Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. каф. ЭВМ
_____ Д.И. Самаль

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему
СМАРТ-ЗЕРКАЛО С ГОЛОСОВЫМ УПРАВЛЕНИЕМ

БГУИР ДП 1 – 40 02 01 01 040 ПЗ

Студент А.А. Ковш

Руководитель А.Г. Третьяков

Консультанты:

от кафедры ЭВМ А.Г. Третьяков

по экономической части И.В. Смирнов

Нормоконтролер А.С. Сидорович

Рецензент:

МИНСК 2017

РЕФЕРАТ

Дипломный проект предоставлен следующим образом. Чертежный материал: 6 листов формата А1. Пояснительная записка: 89 страниц, 28 рисунков, 13 таблиц, 17 литературных источников, 3 приложения.

Ключевые слова: программно-аппаратный комплекс, микроконтроллер, Raspberry Pi, смарт-зеркало, веб-приложение, .NET, Windows IoT, Universal Windows Platform, Windows Speech Recognizer, информативные блоки, голосовое управление, многофункциональность, практическое применение.

Объектом исследования и разработки является возможность получать информацию с помощью программно-аппаратного комплекса и последующее использование данного устройства.

Целью проекта является разработка программно-аппаратного комплекса смарт-зеркало, позволяющего получать различную контентную информацию: виджет часов, погоды, событий календаря, распознавание эмоций и музыкальный плеер. Смарт-зеркало будет включать в себя возможность управлять устройством при помощи голосовых команд. Для удобства управления и настройки данным устройством разработать приложение-интерфейс, с помощью которого можно настроить зеркало под конкретного человека и его нужды.

При разработке использовались миникомпьютер Raspberry Pi, среда разработки Microsoft Visual Studio, библиотеки Windows Speech Recognizer, Azure Emotion API, Entity Framework, Windsor Castle, а также фреймворки MVC3 и Bootstrap. Итоговый проект представляет собой веб-приложение по настройке смарт-зеркала, программное обеспечение для Raspberry Pi, реализованное на Universal Windows Platform для Windows 10 IoT Core и готовый экземпляр зеркала.

Областью практического применения – домашнее использование. Пользователями программы могут быть как взрослые, так и дети.

Данный программный продукт можно считать экономически эффективным, и он полностью оправдывает вложенные в него средства.

Дипломный проект является завершенным, поставленная задача решена в полной мере, присутствует возможность дальнейшего развития программного комплекса и увеличение его функционала.

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: ФКСиС. Кафедра: ЭВМ.
Специальность: 40 02 01 «Вычислительные машины, системы и сети».
Специализация: 40 02 01 – 01 «Проектирование и применение локальных компьютерных сетей».

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.И. Самаль
«____» _____ 20__ г.

ЗАДАНИЕ
по дипломному проекту студента
Ковш Анастасии Александровны

1 Тема проекта: «Смарт-зеркало с голосовым управлением» – утверждена приказом по университету от 7 февраля 2017 г. № 238-с.

2 Срок сдачи студентом законченного проекта: 1 июня 2017 г.

3 Исходные данные к проекту:
3.1 Миникомпьютер Raspberry Pi. Камера, микрофон, светодиоды, зеркало, дисплей.
3.2 Языки программирования: C#, javascript, HTML, CSS, XAML.
3.3 Среда разработки: Microsoft Visual Studio 2017.
3.4 Фреймворк: MVC3.
3.5 Графические инструменты: Razor, Bootstrap.
3.6 ORM технология: Entity Framework.
3.7 СУБД: Microsoft SQL Server 2014.
3.8 IoC-контейнер: Windsor Castle.
3.9 Операционная система: Windows 10 и Windows 10 IoT Core.
3.10 Платформа: Universal Windows Platform.

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов):
Введение 1. Обзор литературы. 2. Системное проектирование. 3. Разработка функциональной схемы. 4. Описание элементов схемы. 5. Разработка программных модулей. 6. Программа и методика испытаний. 7. Описание работы устройства. 8. Технико-экономическое обоснование

разработки программно-аппаратного комплекса смарт-зеркала с голосовым управлением. Заключение. Список литературы. Приложения.

5 Перечень графического материала (с точным указанием обязательных чертежей):

5.1 Вводный плакат. Плакат.

5.2 Заключительный плакат. Плакат.

5.3 Смарт-зеркало с голосовым управлением. Схема структурная.

5.4 Смарт-зеркало с голосовым управлением. Схема функциональная.

5.5 Смарт-зеркало с голосовым управлением. Диаграмма последовательности.

5.6 Смарт-зеркало с голосовым управлением. Диаграмма классов.

6 Содержание задания по экономической части: «Технико-экономическое обоснование разработки программно-аппаратного комплекса смарт-зеркала с голосовым управлением».

ЗАДАНИЕ ВЫДАЛ И.В. Смирнов

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов дипломного проекта	Объем этапа, %	Срок выполнения этапа	Примечания
Подбор и изучение литературы	10	02.02 – 10.02	
Разработка структурной схемы	20	10.02 – 23.02	
Разработка функциональной схемы	10	23.02 – 28.02	
Разработка диаграммы последовательности	10	28.02 – 15.03	
Разработка диаграммы классов	10	15.03 – 25.03	
Оформление первых разделов	10	25.03 – 31.03	
Выполнение заданий по экономике и охране труда	10	31.03 – 15.04	
Написание интерфейсного модуля	10	15.04 –01.05	
Оформление пояснительной записки	10	01.05 – 30.05	

Дата выдачи задания: 2 февраля 2017 г.

Руководитель А.Г. Третьяков

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В тридесятом царстве в тридевятом государстве жил-был Александр Сергеевич Пушкин и писал замечательные сказки, которыми на протяжении многих столетий зачитываются не только дети, но и взрослые. А ведь многие сказки становятся реальностью и уже давно живут среди нас. Человечество с каждым годом приобретает все больше недоверия. Почему нет? Информация поглощает нас с раннего утра и до позднего вечера. Чтобы чему-то поверить, необходимо увидеть это своими глазами, проверить, потрогать.

Война форматов, гаджетов, виртуальных игр и устройств виртуальной реальности. Вот что мы имеем в текущем столетии. Последние года два производители на крупнейших международных выставках электроники соревнуются уже не только в гаджетах, но и в «волшебных» зеркалах.

«Свет мой, зеркальце, скажи...» Сказочные прогнозы стали реальностью. Попытайтесь вспомнить то чувство волшебства, которое поглощало при чтении и просмотре сказок. Хотелось иметь волшебную вещь? Наверное, не только Вам.

Область по созданию устройств для удобного и наглядного отображения важной информации развивается с огромной скоростью. Только за прошедший год было придумано несколько проектов «умных» зеркал.

Смотреть в зеркало утром, среди дня или перед выходом – привычное дело для любого человека. Однако насколько информативнее будет при этих действиях видеть еще время, погоду, последние сводки новостей, курсы валют, список дел на сегодня и многое другое?

Данная тематика становится настолько популярной, что скоро будет не в новинку увидеть в магазине не толпу консультантов, готовых тебе помочь в выборе одежды, а смарт-зеркала. Данный подход уже вводится в магазинах на западе. Теперь покупатели могут подобрать идеальную по размеру одежду без навязывания чужого мнения и «человеческого фактора». Стоит лишь повернуться перед умным зеркалом на 360 градусов. Датчики проанализируют параметры человека и предложат готовый каталог подходящей одежды. Примерка может пройти без усилий и нервов. Главное выбрать на зеркале нужное платье и цвет. Возникает желание поделиться покупкой с близким человеком или спросить у него совета? Камера, встроенная в смарт-зеркало, без проблем справится с такой задачей. Нужно будет указать лишь адрес для отправки фотографии. Смарт-зеркало – это больше, чем Ваше отражение. И сказка становится реальностью.

За основу дипломного проекта взята идея удобства и многофункциональности – смарт-зеркало. В частности стоят задачи показывать информативные блоки, включать подсветку при распознавании человека, а так же возможность управлять устройством при помощи голосовых команд. Для удобства управления данным устройством разработать приложение-интерфейс, с помощью которого можно настроить зеркало под конкретного человека и его нужды.

1 ОБЗОР ЛИТЕРАТУРЫ

Согласно теме дипломного проекта задачей проектирования является разработка программно-аппаратного комплекса смарт-зеркало с голосовым управлением. Далее будут рассмотрены существующие аналоги проекта, наиболее популярные микроконтроллеры и мини компьютеры, решения, касающиеся серверной части.

1.1 Обзор существующих аналогов

Не так давно американская компания Griffin анонсировала свою версию смарт-зеркала: Griffin Connected Mirror [1]. Внушительное на внешний вид зеркало оснащено такой информацией, как прогноз погоды, пропущенные уведомления с мобильного телефона и любой другой совместимой техники: зарядного устройства, кофе-машины или даже тостера. Для настройки используется мобильного приложение, но вот обращение со смарт-зеркалом идет посредством касания. Можно себе представить внешний вид этого зеркала через пару часов после использования, а за ту цену, что предлагает представитель вряд ли кто-то захочет купить его в домашнее использование. Красиво только на картинке.

Более интересный вариант готова предложить немецкая компания. Ее продукт Perseus [2] оснащен QuadCore-процессором с частотой 2 ГГц, камерой, динамиком, микрофоном и, безусловно, Wi-Fi и Bluetooth. Данный продукт заинтересовал много инвестеров на платформе Kickstarter. Производители уверяют, что зеркало будет обладать водостойкой поверхностью. Данное устройство способно помимо вывода контентной информации персонализироваться при помощи распознавания голоса под разных пользователей. В планах у компании добавить возможность ведения видеочата. Все это звучит заманчиво, да и стоит дешевле, нежели предыдущий вариант.

Не отстают в данном вопросе и разработчики китайского рынка. В отличие от подобных продуктов китайский вариант компании Lenovo выглядит просто, однако оснащен машинным зрением Intel RealSense, что придает устройству определенную изюминку [3]. Благодаря этому зеркало способно выставлять свои настройки после распознавания конкретного человека, тем самым мгновенно быть универсальным. Неплохое начало и амбициозные цели впереди.

Ну и нельзя забывать о самодельных вариантах устройств. Да, на первый взгляд это может показаться хлопотным и затратным. Стоят ли усилия результата решать каждому. Например, сотрудник компании Google с помощью подручных средств решил повторить нашумевшое «умное» зеркало[4]. Пока пользователь может увидеть лишь время, дату, прогноз погоды и заголовки последних новостей, но разработчик не собирается останавливаться на этом.

1.2 Микроконтроллеры

Микроконтроллеры решают множество задач в современном мире, будь то управление автомобилем, исследование погоды, управление своим «умным домом» или роботом. Возможностей применения микроконтроллеров неимоверное множество.

Микроконтроллер [5] является небольшим компьютером, выполненным в виде небольшой микросхемы, в которой на одном «кристалле» содержатся все основные компоненты: процессор, периферия, устройства ввода-вывода, а также, чаще всего, оперативная память (ОЗУ) и энергонезависимая память (ПЗУ). Мощность такого компьютера совсем небольшая и не сравнится с мощностью настольного или портативного компьютера, но для, относительно, простых задач, для которых, обычно, и применяют микроконтроллеры их мощности предостаточно. Основным же плюсом использования одного микроконтроллера, в котором интегрированы все необходимые компоненты, вместо россыпи отдельных микросхем (процессор, ОЗУ, ПЗУ, периферия), является снижение стоимости, размеров, энергопотребления, а также затрат на разработку и сборку необходимого устройства.

Сегодня на рынке существует множество фирм-производителей, выпускающих различные микроконтроллеры. Рассмотрим несколько из них.

1.2.1 Семейства AVR

В настоящее время в серийном производстве у Atmel находятся три семейства AVR – «tiny», «classic» и «mega» [6]. Многие разработчики уже по достоинству оценили высокую скорость работы и мощную систему команд AVR, наличие двух типов энергонезависимой памяти на одном кристалле и активно развивающуюся периферию. AVR, пожалуй, одно из самых интересных направлений, развиваемых корпорацией Atmel. Они представляют собой мощный инструмент для создания современных высокопроизводительных и экономичных многоцелевых контроллеров. На настоящий момент соотношение "цена-производительность-энергопотребление" для AVR является одним из лучших на мировом рынке 8-разрядных микроконтроллеров. Области применения AVR многогранны.

В данной линии микроконтроллеров постоянно появляются новые кристаллы, обновляются версии уже существующих микросхем, расширяется программное обеспечение поддержки. Последнему пункту компания уделяет не малое внимание. Программные и аппаратные средства всегда разрабатывались и разрабатываются параллельно с самими микроконтроллерами. Несмотря на то, что компания Atmel уже имеет ряд сбалансированных по функциональности, экономичности, производительности и стоимости микроконтроллеров AVR, она не собирается останавливаться на достигнутом и продолжает развитие этой платформы.

Для семейства «tiny» (см. рисунок 1.1) – это интеллектуальные автомобильные датчики различного назначения, игрушки, игровые приставки, материнские платы персональных компьютеров, контроллеры защиты доступа в мобильных телефонах, зарядные устройства, детекторы дыма и пламени, бытовая техника, разнообразные инфракрасные пульты дистанционного управления.

Рисунок 1.1 – Микроконтроллер Attiny13

Для семейства «classic» – это модемы различных типов, современные зарядные устройства, изделия класса Smart Cards и устройства чтения для них, спутниковые навигационные системы для определения местоположения автомобилей на трассе, сложная бытовая техника, пульты дистанционного управления, сетевые карты, материнские платы компьютеров, сотовые телефоны нового поколения, а также различные и разнообразные промышленные системы контроля и управления.

Для «mega» AVR (см. рисунок 1.2) – это аналоговые (NMT, ETACS, AMPS) и цифровые (GSM, CDMA) мобильные телефоны, принтеры и ключевые контроллеры для них, контроллеры аппаратов факсимильной связи и ксероксов, контроллеры современных дисковых накопителей.

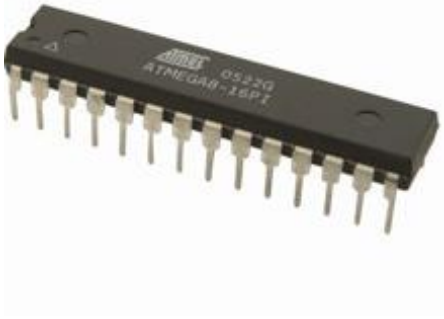

Рисунок 1.2 – Микроконтроллер Atmega8

Стоит отметить главную особенность всех вышеперечисленных устройств: все они имеют единую архитектуру, и это позволяет с легкостью переносить код с одного микроконтроллера на другой.

1.2.2 Процессорная платформа MSP430

Появившиеся в последнее время новые процессорные платформы MSP430 фирмы Texas Instruments (см. рисунок 1.3) и XE8000 фирмы Xemics также заслуживают внимания, особенно если основным критерием для конечного приложения является минимальное энергопотребление.

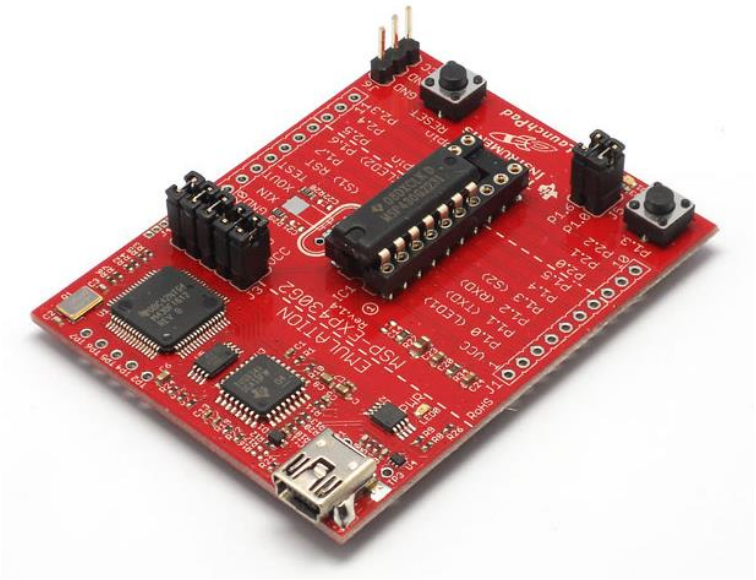

Рисунок 1.3 – Процессорная платформа MSP430G2

Сегодня микроконтроллеры со сверхнизким потреблением MSP430 — это всемирноизвестная и популярная серия устройств, включающая более 100 различных модификаций, имеющих Flash-память от 1 до 128 кбайт и количество выводов от 14 до 100, что позволяет подобрать идеальное решение для различных приложений [7]. Эффективная 16-разрядная RISC-архитектура процессоров MSP430 позволяет значительно уменьшить размер кода и повысить эффективность обработки сигналов по сравнению с современными 8-разрядными MCU.

1.2.3 Микроконтроллеры Arduino

Еще можно обратить внимание на микроконтроллеры Arduino [8]. На них нет операционной системы, как на Raspberry Pi, они не сложны в изучении и подойдут как для новичков, так и для более продвинутых пользователей.

Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройство программируется через USB без использования программаторов.

Основные преимущества данной платы:

– кроссплатформенность;
– простая среда программирования;
– открытый исходный код;
– открытые спецификации и схемы оборудования.

Наиболее распространенные версии плат: Uno, Leonardo, Nano, Mini, Mega2560 (см. рисунок 1.4).

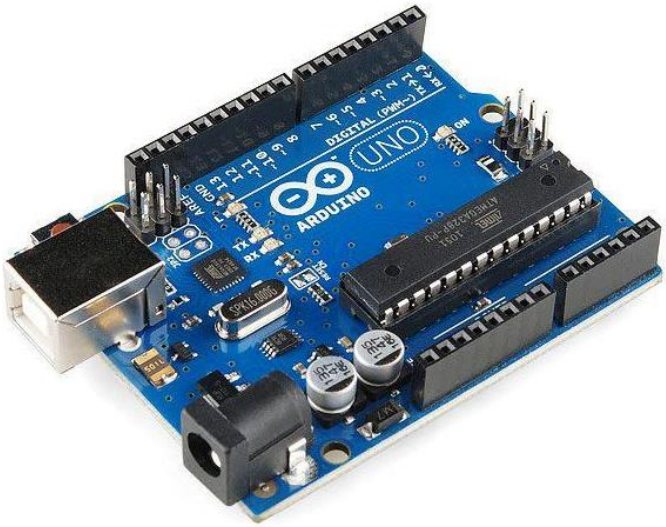

Рисунок 1.4 – Микроконтроллер Arduino Uno

1.2.4 Мини компьютеры Raspberry Pi, Banana Pi, Orange Pi

Так же на рынке представлены мини компьютеры. Наиболее популярный это Raspberry Pi и его аналоги, такие как Banana Pi и Orange Pi.

Raspberry Pi [9] – это компьютер, умещающийся в ладони. Разработано два варианта плат Raspberry Pi: «модель A» и «модель B». Модель «B» отличается наличием порта Ethernet и второго USB, у модели «A» возможности подключиться к сети не будет и USB всего один.

Banana Pi и Orange Pi позиционируются как более современные и мощные аналоги Raspberry Pi. Устройства комплектуются двухядерным процессором, более мощным графическим ускорителем, портом SATA, micro-USB и гигабитным портом Ethernet.

1.3 Использование микроконтроллеров

Чтобы использовать микроконтроллер его необходимо прошить соответствующей программой. Обычно этот процесс состоит из нескольких пунктов:

1. Определение задач, которые будет исполнять микроконтроллер.
2. Создание схемы на основе микроконтроллера или, как бывает чаще, поиск нужной схемы в интернете.
3. Написание программы-прошивки для микроконтроллера.

4. Прошивка программы в микроконтроллер.
5. Сборка и подключение всего устройства.
6. Использование самодельного гаджета.

Для того чтобы прошить микроконтроллер его необходимо подключить к ПК, для чего используется специальное устройство, которое называется программатор. С его помощью и осуществляется взаимосвязь между микроконтроллером и компьютером. Можно даже сказать, что программатор — это своеобразный мост. Так же в качестве среды разработки можно выбрать дополнение к Visual Studio или программу Fritzing. Последняя программа позволяет не только прошивать плату, но и разрабатывать проекты, строить принципиальные схемы и даже печатные платы.

1.3.1 Программа Fritzing

Работа с новым проектом в пакете Fritzing начинается с выбора готовых компонентов (см. рисунок 1.5). Здесь можно найти различные макетные и монтажные платы (в том числе Raspberry Pi), целый набор аналоговых и цифровых микросхем, любые радиодетали: конденсаторы, транзисторы, резисторы, светодиоды, батарейки, кнопки. Дополнительно присутствует большая коллекция устройств для робототехники: моторы, дальномеры, динамики, сервоприводы, шаговые двигатели, LCD и цифровые индикаторы, а также многое другое. Также можно создавать собственные элементы и обновлять существующую базу. Схема доступна для рисования, как в окне «Макетная плата», так и в окне «Принципиальная схема» простым перетаскиванием нужных компонентов на рабочее поле. В наличие есть функция автотрассировки. При выборе окна «Печатная плата» можно приступить к разводке проводников и размещению элементов. Результат работы экспортируется в pdf-файл для распечатки на лазерном принтере с дальнейшим изготовлением платы методом нанесения рисунка на фольгированный текстолит горячим утюгом.

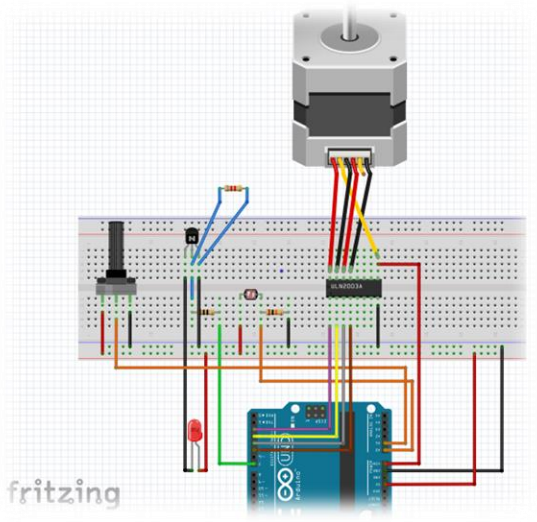

Рисунок 1.5 – Fritzing

К сожалению, самостоятельно разводить печатные платы Fritzing не может, так же, как и симулировать работу схемы. Рисование чересчур упрощено, а элементная база крайне мала. Красивые, ярко-красочные, детальные схемы больше подходят для фотоотчетов и быстрых набросок, объясняющих принципы работы того или иного устройства.

1.4 Интерфейсный модуль

В качестве среды разработки для интерфейсного модуля была выбрана технология Microsoft .NET, в частности ее часть ASP.NET MVC. Данная среда содержит большинство компонентов, необходимых для построения приложения, а также для подключения и работы с базой данных. Разработка модуля осуществляется в среде Microsoft Visual Studio 2017.

При первоначальном знакомстве с языком программирования C# полезно изучить базовые конструкции и общие правила языка [10], а также получить представление о принципах объектно-ориентированного программирования.

Далее, для более полного понимания всего, что будет происходить с программным кодом, необходимо углубиться в архитектуру и устройство библиотеки .NET Framework [11, 12]. Библиотека .NET Framework состоит из двух частей: общеязыковой исполняющей среды (Common Language Runtime, CLR) и библиотеки классов Framework Class Library (FCL).

Программа на языке C# выполняется в среде .NET Framework – интегрированном компоненте Windows, содержащем виртуальную систему выполнения (среда CLR) и унифицированный набор библиотек классов. Среда CLR представляет собой коммерческую реализацию корпорацией Майкрософт инфраструктуры CLI, которая является международным стандартом, лежащим в основе создания сред выполнения и разработки, в которых обеспечивается тесное взаимодействие между языками и библиотеками.

Исходный код, написанный на языке C#, компилируется в промежуточный язык (IL) в соответствии со спецификацией CLI. Код IL и ресурсы, такие как растровые изображения и строки, хранятся на диске в исполняемом файле, называемом сборкой, с расширением EXE или DLL в большинстве случаев. Сборка содержит манифест со сведениями о типах сборки, версии, языке и региональных параметрах и требованиях безопасности.

При выполнении программы на C# сборка загружается в среду CLR в зависимости от сведений в манифесте. Далее, если требования безопасности соблюдены, среда CLR выполняет JIT-компиляцию для преобразования кода IL в инструкции машинного кода. Среда CLR также предоставляет другие службы, относящиеся к автоматическому сбору мусора, обработке исключений и управлению ресурсами. Код, выполняемый средой CLR, иногда называют «управляемым кодом» в противопоставление

«неуправляемому коду», который компилируется в машинный код, предназначенный для определенной системы. На рисунке 1.6 показаны отношения во время компиляции и время выполнения между файлами с исходным кодом C#, библиотеками классов .NET Framework, сборками и средой CLR.

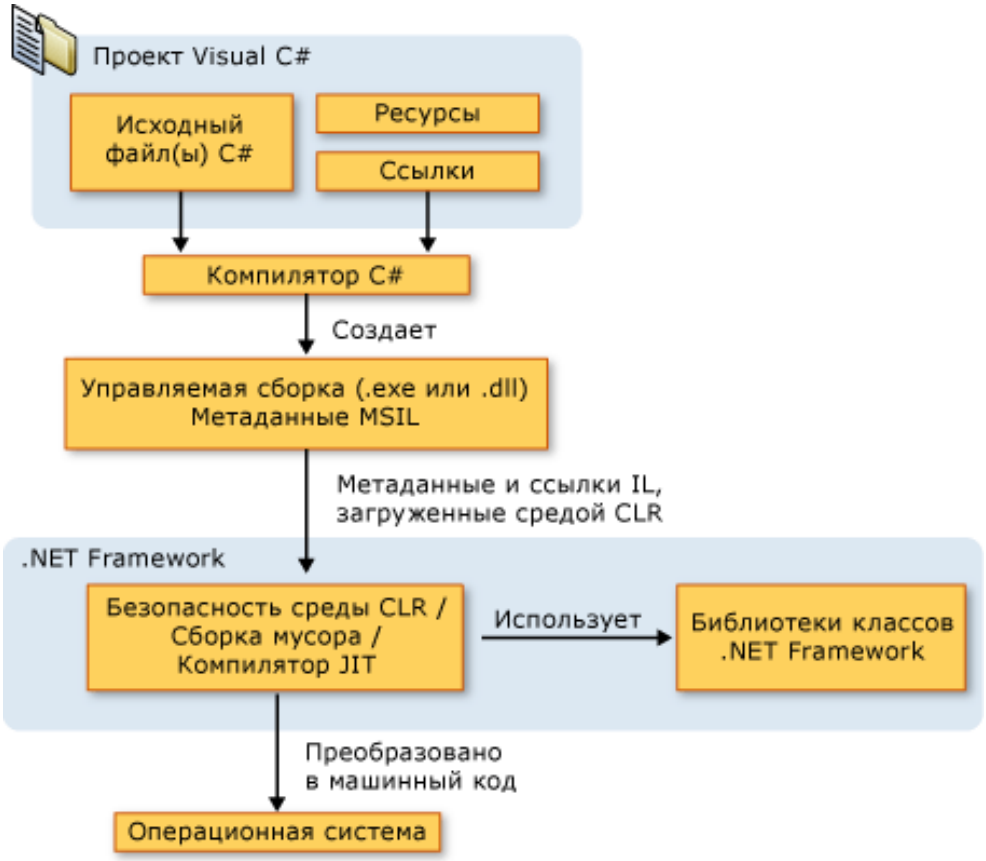

Рисунок 1.6 – Отношения во время компиляции и время выполнения между файлами с исходным кодом C#, библиотеками классов .NET Framework, сборками и средой CLR

Взаимодействие между языками является ключевой особенностью .NET Framework. Поскольку код IL, создаваемый компилятором C# соответствует спецификации CTS, код IL на основе C# может взаимодействовать с кодом, создаваемым версиями языков Visual Basic, Visual C++, Visual J# платформы .NET Framework и еще более чем 20 CTS-совместимых языков. В одной сборке может быть несколько модулей, написанных на разных языках платформы .NET Framework, и типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

Помимо служб времени выполнения, в .NET Framework также имеется обширная библиотека, состоящая из более чем 4000 классов, организованных по пространствам имен, которые обеспечивают разнообразные полезные функции для любых действий, начиная от ввода и вывода файлов для управлением строками для разбивки XML, и заканчивая элементами

управления Windows Forms. В обычном приложении на языке C# библиотека классов .NET Framework интенсивно используется для «устройства» кода.

При разработке web-приложения использована технология Microsoft .NET, в частности ее часть ASP.NET, предоставляющая возможность быстрой и эффективной разработки клиент-серверных интернет приложений. Для обеспечения реализации шаблона проектирования MVC необходимо использовать фреймворк MVC3 [13].

1.4.1 ASP.NET MVC Framework

ASP.NET MVC Framework — фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller (MVC). Схема модели шаблона MVC приведена на рисунке 1.7.

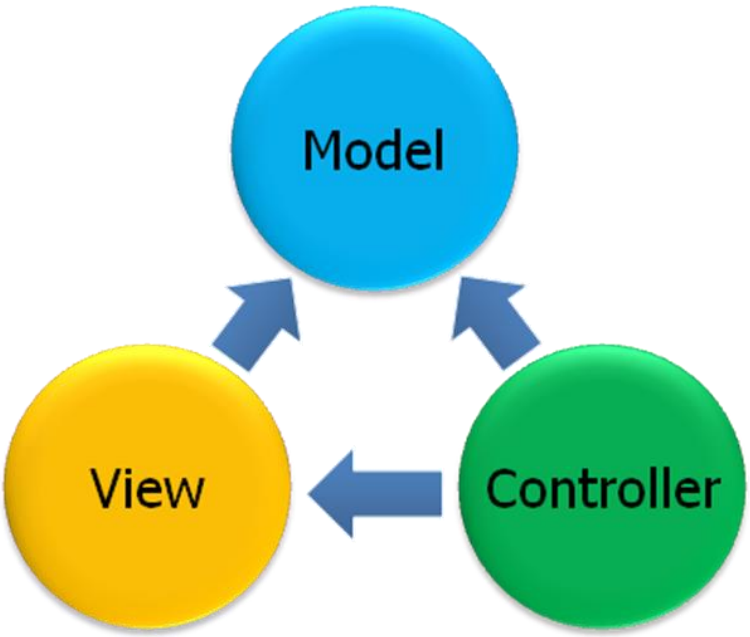

Рисунок 1.7 – Схема модели шаблона MVC

Model – модель данных. Часть приложения, содержащая в себе бизнес-логику, она описывает какую-то предметную область. Модель знает, как устроены данные и как с ними работать, она понятия не имеет о контроллере и представлении. Основная цель паттерна сделать так, чтобы модель ничего не знала об остальных частях приложения, что позволит без проблем менять представление или контроллер этой модели.

View – представление. Это та часть, которая визуализирует модель данных. Обычно представление имеет прямой доступ к модели, но разрешено только чтение этих данных. Для работы с контроллером реализует в себе некий интерфейс, который известен контроллеру, а представление умет с ним работать.

Controller – контроллер. Компонент, немного, знающий о представлениях и что есть модель. Из названия становится понятно, что это

то, что контролирует остальные части программы, он реагирует на какие-то события и, исходя из этого, влияет на модель или представление.

Маршрутизация ASP.NET позволяет использовать URL-адреса, не сопоставляемые с определенными файлами на веб-узле. Поскольку URL-адрес не сопоставляется с файлом, можно использовать в веб-приложении URL-адреса, описывающие действия пользователя, вследствие чего они более понятны пользователям. В приложении ASP.NET, не использующем маршрутизацию, входящий запрос URL-адреса обычно сопоставляется физическому файлу на диске, например ASPX-файлу.

Для упрощения использования фреймворка MVC 3 и для некоторой автоматизации необходимо использовать IoC-контейнер. Инверсия управления (Inversion of Control, IoC) — важный принцип объектно-ориентированного программирования, используемый для уменьшения связанности в компьютерных программах. Суть паттерна состоит в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов. Внедрение зависимостей (dependency injection) — это одна из реализаций этого принципа.

IoC-контейнер — это какая-то библиотека, фреймворк, который позволит упростить и автоматизировать написание кода с использованием данного подхода на столько, на сколько это возможно. Существует множество IoC-контейнеров, но, на мой взгляд, быстрейшим и самым удобным в использовании является Windsor Castle [14].

1.4.2 СУБД Microsoft SQL Server 2014

Для хранения данных была выбрана СУБД Microsoft SQL Server 2014. Microsoft SQL Server является надежной СУБД, не требующей высокопроизводительных компьютеров.

SQL Server является всеобъемлющим, интегрированным сквозным решением, которое наделяет пользователей организации безопасной, надежной, и продуктивной платформой для обработки промышленной информации и приложений, касающихся интеллектуальных ресурсов предприятия.

Благодаря исчерпывающему набору функций, взаимодействию с существующими системами и автоматизации типовых задач, SQL Server 2008 предоставляет полное решение в области хранения данных для предприятий всех масштабов.

Платформа данных SQL Server включает следующие инструменты:

1. Реляционная база данных: безопасное, надёжное, масштабируемое, высокодоступное ядро с улучшенной производительностью и поддержкой структурированных и неструктурированных (XML) данных.
2. Replication services: репликация данных для распределённых и мобильных приложений обработки данных, высокая доступность систем, масштабируемый параллелизм с вторичными хранилищами данных для

отчётных решений предприятия и интеграция с разнородными системами, включая существующие базы данных Oracle.

3. Notification services: развитые возможности уведомлений для разработки и внедрения масштабируемых приложений, способных доставлять персонализированные, своевременные обновления информации множеству соединённых и мобильных устройств.

4. Integration services: возможности извлечения, преобразования и загрузки для хранилищ данных и интеграции данных в масштабе предприятия.

5. Analysis services: аналитическая обработка в реальном времени (OLAP) для быстрого, сложного анализа больших и смешанных наборов данных, использующая многомерное хранение.

6. Reporting services: исчерпывающее решение для создания, управления и доставки как традиционных бумажных отчётов, так и интерактивных, основанных на технологии WWW отчётов.

7. Инструменты управления: SQL Server включает средства управления для развитого управления и настройки баз данных, также, как и тесную интеграцию с такими инструментами, как Microsoft Operations Manager (MOM) и Microsoft Systems Management Server (SMS). Стандартные протоколы доступа к данным существенно уменьшают время, необходимое для интеграции данных SQL Server с существующими системами. В дополнение, поддержка Web служб встроена для обеспечения взаимодействия с другими приложениями и платформами.

8. Инструменты разработки: SQL Server предлагает интегрированные инструменты разработки для ядра базы данных, извлечения, трансформации и загрузки данных, извлечения информации. Каждая главная подсистема SQL Server поставляется со своей собственной объектной моделью и набором API для расширения системы данных в любом направлении, которое уникально для бизнеса.

1.4.3 ORM, графический движок Razor, Bootstrap

В современных Web-приложениях для реализации доступа к данным используются ORM технологии. ORM (англ. Object-relational mapping, рус. Объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии. Главная задача ORM(Object-Relational Mapping) заключается в том, чтобы являться тем связующим звеном, которое прозрачно для нас будет делать всю работу по сохранению данных наших бизнес-объектов и заполнению их даными из базы при необходимости. К тому же ORM избавляет от необходимости большого количества однотипного примитивного кода выполняющего всего лишь CRUD-операции(Create, Read,

Update, Delete). Из ряда ORM технологий можно выделить Entity Framework[15].

Для моделирования пользовательского интерфейса был выбран графический движок Razor [16]. Razor уменьшает «нагрузку на клавиатуру», требуемую для работы в файле, предоставляет быстрый и гибкий процесс программирования. В отличие от большинства синтаксиса шаблонов, не нужно прерывать написание кода, чтобы явно определить серверные блоки кода в HTML. Парсер достаточно умен для того чтобы сделать это самостоятельно, основываясь на коде. В итоге получается компактный и выразительный синтаксис, который легко набирать. Razor не требует определенного инструмента и позволяет быть продуктивным в любом старом текстовом редакторе. Не смотря на то, что Razor разрабатывался с учетом отсутствия привязки к определенному инструменту или редактору кода, у него будет отличная поддержка в Visual Studio.

Для верстки клиентской части используется набор инструментов для создания веб-сайтов Bootstrap [17]. Bootstrap – это фреймворк от разработчиков Twitter – Марка Отто и Якоба Торнтона, изначально планировавшийся как внутренний продукт, предназначенный для поддержки единообразия внутренних инструментов. Сегодня Twitter Bootstrap является одним из самых популярных html/css фреймворков.

Этот инструмент позволяет легко и быстро сверстать несложный сайт, обладая минимальными знаниями в области верстки html-страниц. По сути, Bootstrap содержит готовые стили для оформления основных элементов html, в том числе кнопок, таблиц, графиков, картинок и видео, форм, элементов навигации, алертов, стили для заголовков, списков, цитат, определений, кодов, а также самые базовые и часто используемые динамические элементы на javascript, такие как несложный слайдер, выпадающие списки, вложенные меню, лайтбокс, пагинацию и так далее. Этот фреймворк позволяет создавать первоначальную структуру и стилизировать на базовом уровне большинство элементов.

1.5 Аналитический обзор

Для проектирования «Смарт-зеркало с голосовым управлением» был выбран мини компьютер Raspberry Pi, исходя из вышеперечисленных его достоинств и так как он является более оптимальным решением для проекта, среда разработки Fritzing, содержащая необходимые для построения схемы компоненты, а также для прошивки самой платы. В качестве среды разработки для интерфейсного модуля была выбрана технология Microsoft .NET, в частности ее часть ASP.NET MVC. Для хранения данных была выбрана СУБД Microsoft SQL Server 2014. Для моделирования пользовательского интерфейса был выбран графический движок Razor. Для верстки клиентской части используется набор инструментов для создания веб-сайтов Bootstrap.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

2.1 Определение структуры комплекса

Целью проекта является разработка системы визуализации информационного контента и управление им с помощью голосовых команд. Основные требования, предъявляемые системе следующие:

1. Необходим дисплей для вывода информации и само зеркало.
2. Необходим контроллер для управления диплеем.
3. Необходимы светодиоды, которые будут обеспечивать необходимую функциональность освещения при подаче голосовых команд либо с автоматическим запуском при распознавании образов.
4. Необходим датчик для принятия и обработки голосовых команд, т.е. микрофон.
5. Необходим датчик для получения и дальнейшего распознавания образов, т.к. камера.
6. Необходим интерфейс для управления и настройки системы.
7. Необходима база данных, для сохранения пользовательских настроек.
8. Необходим управляющий блок, который будет управлять всей системой.

Из данных требований можно сделать вывод о необходимых элементах структурной схемы.

Рассмотрим каждый блок отдельно.

2.2 Блок визуализации

Блок визуализации представляет собой зеркало и дисплей. Во многих детективных фильмах можно увидеть одностороннее зеркало или зеркало Гезела. Находящиеся в комнате не могут увидеть, что за ними наблюдают, они видят лишь свое отражение. В то же самое время снаружи посетители могут без труда все рассмотреть. Для того, чтобы добиться такого эффекта необходимо чтобы стекло было прозрачным, могло пропускать свет. В то же самое время оно должно обладать зеркальными свойствами. Часть света, падающего на стекло отражается передней и задней поверхностями стекла, часть – проходит через стекло и часть поглощается, нагревая стекло и затем излучаясь в инфракрасном спектре. Количество света, проходящего через стекло определяется коэффициентом пропускания света. Количество отражаемого света определяется коэффициентом отражения.

Принцип действия зеркального стекла в том, что затемненный интерьер не виден на фоне яркого отражения. Полупрозрачных зеркал, которые пропускали бы свет в одну сторону и не пропускали во вторую, не существует. Зеркало Гезела в физическом смысле (как конструкция односторонней видимости, которая не зависит ни от освещенности, ни от других условий работы) – противоречит физическим законам. Используется

это понятие в основном в психологии для обсуждения морально-этических вопросов.

Рассмотрим вышеуказанное на примере улицы и помещения. Зритель в помещении будет отчетливо видеть улицу до тех пор пока разница освещенностей не станет меньше 0,1. Это значит, что уже через некоторое время после заката улица будет видна не слишком хорошо. В ночное время зрители в помещении не увидят ничего. Для того же, чтобы быть скрытыми от глаз наблюдателей на улице необходимо, чтобы разница освещенностей была не менее двух. Поэтому видеть снаружи помещение можно будет поздно в сумерки, когда на улице будет лишь немного светлее, чем внутри.

Для того, чтобы избежать этого, необходимо увеличить освещенность снаружи хотя бы в 2 раза по сравнению с помещением. Это обозначает, что перед стеклом с односторонней прозрачностью может понадобиться установить лампы, которые будут создавать в ночное время освещение ярче, чем внутри помещения.

Обычные зеркала – это как правило стекло с нанесенным на его заднюю сторону отражающим покрытием (различными сплавами олова, серебра, меди, титана или алюминия). При этом напыление очень плотное и слой напыления толстый. Зеркала с односторонней прозрачностью изготавливаются аналогично, но слой напыления тонкий и пропускает часть света. Альтертнативный вариант – нанесение полупрозрачной зеркальной пленки. Ее можно наносить на существующее остекление. Именно этот подход будет осуществляться при разработке устройства.

За стеклом с зеркальной пленкой будет находиться монитор на который и будет выводиться информация для пользователя. Задняя стенка зеркала будет заклеена темной непроницаемой пленкой для того, чтобы по другую сторону зеркала свет не попадал, а значит устройство будет работать даже в темное время суток.

Как уже было сказано, со стороны это будет выглядеть как обычная информация на зеркале.

2.3 Блок управления визуализацией

Блок управления визуализацией необходим для управления монитором, на который будет выводиться информация. Он будет связующим звеном между блоком визуализации, который представляет собой монитор, и главным управляющим блоком.

2.4 Блок освещения

Блок освещения необходим для освещения данной конструкции. Варианты использования приведены ниже:

1. Устройство разпознает образ человека и автоматически включает подсветку.

2. Устройство распознает голосовую команду, дающую указание включить освещение.

2.5 Блок обработки голосовых команд

Блок обработки голосовых команд будет заниматься прослушиванием и распознаванием речи с целью реакции на заранее заданную фразу. При этом все другие звуки и речь будет отбрасываться. Для активации фразы понадобится ключевая команда, которая будет сигналом для распознавания дальнейшей команды. Это дает нам возможность реагировать только на определенные команды и избежать ложных срабатываний. Для реализации возможности голосового управления в корпус устройства будет встроен микрофон.

2.6 Блок распознавания образов

Блок распознавания образов необходим для автоматического включения освещения. Распознавание образа производится на основе распознования лица человека. В случае обнаружения необходимого образа, образ должен находиться в зоне обнаружения в течении определенного периодна времени для избежания ложных срабатываний, будет осуществляться подача сигнала на управляющий модуль. Управляющий модуль принимает решение о включении освещения.

2.7 Блок управления интерфейсом

Для настройки устройства под конкретного пользователя необходим удобный интерфейс. Блок управления интрфейсом будет представлять собой веб-приложение доступное по локальной сети. Пользователь всегда сможет зайти в приложение, выбрать какие именно виджеты отображать на зеркале, на какую фразу должно реагировать устройство, с какими учетными записями необходимо синхронизироваться. Система виджетов будет представлять собой набор инструментов для отображения интересующей пользователя информации, а также возможность управления устройством. Например, задавать управляющую фразу, изменять настройки отображения и прочее.

2.8 База данных

База данных необходима для сохранения пользовательских настроек и синхронизации их между различными устройствами с общей учетной записью. Также база данных будет хранить некоторую информацию, которую пользователь предпочитает видеть чаще всего. Например, если ежедневно на экран выводится состояние погоды, то устройство будет зарание подготавливать необходимую информацию чтобы избежать задержек при поиске информации в интернете. Также в базе данных будут храниться

управляющие фразы, которые соответствуют определенным командам на программном уровне.

2.9 Управляющий блок

Основной задачей данного блока является управление всеми вышеперечисленных блоками: обработка приходящих от них сигналов, принятие решения об изменении состояния системы и само изменение состояний проектируемого объекта.

Управляющий блок взаимодействует напрямую практически со всеми остальными блоками, т.к. является основным вычислительным модулем.

На корпусе управляющего блока будет находиться также модуль беспроводного соединения, благодаря которому устройство может быть управляемо по локальной сети с мобильного телефона или компьютера.

Управляющий блок будет оснащен картой памяти, которая будет кэшировать информацию, приходящую из базы данных, находящейся вне самого устройства.

Также интересной особенностью может являться подключение к устройству напрямую к управляющему блоку для перепрошивки и других настроек. Тем не менее это будет работать и по локальной сети.

2.10 Взаимодействие блоков

Система функционирует следующим образом. Блок обработки голосовых команд получает звуковые сигналы и передает их на управляющий блок. Управляющий блок делает вывод о контрольной фразе для срабатывания, и если она совпала, выполняет заданную команду. Он может послать сигнал на блок освещения или на блок визуализации через связующий управляющий блок. Управляющий блок подает сигналы на обновление информации на мониторе, получает по сети новые порции данных для отображения, обрабатывает запросы от блока распознавания образов.

Также управляющий блок может взаимодействовать с другими блоками и без получения специальных команд от пользователя. Например, если он выполняет операции, которые настроены запускаться по расписанию. Примером такой операции может послужить получение информации о погоде и расписании транспорта за некоторое время до того, как пользователь захочет ей воспользоваться. Такой подход значительно сократит время ожидания результата и по осущениям пользователя результат будет получен мгновенно.

3 РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ СХЕМЫ

3.1 Программная часть

В качестве нотации для логического моделирования проекта был выбран унифицированный язык моделирования UML (англ. Unified Modeling Language) — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем. UML не является языком программирования, но в средствах выполнения UML-моделей как интерпретируемого кода возможна кодогенерация.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. В настоящее время идет процесс представления UML в качестве стандарта ISO. UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (generalization), объединение (aggregation) и поведение), и больше сконцентрироваться на проектировании и архитектуре.

Преимущества UML:

1. UML объектно-ориентированный, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных ОО-языках.
2. UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы.
3. диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом.
4. UML расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии.
5. UML получил широкое распространение и динамично развивается.

Несмотря на то, что UML достаточно широко распространённый и используемый стандарт, его часто критикуют.

Для визуального моделирования в рамках дипломного проекта использовались специальные архитектурные инструменты Visual Studio 2017 и Visio от компании Microsoft Corporation.

3.1.1 Разработка диаграммы вариантов использования

Визуальное моделирование в UML можно представить, как некоторый процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует цели:

1. Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
2. Формулировать общие требования к функциональному поведению проектируемой системы.
3. Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
4. Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

Действующие лица могут иметь два типа связей с вариантами использования:

1. Простая ассоциация — отражается линией между актером и вариантом использования (без стрелки).
2. Направленная ассоциация — то же что и простая ассоциация, но показывает, что вариант использования инициализируется актером.

3.1.1.1 Действующие лица

Пользователь – авторизация, навигация по пунктам меню, управление профилем (добавление, редактирование, удаление информации), управление настройками виджетов и голосовых команд, взаимодействие с зеркалом

посредством голосовых команд, текстовой и звуковой информаций от зеркала в ответ на запрос.

Гость – регистрация, просмотр виджета часов.

База данных – хранение и отдача информации.

Зеркало – прием запрос от пользователя и отдача информации.

3.1.1.2 Варианты использования

Варианты использования приведены ниже:

1. Вход в систему.
2. Регистрация.
3. Аутентификация.
4. Редактирование базы данных: добавление информации, изменение информации, удаление информации.
5. Управление профилем.
6. Использование базы данных.
7. Управление настройками виджетов.
8. Управление настройками голосовых команд.
9. Просмотр информации предоставляемой зеркалом посредством виджетов.
10. Голосовые запросы к зеркалу.

3.1.1.3 Диаграмма вариантов использования

Действующими лицами являются гость, пользователь, база данных, веб-приложение, зеркало.

На рисунке 3.1 представлена диаграмма вариантов использования для данного дипломного проекта.

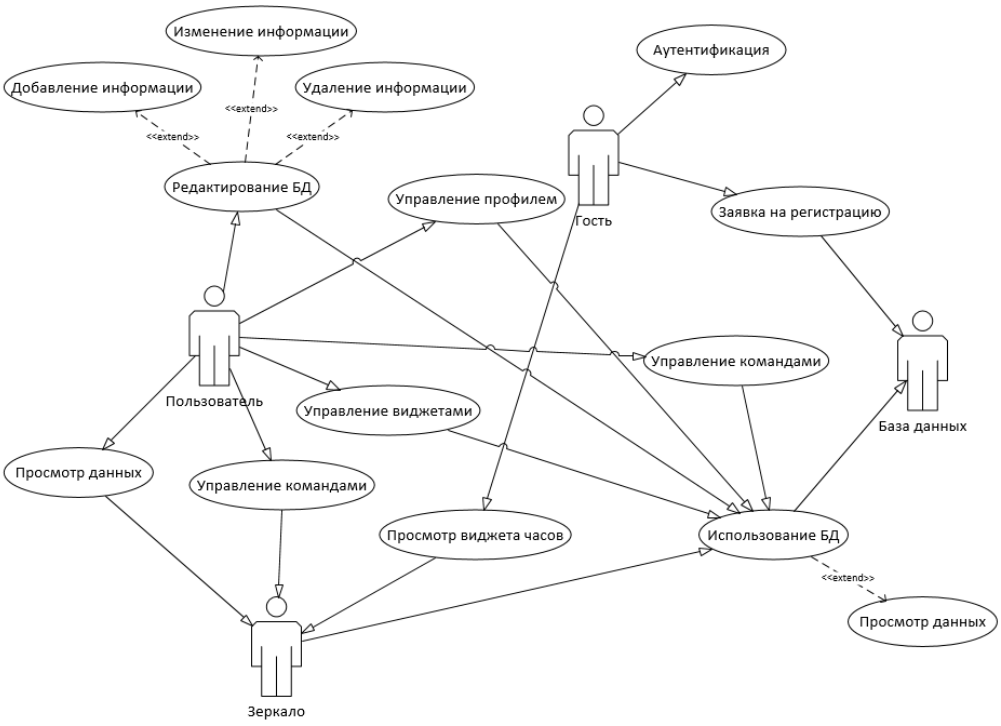

Рисунок 3.1 – Диаграмма вариантов использования

Как уже было сказано, вариант использования описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату.

Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать (именно что, а не как).

Действующее лицо является внешним источником (не элементом системы), который взаимодействует с системой через вариант использования. Действующие лица могут быть как реальными людьми (например, пользователями системы), так и другими компьютерными системами или внешними событиями.

Рассмотрим варианты использования, представленные на диаграмме.

Вариант использования «Заявка на регистрацию» описывает возможность посетителя зарегистрироваться и получить доступ к информационной системе на уровне пользователя. Для этого пользователь должен предоставить основную информацию о себе, а также ввести логин и пароль. В случае если пользователя с таким логином существует, пользователь получит уведомление о существовании такого логина в базе. Иначе информация о новом пользователе будет внесена в базу данных.

Вариант использования «Аутентификация» описывает вход пользователя на ресурс как зарегистрированного:

1. Система запрашивает имя пользователя и пароль.
2. Пользователь вводит имя и пароль.
3. Система проверяет введенные имя и пароль, после чего открывает доступ в систему в соответствии с ролью пользователя в ней.

Если во время выполнения основного потока обнаружится, что пользователь ввел неверное имя и/или пароль, то система выводит сообщение об отсутствии пользователя с такими логином и паролем. Пользователь может попробовать ввести данные еще раз или отказаться от входа в систему (при этом выполнение варианта использования завершается).

Вариант использования «Просмотр данных» позволяет просмотреть расположенную на сайте информацию. Навигация по информационным разделам осуществляется при помощи основного меню или поиска.

Вариант использования «Редактирование БД» представляет собой основной способ добавления, редактирования, удаления материалов расположенных на сайте, а также управления пользователями.

Управлять профилем могут только зарегистрированные пользователи. В состав функционала этой возможности входят такие операции как редактирование своих учетных данных.

Управлять настройкой виджетов и голосовыми командами может также только зарегистрированный пользователь. В состав функционала этой возможности входят такие операции как редактирование данных для виджетов, настройка их отображения, добавление и сокрытие виджетов для

зеркала, просмотр всевозможных виджетов и команд, выбор необходимых команд и сокрытие оставшихся.

3.1.2 Архитектура проекта

Архитектура проекта интерфейсного модуля (см. рисунок 3.2) содержит следующие узлы:

1. Клиент (данный блок может содержать несколько клиентов: клиент для работы с сервером, клиент администратора).
2. Уровень представления позволяет получать модели, необходимые для отображения на определенной html странице по запросу клиента, модель формируется на основании доменной модели посредством запроса к уровню Domain.
3. Доменый уровень средствами Entity Framework позволяет получать готовые объекты из базы данных.
4. SQL Server хранит базу данных проекта.

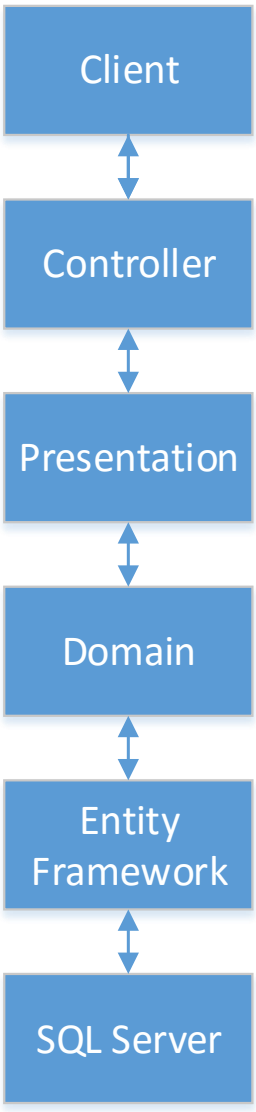

Рисунок 3.2 – Архитектура проекта интерфейсного модуля

Архитектура проекта для аппаратной части состоит из:

1. Клиент (данный блок содержит клиента по работе с api таких библиотек, которые предоставляют возможность работать с календарем, распознаванием эмоций, аудио проигрывателем, и др.).
2. Уровень контроллер, который отвечает за обработку входящих клиентских запросов и является основным уровнем приложения, т.е. содержит основную логику.
3. Уровень представления, который работает с моделями для отображения на xaml страницах.

Архитектура проекта работающего с аппаратной частью представлена на рисунке 3.3.

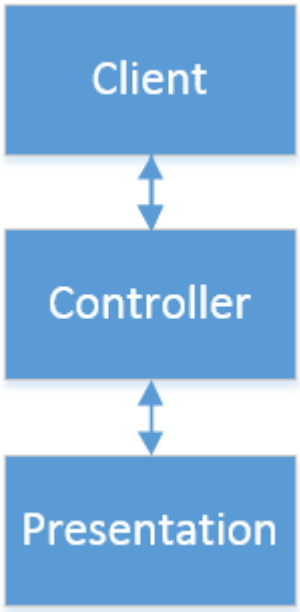

Рисунок 3.3 – Архитектура проекта аппаратного модуля

3.1.3 Идентификация классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Архитектура приложения интерфейсного модуля на самом верхнем уровне представляется основной диаграммой классов в виде набора пакетов, показанной на рисунке 3.4. На ней представлены все сборки и исполняемые файлы, используемые в проекте.

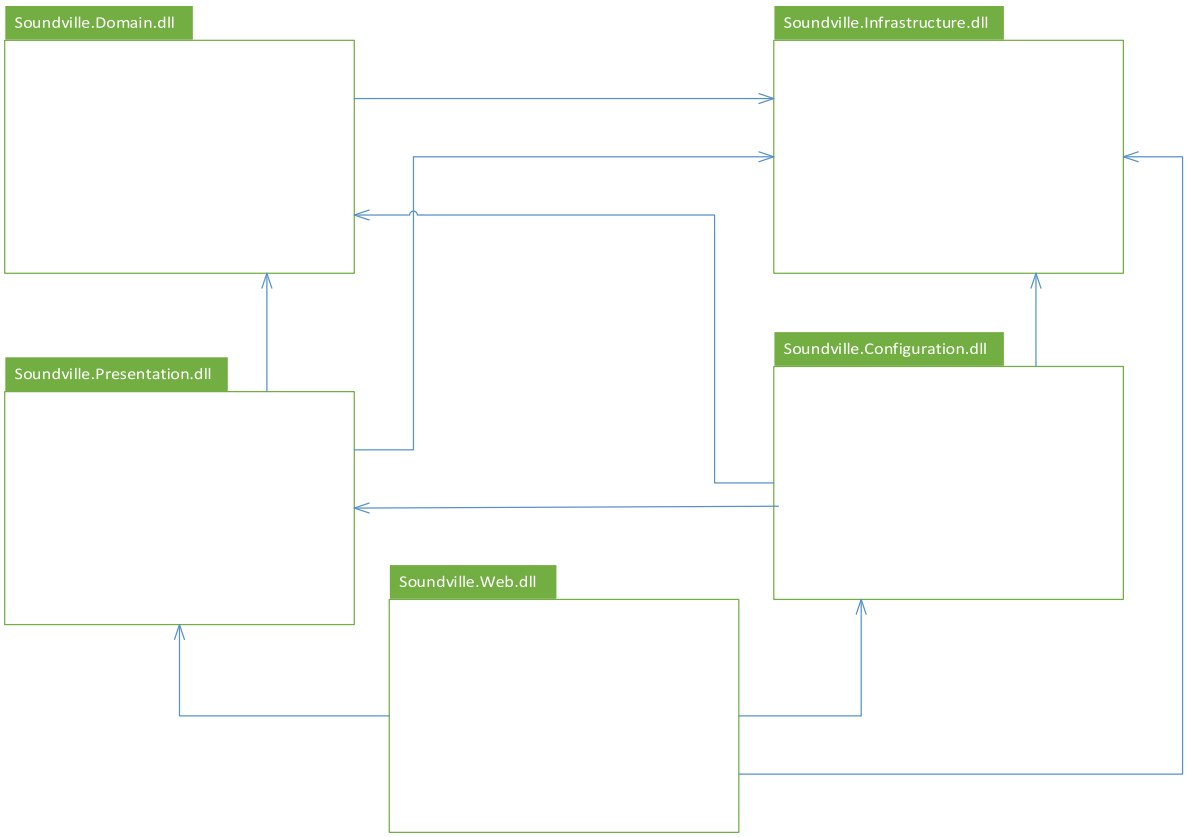

Рисунок 3.4 – Диаграмма классов в виде набора пакетов для интерфейсного модуля

Сборка `Soundville.Domain.dll` является уровнем доступа к данным (DAL), на этом уровне происходит обращение к базе данных, также этот уровень содержит модели доменного уровня, которые соответствуют таблицам в базе данных.

Сборка `Soundville.Infrastructure.dll` является вспомогательной. В ней содержатся расширения для классов и различные вспомогательные методы, необходимые в любой сборке.

Сборка `Soundville.Configuration.dll` также является вспомогательной и содержит классы, для внедрения зависимостей в проект с помощью IoC-контейнера Windsor Castle.

Сборка `Soundville.Presentation.dll` содержит классы, являющиеся связующим звеном между клиентом и сервером. Здесь происходит подготовка моделей представления.

Сборка `Soundville.Web.dll` содержит классы, предназначенные для управления веб-приложением. Кроме того здесь располагаются статические файлы такие как скрипты, стили и страницы.

На рисунке 3.5 представлены все сборки и исполняемые файлы, используемые в проекте для аппаратной части. Архитектура так же представлена на самом верхнем уровне диаграммой классов в виде набора пакетов.

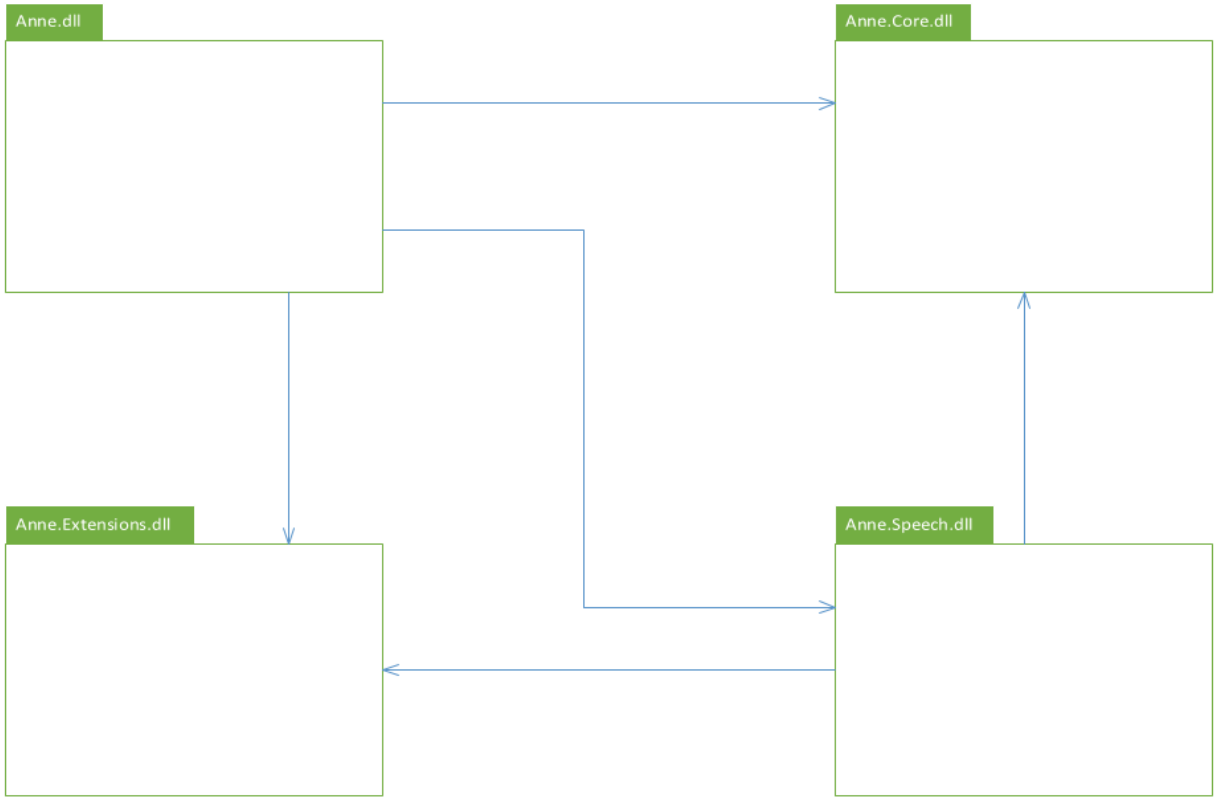

Рисунок 3.5 – Диаграмма классов в виде набора пакетов для аппаратного модуля

Сборка `Anne.dll` содержит классы, являющиеся связующим звеном для работы с api сторонних сервисов и связана с модулем для распознавания голосовых команд. Кроме того здесь располагаются статические файлы такие как скрипты, стили и страницы. Также здесь обрабатывается событие распознавания фразы и дальнейшая обработка пришедшей команды: вызов музыкального плеера, обработка видеопотока и распознавание эмоций, выполнение воспроизведения входящей фразы для системы.

Сборка `Anne.Core.dll` является вспомогательной и содержит классы необходимые для дебага, логирования и асинхронных вызовов.

Сборка `Anne.Extensions.dll` также является вспомогательной и содержит методы-расширения, необходимые для работы приложения и любой описанной сборки.

Сборка `Anne.Speech.dll` содержит классы, работающие с библиотекой для распознавания голосовых команд. Данная сборка работает с грамматикой в приложении, обрабатывает входящий аудиопоток пользователя, распознает фразы, преобразовывает их в команды, если это возможно.

Данные структуры примечательны тем, что позволяют заменить любой из пакетов на пакет, используемый другую технологию, но реализующий определенные в проекте интерфейсы. Система будет работать, как и раньше,

без видимых изменений. Другие слои даже не узнают о каких-либо изменениях в том или ином пакете.

3.1.4 Разработка сценариев

Одной из характерных особенностей систем различной природы и назначения является взаимодействие между собой отдельных элементов, из которых образованы эти системы. Речь идет о том, что различные составные элементы систем не существуют изолированно, а оказывают определенное влияние друг на друга, что и отличает систему как целостное образование от простой совокупности элементов.

В языке UML взаимодействие элементов рассматривается в информационном аспекте их коммуникации, т. е. взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений. Другими словами, хотя сообщение и имеет информационное содержание, оно приобретает дополнительное свойство оказывать направленное влияние на своего получателя.

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы последовательности. На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами.

3.1.5 Разработка базы данных

В данном проекте программа должна хранить информацию о пользователях, наборе виджетов для пользователя, сами виджеты и команды для управления.

При проектировании базы данных можно выделить ряд сущностей, таких как Users, CalendarWidget, MusicWidget, WeatherWidget, EmotionWidget, Commands, UserCommands.

Далее необходимо четко предствить взаимоотношения этих сущностей, на базе чего можно составить схему.

Рассмотрим связи, установившиеся между таблицами базы данных.

У зеркала возможно множество голосовых команд, а команды могут относиться к разным зеркалам. Таблицы Users и Commands имеют отношение многие ко многим, поэтому необходима таблица-связка UserCommands.

По данным сущностям построены таблицы. Имя сущности – название таблицы, атрибуты сущности – столбцы таблицы.

В итоге имеется семь таблиц, связанных между собой: таблицы Users и UserCommands, UserCommans и Commands, Users и CalendarWidget, Users и MusicWidget, Users и WeatherWidget, Users и EmotionWidget связаны отношением один ко многим.

На рисунке 3.6 представлена схема базы данных.

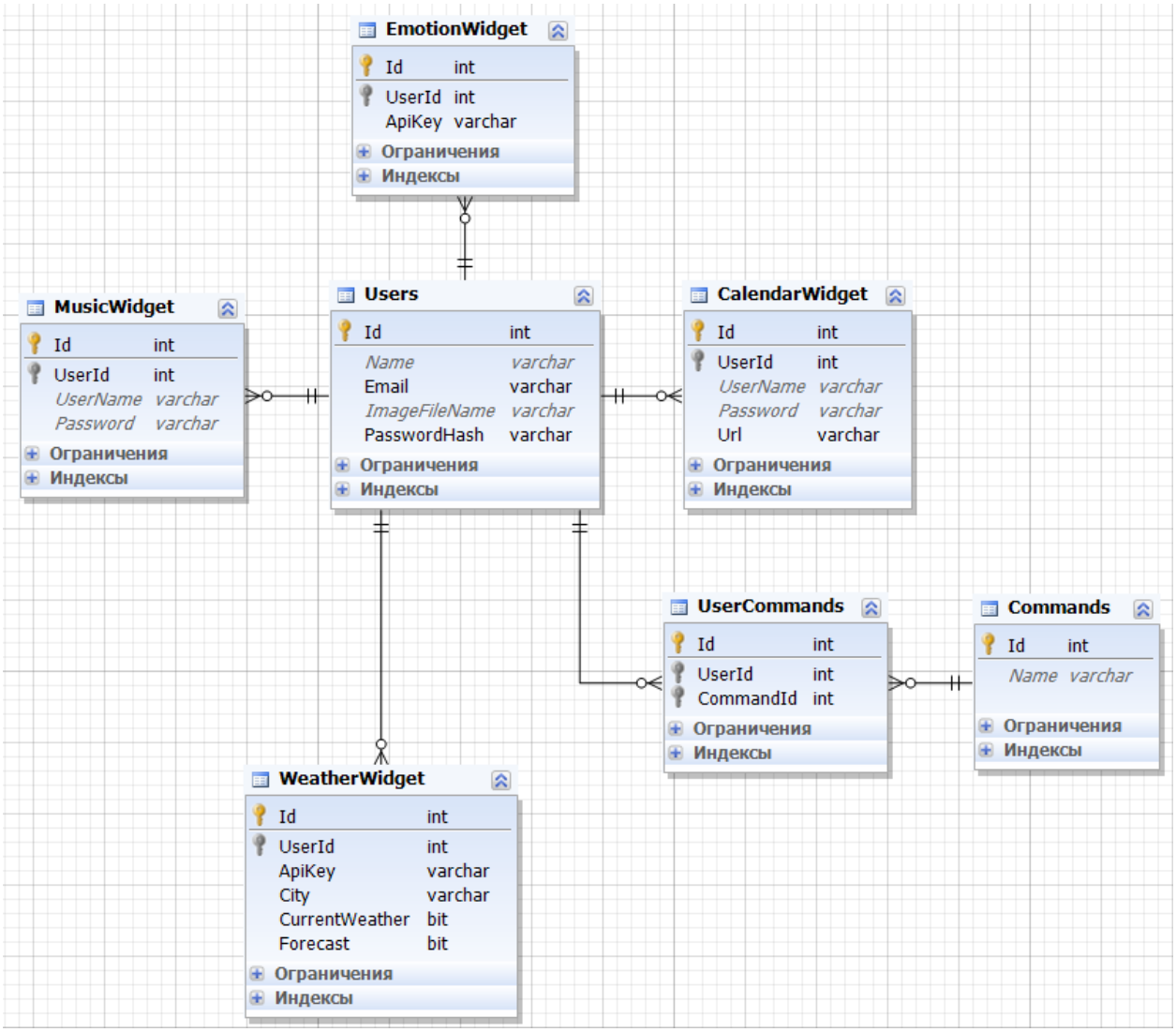

Рисунок 3.6 – Схема базы данных «Soundville»

3.2 Аппаратная часть

Основные аппаратные составляющие, необходимые на основе анализа структурного проектирования это:

1. Управляющий блок, который представлен миникомпьютером Raspberry Pi 3b.
2. Блок визуализации, представленный матрицей.
3. Блок управления визуализацией – универсальный контроллер для управления матрицей.
4. Блок освещения – светодиодные лампы, включающиеся через управляющее реле и управляющий блок.
5. Блок распознавания голосовых команд и блок распознавания образов, совмещенный в веб-камере со встроенным микрофоном.

Рассмотрим подключение элементов, на базе чего будет построена функциональная схема.

3.2.1 Подключение элементов

Светодиодные лампочки подключены через реле.

Каждое реле на модуле имеет 3 вывода:

– NO – нормально-разомкнутый контакт.
– COM – общий контакт.
– NC – нормально-замкнутый контакт.

В нормальном состоянии, когда реле выключено замкнуты контакты COM и NC, когда реле вкючено контакты COM и NC размыкаются и замыкаются контакты COM и NO.

Подключение весьма простое: COM в розетку, NO к лампочке и от лампочки обратно в розетку.

Для того чтобы подключить универсальный контроллер LA.MV29.P к матрица HSD150PX11-B рассмотрим их распиновку из даташита.

В таблице 3.1 представлен набор пинов, необходимых для подключения к контроллеру LA.MV29.P.

Таблица 3.1 – Необходимые пины подключения контроллера LA.MV29.P

Номер порта	Обозначение	Описание
1	VSEL	Питание для платы
2		
5	GND	Земля
7	TXO0-	LVDS ODD 0-
8	TXO0+	LVDS ODD 0+
9	TXO1-	LVDS ODD 1-
10	TXO1+	LVDS ODD 1+
11	TX02-	LVDS ODD 2-
12	TX02+	LVDS ODD 2+
13	GND	Земля
14		
15	TXOC-	LVDS Clock-
16	TXOC+	LVDS Clock+
25	GND	Земля
26		

Проводим анализ двух распиновок: контроллера и самой матрицы (даташиты плат можно найти в свободном доступе в любой промежуток времени). Из чего становится ясно, что подключение происходит следующим образом:

1. Питание к питанию (1 и 2 пины обоих плат).
2. Земля к земле (5, 13, 14, 25, 26 пины контроллера к 3, 4, 7, 10, 13 пинам матрицы).
3. Пины передачи RX и TX (7, 8, 9, 10, 11, 12, 15, 16 пины контроллера к 5, 6, 8, 9, 11, 12, 14, 15 пинам матрицы).

4. Положительные сигналы друг с другом (+ к +).
5. Отрицательные сигналы также (- к -).

В таблице 3.2 представлен набор пинов, необходимых для подключения матрицы HSD150PX11-B.

Таблица 3.2 – Необходимые пины подключения матрицы HSD150PX11-B

Номер порта	Обозначение	Описание
1	VSEL	Питание для платы
2		
3	GND	Земля
4		
5	RX0-	Передатчик 0-
6	RX0+	Передатчик 0+
7	GND	Земля
8	RX1-	Передатчик 1-
9	RX1+	Передатчик 1+
10	GND	Земля
11	RX2-	Передатчик 2-
12	RX2+	Передатчик 2+
13	GND	Земля
14	CLK-	Sampling Clock-
15	CLK+	Sampling Clock+

Веб-камера подключается еще проще: через usb-кабель.

Основное подключение всех устройств происходит в плату миникомпьютера Raspberry Pi 3b. Остановимся на этом поподробнее.

Контроллер управления матрицей подключается по hdmi кабелю, веб-камера со встроенным микрофоном подключается через usb-порт. Колонки намеренно ни к чему не подключаю по кабелю, так как в raspberry pi есть встроенный bluetooth-модуль через который удобно будем подключать bluetooth-колонки. Далее остается подключить только реле, для включения светодиодных ламп.

Как уже было сказано, для подключения реле необходимо четыре контакта: GND, IN1, IN2 и VCC. На основе этого подключаем контакт GND к девятому пину платы, IN1 и IN2 к 11 и 13 пинам платы, а питание VCC к четвертому пину.

Плата Raspberry Pi 3b имеет 40 контактов GPIO, что является весьма удобным для использования. Однако для целей проекта это избыточно и используется всего 4 разъема. Возможно, в дальнейшем при совершенствовании и улучшении проекта будут задействованы еще несколько.

На рисунку 3.7 представлена распиновка, назначение и описание разъемов GPIO Paspberry Pi 3b.

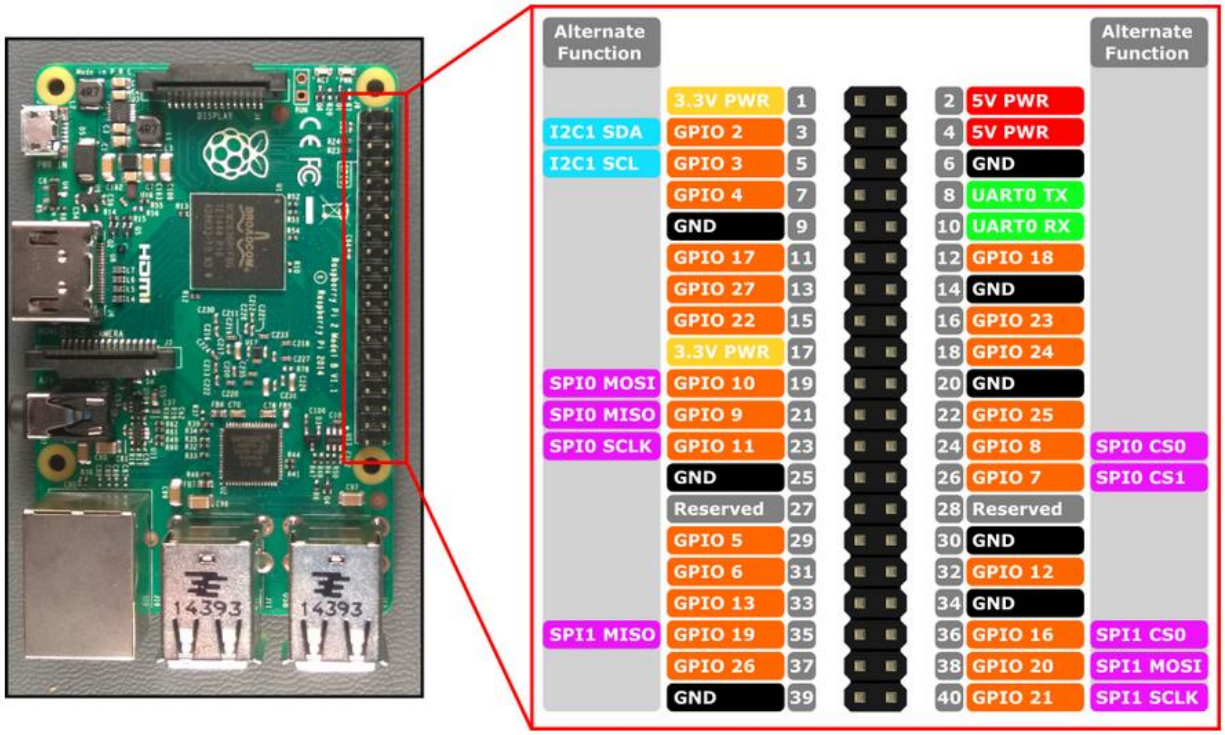

Рисунок 3.7 – Разъемы GPIO Paspberry Pi 3b

На основе данных заключений была разработана функциональная схема, представленная в приложении.

4 ОПИСАНИЕ ЭЛЕМЕНТОВ СХЕМЫ

4.1 Программная часть

Рассмотрим более детально функционирование программного проекта. Для этого проведем анализ всех классов, рассмотрим назначение методов и свойств класса.

4.1.1 Сборка `Soundville.Domain.dll`

Сборка `Soundville.Domain.dll` является уровнем доступа к данным (DAL), на этом уровне происходит обращение к базе данных.

В этой сборке находится четыре пространства имен. Перейдем к рассмотрению каждого из них.

4.1.1.1 Пространство имен `Soundville.Domain.Models`

Данное пространство имен содержит все модели, сопоставляемые таблицам базы данных с помощью Entity Framework. Маппинг этих моделей позволяет получать модели со всеми необходимыми связями.

Все классы, находящиеся в пространстве имен `Soundville.Domain.Models` реализуют интерфейс `IBaseDomainModel`. Данный интерфейс содержит свойство `Id`, являющийся идентификатором любой модели.

Класс `User` хранит в себе информацию о пользователе.

Перечисляемые виджеты могут быть не заданы, в таком случае, на зеркале будет отображаться один главный виджет с часами. Этот виджет идет по умолчанию и убрать его невозможно.

Все голосовые команды также являются не обязательными. Пользователь может самостоятельно выбрать необходимые именно ему команды.

Класс `CalendarWidget` – модель виджета календаря событий, имеет такие свойства как: идентификатор, идентификатор юзера, имя и пароль при необходимости подтягивать данные через данные креденшелы и урл на закрытый доступ к данным пользователя, которые располагаются в .ics файле в json формате.

Класс `MusicWidget` хранит информацию ою имени и пароле пользователе, через которые можно получить доступ на интернет-ресурс с музыкой.

Класс `WeatherWidget` содержит информацию о том, в каком городе необходимо выводить информацию о погоде, показывать только текущую погоду или предсказывать ее на несколько дней и ключ к открытому api погодному ресурсу.

Класс `EmotionWidget` содержит ключ к открытому api ресурсу AzureEmotion API.

Класс Commands хранит информацию о названии голосовой комманды, а класс UserCommands связывает эту команду с конкретным юзером.

4.1.1.2 Пространство имен Soundville.Domain.Services

Данное пространство имен содержит сервисы, благодаря которым модели уровня представления могут получать необходимые данные из доменных моделей. Все классы, находящиеся в пространстве имен Soundville.Domain.Services реализуют интерфейс DomainService. Данный интерфейс содержит базовые методы для работы классов. Рассмотрим сервисы подробнее.

Класс UserDomainService представляет собой сервис для работы с моделью User.

Класс CalendarWidgetDomainService работает с моделью CalendarWidget.

Класс MusicWidgetDomainService реализует методы для работы с моделью MusicWidget.

Класс WeatherWidgetDomainService – сервис для работы с моделью WeatherWidget.

Класс EmotionWidgetDomainService – сервис для работы с моделью EmotionWidget.

Класс CommandsDomainService – сервис для работы с моделью Commands и UserCommands.

4.1.1.3 Пространство имен Soundville.Domain.Installers

Данное пространство служит для внедрения зависимостей с помощью IoC-контейнера Windsor Castle в рассматриваемую сборку и инсталлирования DbContext ORM EntityFramework.

4.1.1.4 Пространство имен Soundville.Domain.EntityFramework

Пространство Soundville.Domain.EntityFramework служит для кофигурирования доменных моделей и создания SoundvilleContext, необходимого для работы EntityFramework.

4.1.2 Сборка Soundville.Infrastructure.dll

Сборка Soundville.Infrastructure.dll является вспомогательной и содержит в себе необходимые константы для приложения.

4.1.3 Сборка Soundville.Configuration.dll

Сборка Soundville.Configuration.dll также является вспомогательной и содержит классы, для внедрения зависимостей в проект с помощью IoC-контейнера Windsor Castle.

4.1.4 Сборка `Soundville.Presentation.dll`

Сборка `Soundville.Presentation.dll` содержит классы, являющиеся связующим звеном между клиентом и сервером и состоит из пяти пространств имен. Рассмотрим некоторые из них.

4.1.4.1 Пространство имен `Soundville.Presentation.Models`

Пространство имен `Soundville.Presentation.Models` состоит из трех подпространств:

- `Soundville.Presentation.Models.MirrorSettings;`
- `Soundville.Presentation.Models.Profiles;`
- `Soundville.Presentation.Models.Account.`

Данные подпространства содержат классы уровня представления для работы с настройкой виджетов для зеркала, профилем пользователя.

4.1.4.2 Пространство имен `Soundville.Presentation.Services`

Данное пространство имен содержит сервисы, для работы с моделями уровня представления на самом верхнем уровне приложения. В нем содержатся такие классы, как:

- `ProfilePresentationService;`
- `MirrorSettingsPresentationService.`

4.1.5 Сборка `Soundville.Web.dll`

Сборка `Soundville.Web.dll` содержит классы, предназначенные для управления веб-приложением. В данной сборке содержатся контроллеры, скрипты и код для рендеринга страниц.

4.1.6 Сборка `Anne.dll`

Пространство имен `Anne.dll` состоит из восьми подпространств:

- `Anne.Calendar;`
- `Anne.Controls;`
- `Anne.Core;`
- `Anne.IO;`
- `Anne.Models;`
- `Anne.Networking;`
- `Anne.Emotion;`
- `Anne.ViewModels.`

Данное пространство имен содержит сервисы, модели уровня представлений, для работы на самом верхнем уровне приложения. В данной сборке содержатся контроллеры, скрипты и код для рендеринга страниц и виджетов. Например, в данной сборке находятся классы для настройки сервисов, сервис для аудио – `AudioService`, блютуз сервис – `BluetoothService` и сервис для работы с фотографией и событиями из календаря пользователя – `PhotoService` и `CalendarService`, а также `WeatherService` класс для работы виджета погоды. Также здесь

обрабатывается событие распознавания фразы и дальнейшая обработка пришедшей команды: вызов музыкального плеера, обработка видеопотока и распознавание эмоций, выполнение воспроизведения входящей фразы для системы.

Например, класс `MainPage` отвечает за работу и рендеринг страницы `MainPage.xaml`, которая является главной составляющей приложения. Сюда динамически подгружаются несколько виджетов, представленных следующими классами: `AudioPlayer`, `CurrentWeather`, `EventCalendar`, `ForecastWeather`, `SystemClock`. Основное действие, которое способствует работе данной страницы – это срабатывание события распознавания фразы `OnSpeechEnginePhraseRecognized`. После того, как событие сработало происходят дальнейшие вызовы функций, в зависимости от пришедшей фразы, а значит голосовой команды пользователя.

4.1.7 Сборка `Anne.Core.dll`

Данная сборка необходима для удобного взаимодействия между сервисами, так как содержит интерфейсы для классов комманд, фабрики и обработчики событий для логирования и дебага приложения.

4.1.8 Сборка `Anne.Extensions.dll`

Пространство имен `Anne.Extensions.dll` содержит методы-расширения, необходимые для работы приложения.

4.1.9 Сборка `Anne.Speech.dll`

Сборка `Anne.Speech.dll` содержит классы для интерпретации голосовых команд, обработки фраз. Данная сборка работает с грамматикой в приложении, обрабатывает входящий аудиопоток пользователя, распознает фразы, преобразовывает их в команды, если это возможно.

Данная сборка содержит такие классы, как `PhraseRecognizedEventArgs` и `StateChangedEventArgs`. Последнее событие срабатывает при смене состояния голосовой команды, а `PhraseRecognizedEventArgs` – если срабатывает событие распознавания фразы, что ведет к дальнейшему выполнению действий в зависимости от распознанной фразы.

Также в данной сборке содержатся классы `CommandsInterpreter` и `SpeechEngine`, являющиеся основными для распознавания фраз и преобразования их в команды для приложения.

4.2 Аппаратная часть

4.2.1 Raspberry Pi 3 B

Внешне миникомпьютер представляет собой небольшую плату, однако под капотом у него Broadcom 2837 quad-core ARMv8 Cortex-A53 64bit

(1,2GHz). Более подробные характеристики платы представлены в таблице 4.1.

Таблица 4.1 – Характеристики Raspberry Pi 3 B

Процессор	Broadcom 2837 quad-core ARM Cortex-A53 64bit (1,2GHz)
Оперативная память	1Gb
Видеовыход	HDMI
A/V выход	A/V выход 3.5мм jack 4 pin
USB порты	USB 2.0 x 4
Сеть	WiFi 802.11n, 10/100Mb RJ45 Ethernet
Bluetooth	Bluetooth 4.1, Bluetooth Low Energy
Слот для карты памяти	Micro SD
GPIO	40
Процессор	Broadcom 2837 quad-core ARM Cortex-A53 64bit (1,2GHz)

Основное отличие от предшественников это чип, отличающийся своей высокой производительностью, встроенный модуль WiFi (802.11n) ну и поддержка Bluetooth 4.1 и Bluetooth Low Energy (BLE). Внешний вид платы показан на рисунке 4.1.

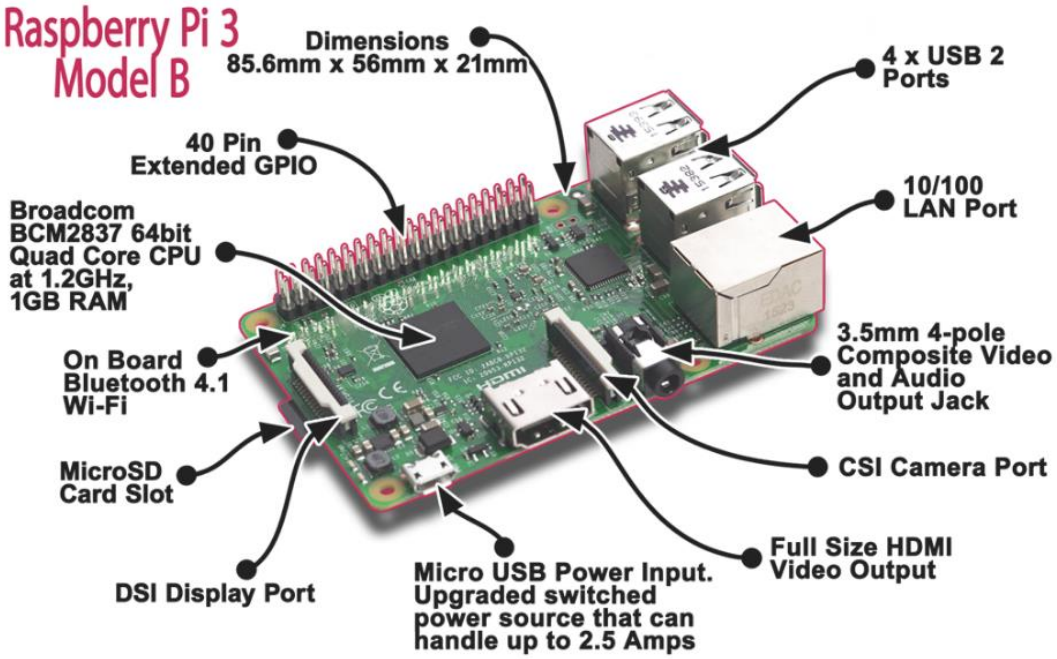

Рисунок 4.1 – Внешний вид контроллера Raspberry Pi 3 B

4.2.2 Матрица HSD150PX11-B и контроллер LA.MV29.P

Про матрицу сказать особо нечего, кроме его характеристик, которые представлены в таблице 4.2.

Таблица 4.2 – Характеристики матрицы HSD150PX11-B

Диагональ	15
Модель (Бренд)	HannStar
Поверхность дисплея	Матовая
Подсветка экрана	Ламповая (1 CCFL)
Разрешение матрицы	1024x768 (XGA)

Более интересная вещь – это контроллер для него.

Для сборки монитора из старой матрицы требуется приобрести только универсальный контроллер. Для проекта был выбран контроллер данной модели: LA.MV29.P. Внешний вид платы показан на рисунке 4.2.

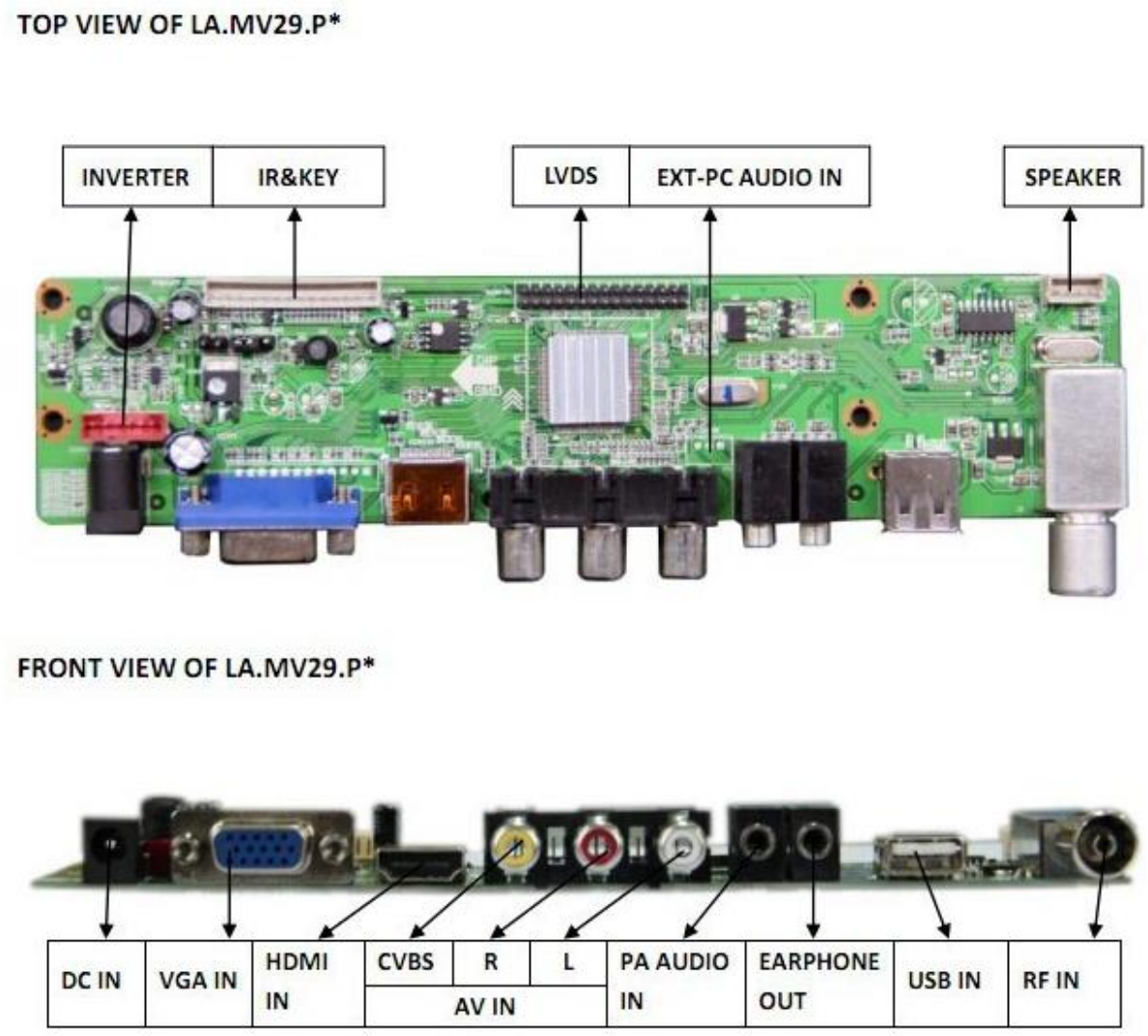

Рисунок 4.2 – Внешний вид контроллера LA.MV29.P

LA.MV29.P контроллер является универсальным LCD контроллером, оснащенный такими интерфейсами как VGA, HDMI, AV, TV, USB.

Максимальное разрешение 1920 * 1200, поддержка 1080P, сигнал HD, а так же поддержка обновления через USB. Встроенный аудио модуль обработки сигнала, поддержка баланса, высокие частоты, регулировка баса. 3D-видео декодирование, яркое изображение, четкое изображение, высокое качество. Автоматическое определение видео и аудио форматов. Динамическая регулировка яркости. Аналоговый аудио усилитель. В режиме ожидания потребление <0.5 Вт. Поддерживает 15-дюймовый – 32 дюймовый экраны. Вот что можно сказать о данной модели контроллера. Так же характеристики можно увидеть в таблице 4.3.

Таблица 4.3 – Характеристики контроллера LA.MV29.P

Программируемый чип	TSUMV29
Экран интерфейса	LVDS
Напряжение экрана	3.3В / 5В / 12В
Требования к питанию	12В 3А
Чип обработки	MStar TSUMV59XU
Размер печатной платы	190 мм * 44 мм
Размер общий (с разъемами)	190мм * 56мм * 15мм

При выборе контроллера следует обратить внимание на питание матрицы (3.3В\5В\12В) поддержку необходимого разрешения, битности, интерфейс, а так же желательно наличие кабелей, дабы не попасть в просак.

4.2.3 Камера

Все камеры можно условно поделить на компьютерные, планшетные, IP-камеры, беспроводные и игровые. Но в действительности их различают по таким параметрам — разрешение матрицы, качество оптики, тип крепления, способ передачи информации.

Различают два основных типа матриц CMOS и CCD. Веб-камеры, которые предназначены для общения в интернете, оснащаются матрицей типа CMOS. Матрица CCD имеет лучшее качество передачи изображения, нежели первый тип. Поэтому ее устанавливают в веб-камерах, которые предназначены для охранных целей.

Разрешающие способности – еще один важный момент, который влияет на качество изображения. Минимальное разрешение веб-камеры 0,3 мпик, т.е. 320x240 пикселей, которого будет достаточно для проведения видеоконференций. Максимальное разрешение веб-камеры составляет 2592x1944 пикселей. Оптимальным вариантом разрешения для веб-камер является 640x480 и 1280x960 пикселей, что позволяет четко видеть собеседника, делать снимки и видеоролики.

Оптика, установленная на веб-камерах влияет на качество картинки. Производители стараются устанавливать качественную оптику. Она может идти как в дорогих, так и в бюджетных моделях. К высококачественной стеклянной оптике, например, относят Carl Zeiss.

Частота кадров – параметр, к которому стоит проявлять осторожность. Частота кадров колеблется от 9 до 90Гц. Маленькая частота будет передавать изображение в виде слайд-шоу. Большая частота кадров может быть не совместима со скоростью интернета и в результате общение станет не возможным. Оптимальным вариантом является 40 - 70Гц.

Следующий параметр, который влияет на качество изображения — чувствительность матрицы. От уровня чувствительности матрицы зависит качество изображения в той или иной освещенности. Если чувствительность матрицы будет не высокая, то плохая освещенность создаст помехи при передаче картинки.

Также камера может быть оснащена встроенным микрофоном, функциями zoom, подсветки и автофокусом.

Подключить веб-камеру можно двумя способами – через USB провод или модем. В основном используют первый способ подключения, потому что скорость всегда будет высокой.

Для проекта был выбран вариант веб-камеры Logitech HD Webcam C270 со встроенным микрофоном. Характеристики камеры приведены в таблице 4.4.

Таблица 4.4 – Характеристики веб-камеры Logitech HD Webcam C270

Количество точек матрицы	1.3 Мп (3 Мп интерполированное)
Максимальное разрешение видео	1280x720
Максимальное разрешение снимка	2048 x 1536
Запись видео	да
Мин. дистанция фокусировки	0.4 м
Встроенный микрофон	да

Кроме того, веб-камера снабжена встроенным программным обеспечением, которое подстраивается под условия недостаточного освещения и фоновые шумы. Внешний вид камеры представлен на рисунке 4.3.

Рисунок 4.3 – Внешний вид веб-камеры Logitech HD Webcam C270

Идущая как бы в комплекте камера Raspberry Camera Board была не включена в проект намеренно. Во-первых, шлейф у данной камеры значительно мал, по сравнению с выбранной моделью, а во-вторых, было принято решение совместить камеру и микрофон в одном устройстве, чтобы облегчить конструкцию.

4.2.4 Микрофон

В наше время микрофон из устройства, которое необходимо узкому кругу людей превратился в популярный девайс. Из многообразия представленных моделей бывает трудно на чем-то остановиться. Однако для цели проекта подойдет абсолютно любой микрофон, имеющий более менее хорошие характеристики.

Основная разница между моделями заключается в аудио-интерфейсе, технических характеристиках и механизме передачи звука. Рассмотрим их более подробно.

Существует два основных способа подключения микрофона к компьютеру:

1. Аналоговый-джек микрофона подключается в стандартный разъем аудио входа. Такой интерфейс имеет множество плюсов. Есть возможность включить в схему различную электронику для работы со звуком, например предусилитель.
2. USB-микрофон подключается к USB разъему. Дорогие модели могут иметь встроенные предусилитель и другую различную электронику. В дешевых микрофонах дополнительной электроники нет. USB микрофон не обладает такой гибкостью как аналоговый, но если подобрать достаточно дорогую модель, то ее функционал удовлетворит большинство рядовых пользователей.

Есть несколько главных технических характеристик, которые оказывает определяющее влияние на качество звука, а именно:

1. Звуковое давление – данный параметр определяет максимальный уровень звука в децибелах, который может передаваться микрофоном без сильных искажений.
2. Направленность – определяет чувствительность микрофона к звукам, которые поступают с разных направлений. В основном встречаются не направленные устройства. Они одинаково улавливают звук из разных направлений и хорошо подходят для домашнего компьютера. В некоторых случаях может потребоваться четкая запись конкретного звука, в таком случае используют односторонние микрофоны.
3. Уровень чувствительности – определяет минимальный уровень звука, который может уловить устройство. Девайсы с высокой чувствительностью также восприимчивы к лишним шумам. Поэтому, тут нельзя однозначно сказать, что хорошие показатели этой характеристики это хорошо.

4. Рабочий частотный диапазон – понятно, что чем больше у микрофона диапазон поддерживаемых частот, тем конечно же лучше. Только следует понимать: частота человеческой речи находится в диапазоне 0,1 – 10 кГц, а значит для записи речи вполне хватит девайса, который поддерживает данные параметры.

Ну и существует два вида передачи звука:

1. Конденсаторный – звук передается благодаря изменению емкости в конденсаторах, которые располагаются между чувствительным мембранами. Для хорошей звукозаписи профессионалы рекомендуют приобретать микрофоны конденсатного типа, так как именно такие устройства применяются профессиональных студиях.
2. Динамический – при передаче звуковых волн применяется чувствительная мембрана, располагающаяся в магнитном поле. Данная конструкция более проста, поэтому чаще используется в бюджетном варианте.

Для проекта был выбран вариант веб-камеры Logitech HD Webcam C270 со встроенным микрофоном.

5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

5.1 Программная часть

Веб-приложение по настройке смарт-зеркала представляет из себя несколько страниц и ограниченный функционал, необходимый для настройки. Главных модулей в приложении два: страница настройки виджетов и страница настройки голосовых команд. Функционал данных страниц весьма прост: достать и отобразить пользовательские настройки, а затем сохранить измененную информацию.

5.2 Аппаратная часть

5.2.1 UWP (Universal Windows Platform)

UWP (Universal Windows Platform) представляет собой унифицированную платформу для создания и запуска приложений в Windows 10. Абстрактная модель платформы представлена на рисунке 5.1.

Рисунок 5.1 – Схема платформы UWP (Universal Windows Platform)

UWP результат работы и усовершенствования более ранних технологий. С выходом Windows 8 была внедрена новая архитектурная платформа для приложений – Windows Runtime (WinRT), которая позволяла запускать приложения в так называемом режиме Modern (Metro) на десктопах, планшетах. Затем с выходом Windows 8.1 и Windows Phone 8.1 эта технология получила развитие – появились "универсальные приложения", которые можно было запускать сразу Windows 8.1 и WP8.1. И в июле 2015 года официально вышла новая ОС Windows 10. Она использует платформу UWP, которая представляет собой развитие Windows Runtime. UWP имеет ряд преимуществ:

1. Широта распространения.

2. Поддержка широкого круга устройств, разных языков и технологий.

3. Удобство распространения и широкие возможности платформы.

Для более удобного понимания на рисунке 5.2 приведена схема преимуществ платформы.

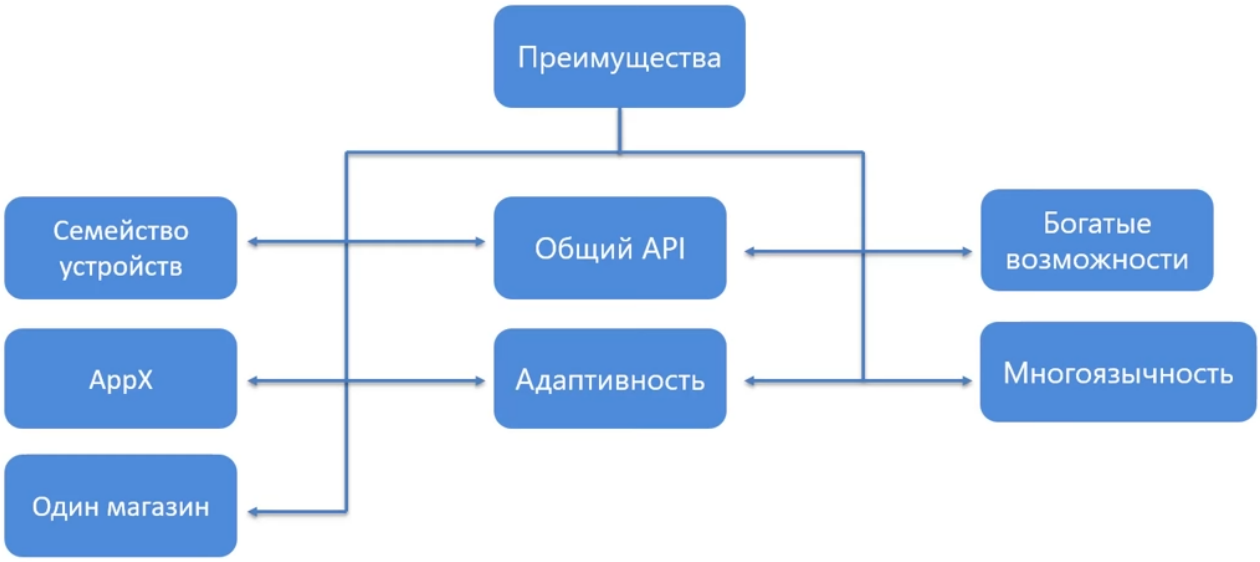

Рисунок 5.2 – Схема преимуществ платформы UWP

Для программирования необходима ОС Windows 10, среда разработки Visual Studio 2017 и режим разработчика.

Как подсказывает название платформы, она является универсальной для всех устройств экосистемы Windows 10. А это обычные дестопы, планшеты, мобильные устройства, устройства IoT (интернет вещей), Xbox, устройства Surface Hub. И приложение UWP может одинаково работать на всех этих платформах, если на них установлена Windows 10.

Сама по себе платформа означает, что до тех пор, пока не будет использоваться какой-либо специфичный для платформы API, можно запустить приложение на любом устройстве с Windows 10, будь то на Raspberry или персональном компьютере. Благодаря этим знаниям разработка приложений происходит намного быстрее, потому что не приходится каждый раз ее разворачивать по сети на другое устройство.

5.2.2 Windows 10 IoT Core

Windows 10 IoT Core – это операционная система, оптимизированная для небольших устройств. На данный момент его можно запустить на Raspberry Pi 2/3, Arrow DragonBoard 410c и MinnowBoard MAX. Windows 10 IoT Core позволяет нам писать приложения UWP, как это было бы на обычном рабочем столе Windows 10, а также добавляет некоторые специфичные для платформы API.

Чтобы управлять устройствами Windows IoT, необходимо загрузить и установить Windows IoT Core Dashboard. Во вкладке «Мои устройства» устройств нет, потому что они еще не настроены.

Далее необходимо подготовить SD-карточку. Следует отметить, что это карточка потеряет все текущие данные и она должна быть как минимум 10 класса. Чтобы установить IoT Core Windows 10 необходимо выполнить следующие действия:

1. Открыть панель инструментов IoT.
2. Откройть меню «Настройка нового устройства».
3. Подключите SD-карту к компьютеру.
4. Настроить имя своего устройства, пароль администратора и подключение Wi-Fi (будет доступно, если компьютер уже подключен к любой сети Wi-Fi).
5. Необходимо принять условия лицензии, нажать кнопку «Загрузить и установить».

В итоге вкладка установки нового устройства будет выглядеть как показано на рисунке 5.3.

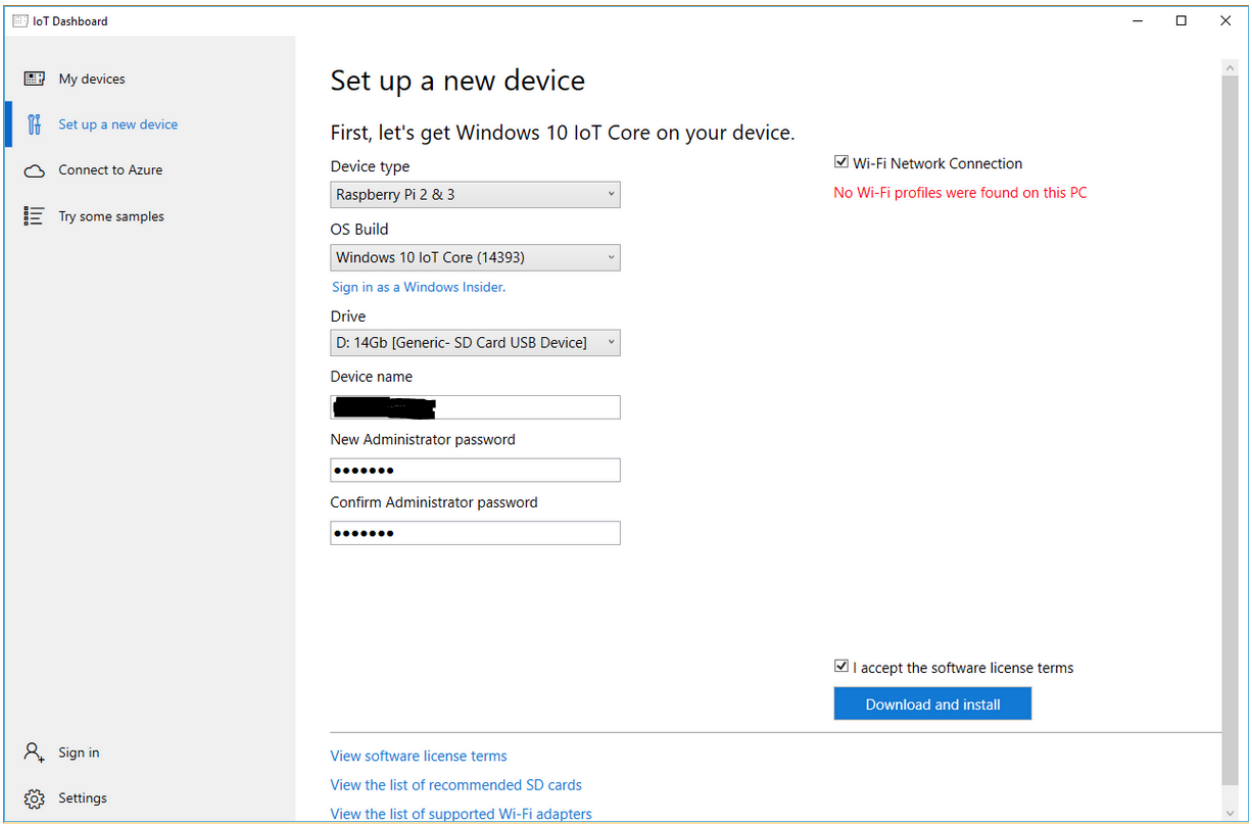

Рисунок 5.3 – Установка нового устройства

Далее необходимо просто подключить карточку, включить устройство и открыть панель инструментов IoT. Далее можно работать с Raspberry Pi либо по кабелю HDMI, либо использовать удаленный клиент Windows IoT.

5.2.3 Windows Speech Recognition

Распознаватели речи оперируют такими понятиями, как грамматический процессор, пользовательские агенты, грамматика. Познакомимся с ними поближе.

Грамматический процессор – это любой объект, который принимает входные грамматики.

Пользовательский агент – грамматический процессор, который принимает входные данные пользователя и сопоставляет этот вход с грамматикой для получения результата распознавания.

Распознаватель речи является пользовательским агентом со следующими входами и выходами:

1. Вход: грамматика или несколько грамматик. Эти грамматики сообщают распознавателю слова и паттерны слов для прослушивания.
2. Вход: аудиопоток, который может содержать речевой контент, соответствующий грамматике.
3. Выход: описание результатов, которые указывают подробности о речевом содержимом, обнаруженном устройством распознавания речи. В информативных целях большинство практических распознавателей будут включать по крайней мере транскрипцию любых обнаруженных слов.
4. Выход: в среду хоста могут быть переданы ошибка и другая информация о производительности.

В качестве системы голосового распознавания в проекте использован Windows Speech Recognition. Благодаря WindowsMedia.SpeechRecognition API очень легко реализовать голосовой пользовательский интерфейс в Windows 10. Данная система предлагает синтаксис для предоставления грамматики. Грамматика предназначена для использования распознавателями речи и другими грамматическими процессорами, чтобы можно было опеределять слова и шаблоны слов, которые будут прослушиваться распознавателем речи.

Основное использование грамматики распознавания речи состоит в том, чтобы позволить речевому приложению сказать распознавателю, что слушать, а именно:

1. Слова, которые можно произнести.
2. Шаблоны, в которых эти слова могут встретиться.
3. Разговорный язык каждого слова.

Некоторые распознаватели речи поддерживают способность динамически подстраиваться под голос говорящего и часто сохраняют способность сохранять данные адаптации для этого голоса для будущего использования. Данные спикера могут также включать списки слов, которые чаще всего произносятся пользователями. Формат грамматики, очевидно, не учитывает эти возможности.

Существует технология обработки речи для идентификации языка, проверки громкоговорителей (также известная как голосовая печать),

распознавания динамиков (также называемых идентификацией динамика) среди многих других функций.

Синтаксис формата грамматики представлен в двух формах: расширенная BNF (ABNF) и XML-форме. Расширенная BNF – это простое текстовое представление, похожее на традиционную БНФ-грамматику, которая обычно используется в области распознавания речи. XML форма использует в свою очередь XML элементы для представления грамматических конструкций.

Форма ABNF и XML-форма указаны для обеспечения того, чтобы два представления были семантически отображены. Должна быть предусмотрена возможность автоматического преобразования грамматики формы ABNF в грамматику формы XML (или наоборот), чтобы семантическая производительность грамматик была идентичной. Эквивалентность семантического представления подразумевает, что:

1. Обе грамматики принимают тот же язык, что вводят, и отклоняют тот же язык, что и ввод.
2. Обе грамматика разбирают любую входную строку одинаково.

Распознаватель речи способен сопоставлять аудиовход с грамматикой для создания транскрипции исходного текста (также известного как буквальный текст) обнаруженного ввода. Распознаватель может быть способен, но не обязан выполнять последующую обработку необработанного текста для создания семантической интерпретации ввода.

Например, естественное высказывание языка «Я хочу заказать перелет из Праги в Париж» может привести к следующей структуре данных XML.

```
<book-flight>
    <depart>Prague</depart>
    <arrive>Paris</arrive>
</book-flight>
```

Для выполнения этого этапа интерпретации требуются инструкции семантической обработки, которые могут содержаться в грамматике, определяющей законный разговорный ввод или связанный с ним документ.

В проекте используется XML-форма для системы распознавания команд. Ниже приведен пример простой грамматики, поддерживающей такие команды, как «открыть файл» и «пожалуйста, переместите окно».

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
"http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar xmlns="http://www.w3.org/2001/06/grammar"
xml:lang="en"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
     http://www.w3.org/TR/speech-grammar/grammar.xsd"
     version="1.0" mode="voice" root="basicCmd">

     <meta name="author" content="Stephanie Williams"/>

     <rule id="basicCmd" scope="public">
       <example> please move the window </example>
       <example> open a file </example>

       <ruleref
uri="http://grammar.example.com/politeness.grxml#startPolite"/>

       <ruleref uri="#command"/>
       <ruleref
uri="http://grammar.example.com/politeness.grxml#endPolite"/>

     </rule>

     <rule id="command">
       <ruleref uri="#action"/> <ruleref uri="#object"/>
     </rule>

     <rule id="action">
        <one-of>
           <item weight="10"> open <tag>TAG-CONTENT-1</tag>
</item>
           <item weight="2"> close <tag>TAG-CONTENT-2</tag>
</item>
           <item weight="1"> delete <tag>TAG-CONTENT-3</tag>
</item>
           <item weight="1"> move <tag>TAG-CONTENT-4</tag>
</item>
         </one-of>
     </rule>

     <rule id="object">
       <item repeat="0-1">
         <one-of>
           <item> the </item>
           <item> a </item>
         </one-of>
       </item>

       <one-of>
           <item> window </item>
           <item> file </item>
           <item> menu </item>
       </one-of>
     </rule>

     </grammar>
```

Universal Windows Platform обеспечивает возможность работать с такими классами, как SpeechRecognizer и SpeechSynthesizer.

Для работы голосовых команд введен класс _speechEngine с событиями StateChangedEventArgs и PhraseRecognizedEventArgs. При смене состояния голосовой команды срабатывает событие StateChangedEventArgs и на зеркале отображается новое значение команды.

```
    async void OnSpeechEngineStateChanged(object sender,
StateChangedEventArgs e)
                => await this.ThreadSafeAsync(() =>
_hypothesis.Text = e.ToString());
```

Состояние распознавателя речи реализуется стандартным перечислением SpeechRecognizerState. Данное перечисление имеет следующие поля:

1. Capturing – обозначает, что распознаватель речи прослушивает аудиовход от пользователя (полезно при непрерывном прослушивании канала).
2. Idle – указывает, что распознавание речи неактивно и устройство распознавания речи не прослушивает аудиовход.
3. Paused – только для непрерывного распознавания. Указывает, что сеанс распознавания речи все еще активен, но распознаватель речи больше не обрабатывает (пытается распознать) аудиовход. Входящий аудиовход буферизуется. В этом состоянии ограничения могут быть добавлены, удалены и скомпилированы. Если задано одно или несколько ограничений при инициализации сеанса распознавания, удаление всех ограничений и возобновление распознавания приведет к ошибке. Аналогично, если не заданы ограничения при инициализации сеанса распознавания, добавление ограничений и возобновление распознавания также приведет к ошибке.
4. Processing – указывает, что распознаватель речи обрабатывает (пытается распознать) аудиовход от пользователя. Распознаватель больше не захватывает (прослушивает) аудиовход от пользователя. Во время стандартного распознавания состояние может произойти после того, как распознаватель прекратил запись аудиовхода и до того, как будет получен результат распознавания. При непрерывном распознавании это состояние может возникнуть после вызова StopAsync и до запуска события Completed. Полезно для указания, что пользователь должен прекратить говорить.
5. SoundEnded – указывает, что распознаватель речи больше не обнаруживает звук в аудиопотоке. Полезно для скрытия интерфейса распознавания речи. Однако сеанс распознавания остается активным.

6. SoundStarted – указывает, что распознаватель речи обнаружил звук в аудиопотоке. Полезно для обозначения того, что звук (не обязательно речь) был обнаружен.

7. SpeechDetected – указывает, что распознаватель речи обнаружил речевой ввод в аудиопотоке. Полезно для указания того, что речь была обнаружена.

Если срабатывает событие распознавания фразы PhraseRecognizedEventArgs, то происходит дальнейшее выполнение действий в зависимости от распознанной фразы.

```
    async void OnSpeechEnginePhraseRecognized(object sender,
PhraseRecognizedEventArgs e)
    {
        await this.ThreadSafeAsync(() => _voiceCommand.Text =
e.CommandsContext.Command.ToString().SplitCamelCase());

        IContextSynthesizer synthesizer = null;
        switch (e.CommandsContext.Command)
        {
            case CommandsEnum.CurrentWeather:
                synthesizer = _currentWeather;
                break;
            case CommandsEnum.ForecastWeather:
                synthesizer = _forecastWeather;
                break;
            case CommandsEnum.CalendarEvents:
                synthesizer = _eventCalendar;
                break;
            case CommandsEnum.Audio:
                await
_speechEngine.SetRecognitionModeAsync(SpeechRecognitionMode.Paus
ed);
                IAudioCommandListener audioListener =
_audioPlayer;
                await audioListener.PlayRandomSongAsync();
                break;
            case CommandsEnum.Volume:
                IVolumeCommandListener volumeListener =
_audioPlayer;
                await
volumeListener.SetVolumeAsync(e.PhraseText);
                await
_speaker.SetVolumeFromCommandAsync(e.PhraseText);
                break;
            case CommandsEnum.Emotion:
                await
_speechEngine.SetRecognitionModeAsync(SpeechRecognitionMode.Paus
ed);
                await this.ThreadSafeAsync(async () => await
ChangeStreamStateAsync(true));
```

```
                break;
            case CommandsEnum.Help:
                synthesizer = this;
                break;
            case CommandsEnum.Close:
                CoreApplication.Exit();
                break;
        }

        if (synthesizer != null)
        {
            await this.ThreadSafeAsync(
                async () =>
                    await _speechEngine.SpeakAsync(
                        await
synthesizer.GetContextualMessageAsync(e.CommandsContext.DateCont
ext), _speaker));
        }
    }
```

Из кода понятно, что если приходит одна из команд, таких как `CommandsEnum.CurrentWeather`,`CommandsEnum.ForecastWeather` `CommandsEnum.CalendarEvents` или `CommandsEnum.Help`, то необходимо только дать возможность голосовому помощнику что-то прочитать, не вызывая другие сервисы.

И наоборот, если приходит команда `CommandsEnum.Emotion`, то прослушивание аудиопотока пользователя необходимо приостановить на время, а затем включить видеопоток для получения изображения человека и его эмоций для работы с библотекой Emotion API.

В начале компилируется доступная грамматика и происходит прослушивание аудиопотока. При распознавании голоса срабатывает обработчик события и полученная фраза пытается преобразоваться в команду.

```
    CommandsContext ICommandsInterpreter.GetPhraseIntent(string
phrase)
    {
        if (string.IsNullOrWhiteSpace(phrase))
        {
            return CommandsEnum.None;
        }

        if (phrase.StartsWith("Help",
StringComparison.OrdinalIgnoreCase) || Contains(phrase, "What
can I say"))
        {
            return CommandsEnum.Help;
        }
```

```
        if (phrase.StartsWith("Close",
StringComparison.OrdinalIgnoreCase))
        {
            return CommandsEnum.Close;
        }

        if (StartsWith(phrase, "Look") ||
            Contains(phrase, "How do I look") ||
            Contains(phrase, "Tell me how I look"))
        {
            return CommandsEnum.Emotion;
        }

        if (StartsWithAny(phrase, "What", "Read", "How", "On"))
        {
            if (ContainsAny(phrase, "event", "calendar"))
            {
                DateTime? dateContext;
                if (Contains(phrase, "on") &&
TryParseContext(phrase, out dateContext))
                {
                    return new
CommandsContext(CommandsEnum.CalendarEvents, dateContext);
                }
                return CommandsEnum.CalendarEvents;
            }

            if (Contains(phrase, "forecast"))
            {
                return CommandsEnum.ForecastWeather;
            }

            if (ContainsAny(phrase, "temp", "weather"))
            {
                DateTime? dateContext;
                if (Contains(phrase, "on") &&
TryParseContext(phrase, out dateContext))
                {
                    return new
CommandsContext(CommandsEnum.ForecastWeather, dateContext);
                }
                return CommandsEnum.CurrentWeather;
            }
        }
        else if (ContainsAny(phrase, "turn", "mute", "volume",
"loud", "quiet"))
        {
            return CommandsEnum.Volume;
        }
        else if (Contains(phrase, "play"))
        {
            return CommandsEnum.Audio;
```

```
        }

        return CommandsEnum.Dictation;
    }
```

Если фраза распознана и ей есть соответствующая команда, то далее происходит запуск соответствующего модуля по работе этой команды.

В итоге распознавание команд можно изобразить в виде блок-схемы представленной на рисунке 5.4.

Рисунок 5.4 – Блок-схема распознавания голосовых команд

Из блок-схемы более становится более понятно, как происходит распознавание речи в приложении.

В итоге захват аудиопотока пользователя происходит постоянно, за исключением, когда система сама отключает захват. Например, при воспроизведении музыки по команде, либо при распознавании эмоций человека и работе с Emotion API.

5.2.4 Emotion API

Emotion API принимает в качестве входных данных выражение лица на фотографии и возвращает результат в формате json для всех эмоций на изображении, а также к ограничительной рамке для лица, используя Face API. Если же пользователь уже воспользовался Face API, он может отправить просто изображение лица в прямоугольнике.

API выявляет такие эмоции, как гнев, презрение, отвращение, страх, счастье, нейтральность, печаль и удивление. Результат работы Emotion API можно посмотреть на рисунке 5.5.

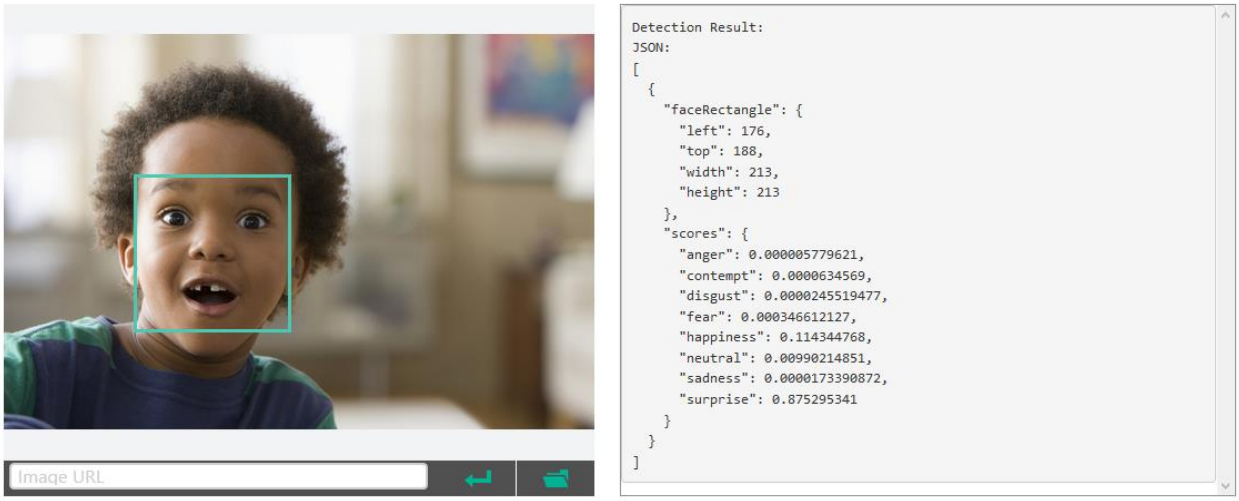

Рисунок 5.5 – Результат работы Emotion API

Характеристики данного Emotion API можно выделить следующие:

1. Поддерживаемые форматы входных изображений: JPEG, PNG, GIF (первый кадр), BMP. Размер файла изображения не должен превышать 4 МБ.
2. Если пользователи уже вызвали Face API, они могут представить прямоугольники лица в качестве необязательного ввода. В противном случае API Emotion сначала вычислит прямоугольники.
3. Обнаруживаемый диапазон размеров лица составляет от 36x36 до 4096x4096 пикселей. Лица вне этого диапазона не будут обнаружены.
4. Для каждого изображения максимальное число обнаруженных лиц равно 64, а лица ранжируются по размеру прямоугольника лица в

порядке убывания. Если лицо не обнаружено, будет возвращен пустой массив.

5. Некоторые лица могут быть не обнаружены из-за технических проблем, например. Очень большие углы лица, большая окклюзия. Наилучшие результаты имеют фронтальные и прифронтовые грани лица.

6. Эмоциональное презрение и отвращение являются экспериментальными.

Более детально работа с библиотекой происходит следующим образом. При распознавании голосовой команды для работы виджета эмоций происходит изменение состояния стриминга видеопотока с камеры.

```
async Task ChangeStreamStateAsync(bool isStreaming)
{
    switch (isStreaming)
    {
        case false:
            await ShutdownWebcamAsync();
            break;

        case true:
            if (!await StartWebcamStreamingAsync())
            {
                await ChangeStreamStateAsync(false);
            }
            break;
    }
}
```

А далее происходит работа со стандартным классом для работы с захватыванием видеопотока `MediaCapture`.

```
async Task<bool> StartWebcamStreamingAsync()
{
    bool successful = true;

    try
    {
        if (_mediaManager != null)
        {
            _mediaManager.Failed -= OnCameraStreamFailed;
            _mediaManager.Dispose();
            _mediaManager = new MediaCapture();
        }

        var cameraId = await GetNamedCameraOrDefault();
        await _mediaManager.InitializeAsync(new
MediaCaptureInitializationSettings
        {
```

```
                StreamingCaptureMode =
StreamingCaptureMode.Video,
                VideoDeviceId = cameraId
            });

            _mediaManager.Failed += OnCameraStreamFailed;
            _preview.Source = _mediaManager;

            await _mediaManager.StartPreviewAsync();
            await CaptureAndProcessEmotionAsync();
        }
        catch (Exception ex) when
(DebugHelper.IsHandled<MainPage>(ex))
        {
            successful = false;
        }

        return successful;
    }
```

Затем создается временный файл фотографии и происходит дальнейшая обработка эмоций на фото с помощью библиотеки Emotion API. Как уже было сказано, библиотека в качестве входных данных принимает выражение лица на изображении.

```
    async Task<IEnumerable<RawEmotion>> CaptureEmotionAsync()
    {
        RawEmotion[] result;

        try
        {
            var photoFile = await _photoService.CreateAsync();
            var imageProperties =
ImageEncodingProperties.CreateBmp();
            await
_mediaManager.CapturePhotoToStorageFileAsync(imageProperties,
photoFile);
            result = await _emotionClient.RecognizeAsync(await
photoFile.OpenStreamForReadAsync());
        }
        finally
        {
            await _photoService.CleanupAsync();
        }

        return result.IsNullOrEmpty()
                ?  await
TaskCache<IEnumerable<RawEmotion>>.Value(() =>
Enumerable.Empty<RawEmotion>())
                : result;
    }
```

Как уже было сказано выше, ответ приходит в формате json: `FaceRectangle` и `Scores`. Затем выбирается наиболее вероятная эмоция фотографии, выбирается смайл полученной эмоции и текст, который будет отображен и произнесен системой.

Блок-схема алгоритма распознавания приведена на рисунке 5.6.

Рисунок 5.6 – Блок-схема распознавания эмоций

Например, если человек улыбался в момент фотографирования, то наиболее вероятный результат, возвращенный Emotion API, конечно будет радость. Система обработает результат библиотеки, подберет необходимый смайл для отображения и возьмет хаотично одну из заготовленных фраз, символизирующую радость. Положим, она может вернуть фразу: «You look happy!», что окажется верным результатом работы виджета.

```
     var emotions = await CaptureEmotionAsync();
     var mostProbable = emotions.ToResults()
                                .FirstOrDefault(result => result
!= Result.Empty);

     _emoticon.Text = Emoticons.From(mostProbable.Emotion);
     var current = _messageLabel.Text;
     var message =
EmotionMessages.Messages[mostProbable.Emotion]
                                  .First(msg => msg != current);

     _messageLabel.Text = message;

     await ChangeStreamStateAsync(false);
     await _speechEngine.SpeakAsync(message, _speaker);
     await
_speechEngine.SetRecognitionModeAsync(SpeechRecognitionMode.Comm
andPhrases);
```

По завершению работы данного модуля временная фотография удаляется и система вновь прослушивает аудиопоток пользователя.

6 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование программного обеспечения (Software Testing) – это проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя действия по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

Принято разделять тестирование по уровням задач и объектов на разных стадиях и этапах разработки ПО:

1. Тестирование частей ПО (модулей, компонентов) с целью проверки правильности реализации алгоритмов – выполняется разработчиками.
2. Функциональное тестирование подсистем и ПО в целом с целью проверки степени выполнения функциональных требований к ПО – рекомендуется проводить отдельной группой тестировщиков, не подчиненной руководителю разработки.
3. Нагрузочное тестирование (в том числе стрессовое) для выявления характеристик функционирования ПО при изменении нагрузки (интенсивности обращений к нему, наполнения базы данных и т. п.) – для выполнения этой работы требуются высококвалифицированные тестировщики и дорогостоящие средства автоматизации экспериментов.
4. Стресс-тестирование — один из видов тестирования программного обеспечения, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования. Стресс-тестирование особенно необходимо для «критически важного» ПО, однако также используется и для остального ПО. Обычно стресс-тестирование лучше обнаруживает устойчивость, доступность и обработку исключений системой под большой нагрузкой, чем-то, что считается корректным поведением в нормальных условиях.
5. Регрессионное тестирование проводят по результатам исправления выявленных на этапе эксплуатации программного продукта ошибок и дефектов. Также, к этому виду тестирования относят испытания программного продукта после внесения в него незначительных изменений, которые не должны влиять на общую функциональность, а вызваны такими обстоятельствами, как необходимость перехода на новую версию операционной системы или новый набор прикладных библиотек стороннего производителя. Цель регрессионного тестирования: доказать, что «ничего не сломалось», и что программный продукт по-прежнему соответствует всем заявленным ранее требованиям.

Этапы тестирования приведены в таблице 6.1.

Таблица 6.1 – Этапы тестирования

Вид тестирования	Стадия, этап	Объект	Критерий
Структурное, надежности	Разработка	Компоненты	Покрытые ветвлений, функций
Сборочное	Разработка	Подсистемы	Функциональность, степень проверки компонентов
Функциональное	Разработка	Система в целом	Соответствие функциональным требованиям ТЗ
Регрессионное	Разработка, сопровождение	Система в целом	Проверка качества внесения изменений
Нагрузочное	Разработка, сопровождение	Система в целом	Оценка статистических характеристик системы, соответствие ТЗ, ТТХ, подбор конфигурация оборудования
Стрессовое	Разработка, сопровождение	Система в целом	Корректность работы системы при предельных нагрузках

Тестирование программного обеспечения в рамках дипломного проекта рекомендуется ограничить проведением функционального тестирования, разделив его на критическое и углубленное.

Критическое тестирование – это процесс поиска ошибок в программе при стандартной ее работе (при правильной последовательности действий, при верном заполнении полей и так далее).

Углубленное (расширенное) тестирование – это процесс поиска ошибок в программе в нестандартных, непредвиденных ситуациях (например, при некорректно вводимых данных).

6.1 Программная часть

Тестирование клиентской части проводилось в многочисленных версиях пяти основных браузеров:

- Internet Explorer;
- Google Chrome;
- Opera;
- Mozilla Firefox.

Перечень граничных и эквивалентных значений для данного программного комплекса приведен в таблице 6.2.

Таблица 6.2 – Перечень граничных и эквивалентных значений

Название поля	Формат данных (из требований)	Перечень граничных значений	Перечень эквивалентных значений
Адрес электронной почты	1. Длина должна быть не более 50 символов. 2. Обязательное поле для заполнения. 3. Строгий формат: test@test.com.	Строка, состоящая из 51 символов; пустая строка	Строка, состоящая из 1 символов, строка, состоящая из 50 символов
Пароль	1. Содержит любые символы. 2 Длина должна быть не менее 5 символов. 3. Обязательное поле для заполнения.	Строка длиной 4 символ	Строка длиной 8 символов
Подтверждение пароля	1. Должно полностью совпадать с полем Пароль.	Строка длиной 4 символ	Строка длиной 8 символов

В таблице 6.3 приведены примеры тестовых случаев для критического тестирования для проверки функциональности работы с данными: просмотр, добавление, редактирование, удаление. В данной таблице рассматривается работа с модулем настроек виджетов и команд для зеркала.

Таблица 6.3 – Пример тестового случая критического тестирования

№	Название модуля	Описание тестового случая	Ожидаемые результаты	Тест пройден
1	2	3	4	5
1	Модуль настроек виджетов	1. В панели меню выбрать пункт «Widgets». 2. Из списка виджетов выбрать «Event Calendar» и нажать кнопку «Enable». 3. В появившемся поле ввода url вставить урл на google календарь.	1. Появляется страница со списком всех виджетов. 2. Кнопка становится яркого цвета и появляется меню по настройке виджета: поле для ввода url календаря и выбор показа виджета (отображать всегда или по команде). 3. Информация появляется в поле.	Да

Продолжение таблицы 6.3

1	2	3	4	5
		4. Выбрать в появившемся меню настройки виджета пункт «Always display». 5. Нажать кнопку «Save». 6. Обновить страницу.	4. Выбран пункт «Always display». 5. Выдается сообщение «Widget setting updated». 6. Выбранные пункты сохранены.	
2	Модуль настроек голосовых команд	1. В панели меню выбрать пункт «Voice Commands». 2. Из списка голосовых команд выбрать «How do I look» и нажать кнопку «Enable». 3. Из списка голосовых команд выбрать «What the weather today» и нажать кнопку «Enable». 4. Нажать кнопку «Save». 5. Обновить страницу.	1. Появляется страница со списком всех голосовых команд. 2. Кнопка становится яркого цвета. 3. Кнопка становится яркого цвета. 4. Выдается сообщение «Voice Commands setting updated». 5. Выбранные пункты сохранены.	Да
3	Модуль настроек профиля пользователя	1. В панели меню выбрать пункт «Profile Settings». 2. В поле «Greeting» ввести приветственное сообщение: «Hello, Anne!». 3. Нажать кнопку «Save». 4. Обновить страницу.	1. Появляется страница с настройками пользовательского профиля. 2. Информация появляется в поле. 3. Выдается сообщение «Profile setting updated». 4. Приветственное сообщение сохранено.	Да

6.2 Аппаратная часть

Тестирование программного обеспечения для аппаратной части проводилось на основе голосовых команд. В ходе тестирования обеспечение было запущено на системе Windows 10 на ноутбуке и на Windows 10 IoT Core на самом устройстве – Raspberry Pi. В таблице 6.4 приведены тесты, проведенные над программным обеспечением.

Таблица 6.4 – Тестирование программы для аппаратной части

№	Название модуля	Описание тестового случая	Ожидаемые результаты	Тест пройден
1	Модуль голосового управления зеркалом	1. Произнести фразу «What the weather today?».	1. Голосовой помощник произносит текущую погоду.	Да
2	Модуль голосового управления зеркалом	1. Произнести фразу «How do I look?».	1. Зеркало включило камеру, выключило, показало и одно из текстовых отображений эмоции. В данном случае «You look happy!».	Да
3	Модуль голосового управления зеркалом	1. Произнести фразу «Help».	1. Голосовой помощник перечисляет всевозможные команды по управлению и вывел их текстовое отображение на зеркало.	Да
4	Модуль голосового управления зеркалом	1. Произнести фразу «Close».	1. Зеркало безопасно завершило свою работу и закрыло свое приложение.	Да

Как видно из таблиц, приложения хорошо справились с тестами, что говорит о высокой их работоспособности.

7 ОПИСАНИЕ РАБОТЫ УСТРОЙСТВА

7.1 Программная часть

Для работы программного веб-приложения Soundville необходимо наличие любой операционной системы с поддержкой любого современного браузера.

Попадая на главную страницу сайта, пользователь видит авторизационную панель либо страницу регистрации (см. рисунок 7.1).

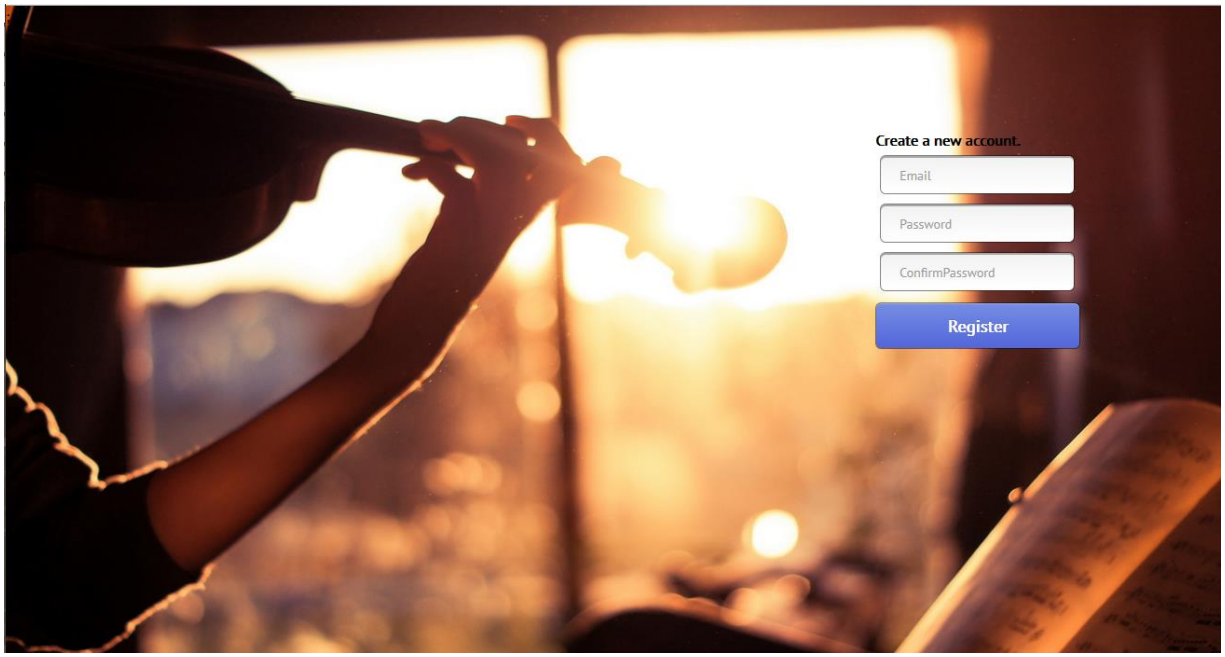

Рисунок 7.1 – Страница регистрации сайта Soundville

После регистрации пользователь попадает на страницу заполнения профиля. Также на эту страницу можно попасть выбрав в меню, которое находится в правой части экрана, кнопку «PROFILE SETTINGS».

На данной странице владелец профиля может отредактировать информацию, находящуюся на странице. Можно изменить имя, почту и фотографию.

Данная информацию может быть необходима для следующих целей в работе виджетов:

1. Имя может понадобиться для приветственного сообщения, а также при ответах системы на вопросы пользователя.
2. Почта необходима для настройки и идентификации персонального кабине пользователя.
3. Фотография нужна для приветственного сообщения.

В дальнейшем планируется подстраивать зеркало под персональные настройки любого пользователя автоматически, после того, как зеркало

распознает конкретного человека, либо оставит только виджет часов, сообщив, что такой пользователь не распознан.

На рисунке 7.2 представлена страница личных настроек пользователя.

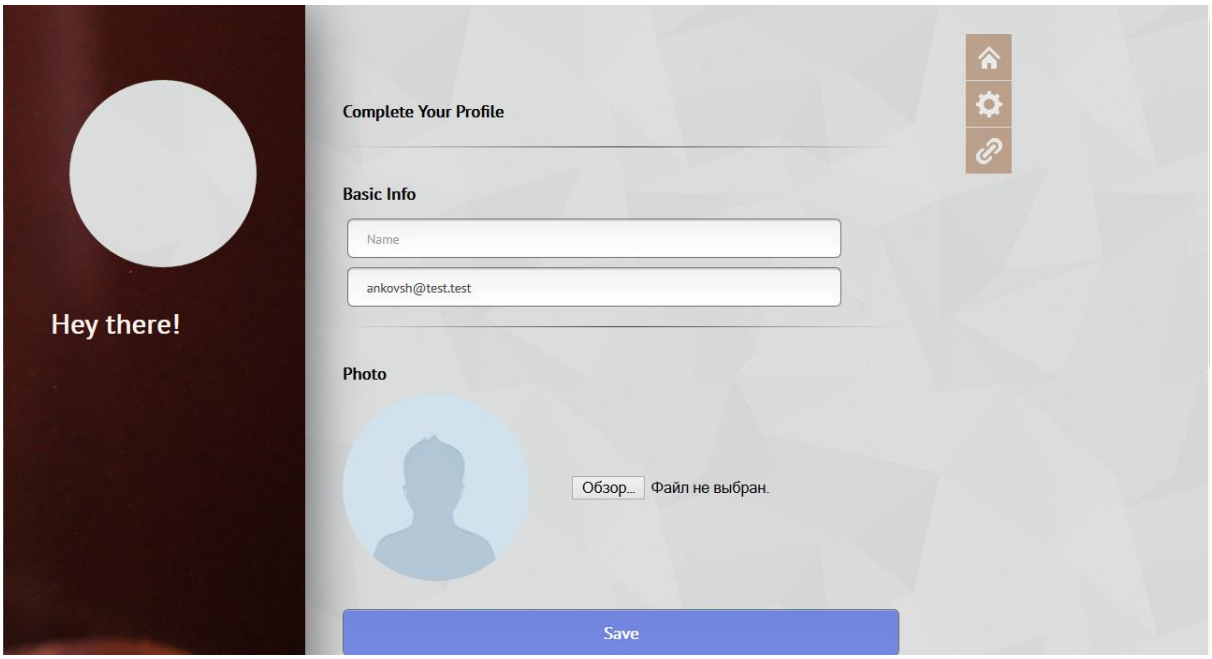

Рисунок 7.2 – Страница редактирования профиля пользователя

После заполнения профиля или, если пользователь просто вошел в систему, открывается страница настроек виджетов для зеркала (см. рисунок 7.3).

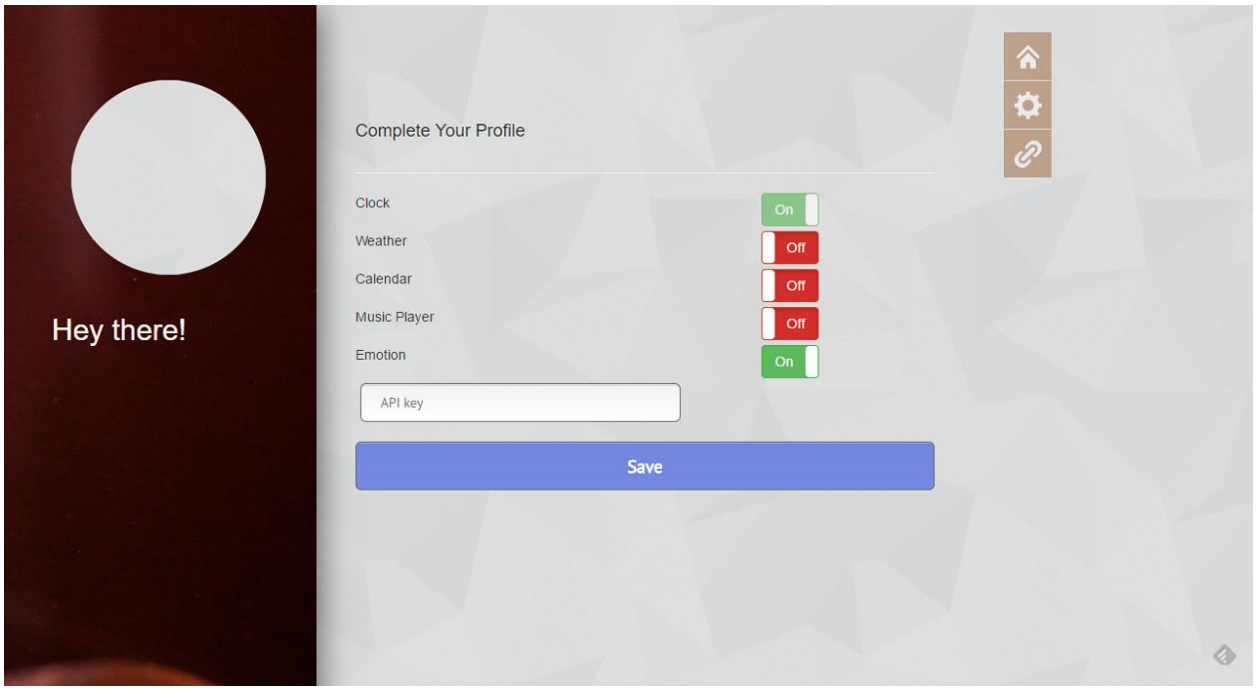

Рисунок 7.3 – Страница настроек виджетов для зеркала

Попадая на страницу настроек виджетов, пользователь видит все существующие виджеты. Для добавления или удаления виджета необходимо переключить кнопку включения/выключения. Данная кнопка располагается около каждого виджета, кроме часового. Виджет с часами нельзя скрыть. Если кнопка серого цвета – это означает, что виджет в данный момент скрыт от пользователя на зеркале. При переключении во включенное состояние около виджета появляются поля для его настройки, а сама кнопка становится яркого цвета.

Подводя итог, на данной странице пользователь может совершить несколько действий:

1. Отредактировать приветственное сообщение, которое пользователь хочет видеть на зеркале.
2. Поменять данные api ключей, если это необходимо для конкретного виджета.
3. Поменять настройки виджета для отображения: показывать постоянно либо при вызове с помощью голосовой команды.
4. Убрать виджет при необходимости.

Также пользователю доступна еще одна страница для настроек. Это страница настроек голосовых команд для зеркала. Данная страница представлена на рисунке 7.4.

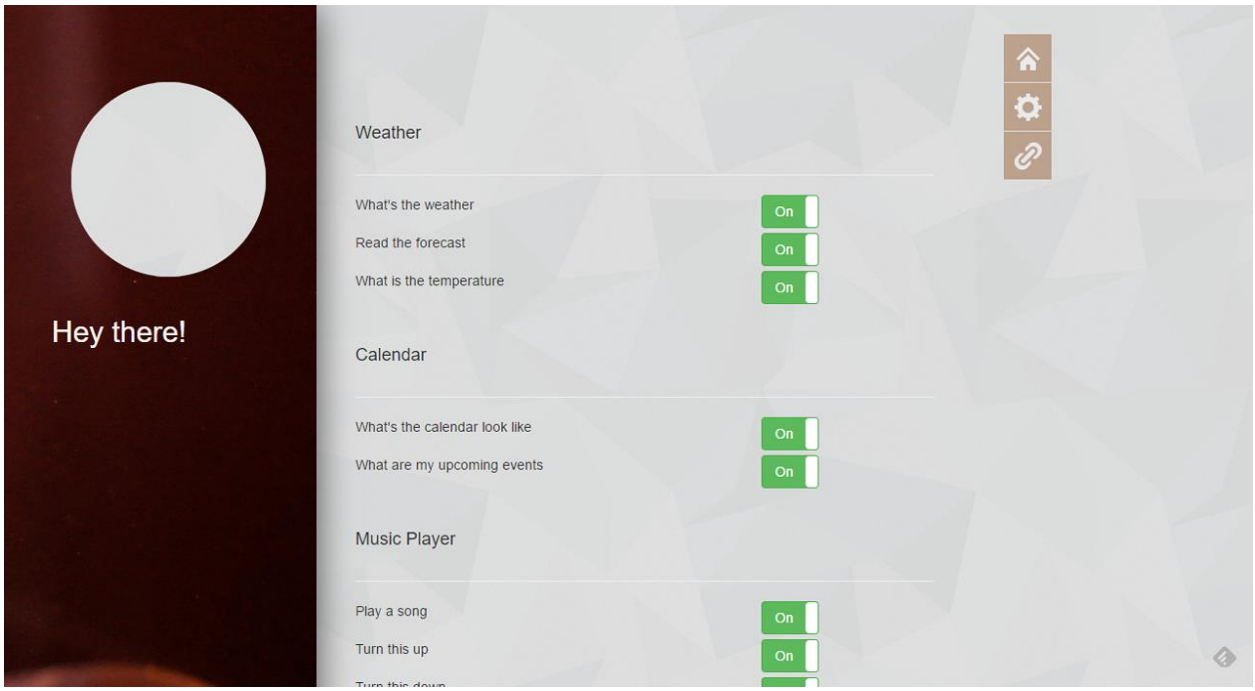

Рисунок 7.4 – Страница настроек голосовых команд для зеркала

Для удобства пользования стиль страницы схож со страницей настроек виджетов. Пользователь видит список всех команд. Около каждой команды находится кнопка включения/выключения. Интерфейс ее такой же, как

написано выше: серый цвет – команда скрыта для пользователя, яркий цвет – голосовая команда доступна для использования.

Веб-приложение содержит еще пару вспомогательных страниц, таких как информация о зеркале и страница для связи с разработчиком. Однако они не представляют особого интереса и рассмотрены не будут.

Интерфейс приложения создан максимально удобным и простым для рядового пользователя.

7.2 Аппаратная часть

В данном подразделе будет рассмотрен интерфейс взаимодействия с пользователем посредством самого устройста – смарт-зеркала с голосовым управлением.

На рисунке 7.5 представлен интерфейс зеркала при всех включенных виджетах.

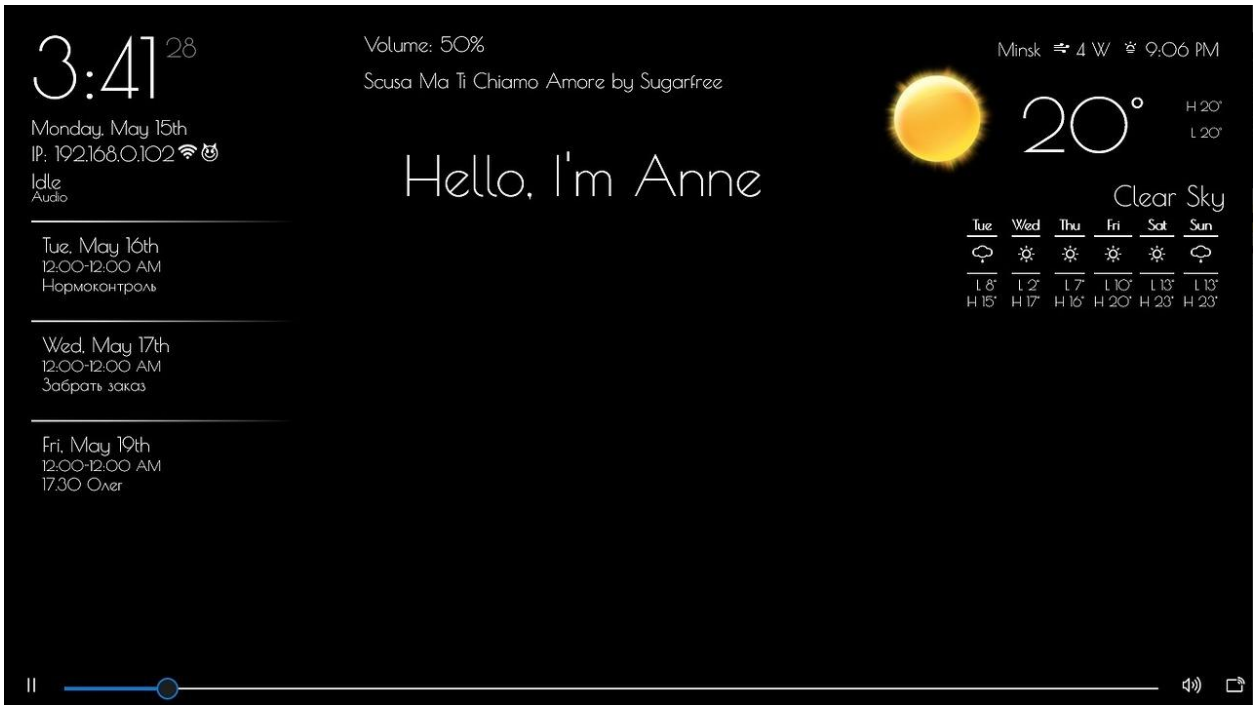

Рисунок 7.5 – Интерфейс смарт-зеркала

Поговорим о каждом виджете отдельно.

7.2.1 Виджет часов

Виджет часов отображает текущее время, день недели, число и месяц. Данный виджет является стандартным для пользователя и отображается всегда.

Он является весьма простым для понимая. Обновление виджета происходит каждую секунду. Формат отображения следующий: часы, минуты, секунды и кроме того названия дня недели, месяц и текущее число.

7.2.2 Виджет погоды

Виджет погоды является встроенным и пользователь может скрыть его через веб-приложение по настройке зеркала.

На данном виджете отображается следующая информация:

1. Название города, для которого показана погода.
2. Время восхода и захода солнца на текущий день.
3. Текущая температура.
4. Прогноз на несколько следующих дней.

Также пользователь может выбрать что именно ему отображать на зеркале: состояние текущей погоды, прогноз на несколько дней, либо вся информация одновременно.

Текущая погода и прогноз обновляются каждые пятнадцать минут.

7.2.3 Виджет событий календаря

Данный виджет также является встроенным и пользователь может скрыть его через веб-приложение по настройке зеркала.

Информацию, размещенная на виджете следующая:

1. Сокращенное название дня недели, месяца и число события.
2. Время события.
3. Информация о событии.

Располагаются события по возрастанию, а информация подтягивается по закрытому урлу календаря пользователя.

7.2.4 Виджет музыкального плеера

На данный момент музыка для плеера не может подтягиваться с какого-то интернет-ресурса или синхронизироваться с телефоном для проигрывания. Это еще предстоит сделать. Однако на данный момент можно прослушать ограниченный набор песен, который добавлен в программу для зеркала. В будущем будет добавлена возможность интеграции с таким веб-сервисом как Яндекс Музыка.

Данный виджет также является встроенным и пользователь может скрыть его через веб-приложение по настройке зеркала.

7.2.5 Виджет эмоций

Виджет эмоций представляет собой индикатор эмоции пользователя, а также текстовое сообщение об эмоции.

7.2.6 Голосовые команды

Для пользователя доступен ряд голосовых команд, которые можно включать и выключать по необходимости из веб-приложения по настройке зеркала.

Взаимодействие с пользователем максимально упрощено. Для удобства использования и понимания текущего статуса зеркала оно оснащено виджетом состояния голосового помощника. Например, зачастую пользователь сможет увидеть такую фразу как «I’m listening…», ведь зеркало

всегда прослушивает аудиопоток пользователя и готово выполнять его команды.

Также под виджетом состояния голосового помощника можно увидеть либо текущую выполняемую задачу, либо одну из последних выполненных. Например, отображение текста «Audio», показывает, что сейчас проигрывается песня. Либо, что последняя выполненная задача была по воспроизведению музыки.

Список доступных команд по управлению:

1. «What’s the weather» для получения информации о текущей погоде.
2. «Read the forecast» для получения информации о погоде на несколько дней.
3. «What is the temparature» для получения информации о температуре.
4. «What’s the calendar lool like» или «What are my upcoming events» для получения информации о предстоящих событий из календаря пользователя.
5. «Play a song» для проигрывания произвольной песни, а также «Turn this up», «Turn this down» или «Mute» для регулировки громкости.
6. «How do I look» для работы с виджетом эмоций.
7. «Help» для отображения и прослушивания всех команд управления.
8. «Close» для безопасного завершения работы приложения.

7.2.7 Итоги

Помимо всего уже описанного на зеркале присутствует блок для отображения текста. На нем может быть изображено приветственное сообщение, либо какая-то ответная информация зеркала на запрос пользователя.

Также около под виджетом часов, около виджета эмоций, можно увидеть текущий ip-адрес зеркала, если это необходимо для соединения со сторонним устройством и мощность текущего wi-fi соединения.

По итогу взаимодействие зеркала с пользователем старалось быть спроектировано максимально удобно, не граможденно и информативно.

8 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА СМАРТ-ЗЕРКАЛА С ГОЛОСОВЫМ УПРАВЛЕНИЕМ

Сегодня программно-аппаратные продукты являются продукцией реализуемой покупателем по рыночным отпускным ценам. Широкое применение вычислительной техники требует постоянного обновления и совершенствования программных средств. Выбор эффективных проектов программных средств связан с их экономической оценкой и расчетом экономического эффекта. Поэтому разрабатываемые системы должны быть не только совершенными в техническом плане, но и выгодными с точки зрения экономики.

Целью технико-экономического обоснования программных средств является определение экономической выгодности создания данного продукта и дальнейшего его применения.

8.1 Описание функций, назначения и потенциальных пользователей программно-аппаратного комплекса

Целью дипломного проекта является разработка программно-аппаратного комплекса смарт-зеркало с голосовым управлением. Причиной разработки данного комплекса стала необходимость получения мгновенно информации и удобства управления сторонними сервисами.

В последнее время складывается тенденция к получению информации с использованием интернет-ресурсов с различных гаджетов. Однако для этой цели необходимо посещать множество ресурсов: заходить на новостные сайты, на почту, календарь, мессенджеры, социальные сети, музыкальные сайты. Тем не менее, все это можно собрать в одном месте.

В рамках дипломного проекта разрабатывается смарт-зеркало, способное получать данные с различных интернет ресурсов, управлять личными данными и сторонними устройствами. В частности стоят задачи показывать информативные блоки, включать подсветку при распознавании человека, а также возможность управлять устройством при помощи голосовых команд. Для удобства управления данным устройством разработать приложение-интерфейс, с помощью которого можно настроить зеркало под конкретного человека и его нужды.

Данный проект позволит значительно снизить временные затраты человека, значительно увеличит его производительность. Потенциальным пользователем, как уже понятно, может быть любой человек, желающий получить многофункциональное устройство. Следовательно, производится разработка программно-аппаратного комплекса для свободной реализации на рынке. Так как смарт зеркало управляется голосовыми командами, то потенциальными пользователями могу являться люди с ограниченными фичическими способностями.

Управление голосовыми командами является более естественным, чем любые интерактивные меню. Оно легко в использовании и будет доступно для людей, котором сложно освоить современные интерфейсы, т.к. для управление не нужно ничего кроме естественных голосовых команд. Таким образом любой человек может получить интересующую его информацию без каких-либо усилий и очень быстро.

Для оценки экономической эффективности разработанного программно-аппаратного проекта проводится расчет затрат на разработку системы, оценка прибыли от продажи одной такой системы и расчет показателей эффективности инвестиций в разработку программно-аппаратного комплекса.

Расчеты выполнены на основе методического пособия.

При расчете сметы затрат будут использоваться данные, приведенные в таблице 8.1.

Таблица 8.1 – Исходные данные для расчета

Наименование показателей	Буквенные обозначения	Единицы измерения	Количество
Фонд социальной защиты населения(от заработной платы)	$Н_{соц}$	%	34
Обязательное страхование (от несчастных случаев на производстве, от заработной платы)	$Н_{стр}$	%	0,6
Налог на прибыль	$Н_{приб}$	%	18
Налог на недвижимость(от стоимости зданий и сооружений)	$Н_{недв}$	%	1
НДС (Налог на добавленную стоимость)	НДС	%	20
Норма дисконта	$Е_{н}$	%	17
Тарифная ставка 1-го разряда	$Т_{м1}$	руб	31
Часовая тарифная ставка 1-го разряда	$Т_{ч}$	руб	0,19
Установленный фонд рабочего времени	$Ф_{рв}$	часов	166
Продолжительность рабочего дня	$Т_{ч}$	часов.	8
Тарифный коэффициент	$Т_{к}$	-	2,03
Коэффициент премирования	$К_{П}$	единиц	1,3

8.2 Расчет затрат на разработку программно-аппаратного комплекса

Основой для расчёта сметы затрат является основная заработная плата разработчиков проекта. Затраты на основную заработную плату команды разработчиков определяются исходя из состава и численности команды, размеров месячной заработной платы каждого из участников команды, а также общей трудоемкости разработки программного обеспечения.

В данном случае имеются два работника – инженер-программист II-й категории (тарифный разряд – 7, тарифный коэффициент – 2,03) и руководитель проекта (тарифный разряд – 18, тарифный коэффициент – 4,26).

8.2.1 Основная заработная плата

Основная заработная плата исполнителей рассчитывается по формуле:

$$З_о = \sum_{i=1}^{n} Т_{чi} \cdot t_i, \quad (8.1)$$

где n – количество исполнителей, занятый разработкой конкретного ПО;

$Т_{чi}$ – часовая заработная плата i-го исполнителя (руб.);

t_i – трудоемкость работ, выполняемых i-м исполнителем (ч.).

Месячная тарифная ставка исполнителя ($Т_м$) определяется по формуле:

$$Т_м = Т_{м1} \cdot Т_к, \quad (8.2)$$

где $Т_{м1}$ – месячная тарифная ставка первого разряда, руб.;

$Т_к$ – тарифный коэффициент.

Исходя из месячной тарифной ставки рассчитывается часовая тарифная ставка ($Т_ч$):

$$Т_ч = \frac{Т_м}{Ф_р}, \quad (8.3)$$

где $Ф_р$ – среднемесячная норма рабочего времени, ч.

Учитывая число разработчиков n = 2, определим основную заработную плату по формуле (8.1):

$$З_о = (0{,}38 \cdot 320 + 0{,}79 \cdot 150) \cdot 1{,}3 = 312{,}13 \text{ руб.}$$

Результаты расчета основной заработной платы исполнителей представлена в таблице 8.2.

Таблица 8.2 – Расчет основной заработной платы исполнителей

№	Участник команды	Тарифный коэффициент	Месячная заработная плата, р.	Часовая заработная плата, р.	Трудоёмкость работ, ч.	Основная заработная плата, р.
1	2	3	4	5	6	7
1	Инженер-программист	2,03	62,93	0,38	320	121,6
2	Руководитель проекта	4,26	132,06	0,79	150	118,5
Итого, р.				240,1		
Премия, р.				72,03		
Основная заработная плата ($З_о$), р.				312,13		

8.2.2 Дополнительная заработная плата

Дополнительная заработная плата ($З_д$) включает в себя оплаты отпусков и другие выплаты, предусмотренные законодательством, и определяется по формуле:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (8.4)$$

где $Н_д$ – норматив дополнительной заработной платы (15%).

Тогда получим:

$$З_д = \frac{312{,}13 \cdot 15}{100} = 46{,}82 \text{ руб.}$$

8.2.3 Отчисления на социальные нужды

Отчисления в фонды социальной защиты и социального страхования определяются по следующей формуле:

$$З_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (8.5)$$

где $Н_{соц}$ – норматив отчислений на социальные нужды (34%);

Отчисления в фонд социальной защиты – 34%, отчисления в фонд социального страхования – 0,6%. Исходя из этого, получаем, 34 + 0,6 = 34,6%

По формуле (8.5) получим:

$$З_{соц} = \frac{(312{,}13 + 46{,}82) \cdot 34{,}6}{100} = 124{,}2 \text{ руб.}$$

8.2.4 Прочие затраты

Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по формуле:

$$З_{пз} = \frac{З_о \cdot Н_{пз}}{100}, \quad (8.6)$$

где $Н_{пз}$ – норматив прочих затрат в целом по организации (20%).

$$З_{пз} = \frac{312{,}13 \cdot 20}{100} = 62{,}43 \text{ руб.}$$

Полная сумма затрат на разработку программно-аппаратного комплекса находится путем суммирования всех рассчитанных статей затрат. Все расчеты сведены в таблицу 8.3.

Таблица 8.3 – Результаты расчетов и полная сумма затрат

Статья затрат	Сумма, руб.
Основная заработная плата команды разработчиков	312,13
Дополнительная заработная плата команды разработчиков	46,82
Отчисления на социальные нужды	124,2
Прочие затраты	62,43
Общая сумма затрат на разработку	545,58

8.3 Оценка результата от использования программно-аппаратного комплекса

Рассчитаем экономический эффект при разработке программно-аппаратного комплекса для свободной реализации на рынке.

Экономический эффект заключается в получении прибыли от его продажи множеству потребителей. Прибыль от реализации в данном случае

напрямую зависит от объемов продаж, цены реализации и затрат на разработку данного комплекса.

Предполагаемый объем продаж за год возьмем пять копий. Прогнозируюиую цену возьмем 269 руб., из расчета рентабельности одного продукта в 49%.

При расчёте прибыли от продажи дополнительно учитывается налог на добавочную стоимость (НДС):

$$НДС = \frac{Ц_п \cdot Н_{дс}}{100 + Н_{дс}}, \quad (8.\ 7)$$

где $Н_{дс}$ – норматив налога на добавленную стоимость (20%).

По формуле (8.7) налог на добавочную стоимость равен:

$$НДС = \frac{269 \cdot 20}{100 + 20} = 44{,}83 \text{ руб.}$$

Прибыль от продажи одной копии осуществляется по формуле:

$$П_{ед} = Ц - НДС - \frac{З_р}{N}, \quad (8.8)$$

где Ц – предполагаемая цена (руб.);

НДС – налог на добавочную стоимость (руб.);

$З_р$ – затраты на разработку и реализацию (руб.);

N – количество копий, которое будет куплено за год.

Затраты на реализацию примем 5% от расходов на разработку комплекса.

Тогда получим:

$$П_{ед} = 269 - 44{,}83 - \frac{545{,}58 + 545{,}58 \cdot 0{,}05}{5} = 109{,}57 \text{ руб.}$$

Годовая прибыль по проекту определяется по формуле:

$$П = П_{ед} \cdot N, \quad (8.9)$$

где $П_{ед}$ – прибыль от продажи одной копии (руб.);

N – количество копий, которое будет куплено за год.

$$П = 109{,}57 \cdot 5 = 547{,}85 \text{ руб.}$$

Рентабельность затрат рассчитаем по формуле 8.10. Проект будет экономически эффективным, если рентабельность затрат на разработку программно-аппаратного косплекса будет не меньше средней процентной ставки по банковским депозитным вкладам.

$$Р = \frac{П}{З_р} \cdot 100\%, \quad (8.10)$$

где П – годовая прибыль (руб.);

$З_р$ – сумма расходов на разработку и реализацию (руб.).

$$Р = \frac{547{,}85}{545{,}58 + 545{,}58 \cdot 0{,}05} \cdot 100\% = 95\ \%.$$

Учитывая налог на прибыль, можно рассчитать итоговую сумму, которая останется разработчику и будет является его экономическим эффектом:

$$ЧП = П - \frac{П \cdot Н_{приб}}{100}, \quad (8.11)$$

где ЧП – чистая прибыль;

П – прогнозируемая прибыль;

$Н_{приб}$ – норматив налога на прибыль (18%).

Подставив значения в формулу (8.11), определим чистую прибыль:

$$ЧП = 547{,}85\ - \frac{547{,}85\ \cdot 18}{100} = 449{,}24 \text{ руб.}$$

Чистая прибыль от реализации программно-аппаратного комплекса (ЧП = 449,24 рублей) остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства.

8.4 Расчет показателей эффективности инвестиций в разработку программно-аппаратного комплекса

Сравним размер инвестиций в разработку программного продукта и получаемый годовой экономический эффект.

Сумма инвестиций (545,58 руб.) меньше, чем сумма годового эффекта (547,85 руб.), это означает, что инвестиции окупятся менее чем за год. На основе этого рассчитаем рентабельности инвестиций по формуле:

$$Р_{и} = \frac{П_{ч}}{С_{п}} \cdot 100\%, \quad (8.\ 12)$$

где $П_{ч}$ – чистая прибыль (руб.);

$С_{п}$ – сумма затрат на разработку (руб.).

По формуле (8.12) рентабельность инвестиций равна:

$$Р_{и} = \frac{449{,}24}{545{,}58} \cdot 100\% = 82{,}34\ \%.$$

В данном разделе была рассчитана себестоимость проекта, на основании которой была сформирована рыночная цена. Был рассчитан экономический эффект для разработчика: чистая прибыль от реализации программно-аппаратного комплекса составила 449,24 рублей.

Кроме этого, была рассчитана экономическая эффективность для заказчика, выраженная в рентабельности инвестиций 82,34%. Стоит отметить, что затраты заказчика окупятся менее, чем за год.

Таким образом, учитывая снижение трудоёмкости, значительную экономию затрат, достаточно быстрый срок окупаемости, можно сделать вывод, что проект является полезным и экономически эффективным для заказчика.

ЗАКЛЮЧЕНИЕ

В ходе дипломного проектирования был разработан программно-аппаратный комплекс «Смарт-зеркало с голосовым управлением». Были рассмотрены существующие аналоги, затронута тема микроконтроллеров и их использование.

Для проектирования «Смарт-зеркало с голосовым управлением» был выбран миникомпьютер Raspberry Pi, исходя из его достоинств и так как он является более оптимальным решением для проекта, среда разработки Fritzing, содержащая необходимые для построения схемы компоненты, а также для прошивки самой платы.

В качестве среды разработки для интерфейсного модуля была выбрана технология Microsoft .NET, в частности ее часть ASP.NET MVC. Для хранения данных была выбрана СУБД Microsoft SQL Server 2014. Для моделирования пользовательского интерфейса был выбран графический движок Razor. Для верстки клиентской части используется набор инструментов для создания веб-сайтов Bootstrap.

Аппаратная часть реализована на UWP (Universal Windows Platform) – унифицированная платформа для создания и запуска приложений в Windows 10, а также была использована система Windows IoT Core. В процессе работы над проектом также были изучены библиотеки Windows Speech Recognition для работы с голосовым помощником и распознавания голосовых команд, Azure Emotion API для распознавания эмоций человека.

Разработанная система предоставляет следующие возможности:

1. Отображение виджетов часов, погоды, событий календаря.
2. Виджет музыкального плеера.
3. Распознавание эмоций человека и соответствующий виджет.
4. Управление освещением с помощью голосовых команд.
5. Голосовые команды для управления системой.
6. Локальное веб-приложение для настройки виджетов и команд управления.

В дальнейшем планируется добавить возможность интеграции зеркала и встроенного в него миникомпьютера Raspberry Pi 3b с фитнес-браслетом Mi Band 2 по Bluetooth интерфейсу для отображения графиков дневной деятельности человека и его сна, а также с микроконтроллером Arduino, который будет посылать данные со счетчиков воды при помощи фотографий для удобного пользования и подачи коммунальных платежей по квартире.

СПИСОК ЛИТЕРАТУРЫ

[1] Смарт-зеркало Griffin Connected Mirror [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.cnet.com/products/griffin-technology-connected-mirror/preview/

[2] Смарт-зеркало Perseus [Электронный ресурс]. – Электронные данные. – Режим доступа: https://tehnobzor.ru/smart-zerkalo-perseus/

[3] Смарт-зеркало компании Lenovo [Электронный ресурс]. – Электронные данные. – Режим доступа: https://3dnews.ru/940499

[4] Смарт-зеркало самодельный вариант [Электронный ресурс]. – Электронные данные. – Режим доступа: http://24gadget.ru/1161062626-samodelnoe-smart-zerkalo-6-foto-video.html

[5] Белов, А. В. Самоучитель по микропроцессорной технике / А. В. Белов – [2-е изд.] – СПб.: Наука и техника, 2007. – 240 с.

[6] Белов, А. В. Создаем устройства на микроконтроллерах / А. В. Белов – СПб.: Наука и техника, 2007. — 304 с.

[7] Семенов, Б. Ю. Микроконтроллеры MSP430: первое знакомство / Б. Ю. Семенов– М.:СОЛОН-ПРЕСС, 2006. – 128 с.

[8] Соммер, У. Программирование микроконтроллерных плат Arduino/Freeduino / У. Соммер – БХВ-Петербург, 2012. – 256 с.

[9] Магда, Ю. С. Raspberry Pi. Руководство по настройке и применению / Ю. С. Магда – ДМК Пресс, 2014. – 188 с.

[10] Шилдт, Г. Полный справочник по C#.: / Г. Шилдт – М.: Издательский дом «Вильямс», 2004. – 752 с.

[11] Рихтер, Дж. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Дж. Рихтер – [3-е изд.] – М.: Издательский дом «Питер», 2012. – 928 с.

[12] Троелсен, Э. Язык программирования C# 2010 и платформа .NET4/ Э. Троелсен – [5-е изд.] – М.: Издательский дом «Вильямс», 2011. – 1392 с.

[13] Фримен, А. ASP.NET MVC 3 Framework с примерами на C# для профессионалов / А. Фримен, С. Сандерсон – [3-е изд.] – М.: Издательский дом «Вильямс», 2011. – 672 с.

[14] Документация Windsor Castle [Электронный ресурс]. – Электронные данные. – Режим доступа: http://docs.castleproject.org/Windsor.MainPage.ashx

[15] Документация Entity Framework [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.entityframeworktutorial.net/

[16] Документация движка Razor [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.asp.net/web-pages/tutorials/basics/2-introduction-to-asp-net-web-programming-using-the-razor-syntax

[17] Документация фреймворка Bootstrap [Электронный ресурс]. – Электронные данные. – Режим доступа: http://bootstrap-ru.com/

ПРИЛОЖЕНИЕ А
(*обязательное*)
Исходный текст класса `CommandsInterpreter`

```
using System;
using System.Linq;
using Anne.Extensions;
using Anne.Speech.Interfaces;

namespace Anne.Speech.Commands
{
    class CommandsInterpreter : ICommandsInterpreter
    {
        static string[] Days =>
Enum.GetNames(typeof(DayOfWeek));

        static bool StartsWith(string phrase, string match)
=> phrase.StartsWith(match, StringComparison.OrdinalIgnoreCase);

        static bool StartsWithAny(string phrase, params
string[] matches) => matches?.Any(match => StartsWith(phrase,
match)) ?? false;

        static bool Contains(string phrase, string match) =>
phrase.ContainsIgnoringCase(match);

        static bool ContainsAny(string phrase, params
string[] matches) => matches?.Any(match => Contains(phrase,
match)) ?? false;

        CommandsContext
ICommandsInterpreter.GetPhraseIntent(string phrase)
        {
            if (string.IsNullOrWhiteSpace(phrase))
            {
                return CommandsEnum.None;
            }

            if (phrase.StartsWith("Help",
StringComparison.OrdinalIgnoreCase) || Contains(phrase, "What
can I say"))
            {
                return CommandsEnum.Help;
            }

            if (phrase.StartsWith("Close",
StringComparison.OrdinalIgnoreCase))
            {
                return CommandsEnum.Close;
            }
```

```
                if (StartsWith(phrase, "Look") ||
                    Contains(phrase, "How do I look") ||
                    Contains(phrase, "Tell me how I look"))
                {
                    return CommandsEnum.Emotion;
                }

                if (StartsWithAny(phrase, "What", "Read", "How",
"On"))
                {
                    if (ContainsAny(phrase, "event",
"calendar"))
                    {
                        DateTime? dateContext;
                        if (Contains(phrase, "on") &&
TryParseContext(phrase, out dateContext))
                        {
                            return new
CommandsContext(CommandsEnum.CalendarEvents, dateContext);
                        }
                        return CommandsEnum.CalendarEvents;
                    }

                    if (Contains(phrase, "forecast"))
                    {
                        return CommandsEnum.ForecastWeather;
                    }

                    if (ContainsAny(phrase, "temp", "weather"))
                    {
                        DateTime? dateContext;
                        if (Contains(phrase, "on") &&
TryParseContext(phrase, out dateContext))
                        {
                            return new
CommandsContext(CommandsEnum.ForecastWeather, dateContext);
                        }
                        return CommandsEnum.CurrentWeather;
                    }
                }
                else if (ContainsAny(phrase, "turn", "mute",
"volume", "loud", "quiet"))
                {
                    return CommandsEnum.Volume;
                }
                else if (Contains(phrase, "play"))
                {
                    return CommandsEnum.Audio;
                }

                return CommandsEnum.Dictation;
            }
```

```
            static bool TryParseContext(string phrase, out
DateTime? dateContext)
            {
                dateContext = null;
                foreach (var day in Days.Where(day =>
Contains(phrase, day)))
                {
                    dateContext =
DateTime.Now.Next(day.ToEnum<DayOfWeek>());
                    return true;
                }

                if (Contains(phrase, "today"))
                {
                    dateContext = DateTime.Now;
                    return true;
                }

                if (Contains(phrase, "tomorrow"))
                {
                    dateContext = DateTime.Now.AddDays(1);
                    return true;
                }

                return false;
            }
        }
    }
```

ПРИЛОЖЕНИЕ Б
(*обязательное*)
Смарт-зеркало с голосовым управлением. Спецификация

ПРИЛОЖЕНИЕ В
(*обязательное*)
Смарт-зеркало с голосовым управлением. Ведомость документов