ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ВИЗУАЛИЗАЦИИ ПОТОКА ДАННЫХ, ПОСТУПАЮЩИХ В ПЭВМ ОТ МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ

Латий О. О.
Кафедра ЭВМ И Систем, Брестский государственный технический университет информатики
Брест, Республика Беларусь
E-mail: latijoo@tut.by

В докладе представлен вариант программного обеспечения для работы с микроконтроллерной системой для биометрической оценки состояния пользователя ПК [1], включающий авторский высокопроизводительный компонент библиотеки Qt для отрисовки 2D-графиков в реальном времени. Приводятся архитектура и особенности реализации, а также результаты сравнения с другими средствами построения графиков для платформы Qt.

Введение

В ряде задач, связанных с измерительными приборами, способными обмениваться данными с компьютером, возникает необходимость оперативной визуализации получаемых компьютером данных. Жесткие временные ограничения накладывают дополнительные условия на программные компоненты, выполняющие отрисовку графика по поступающим данным. При решении подобной задачи для системы биометрической оценки состояния пользователя [1], автором был разработан кросс-платформенный модуль для библиотеки Qt, выполняющий высококачественную визуализацию таких данных. Код проекта доступен по адресу `https://github.com/lattoo/plotter` под лицензией GPL v.2.

I. Архитектура системы

Структурная схема программного обеспечения, требующего визуализации данных, показана на рисунке 1.

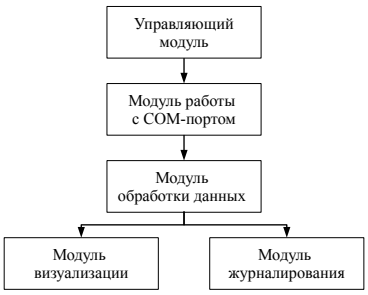

Рис. 1 – Структурная схема программного обеспечения

Эмпирически было получено минимальное значение задержки в 16 мс между принятием данных в режиме работы программного обеспечения в общем потоке. Данная величина является недопустимо большой, поэтому особенностью модуля работы с COM-портом (и его реализацией в виде USB-переходника, через который в настоящее время работают многие источники данных) является запуск подпрограммы получения и распределения данных в отдельном потоке в целях ускорения взаимодействия между программной частью ПЭВМ и устройством.

На принимающей стороне в зависимости от номера индекса формируются реальные значения, включая выполнение операций сдвига и объединения байт для получения слов. Таким образом удалось достичь контролируемой величины временной задержки в 1 мс.

Модуль обработки данных осуществляет цифровую фильтрацию данных, подсчет средних, минимальных, максимальных величин, отклонений от параметров, а также некоторый статистический анализ полученных результатов.

Модуль журналирования осуществляет запись необходимых данных в csv-файл. Операция записи осуществляется с интервалом 250 мс. Данный временной промежуток является компромиссным, предполагающим получение журналирования результатов с необходимой и достаточной частотой с одной стороны и получение конечного файла относительно небольшого объема, с другой.

Интерфейс можно увидеть на рисунке 2.

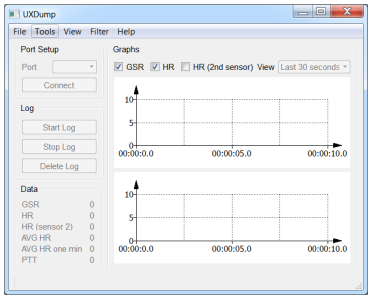

Рис. 2 – Интерфейс разрабатываемого ПО

Модуль визуализации отвечает за отрисовку данных в виде графиков/гистограмм в режиме реального времени. В качестве этого модуля был разработан и опубликован под свободной лицензией авторский построитель диаграмм, отвечающий как необходимым требованиям по времени, так и современным потребностям визуального оформления.

ПО написано на языке программирования C++ с использованием библиотеки Qt и в на-

стоящее время скомпилировано для двух платформ – GNU/Linux и MS Windows.

II. Обзор существующих средств построения графиков в Qt

При необходимости реализовать функционал построения графиков Qt-разработчик сталкивается с выбором из нескольких вариантов, причем это одна из немногочисленных задач, где на сегодняшний день все ещё сохраняется конкурентноспособность проприетарных Qt-компоенентов. Так, к актуальным построителям графиков для Qt можно отнести следующие:

- QCustomPlot [2];
- QChart [3];
- Qwt [4];
- ChartDirector [5].

Однако, ни один из пречисленных построителей не отвечают поставленным требованиям.

III. Особенности программной реализации отрисовщика

Причиной разработки собственного инструмента построения графиков для Qt является недостаточная производительность одних бесплатных фреймворков, которая делает их малопригодными для отрисовки динамично изменяющихся графиков в режиме реального времени, и сильно ограниченные визуальные возможности других. Для написании виджета отрисовки графиков выбор между языком C++ и языком разметки QML был сделан в пользу C++ в силу требований максимальной производительности кода.

Программная реализация компонента Plotter представляет собой многоуровневую архитектуру, что позволяет настроить построитель под требования пользователя.

В общем случае структура классов принимает вид, изображенный на рисунке 3.

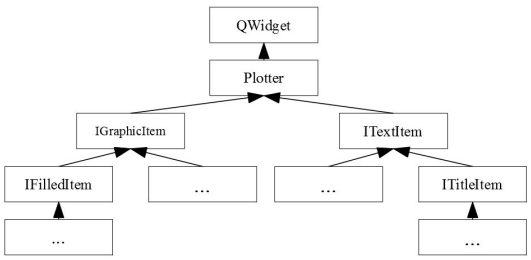

Рис. 3 – Структура классов отрисовщика

Как видно из данной диаграммы, основные интерфейсные базовые классы включают IGraphicItem, IFilledItem, ITextItem, ITitleItem.

Подклассы, унаследованные непосредственно от IGraphicItem соответствуют различным видам линий: для отрисовки главной и второстепенной сеток, для отрисовки нулевых линий, для отрисовки штрихов главной и вторстепенной сеток.

Подклассы интерфейса IFilledItem отвечают за отрисовку графических элементов с заливкой: внешняя область виджета, заливка области построения, заливка области под графиком.

Подклассы интерфейса ITextItem отвечают за отрисовку текстовых полей: подписи осей, названия осей, легенда, название графика.

Также стоит упомянуть, что благодаря многоуровневой архитектуре расчеты для отрисовки графических элементов отделены от непосредственной отрисовки, а в качестве дополнительных бонусов реализован экспорт графиков в формат масштабируемой векторной графики SVG, и опциональное хранение пакета настроек графика в конфигурационном файле.

На рисунке 4 приведен минимальный код для подключения, инициализации, добавления данных и отрисовки графика.

```
1 Plotter* plotter = new Plotter;
2 const QString chart_name = "Chart 1";
3 plotter->add_chart(chart_name);
4 const QPen pen = QPen(QColor(220,170,170), 1, Qt::SolidLine);
5 plotter->chart(chart_name)->set_pen(pen);
6 plotter->chart(chart_name)->add_data(x, y);
7 plotter->scroll_graph();
8 plotter->replot();
```

Рис. 4 – Минимальный код для работы с отрисовщиком

Строка 3 отвечает за добавление графика на область построения. Строка 5 отвечает за инициализацию пера отрисовки. Строка 6 отвечает за добавление данных. Строка 7 необходима для прокрутки области построения графика. Строка 8 инициирует общее обновление для области построения.

Пример работы построителя приведен на рисунке 5.

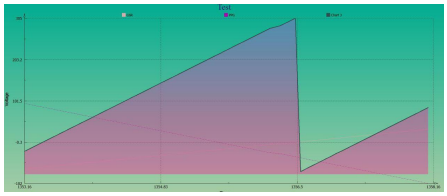

Рис. 5 – Пример работы построителя диаграмм

1. Латий, О. О. Микроконтроллерная система для биометрической оценки состояния пользователя ПК / Д. А. Костюк, О. О. Латий, В. П. Шамонин // Информационные технологии и системы 2017 (ИТС 2017) : материалы международной научной конференции, Минск, БГУИР, 25 октября 2017 г. – Минск, 2017. – С. 238–239.
2. QCustomPlot Qt Plotting Widget QCustomPlot - Introduction [Electronic resource] – Mode of access: www.qcustomplot.com/. – Date of access: 14.09.2018.
3. QChart Class [Electronic resource] – Mode of access: https://doc.qt.io/qt-5.10/qchart.htm. – Date of access: 14.09.2018.
4. Qwt User's Guide: Qwt - Qt Widgets for Technical Applications [Electronic resource] – Mode of access: qwt.sourceforge.net. – Date of access: 14.09.2018.
5. ChartDirector Chart Component and Control Library [Electronic resource] – Mode of access: www.advsofteng.com – Date of access: 14.09.2018.