd 端口一些命令行的使用。
技能：
增强了自身对 quartus 的理解和使用，学会了新的语法，以及新的思路，更改时钟频率和占空比，也学会对示波器的查看，也更熟悉了对 proteus 软件的使用。最重要的是学会通过自己的努力去翻看大量网上资源，拼拼补补，修修改改组成自己的东西，以及通过网上的指导，学会下载相应的 module。