大综合实验报告

计科 1907　　杨杰　　201908010705

一、实验目标:

初步设计一个复杂数字系统-传感器数字采集系统，理解扩展接口的超声波测距和通用USB-TTL串口电路原理；实现其实作模块功能验证；分层次编写VHDL完成系统模块优化方案，验证平台，FPGA串口与PC上位机软件（可选），并验证调试。

二、实验资源:

QuartusII proteus8.8

计算机硬件技术基础实验教程

python（用于实现上位机界面的各个功能）

PyQt5、Qt designer（绘制界面）

Pycharm

三、实验任务

B 级任务（90%）

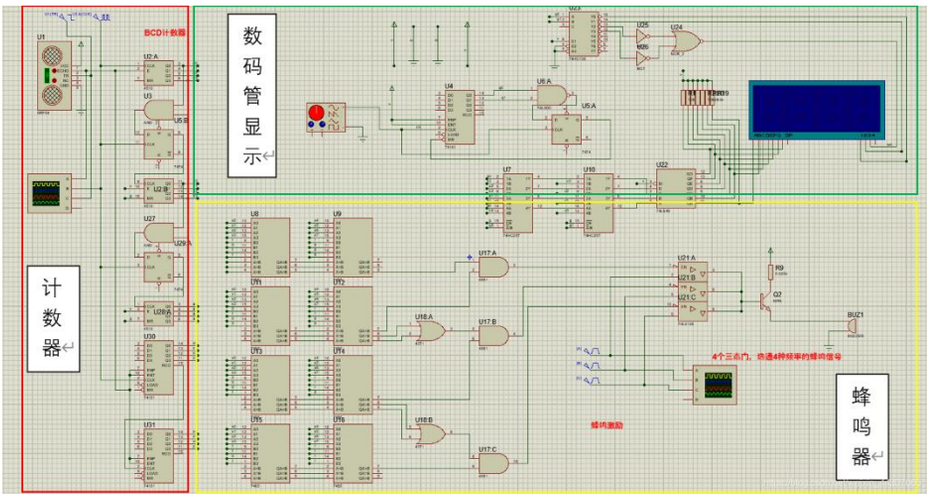
设计思路：

首先了解超声波测距传感器的工作原理。超声波测距传感器的工作原理，超声波发射器向一个方向发射超声波，在发射的同时计数器开始计时，超声波在空气中传播，途中碰到障碍物就立即反射回来，超声波接收器收到反射回的超声波就立即停止计时。计算出往返时间除以 2，乘上波速就是需要测量的距离。

本实验是在 proteus 中接超声波测距模块（SRF04）来完成，工作原理为超声波测距传感器模块有 4 个引脚，Vcc 接+5V，GND 接地，trig 为触发信号引脚，echo 为回响信号接收引脚。当 trig 引脚上产生一个大于 10μs 的高电平信号，模块开始工作，自动发送 8 个 40kHz 的方波，并检测是否有信号返回。当模块接收到返回信号，echo 引脚自动变为高电平。模块接收到信号返回，echo 引脚上由高电平自动变成低电平，则高电平持续时间为超声波传输时间，根据超声波在空气中传输的速度计算出实际的测量距离。

单次启动测距系统电路：

电路由三部分组成，第一部分是图中红色框，计数器部分，用于计数echo的高电平数。第二部分是图中绿色框，数码管显示部分，用于将BCD码动态的显示在数码管上；第三部分是蜂鸣器部分，根据采样值、发出多种不同频率组合声音。如下图：



①计数器部分：（见图中红色框）

计数器用于给echo计数，将echo作为计数器的使能端，时钟上升沿计数；计数器分为BCD采样部分和8位采样部分；BCD采样部分用4518模块相互级联，低位的Q0和Q3通过与门产生一个进位，存在一个D触发器中连接到高位的时钟端。级联产生个十百位的计数值，用于数码管显示；8位采样部分由两个**74161**计数器级联组成，产生8位采样值。

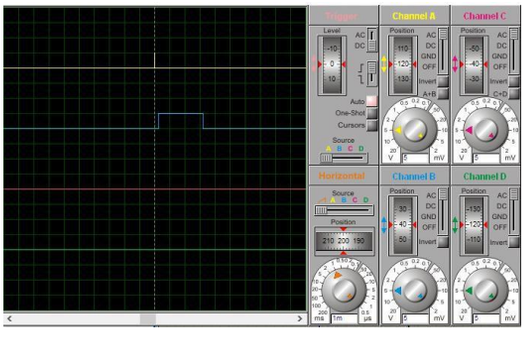
②数码管显示部分：（见绿色框图）

数码管显示部分，用于将BCD码动态的显示在数码管上；由一个计数器产生信号q0,q1,q2，作为两个二选一MUX的选择端选择一个BCD计数值，通过**7449**译码产生七段码；同时74HC138用于产生位选信号，选择一位采样值显示在数码管上。

③蜂鸣器部分（见黄色框图）

蜂鸣器部分用于根据不同的采样结果发出不同单音间隔的声音。分为选择器组和声源选通两部分。

验证：



结果分析：当 echo 检测到信号变为高电平，维持的时长设为 t，则距离 d=（t*340）/2。对于波形来说设置了单次的 trig 启动脉冲，然后产生了 echo 信号，符合预期。此时电路状态传感器显示 51，数码管显示 51,所以电路能够实现预期的功能。

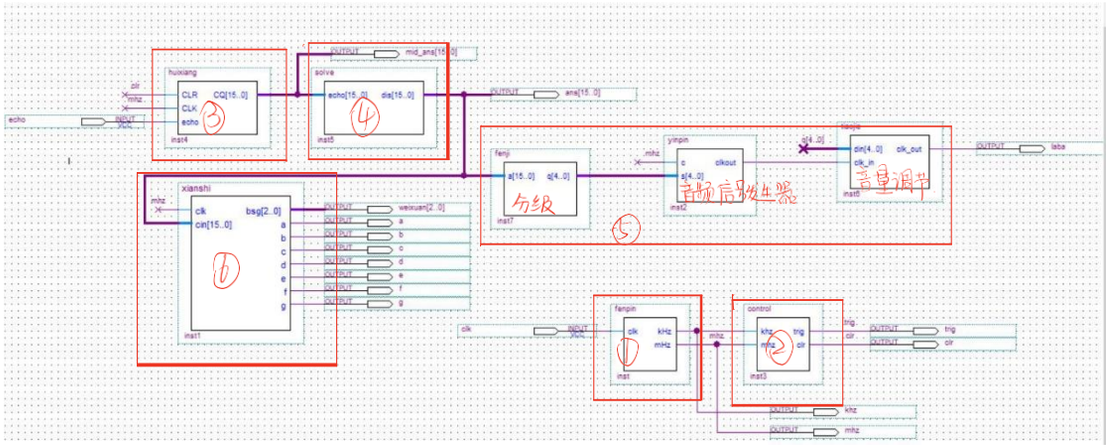
随着数码管中数字的变化即距离发生变化时， 蜂鸣器的声音也发生相应的变化。

A 级任务（100%）

设计思路：

输入一个 24Mhz 的时钟信号；输入的时钟经过分频产生 1kHz 和1MHz 的时钟输出，将分频之后的时钟作为控制模块的时钟输入，每 1 秒开始新一轮测量，在每秒最后输出计数清零信号，前 15us 输出一个 trig 激励信号，最终接收 echo 信号，根据比例测算出距离长度，由数码管和蜂鸣器根据测距长度做出不同输出。

顶层电路：



①分频模块：

信号名	位宽	方向	释义
clk	1	输入	24mHz 时钟输入
khz	1	输出	分频约产生1kHz时钟输出
mhz	1	输出	分频约产生1mHz时钟输出

```
entity fenpin is
port(
    clk:in std_logic;
    kHz:out std_logic;
    mHz:out std_logic
);
end entity;

architecture bhv of fenpin is
    signal cnt:std_logic_vector(15 downto 0):="0000000000000000";
    signal cnt2:std_logic_vector(7 downto 0):="00000000";
    signal tmhz,tkhz:std_logic:='0';
    begin
    process(clk)
    begin
        if(clk'event and clk='1')then
            cnt<=cnt+1;
            cnt2<=cnt2+1;
            if (cnt=12000) then
                tkHz<=not tkHz;
                cnt<="0000000000000000";
            end if;
            if (cnt2=12) then
                tmHz<=not tmHz;
                cnt2<="00000000";
            end if;
        end if;
    end process;
    khz<=tkhz;
    mhz<=tmhz;
end;
```

②控制模块：

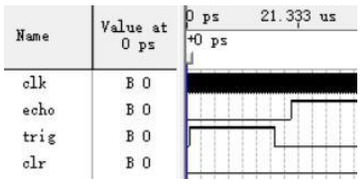
该模块要求计算 1s 的长度，在其中 15us 要求激励信号 trig 输出高电平，最后要求清零信号 clr 有效。

信号名	位宽	方向	释义
khz	1	输入	输入约1kHz的时钟
mhz	1	输入	输入约1mHz的时钟
trig	1	输出	1kHz计数1000次即每秒刷新一次，产生15μs高电平
clr	1	输出	每秒刷新一次

```
architecture bhv of control is
    signal con1:std_logic_vector(3 downto 0):="0000";
    signal con2:std_logic_vector(11 downto 0):="000000000000";
    signal ch1,ch2:std_logic:='0';
    type state_type1 is(send_trig1,send_techo1);
    type state_type2 is(send_trig2,send_techo2);
    signal state1:state_type1:=send_trig1;
    signal state2:state_type2:=send_trig2;
    begin
    process(mhz)
    begin
    if(mhz'event and mhz='1')then
        case state1 is
            when send_trig1=>
                trig<='1';
                con1<=con1+1;
                ch1<='0';
            if(con1="1111")then
                trig<='0';
                con1<="0000";
                state1<=send_techo1;
                ch1<='1';
            end if;
            when send_techo1=>
                if(ch2='1')then state1<=send_trig1;
            end if;
        end case;
    end if;
    end process;
```

```
    process(khz)
    begin
    if(khz'event and khz='1')then
        case state2 is
            when send_trig2=>
                if(ch1='1')then state2<=send_techo2;
            end if;
            when send_techo2=>
                con2<=con2+1;
                ch2<='0';
                if(con2="001111100111")then
                    con2<="000000000000";
                    clr<='1';
                elsif(con2="001111101000")then
                    clr<='0';
                    state2<=send_trig2;
                    ch2<='1';
            end if;
        end case;
    end if;
    end process;
end;
```

由波形仿真图可以验证前 15us 输出 trig 高电平，结果正确。



③回响计数器模块：

该模块需要用一个高频率的计数器，用 echo 信号作为计数器的使能端，计数器测量 echo 信号持续的时钟周期数，对每次测量结果准确清零。

为了使测得距离的范围增大，需要将时序设置为 1 mHz(分频以后)。

信号名	位宽	方向	释义
clr	1	输入	清零信号，当 clr=1，计数器清零
mhz	1	输入	1mHz 时钟作为回响器的输入
echo	1	输入	输入 echo 信号
CQ	16	输出	输出回响计数器的结果

```
entity huixiang is
  port(CLR,CLK,echo: in std_logic;
   CQ:out std_logic_vector(15 downto 0));
end huixiang;
architecture csp OF huixiang IS
  signal count1:std_logic_vector(15 downto 0):="0000000000000000";
 signal count2:std_logic_vector(15 downto 0);
begin
  process(CLR,CLK,echo)
   begin
  IF clr='1' then
      count1 <= (others => '0');
      elsif (count1 = "1111111111111111") then
          count1 <= (others => '0');
  elsif (clk'event and clk='1' ) then
      if(echo='1') then
          count1<=count1+1;
      else count2<=count1;
      end if;
  end if;
  end process;
  CQ <= count2;
end csp;
```

④蜂鸣器：

该模块要求根据不同距离调节蜂鸣器的输出频率以及声音间隔。首先通过音频信号发生器产生不同大 小的音频信号；然后根据分级部分将声音分为三个等级：1-32、32-64、64 以外三个等级；最后通过音量调节部分调整计数器频率与占空比。

⑤计算模块：

该模块要求将返回的 echo 信号转化成长度意义输出。根据输入的 echo 的技术结果记性计算，输入的技术器的频率为 1MHz，所以周期为 10^-6s。当收到的数据为 n 时，在 echo 为高电平时，有 n 个时钟信号上升沿。所以可以得出最后的测量距离为：d=n*10^-6*340/2（cm）。

信号名	位宽	方向	释义
echo	1	输入	回响器的计数结果
ans	1	输出	根据数学计算得出的距离

```
entity solve is
port(
 echo:in integer range 0 to 65527;
 dis:out integer range 0 to 65527);
 end entity;

 architecture rt of solve is
 signal a:integer range 0 to 65527;
 signal b:integer range 0 to 65527;

 begin
 process(a,b)
 begin
 a<=echo;
 if(a>0) then
 b<=(a*17)/1000;
 end if;
 dis<=b;

 end process;
 end rt;
```

分级部分：

信号名	位宽	方向	释义
a	5	输入	距离测量值
q	16	输出	特征值

```
entity fenji is
port( a:in std_logic_vector(15 downto 0);
      q:out std_logic_vector(4 downto 0));
end fenji;
architecture behav of fenji IS
begin
    process(a)
    begin
        if(a>="0000000000000000" and a<"0000000000100000") then
            q<="00001";
        elsif (a>="0000000000100000" and a<"0000000001000000") then
            q<="00100";
        else
            q<="10000";
        end if;
    end process;
end behav;
```

音量调节部分：

信号名	位宽	方向	释义
din	5	输入	控制占位的大小
clk_in	1	输入	输入的音频信号
clk_out	1	输出	加入占空后的音频信号

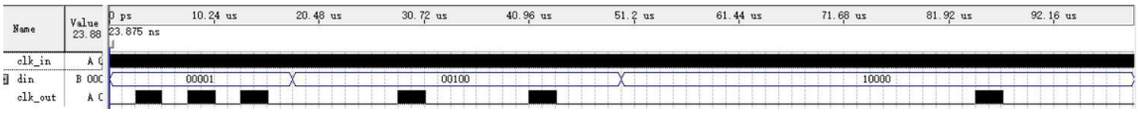
```
entity tiaojie is
port (
    din: in std_logic_vector (4 downto 0);
    clk_in :in std_logic;
    clk_out : out std_logic
);
end tiaojie ;

architecture bhav of tiaojie is
signal tmp:std_logic_vector(12 downto 0):=din&"00000000";
signal tmp1:std_logic_vector(12 downto 0):=din&"11111111";
signal count : std_logic_vector(12 downto 0):="0000000000000";
begin
    process(clk_in,count,din)
    begin
        if (clk_in'event and clk_in='1') then
            if(count<tmp1) then
                count<=count+1;
            else
                count<="0000000000000";
            end if;
        end if;
    end process;

    process(din,clk_in,count)
    begin
        if (count<tmp) then
            clk_out<='0';
        elsif((count>tmp)and(count<tmp1))then
            clk_out<=clk_in;
        end if;
    end process;
end bhav;
```

音频信号发生器：

信号名	位宽	方向	释义
c	1	输入	时钟输入
s	5	输入	音频控制信号
clkout	1	输出	根据控制信号的不同产生不同的 clkout 信号

```
entity yinpin is
    port( c:in std_logic;
          s:in std_logic_vector(4 downto 0);
          clkout:out std_logic);
end yinpin;
architecture art of yinpin is
signal d:integer range 0 to 10000;
signal count:integer range 0 to 10000;
signal temp:std_logic_vector(13 downto 0);
signal Q:std_logic;
begin
    bianma:process(s)
    begin
        case s is
            when "00001"=>temp<="00000010100011";
            when "00100"=>temp<="00000010100011";
            when "10000"=>temp<="00000010100011";
            when others=>temp<="00000010100011";
        end case;
    end process;
count<=conv_integer(temp);
    yin:process(c)
    begin
        if (c'event and c = '1') then
            if(d=count) then
                d<=0;
                Q<=not Q;
            else
                d<=d+1;
            end if;
            clkout<=Q;
    end process;
end art;
```

结果分析：



由波形仿真结果可看出，din 为00001 时，输出音频的占空比为 1/2；din 为0010 时，输出音频的占空比为 4/5；din 为10000 时，输出音频的占空比为 16/17，成功。

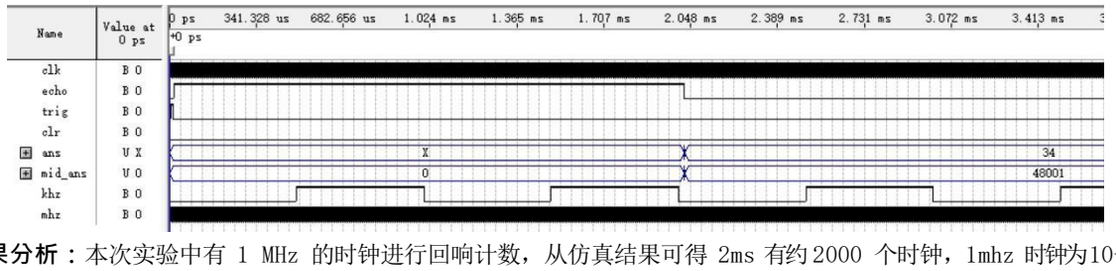
⑥数码管显示模块：（数码显示管在之前的实验中已经详细描述，所以在此不做赘述）

信号名	位宽	方向	释义
cin	16	输入	输入计算模块计算得到的距离
bsg	3	输出	输出译码后的位选信号
a-f	1	输出	输出译码后的段选信号

【仿真】

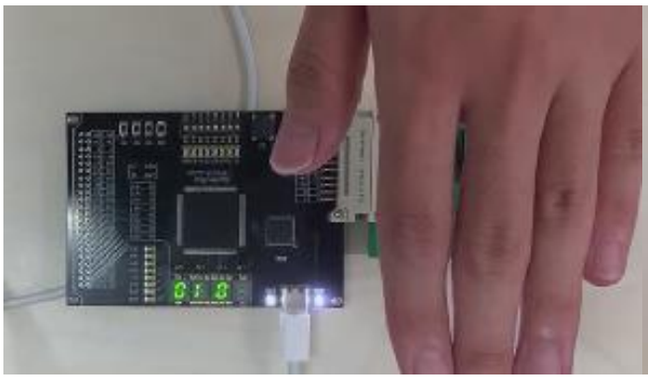
参数设置：clk：24mHz echo:2ms

仿真结果：



结果分析：本次实验中有 1 MHz 的时钟进行回响计数，从仿真结果可得 2ms 有约2000 个时钟，1mhz 时钟为10-6s，用时 2000*1*10-6 s，所以距离为 2000*10-6*340*100/2cm=34cm，最终测量距离为 34cm， 结果正确。

下载实验板验证：



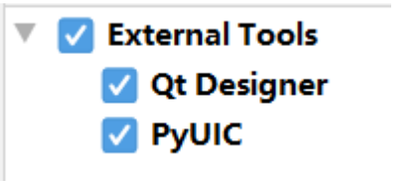
S 级任务1（110%）

1、 实验环境搭建

①主要安装 pycharm，PyQt5，其它还有很多小细节不再赘述；

②将安装好的 Qt designer、QtUic 添加到 pycharm 的外部工具中，以便后面使用；

最终安装好的界面如下：此时可以通过 pycharm 直接打开 desiger 界面以及将绘制好的 ui 文件转为 py 文件；



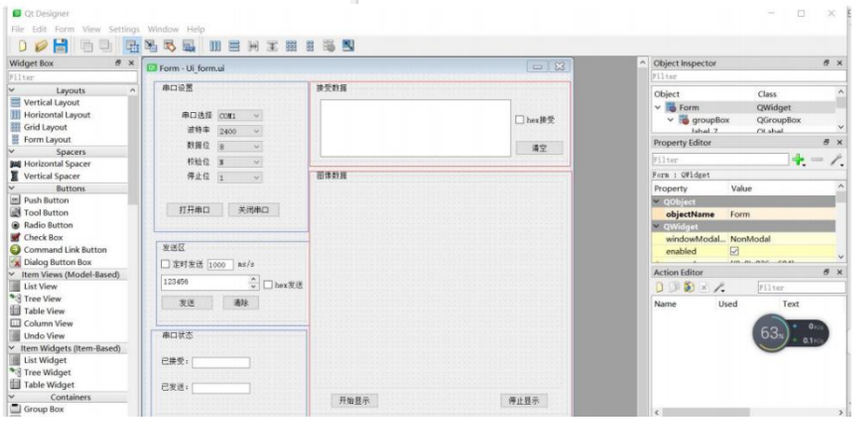
2、 ui 界面的绘制

界面如下图所示，这是最终版，中间修改过好多次，包括把图像显示板块里的初始化功能给删除了（实在没有能力实现）

由五个板块组成，包括串口通信的设置、发送、接收、串口状态以及图像处理；

除了处理部分，前面的通信部分基本上与教学资源的串口助手功能一致；

数据收发都支持 16 进制；



3、 直接使用 Qtuic 来将上面制作的 ui 界面转化为 py 文件

这一部分代码是自动转化得到的，我们只需要使用qt绘制界面。

4、 串口通信功能的实现

这一部分参照了很多代码，最终实现了以下的串口通信部分；主要依靠使用 pyserial 进行串口传输，在 secendstep.py(新建)中补充事件处理功能。

串行口的属性：

name:设备名字	portstr:已废弃，用 name 代替
port：读或者写端口	baudrate：波特率
bytesize：字节大小	parity：校验位
stopbits：停止位	timeout：读超时设置

1） 初始化

2） 按钮功能函数链接

直接使用 PyQt5 库中的 QtWidgets 类来实现将界面上的按钮与相应功能所连接。

```
from PyQt5 import QtCore, QtGui, QtWidgets
from mplCanvasWrapper import MplCanvasWrapper

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(826, 684)
        self.groupBox_2 = QtWidgets.QGroupBox(Form)
        self.groupBox_2.setGeometry(QtCore.QRect(20, 20, 291, 281))
        self.groupBox_2.setObjectName("groupBox_2")
        self.open_button = QtWidgets.QPushButton(self.groupBox_2)
        self.open_button.setGeometry(QtCore.QRect(20, 230, 111, 28))
        self.open_button.setObjectName("open_button")
        self.close_button = QtWidgets.QPushButton(self.groupBox_2)
        self.close_button.setGeometry(QtCore.QRect(140, 230, 101, 28))
        self.close_button.setObjectName("close_button")
        self.label = QtWidgets.QLabel(self.groupBox_2)
        self.label.setGeometry(QtCore.QRect(50, 50, 61, 20))
        self.label.setObjectName("label")
        self.comboBox_2 = QtWidgets.QComboBox(self.groupBox_2)
        self.comboBox_2.setGeometry(QtCore.QRect(120, 80, 87, 22))
        self.comboBox_2.setObjectName("comboBox_2")
```

3） 功能函数实现（以打开串口、接受数据为例）

【打开串口】

利用第三方库 PyQt5 中的类和其中的函数实现串口选择、波特率设置、字节数设置、停止位（只有 1）、校验位（只有 N 也就是不校验）设置；尝试开启串口，开启失败则反馈信息，成功则打开接收数据的定时器，设置按钮的状态更新串口状态模块的 groupBox 的标题信息。

【接受数据】

首先用 inWaiting 函数等待数据，如果无法接受则关闭串口（发生错误），否则根据收到的字节数将数据用read函数读出到字符串 data 中；如果选择 hex 显示，则需要进行进制的转换，否则转化为unicode字符串传给insertPlainText函数，在窗口中显示。

注：其余也都是类似的利用第三方库来实现各个功能，详见工程文件。

【打开串口】 【接受数据】

```python
# 打开串口
def port_open(self):
    self.ser.port = self.comboBox.currentText()
    self.ser.baudrate = int(self.comboBox_2.currentText())
    self.ser.bytesize = int(self.comboBox_3.currentText())
    self.ser.stopbits = int(self.comboBox_5.currentText())
    self.ser.parity = self.comboBox_4.currentText()

    try:
        self.ser.open()
    except:
        QMessageBox.critical(self, "Port Error", "此串口不能被打开！")
        return None

    # 打开串口接收定时器，周期为2ms
    self.timer.start(2)

    if self.ser.isOpen():
        self.open_button.setEnabled(False)
        self.close_button.setEnabled(True)
        self.groupBox.setTitle("串口状态（已开启）")
```

```python
# 接收数据
def data_receive(self):
    try:
        BL.num = self.ser.inWaiting()
    except:
        self.port_close()
        return None
    if BL.num > 0:
        BL.data = self.ser.read(BL.num)
        BL.leng = len(BL.data)
        # hex显示
        if self.res_c.checkState():
            out_s = ''
            for i in range(0, len(BL.data)):
                out_s = out_s + '{:02X}'.format(BL.data[i]) + ' '
            self.rec_txt.insertPlainText(out_s)
        else:
            # 串口接收到的字符串为b,'123'要转化成unicode字符串才能输出到窗口中去
            self.rec_txt.insertPlainText(BL.data.decode('utf-8'))

        # 统计接收字符的数量
        self.data_num_received += BL.leng
        self.lineEdit_2.setText(str(self.data_num_received))
```

4） 串口通信功能验证

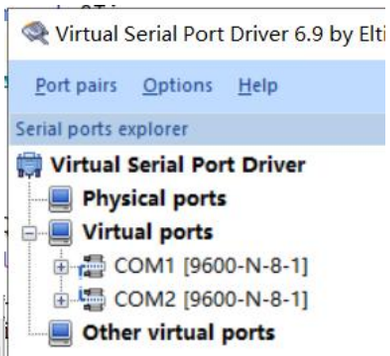
①首先利用虚拟串口打开虚拟串口通道

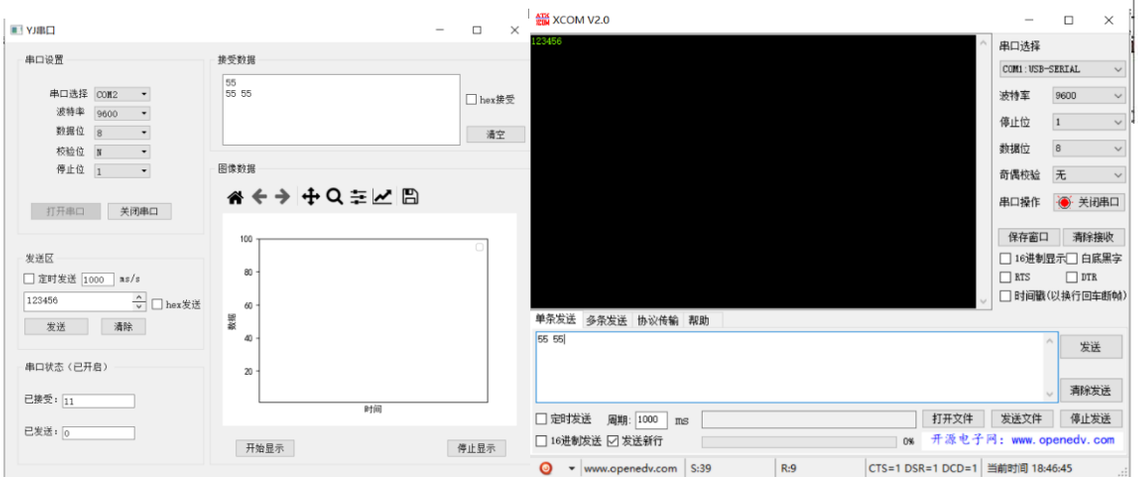
②将串口助手与编写的上位机软件设置好，波特率为 9600，数据位为 8，打开两个串口；

此时虚拟串口也显示 COM1、COM2 连接成功；

③验证发送与接受两方面功能（包括 hex 进制）：

串口助手发送 123456、我的上位机用 hex 发送 55：串口助手 hex’发送 55、我的上位机发送 123456。





接受、发送功能均正常；经过验证，清除数据等小功能均正常实现。

5、 数据图像显示

这一部分网上也很没有直接讲如何讲串口接受的数据转化成图像的，我是先学习了如何根据一组数据绘制图像以及如何将图像内嵌到上位机界面中之后自己编写的代码；

1） 在上位机界面中加入用于图像显示的窗体 Widget

为了嵌入 Matplotlib 在 mplCanvas 中，需要将 mplCanvas 升级，执行 Promote，输入类名 称 为 MplCanvasWrapper ， 这 个 类 就 是 编 写 matplotlib 代 码 的 ， 文 件 名 称 为mplCanvasWrapper。

2） 新建 py 文件（mplCanvasWrapper.py）并在其中编写图像处理的类和相关功能函数,为窗体添加事件处理

①主要依靠 matplotlib 库中的 figure 类来实现画图；

```python
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5 import NavigationToolbar2QT as NavigationToolbar
```

②令 x 轴显示时间并动起来

X 轴显示时间，matplotlib 提供了 plot_date 方法；

设计一个线程，用于产生数据和绘图，根据功能单一原则，需要将产生数据和绘图分成两类来实现，一个数据处理类，一个画板类。

③在 MplCanvas 类中使用 FigureCanvas 和 Figure 类创建 plot 绘制窗口（画板）

```python
class MplCanvas(FigureCanvas):

    def __init__(self):

        self.fig = Figure()
        self.ax = self.fig.add_subplot(111)

        FigureCanvas.__init__(self, self.fig)
```

这里值得注意的是上面的两句设置画板语句，令画板的横坐标在前 1 分钟每 10s 显示一次时间，后面变成每 1 分钟显示一次时间；

```
self.ax.xaxis.set_major_locator(MinuteLocator())  # every minute is a major locator
self.ax.xaxis.set_minor_locator(SecondLocator([10, 20, 30, 40, 50]))  # every 10 second is a minor locator
```

时间的显示格式设置为时分秒；

④在 MplCanvasWrapper 类（数据处理）中写绘图的逻辑，界面和实现完全分离

```
class MplCanvasWrapper(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.canvas = MplCanvas()
        self.vbl = QtWidgets.QVBoxLayout()
        self.ntb = NavigationToolbar(self.canvas, parent)
```

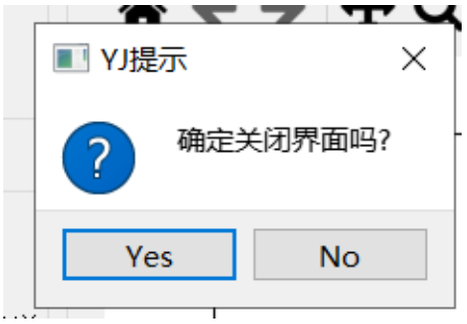
```
def startPlot(self):...

def pausePlot(self):...

def initDataGenerator(self):...

def releasePlot(self):...

def generateData(self):...
```

五个函数分别实现开启画图（startPlot）、停止画图（pausePlot）、开始画图前的准备（initDataGenerator）、退出时的清空（realease）、以及接受数据添加画图（generateData）；

其中前面四个函数的实现较为简单，只需要改变类_generating 的状态就可以；

在 secengstep.py 的 Pyqt5_Serial 类中可以通过引用他们来实现图形绘制的控制：

closeEvent 是我写的一个关闭窗口时的提示信息，关闭窗口时先提示是否关闭，如果选择 yes则需要通过调用 realeasePlot 来清空图像，然后直接关闭界面，效果如下：




```
def generateData(self):
    counter = 0
    while (True):
        if self.__exit:
            break
        if self.__generating:
            newTime = date2num(datetime.now())
            self.dataX.append(newTime)
            self.dataY.append(int.from_bytes(BL.data,byteorder='little'))
            self.canvas.plot(self.dataX, self.dataY)
            if counter >= MAXCOUNTER:
                self.dataX.pop(0)
                self.dataY.pop(0)
            else :
                counter += 1
        time.sleep(1)
```

⑤接受数据并绘制到图像上（接受数据添加画图（generateData））

这是整个图像部分里面最难实现的，我试了很多种方法，一直无法实现接受一个数据就实时更新图像；

退而求其次，我选择让他每隔 1s 就将接受的数据与相应的时间添加到用于绘图的数组中，进而实现绘图，而如果一直没有新的数据，那么每秒将保持将旧的数据画出来。

self.dataY.append(int.from_bytes(BL.data,byteorder='little'))

里面的这一句表示将接受的数据从 byte 转化为 int，也就是说串口助手需要 hex 发送才可以正确显示；

而接受数据的类我是写在了 secendstep.py 文件中的，如何让 mplCanvasWrapper.py 文件中的类来使用它呢，很简单直接开一个新的 py 文件用于保存这些量，让其他文件直接 import这个文件即可，里面的数据将变成公用的（上右图）；

到这里所有功能就全部实现了，接下来验证图像显示。

3） 图像显示验证

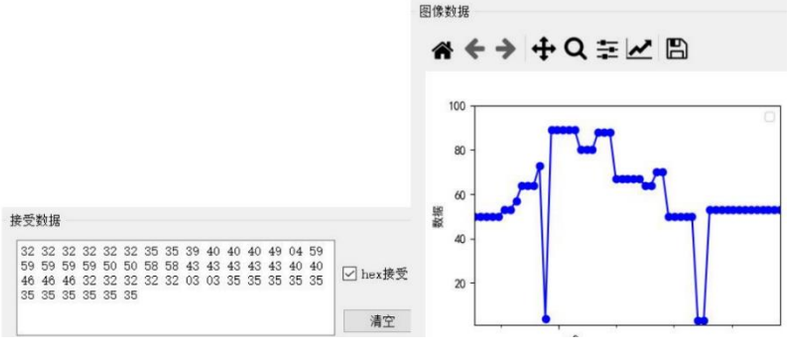
①打开串口助手与写好的上位机软件，令串口助手 hex 的形式一个一个数据发送；

②在上位机软件中选择 hex 接受并且开始绘图；

②在发送的过程中更改串口助手发送的数据，观察图形的显示；

图像是一个动态过程，这里不方便显示出它的过程，详见工程文件的上位机软件成品。

可以看出，图像是根据接受到的数据来进行显示的，验证正确。



数据的发送通过串口助手完成，每次仅能发送一个不大于 100（10 进制）数据，而不能一次性发送多个数据，这样的话我的程序会将几个数据识别为一个很大的数据，超过图像显示的范围 100，无法看出轨迹，数据的改变只能通过手动更改，这里还有待改进。

四、 利用 pyinstaller 将 py 文件打包生成 exe 文件

直接在 pycharm 的终端 Terminal 中输入指令：pyinstaller -i tubiao.ico -w -F secendstep.py Ui_form.py mplCanvasWrapper.py BL.py 即可；.ico 文件图标作为生成的 exe 的图标，最终在 dist 目录下生成了 exe 文件。

经过验证，exe 文件的功能与 py 文件的相同。

四、总结

人文方面：这学期的实验算是逐层加深难度并且层层相扣，从最开始的三位数码管到串口通信、FIFO存储，再到最后的这个传感器数字采集系统实验，为我们展开了非常丰富的数字世界，虽然并不是非常深入的学习，但也了解了很多，可以说是为后面的课程打下了基础。本次实验对于我来说整体有一定的难度，完成过程中遇到了很多问题，最后通过自己查阅资料、和同学讨论，最终完成了整个实验。

知识方面：本次实验我学会的最重要的知识点就是参数化分频器的设计。本次实验的A级任务顶层电路需要实现的模块较多，需要编写较多的代码，相对来说耗时较多且有一定的难度。同时通过不同层级的实验逐步掌握了倒车雷达的基本结构，学会了如何对工程的整体时序进行分配。此外，这一个学期学习了很多新的知识，包括数码管显示原理、串口通信原理、FIFO存储原理等等。这些知识有的很晦涩难懂，但是真正学懂后又会觉得十分的精巧。教学组提供的课后思考题和相关资源十分全面，帮助我们更好的理解和消化这些知识。

技能方面：这一学期首先掌握了proteus的使用方法，并且主要加深了quartus的应用技巧，三位数码管里的修正输出的电路设计和实验最后的分频电路都非常有趣并且十分有用，我相信这些技能将来即使用不上也能开拓思路。

数字系统实验圆满结束，感谢教学组精心准备的scrom资源以及每周的任务说明，一个学期让我学到了很多新的东西。