上海交通大學

生物医学信号处理

综合实验项目一报告

小组成员姓名：戴其铮　　学号:515021910253

小组成员姓名：刘睿豪　　学号:515021910266

一、程序开发逻辑

以 DAQNavi 下的 AI_InstantAI 工程文件为基础进行函数改写及功能拓展。

1）理解分析 AI_InstantAI 实例的代码原理；

2）在实例基础上扩展设计 ai_instant.ui 交互界面，在 ai_instant.cpp 增加中编写新函数并在头文件中声明相关参数与函数；

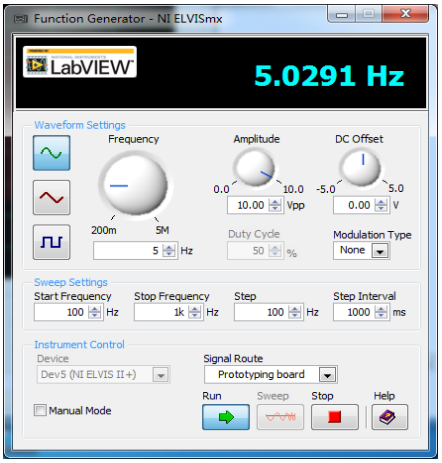
3）设计 fft_dialog.ui 交互界面专用于 FFT 相关功能展示，编写相应代码文件与头文件以实现所需功能；

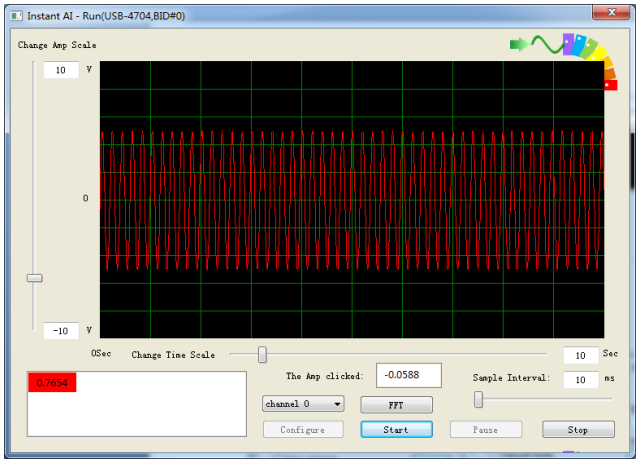
4）发布可执行文件，并在第三方 PC 上进行功能展示。

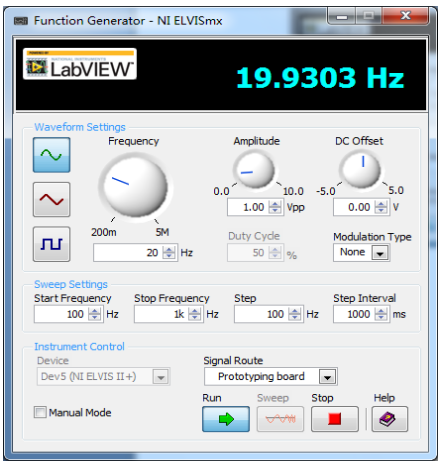
二、程序各功能实现与测试

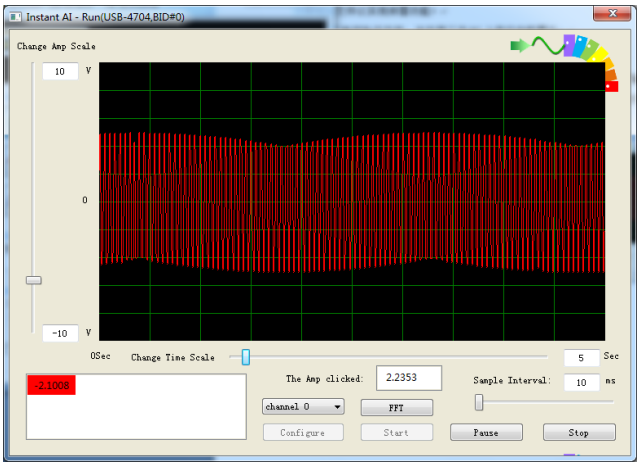
2.1、通过 USB-4704 的模拟输入端采集信号， 并实时显示至用户界面上

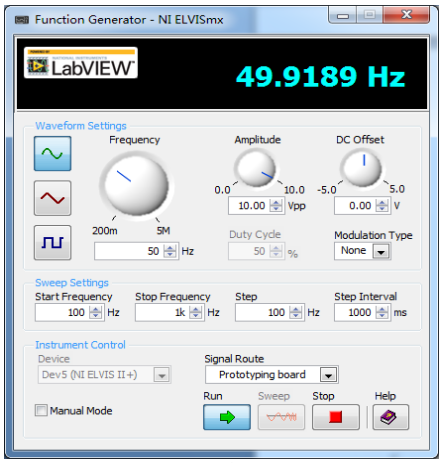
通过 USB-4704 的模拟输入端和 DAQ Navi 采集所产生信号，与产生波形相比，存在一定程度的失真。如下图所示：

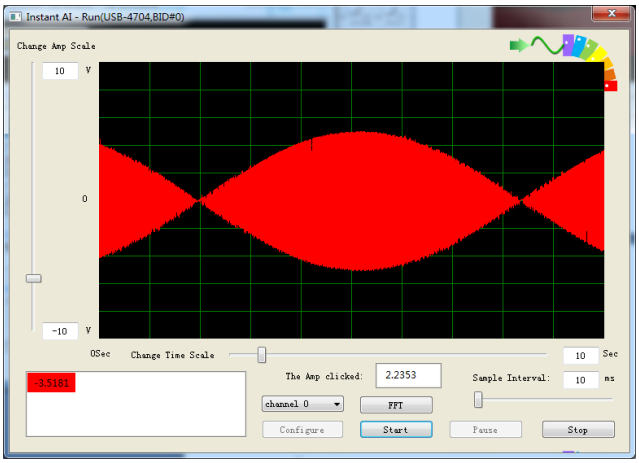












当采样频率远高于产生波形的最高频率成分时，采集波形与产生波形较为接近，当采样频率在产生波形的最高频率的 3-5 倍以内时可观察到一定程度的差异；当采样频率在产生波形的最高频率的 2 倍以内时波形已明显失真。

对于不同频率的信号，需设置采样频率在其最高频成分的 5 倍以上可较好地保证信号的完整度，考虑因素包括但不限于输入信号的频率成分、硬件输入通道的工作频率、程序的执行时间。

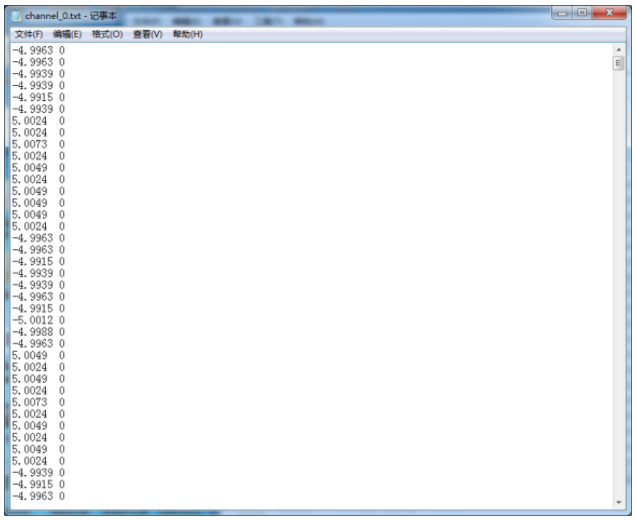
2.2、文件存储模块

1）在点击 Start 按钮后，自动创建并打开相应通道的 txt 文件，若原来文件已经存在，则清空

2）在点击 Pause 按钮时，停止读入数据，写入 txt 数据操作也相应暂停。

3）在点击 Stop 按钮后，关闭相应文档

成果如下图所示：



2.3、信号处理模块：

在本部分中，我们将介绍 FFT 信号处理原理与我们所采用的实现算法，基于

FFT 信号处理的信号处理模块构建思路，以及成果展示。

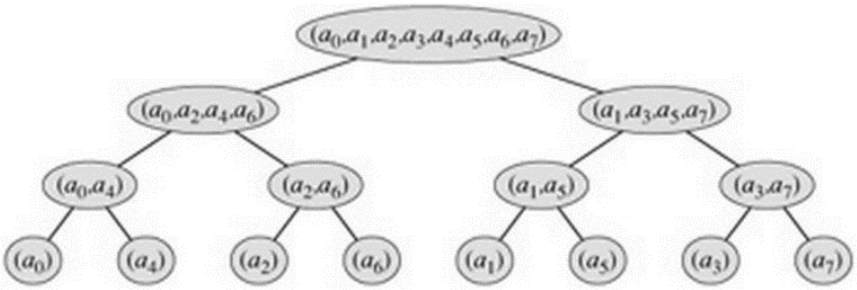
2.3.1、FFT 信号处理原理与实现算法：

快速傅立叶变换（Fast Fourier Transform，FFT）是离散傅立叶变换（Discrete Fourier transform，DFT）的快速算法，它是根据离散傅立叶变换的奇、偶、虚、实等特性，对离散傅立叶变换的算法进行改进获得的。它对傅立叶变换的理论并没有新的发现，但是对于在计算机系统或者说数字系统中应用离散傅立叶变换，可以说是进了一大步。

设 Xn 为 N 项的复数序列，由 DFT 变换，任一 Xi 的计算都需要 N 次复数乘法和 N -1 次复数加法，而一次复数乘法等于四次实数乘法和两次实数加法，一次复数加法等于两次实数加法，即使把一次复数乘法和一次复数加法定义成一次“运算”（四次实数乘法和四次实数加法），那么求出 N 项复数序列的 Xi ，即 N 点 DFT 变换大约就需要 N2 次运算。当 N =1024 点甚至更多的时候，需要 N2 = 1048576 次运算，在 FFT 中，利用 ωn 的周期性和对称性，把一个 N 项序列（设 N 为偶数），分为两个 N / 2 项的子序列，每个 N / 2 点 DFT 变换需要 (N / 2)2 次运算，再用 N 次运算把两个 N / 2 点的 DFT 变换组合成一个 N 点的 DFT 变换。这样变换以后，总的运算次数就变成 N + 2 * (N / 2)2 = N + N2 / 2。继续上面的例子，N =1024 时，总的运算次数就变成了 525312 次，节省了大约 50% 的运算量。而如果我们将这种“一分为二”的思想不断进行下去，直到分成两两一组的 DFT 运算单元，那么 N 点的 DFT 变换就只需要 N * log2N 次的运算，N = 1024 点时，运算量仅有 10240 次，是先前的直接算法的 1% ，点数越多，运算量的节约就越大，这就是 FFT 的优越性。

FFT 的实现可以自顶而下，采用递归，但是对于硬件实现成本高，对于软件实现都不够高效，改用迭代较好，自底而上地解决问题。感觉和归并排序的迭代版很类似，不过先要采用“位反转置换”的方法把 Xi 放到合适的位置，设 i 和 j 互为 s = log2N 位二进制的回文数，假设 s = 3, i = (110)2 = 6, j = (011)2 = 3, 如果 i≠ j ，那么 Xi 和 Xj 应该互换位置。（关于这个回文数的生成，是很有趣而且是很基本的操作，想当初偶初学 C++ 的时候就有这样的习题。）当“位反转置换”完成后，先将每一个 Xi 看作是独立的多项式，然后两个两个地将它们合并成一个多项式（每个多项式有 2 项），合并实际上是“蝶形运算”（Butterfly Operation, 参考《算法导论》吧^_^），继续合并（第二次

的每个多项式有 4 项），直到只剩下一个多项式（有 N 项），这样，合并的层数就是 log2N ，每层都有 N 次操作，所以总共有 N * log2N 次操作。迭代过程如下图所示，自底而上。



2.3.2、信号处理模块构建思路：

我们在上述算法及修改已有 C++代码的基础上，将 FFT 构造为了我们信号处理模块的基本操作函数。

整体信号处理模块的基本操作流程如下：

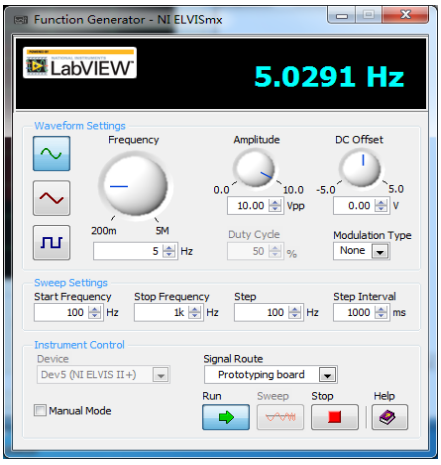
1）读取已经写入信号的原始文档

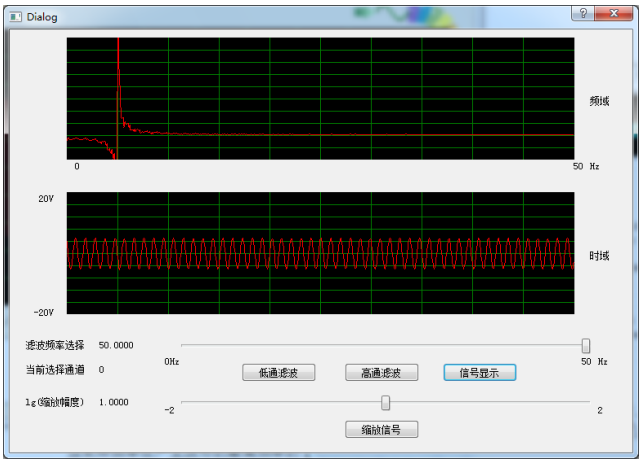
2）对信号进行傅里叶变换，将频域幅值结果与原始信号进行显示

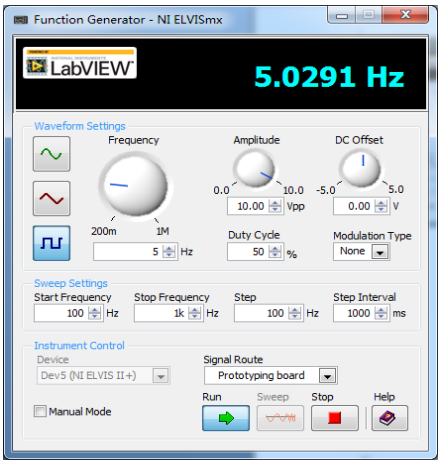
3）用户通过操作界面，选取信号放大倍数与滤波频率

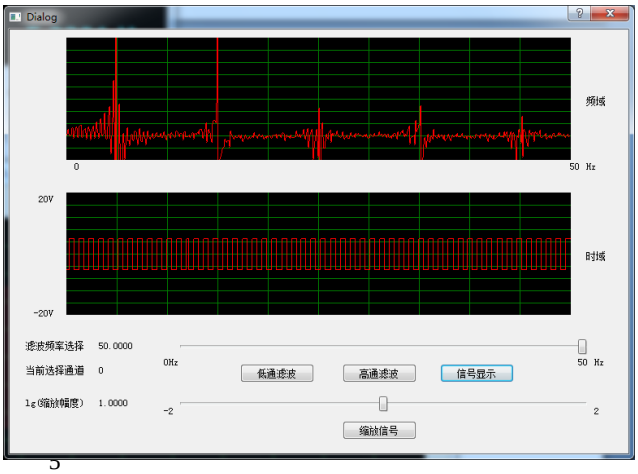
4）获取用户选取数值，对信号进行放大，高通低通滤波等相应操作，并进行显示

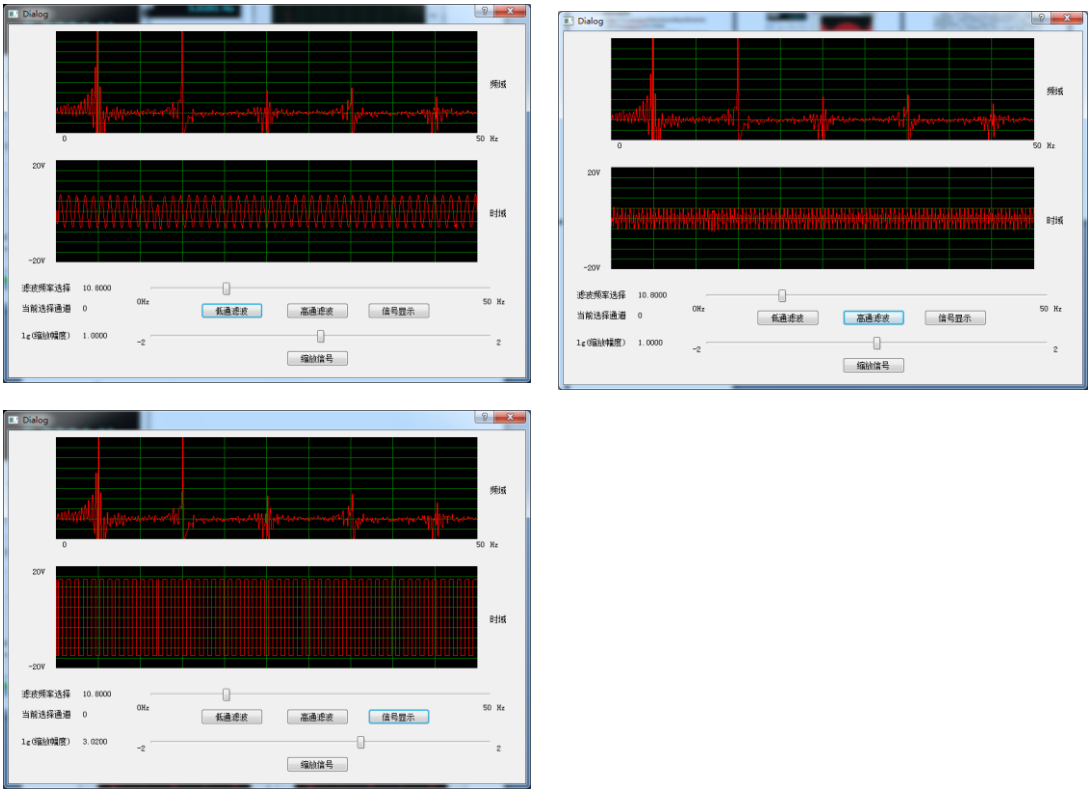
成果展示如下图所示：











2.4、设置不同采样率，停止和继续采集信号

通过调节滑轨 sldTimerValue 的值，将触发 SliderValueChanged(int value)函数，将改变后的 sldTimerValue 滑轨值赋给定时器的时间间隔以改变采样率并在 edtTimeValue 文本框内显示。当按下 btnPause 按钮或 btnStop 按钮时，定时器将被停止，以此停止采集信号；当按下 btnStart 按钮时，定时器将重新启动，以此继续采集信号。

2.5、对采样信号进行时间轴和电压轴的缩放

通过调节滑轨 sldXscale 的值，将触发 SliderXScaleChanged(int value)函数，将改变后的 sldXscale 滑轨值进行单位换算后赋给 m_xCordTimeDiv，更改曲线图显示的横轴间隔，以此实现时间轴的缩放，并在 edtTimeValue_scale 文本框内显示时间轴范围。

通过调节滑轨 sldYscale 的值，将触发 SliderYScaleChanged(int value)函数，将改变后的 sldYscale 滑轨值赋给，m_yCordRangeMax 作为曲线图的电压值上限，将其相反数赋给 m_yCordRangeMin 作为曲线图的电压值下限，以此实现电压轴的缩放，并在 edtAmp_scale_max 和 edtAmp_scale_min 文本框内显示电压轴范围。

2.6、用鼠标选择采样信号上某个数据点时可显示该点对应的电压值

当用户点击鼠标时，mousePressEvent(QMouseEvent *event)函数将被该事件触发执行，获取此时鼠标在显示屏上的全局坐标位置，而后将其转化为窗口下的坐标。如果鼠标落在曲线图区域，则通过和曲线图区域位置、曲线图电压轴尺寸、曲线图时间轴尺寸的对比，可将其坐标位置换算成曲线图上的坐标位置。此时对当前显示屏进行截图并将像素图转化为 QImage 类，获取鼠标所在位置像素的 Qcolor 颜色值，而后与各个输入通道的信号曲线 Qcolor 颜色相比。如果二者相符，则可认定用户成功点击曲线上的点，由此把该点的电压值以 QString 字符串的形式输出到 listWidget_click 文本框显示。

在测试中，由于单个像素点尺寸过小，用户难以准确点击曲线上的点，实用性较低，故我们以上下 2 个像素为容差，扩展了校准区域，由此在不损害功能的前提下改善了用户体验。

三、分析 USB-4704 模拟输入功能可采集信号的频率范围

USB-4704 模拟输入功能的采样频率为 10Hz-1000Hz。

若输入信号在该范围外，则可能导致欠采样的发生，出现波形的失真，高频成分的丢失，或者因相邻采样点落入不同周期而导致采集到的信号的频率的降低。若超出范围较小可通过拟合曲线并插值的方式弥补未能采集的数据点，若超出范围较大则只能更换硬件设备。