Содержание:

ВВЕДЕНИЕ

В настоящее время любая современная мониторинговая система включает в себя прикладное программное обеспечение (ПО) для визуализации данных. Как правило, запуск этого ПО предполагает наличие рекомендуемой операционной системы (ОС), в большинстве своих случаев ОС компании Microsoft. Однако сейчас наблюдается тенденция использования кроссплатформенных средств для разработки ПО. В результате этого появляется возможность запуска готового программного продукта на разных ОС, включая и мобильные ОС. Кроме того, в связи с бурным распространением Интернета популярным направлением разработки ПО стала разработка веб-приложений или веб-сервисов.

Актуальность темы исследования: Веб-приложение является полезным дополнением к клиентской прикладной программе (приложению). Обычно веб-приложение дает возможность удаленного использования мониторинговой системы. Это означает, что пользователь не привязан к месту расположения аппаратной части мониторинговой системы и может использовать ее из любой точки мира, где есть рекомендуемое интернет- соединение. Важно заметить, что разработка веб-приложений в значительной степени отличается от разработки клиентских приложений, и это, в свою очередь, создает некоторые проблемы. В частности, это проблема создания универсального графического интерфейса пользователя (GUI). Чтобы клиентское приложение и веб-приложение были реализованы в едином графическом стиле, необходимо приложить достаточно усилий как разработчику интерфейса клиентского приложения, так и разработчику интерфейса веб-приложения. В конечном счете величина усилий одного или другого разработчика будет зависеть от того, интерфейс какого приложения будет задавать общий стиль.

Целью работы является изучение вариантов построения интерфейса программ: особенности и эволюция.

Предметом работы являются информационные технологии.

Объектом принципы построение интерфейсов.

Задачи работы:

Описать основные характеристики и принципы интерфейса;

Изучить эволюцию интерфейса;

Показать практику применения интерфейсов.

Методами исследования является общенаучный диалектический метод познания и вытекающие из него частно-научные методы: исторический, социологический, логический, системно-структурный.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕТКТЫ ПОСТРОЕНИЯ ИНТЕРФЕЙСА

1.1.Основные характеристики и принципы интерфейса

Интерфейс - совокупность средств, при помощи которых пользователь общается с различными устройствами.

Лучший пользовательский интерфейс — это такой интерфейс, которому пользователь не должен уделять много внимания, почти не замечать его.

Если говорить о самых общих принципах проектирования пользовательских интерфейсов, то можно назвать три основных положения:

Программа должна помогать выполнить задачу, а не становиться этой задачей.

Первый принцип — это уже упоминавшаяся выше прозрачность интерфейса. Интерфейс должен быть легким для освоения и не создавать перед пользователем преграду, которую он должен будет преодолеть, чтобы приступить к работе. [2]

При работе с программой пользователь не должен ощущать себя дураком.

Второй принцип часто нарушают те авторы программ, которые слишком недооценивают умственные способности пользователей. В глазах таких разработчиков пользователи видятся толпой этаких бестолковых болванов, в

лучшем случае — беспомощных и нерадивых созданий, не способных разобраться в самых элементарных ситуациях. Это обусловлено разными причинами.

Программа должна работать так, чтобы пользователь не считал компьютер дураком.

Несмотря на стремительное развитие информационных технологий, многие компьютерные программы все еще имеют примитивный искусственный интеллект. Они прерывают работу пользователя глупыми вопросами и выводят на экран бессмысленные сообщения, повергая его в недоумение в самых простых ситуациях [4]

Пользовательский интерфейс является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера.

Лучший пользовательский интерфейс — это такой интерфейс, которому пользователь не должен уделять много внимания, почти не замечать его. Пользователь просто работает, вместо того, чтобы размышлять, какую кнопку нажать или где щелкнуть мышью. Такой интерфейс называют прозрачным — пользователь как бы смотрит сквозь него на свою работу.

Чтобы создать эффективный интерфейс, который делал бы работу с программой приятной, нужно понимать, какие задачи будут решать пользователи с помощью данной программы и какие требования к интерфейсу могут возникнуть у пользователей. Это сделать гораздо легче, если вы используете свою программу для собственных нужд, ведь в данном случае вы являетесь не только разработчиком, но и пользователем программы, смотрите на нее глазами ее аудитории. [15]

Огромную роль играет интуиция — если разработчик сам терпеть не может некрасивые и неудобные интерфейсы, то при создании собственной программы он будет чувствовать, где и какой именно элемент нужно убрать или добавить. Необходимо иметь художественный вкус, чтобы понимать, что именно придаст интерфейсу красоту и привлекательность.

Западные исследователи в области HCI сформулировали основные принципы проектирования пользовательских интерфейсов компьютерных программ. Как и в любой другой науке, существует довольно много различных методик и классификаций, которые можно найти в книгах по HCI, выпушенных за рубежом, а

также на иностранных Web-сайтах.

Если говорить о самых общих принципах проектирования пользовательских интерфейсов, то можно назвать три основных положения:

1. Программа должна помогать выполнить задачу, а не становиться этой задачей.

2. При работе с программой пользователь не должен ощущать себя дураком.

3. Программа должна работать так, чтобы пользователь не считал компьютер дураком.

Довольно эмоциональные формулировки, но, тем не менее, поразительно верные. [16]

Во-первых, традиционным слегка высокомерным отношением программистов к простым пользователям. Это еще можно было понять в восьмидесятых и начале девяностых годов XX века, когда обычные персональные компьютеры не имели доступных широкой аудитории программных и аппаратных средств для построения привлекательных графических интерфейсов и работы с ними. Самой распространенной операционной системой в то время была MS DOS, основанная на интерфейсе командной строки. Поэтому эффективно работать с персональным компьютером могли люди только с довольно серьезной подготовкой. Кроме того, парк «персоналок» был относительно невелик даже в США, не говоря уже об остальных странах, и, как следствие, число пользователей компьютеров было небольшим. [3]

Сегодня же такой пренебрежительный взгляд на пользователя явно неуместен. Работа с персональным компьютером предполагает относительно не большую начальную подготовку пользователя: интерфейсы компьютерных программ, в первую очередь операционной системы Windows, являющейся законодателем мод в индустрии массового программного обеспечения, становятся все проще и доступнее для понимания людей. Да и число компьютеров в мире сегодня в несколько раз больше, чем десять лет назад.

Вторая причина слишком большой недоверчивости программистов к познаниям и квалификации пользователей – чрезмерное увлечение построением так называемой «защиты от дурака». Дело в том, что классические учебные курсы по программированию учат, что большинство ошибок в работе программы вызываются не дефектами исходного кода или программного окружения, а действиями

пользователя — например, вводом данных неправильного формата (допустим, текста вместо цифр). Поэтому программист при разработке приложения должен написать функции по проверке результатов как можно большего числа действий пользователя и предусмотреть максимальное количество вариантов развития событий. Это совершенно правильный подход, но многие программисты настолько усложняют «защиту от дурака», делают ее такой громоздкой, что работа пользователя с программой начинает напоминать известное «шаг вправо, шаг влево считается побегом». Происходит довольно обычная вещь: то, что задумывалось как решение проблемы, само начинает создавать проблемы. [7]

И, наконец, третья причина во многом обусловлена поведением самих пользователей. Часто при возникновении малейших затруднений при работе с программой пользователь тут же обращается в службу технической поддержки, не удосужившись даже взглянуть на справочную систему продукта, посмотреть секцию «Ответы на частые вопросы» на Web-сайте программы или даже просто чуть-чуть подумать! Отчасти тут вина самих авторов программ. Как говорят опытные разработчики пользовательских интерфейсов: «Если уже на этапе знакомства с программой пользователь вынужден обращаться к справочной системе, над интерфейсом нужно серьезно работать». Поэтому, чтобы соблюсти второй из общих принципов построения интерфейсов и не давать пользователю повода почувствовать, будто его принимают за идиота, не нужно давать разрабатываемой программе слишком большие полномочия и право указывать пользователю, что именно ему делать. Некоторые программисты не знают или не желают осознавать этого и загоняют пользователей своих программных продуктов в тесные рамки, навязывая определенный стиль работы. [12]

Один из примеров такого неправильного отношения к пользователю является отказ программы выполнить вполне естественную с точки зрения пользователя программных продуктов такого рода операцию и вывод диалогового окна, требующего выполнить какую-то другую последовательность действий. Этим «прославился», например, известный текстовый редактор «Блокнот» из состава Windows 95. Если пользователь открывал эту программу и решал перед началом набора текста дать создаваемому «Блокнотом» по умолчанию файлу «Untitled» какое-нибудь имя, выбрав из меню команду Сохранить как, редактор отказывался сделать это, показывая сообщение: «Вы не ввели какой-либо текст, чтобы его можно было сохранить. Наберите какой-нибудь текст, а затем попытайтесь [сохранить его] снова». Этим создатели «Блокнота» не только отвергли стиль работы очень многих пользователей (перед началом набора текста дать имя

файла), но сбили с толку и тех, кто был знаком с «Блокнотом» по предыдущим версиям Windows. Например, в шестнадцатиразрядной Windows 3.1 «Блокнот» позволял сохранять пустые файлы безо всяких проблем. Опытные пользователи, знакомые с принципами работы операционной системы, тоже были в недоумении: если из контекстного меню Проводника Windows в меню Создать выбрать пункт Текстовый документ, то получившийся файл длиной 0 байт открывается «Блокнотом» без каких-либо затруднений. К счастью, в последующих версиях Windows «Блокнот» стал более дружественен к пользователю. [10]

Другой пример недооценки возможностей пользователя — вывод информационных сообщений в ситуациях, когда этого не требуется. Многие авторы наделяют свои программы излишней «болтливостью» из благих намерений — например, для того, чтобы облегчить освоение продукта или информировать пользователей о полезных функциях программы. Однако вполне может оказаться так, что пользователь уже достаточно уверенно чувствует себя при работе с программой и не нуждается в подсказках, выскакивающих каждую минуту, а некоторые полезные, с точки зрения автора программного продукта, функции для конкретного пользователя таковыми не являются, Поэтому среди разработчиков программного обеспечения хорошим тоном считается предоставление пользователю возможности отключить вывод информационных сообщений. Это позволяет сохранить легкость освоения продукта для начинающих пользователей и одновременно с этим добиться, чтобы информационные сообщения не вызывали у опытных пользователей раздражения. [11]

И, наконец, третий принцип — «Программа должна работать так, чтобы пользователь не считал компьютер дураком».»

Несмотря на стремительное развитие информационных технологий, многие компьютерные программы все еще имеют примитивный искусственный интеллект. Они прерывают работу пользователя глупыми вопросами и выводят на экран бессмысленные сообщения, повергая его в недоумение в самых простых ситуациях. В результате многие люди, работающие с компьютерами, раздраженно восклицают: «Как мне надоела эта тупая машина!»

1.2.Эволюция интерфейса

Несколько десятилетий назад компания Apple и во сне не помышляла о тех богатых функциональных возможностях, которые теперь считаются

неотъемлемыми особенностями ее машин. В те времена пользователи не могли одновременно запускать две или более программы, включая системное приложение Finder.

Даже увеличение объема оперативной памяти не дало тесной интеграции приложений друг с другом и с Finder. Единственным инструментом управления ОС была программа Switcher. Она позволяла переключаться между прикладными программами, причем окно текущего приложения «соскальзывало» в левую сторону экрана, в то время как окно другой программы «выползало» справа. Таким образом пользователь получал доступ к любому из запущенных приложений, однако на экране по-прежнему могло отображаться только единственное окно. [5]

Рис. 1. Графический интерфейс пользователя классической» Mac OS со стандартными системными значками на Рабочем столе

Радикально изменилась эта ситуация в 1985 г., когда двое студентов из Беркли продемонстрировали в головном офисе компании Apple ПО MacWrite. В самом текстовом редакторе MacWrite не было ничего примечательного. Однако когда они уменьшили окно программы, заполнявшее весь экран, под ним неожиданно обнаружилось окно приложения Finder, запущенного в фоновом режиме. Это событие ознаменовало рождение новой, многозадачной Mac OS. [14]

К чему мы пришли. После этого стремительного рывка вперед Mac OS постепенно обретала свои характерные особенности: от меню (иерархического либо всплывающего) и вкладок до перетаскивания — наиболее простого и эффективного

метода взаимодействия человека и компьютера. И сейчас Apple опережает Microsoft в таких важных областях, как удаление приложений и управление памятью.

Рис. 2.Графический интерфейс пользователя Mac OS X с Dock и новым файловым браузером поистине превращает Macintosh в Macintosh

Mac OS X обещает далеко уйти от конкурентов и в другой сфере. Эта версия ОС должна стать надежнее предыдущих. Конечно, Apple и раньше предлагала инструменты, с помощью которых удавалось аварийно выгрузить «зависшие» приложения (например, путем нажатия соответствующих функциональных клавиш). Однако по-настоящему действенное «лечение» — это устранение причины, вызывающей сбой, а не борьба с его последствиями, что и было реализовано в Mac OS X. Но богатые функциональные возможности — это палка о двух концах, тут и возникает закономерный вопрос: останутся ли компьютеры Macintosh под управлением новой ОС самими собой, сохранив свои прежние достоинства? [8]

Пользователи Macintosh всегда вольны были перемещать и сохранять приложения и документы в своем «виртуальном» рабочем пространстве как им заблагорассудится. Другие системы, от Windows до Unix, обязывают запоминать местоположение объекта в иерархической структуре папок и файлов. Теоретически это должно сводить к минимуму беспорядок на жестком диске, однако на практике беспорядок всего-навсего переносится с Рабочего стола машины в глубины памяти пользователя. Поскольку большинство из нас лучше

справляются с неразберихой на Рабочем столе (который, по крайней мере можно увидеть), чем с кашей в собственной голове, преимущество Mac OS перед другими ОС налицо.

Dock вместо привычной панели внизу экрана. В первых, «черновых» вариантах Mac OS X Рабочий стол имел традиционный вид с располагающимися на Dock стандартными значками типа Trash (Корзины) и пр. Они были доступны по щелчку мыши в виде новых открытых диалоговых окон. Однако за последующие несколько месяцев Apple немало потрудилась над совершенствованием своего детища, и уже в бета-версии новая ОС обрела ряд особенностей, отличающих Macintosh от всех прочих компьютеров. [7]

В Mac OS X отсутствует характерная для Mac OS 9 панель со вкладками в нижней части экрана, содержавшая пиктограммы. Ее заменил новый системный объект — Dock, на котором пользователь может располагать от двадцати до тридцати самых различных элементов. Это очень изящное решение, но большое количество помещенных на Dock одинаковых значков делает его существенно менее удобным для работы, чем обычная панель Mac OS 9.

Тут нелишне вспомнить: в свое время разработчики Apple пришли к выводу, что непоименованные пиктограммы вызывают у пользователей затруднения, и руководством к действию избрали лозунг: «Одно слово стоит тысячи картинок», — который, впрочем, и сейчас остается актуальным. К сожалению, названия значков, помещенных на Dock, не возникают до тех пор, пока не подведешь к ним мышь. И если у пользователя открыто шесть документов MS Word, то в поисках нужного ему придется подводить мышь поочередно к каждому значку, что долго и неудобно. [11]

Internet на автономных компьютерах. Графический интерфейс пользователя Mac OS X нацелен на работу в Internet. Новый файловый браузер, по мысли Apple, призван «перенести» мощь Web-браузеров на Рабочий стол несетевых машин. К сожалению, глобальная компьютерная сеть и «локальный» Рабочий стол — это все-таки разные вещи. Ведь «посетитель» Сети путешествует среди миллионов Web-узлов, объединяющих миллиарды страниц, о содержании которых он ничего не знает. Операции с цифровыми данными на автономном ПК — совсем другое дело: здесь имеется сравнительно небольшое количество файлов, созданных (или переписанных из какого-либо источника) самим пользователем и организованных по хорошо известному ему принципу.

Возможно, новый файловый браузер найдет свою нишу в крупных локальных сетях, где необходимо отслеживать «судьбу» значительного числа документов, созданных (либо собранных) различными людьми. Однако, насколько известно, поклонники компьютеров Macintosh предпочитают организовывать свою жизнь по собственному усмотрению, а не так, как это задумали производители ОС. [6]

«Изюминки» индивидуальной настройки

Графический интерфейс пользователя Mac OS X не является чем-то «застывшим», раз и навсегда определенным: Apple предусмотрела возможность выполнять его настройку. Индивидуальные установки позволяют пользователю придать ОС любой внешний вид и выбрать наиболее удобный метод управления компьютером. Однако достаточно взглянуть на MS Windows, чтобы понять, к каким неудобствам может привести злоупотребление подобными вещами.

Создайте «свою» Mac OS. Конечно, владельцы Macintosh и раньше имели возможность варьировать системные настройки, но в жестко ограниченных пределах. Среди доступных средств оформления выделялись «кожицы» (skins) — специальные файлы, которые обеспечивали различные цветовые схемы Finder и панелей инструментов приложений. Одни «кожицы» были красивы, другие — отвратительны, но главное, они привносили в оформление Mac OS элемент эстетики. Некоторые инструменты, такие как QuicKeys и AppleScript, позволяли задавать и автоматизировать последовательность смены системных событий, но радикально изменить базовые установки ОС было нельзя.

Mac OS X подарила обладателям Macintosh, во-первых, долгожданную свободу выбора системных настроек, во-вторых, возможность с помощью огромного количества бесплатных дополнительных программ преображать ОС и приложения по собственному усмотрению. И если раньше «кожицы» только придавали Mac OS внешний вид последних версий Linux, то теперь Mac OS X обеспечит графический интерфейс пользователя Linux на вашем компьютере. [8]

Переработанная Mac OS выглядит так, будто Apple решила полностью разрушить все то, что раньше считалось неотъемлемыми особенностями Macintosh. На самом же деле под этим скрывается задача радикального «исправления» недостатков ОС. Одни «исправления» изящны, но бесполезны, другие грозят перевернуть с ног на голову, казалось бы, раз и навсегда сложившиеся представления пользователей о работе с Macintosh. Некоторые вещи так же удивительны, как и новый Finder. И если Mac OS X завоюет признание потребителей, Apple продолжит

совершенствование своего продукта в том же направлении и реализует в следующих версиях ОС самые передовые идеи.

Сохраняя простоту. И все же пользователей волнует, сохранит ли Mac OS X легкость эксплуатации, издавна присущую всем продуктам Apple? Возможно ли, чтобы столь мощная ОС была удобной и понятной для неспециалиста? Безусловно, богатые функциональные возможности и простота графического интерфейса пользователя должны быть двумя сторонами одной медали. К примеру, компьютеры Palm Pilot удобны и любимы миллионами пользователей именно потому, что, имея относительно простой интерфейс, обеспечивающий выполнение узких задач, они в то же время сравнимы по производительности с «настоящими» Macintosh. [7]

Новые функциональные возможности Mac OS X превосходят все то, что мы видели на ПК когда-либо ранее. Однако способна ли Apple и дальше сохранять декларируемую простоту использования своих продуктов? Или она последует неудачному примеру Microsoft, Sun и других компаний, создающих громоздкие и неповоротливые «тракторы» для рядовых пользователей? Может быть, Mac OS X все-таки превратится в послушный в управлении и стремительный «Порш»?

В любом случае, по какому пути производители Macintosh ни пойдут, начало следующего года обещает стать знаменательным для всех пользователей этих машин.

ГЛАВА 2.ПРАКТИКА ПРИМЕНЕНИЯ ИНТЕРФЕЙСА

2.1. Интерфейс передачи цифровых аудио сигналов

В настоящее время на потребительском рынке представлены многочисленные цифровые аудиосистемы: проигрыватели компакт-дисков со звуковыми записями, цифровые процессоры звука, системы передачи звукового сопровождения цифрового телевизионного сигнала и т.д. [4]Звуковой сигнал в цифровой форме обрабатывается в таких системах интегральными схемами различного назначения:

• аналого-цифровые и цифро-аналоговые процессоры (АЦП и ЦАП);

• цифровые сигнальные процессоры;

• микросхемы декодирования и коррекции ошибок;

• цифровые фильтры;

• микросхемы цифровых интерфейсов ввода/вывода. Вследствие этого стандартизированные коммуникационные структуры, как и в случае с другими интерфейсами, крайне необходимы как разработчикам и производителям звукового оборудования, так и фирмам, выпускающим интегральные схемы, т.к. это увеличивает гибкость систем и минимизирует затраты на их создание. Для достижения данных преимуществ и был разработан интерфейс I2 S (Inter-IC Sound – звук, передаваемый между интегральными схемами) – специализированный однонаправленный последовательный интерфейс для цифрового аудио. [9]

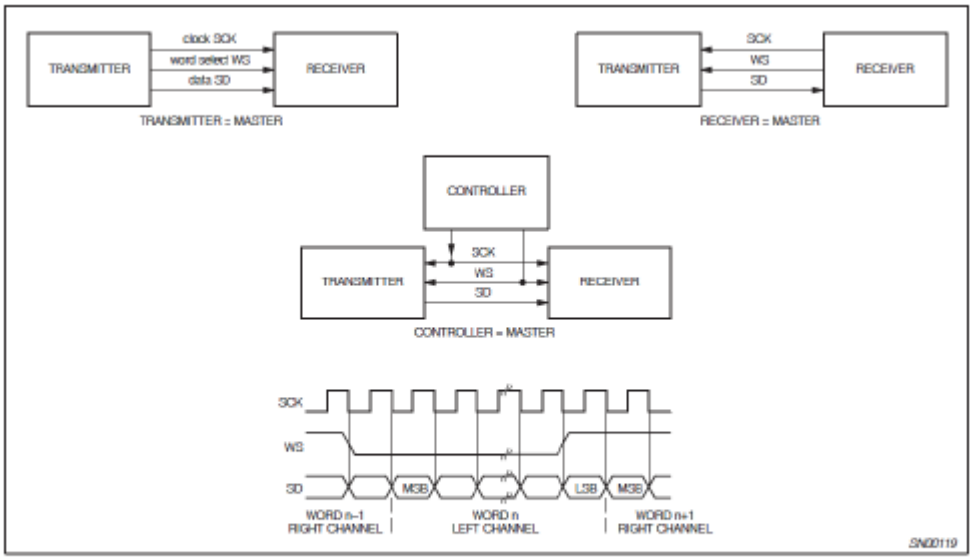

Рисунок 34. Варианты реализации обмена информацией по интерфейсу I^2S

Рис. 3. Варианты реализации обмена веществ по интерфейсу

Последовательный данные передаются по I2S в двоичном виде, старшим битом вперед. Такой порядок передачи бит выбран не случайно – это позволяет передатчику и приемнику использовать различную разрядность звуковых данных и при этом успешно передавать информацию без каких-либо дополнительных настроек. Принцип реализации данной возможности следующий:

если приемник получает больше бит, чем его собственная разрядность данных, то все выходящие за разрядную сетку биты, начиная с самого младшего, игнорируются;

если приемник получает меньше бит, чем его собственная разрядность, то все недостающие младшие биты в составе принятого слова принимаются за ‘0’.

Таким образом, самый старший бит передаваемых слов всегда имеет фиксированную позицию, а положение самого младшего бита зависит от принятой в каждом из абонентов разрядности звуковых данных.

Передатчик всегда отправляет старший бит очередного слова со звуковыми данными в первом тактовом периоде, следующим за изменением WS. То есть время, в течение которого, WS остаётся относительно тактовых импульсов неизменной определяет длину слов со звуковыми данными, отправляемыми передатчиком. [14]

Последовательные данные от передатчика могут быть синхронизированы как задним фронтом (изменение от высокого к низкому уровню), так и передним фронтом (изменение от низкого к высокому уровню) такового сигнала.

Однако биты данных должны защёлкиваться приемником только по переднему фронту тактового импульса. При проектировании встраиваемой системы следует обратить на данное требование внимание во избежание битового смещения данных при передаче.

Линия выбора канала своим состоянием определяет передачу «левого» или «правого» звукового канала в каждый момент времени:

WS = 0 – передается слово канала №1 («левый»);

WS = 1 – передаётся слово канала №2 («правый»). WS может изменяться и по переднему, и по заднему фронту тактового импульса. Изменение должно происходить за один тактовый импульс до начала передачи самого старшего бита слова звуковых данных. Ведомый, при этом, защёлкивает состояние линии WS только по переднему фронту такта.

Временные требования

В соответствии с требованиями к структуре систем, объединенных нтерфейсом I2S, любое из устройств может выполнять функции ведущего, обеспечивая генерацию

тактового сигнала. А ведомое устройство обычно использует внешний тактовый сигнал, поступающий от ведущего, для внутренней синхронизации. Это означает, что необходимо принимать в расчет вероятные задержки распространения сигналов синхронизации, выбора канала и собственно последовательных данных. Общая задержка является суммой следующих слагаемых:

1. Задержка между внешним тактовым сигналом (от ведущего) и внутреннего тактового сигнала ведомого;

2. Задержка между внутренним тактовым сигналом последовательными данными и/или сигналом выбора канала.

Для входа данных и входа выбора канала задержка между внешними и внутренними тактовыми импульсами не имеет значения, т.к. она только увеличивает время установки действительных уровней сигнала. Основное значение имеет разница между временем распространения сигнала от передатчика и временем восстановления приемника для начала приема данных. [11]

Все временные требования специфицируются относительно периода тактовых импульсов или минимально допустимой длительности периода для интегральной схемы. Это означает, что в будущем возможно получение более высокой скорости передачи данных через интерфейс.

Спецификация интерфейса I2S определяет соответствие логических сигналов ‘0’ и ‘1’ уровням напряжения традиционной технологии производства интегральных схем ТТЛ (Транзисторно-Транзисторная Логика):

низкий уровень (логический ‘0’) на входе: менее 0.8В;

высокий уровень (логическая ‘1’) на входе: более 2.0В;

низкий уровень (логический ‘0’) на выходе: менее 0.4В;

уровень (логическая ‘1’) на выходе: более 2.4В.

(Нагрузочная способность выходов также соответствует ТТЛ-технологии: при низком уровне не менее -1.6мА, при высоком уровне не более 0.04мА.)

В настоящее время, однако, в результате гораздо более широкого распространения интегральных схем, выполненных по технологии КМОП (Комплементарная логика на транзисторах Металл-Оксид-Полупроводник), имеющих различное напряжение

питания, уровни напряжения для логических сигналов интерфейса обычно соответствуют уровням напряжения всех остальных цифровых сигналов, которыми оперирует интегральная схема.

2.2 Универсальный графический интерфейс пользователя на примере системы акустического мониторинга

Рассмотрим наиболее популярные в настоящий момент способы построения интерфейсов клиентских приложений на языке C++ как наиболее используемом для разработки ПО, для ОС Microsoft Windows (MS Windows) и ОС Linux. Главным средством разработки ПО для MS Windows является MS Visual Studio [1].

Эта интегрированная среда разработки (IDE) позволяет разрабатывать ПО на разных язы- ках программирования, но основными языками, конечно, являются C++ и C#. Для разработки интерфейса приложения имеются два ос- новных средства – Windows Forms (WinForms) и Windows Presentation Foundation (WPF). Большая часть существующих приложений для MS Windows разработана с использованием WinForms, однако с по- явлением более современных версий ОС (MS Windows 7, 8) система WPF становится более популярной. Кроме того, последние версии MS Visual Studio позволяют использовать язык разметки HTML5 для построения интерфейсов, близких по стилю к нативным веб- приложениям.

Однако стоит заметить, что коммерческая лицензия MS Visual Studio является платной, как и лицензия MS Windows, что, несомненно, является недостатком для низкобюджетных проектов. Если говорить о низкобюджетных проектах, то тут наиболее подходящим вариантом является ОС Linux. Помимо того, что большинство дистрибутивов этой ОС является абсолютно бесплатным, в том числе и для коммерческого использования, также имеется ряд бесплатных средств для разработки качественного ПО для ОС Linux. 348 Самым распространенным средством для разработки ПО на языке C++ является кроссплатформенный инструментарий Qt [2]. Важно подчеркнуть, что Qt позволяет разрабатывать приложения не только для ОС Linux, но и для MS Windows, Mac OS X, Android и других UNIX-подобных ОС. Разработчики Qt предлагают как бесплатную для коммерческого использования, так и платную лицензию с дополнительными возможностями. Но, исходя из современной практики разработки ПО с помощью

этого инструментария, бесплатной лицен- зии оказывается больше чем достаточно. Если проводить аналогию с MS Visual Studio, то в Qt мы имеем IDE Qt Creator.

Здесь альтернативой WinForms являются так назы- ваемые виджеты (Qt Widgets), а альтернатива для WPF – Qt Quick. Также в Qt Creator имеется возможность создания интерфейсов на основе HTML5. Но наиболее интересным модулем инструментария является встраиваемый веб-движок WebKit, который лежит в основе всех современных веб-браузеров. Подобный модуль имеется и в MS Visual Studio, но он имеет ряд ограничений, и тем более нас больше интересуют низкобюджетные средства, которые позволяют уменьшить издержки при создании программного продукта. Веб-движок – это ядро браузера, он отвечает за правильное отображение веб- страниц. Модуль Qt WebKit позволяет создавать интерфейс клиентского приложения с использованием техники разработки интерфейсов веб-приложений.

В основе создания интерфейса веб-приложения лежит устоявшийся стек технологий. Он включает язык разметки HTML (HTML 4, 5), каскадные таблицы стилей (CSS 2, 3) и скрипто-вый язык JavaScript с богатым выбором дополнительных библиотек (каркасов).

Отдельного внимания заслуживает тот факт, что скорость появления новых полезных каркасов для языка JavaScript стремительно растет, а это делает разработку насыщенных функционалом приложений более быстрой и удобной. Теперь решение проблемы создания универсального GUI лежит на поверхности, но это только на первый взгляд. Рассмотрим традиционный способ создания универсального GUI с помощью модуля Qt WebKit на примере модуля визуализации данных системы акустиче- ского мониторинга, разрабатываемой в рамках кандидатской диссерта- ционной работы [3].

Под системой акустического мониторинга подразумевается система, включающая аппаратную и программную части. Аппаратная часть системы состоит из сенсорной сети акустических датчиков, данные с которых обрабатываются на микроконтроллере и отправляются по какому-либо интерфейсу (UART, IEEE 802.X и др.) на персональный компьютер (ПК). Программная часть состоит из на- бора прикладных программ, работающих как на локальном ПК (клиентское ПО), так и на удаленном сервере (серверное ПО). Традиционный метод подразумевает использование межпроцессного взаимодействия посредством встроенного механизма Qt. Здесь подразумевается взаимодействие между основной логикой клиентского приложения, изображенной на рис. 4 как Обработчик данных, и GUI-

элементом.

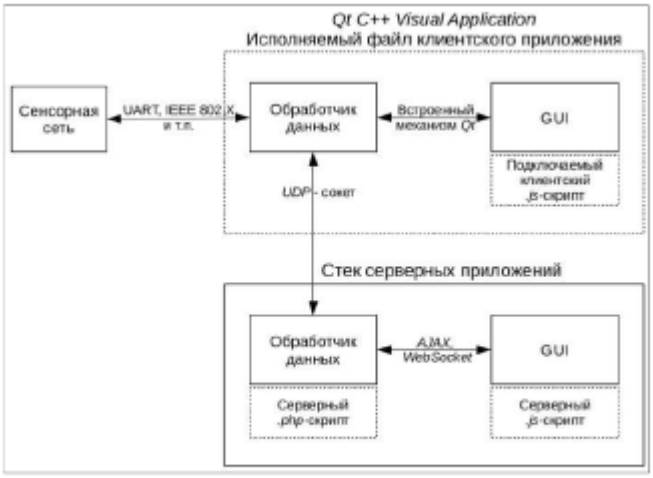

Рис. 1. Традиционный метод реализации универсального GUI

Рис. 4. Традиционный метод универсального GUI

Одним из недостатков такого подхода является то, что код для реализации GUI на языке JavaScript будет иметь специфические функции, которые будут актуальны только для клиентского Qt-приложения. Для серверного приложения, отвечающего за GUI, нужен будет другой, специфичный для серверной реализации, код. Например, в случае использования PHP-скрипта для реализации основной логики серверного приложения понадобится реализация межпроцессного взаимодействия с помощью какой-либо другой технологии (AJAX или WebSocket). Отсюда следует еще один недостаток, а имен- но использование дополнительного языка программирования для реализации основной логики серверного приложения и разработка нового алгоритма межпроцессного взаимодействия. Для решения этих проблем предлагается новый метод, основанный на использования упомянутой выше технологии WebSocket.

Суть метода заключается в том, чтобы унифицировать метод межпроцессного взаимодействия между основной логикой приложения и GUI как на клиентской стороне, так и на серверной. В этом случае появляется возможность использования JavaScript-кода для реализации универсального для обеих сторон GUI.

На рис. 5 видно, что теперь для межпроцессного взаимодействия как для клиентской, так и для серверной части используется технология WebSocket. Следовательно, теперь мы имеем один универсальный JavaScript-код для разных приложений.

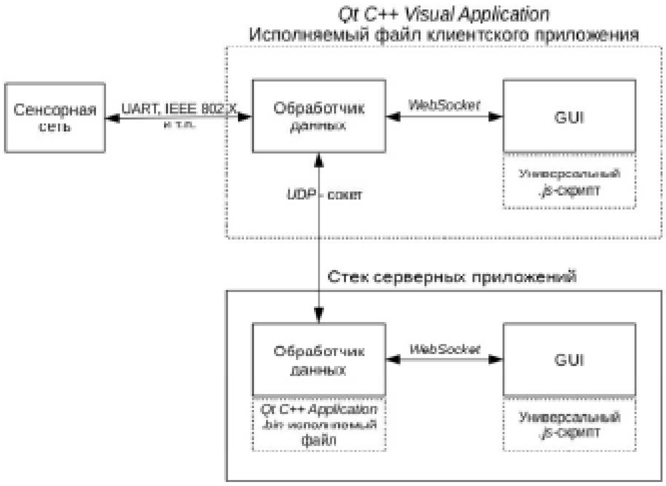

Рис. 2. Новый метод реализации универсального GUI

Рис. 5. Новый метод реализации универсального GUI

В этом случае необходимым условием является серверное приложение, основная логика которого реализована с помощью Qt, на не совсем привычном для веб-разработчиков языке C++. С одной стороны, такой подход к реализации серверного приложения усложняет задачу для узкоспециализированного веб-разработчика. Но, с другой стороны, мы имеем универсальные части кода, которые позволяют нам сэкономить время на дублировании одних и тех же по смыслу алгоритмов … использования технологи… …оторая имеется в интернете … …олее поздние

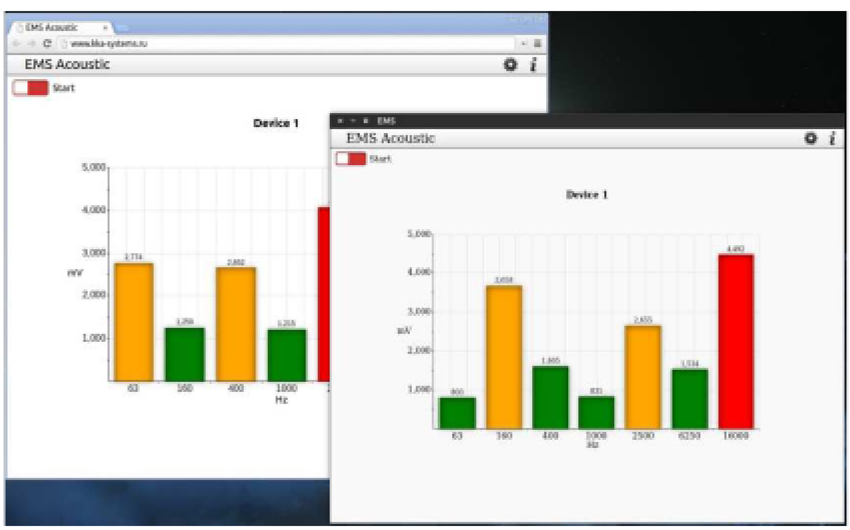

Рис. 3. Локальное (справа) и серверное (слева) приложения, запущенные на ОС Ubuntu 14.04

версии Qt.

Рис 6. Локальное и северное приложения

На рис. 6 приведен пример реализации нового метода создания универсального GUI для ОС Ubuntu 14.04. Как видно на рисунке, в конечном итоге мы получаем универсальный интерфейс как для локального приложения, запущенного в качестве исполняемого файла ОС, так и для серверного приложения, запущенного в современном веб-браузере. Так как для разработки ПО используются кросс-платформенные инструменты, это позволяет говорить о простой переносимости программного продукта на другие ОС в будущем.

ЗАКЛЮЧЕНИЕ

Понятие «интерфейс» сегодня имеет много разных аспектов, связанных со способами соединения чего-либо с чем-либо. Так, бывает интерфейс между устройствами, программный интерфейс, пользовательский интерфейс. Среди всех них, нас интересует пользовательский интерфейс, но что же это такое? Если кратко, то это способ общения конечного пользователя с программой. Естественно, он определяется создателем или чаще создателями программы, причем множественное число относится не к количеству писавших программу людей, а к количеству уровней, которыми пользуется проблемный программист.

Часть интерфейса определяется аппаратными средствами, часть - особенностями операционной системы и т.д., и каждый из этих уровней вносит свои ограничения в построение интерфейса. Но самая важная его часть, конечно, определяется тем, что делает проблемная программа, но здесь нас больше будет интересовать не ЧТО именно она делает, а КАК организовано её взаимодействие с пользователем.

Если программист создает интерфейс своей программы «с нуля», без использования каких-либо средств, то он обрекает себя на весьма трудоемкую работу, которую, как показывает опыт, ему вряд ли удастся довести до сколь-нибудь приличного вида. С другой стороны, на многих машинах существуют мощные средства организации интерфейса (в качестве примера, разумеется, годится ОС Windows), позволяющие строить довольно сложный интерфейс, не затрачивая на это больших усилий.

Однако использование стандартных средств создания интерфейса накладывает ограничения не только на способ общения программы с пользователем, что в общем-то полезно, но и на структуру самой программы, а тем самым осложняет построение программ, весьма простых при традиционных подходах.

СПИСОК ЛИТЕРАТУРЫ

1. Агуров, П. Интерфейс USB / П. Агуров. - М.: Книга по Требованию, 2006. - 556 с.
2. Агуров, Павел Интерфейс USB. Практика использования и программирования (+ CD-ROM) / Павел Агуров. - М.: БХВ-Петербург, 2006. - 564 с.
3. Гультяев А.К., Машин В.А. «Проектирование и дизайн пользовательского интерфейса», СПб.: КОРОНА принт, 2010. – 599с.
4. Грановская Р.М. Элементы практической психологии. СПб.: Свет, 2007. – 600с.
5. Гук, Михаил Интерфейсы устройств хранения. ATA, SCSI и другие. Энциклопедия / Михаил Гук. - М.: Питер, 2007. - 448 с.
6. 11. Интерфейсы СОРМ / Б.С. Гольдштейн и др. - М.: БХВ-Петербург, 2006. - 160 с.
7. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова, СПб.: Символ-Плюс, 2009. – 732с.
8. Коутс Р., Влейминк И. Интерфейс «человек-компьютер, Москва: Мир, 2010.
9. Мандел Т. Дизайн интерфейсов. – М.: ДМК Пресс, 2015. – 434с.
10. Минаси М. Графический интерфейс пользователя: секреты проектирования. – М.: Мир,2016. – 883с.
11. Проектирование пользовательского интерфейса на персональных компьютерах. Стандарт Фирмы IBM. – Вильнюс: DBS LTD, 2012. – 283с.
12. Психология цвета, «Рефл Бук», «Ваклер», 2016. – 488с.
13. Птахова И. Простая красота буквы. СПб.: Русская графика, 2007. – 593с.
14. Рассел, Джесси Интерфейс пользователя / Джесси Рассел. - М.: Книга по Требованию, 2012. - 431 с.
15. Уаттс Р. ЭВМ и непрофессиональные пользователи: Организация взаимодействия. – М.: Радио и связь, 1989. – 442с.
16. Молодые ученые – развитию текстильно-промышленного кластера (ПОИСК–2014): сб. матер. межвуз. науч.-техн. конф. аспи- рантов и студентов с междунар. участием. Ч. 2. – Иваново: Изд-во Иванов. гос. политехн. ун-та, 2014. – С. 25 - 29