Report of Project 1

【Group Members】

孙诗语 516082910006
刘万山 516082910013

【Experimental Objectives】

1. **Understand the programming process of analog input signal acquisition and perform simple signal processing;**
2. **Implement various functions of graphical user interface (GUI);**
3. **Set an appropriate sampling rate for different input signals;**
4. **Analyze the possible problems of different frequency signals and propose solutions.**

【Experimental Apparatus】

1. **USB-4704；**
2. **DAQNavi Driver for USB-4704；**
3. **ELVIS II+;**
4. **Qt creator.**

【Experimental Procedures and Requirements】

Part 1. Use the signal generator of ELVIS II+ to generate a waveform whose frequency is lower than 100 Hz. Then use the analog input of USB-4704 and DAQNavi to collect signals and compare the waveform which is generated and collected, and record it.

Part 2. Select a language for programming using DAQ Navi SDK and write the user interface for display and user interaction. The program needs to realize the following functions:

- **Signal is collected through the analog input of USB-4704 and displayed on the user interface (real-time or quasi-real-time);**
- **The collected signal data could be saved to the hard disk in the file format (CSV or other) for the convenience of subsequent signal processing;**
- **The collected signals can be processed by FFT, and filter (low pass and high pass for all the signals);**
- **Different sampling rates can be set;**

- **It can start, stop and continue signal collection processing;**
- **It has the function of scaling the time axis and the voltage axis;**
- **It can realize multi-channel analog input at one time;**
- **When selecting a data point on the waveform with the mouse, the corresponding value of the point can be displayed.**

After the program coding, test and debug the program, record the problems encountered and how to deal with them; After passing the test, publish the executable file and display the function on another PC.

【Results and discussion】

Part 1. Select a language for programming using DAQ Navi SDK and write the user interface for display and user interaction.

2.1 The logic of program development

The program is mainly divided into four parts which respectively are "configuredialog.cpp" "ai_instant.cpp" "simplegraph.cpp" "data_processing.cpp" We will introduce the four modules respectively as following.

The first part is the "configuredialog.cpp" which is responsible for the hardware selection and some parameter settings.

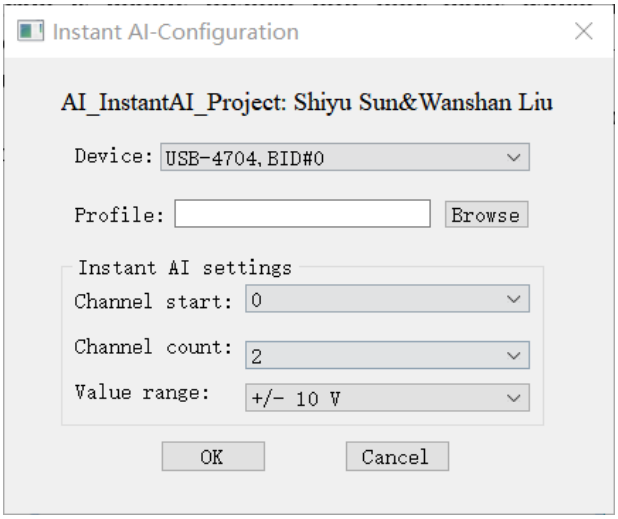

Figure. 1

Figure. 1 is the Graphical User Interface (GUI) of the "configuredialog.cpp" part. As we can see from the figure we must select a specific equipment and its profile. Besides, we need to confirm the number of consecutive channels we have connected and other information such as start channel/channel count/value range and so on. If we have confirmed all information of the configuration dialog and press the OK button, then we will enter into the main part of the program which is "ai_instant.cpp".

The second part which is also the most important part is the "ai_instant.cpp".

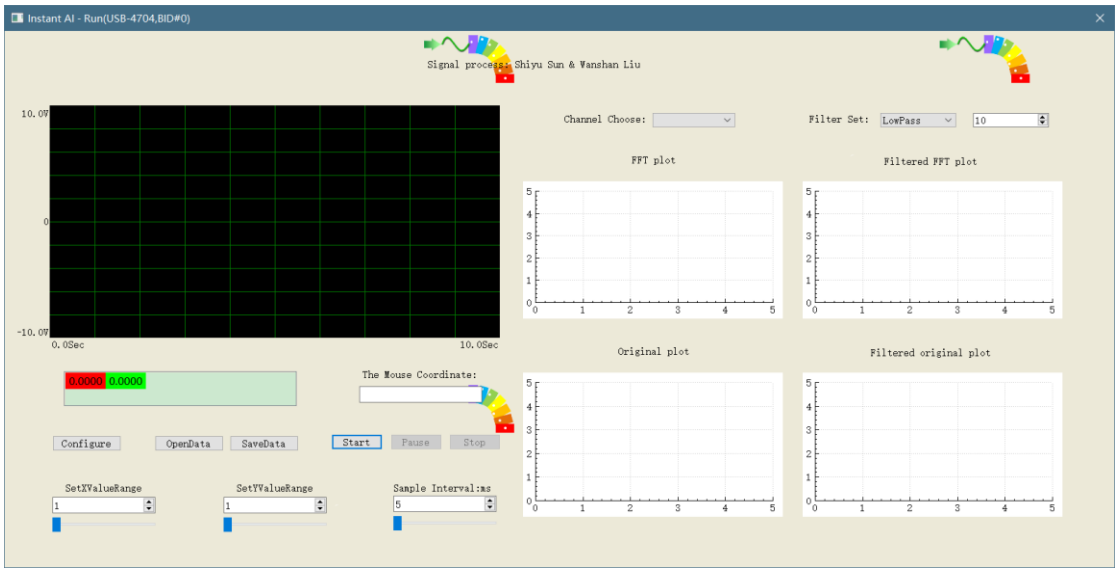

Figure. 2

The Figure. 2 above is the corresponding GUI interface. We will enter this interface after we confirm the information in the configuration dialog. In this part, we set a QTimer - Sample Interval (will be called clock below, can be changed in the "Sample interval" window) and it will load one data of every channel to the buffer which is scaledData [16] when a sample interval passed by with the following code.

```
instantAiCtrl->Read(configure.channelStart, configure.channelCount, scaledData);
```

Then we will save all data in m_drawDataBuffer and draw the waveform in the black window when every clock goes by with the following code. The above description is our idea of data collection and real-time display method. We will describe the drawing process in detail in the "simplegraph.cpp" section.

```
graph->Chart(scaledData, configure.channelCount, 1, 1.0 * dataCylic / 1000);
```

Besides, we can switch to the configure dialog by clicking "Configure" button and reselect the device or channels. We also can save all the data as CSV files through the "SaveData" button. Similarly, we can read a CSV file and display it on the black window. When we move the mouse to a point of the waveform, the coordinate of which will display in the text window. In addition, we can adjust the range of coordinate axes and control the state of data reading and waveform display. Last but not least, we can filter the signal and display the results in the black window.

Then, the third part (simplegraph.cpp) and fourth part (data_processing.cpp) are mainly to supplement the second part (ai_instant.cpp). The simplegraph.cpp part is mainly the code to draw waveform in the black window. We will call the "chart" function in the "ai_instant.cpp" part, we can see the main part of the implementation of chart function in the simplegraph.cpp from following code.

```
CalcDrawParams(m_xIncByTime);
SaveData(dataScaled, plotCount, m_dataCountPerPlot);
MapDataPoints();
update();
```

The "CalcDrawParams" function is mainly to calculate some parameters for drawing the waveform. And the "SaveData" function will save the data in "m_drawDataBuffer" with below code.

```
memcpy(m_drawDataBuffer + m_dataCountCachePerPlot * unsigned long long(plotCount), data,
    unsigned long long (plotCount) * dataCountPerPlot * sizeof(double));
m_dataCountCachePerPlot += dataCountPerPlot;
```

The "MapDataPoints" function is mainly to calculate the coordinate of corresponding signal data. The "update" function is to draw the waveform and the grid lines and termination lines.

The fourth part is mainly to complete some data processing. Which are composed with following functions.

```
void FFT();   /// FFT
void IFFT();   /// IFFT
void Filterlp();   /// Filter with low pass
void Filterhp();   /// Filter with high pass
void Magnify(int Multiple);   /// Magnify the signal
```

So we can calculate the FFT, IFFT, High-Pass-Filter, Low-Pass-Filter and Magnify of the original signals. Besides, we can set the filter type and the cut-off frequency. And show all the results including "FFT, FilteredFFT, Original, Filtered original" in the GUI.

【Analysis and discussion】

Part 1. Analysis of sampling rate

1.1 How to set the appropriate sampling rate for different frequency signals

According to Nyquist's sampling theorem, the condition for correct signal recovery is that the sampling rate is at least twice the maximum frequency of the signal. In practice, the sampling rate is usually selected to be larger than 2 times the highest frequency so that we can recovery the signal better.

1.2 Factors to consider when setting the sampling rate

- **The minimum sampling rate cannot make the input signal affected by noise.**
- **The maximum sampling rate must ensure that the algorithm responds quickly enough**
- **If the signal recovery is as good as possible, the sampling rate should be as high as possible.**
- **If the sampling rate is not high enough, the high frequency part will cover up the low frequency part and cause aliasing. This can be avoided by increasing the sampling rate.**
- **The sampling rate should not be too high, otherwise undersampling will occur.**

Part 2. The analog input function of USB-4704 can collect the frequency range of signals and deal with abnormal conditions

2.1 The analog input function of usb-4704 can collect the frequency range of signal：

When the frequency range is about 1Hz-20Hz, the analog input signal is good. When the frequency range is above 20Hz, The signal distortion is getting worse and worse.

2.2 Problems that may arise if the input signal is outside this range, and possible solutions

Problems: If the input signal exceeds this range, there may be undersampling and high frequency distortion.

Solutions: If the range is small, fitting curve and interpolation can be used to correct the uncollected data points. If it is larger than the scope, the acquisition equipment needs to be replaced.

Part 3. Problems and solutions

Problems: The link failed while generating the EXE file.

Solutions: There are different libstdc-6.dll files in QT, causing the computer not to know which one to use, so drag the file to the exported EXE folder to solve the problem.

Problems: We don't know how much data we need to collect before we collect the data. It is easy to cross the boundary with the traditional pointer method.

Solutions: Use STL-Vector in C++ to store data better.

Problems: There are a lot of unknown reasons for Qt development using VS, and a lot of time is wasted.

Solutions: Qt Designer is directly used for development with better compatibility.

Problems: In the process of real-time display, dynamic display cannot be achieved.

Solutions: Continuous refresh is adopted to achieve real-time dynamic display.