ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА»

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра общей физики

БАКАЛАВРСКАЯ РАБОТА

Программно-аппаратная реализация устройств синтеза и анализа звука для учебного эксперимента на базе одноплатного компьютера

Выполнял студент
405 группы
Пышков Николай Иванович

_____

Научный руководитель
к.п.н., ст. преп. КОФ Селиверстов А.В.

_____

Допущен к защите «_____» _____2019 г.

Заведующий кафедрой общей физики
д.ф.-м.н., профессор А.М. Салецкий

_____

Москва
2019г.

Содержание

Введение.....3

Часть 1. Методические основы создания компьютерных демонстрационных приложений 5

1.1. Требования к реализации демонстрационных приложений 5

1.2. Концепция интерфейса компьютерных демонстрационных приложений. Понятие об операторском и демонстрационном интерфейсах 10

Часть 2. Аппаратная платформа, принципы построения интерфейса демонстрационного приложения и его типовые элементы 12

2.1. Общая структура интерфейса демонстрационного приложения.....12

2.2. Семейство одноплатных компьютеров Raspberry Pi и их аналоги...13

2.3. Кроссплатформенная среда разработки Qt 16

2.4. Работа со звуком в библиотеке QtMultimedia.....18

2.5. Реализация элементов графического интерфейса с помощью языка разметки QML 20

Часть 3. Разработанные демонстрационные программы.....21

3.1. Генератор звуковых сигналов.....21

3.2. Анализатор спектра звуковых сигналов.....23

3.3. Осциллографический анализатор звуковых сигналов 24

3.4. Использование внешних утилит.....25

3.5. Интерфейс приложения.....26

Заключение 28

Библиография 29

Введение

Одним из основных общедидактических требований в преподавании физики является требование наглядности изложения материала [1]. В курсе общей физики в качестве средства наглядности могут выступать различные демонстрационные материалы (плакаты, фотографии, видеозаписи и проч.), но основным средством традиционно был и остается демонстрационный эксперимент.

С 90-х годов XX века в процессе обучения активно используется компьютерная техника. В связи с этим получили широкое распространение компьютерные демонстрации, дополняющие, а иногда даже заменяющие натурный эксперимент.

Безусловно, натурный эксперимент имеет целый ряд сложностей. Некоторые демонстрации очень капризны в постановке или настройке, а значит, могут не всегда получаться на лекции. Необходимое для их проведения оборудование может иметься далеко не во всех физических аудиториях. Кроме того, параметры экспериментальной установки часто нельзя изменять быстро в широких пределах, а это увеличивает время проведения демонстрации, снижает темп лекции, концентрацию внимания у учащихся и с большей долей вероятности приводит к тому, что эксперимент вообще не будет выбран лектором для показа. Ряд демонстраций требует громоздкого оборудования, которое сложно расположить в лекционной аудитории. К тому же не все изучаемые в курсе физические явления можно представить с помощью натурных демонстраций.

Настоящая работа посвящена разработке и созданию устройств синтеза и анализа звука в составе автоматизированных программно-аппаратных комплексов и их использование в количественном демонстрационном эксперименте. Стоит отметить, что разработанные

программы являются не широко используемыми на лекциях компьютерными моделями, а программно реализованными аналогами обычных приборов. Поэтому далее вместо словосочетания «демонстрационная программа», мы будем использовать термин «**демонстрационное приложение»**. Он означает компьютерную программу, предназначенную для демонстрации и выполняющую роль элемента экспериментальной установки.

Целью настоящей работы являлось создание недорогого, компактного, эргономичного, легко тиражируемого программно-аппаратного комплекса для синтеза и анализа звука.

Мы поставили перед собой следующие задачи:

- разработка списка демонстрационных приложений для работы со звуковыми сигналами;
- анализ требований к реализации компьютерных демонстрационных приложений;
- создание интерфейса приложения с использованием языка разметки QML;
- реализация приложений на основе разработанной модели интерфейса с учётом сформулированных требований.

Часть 1. Методические основы создания компьютерных демонстрационных приложений

1.1. Требования к реализации демонстрационных приложений

Как уже отмечалось во введении, компьютерные демонстрации имеют ряд преимуществ перед натурными экспериментами. К основным из них можно отнести:

- отсутствие необходимости длительной настройки;
- отсутствие необходимости специфического дорогостоящего оборудования;
- необходимость наличия только компьютера и мультимедийного проектора;
- возможность быстро изменять параметры в широких пределах;

Чтобы приложение подходило для использования на лекции, нужно, чтобы оно удовлетворяло следующим требованиям [2, 3]:

- быстрота действия;
- устойчивость работы;
- интероперабельность;
- удобный интерфейс.

Чем медленнее работает приложение, тем менее оно привлекательно в сравнении с натурным экспериментом, поэтому **быстрота действия** – одно из основных требований к нему.

Устойчивость работы – это гарантия воспроизводимости. При одинаковых условиях приложение должно выдавать один и тот же результат.

Интероперабельность означает возможность использования приложения в различных операционных средах. Чем больше

операционных систем, в которых демонстрационное приложение может работать, тем более оно будет востребовано.

Удобный интерфейс должен быть эргономичным, чтобы можно было быстро найти необходимый элемент интерфейса и быстро выполнить требуемое действие.

Эргономика – научно-прикладная дисциплина, занимающаяся изучением и созданием эффективных систем, управляемых человеком [4]. Эргономика изучает движение человека в процессе производственной деятельности, затраты его энергии, производительность и интенсивность при конкретных видах работ.

Каждый человек оценивает интерфейс приложения по своим индивидуальным критериям. Однако исследования показывают, что многие из этих критериев объективны и являются общими для всех людей. При условии, что нет проблем совместимости приложения с конкретной ЭВМ, любой человек изначально оценивает функциональные возможности приложения на основе интерпретации видимых элементов ее интерфейса. Также он оценивает, действительно ли все работает в соответствии с его ожиданиями и устраивает ли его визуальное представление результатов приложением.

Можно сформулировать основные требования к реализации интерфейса демонстрационных приложений [2, 3]:

- простой и удобный способ ввода данных;
- высокая скорость работы с интерфейсом;
- наглядность результатов;

Простой и удобный способ ввода данных означает отсутствие ряда проблем, возникающих при вводе параметров работы приложения.

Высокая скорость работы с интерфейсом должна достигаться путем модификации и оптимизации размещения элементов интерфейса

таким образом, чтобы на их поиск, интерпретацию и использование уходило как можно меньше времени.

Наглядность результатов работы приложения зависит от многих факторов, например, корректного воспроизведения изображения или правильного представления числовой информации [3].

Кроме того, можно выделить ряд психофизиологических и эргономических требований, которые необходимо учитывать при создании компьютерных демонстрационных приложений.

Одним из характерных отличий демонстрационного приложения, используемого на лекции, от любой другой компьютерной программы является очень малое время взаимодействия с ней аудитории. При демонстрации на лекции какого-либо эксперимента важно, чтобы внимание зрителей концентрировалось на той точке пространства, где это нужно в данный момент. Для этого преподаватель специально акцентирует внимание слушателей на том, что считает важным. Учебные экспериментальные установки проектируются, собираются и устанавливаются с тем расчетом, чтобы в нужный момент привлечь внимание зрителей к физическому явлению. Кроме того следует понимать, что на демонстрации присутствуют два типа людей: лектор и зритель. Лектор выступает в роли обычного пользователя то есть, изменяет параметры работы приложения и наблюдает за результатом, тогда как зритель видит только результат и должен понять какие изменения произошли.

При работе с демонстрационным приложением также необходимо учитывать факторы, обеспечивающие привлечение внимания. Однако при разработке приложения они, как правило, игнорируются или учитываются некорректно. То, что представляется хорошо видимым на экране монитора при индивидуальной работе с приложением, может оказаться малозаметным при показе в аудитории с использованием вспомогательных

технических средств (например, мультимедийного проектора), с другого, причем фиксированного, ракурса, при отличающемся освещении. Внимание аудитории будет ослабевать из-за того, что возрастут усилия, связанные с напряжением зрения и необходимостью распознования увиденного.

Если возникают проблемы при работе с приложением, то оно создает паузы в лекции и заставляет отвлекаться от учебного процесса, ослабляет внимание аудитории или концентрирует его на посторонних вещах. Безусловно, эти проблемы не являются непреодолимыми, и их наличие вряд ли сделает невозможным использование приложения. Однако их учет и устранение позволит существенно повысить внимание студентов непосредственно к физическим явлениям и увеличить эффективность работы лектора с аудиторией. Все это может оказать влияние на повышение эффективности лекции в целом.

Внимание можно определить как «сосредоточенность деятельности субъекта в данный момент времени на каком-либо реальном или идеальном объекте (предмете, событии, образе, рассуждении и т.д.)» [4]. Состояние внимания противоположно рассеянному состоянию. Внимание необходимо человеку для понимания происходящего, точной оценки ситуации и анализа сложных систем, понимание которых занимает много времени. Выделяют три вида внимания [5]. **Произвольное**, или активное, внимание характеризуется направленностью субъекта на сознательно выбранную цель. Существует и другая форма **непроизвольного** внимания. Оно выражается в переключении внимания на неожиданное изменение или появление значимых сигналов. Кроме того существует третий вид внимания называется **постпроизвольным** вниманием, которое появляется в процессе освоения деятельности и увлеченности выполняемой работой. Оно не требует усилий воли, так как поддерживается интересом к ней. К

характеристикам внимания относят его избирательность, объем, устойчивость, возможность распределения или переключения.

Избирательность внимания характеризует его направленность на любой аспект стимула. Избирательность внимания измеряется количеством одновременно отчетливо осознаваемых объектов и характеризует ресурсы внимания человека. **Объем** внимания близок к объему кратковременной памяти и составляет 7-9 элементов. **Устойчивость** внимания определяется по длительности выполнения задания, требующего непрерывного внимания. Под **переключением** внимания понимается возможность более или менее легкого перехода от одного вида деятельности к другому.

Чтобы учитывать особенности внимания учащихся, сформулируем на основе обобщения данных из научно-методической литературы [3, 5]. основные психофизиологические требования к интерфейсу демонстрационных приложений:

- наиболее важные в данный момент элементы приложения должны быть лучше всего видны;
- изображения должны быть хорошо видны, то есть иметь большой размер, высокие яркость и контрастность;
- все существенные элементы должны быть видимы одновременно;
- одновременно в поле зрения должно быть не более 7-9 элементов;
- вспомогательные элементы не должны быть в поле зрения;
- внешний вид приложения должен учитывать особенности непроизвольного внимания;
- должно быть выбрано цветовое содержание (например, цветовое зонирование), повышающее концентрацию внимания на определенных элементах интерфейса.

1.2. Концепция интерфейса компьютерных демонстрационных приложений. Понятие об операторском и демонстрационном интерфейсах

На основе изложенных выше требований к интерфейсу демонстрационных приложений, рассмотрим общую концепцию интерфейса, реализующего их.

Высокоуровневые языки программирования всегда предоставляют в пользование разработчиков программного обеспечения определенный набор универсальных интерфейсных элементов. Эти элементы проектируются с тем расчетом, чтобы их возможности позволяли решать большинство часто встречающихся задач.

Но при использовании набора интерфейсных элементов для написания интерфейса необходимо учитывать, что демонстрационные приложения представляют собой специфическое направление компьютерных программ. Анализ существующего набора интерфейсов демонстрационных приложений показывает, что для правильной работы таких программ необходимо наличие всего нескольких основных функций интерфейса [2]:

- ввод и вывод числовой информации;
- функция активации и остановки работы программы;
- вывод количественных результатов (графиков);

Реализация этих функций с использованием доступных интерфейсам элементов позволяет решать все поставленные задачи.

Интерфейс – определенная граница между независимыми взаимодействующими объектами: пользователем и программой. При взаимодействии пользователя с компьютерной программой мы рассматриваем их как два независимых объекта. При использовании демонстрационных приложений на лекции необходимо рассматривать не два, а три независимых объекта. Это лектор (демонстратор), зрители

(слушатели лекции) и демонстрационное приложение. Подобное разделение обусловлено различием информационных потоков, возникающих на лекции во время показа демонстраций. Для лектора (или демонстратора) поток двунаправленный, то есть такой пользователь совершает произвольное управляющее действие и получает его визуальное подтверждение, то есть задействовано произвольное и постпроизвольное внимание. В случае же пассивного восприятия информации зрителями (слушателями) они фактически должны догадываться, какое действие было произведено, выделяя эту информацию из общего потока с использованием произвольного, а чаще непроизвольного внимания.

Интерфейс, определяющий в условиях проведения лекции границу взаимодействия между программой и лектором, назовем **операторским**. Операторский интерфейс подразумевает восприятие не только с монитора ПК, но и с другого оборудования для вывода информации (проекционный телевизор, мультимедийный проектор). Информативность операторского интерфейса должна быть направлена больше на зрителя, нежели чем на самого оператора.

Интерфейс, определяющий в условиях проведения лекции границу взаимодействия между программой и зрителями, назовем **демонстрационным**. Демонстрационный интерфейс подразумевает восприятие информации без ее ввода.

Часть 2. Аппаратная платформа, принципы построения интерфейса демонстрационного приложения и его типовые элементы

2.1. Общая структура интерфейса демонстрационного приложения

На основе сформулированных ранее требований к интерфейсу демонстрационных приложений можно привести примеры внешнего вида и функциональных возможностей элементов интерфейса, удовлетворяющих сформулированным требованиям и находящихся в рамках предложенной концепции построения интерфейса демонстрационного приложения.

Для выполнения основных функций, необходимых в демонстрационном приложении, достаточно использования следующих типовых интерфейсных элементов:

- поле ввода и вывода числовой информации;
- элемент или функция активации и остановки работы приложения;
- область визуального представления результатов (графиков);

Однако, чтобы интерфейс приложения удовлетворял изложенным в главе 1 требованиям, одного наличия перечисленных выше элементов недостаточно. Немаловажно расположение интерфейсных элементов друг относительно друга.

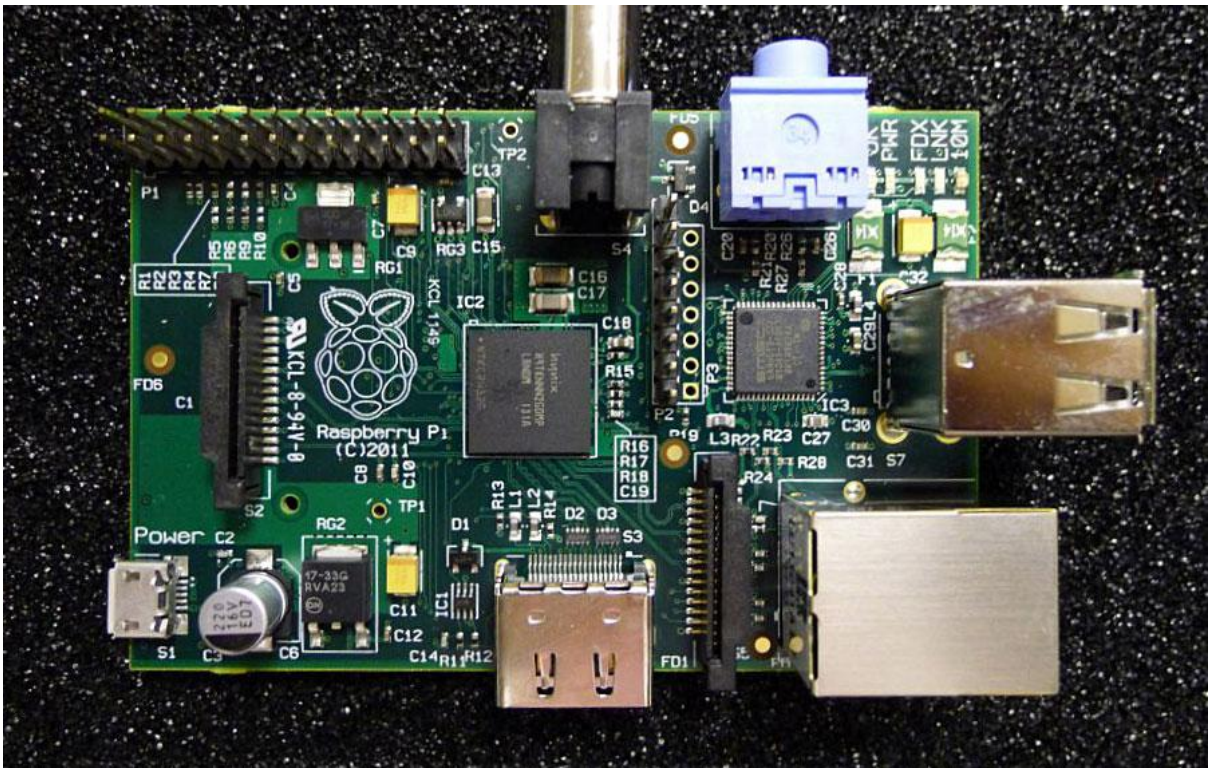
Область визуального представления результатов должна занимать большую часть интерфейсного пространства.

Полей ввода/вывода числовой информации может быть несколько, но они должны быть сгруппированы. Размер этих полей должен быть достаточно большим, но не настолько, чтобы эти поля отвлекали внимание от области визуального представления результатов. Поля ввода/вывода численной информации должны давать возможность

изменять значения или быстро, или дискретно (с помощью соответствующих элементов интерфейса). При начале работы с полем оно должно изменять свой вид для привлечения внимания.

2.2. Семейство одноплатных компьютеров Raspberry Pi и их аналоги

Raspberry Pi – это одноплатный микрокомпьютер. Внешне он представляет собой небольшую плату размерами 85,6 × 53,98 × 17 мм.



На данный момент существует около 8 моделей Raspberry Pi. Подробно мы рассмотрим модель Raspberry Pi 3B, так как именно она используется в данной работе. Рассмотрим технические характеристики данной модели:

Процессор:	ARM Cortex-A53
Частота:	1,2 ГГц
Количество ядер:	4
Объем оперативной памяти:	1024 Мб
Звуковой контроллер:	интегрированный, 16 бит, 44100 Гц

Внешние порты:	4 USB, 1 Audio Jack, 1 LAN, 1 HDMI
Необходимое питание:	5 В, 1 А / 12 В, 2 А
Дополнительно:	40 GPIO pins

Процессор ARM Cortex-A53 обеспечивает хорошую производительность (для микрокомпьютера). Наличие 4 USB портов позволяет использовать различные периферийные устройства. Видеовыход HDMI позволяет выводить изображение на различные мультимедийные устройства. Звуковой контроллер выводит на цифровой звуковой выход сигнал разрядностью 16 бит с частотой дискретизации 44100 Гц. Наличие 40 пинов GPIO позволяет организовывать различные схемы ввода/вывода.

Главными положительными особенностями микрокомпьютера Raspberry Pi являются:

- компактные размеры;
- низкое энергопотребление;
- низкая стоимость;
- высокая производительность в классе одноплатных компьютеров;
- ориентированность на возможности операционной системы Linux;
- широкое сообщество разработчиков программного обеспечения;
- наличие различных интерфейсов ввода-вывода данных;

Компактные размеры, высокая производительность и низкое энергопотребление обеспечивают Raspberry Pi высокую мобильность, что позволяет использовать его в самых различных прикладных задачах. Стоимость данного микрокомпьютера составляет всего лишь 50 долларов США (менее 3500 рублей на момент написания работы), то есть за столь малую сумму мы получаем полноценный компьютер с широкими возможностями применения. Так как основной операционной системой Raspberry Pi является Linux, то, как следствие, существует большое

разнообразие программного обеспечения, ориентированного на решение различных прикладных задач. Если же подходящего программного обеспечения нет, то Linux предоставляет все возможности для того, чтобы создать ПО для решения конкретной прикладной задачи. Широкое сообщество разработчиков программного обеспечения, которое, во-первых, динамично развивает возможности Raspberry Pi, во-вторых, многие прикладные задачи уже решены, в-третьих, существует огромное количество статей, форумов, где можно найти информацию, касающуюся того или иного вопроса. Наличие различных интерфейсов позволяет использовать Raspberry Pi для чтения информации от различных внешних датчиков (температуры, акслелерометра, ускорения), а также для управления двигателями постоянного тока (используя ШИМ), аудио, жидкокристаллическими дисплеями или светодиодными индикаторами.

Главным недостатком Raspberry Pi является отсутствие аналогового ввода сигналов встроенным способом (не имеет входа, в который бы можно было подключить микрофон). Это не позволяет обрабатывать цифровые сигналы, используя аппаратные возможности Raspberry Pi. Однако эта проблема решается путем создания схемы ввода с помощью GPIO или же с помощью различных периферийных устройств.

Разработка оказалась настолько удачной, что у неё появилось много клонов (Banana Pi, Orange Pi, и т.п.). Высокая конкуренция на рынке приводит к стремительному развитию одноплатных компьютеров, росту их производительности и падению стоимости. Их аппаратное обеспечение может отличаться друг от друга, поэтому для совместимости в прикладных программах уместно использовать только стандартные периферийные устройства с возможностью их простого конфигурирования средствами операционной системы или платформы программирования.

2.3. Кроссплатформенная среда разработки Qt

Для создания интерфейса, соответствующего описанным выше требованиям, удобно использовать среду разработки Qt. Qt это кроссплатформенный инструментарий разработки программного обеспечения на языке программирования C++, имеющий «привязки» ко многим другим языкам программирования [6, 7].

Среда Qt позволяет запускать написанное с ее помощью программное обеспечение в большинстве современных операционных систем путем простой перекомпиляции программы для каждой ОС без изменения исходного кода. Она включает в себя все основные библиотеки, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая библиотеками для работы со звуком и базами данных. Qt является полностью объектно-ориентированной, легко расширяемой и поддерживающей технику компонентного программирования. Таким образом, использование этой библиотеки позволяет создать учебно-методическую среду, одинаковую при работе в различных операционных системах: Microsoft Windows, Linux/UNIX, MacOS.

Библиотека Qt является свободно распространяемой (согласно лицензиям GPL-2 или QPL-1, с определенными ограничениями в случае коммерческого использования), что становится актуальным в связи с лицензированием программного обеспечения, используемого в учебном процессе.

Со времени своего появления в 1996 году коммерческая версия библиотеки Qt легла в основу тысяч успешных проектов во всем мире. Кроме того, Qt является фундаментом популярной рабочей среды KDE, входящей в состав многих дистрибутивов GNU/Linux.

Перечислим основные достоинства Qt, определившие наш выбор этой среды разработки для создания программы:

1. **Объектно-ориентированное программирование** – это парадигма программирования, в которой основными концепциями являются понятия объектов и классов. Класс – это тип, описывающий устройство объектов-экземпляров, инкапсулирующий (т.е. включающий в себя) как данные, так и процедуры их обработки. На основе простых классов можно описывать более сложные, при этом возможно как наследование данных и процедур, так и их полиморфизм (изменение). Объектно-ориентированное программирование возникло в результате развития идеологии процедурного программирования, где данные и процедуры обработки формально не связаны. Объектно-ориентированное программирование в настоящее время является абсолютным лидеров в области прикладного программирования. В то же время в области системного программирования до сих пор лидирует сугубо процедурный язык C, хотя при взаимодействии системного и прикладного уровней операционных систем заметное влияние стали оказывать языки объектно-ориентированного программирования. Поэтому Qt, написанная на языке C++, стала своеобразным мультиплатформенным стандартом.
2. Большое число **стандартных функций**, необходимых для численного моделирования. Так, например, в среде уже определены классы векторов, матриц и операции над ними.
3. Мощные и удобные **средства для разработки графического пользовательского интерфейса**, включающие в себя все необходимые стандартные элементы, которые к тому же могут быть легко модифицированы для конкретной программы.
4. Удобные **средства для работы со звуком**, включающие в себя все необходимые элементы для низкоуровневой работы со звуком.

5. **Интеграция в различные среды разработки** программного обеспечения, как коммерческие (Microsoft Visual Studio 2003/2005), так и свободно распространяемые (Dev-Cpp).
6. **Простота и гибкость программирования**, основанная на концепции слотов и сигналов, повышающая эффективность и скорость разработки программы.
7. **Кроссплатформенность** позволяет запускать написанное в среде Qt программное обеспечение в большинстве современных ОС путем простой перекомпиляции программы для каждой ОС без изменения исходного кода.

2.4. Работа со звуком в библиотеке QtMultimedia

Общий принцип работы приложения основан на работе со стандартными устройствами ввода-вывода звука средствами библиотеки QtMultimedia. В роли стандартного устройства вывода выступает звуковая карта микрокомпьютера.

Необходимо отметить существенную особенность аппаратного обеспечения одноплатных микрокомпьютеров семейства Raspberry Pi: встроенная звуковая карта имеет два выхода (на разъём HDMI и на отдельное гнездо для аудиоразъёма Jack 3.5 mm), но не имеет входа, в который можно было бы подключить микрофон. Поэтому для ввода сигналов необходимо использовать дополнительные внешние переферийные устройства, например, внешнюю звуковую карту.

Для записи данных в устройство в библиотеке QtMultimedia определён специальный класс QAudioOutput, который позволяет выбрать параметры вывода данных и воспроизвести их через внешнее устройство вывода. Можно выбрать следующие параметры: частота дискретизации (44100 Гц), количество каналов (моно или стерео), размер сэмпла (8, 16

или 24 бит), порядок записи байтов (LittleEndian или BigEndian), тип сэмпла (SignedInt или UnsignedInt).

Чтение данных из устройства и их последующая обработка происходит с помощью класса QAudioInput, который также определен в библиотеке QtMultimedia.

После описания всех необходимых классов для работы со звуком схематически можно изобразить логику работы программы для синтеза звука в следующем виде.

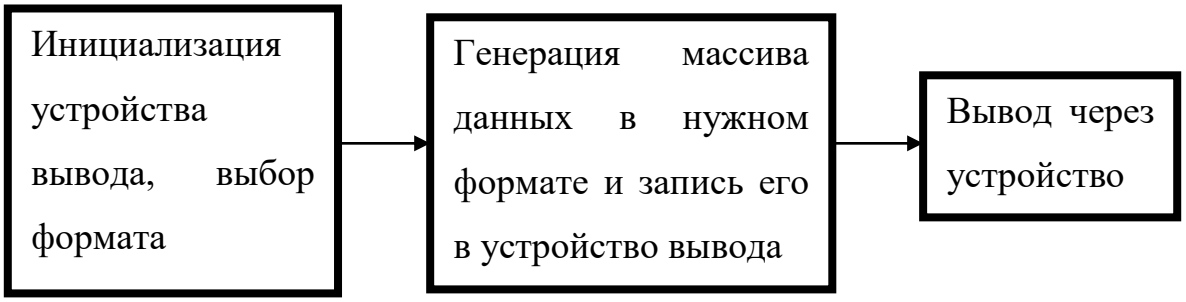


Рис 1. Схема работы программы для синтеза звука

Схематически логика работы программы для анализа звука будет выглядеть следующим образом.

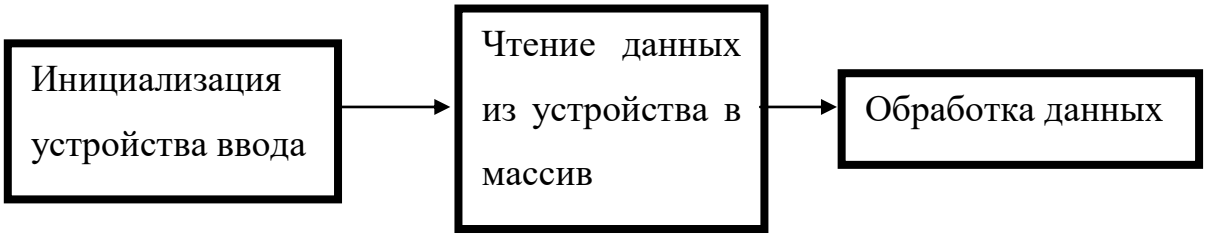


Рис 2. Схема работы программы для анализа звука

2.5. Реализация элементов графического интерфейса с помощью языка разметки QML

Как было сказано выше, среда разработки Qt имеет удобные средства разработки графического интерфейса программ. Интерфейс программ реализован на основе QML – декларативного языка разметки и программирования, который базируется на JavaScript и предназначен для реализации быстро работающих пользовательских интерфейсов [8]. Он часто используется для создания приложений, ориентированных на мобильные устройства, а, следовательно, прост и удобен в использовании и экономичен в плане потребления ресурсов.

QML позволяет использовать парадигму **Model-View-Controller** (**MVC**, «Модель-Представление-Контроллер») – схему разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента таким образом, что модификация каждого элемента может осуществляться независимо.

Модель предоставляет данные и реагирует на команды контроллера, изменяя свое состояние. **Представление** отвечает за отображение данных модели пользователю, реагируя на изменения модели. **Контроллер** интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Использование данной парадигмы, безусловно, упрощает и убыстряет разработку.

Часть 3. Разработанные демонстрационные программы

Исходя из сформулированных требований и предложенной визуальной модели интерфейса, нами были разработаны устройства для синтеза и анализа звука: генератор звуковых сигналов, анализатор спектра звуковых сигналов, осциллографический анализатор звуковых сигналов.

При разработке учитывалось то, какими возможностями должны обладать разработанные устройства.

Генератор звуковых сигналов:

- выбор формы сигнала;
- регулировка частоты, амплитуды;
- наглядное представление формы сигнала.

Анализатор спектра звуковых сигналов:

- наглядное представление результатов расчёта (спектр сигнала);
- возможность работы как с устройством ввода, так и с внешними файлами.

Осциллографический анализатор звуковых сигналов:

- наглядное отображение данных, считываемых с устройства ввода;
- регулировка шкалы времени, амплитуды.

3.1. Генератор звуковых сигналов

В качестве одного из тестовых приборов, использующего возможности одноплатного компьютера Raspberry Pi, нами был выбран генератор сигналов звуковой частоты. Этот выбор был обусловлен следующими факторами:

- Достаточно высокая востребованность звуковых генераторов в демонстрационных экспериментах;
- Наличие встроенного цифрового звукового выхода;

- Возможность преобразования уровня сигнала до необходимого с помощью внешних бытовых усилителей, например, активные компьютерные колонки;

Данное приложение генерирует звуковой сигнал заданной частоты, амплитуды и формы и выводит его на устройство вывода (звуковую карту).

В его возможности входят: регулировка частоты и амплитуды звука, а также выбор формы сигнала. На настоящий момент реализованы следующие формы сигналов: синусоидальная, прямоугольная (меандр), треугольная и пилообразная. Добавление других форм не представляет сложности, поскольку для этого необходимо лишь написание функции, выводимый через звуковую карту массив значений. Следует понимать, что не любой сигнал мы можем синтезировать точно. Если количество отсчётов (N), приходящихся на период сигнала, целое число, то частота генерации соответствует заданной. Если же N – не целое, то частота генерации отличается от заданной в приложении.

Если массив содержит один период сигнала, то частоты, которые можно воспроизвести таким образом, равны $f_N = \frac{f_0}{N}$, где f_0 - частота дискретизации встроенной звуковой карты (44100 Гц), N – длина массива. Погрешность воспроизведения заданной частоты можно уменьшить, используя массив, содержащий m периодов: $f_{m,N} = \frac{mf_0}{N}$. Согласно исследованиям человек не различает на слух частоты, отличающиеся на 0,1 тона. Поэтому за допустимое отклонение принято отношение частот, составляющее 0,1 тона (то есть $2^{1/60} \approx 1{,}01162$) [9].

Для достижения необходимой точности был разработан алгоритм адаптивной настройки частоты, подбирающий параметры генерации цифрового сигнала.

Данное устройство способно сгенерировать сигнал в диапазоне частот от 20 до 4096 Гц. Такое ограничения связано с тем, что при меньших частотах сигнал будет не различим на слух. Кроме того будут проявляться нелинейные эффекты, связанные с аппаратным ограничением одноплатного компьютера, что приведет к искажению формы сигнала. При больших частотах будет проявляться эффект Муара, заключающийся в наложении частоты генерации сигнала на частоту дискретизации встроенной звуковой карты, что тоже приведёт к искажению формы сигнала.

3.2. Анализатор спектра звуковых сигналов

Это приложение рассчитывает и представляет графически спектр генерируемого сигнала. Её работа основана на алгоритме быстрого преобразования Фурье с прореживанием по времени (БПФ).

Генерируемый периодический сигнал представляет собой набор дискретных значений. Рассмотрим процедуру вычисления спектра периодического дискретного сигнала. Так как сигнал периодический, будем раскладывать его в ряд Фурье. Коэффициенты X_n этого ряда будут равны:

$$X_n = \sum_{k=0}^{N-1} x_k e^{-j\frac{2\pi nk}{N}}, \quad (1)$$

где X_n – амплитуда n-й гармоники, x_k – k-й отсчет дискретного сигнала, N – количество дискретов.

Данная формула представляет собой дискретное преобразование Фурье (ДПФ). Расчет по этой формуле требует много вычислительных ресурсов, так как сводится к N комплексным операциям сложения и умножения. Таким образом, расчет всего дискретного преобразования Фурье потребует N^2 пар операций «умножение-сложение».

Чтобы ускорить процесс вычислений используют алгоритм БПФ. Пусть N – четное число. Выделим в формуле (1) два слагаемых,

соответствующих элементам исходной последовательности с четными и нечетными номерами:

$$X_n = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-j\frac{4\pi mn}{N}} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-j\frac{2\pi(2m+1)n}{N}} \quad (2)$$

Таким образом, в формуле (2) получаются две суммы, которые представляют собой ДПФ последовательностей отсчетов с четными и нечетными номерами размерностью $\frac{N}{2}$. Следовательно, если количество дискретов $N = 2^k$, то деление последовательностей на две части можно продолжать до тех пор, пока не получатся двухэлементные последовательности, ДПФ которых рассчитывается вообще без использования операций умножения (достаточно вычислить сумму и разность двух отсчетов). Число требуемых при этом пар операций оценивается как $N * log_2 N$, что значительно быстрее, чем при обычном ДПФ [10].

В данной работе берётся выборка из 4096 отсчётов и по этой выборке рассчитывается БПФ. Выбор именно такого количества отсчётов связан с выделением числа гармоник (больше 2) в получившемся спектре сигнала для того, чтобы определить закономерность изменения зависимости спектральной плотности от частоты.

3.3. Осциллографический анализатор звуковых сигналов

В силу технических трудностей, связанных со считыванием данных с внешнего устройства ввода, осциллографический анализатор звуковых сигналов не удалось реализовать.

Однако, можно кратко сформулировать основные моменты реализации данного устройства:

1) считывание данных с внешнего устройства ввода и запись этих данных в массив;
2) обработка данных;

3) наглядное представление данных.

3.4. Использование внешних утилит

Среда разработки Qt имеет в своем распоряжении библиотеку QtCharts для построения графиков. Однако, данная библиотека появилась не так давно и ещё не адаптирована для Raspberry Pi. Поэтому для построения графиков наглядных результатов работы приложения использовалась внешняя программа Gnuplot.

Gnuplot - свободная программа для создания двух- и трёхмерных графиков. Она предоставляет все необходимые средства для построения графиков наглядного результата работы приложения. Кроме того, это утилита командной строки, что позволяет вызывать её изнутри работающего приложения.

Массив генерируемого сигнала и массив значений спектра сигнала записываются в файл. Затем вызывается утилита Gnuplot, считывающая данные из этого файла, и записывает соответствующий графический файл. Далее эти графики считываются приложением и помещаются на область отображения наглядных результатов работы приложения.

3.5. Интерфейс приложения

Интерфейс приложения представлен на рисунке.

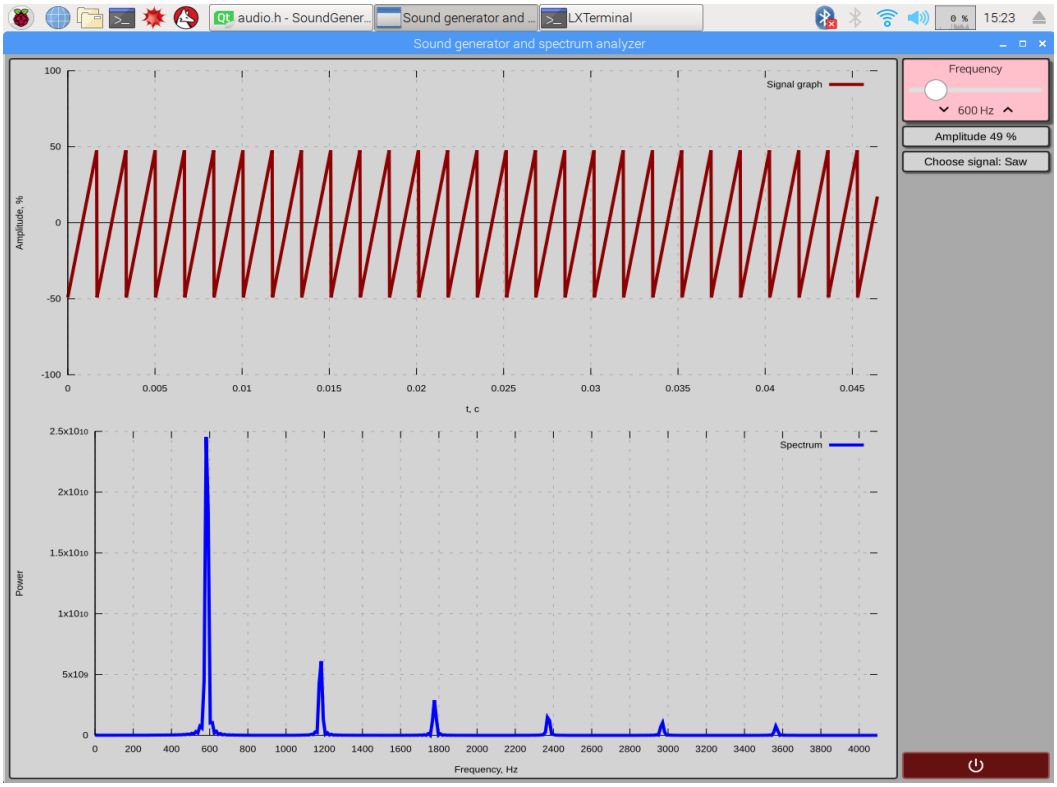


Рис. 3. Интерфейс разработанного демонстрационного приложения

В правой верхней части окна приложения расположены элементы управления параметрами генерируемого сигнала. При выборе какого-либо управляющего элемента интерфейса он увеличивается в размерах и появляются элементы, позволяющие изменять значение выбранного параметра как в широких (ползунок), так и в узких (стрелочки вверх/вниз) пределах. Кроме того открывшееся поле выделяется цветом, тем самым фокусируя непроизвольное внимание зрителей. Им уже не нужно догадываться, какой параметр изменяется. При переходе к любому другому элементу интерфейса текущий элемент возвращается к исходному размеру, и в нём остается только информация о названии и значении той величины, которую изменил оператор.

В центре находится область отображения наглядных результатов работы приложения. Сверху представлен график генерируемого сигнала, снизу его спектр. Графики построены разными цветами, чтобы можно было легко отличить один от другого.

Заключение

1. На основе анализа психофизиологических, методических и эргономических требований к интерфейсам демонстрационных приложений разработана визуальная модель интерфейса, учитывающая возможности свободно распространяемой объектно-ориентированной кроссплатформенной среды разработки Qt и языка разметки QML.
2. С использованием предложенной визуальной модели интерфейса разработаны следующие демонстрационные приложения:
   - генератор звуковых сигналов;
   - анализатор спектра звуковых сигналов;
   - осциллографический анализатор звуковых сигналов.
3. В среде Qt создано тестовое демонстрационное приложение для Raspberry Pi, объединяющее в себе генератор звуковых сигналов, осциллографический анализатор звуковых сигналов и анализатор спектра звуковых сигналов.

Библиография

1. Ситаров В. А. Дидактика. М. Издательский центр «Академия», 2002. – 386с.
2. Селиверстов А. В. Современные лекционные демонстрации по разделу «Волновая оптика» курса общей физики – М., Физический факультет МГУ, 2005. – 260с.
3. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем.
4. Служба тематических толковых словарей. URL: http://www.glossary.ru/
5. Данилова Н. Н. Психофизиология. М., Аксепт-Пресс, 1998. – 367с.
6. Бланшет Ж., Саммерфилд М. Qt 4: Программирование GUI на C++. 2-е дополнен.
7. Шлее М. Qt 4: Профессиональное программирование на C++. СПб., БХВ-Петербург., 2007. 880с.
8. QML // Википедия. [2011—2018]. Дата обновления: 30.08.2018. URL: https://ru.wikipedia.org/?oldid=94791382 (дата обращения: 25.01.2019).
9. Слух // Википедия. [2006—2018]. Дата обновления: 17.10.2018. URL: https://ru.wikipedia.org/?oldid=95658379 (дата обращения: 25.01.2019).
10. Сергиенко А.Б. Цифровая обработка сигналов. – 2-е. –Спб: Питер, 2006.