**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл

Факультет компьютерных наук

Департамент программной инженерии

СОГЛАСОВАНО

Профессор департамента

программной инженерии факультета компьютерных наук, кандидат физико-математических наук

\_\_\_\_\_\_\_\_\_\_\_\_ И.Р. Агамирзян

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», профессор департамента программной инженерии, кандидат технических наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**Беспроводная система управления домашними устройствами на платформе Android и Arduino**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.505900-01 81 01-1-ЛУ**

Исполнитель студент группы БПИ 161

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /К.М. Головко/

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**2017**

УТВЕРЖДЕН RU.17701729.505900-01 81 01-1-ЛУ

**Беспроводная система управления домашними устройствами на платформе Android и Arduino**

**Пояснительная записка**

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл

**RU.17701729.505900-01 81 01-1**

**Листов 44**

**2017**

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ 4](#_Toc482223040)

[1.1 Наименование программы 4](#_Toc482223041)

[1.2 Основания для разработки 4](#_Toc482223042)

[1.3 Стандарты документации 4](#_Toc482223043)

[2. НАЗНАЧЕНИЕ ПРОГРАММЫ И ОБЛАСТЬ ЕЕ ПРИМЕНЕНИЯ 5](#_Toc482223044)

[2.1 Назначение 5](#_Toc482223045)

[2.2 Область применения 5](#_Toc482223046)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc482223047)

[3.1 Постановка задачи 6](#_Toc482223048)

[3.2 Описание алгоритма и функционирования программы 6](#_Toc482223049)

[Смартфон и управляющее устройство общаются по протоколу, описанному в приложении 3. 8](#_Toc482223050)

[3.3 Описание и обоснование способов предоставления входных и выходных данных 9](#_Toc482223051)

[3.3.1 Данные части программы на смартфоне. 9](#_Toc482223052)

[3.3.2 Данные части программы на микроконтроллере Arduino. 9](#_Toc482223053)

[3.4 Выбор состава технических средств и его обоснование. 9](#_Toc482223054)

[3.4.1 Технические средства для выполнения части на смартфоне и их обоснование. 9](#_Toc482223055)

[3.4.2 Технические средства, необходимые для запуска части программы на микроконтроллере Arduino Uno и их обоснование. 10](#_Toc482223056)

[3.5 Выбор программных средств и языков программирования и его обоснование. 10](#_Toc482223057)

[3.5.1 Выбор программных средств и языка для части программы на смартфоне. 10](#_Toc482223058)

[3.5.2 Выбор программных средств и языка для части программы на микроконтроллере Arduino Uno. 11](#_Toc482223059)

[3.6 Выбор способов и алгоритмов решения, их подробное описание и обоснование. 11](#_Toc482223060)

[3.6.1 Способы и алгоритмы решения применяемые для модуля, выполняемого на смартфоне 11](#_Toc482223061)

[3.6.1 Способы и алгоритмы решения применяемые для модуля, выполняемого на микроконтроллере 16](#_Toc482223062)

[4 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 18](#_Toc482223063)

[4.1 Предположительная потребность 18](#_Toc482223064)

[4.2 Преимущества разработки в сравнении с аналогами (отечественными и зарубежными) 18](#_Toc482223065)

[5 ИСТОЧНИКИ 19](#_Toc482223066)

[ПРИЛОЖЕНИЕ 1 ОИСАНИЕ И НАЗНАЧЕНИЕ КЛАССОВ 20](#_Toc482223067)

[КЛАССЫ ДЛЯ ЧАСТИ НА СМАРТФОНЕ 20](#_Toc482223068)

[КЛАССЫ IRemote 20](#_Toc482223069)

[ИНТЕРФЕЙСЫ IRemote 21](#_Toc482223070)

[КЛАССЫ IRemote.Droid 21](#_Toc482223071)

[КЛАССЫ ДЛЯ ЧАСТИ НА УПРАВЛЯЮЩЕМ УСТРОЙСТВЕ 21](#_Toc482223072)

[ПРИЛОЖЕНИЕ 2 НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ, СВОЙСТВ, СОБЫТИЙ КЛАССОВ 22](#_Toc482223073)

[КЛАССЫ IRemote 22](#_Toc482223074)

[Класс RemotesDatabase: 22](#_Toc482223075)

[Класс DefaultRemote: 23](#_Toc482223076)

[Класс Remote: 23](#_Toc482223077)

[Класс RemoteButton: 24](#_Toc482223078)

[Класс RemoteToSave: 25](#_Toc482223079)

[Класс IRSignal: 25](#_Toc482223080)

[Класс App.xaml 26](#_Toc482223081)

[Класс ConnectionPage.xaml 27](#_Toc482223082)

[Класс EditPage.xaml: 28](#_Toc482223083)

[Класс HelpPage.xaml 31](#_Toc482223084)

[Класс HomePage.xaml 31](#_Toc482223085)

[Класс MainPage.xaml 32](#_Toc482223086)

[Класс CategorySelectedEventArgs : EventArgs 33](#_Toc482223087)

[Класс MasterPage.xaml 33](#_Toc482223088)

[Класс RecivePage.xaml 35](#_Toc482223089)

[Класс RemotePage.xaml 35](#_Toc482223090)

[Класс SelectionPage.xaml: 36](#_Toc482223091)

[Класс TabbedRemote.xaml: 37](#_Toc482223092)

[ИНТЕРФЕЙСЫ IRemote 37](#_Toc482223093)

[Интерфейс IBlueConnection: 37](#_Toc482223094)

[Интерфейс IDbFileProvider: 38](#_Toc482223095)

[Интерфейс IToastMaker: 38](#_Toc482223096)

[КЛАССЫ IRemote.Droid 38](#_Toc482223097)

[Класс BlueConnection:IBlueConnection: 38](#_Toc482223098)

[Класс DbFileProvider:IDbFileProvider: 39](#_Toc482223099)

[Класс MakeToast:IMakeToast 40](#_Toc482223100)

[Класс MainActivity: 40](#_Toc482223101)

[КЛАССЫ НА УПРАВЛЯЮЩЕМ УСТРОЙСТВЕ 40](#_Toc482223102)

[CW.ino 40](#_Toc482223103)

[Класс Trigger.cpp 41](#_Toc482223104)

[ПРИЛОЖЕНИЕ 3 ПРОТОКОЛ ПЕРЕДАЧИ ДАННЫХ 42](#_Toc482223105)

[ПИЛОЖЕНИЕ 4 ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ 43](#_Toc482223106)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 44](#_Toc482223107)

# 1. ВВЕДЕНИЕ

## 1.1 Наименование программы

Программа называется: “Беспроводная система управления домашними устройствами на платформе Android и Arduino”.

## 1.2 Основания для разработки

Документом, на основании которого ведется разработка, является приказ Национального исследовательского университета "Высшая школа экономики" №2.3-02/0812-01 от 08.12.2016. Разработка выполняется в рамках темы курсовой работы «Беспроводная система управления домашними устройствами на платформе Android и Arduino» в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

## 1.3 Стандарты документации

Документ разработан в соответствии с ГОСТ и ЕСПД [1-9]

# 2. НАЗНАЧЕНИЕ ПРОГРАММЫ И ОБЛАСТЬ ЕЕ ПРИМЕНЕНИЯ

## 2.1 Назначение

Программа предназначена для управления домашними устройствами с инфракрасным управлением с помощью виртуальных пультов, создаваемых в программе и специального устройства управления, описанного в техническом задании п4.5 .

## 2.2 Область применения

Программа создана для пользователей, имеющих большое количество управляемых устройств в одном помещении. Она позволяет с легкостью заменить большое количество инфракрасных пультов одним устройством управления и одним смартфоном под управлением ОС Android со встроенным модулем Bluetooth. Имеется возможность быстрого переключения между виртуальными пультами и их редактирования.

# 3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

## 3.1 Постановка задачи

Разработать программу, состоящую из двух модулей. Первый модуль выполняется на смартфоне под управлением ОС Android. Второй модуль выполняется на устройстве управления на базе микроконтроллера Arduino Uno (устройство управления описано в п 4.5 технического задания). Первый модуль должен уметь отправлять коды для передачи управляемым устройствам на управляющее устройство при нажатии соответствующей кнопки виртуального пульта, создавать виртуальные пульты, редактировать их, разбивать по категориям, принимать код инфракрасного сигнала от управляющего устройства в режиме редактирования виртуального пульта, сохранять виртуальные пульты. Второй модуль должен передавать принятый от смартфона сигнал по инфракрасному каналу управляемому устройству, передавать в режиме редактирования виртуального пульта коды, принятые от реального пульта. Общение между модулями происходит при помощи аппаратно-программного протокола Bluetooth.

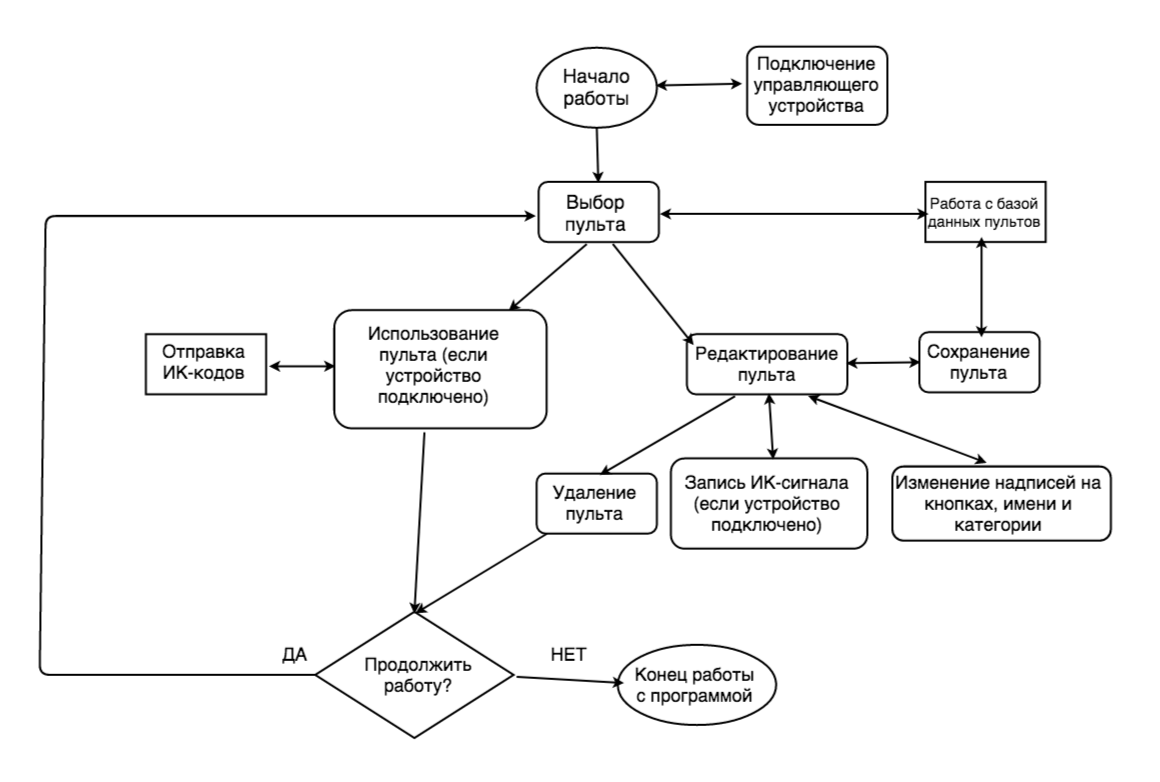
Задача поставлена в соответствии с приказом Национального исследовательского университета "Высшая школа экономики" №2.3-02/0812-01 от 08.12.2016. Программа разрабатывается в рамках курсовой работы и в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия»

## 3.2 Описание алгоритма и функционирования программы

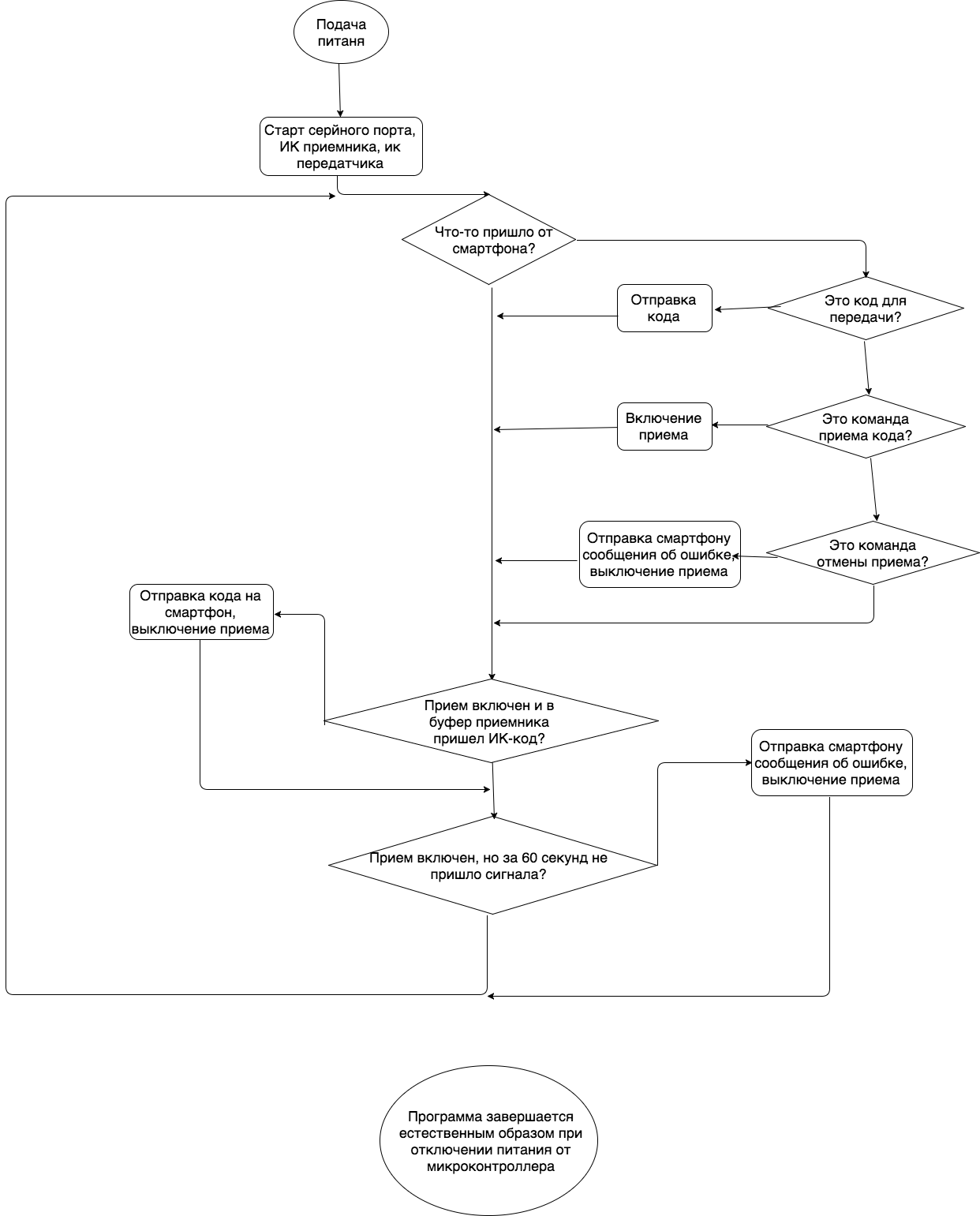
Работу программы на смартфоне можно разделить на следующие основные этапы:

1. Подключение к управляющему устройству.
2. Выбор пульта (загрузка его из базы данных)
3. Использование пульта (Отправка инфракрасного сигнала)
4. Редактирование пульта (Запись инфракрасного сигнала, изменение имени и категории, надписей на кнопках)
5. Создание пульта
6. Удаление пульта
7. Отключение от управляющего устройства и завершение работы программы

и представить в виде следующей схемы:



Часть программы на управляющем устройстве имеет более простую логику, которая заключается в следующей схеме:



## Смартфон и управляющее устройство общаются по протоколу, описанному в приложении 3.

## 3.3 Описание и обоснование способов предоставления входных и выходных данных

Смартфон и управляющее устройство обмениваются данными согласно протоколу, описанному в Приложении 3.

### 3.3.1 Данные части программы на смартфоне.

Входными данными части на смартфоне являются инфракрасные коды принятые от управляющего устройства, а так же названия и категории пультов, надписи на кнопках с пользовательскими надписями.

Выходными данными являются передаваемые управляющему устройству команды и визуальный интерфейс для управления программой.

### 3.3.2 Данные части программы на микроконтроллере Arduino.

Входными данными для части программы, выполняемой на микроконтроллере Arduino являются принятые по каналу Bluetooth команды и принятые по инфракрасному каналу коды от реальных пультов управления.

Выходными данными являются инфракрасные коды, передаваемые смартфону, а так же передаваемые по инфракрасному каналу коды управления.

## 3.4 Выбор состава технических средств и его обоснование.

Для установки программы на смартфон и микроконтроллер требуется персональный компьютер со следующими системными характеристиками:

1. Установленная ОС Windows 7 или старше
2. Установленная среда Arduino
3. Наличие USB
4. Наличие CD-ROM
5. Клавиатура, мышь, монитор с разрешением не менее 800x600 точек

Обоснование:

1. Необходима для правильного подключения к смартфону и Arduino
2. Необходима для прошивки программы на Arduino
3. Необходим для подключения смартфона и Arduino
4. Необходим для открытия диска с программой
5. Для комфортной работы с персональным компьютером

### 3.4.1 Технические средства для выполнения части на смартфоне и их обоснование.

Для выполнения модуля для смартфона необходим смартфон со следующими характеристиками:

1. Аппаратная поддержка ОС Android старше 4.0.3
2. Процессор с частотой не менее 1 ГГц
3. Не менее 100 мегабайт свободного места ОЗУ
4. Не менее 300 мегабайт свободного места, доступного для установки программ
5. Наличие модуля Bluetooth.

Обоснование требований:

1. Требование Xamrin.Forms[12]
2. Для стабильной работы приложения и обеспечения приемлемого времени выполнения скрытых от пользователя процессов
3. Выявлено путем тестирования
4. Выявлено путем тестирования и исследования занимаемого места
5. Иначе нет возможности подключения к управляющему устройству

### 3.4.2 Технические средства, необходимые для запуска части программы на микроконтроллере Arduino Uno и их обоснование.

Необходим микроконтроллер Arduino Uno или аналогичный со следующими подключенными устройствами:

1. К стандартному серийному порту должен быть подключен Bluetooth модуль НС-05 или HC-06 или аналогичный для обеспечения связи со смартфоном.
2. К цифровому пину 3 должен быть подключен инфракрасный светодиод с соответствующими характеристиками питания для передачи управляющих сигналов.
3. К цифровому пину 11 должен быть подключен приемник инфракрасных сигналов с желаемой частотой приема для приема управляющих сигналов.

## 3.5 Выбор программных средств и языков программирования и его обоснование.

Смартфон и управляющее устройство обмениваются данными согласно протоколу, описанному в Приложении 3.

### 3.5.1 Выбор программных средств и языка для части программы на смартфоне.

Программа создана для смартфонов под управлением ОС Android версии, старше чем 4.0.3 (требование фреймворка Xamarin.Forms). Имеется возможность быстрого портирования программы на IOS и Windows Phone. Большая часть кода является кросплатформенной. Однако ввиду отсутствия устройств для отладки программа на данный момент существует только для Android. Программа не является ресурсоемкой и условия запуска на смартфоне ОС Android достаточно, чтобы судить о достаточной производительности смартфона для выполнения данной программы. В качестве языка программирования выбран язык C# (требование учебного плана) и кроссплатформенный фреймворк Xamarin.Forms, который является единственным приемлемым решением для запуска сложных C# программ на ОС Android.

Были использованы следующие сторонние библиотеки и пакеты:

1. JSON.NET – для хранения сериализованных объектов виртуальных пультов в памяти.
2. SQLite для .NET – реализующая базу данных SQLite для хранения виртуальных пультов.

### 3.5.2 Выбор программных средств и языка для части программы на микроконтроллере Arduino Uno.

Эта часть программы написана на языке С++, как на самом удобном для написания кода для микроконтроллеров Atmega, на котором построен Arduino Uno.

Были использованы следующие сторонние библиотеки и пакеты:

1. IRremote.h – для обеспечения простого доступа к инфракрасным интерфейсам.

## 3.6 Выбор способов и алгоритмов решения, их подробное описание и обоснование.

### 3.6.1 Способы и алгоритмы решения применяемые для модуля, выполняемого на смартфоне

Модуль, выполняющийся на смартфоне является стандартным решением Xamarin Studio (Обратно совместим с Microsoft Visual Studio) и, ввиду кросплатформенности приложения и наличия кросплатформенного и нативного кода, состоит из двух проектов. В проекте IRemote содержится кросплатформенный код, а в IRemote.Droid – нативный. Классы нативного кода при этом реализуют необходимые интерфейсы кросплатформенного кода и служат для реализации функционала, специфичного для каждой платформы. Конкретно в моем случае таковым функционалом является использование классов для Android, которые позволяют использовать встроенный модуль Bluetooth, работать с директорией приложения для хранения там файлов базы данных и выводить Toast-сообщения для пользователя. В качестве базы данных выбрана библиотека SQLlite для языка C#, которая автоматически реализует локальную базу данных в указанном файле. Эта библиотека свободно распространяется через менеджер пакетов Nu-Get.

В основном классе кросплатформенного кода - App.xaml имеется несколько важных для понимания работы программы статических свойств, которые видны из любого класса кросплатформенного кода и позволяют использовать один объект базы данных, подключения Bluetooth и класса для работы с Toast-сообщениями. Вот эти свойства:

* App.Databse – возвращает единственный объект базы данных, который единожды создается при запуске приложения.
* App.BlueCon – возвращает единственный объект подключения Bluetooth, который так же создается при запуске приложения и реализует весь функционал связанный с этим. При этом возвращается объект типа интерфейса IBlueConnection, который непосредственно связан с нативным кодом. Этот интерфейс в нативном коде реализован классом BlueConnection, который обеспечивает весь необходимый фукционал подключения (Отправка и прием сигнала, подключение к необходимому устройству и т.д.)
* App.ToastMaker – позволяет из любого метода любого класса вывести сообщение пользователю.

Выделим основные этапы и процессы, происходящие в модуле для смартфона:

1. Подключение к управляющему устройству
2. Хранение пульта в оперативной памяти устройства
3. Работа базы данных и обращение к ней
4. Выбор пульта для работы
5. Генерация визуального отображения пульта.
6. Отправка инфракрасного сигнала
7. Настройки пульта
8. Прием инфракрасного сигнала и его запись
9. Удаление пульта
10. Работа визуальной части приложения

1. Подключение к управляющему устройству.

Алгоритм подключения к устройству для работы является достаточно простым. Сначала при открытии пользователем ConnectionPage метод, срабатывающий при появлении этой страницы на экране смартфона, запрашивает через App.BlueCon список сопряженных со смартфоном устройств и выводит его пользователю. Если при этом Bluetooth выключен или отсутствует на устройстве под управлением Android, будут выведены соответствующие предупреждения о невозможности продолжить работу с программой. Когда пользователь выбирает устройство для подключения, страница указывает на это App.BlueCon и в этом классе устанавливается устройство, с которым работаем в дальнейшем. Когда пользователь нажимает кнопку подключения, управление передается асинхронному методу ConnectToSelectedDivice, который открывает подключение к выбранному устройству и открывает сокет для отправки и приема сигналов на управляющее устройство. Если по каким-то причинам сокет открыть не удалось – метод вернет логическое значение false, если подключение выполнено успешно, возвращается значение true. В любом случае, страница сообщает пользователю о результатах подключения. Страница так же проверяет, подключен ли смартфон к чему-либо и выводит сообщение о статусе снизу экрана. В случае нажатия пользователем клавиши отключения, сокет в App.BlueCon принудительно закрывается, и ему присваивается null.

2. Хранение пульта в оперативной памяти устройства.

В оперативной памяти устройства хранится пульт с помощью класса Remote. В этом классе хранится название пульта, его категория, идентификационный номер и список кнопок. Кнопка – это отдельный класс RemoteButoon в котором хранится надпись на кнопке, ее порядковый номер на панели пульта и инфракрасный сигнал, который соответствует данной кнопке. Инфракрасный сигнал так же является отдельным классом, следующая таблица отражает суть хранимой им информации:

|  |  |  |  |
| --- | --- | --- | --- |
| Название члена | Тип члена | Модификатор доступа | Смысл члена |
| Code | Автреализуемое свойство, uint | public | Хранит в себе непосредственно инфракрасный код, несущий управляющую информацию для управляемого устройства. |
| Length | Автреализуемое свойство, int | public | Хранит в себе длину кода |
| Adress | Автреализуемое свойство, int | public | Хранит в себе адрес, для которого предназначен сигнал. (Используется в некоторых пультах) |
| ArduinoIRemoteType | Автреализуемое свойство, int | public | Хранит в себе код пульта по версии библиотеки IRremote для Arduino (зависит от производителя реального пульта) [17] |
| ToString | Метод без параметров, string | public | Возвращает строковую запись в форме, приемлемой для передачи управляющему устройству: LLL@CCCCCCCCCC@TT@AAAAA, где LLL - длина, СССССССССС – код, ТТ – тип и ААААА – адрес. |

Таким образом виртуальный пульт хранится в оперативной памяти устройства.

3. Работа базы данных и обращение к ней.

База данных реализована библиотекой SQLite и классом RemotesDatabase, который позволяет получить все пульты сразу, список строк с названиями пультов, список строк с названиями категорий пультов, список пультов, соответствующих одной отдельной категории, сохранить в базе отдельный пульт, загрузить отдельный пульт по его идентификационному номеру, удалить пульт, передав его в качестве параметра соответствующему методу. Пульт в базе данных хранится в специальном, пригодном для хранении в базе данных классе. В нем содержится категория пульта строкой, идентификационный номер пульта числом и сериализованный с помощью JSON объект пульта. Категория пульта хранится отдельно для ускорения поиска подходящих под нее пультов. Выбор пал на JSON ввиду оптимального соотношения скорости работы, занимаемой памяти сереиализованным объектом и удобства пользования такой формой хранения. База данных подключается сразу при запуске приложения и используется по мере необходимости с помощью асинхронных запросов к ней. Все обращения к базе данных являются асинхронными и имеют выходной тип Task<T>, где Т – необходимый тип ответа. В качестве аргумента для создания объекта такой базы данных достаточно предоставить файл этой базы данных. Далее база автоматически создаст или загрузит таблицу необходимого типа, как указано в конструкторе объекта базы данных. Доступ к объектам в базе данных осуществляется с помощью методов, предоставляемых классами библиотеки SQLite из методов класса RemotesDatabase согласно руководству на сайте Xamarin. [13]

Работа с JSON.NET проводится согласно руководству. [14]

Запрос получения списка категорий считывает все категории в Хэш-Сет (так категории не повторяются) и затем, преобразовав его в список возвращает вызывающему коду.

Запрос получения пульта с конкретной категорией добавляет в выходной список только пульты с подходящей категорией.

Остальные запросы являются оболочкой над методами SQLite для приведения типов, проверок и т.п.

4. Выбор пульта для работы.

Пульт можно выбрать из уже имеющихся, для этого можно вывести либо все пульты на экран, любо только те, что соответствуют определенной категории. Когда появляется страница выбора пульта, в зависимости от свойства BindingContext страницы (если свойство пустое – выбираются все пульты, если является строкой, то выбираются пульты только с категорией, соответствующей строке) делается запрос к базе данных и из нее загружаются необходимые пульты и выводятся пользователю в виде списка, из которого пользователь может выбрать необходимый пульт. Далее пульты выводятся на страницу списком по их имени и, когда пользователь выбирает нужный, управление передается на страницу пульта, а в BindingContext этой страницы ставится соответствующий объект виртуального пульта. В случае нажатия кнопки новый пульт в базе данных сохраняется пустой новый пульт, и управление передается на страницу с этим новым объектом.

5. Генерация визуального отображения пульта.

Страница с пультом состоит для удобства из трех вкладок, которые делятся по типу клавиш: основные, дополнительные и цифровые. При генерации таблицы кнопок, каждой с помощью BindingContext ставится в сопоставление кнопка из объекта виртуального пульта, связанного со страницей пульта. При этом на кнопку устанавливается надпись и прочие свойства из связанной виртуальной кнопки. Все надписи на кнопках являются символами Юникода для упрощения работы приложения, все характерные пиктограммы берутся из расширенной таблицы Юникода. При нажатии соответствующей кнопки, запускается алгоритм передачи инфракрасного кода. В заголовок страницы ставится название пульта.

В режиме редактирования пульта категории располагаются друг над другом, а под ними располагаются поля ввода пользовательских надписей для некоторых кнопок. При изменении этих полей, надписи автоматически сохраняются в объекте пульта и меняются в визуальном интерфейсе, в базе данных – при нажатии кнопки сохранения. В режиме редактирования можно так же сменить имя пульта и его категорию, изменения сохраняются аналогично надписям. При нажатии на кнопку пульта, управление передается странице, отвечающей за прием кода для записи, при этом на экран выводятся соответствующие инструкции.

6. Отправка инфракрасного сигнала.

Для отправки инфракрасного сигнала используется App.BlueCon. В качестве обработчика нажатия для каждой кнопки виртуального пульта ставится единственный метод (во время генерации страницы отображения виртуального пульта), принимающий на вход объект, вызвавший событие и аргументы события. Таким образом из BindingContext вызвавшего объекта можно восстановить кнопку и ее инфракрасный код. Метод восстанавливает объект инфракрасного кода, проверяет, установлено ли соединение с управляющим устройством должным образом(открыт ли сокет) и, если соединение установлено, передает его для отправки методу SendIR класса BluetoothConnection. Метод передает управляющему устройству команду отправки инфракрасного устройства согласно протоколу, описанному в приложении 3. Для этого метод прибавляет к строке полученной из вызова IRSignal.ToString командный символ S, преобразовывает данную строку в последовательность байтов и записывает эту последовательность в выходной поток сокета. В случае ошибки последней записи пользователю выводится соответствующее сообщение.

7. Настройки пульта

При настройке пульта все изменения записываются в объект, связанный со страницей настройки и полученный при создании этой страницы на странице работы с пультом. Процесс настройки более детально описан в п 5 этого подраздела (3.6.1) и п 8 этого подраздела (3.6.1)

8. Прием инфракрасного сигнала

При приеме инфракрасного сигнала используется асинхронный метод класса BlueConnection ReciveIR. Он запускается вместе с токеном отмены. В случае, если пользователь не нажав кнопки на реальном пульте, покинул страницу получения сигнала, токен срабатывает и на управляющее устройство отправляется сигнал отмены, а от управляющего устройства приходит ошибка чтения. Связано это с тем, что метод чтения потока в языке C# невозможно отменить, пока не придет хоть что-то. Таким образом, сообщение об ошибке позволяет завершить метод чтения. В случае если пользователь слишком долго не нажимал реальной кнопки, так же приходит сигнал ошибки.

Сначала метод отправляет устройству сообщение о начале приема S, предусмотренное протоколом, описанном в Приложении 3. Далее происходит чтение входящего потока. Чтение происходит посимвольно ввиду низкой скорости передачи данных устройством Arduino. Чтение входного потока происходит посимвольно, пока не будет считано 23 символа – именно такую длину имеют входные сообщения, предусмотренные протоколом, описанным в Приложении 3. Символы читаются из входящего потока сокета. Далее происходит парсинг входного сигнала, и если это не ошибка, происходит создание нового объекта IRSignal и он возвращается вызывающему коду, если пришел сигнал ошибки, возвращается объект IRSignal с пометкой об ошибке. Если метод чтения класса BlueConnection завершился, страница приема сигнала закрывается, поученный объект в случае успешного приема записывается в соответствующую кнопку виртуального пульта, а пользователю выводится сообщение об успешном или ошибочном завершении операции записи нового сигнала для кнопки. Метод чтения сигнала автоматически запускается при открытии страницы чтения.

9. Удаление пульта

Для удаления пульта на странице работы с пультом есть кнопка удаления пульта. При ее нажатии совершается соответствующий запрос к базе данных. Страница с пультом закрывается, а на экран выводится страница выбора пультов.

10. Работа визуальной части приложения.

Приложение представляет собой активную страницу (Подключение, Быстрая справка, Выбор пульта, Работа с пультом, Редактирование пульта, Прием сигнала) и боковое выдвижное меню, которое содержит кнопки для перехода к странице подключения, быстрой справки, выбора пульта из всех и из отдельной категории. Такая компоновка называется MasterDetailPage. Таким образом, события на выдвижной панели определяют какой будет загруженная основная страница.

Все компоненты написаны в соответствии с руководством по программированию на языке C# [10][11] и руководству к написанию программ с использованием Xamarin.Forms [12].

### 3.6.1 Способы и алгоритмы решения применяемые для модуля, выполняемого на микроконтроллере

Часть программы, выполняемая на микроконтроллере является бесконечным циклом и прекращается только при отключении питания устройства. Для этого архитектурой Arduino предусмотрено две функции setup и loop. setup выполняется при старте микроконтроллера, loop запускается каждый раз по завершении ее предыдущего вызова. В функции setup запускается работа инфракрасного приемника и серийного порта, к которому подключен Bluetooth модуль.

Для работы с инфракрасным интерфейсом используются два класса из IRremote: IRrecv и IRsend При каждом запуске loop проверяется пришло ли что-то в серийный порт. Если пришла команда отправки, с помощью встроенной функции Serial.parseInt команда разбирается на данные (адрес, код, длина и тип) и передает соответствующие аргументы функциям отправки инфракрасного сигнала из IRsend. В программе существует отдельный класс триггера приема. Если пришло сообщение о приеме триггер включается, если триггер включен и код принят или триггер включен и код не принят в течении 60 секунд, триггер выключается, при этом в последнем случае передается сообщение об ошибке. В случае приема команды отмены триггер так же выключается и отправляется сообщение об ошибке. Если триггер включен, но сообщение о начале прима пришло еще раз, триггер выключается и передается сообщение об ошибке. Далее в loop проверяется, включен ли триггер, и, если включен, не принят ли код. Если код принят, он разбирается из объекта decodeResults (из IRrecv) и записывается согласно протоколу (Приложение 3) в серийный порт. В конце выполнения loop триггер проверяет, не прошло ли 60 секунд с момента включения триггера.

Разработка велась с учетом руководства к библиотеке IRremote [17]и руководства для разработки под Arduino [15][16].

Микроконтроллер взаимодействует со смартфоном согласно протоколу в приложении 3.

# 4 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

## 4.1 Предположительная потребность

Программа предполагается к использованию людьми, с большим количеством бытовой техники, управляемой с помощью инфракрасного интерфейса. Программа позволяет избавиться от большого количества пультов и использовать один смартфон для управления. Программа так же может быть полезна на мероприятиях, использующих большое количество техники, управляемых посредством инфракрасных сигналов.

## 4.2 Преимущества разработки в сравнении с аналогами (отечественными и зарубежными)

Быстрый поиск в сети Интернет не позволил выявить конкурентоспособных аналогов данной программы. Ближайшими аналогами являются низкокачественные любительские программы, предусмотренные для использования с управляющими устройствами, однако их функционал и надежность вызывают сомнения и не реализуют всех функций, выполняемых данной программой. В качестве аналогов можно так же рассматривать корпоративные решения, однако они предусмотрены для использования со встроенными в устройство ИК-портами (Quick Remote от LG).

# 5 ИСТОЧНИКИ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом – М.: ИПК Издательство стандартов, 2001.
10. Подбельский В.В. , Язык C#. Базовый курс: учеб. пособие; Финансы и статистика, 2013.- 408с.
11. Руководство по языку C# [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/articles/csharp/index (даты обращения: 09.01.2017-30.04.2017)
12. Руководство по использованию фреймворка Xamarin и Xamarin Forms, стандартным API [Электронный ресурс]. URL: https://developer.xamarin.com/ (даты обращения: 09.01.2017-30.04.2017)
13. Руководство по использованию SQLite представленное Xamarin [Электронный ресурс]. URL: https://developer.xamarin.com/guides/xamarin-forms/application-fundamentals/databases/ (даты обращения: 09.01.2017-30.04.2017)
14. Руководство по использованию JSON.NET [Электронный ресурс]. URL: http://www.newtonsoft.com/json/help/html/Introduction.htm (даты обращения: 09.01.2017-30.04.2017)
15. Руководство по программированию Arduino [Электронный ресурс]. URL: https://www.arduino.cc/en/Reference/HomePage (даты обращения: 09.01.2017-30.04.2017)
16. Спецификации Arduino [Электронный ресурс]. URL: https://www.arduino.cc/en/Guide/HomePage (даты обращения: 09.01.2017-30.04.2017)
17. Библиотека IRremote: Исходные коды, руководства, примеры [Электронный ресурс]. https://github.com/z3t0/Arduino-IRremote (даты обращения: 09.01.2017-30.04.2017)

# ПРИЛОЖЕНИЕ 1 ОИСАНИЕ И НАЗНАЧЕНИЕ КЛАССОВ

## КЛАССЫ ДЛЯ ЧАСТИ НА СМАРТФОНЕ

Так как код делится на кросплатформенный и нативный, решение содержит два проекта: IRemote и IRemote.Droid. В первом – кросплатформенный код, во втором – специфичный для платформы Android (классы, обеспечивающие непосредственное взаимодействие с Bluetooth, файловой системой, системными уведомлениями). Классы IRemote.Droid реализуют соответствующие интерфейсы IRemote.

### КЛАССЫ IRemote

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| RemotesDatabase | Полное взаимодействие с базой данных пультов, сохранение и чтение из нее. |
| DefaultRemote | Содержит статические объекты для генерации новых пультов. (Надписи на клавишах и т.д.) |
| Remote | Класс представления виртуального пульта с кодами и остальными параметрами |
| RemoteButton | Класс для описания кнопки виртуального пульта |
| RemoteToSave | Специальный класс для сохранения сериализованного пульта в базе данных |
| IRSignal | Класс хранения инфракрасного сигнала с адресом, кодом, длинной и типом |
| App.xaml | Главный класс приложения, запускает приложение, хранит статические объекты(база данных и т.д.) |
| ConnectionPage.xaml | Класс страницы подключения |
| EditPage.xaml | Класс страницы настроек пульта |
| HelpPage.xaml | Класс страницы краткой справки по приложению |
| HomePage.xaml | Класс страницы, отображаемой при запуске приложения. |
| MainPage | Класс главной MasterDetailPage страницы-оболочки |
| CategorySelectedEventArgs : EventArgs | Аргументы события выбора категории |
| MasterPage.xaml | Класс выдвижной боковой страницы с навигацией по приложению |
| RecivePage | Класс страницы приема инфракрасного сигнала |
| RemotePage | Класс-основа для страницы с вкладками для пульта. |
| SelectionPage | Класс страницы выбора пульта для работы |
| TabbedRemote | Класс вкладок страниц пульта |

### ИНТЕРФЕЙСЫ IRemote

|  |  |
| --- | --- |
| **Интерфейс** | **Назначание** |
| IBlueConnection | Для реализации Bluetooth подключения с отправкой и приемом данных от управляющего устройства. |
| IDbFileProvider | Для реализации предоставления файла базы данных |
| IMakeToast | Для реализации показа «тостов» |

### КЛАССЫ IRemote.Droid

|  |  |
| --- | --- |
| **Класс** | **Назаначение** |
| BlueConnection | Реализация Bluetooth подключения с отправкой и приемом данных от управляющего устройства. |
| DbFileProvider | Реализация предоставления файла базы данных |
| MakeToast | Реализации показа «тостов» |
| MainActivity | Класс запуска приложения для Android (главная активность) |

## КЛАССЫ ДЛЯ ЧАСТИ НА УПРАВЛЯЮЩЕМ УСТРОЙСТВЕ

|  |  |
| --- | --- |
| **Класс** | **Назаначение** |
| CW.ino | Главная часть программы, setup() и loop() |
| Trigger | Триггер включения/выключения приема |

# ПРИЛОЖЕНИЕ 2 НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ, СВОЙСТВ, СОБЫТИЙ КЛАССОВ

## КЛАССЫ IRemote

### Класс RemotesDatabase:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| database | поле | SQLiteAsyncConnection | - | private readonly | Объект базы данных, с которой работаем |
| RemotesDatabase | конструктор | - | string path | public | Конструктор класса с параметром пути к файлу азы данных |
| GetRemotesAsync | метод | Task<List<Remote>> | нет | async public | Асинхронный запрос, возвращающий список всех объектов пультов |
| GetRemotesWithCategoryAsync | метод | Task<List<Remote>> | string category | async public | Асинхронный запрос, возвращающий список всех объектов выбранной категории |
| GetCategoriesAsync | метод | Task<List<string>> | нет | async public | Асинхронный запрос, возвращающий список всех категорий |
| GetRemoteByIDAsync | метод | Task<Remote> | Int id | async public | Асинхронный запрос, возвращающий пульт по его ID. |
| SaveRemoteAsync | метод | Task<int> | Remote rem | async public | Асинхронный запрос, сохраняющий пульт и возвращающий его ID. |
| RemoveRemoteAsync | метод | Task<int> | Remote rem | async public | Асинхронный запрос, удаляющий пульт и возвращающий его ID. |

### Класс DefaultRemote:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| DefaultButtonNames | свойство | string[] | - | public static | Возвращает массив стандартных надписей на кнопках |

### Класс Remote:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| Remote | конструктор | - | - | public | Конструктор, возвращающий пустой (не забитый начальными данными) пульт |
| Remote | конструктор | - | bool IsNew | public | Конструктор при истинном параметре возвращает заполненный стандартными значениями |
| CustomKeys | свойство | int[] | - | public static | Массив идентификаторов кнопок с возможностью изменить надпись |
| Name | свойство | string | - | public | Название пульта |
| Category | свойство | string | - | public | Категория пульта |
| ID | свойство | int | - | public | Идентификатор пульта |
| Buttons | поле | List<RemoteButtons> | - | public | Список с кнопками пульта |

### Класс RemoteButton:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| RemoteButton | конструктор | - | - | public | Конструктор, возвращающий кнопку пульта стандартную |
| RemoteButton | конструктор | - | int id | public | Конструктор, возвращающий кнопку пульта с указанным ID |
| RemoteButton | конструктор | - | int id, string text | public | Конструктор, возвращающий кнопку пульта с указанным ID и надписью |
| RemoteButton | конструктор | - | int id, string text, IRSignal signal | public | Конструктор, возвращающий кнопку пульта с указанным ID, надписью и инфракрасным кодом |
| Text | свойство | string | - | public | Надпись на кнопке |
| ID | свойство | int | - | public | Идентификатор кнопки |
| Signal | свойство | IRSignal | - | public | Инфракрасный сигнал, назначенный кнопки |

### Класс RemoteToSave:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| ID | свойство | int | - | public | Идентификатор пульта в базе данных |
| Category | свойство | string | - | public | Категория пульта |
| JSONRemote | свойство | string | - | public | Сериализованный в JSON строку объект Remote |

### Класс IRSignal:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| IRSignal | конструктор | - | - | public | Создает новый инфракрасный сигнал |
| IRSignal | конструктор | - | uint code, int type | public | Создает новый инфракрасный сигнал с указанным кодом, типом |
| IRSignal | конструктор | - | uint code, int len, int type | public | Создает новый инфракрасный сигнал с указанным кодом, типом, длиной |
| IRSignal | конструктор | - | uint code, int len, int type, int adress | public | Создает новый инфракрасный сигнал с указанным кодом, типом, длиной и адресом |
| Code | свойство | uint | - | public | Код для передачи |
| Length | свойство | int | - | public | Длина кода |
| Adress | свойство | int | - | public | Адрес для передачи |
| ArduinoIRemoteType | свойство | int | - | public | Тип по версии библиотеки IRremote для Arduino |
| ToString | метод | string | - | public | Строка в формате для передачи управляющему устройству |
| Ok | свойство | bool | - | public | Не ошибочный ли сигнал |

### Класс App.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| App | конструктор | - | - | public | Выполняется при запуске приложения, задает основные параметры, стартовую страницу |
| \_con | поле | IBlueConnection | - | private static | Хранит единственный экземпляр Bluetooth подключения |
| \_database | поле | RemotesDatabase | - | private static | Хранит единственный экземпляр базы данных |
| \_toastMaker | поле | IMakeToast | - | private static | Хранит единственный экземпляр создателя “Тостов” |
| BlueCon | свойство | IBlueConnection | - | public static {get} | Если \_con пустой, создает объект IBlueConnection и отдает \_con, возвращает \_con |
| Database | свойство | RemotesDatabse | - | public static {get} | Если \_database пустой, создает объект RemotesDatabase и отдает \_database, возвращает \_database |
| ToastMaker | свойство | IToastMaker | - | public static {get} | Если \_toastMaker пустой, создает объект IToastMaker и отдает \_toastMaker, возвращает \_toastMaker. |

### Класс ConnectionPage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| ConnectionPage | конструктор | - | - | public | Создает новый экземпляр страницы подключения, проверяет наличие Bluetooth модуля |
| Devices | поле | ListView | - | private | Визуальный список сопряженных устройств |
| status | поле | label | - | private | Выводит статус подключения |
| selectedNumber | поле | int | - | private | Хранит номер выбранного в списке устройства |
| OnAppearing | метод | void | - | protected override | Выполняется при каждом открытии страницы, проверяет включён ли адаптер, выводит спсиок устройств |
| OnConnectClicked | метод | void | object sender, System.EventArgs e | protected | Выполняет попытку подключения к выбранному устройству при нажатии кнопки подключения |
| OnItemSelected | метод | void | object sender, SelectedItemChangedEventArgs e | protected | Срабатывает при выборе устройства, сообщает о выбранном устройстве BlueCon |
| OnDisconnectClicked | метод | void | object sender, EventArgs e | protected | Отключает устройство при нажатии кнопки отключения |

### Класс EditPage.xaml:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| EditPage | конструктор | - | - | public | Создает новый экземпляр страницы редактирвания |
| RemoteNameEntry | поле | Entry | - | private | Ввод имени пульта |
| CategoryEntry | поле | Entry | - | private | Ввод категории пульта |
| CategoryPicker | поле | Entry | - | private | Выбор категории из уже известных |
| MainGrid | поле | Grid | - | private | Сетка для заполнения основными кнопками |
| AdditionalGrid | поле | Grid | - | private | Сетка для заполнения дополнительными кнопками |
| NumbersGrid | поле | Grid | - | private | Сетка для заполнения цифровыми кнопками |
| CustomButtonsEdit | поле | StackLayout | - | private | Стек для элементов редактирования пользовательских кнопок |
| bindedRemote | поле | Remote | - | private | Связанный со страницей пульт |
| allButtons | поле | List<Button> | - | private | Список всех визуальных кнопок |
| OnAppearing | метод | void | - | async protected override | Выполняется при каждом открытии страницы, запускает заполнение страницы кнопками и другими элементами |
| OnRemoteNameChanged | метод | void | object sender, EventArgs e | protected | Выполняется при изменении имени пульта, изменяет его в связанном объекте и в заголовке страницы |
| OnDisappearing | метод | void | object sender, EventArgs e | async protected override | Выполняется при закрытии страницы, выводит уведомление о несохраненных изменениях |
| OnSaveClicked | метод | void | object sender, EventArgs e | protected async | Выполняется при нажатии кнопки сохранения, сохраняет связанный пульт в базе данных |
| CategoryPicker\_SelectedIndexChanged | метод | void | object sender, EventArgs e | protected | Выполняется при выборе категории из уже известных, сохраняет выбранную категорию в объект и выписывает в CategoryEntry |
| CategoryEntery\_TextChanged | метод | void | object sender, TextChangedEventArgs e | protected | Выполняется при изменении категории пульта, изменяет его в связанном объекте |
| GenerateGrid | метод | void | Grid grid, int istart, int ifinish | protected | Заполняет указанную таблицу кнопками c идентификаторами от istart до ifinish, используется в генерации страницы |
| GenerateCustomKeysEditor | метод | void | - | protected | Генерирует поля ввода пользовательских надписей на кнопках |
| SetAllButtonsList | метод | void | - | protected | Создает в allButtons список объектов всех сгенерированных кнопок |
| OnCustomKeyTextChanged | метод | void | object sender, EventArgs e | protected | Вносит изменения в надпись на кнопке и в связанный со страницей объект пульта |
| NextButton\_Clicked | метод | void | object sender, EventArgs e | protected | Запускает процесс записи инфракрасного кода и при успехе сохраняет код в связанный объект |

### Класс HelpPage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| HelpPage | конструктор | - | - | public | Создает новый экземпляр страницы краткой справки |
| about | поле | Label | - | private | Вывод краткой справки на экран |

### Класс HomePage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| HelpPage | конструктор | - | - | public | Создает новый экземпляр домашней страницы |
| aboutLabel | поле | Label | - | private | Вывод краткой справки на экран |

### Класс MainPage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| MainPage | конструктор | - | - | public | Создает новый экземпляр главной страницы-оболочки, устанавливает связь событий навигации на выдвижной странице с изменением основной страницы |
| master | поле | MasterPage | - | private | Хранит связанную боковую выдвижную страницу |
| MainPage\_IsPresentedChanged | метод | void | object sender, EventArgs e | private | Выполняет обновление списка категорий на выдвижной панели при ее появлении |
| OnConnectButtonClicked | метод | void | object sender, EventArgs e | public | Навигация на страницу подключения |
| OnShowAllButtonClicked | метод | void | object sender, EventArgs e | public | Навигация на страницу выбора пульта из всех |
| OnShowAllButtonClicked | метод | void | object sender, CategorySelectedEventArgs e | private | Навигация на страницу выбора пульта из категории |
| OnHelpButtonClicked | метод | void | object sender, EventArgs e | private | Навигация на страницу краткой справки. |

### Класс CategorySelectedEventArgs : EventArgs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| CategorySelectedEventArgs | конструктор | - | string category | public | Создает новый экземпляр событий с аргументом категории |
| Category | свойство | string | - | public {get} | Хранение аргумента категории |

### Класс MasterPage.xaml

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | | **Входные параметры** | **Модификаторы** | **Назначение** |
| MasterPage | конструктор | - | | - | public | Создает новый экземпляр выдвижной страницы |
| LogoView | поле | ContentView | | - | private | Место для логотипа приложения |
| LogoLabel | поле | Label | | - | private | Надпись с названием |
| ConnectButton | поле | Button | | - | private | Кнопка перехода к странице подключения |
| HelpButton | поле | Button | | - | private | Кнопка перехода к странице краткой справки |
| ShowAllButton | поле | Button | | - | private | Кнопка показа всех пультов |
| CategoryList | поле | ListView | | - | private | Список категорий пультов |
| OnAppearing | метод | void | - | | async protected override | Выполняется при каждом открытии страницы, запускает заполнение страницы категориями |
| CategorySelected | событие | Делегат CategorySelectedEventHandler  void (object sender, CategorySelectedEventArgs e)  Встречается только для этого события, отдельное описание излишне. | | | public | Служит для сообщения главной странице о выбранной категории, что бы загрузилась страница выбора пульта нужной категории |
| HelpButtonClicked | событие | Стандартный делегат EventHandler | | | public | Служит для сообщения главной странице о нажатии кнопки краткой справки, что бы загрузилась страница краткой справки |
| ConnectButtonClicked | событие | Стандартный делегат EventHandler | | | public | Служит для сообщения главной странице о нажатии кнопки подключения, что бы загрузилась страница подключения |
| ShowAllButtonClicked | событие | Стандартный делегат EventHandler | | | public | Служит для сообщения главной странице о нажатии кнопки отображения всех пультов, что бы загрузилась страница выбора |
| OnNavigation | метод | void | object sender, EventArgs e | | public | Подключается ко всем элементам управления и в зависимости от срабатывания одного из них вызывает соответствующее событие |
| UpdateCategories | метод | void | - | | async public | Обновляет список категорий |

### Класс RecivePage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| RecivePage | конструктор | - |  | public | Создает новый экземпляр страницы приема |
| OnAppearing | метод | void | - | protected override async | Вызов приема инфракрасного сигнала, ожидание его и запись при успехе |
| OnBackButtonPressed | метод | bool | - | protected override | Вызов отмены приема и завершение приема |
| Handle\_Clicked | метод | void | object sender, EventArgs e | private | Вызов отмены приема и завершение приема |

### Класс RemotePage.xaml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| RemotePage | конструктор | - | - | public | Создает новый экземпляр страницы работы с пультом |
| OnAppearing | метод | void | - | protected override async | Генерация внешнего вида |
| ButtonsGrid | поле | Grid | - | private | Таблица для кнопок пульта |
| bindedRemote | поле | Remote | - | protected | Связанный объект пульта |
| IStart | свойство | int | - | public | Начальный идентификатор генерации кнопок |
| IFinish | свойство | int | - | public | Конечный идентификатор генерации кнопок |
| OnDeleteClicked | метод | void | object sender, EventArgs e | protected async | Удаление объекта пульта при нажатии кнопки удаления |
| OnEditClicked | метод | void | object sender, EventArgs e | protected async | Переход к редактированию пульта при нажатии кнопки редактирования |
| NextButtonClicked | метод | void | object sender, EventArgs e | private | Отправка инфракрасного сигнала, если возможно |

### Класс SelectionPage.xaml:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| SelectionPage | конструктор | - | - | public | Создает новый экземпляр страницы выбора пульта |
| OnAppearing | метод | void | - | protected override async | Генерация внешнего вида |
| AddNewClicked | метод | void | object sender, EventArgs e | protected async | Создает новый пульт в базе данных, выводит его страницу на экран |
| OnRemoteSelected | метод | void | object sender, SelectedItemChangedEventArgs e | protected async | При выборе пульта из списка выводит его страницу на экран |
| RemotesList | поле | ListView | - | protected | Список пультов для выбора |

### Класс TabbedRemote.xaml:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| TabbedRemote | конструктор | - | - | public | Создает новый экземпляр страницы c тремя вкладками для пульта |
| OnAppearing | метод | void | - | protected override | Генерация трех вкладок с объектами RemotePage |

Назначение остальных членов визуальных классов тривиально и легко выявляется из исходного кода, названия и информации выше или они являются унаследованными от базовых визуальных элементов и не используются программистом.

## ИНТЕРФЕЙСЫ IRemote

Для выявления назначений членов интерфейсов смотреть классы IRemote.Droid их реализующие, сигнатуры и списки членов интерфейсов далее:

### Интерфейс IBlueConnection:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** |
| BoundedDevices | свойство | List<string> | - |
| IsBluetoothOn | свойство | bool | - |
| DeviceToWorkSetByNumber | свойство | int | - |
| AnyBluetooth | свойство | bool | - |
| ConnectToSelectedDevice | метод | Task<bool> | - |
| IsConnected | свойство | bool | - |
| Disconnect | метод | void | - |
| ConnectedDeviceName | свойство | string | - |
| SendIR | метод | void | - |
| ReciveIR | метод | Task<IRSignal> | - |
| CalancellReciving | метод | void | - |

### Интерфейс IDbFileProvider:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** |
| GetLocalFilePath | метод | string | string filename |

### Интерфейс IToastMaker:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** |
| ShowMessage | метод | void | string Message, bool IsLong |

## КЛАССЫ IRemote.Droid

### Класс BlueConnection:IBlueConnection:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| BluetoothConection | конструктор | - | - | public | Создает новый экземпляр интерфейса, используя стандартный адаптер |
| BoundedDevices | свойство | List<string> | - | public | Возвращает список сопряженных устройств |
| IsBluetoothOn | свойство | bool | - | public | Включен ли Bluetooth |
| DeviceToWorkSetByNumber | свойство | int | - | public | Устанавливает устройство для работы по номеру из списка |
| AnyBluetooth | свойство | bool | - | public | Есть ли адаптер в смартфоне |
| ConnectToSelectedDevice | метод | Task<bool> | - | public | Подключиться к выбранному устройству |
| IsConnected | свойство | bool | - | public | Подключен ли адаптер |
| Disconnect | метод | void | - | public | Закрыть сокет |
| ConnectedDeviceName | свойство | string | - | public | Имя подключенного устройства |
| SendIR | метод | void | - | public | Отправить сигнал |
| ReciveIR | метод | Task<IRSignal> | - | public | Принять сигнал |
| CalancellReciving | метод | void | - | public | Отменить прием сигнала |
| adapter | поле | BluetoothAdapter | - | private | Адаптер для работы |
| toster | поле | MakeToast | - | private | Объект для показа сообщений пользователю |
| device | поле | BluetoothDevice | - | private | Устройство для работы |
| bondedDevices | поле | List<BluetoothDevice> | - | private | Список сопряженных устройств |
| socket | поле | BluetoothSocket | - | private | Сокет, созданный адаптером |

### Класс DbFileProvider:IDbFileProvider:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| GetLocalFilePath | метод | string | string filename | public | Возвращает путь до указанного файла |

### Класс MakeToast:IMakeToast

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| ShowMessage | метод | void | string Message, bool IsLong | public | Показывает сообщение пользователю |

### Класс MainActivity:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| OnCreate | метод | void | Bundle bundle | protected override | Запуск Android приложения |
| \_mainContext | поле | Context | - | private | Хранит контекст для некоторых нативных методов |
| MainContext | поле | Context | - | public | доступ к \_mainContext |

## КЛАССЫ НА УПРАВЛЯЮЩЕМ УСТРОЙСТВЕ

### CW.ino

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| trigger | поле | TriggerClass | - | - | Триггер чтения |
| RecivePin | поле | int | - | - | Пин для подключения приемника |
| recivier | поле | IRrecv | - | - | Объект приемника инфракрасного сигнала |
| sender | поле | IRsend | - | - | Объект передатчика инфракрасного сигнала |
| result | поле | decode\_results | - | - | Объект декодирования инфракрасного сигнала |
| setup | функция | void | - | - | Установка приемника, передатчика, серийного порта |
| loop | функция | void | - | - | Основной цикл программы, выполняются все действия, описанные в алгоритме |

### Класс Trigger.cpp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Тип члена** | **Тип значения** | **Входные параметры** | **Модификаторы** | **Назначение** |
| \_setAt | поле | long | - | protected | Время установки триггера включенным |
| \_enabled | поле | bool | - | protected | Включен ли триггер |
| init | метод | void | - | public | Как конструктор |
| check | метод | void | - | public | Проверить не прошло ли 60 секунд |
| setEnabled | метод | void | - | public | Включить триггер |
| setDisabled | метод | void | - | public | Отключить триггер |
| isEnabled | метод | bool | - | public | Доступ к \_enabled |

# ПРИЛОЖЕНИЕ 3 ПРОТОКОЛ ПЕРЕДАЧИ ДАННЫХ

1. Кодировка канала ASCII
2. 8 бит на символ
3. Формат инфракрасного кода TTT@XXXXXXXXXX@LL@AAAAA (далее CODE), где TTT – тип кода, XXXXXXXXXX – код, LL – длина в битах, AAAAA – адрес.
4. Команда передачи кода – “SCODE”
5. Команда “R” – прием сигнала.
6. Длина ответа – 23 бита.
7. Команда “C” – отмена приема, при этом в ответ приходит сообщение “EEE@EEEEEEEEEE@EE@EEEEE” об ошибке.
8. Формат ответа – “CODE” – код сигнала или “TTT@TTTTTTTTTT@TT@TTTTT” – если пользователь не нажал кнопку на реальном пульте для сохранения сигнала более чем за 60 секунд.
9. Другие команды запрещены

# ПИЛОЖЕНИЕ 4 ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ

1. Под пином понимается цифровой вывод микроконтроллера
2. Под кросплатформенным кодом понимается код, работающий независимо от платформы, на которой он запущен
3. Под нативным кодом понимают код, работающий на одной конкретной платформе и использующий особенности этой платформы.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номера листов (страниц) | | | | | Всего листов (страниц в документе) | | Номер документа | Входящий номер сопр. документа и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |