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

\_\_\_\_\_\_\_\_\_\_\_ Б.В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

МУЛЬТИЗАДАЧНЫЙ РОБОТ. ПРОГРАММНАЯ ЧАСТЬ

БГУИР ДП 1–40 02 01 01 040 ПЗ

Студент Г.А. Ковбаса

Руководитель Д.Ю. Перцев

Консультанты:

от кафедры ЭВМ Д.Ю. Перцев

по экономической части Т.Л. Слюсарь

Нормоконтролер А.С. Сидорович

Рецензент

МИНСК 2020

**РЕФЕРАТ**

Дипломный проект предоставлен следующим образом. Электронные носители: 1 диск. Чертежный материал: 7 листов формата А1. Пояснительная записка: 130 страниц, 22 рисунка, 72 таблицы, 35 литературных источников, 3 приложения.

Ключевые слова: Мультизадачный робот, обработка команд, распознавания лица, мониторинг субъектов, алгоритм предобработки, автоматизированное перемещение, взаимодействие модулей, распознавание речи, Bluetooth, мобильное приложение.

Предметной областью дипломного проекта является робототехника, объектом разработки является программное обеспечение для мультизадачного робота.

Целью данного дипломного проекта является разработка и реализация программного обеспечения для функционирования аппаратной платформы в составе комплексного проекта «Мультизадачный робот», реализующего концепт персонального робота, предоставляющего возможности автоматизированной системы распознавания и мониторинга пользователей.

Для разработки использовались языки программирования С, C++, Python 3.8, Java 9.0 и среды разработки Arduino IDE, Python 3 IDLE и Android Studio 3.6.1.

В рамках дипломного проекта было разработано полностью функциональное программное обеспечения для «Мультизадачного робота», включающее в себя: прошивку для микроконтроллера Arduino Uno, модуль интерактивного взаимодействия для Raspberry Pi и мобильное приложения. Поставленные задачи были выполнены в полной мере.

«Мультизадачный робот» имеет большой потенциал как для персонального использования, так и для применения в сфере образования. Программное обеспечение проекта предоставляет базу для обучения учащихся и студентов разработке встроенного и системного программного обеспечения для микроконтроллеров семейства Arduino и микрокомпьютеров фирмы Raspberry Pi Foundation. Персональная настройка робота обеспечивается мобильным приложением посредством Bluetooth.

Экономическая эффективность данной разработки обеспечивается широким функционалом и автономностью робота, позволяя заменить сразу несколько аналогичных устройств, таких как система мониторинга помещений, голосовой ассистент и интерактивная охранная система.

Робота можно использовать для удаленного мониторинга состояния помещения при длительном отъезде из дома, в качестве интерактивной охранной системы или как развлечение для детей или всей семьи. Централизованно управляющую несколькими роботами систему на основе разработанной модели можно использовать для перевозки грузов в условиях производства или для прокладки кабеля в шахте, своими размерами не позволяющей работать там человеку.

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

\_\_\_\_\_\_\_\_\_ Б.В. Никульшин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

ЗАДАНИЕ

по дипломному проекту студента

Ковбасы Галины Александровны

**1** Тема проекта: «Мультизадачный робот. Программная часть» – утверждена приказом по университету от 31 марта 2020 г. № 860-с.

**2** Срок сдачи студентом законченного проекта: 1 июня 2020 г.

**3** Исходные данные к проекту:

**3.1** Протоколы передачи данных: RS-232, RFCOMM.

**3.2** Целевая операционная система мобильного устройства: Android 8.0 +.

**3.3** Операционная система микрокомпьютера Raspberry Pi: Raspbian Buster.

**4** Содержание пояснительной записки (перечень подлежащих разработке

вопросов):

Введение 1. Обзор литературы. 2. Системное проектирование.   
3. Функциональное проектирование. 4. Разработка программных модулей.   
5. Программа и методика испытаний. 6. Руководство пользователя. 7. Технико-экономическое обоснование разработки и использования

программной части в составе коллективного проекта «Мультизадачный робот». Заключение. Список использованных источников. Приложения.

**5** Перечень графического материала (с точным указанием обязательных

чертежей):

**5.1** Вводный плакат. Плакат.

**5.2** Мультизадачный робот. Программная часть. Схема структурная.

**5.3** Алгоритм предобработки. Схема программы.

**5.4** Алгоритм добавления субъекта распознавания. Схема программы.

**5.5** Алгоритм обработки ситуаций столкновения. Схема программы.

**5.6** Модуль интерактивного взаимодействия. Диаграмма

последовательности.

**5.7** Прошивка Arduino Uno. Диаграмма последовательности.

**5.8** Мобильное приложение. Диаграмма последовательности.

**5.9** Прошивка Arduino Uno. Диаграмма классов.

**5.10** Мобильное приложение. Диаграмма классов.

**6** Содержание задания по экономической части: «Технико-экономическое

обоснование разработки и использования программной части в составе коллективного проекта «Мультизадачный робот».

ЗАДАНИЕ ВЫДАЛ                                                          Т.Л.Слюсарь

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов  дипломного проекта | Объем  этапа,  % | Срок выполнения этапа | Примечания |
| Подбор и изучение литературы | 10 | 23.03 – 30.03 |  |
| Структурное проектирование | 10 | 31.03 – 18.04 |  |
| Функциональное проектирование | 20 | 19.04 – 20.04 |  |
| Разработка программных модулей | 30 | 21.04 – 05.05 |  |
| Программа и методика испытаний | 10 | 06.05 – 18.05 |  |
| Расчет экономической эффективности | 10 | 11.04 – 18.04 |  |
| Оформление пояснительной записки | 10 | 18.05 – 01.06 |  |

Дата выдачи задания: 23 марта 2020 г.

Руководитель Д.Ю. Перцев

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 7

1 ОБЗОР ЛИТЕРАТУРЫ 8

1.1 Аналитический обзор рынка персональных роботов 8

1.2 Анализ предоставляемых средств разработки для аппаратной

составляющей проекта 11

1.2.1 Языки программирования и среда разработки для

микроконтроллера Arduino Uno 11

1.2.2 Языки программирования и среда разработки для платформы

Raspberry Pi 12

1.3 Сравнительный анализ методов распознавания лица 13

1.4 Обзор пакетов для распознавания речи 16

1.5 Результаты обзора литературы 17

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 18

2.1 Модуль интерактивного взаимодействия 19

2.1.1 Обоснование выбора библиотек и методов 19

2.1.2 Функциональные блоки модуля 19

2.2 Мобильное приложение 22

2.2.1 Обоснование выбора средств разработки и языка

программирования мобильного приложения 22

2.2.2 Функциональные блоки мобильного приложения 22

2.3 Прошивка Arduino Uno 23

2.4 Результаты системного проектирования 24

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 25

3.1 Взаимодействие модулей 25

3.2 Описание функциональности модуля интерактивного взаимодействия 28

3.3 Описание функциональности прошивки Arduino Uno 30

3.4 Описание функциональности мобильного приложения 40

3.5 Результаты функционального проектирования 62

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 63

4.1 Алгоритмы модуля интерактивного взаимодействия 63

4.1.1 Алгоритм предобработки изображений 63

4.1.2 Алгоритм добавления субъекта распознавания 68

4.1.3 Алгоритм поиска 70

4.1.4 Алгоритм обработки команд процессом BluetoothServer 71

4.2 Алгоритмы мобильного приложения 72

4.2.1 Алгоритм старта загрузки изображений 72

4.3 Алгоритмы прошивки Arduino Uno 73

4.3.1 Алгоритм обработки ситуаций столкновения 74

4.3.2 Алгоритм обработки команд класса SerialIntercommunicator 76

4.4 Результаты разработки программных модулей 78

5 ПРОГРАММА И МЕТОДЫ ИСПЫТАНИЙ 79

5.1 Исправная работа программной части мультизадачного робота 79

5.1.1 Модуль интерактивного взаимодействия 79

5.1.2 Мобильное приложение 82

5.2 Проблемы с Bluetooth соединением 85

5.3 Проблемы с соединением по RS-232 88

5.4 Выводы по результатам тестирования программного обеспечения 89

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 90

6.1 Развертывание программной части на аппаратной платформе 90

6.1.1 Требования к аппаратной составляющей платформы 90

6.1.2 Требования к программному обеспечению

аппаратной платформы 90

6.1.3 Установка модуля интерактивного взаимодействия 91

6.1.4 Установка прошивки Arduino Uno 92

6.2 Руководство пользователя для мобильного приложения 93

6.2.1 Минимальные системные требования 93

6.2.2 Рекомендуемые системные требования 93

6.2.3 Обзор интерфейса и функциональных

возможностей приложения 93

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И

ИСПОЛЬЗОВАНИЯ ПРОГРАММНОЙ ЧАСТИ В СОСТАВЕ

КОЛЛЕКТИВНОГО ПРОЕКТА «МУЛЬТИЗАДАЧНЫЙ РОБОТ» 98

7.1 Характеристика программной части мультизадачного робота 98

7.2 Расчет инвестиций в разработку программного средства 99

7.3 Расчет результата от разработки и использования

программного средства для организации-разработчика 102

7.4 Расчет показателей экономической эффективности разработки и

использования программного средства для организации

разработчика 103

ЗАКЛЮЧЕНИЕ 104

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 105

ПРИЛОЖЕНИЕ А 108

ПРИЛОЖЕНИЕ Б 129

ПРИЛОЖЕНИЕ В 130

**ВВЕДЕНИЕ**

Все больше различных компаний представляют на рынок свои варианты роботов-ассистентов для широкого круга пользователей. Такие устройства различаются в возможностях и себестоимости, начиная от более простых умных колонок, заканчивая персональными роботами для домашнего использования. Эти устройства служат различным целям: от игрового обучения до повышения производительности и автоматизации бытовых задач.

Целью данного дипломного проекта является разработка и реализация программного обеспечения для функционирования аппаратной платформы в составе комплексного проекта «Мультизадачный робот», реализующего концепт персонального робота, предоставляющего возможности автоматизированной системы распознавания и мониторинга пользователей.

В первую очередь данный программный модуль предназначен для развертывания на аппаратной платформе, реализуемой в качестве дипломного проекта «Мультизадачный робот. Аппаратная часть». Также программное обеспечение может быть использовано на иной аппаратной платформе, поддерживающей функционал целевой платформы.

В соответствии с поставленной целью были определены следующие задачи:

* выбор языков программирования и сред разработки в соответствии с текущим аппаратным обеспечением робота;
* выбор языка программирования и среды разработки для мобильного приложения;
* реализация детектирования и классифицирования изображений лиц на предмет совпадения с сохраненными пользователями;
* предобработка изображений лиц для повышения эффективности алгоритмов классифицирования;
* обучение классификатора для распознавания лиц на основе изображений видеопотока в реальном времени;
* передача команд на микроконтроллер для перемещения устройства в пространстве и поворота камеры для мониторинга пользователя;
* обход потенциальных препятствий при помощи показаний датчиков аппаратной платформы;
* считывание голосовых команд пользователя, обработка и воспроизведение звукового ответа;
* настройка распознавания лиц, речи и установка параметров функционирования по командам от клиентского мобильного приложения.

**1 ОБЗОР ЛИТЕРАТУРЫ**

## 1.1 Аналитический обзор рынка персональных роботов

Персональные роботы занимают все большую долю на рынке робототехники. Такие устройства применяются для контроля и мониторинга помещений, автоматизации и упрощения бытовых задач, в образовательных и развлекательных целях. Так, согласно данным IFR World Robotics (см. рисунок 1.1), в 2017 г. было продано на 12% больше роботов для образовательно-развлекательных целей и на 31% бытовых роботов, чем за тот же период 2016 года [1].

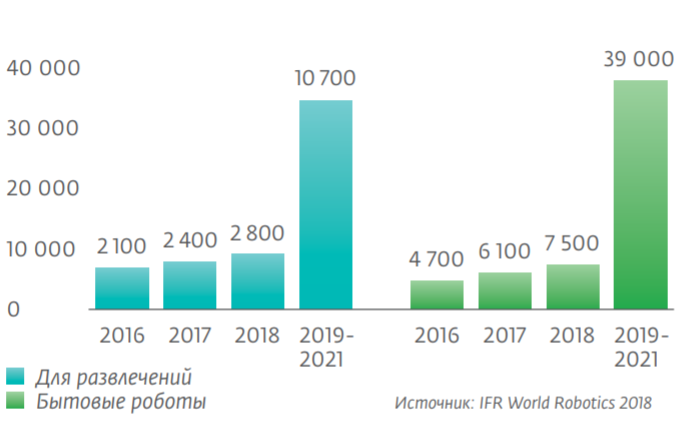


Рисунок 1.1 – Динамика продаж персональных роботов, тыс. шт. [1]

К наиболее перспективным направлениям робототехники относят машинное обучение, искусственный интеллект, человеко-машинное взаимодействие [1]. В приложении к индустрии персональных роботов это подтверждается стремительно развивающимся рынком голосовых ассистентов, чат-ботов и иных подобных разработок [2].

На основе проанализированных данных о перспективных направлениях в разработке персональных роботов было решено уделить внимание таким задачам, как интерактивное взаимодействие робота с пользователем, включающее в себя такие задачи, как распознавания лица, обработка голосовых команд, и независимое передвижение в помещении для возможности применения устройства в качестве системы мониторинга.

В соответствии с поставленными задачами были выбраны наиболее популярные устройства подобной направленности для анализа функционала потенциальных аналогов различных ценовых категорий.

Миниатюрные роботы с возможностью программирования, такие как предоставляет компания Anki, являются неплохим решением для образовательных и игровых целей. Они просты в освоении для начинающих и доступны по цене [3]. Наиболее известный продукт компании Anki – Cozmo – обладает следующими возможностями:

* распознавание лица пользователя и его эмоций;
* ориентирование в пространстве;
* манипулирование небольшими объектами;
* предоставлен SDK на языке Python для разработки.

Таким образом, основным преимуществом программного обеспечения Cozmo, не учитывая разработанных пользователями программных средств на основе SDK, является распознавание человеческих эмоций. Разрабатываемая программная часть мультизадачного робота в данный момент не поддерживает данную функцию, однако является одной из главных перспектив к дальнейшему совершенствованию. С другой стороны, к существенным недостатком робота Anki Cozmo относится отсутствие возможности мониторинга помещений, сужая возможности применения.

Как правило, роботы с системой распознавания лиц чаще всего применяются в составе систем умный дом. К примеру, робот RILEY от компании iPATROL представляет собой устройство для удаленного мониторинга помещений с возможностью распознавания лиц [4]. Механизм распознавания лиц не разглашается, поэтому отсутствует возможность сравнения с нашим алгоритмом. RILEY может работать как в автоматическом режиме, так и управляться со смартфона. К достоинствам робота относится автоматическая подзарядка – робот самостоятельно возвращается на док-станцию при разрядке батареи. К недостаткам iPATROL RILEY относятся:

* отсутствие функции голосового помощника, робот может только транслировать звук с микрофона мобильного устройства в ручном режиме работы;
* распознавание лиц работает только в ручном режиме работы.

Еще одним примером робота для мониторинга состояния помещения является мобильный робот xTurion [5]. Основное предназначение - мониторинг офисов, жилых помещений и промышленных складов, контроль окружающей среды (снятие показаний датчиков дыма, протечек воды, газоанализатора). Он может контролировать ложные вызовы и аппаратную избыточность стационарных систем безопасности. xTurion может самостоятельно спланировать траекторию перемещения, составить карту помещения, передавать видео с места событий, а также автоматически заряжать аккумуляторы. Для управления роботом существует мобильное приложение и веб-клиент. Несмотря на подобный функционал, по сравнению с «Мультизадачным роботом», xTurion, как и iPATROL RILEY, проигрывает из-за отсутствия поддержки голосового интерфейса.

Среди перечисленных аналогов выделяется французский робот Buddy (рисунок 1.2), основное применение которого, как заявляют сами разработчики, — эмоциональный компаньон [6].

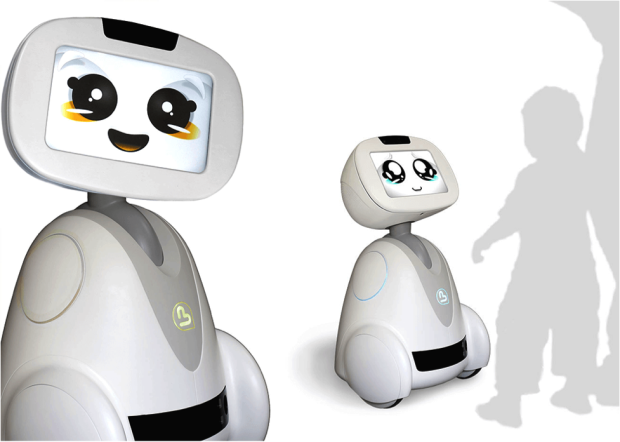


Рисунок 1.2 – Робот Buddy [7]

Данный робот поддерживает как все перечисленные задачи разрабатываемого проекта, так и способен распознавать эмоции пользователя и отображать на дисплее подходящий эмоциональный ответ, проводить видеоконференции, просматривать мультимедиа контент и выполнять иные функции. Широкая вариативность применения робота Buddy делает его главным аналогом для дальнейшего ориентирования в направлениях совершенствования программного обеспечения мультизадачного робота.

Таким образом, проведя тщательный обзор и анализ существующих аналогов проекта «Мультизадачный робот» была подтверждена актуальность разработки и пути ее дальнейшего развития. Главными преимуществами программной части робота являются автономность и разноплановые функциональные возможности, позволяющие заменить сразу несколько аналогичных устройств, такие как система мониторинга помещений и настольный голосовой ассистент.

* 1. **Анализ предоставляемых средств разработки для аппаратной составляющей проекта**

Средства разработки и языки программирования проекта подбирались для составляющих аппаратной части «Мультизадачного робота», а именно микроконтроллера Arduino Uno и микрокомпьютера Raspberry Pi 4 Model B с 4 Гб оперативной памяти.

* + 1. **Языки программирования и среда разработки для**

**микроконтроллера Arduino Uno**

Язык программирования микроконтроллеров семейства Arduino основан на функциях языков C и C++, скомпонован с библиотекой AVR Libc и позволяет использовать любые ее функции [8].

Для дальнейшей работы наилучшим вариантом является официальная среда разработки Arduino IDE версии 1.8.33 (рисунок 1.3), позволяющая не только компилировать исходный код программы, но и загружать готовую прошивку в подключенное к компьютеру устройство [9].



Рисунок 1.3 – Код программы в среде Arduino IDE

Среди дополнительных возможностей можно выделить средства для отладки прошивки, монитор COM порта и встроенные примеры кода для начинающих.

Существует также веб-версия Arduino IDE, располагающая большинством возможностей портативной, однако не поддерживающая работу с COM портом, необходимую в данном проекте.

* + 1. **Языки программирования и среда разработки для платформы**

**Raspberry Pi**

Так как Raspberry Pi является микрокомпьютером, выбор языка и среды разработки был продиктован рекомендованной операционной системой микрокомпьютера, Raspbian Buster, которая в стандартном варианте поставляется с предустановленным Python версий 2 и 3, официальным языком программирования Raspberry Pi, и IDLE 3, интегрированной средой разработки для Python [10]. Python также является и наиболее популярным языком для разработки на устройствах семейства Raspberry Pi, наряду со Scratch и JavaScript [11].

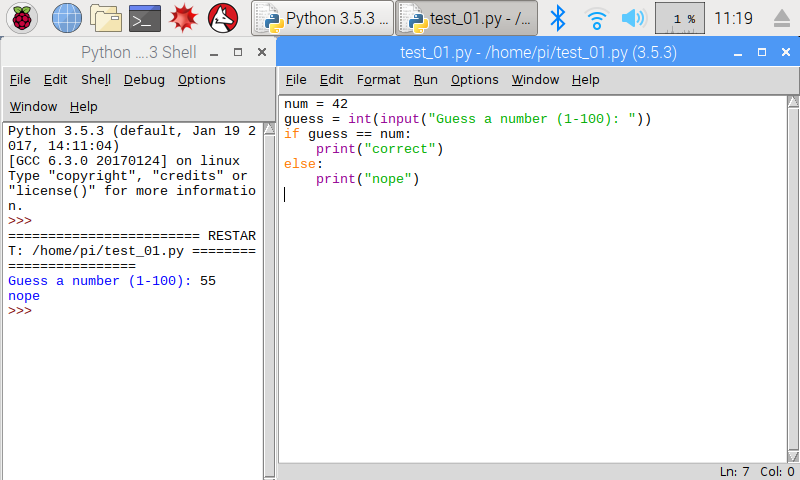


Рисунок 1.4 – Интегрированная среда разработки для Python в системе Raspbian [12]

Возможности операционной системы Raspbian Buster с поддержкой графического интерфейса LXDE позволяют писать, запускать и отлаживать скрипты на самом микрокомпьютере, что удобно при разработке программ для распознавания лица, так как возможно в реальном времени видеть результаты работы (рисунок 1.4).

По сравнению с перечисленными ранее языками Scratch и JavaScript, Python является более удобным решением для проектирования сложной системы, так как является языком общего назначения и поддерживает различные парадигмы программирования [13].

К тому же, задача разработки программного средства упрощается благодаря множеству доступных библиотек для языка Python 3 и удобного инсталлятора pip. Устанавливаемые пакеты можно также легко разделить для различных нужд и не перегружать глобальное пространство языка, устанавливая их в виртуальное окружение.

Из достоинств применения языка Python необходимо также отметить кроссплатформенность [13], позволяющую тестировать и разрабатывать код на устройствах других платформ в отсутствие доступа к микрокомпьютеру.

## 1.3 Сравнительный анализ методов распознавания лица

Алгоритмы компьютерного зрения и распознавания образов являются весьма нетривиальными, так как многие архитектуры современных компьютеров плохо подходят для решения таких задач. При этом для решения подобных задач при помощи компьютерного зрения используются достаточно сложные алгоритмы. Проблемы, которые встают перед разработчиками в данной области, возможно решать различными способами, в зависимости от вычислительных мощностей платформы и временных ресурсов.

Сам процесс распознавания лица можно разделить на следующие этапы:

1. Обнаружение лица и предобработка полученного изображения.

2) Обучение распознавателя на заранее созданном датасете.

3) Сравнение обнаруженного лица с уже известными (непосредственное распознавание).

Для обнаружения лиц на изображении используется метод каскадов Хаара, также известный как алгоритм Виолы-Джонса. Это подход, основанный на механизме обучения, где каскадная функция алгоритма обучается из множества положительных и отрицательных изображений. Обученный классификатор используется для обнаружения объектов на других изображениях. Для того, чтобы определить, является ли объект человеческим лицом, необходимо выделить примитивы, такие как губы, нос, глаза, лоб (см. рисунок 1.5).

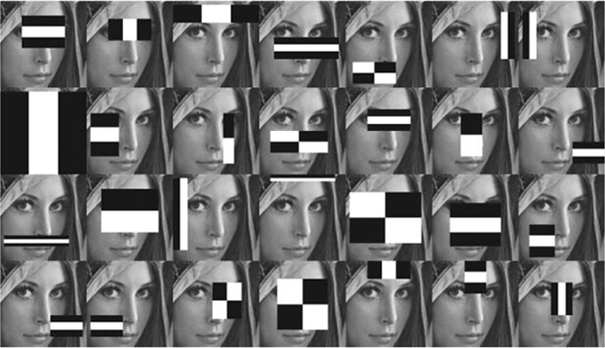


Рисунок 1.5 - Шаблоны каскадов Хаара [14]

Чтобы определять на изображении лица разного размера, используется метод скользящего окна. Вышеназванные примитивы вычисляются внутри окна, которое перемещается по изображению. После этого размер окна увеличивается и вычисления производятся еще раз. Повторяется это до тех пор, пока размер окна не совпадет с размером исходного изображения.

Для обучения распознавателя необходимо создать датасет, содержащий изображения лиц, подлежащих идентификации. Лицам определенного человека присваивается уникальный индекс. Для распознавания лиц был использован метод локальных бинарных шаблонов [15]. Его суть заключается в том, что изображение разбивается на части и в каждой такой части каждый пиксель сравнивается с соседними 8 пикселями. Если значение центрального пикселя больше соседнего, то на соответствующую позицию пишем 0, в противном случае — 1 (см. рисунок 1.6).

Таким образом, для каждого пикселя формируется определенное число. Далее на основе полученных чисел для всех частей, на которые мы разбивали изображение, считается гистограмма. Гистограммы со всех частей объединяются в один вектор, характеризующий все изображение в целом. Чтобы узнать, как сильно похожи два лица, для каждого из них необходимо вычислить такой вектор и сравнить их. Чем меньше векторное расстояние между гистограммами, тем более похожими будут считаться два лица для распознавателя.

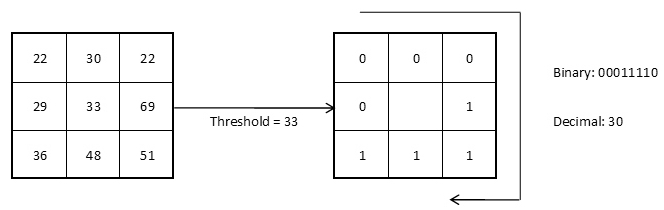


Рисунок 1.6 - Построение локальных бинарных шаблонов [15]

Заключительным этапом является распознавание лица. Каждое лицо, попавшее в камеру робота, будет проанализировано распознавателем. Для изображения человека, уже известного распознавателю, будет возвращен идентификатор этого человека, а также индекс, показывающий степень уверенности распознавателя.

На основе приведенного выше обзора и научных работ по сравнению каскадов Хаара и локальных бинарных шаблонов [16] был проведен сравнительный анализ примененных в проекте подходов и методов распознавания, основанных на сверточных нейронных сетях. Преимущества, недостатки и отличительные особенности каждого из методов приведены в таблице 1.1.

Таблица 1.1 – Сравнительный анализ методов распознавания лица

|  |  |  |  |
| --- | --- | --- | --- |
| Категория сравнения | Метод Виолы-Джонса | Local Binary Patterns | Нейронные сети |
| 1 | 2 | 3 | 4 |
| 1.Основное назначение | Детектирование лиц | Классификаторы и детекторы лиц | Классификаторы и детекторы лиц |
| 3.Скорость работы и качество распознавания | Имеет одно из лучших соотношений скорости работы/процента ошибок/затраченного времени на обучение. | В несколько раз быстрее, чем каскады Хаара, на 10-20% менее точный метод. | Такая система в перспективе может быть более эффективно, но необходимо много времени на обучение. |
| 3.Отличительные особенности | Использует признаки Хаара для детектирования образов. Представление изображения – интегральное. Позволяет вычислять признаки Хаара за постоянное время.  Можно обнаруживать более одного лица на изображении. | Производит вычисления в целых числах, в отличии от метода Хаара (числа с плавающей запятой), снижая нагрузку для встроенных и мобильных систем. | Точность результатов сильно зависит от типа сети и качества базы данных для ее обучения. Время на ее обучения также сильно отличается. |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 4. Достоинства | Используются изображения в интегральном представление, что позволяет быстро вычислять нужные объекты. | Время обучения может быть в 10 раз меньше, чем каскадов Хаара. | Обучение можно производить на доступных и бесплатных серверах, таких как Amazon AWS и аналогах [17]. |
| 5. Недостатки | Распознает черты лица под небольшим углом, примерно до 30 градусов. При угле наклона больше 30 градусов процент обнаружений резко падает, чувствителен к освещенности. | Чувствительны к условиям освещения и поворотам изображения [18]. | Инвариантность зависит от типа сети и качества базы.  Необходимое время и энергозатраты на обучение сети зависят от правильно подобранных гиперпараметров. |

Таким образом, комбинация методов Хаара и LBPH является оптимальным вариантом, так как данный подход не сложен в реализации и эффективен для встроенных и мобильных систем при правильном подходе.

В качестве преимуществ перед нейронными сетями следует отметить более низкое энергопотребление, меньшее время, необходимое для обучения, а также большее количество источников информации.

**1.4 Обзор пакетов для распознавания речи**

С учетом выбранного языка программирования для Raspberry Pi, где и будет производиться обработка голосовых команд, сервисы и пакеты для распознавания речи будут рассматриваться применительно к языку Python версии 3.

Распознавание речи подразделяется на так называемый офлайн и онлайн варианты.

Офлайн обработка голосовых команд происходит при помощи предустановленных в систему пакетов, включающий в себя словарь – собрание слов и фраз, которые может распознать обработчик. Такое распознавание речи используется, когда не доступно или невозможно подключение к сети Интернет.

Среди пакетов для офлайн распознавания речи наиболее подходящим для наших целей является Pocketsphinx. В отличие от аналогов, таких как wit и apiai, Pocketsphinx поддерживает русский язык и имеет большой языковой словарь, который по необходимости можно редактировать [19].

Онлайн вариант обработки голосовых команд предусматривает под собой отправку данных на вычислительные сервера для последующего получения текста из проанализированного аудио файла. В данном проекте рационально применение Google Speech API, не требующего использования ключей и паролей для подключения к серверам распознавания, таким образом, упрощая разработку, а также поддерживающего русский язык и позволяющего бесплатно распознавать аудио файлы длиной до 60 минут [20].

Пакет Speech Recognition позволяет использовать как вышеназванное API [21], так и выбранный офлайн вариант распознавания, выступая удобным вариантом для упрощения работы над блоком обработки голосовых команд.

**1.5 Результаты обзора литературы**

По итогам проведенного обзора литературы были определены наиболее

совпадающие по функционалу разрабатываемого программного обеспечения устройства для дальнейшего ориентирования в разработке, определены их преимущества и недостатки. Также были выбраны языки программирования и интегрированные среды разработки для прошивки Arduino Uno и мобильного приложения, проанализированы и подобраны методы для детектирования и распознавания лиц, а также пакеты для онлайн и офлайн распознавания речи.

**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

Изучив теоретические аспекты разрабатываемого программного обеспечения и разработав список требований, разбиваем систему на функциональные блоки. Функциональные блоки программной части «Мультизадачного робота» можно подразделить на 3 законченных модуля: программный модуль интерактивного взаимодействия для Raspberry Pi (включающий в себя функции Bluetooth сервера для мобильного приложения), мобильное приложение (для настройки робота) и прошивка микроконтроллера Arduino Uno. Взаимодействие между модулями будет обеспечено посредством спроектированной архитектуры команд движения (передача команд от Raspberry Pi 4 Model B к Arduino Uno) и настройки робота (передача команд и данных между Raspberry Pi 4 Model B и мобильным устройством), передача данных будет осуществляться протоколами RS-232 и RFCOMM соответственно.

Данное решение по разделению программной части на составные модули было обусловлено как аппаратной составляющей платформы, так и реализацией концепта автономного устройства, в первую очередь для обеспечения работы устройства в качестве системы удаленного мониторинга помещений.

Модуль интерактивного взаимодействия берет на себя задачи распознавания лиц, обработки речи, позволяя аппаратной платформе функционировать автономно, не используя вычислительные ресурсы мобильных и стационарных устройств.

Обработка команд движения осуществляется на Arduino Uno. Это позволяет освободить микрокомпьютер от необходимости выделения процессорного времени на обращение к датчикам и двигателям для перемещения в пространстве, что положительно сказывается на скорости работы устройства.

Мобильное клиентское приложение предоставляет возможность изменять настройки распознавателя и речи, а также управлять текущим подключение к сети без необходимости прямого доступа к операционной системе устройства или подключения к интернету.

Модуль интерактивного взаимодействия включает в себя следующие блоки:

* блок детектирование и классифицирования;
* блок предобработки изображения лица;
* блок обучения классификатора лиц;
* блок взаимодействия с Arduino Uno;
* блок автоматизированного перемещения;
* блок обработки голосовых команд;
* блок взаимодействия с мобильным приложением;
* блок настройки распознавания лица и речи.

Мобильное приложение подразделяется на:

* блок пользовательского интерфейса;
* блок взаимодействия с Raspberry Pi;
* блок подключения Bluetooth устройств.

Прошивка для Arduino Uno будет включать себя:

* блок взаимодействия с Raspberry Pi;
* блок обработки команд передвижения;
* блок обработки ситуаций столкновения.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.040 C1.

Каждый модуль предназначен для выполнения конкретных задач и взаимодействует с другими модулями путем обмена данными в специально спроектированном формате посредством стандартных протоколов.

Далее детально рассмотрим взаимодействие модулей и их функциональные блоки, а также выбор применяемых программных средств.

## 2.1 Модуль интерактивного взаимодействия

**2.1.1 Обоснование выбора библиотек и методов**

В качестве библиотеки для работы с камерой и обучения классификатора лиц была выбрана OpenCV [22]. OpenCV представляет собой библиотеку программного обеспечения для компьютерного зрения и машинного обучения с открытым исходным кодом. OpenCV была создана для обеспечения общей инфраструктуры приложений для компьютерного зрения и ускорения использования восприятия машины в коммерческих продуктах. Данная библиотека содержит более 2500 оптимизированных алгоритмов, которые включают в себя полный набор классических и современных алгоритмов машинного обучения и компьютерного зрения. Эти алгоритмы могут использоваться для идентификации объектов, обнаружения и распознавания лиц, отслеживания движения камеры, классификации действий человека в видео, извлечения 3D-моделей объектов и отслеживания движущихся объектов [22].

Для достаточно часто используемого алгоритма детектирования лиц, OpenCV имеет уже обученный классификатор, но также предоставляется возможность обучить свой собственный классификатор для обнаружения любого объекта [23].

**2.1.2 Функциональные блоки модуля**

*Блок детектирования и классифицирования* является составной частью сервиса для Raspberry Pi. Данный блок берет на себя основную работу с камерой, делая снимки изображений видеопотока и их дальнейшее распознавание. Кадры проходят через детектор, представляющий собой каскады Хаара, на предмет наличия искомых образов, в нашем случае лица, и, в случае положительного результата обнаружения, – изображение кадрируется и отправляется в блок предобработки изображения лица перед дальнейшим распознаванием. После получения результата предобработки полученные данные распознаются предварительно обученным классификатором. Результат работы классификатора представляет собой наиболее вероятный индекс распознанного пользователя и величину уверенности, предикат.

*Блок предобработки изображения лица* представляет собой модуль, выполняющий операции по нормализации освещенности над изображениями, полученными от детектора лиц. В работе применяется модифицированный алгоритм предобработки (см. рисунок 2.1), основанный на нормализации гистограммы изображения по сегментам [24]. Алгоритм включает следующие действия:

* корректирование уровня яркости;
* нормализацию гистограммы;
* нормализацию контрастности;
* удаление высокочастотных шумов на изображении.

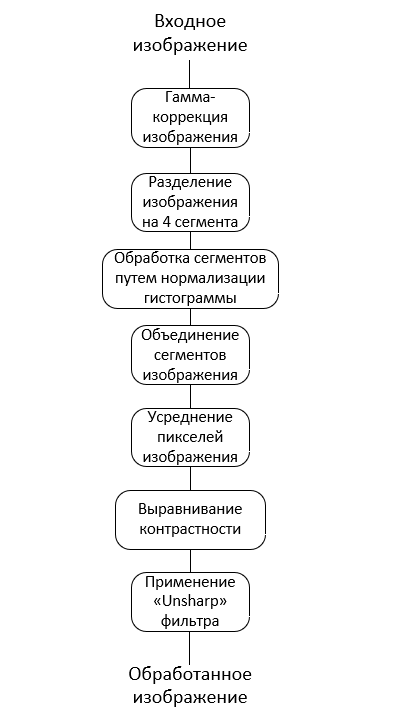


Рисунок 2.1 – Применяемый алгоритм предобработки изображений

Предварительная обработка изображения необходима для улучшения работы классификатора на основе LBPH, чувствительного к вышеперечисленным параметрам [16]. Предобработка изображений применяется в блоке детектирования и классифицирования образов и блоке обучения классификатора.

*Блок обучения классификатора лиц* начинает свою работу сразу после включения устройства, занося первого пользователя, попавшего в камеру, в датасет микрокомпьютера. Повторное обучение классификатора лиц с добавлением или удалением существующего пользователя также можно произвести по команде от мобильного приложения. Обучение классификатора проводится средствами OpenCV.

*Блок взаимодействия с Arduino Uno* обеспечивает передачу команд и прием данных о расстоянии до объектов по протоколу RS-232 через виртуальный COM порт. Arduino Uno, подключенный к Raspberry Pi через USB разъем, распознается системой Raspbian как ACM (abstract control mode) устройство [25], доступное для приема и передачи данных без необходимости установки дополнительных драйверов.

Обращение к виртуальному COM порту происходит при помощи библиотеки PySerial для языка Python [26]. Взаимодействие между микрокомпьютером и микроконтроллером происходит согласно спроектированной архитектуре команд для микроконтроллера Arduino Uno.

*Блок автоматизированного перемещения* предназначен для передвижения робота в помещении. Блок анализирует текущие данные о положении робота в пространстве, полученные из блока взаимодействия с Arduino, и принимает решение о дальнейшем передвижении в пространстве. В этой же части программного модуля выполняется алгоритм для поворотов головы робота, позволяющий увеличить углы обзора камеры для захвата изображения лиц и скорректировать текущее передвижение робота в пространстве на основе показаний ультразвукового датчика расстояний также расположенного в головной части устройства. Благодаря поворотам головы робот способен следить за обнаруженным пользователем и автоматически подъезжать или отдаляться по мере необходимости.

*Блок настройки распознавания лица и речи* обрабатывает команды, полученные от мобильного приложения. В задачи этого блока входит:

* добавление индексов новых пользователей в классификатор;
* удаление данных пользователя с выбранным индексом;
* переобучения классификатора на основе новых данных.
* включение или отключение распознавания речи;
* выбор ближнего или дальнего микрофона для распознавания речи.

*Блок взаимодействия с мобильным приложением* представляет собой сервис на языке Python, который управляет подключение Bluetooth устройств и передачу данных по протоколу RFCOMM, который будет подробнее рассмотрен в блоке взаимодействия с Raspberry Pi мобильного приложения. Прием и передача данных на Raspberry Pi происходит по профилю SPP (Serial Port Profile). Для этого на микрокомпьютере был установлен пакет python-bluez для языка Python версии 3 [27].

## Мобильное приложение

* + 1. **Обоснование выбора средств разработки и языка**

**программирования мобильного приложения**

В качестве среды разработки для мобильного приложения была выбрана Android Studio актуальной версии 3.6.2 для Windows 64-bit. Решение было продиктовано сразу несколькими аспектами.

Первой причиной является то, что Android Studio является официальной интегрированной средой разработки от компании Google, создателей операционной системы Android [28].

Вторая причина обусловлена популярностью данного средства среди разработчиков [29], что позволяет быстро и легко найти обучающие материалы на любую тематику разработки. Последний и не менее важный аспект выбора – полная бесплатность всего функционала данной интегрированной среды разработки [29].

Языком программирования была выбрана Java как один из официальных языков программирования для операционной системы Android [28].

**2.2.2 Функциональные блоки мобильного приложения**

*Блок пользовательского интерфейса* является основным блоком взаимодействия пользователя и мобильного приложения. Разработка данного блока производилась при помощи средства Layout Editor интегрированной среды разработки Android Studio. Layout Editor позволяет быстро создавать макеты, перетаскивая элементы пользовательского интерфейса в редактор визуального дизайна вместо написания XML-макета вручную. Данное программное средство предоставляет возможность просмотра макета на разных устройствах и версиях Android, позволяя динамически изменять размер макета, чтобы убедиться, что он корректно отображается на экранах разных размеров [30].

Layout Editor предоставляет возможность применения стилей и тем в Android приложении, которые позволяют отделить детали дизайна приложения от структуры и поведения пользовательского интерфейса, аналогично таблицам стилей в веб-дизайне.

Стиль — это набор атрибутов, которые определяют внешний вид для одного представления. Стиль может указывать такие атрибуты, как цвет шрифта, размер шрифта, цвет фона и многое другое.

Тема — это тип стиля, который применяется ко всей иерархии приложения или представления. Когда стиль применяется в качестве темы, каждое представление в приложении или действии применяет каждый поддерживаемый атрибут стиля. Темы также могут применять стили к элементам без вида, таким как строка состояния и фон окна.

По умолчанию Android Studio поддерживает разработку графического интерфейса приложения по концепции Material Design. Material Design — это руководство по визуальному, динамическому и интерактивному проектированию на разных платформах и устройствах [31].

*Блок взаимодействия с Raspberry Pi* представляет из себя модуль, отвечающий за передачу и прием данных по протоколу RFCOMM и настройку Wi-Fi соединения робота, путем редактирования файла WPA-Supplicants на микрокомпьютере Raspberry Pi.

RFCOMM (Radio Frequency Communication) [32] является одним из протоколов передачи данных по стандарту связи Bluetooth. RFCOMM производит эмуляцию работы с последовательным портом поверх протокола L2CAP. Данный протокол основан на ETSI-стандарте TS 07.10. RFCOMM представляет собой простой транспортный протокол, с дополнительными возможностями по эмуляции 9 цепей последовательных портов RS-232 и поддерживает одновременно до 60 каналов между двумя устройствами Bluetooth. RFCOMM предназначен для сокрытия приложений, использующих последовательные порты устройств, в которых они расположены.

Также было решено разработать архитектуру команд и структуры данных для более удобной и унифицированной передачи данных между вычислительными узлами. Команды вызываются пользователем путем выбора элементов графического интерфейса приложения.

Языком разработки мобильного приложения была выбрана Java, поэтому для выполнения передачи данных по стандарту Bluetooth используются стандартные библиотеки android.bluetooth.BluetoothAdapter, android.bluetooth.BluetoothDevice и android.bluetooth.BluetoothSocket [33].

*Блок подключения Bluetooth устройств* включает в себя поиск доступных для подключения устройств и установку соединения. Данный блок считывает список сопряжений из операционной системы смартфона, который будет отображен в блоке пользовательского интерфейса. Далее пользователь может выполнить подключение к выбранному Bluetooth-устройству. Для возможности использования Bluetooth приложению также нужны соответсвующие разрешения - android.permission.BLUETOOTH и android.permission.BLUETOOTH\_ADMIN [33].

## 2.3 Прошивка Arduino Uno

*Блок взаимодействия с Raspberry Pi* производитприем команд и передачу данных о расстоянии до объектов по протоколу RS-232 через виртуальный COM порт. Выбор протокола передачи данных обоснован аппаратной структурой проекта. Передача данных происходит в установленном формате согласно спроектированной архитектуре команд для микроконтроллера. Команды подразделяются на перемещение робота, поворот сервоприводов, запрос показаний датчиков и уровня заряда батареи.

*Блок обработки команд передвижения* обеспечивает перемещение робота в пространстве и повороты головной части корпуса. Для управления двигателями постоянного тока была использована библиотека AFMotor, которая позволяет задавать движение двигателя вперед или назад с выбранной скоростью в диапазоне от 0 до 255.

Для работы с сервоприводами используется библиотека Adafruit\_PWMServoDriver. Библиотека работает по протоколу I2C [34]. Она позволяет устанавливать сервопривод в заданное положение через установку длительности импульса. Данная величина находится в промежутке между минимальным и максимальным значением, полученными эмпирическим способом. После поступления команды смены угла поворота сервопривода от микрокомпьютера, величина в градусах переводится в длительность импульса при помощи функции map.

*Блок обработки ситуаций столкновения* выполняет остановку передвижения устройства и автоматический объезд препятствий на основе показаний инфракрасных и ультразвуковых датчиков. Показания инфракрасных датчиков в прошивке микроконтроллера соответствуют одной из двух константных величин языка программирования Arduino – HIGH (препятствие) и LOW (препятствие не обнаружено). В случае обнаружения величины HIGH на одном из ИК датчиков движение робота останавливается для дальнейшего объезда препятствия. С другой стороны, ультразвуковой датчик позволяет рассчитывать расстояния до объекта путем вычисления времени на отправку и возвращение ультразвукового сигнала, отраженного от объекта [35]. Показания датчика ультразвука используются для определения направления движения для объезда препятствий при принудительной остановке устройства.

**2.4 Результаты системного проектирования**

В рамках системного проектирования была разработана гибкая и расширяемая структура для 3 модулей проекта, взаимодействующих между собой посредством различных протоколов. Разделение программной части на модули распределяет нагрузку между вычислительными элементами системы для увеличения скорости работы. Также данная структура позволяет в дальнейшем легко добавлять новые блоки в существующие модули для доработки функционала программного обеспечения.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе будет рассмотрены основные функциональные составляющие проектируемого ПО.

Мобильное приложение и прошивка для Arduino Uno реализую объектно-ориентированную парадигму программирования. Модуль интерактивного взаимодействия, в свою очередь, сочетает императивную и функциональную парадигмы.

Компоненты мобильного приложения и прошивки Arduino Uno подразделяются на классы, описание которых приведено на чертежах ГУИР.400201.040 РР.1 и ГУИР.400201.040 РР.2 соответственно.

У каждого класса присутствует набор методов, которые устанавливают или возвращают значения или объекты одноименных полей. Описание данных методов, а также автоматически сгенерированных обработчиков без дополнительных действий, деструкторов, конструкторов по умолчанию и копирования будет опущено.

**3.1 Взаимодействие модулей системы**

Программное обеспечение «Мультизадачного робота» структурно разделено на 3 модуля, взаимодействующие между собой при помощи наборов команд.

Команды для модуля интерактивного взаимодействия и прошивки Arduino Uno представлены в таблице 3.1. Формулировка команд и их назначение обусловлены их ограниченным числом и применением только для перемещения и снятия показаний датчиков аппаратной платформы.

Таблица 3.1 – Архитектура набора команд USB-Serial соединения

|  |  |  |
| --- | --- | --- |
| Команда | Расшифровка | Пояснение |
| 1 | 2 | 3 |
| s | stop | Остановка. |
| f | forward | Движение вперед. |
| b | back | Движение назад. |
| o | observe | Возвращает строку, содержащую показания датчика освещенности, инфракрасных и ультразвукового сенсоров. Формат [i:x,x,x,x,u:x,l:x], где x – величина полученного значения, i – идентификатор значений инфракрасных датчиков, u – идентификатор значений для ультразвукового датчика, l – идентификатор значений датчика освещенности. |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| t | track battery | Возвращает текущий процент заряда батареи. Формат [b:x], где x – величина полученного значения, b – идентификатор показаний уровня заряда. |
| l[0...1000] | left | Поворот налево в течение указанного количества миллисекунд. |
| r[0...1000] | right | Поворот направо в течение указанного количества миллисекунд. |
| v[0...180] | vertical | Поворот сервопривода, закрепленного вертикально, на указанную позицию в градусах. |
| h[0..180] | horizontal | Поворот сервопривода, закрепленного горизонтально, на указанную позицию в градусах. |
| g[0…100] | go | Движение вперед в течение заданного времени в миллисекундах. |
| r[0…100] | return | Движение назад в течение заданного времени в миллисекундах. |

Команды для мобильного приложения и модуля интерактивного взаимодействия представлены в таблице 3.2. Данный набор подразделяется на команды, отправляемые мобильным приложением, и команды, отправляемые модулем интерактивного взаимодействия.

Таблица 3.2 – Архитектура набора команд Bluetooth соединения

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | Параметр | Возвращаемое значение | Пояснение |
| 1 | 2 | 3 | 4 |
| Команды, отправляемые мобильным приложением | | | |
| /status/ |  | JSON строка формата:  { sound:<Integer>,  mic: < Integer >,  facerec: < Integer >,  submon: < Integer>,  opmode: < Integer >,  battery: <Integer>} | Получение текущих настроек и параметров робота, таких как включение звука, микрофонов, распознавания лиц, мониторинга пользователя, режим функционирования, уровень заряда батареи. |
| /sound/<enable>/ | 0-1 | - | Включение или отключение звукового сопровождения. |
| /mic/<enable>/ | 0-1 | - | Включение или отключение записи звука с микрофонов. |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| /facerec/<enable>/ | 0-1 | - | Включение или отключение распознавания лиц. |
| /submon/<enable>/ | 0-1 | - | Включение или отключение мониторинга пользователя. |
| /opmode/<number>/ | 0-1 | - | Выбор режима функционирования робота. |
| /subjects/ |  | JSON строка формата:  { subjects:[<Integer>,  ,< Integer >,  < Integer >  …  <Integer>]  } | Получение списка номеров субъектов, в данный момент используемых для распознавания лиц. |
| /photo/<number>/ | 0..x | Изображение формата BMP. | Получение изображения субъекта по номеру в датасете. |
| /add/ |  | - | Добавление нового субъекта распознавания в датасет. Возвращает результат выполнения операции. |
| /remove/<number>/ | 0 ..x | - | Удаление субъекта распознавания по номеру в датасете. Возвращает результат выполнения операции. |
| /wifi\_connect/ |  | JSON строка формата:  { status : String} | Старт установки соединения Wi-Fi. Возвращает статус операции. |
| /restart/ |  | - | Рестарт робота. |
| /find/ |  | - | Включение или выключение звукового сигнала для поиска робота. |
| Команды, отправляемые серверной частью модуля интерактивного взаимодействия | | | |
| /wait\_ssid/ |  | bytes[] ssid | Запрос на получения SSID Wi-Fi сети. |
| /wait\_psk/ |  | bytes[] psk | Запрос на получение PSK Wi-Fi сети. |

**3.2 Описание функциональности модуля интерактивного взаимодействия**

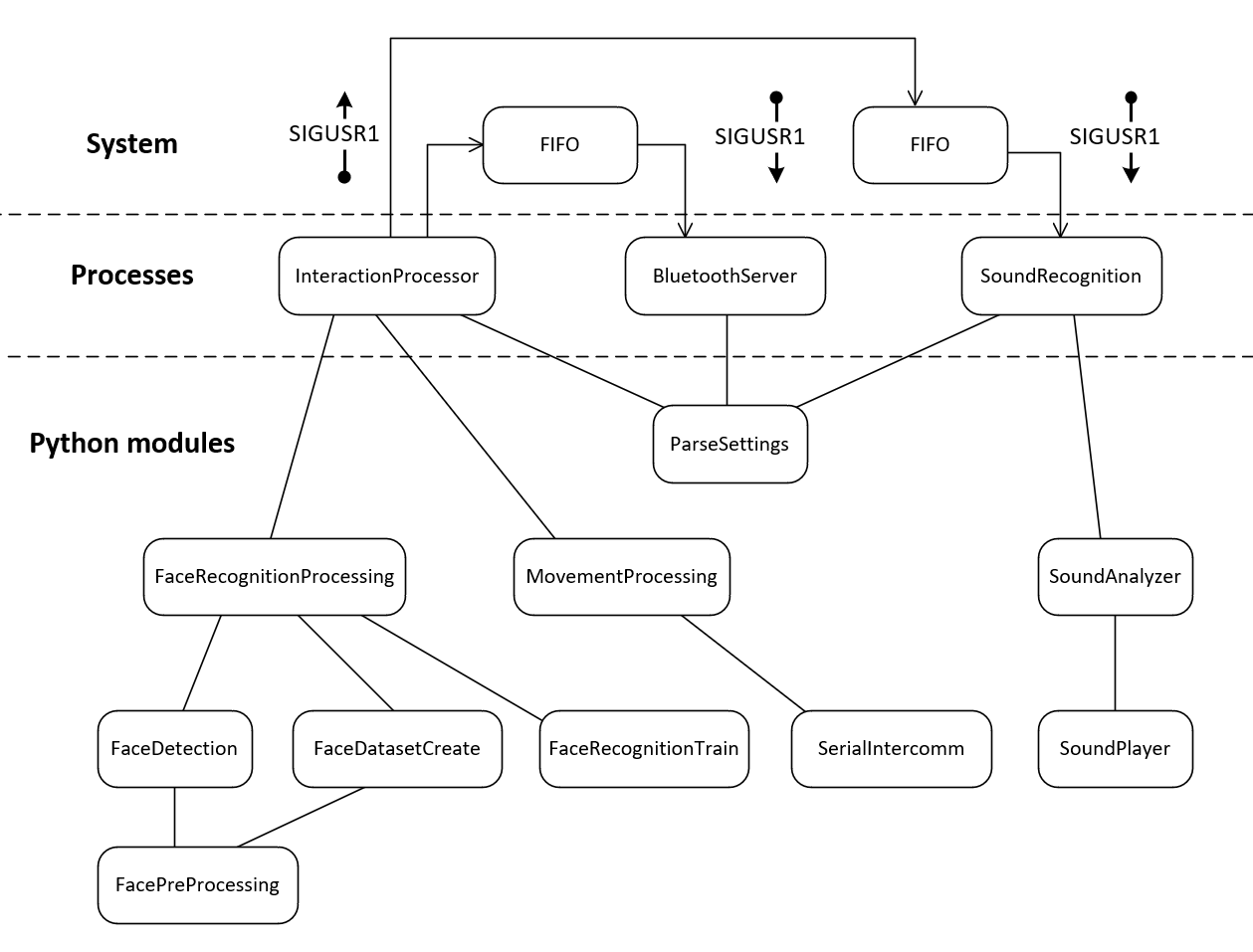


Рисунок 3.1 – Взаимодействие программных частей модуля интерактивного взаимодействия

Модуль интерактивного взаимодействия представляет собой совокупность нескольких сервисов в операционной системе Raspbian аппаратной платформы мультизадачного робота, выполняющих одноименные модули на языке Python. Такое решение позволяет разделить одновременно выполняемые задачи по области применения для параллельного исполнения. Взаимодействие между процессами происходит посредством системных средств, таких как именованные каналы, «tointeraction» для процессов BluetoothServer и InteractionProcessor и «tosound» для процессов BluetoothServer и SoundRecognition, и асинхронные пользовательские сигналы (см. рисунок 3.1). Данное взаимодействие отображено на чертеже ГУИР.400201.040 PP.3.

Основными функциональными частями модуля интерактивного взаимодействия являются:

1. InteractionProcessor – процесс одноименного модуля, занимающийся обработкой операций взаимодействия с пользователем, таких как распознавание лиц, перемещение в пространстве, выделенные в отдельные модули на языке Python, указанные на рисунке. Модуль FaceRecognitionProcessing включает в себя модули для предобработки изображения, детектирования лиц, создания датасета изображений субъектов и обучения классификатора. Модуль MovementProcessing посылает команды передвижения и получает показаний датчиков и уровня батареи. Работа с последовательным портом выделена в отдельный подмодуль MovementProcessing – SerialIntercomm. В случае изменения пользователем настроек в мобильном приложении в процесс приходит сигнал SIGUSR1 от сервиса BluetoothServer и InteractionProcessor перезагружает настройки или вызывает соответствующие команды, такие как «/add/», «/remove/<number>/», в зависимости от данных, поступивших в канал «tointeraction».
2. BluetoothServer – процесс Bluetooth сервера, обрабатывающий команды, поступающие от подключенного мобильного клиента. BluetoothServer взаимодействует с другими процессами посредством вызова пользовательского сигнала SIGUSR1 и последующей записи команды и параметра в именованный канал для конкретного процесса. Команды «/subjects/», «/photo/<number>/», «/wifi\_connect/» и «/restart/» выполняются непосредственно в процессе Bluetooth сервера.
3. SoundRecognition – процесс, обеспечивающий распознавание голоса и обработку коротких голосовых команд через подмодуль SoundAnalyzer. Также этот процесс воспроизводит голосовые ответы на команды и звуковое сопровождение посредством подмодуля SoundPlayer. Список голосовых команд «Мультизадачного робота» отображен в таблице 3.3. Настройки работы процесса загружаются в момент инициализации сервиса в системе при помощи модуля ParseSettings. В случае изменения пользователем настроек в мобильном приложении в процесс приходит сигнал SIGUSR1 от процесса BluetoothServer и SoundRecognition перезагружает настройки или выполняет команду «/find/», в зависимости от данных, поступивших в канал «tosound».

Таблица 3.3 – Голосовые команды

|  |  |  |
| --- | --- | --- |
| Команда | Фразы для вызова | Пояснение |
| 1 | 2 | 3 |
| Hello | «Привет, Робо», «Привет», «Hi, Robo», «Hi» или «Hello» | Робот озвучивает фразу «Привет». |

Продолжение таблицы 3.3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Time | «Скажи время», «Текущее время», «Current time», «Tell time» | Робот озвучивает текст, содержащий текущее системное время модуля интерактивного взаимодействия. Форма ответа: «Текущие время HH:MM, где HH – часы, MM – минуты. |
| HRU | «Робо, как дела?», «Как дела?», «How are you?». | Робот озвучивает одну из фраз «Все хорошо», «Пора бы отдохнуть», «А у тебя как?». |
| Settings | «Скажи настройки», «Текущие настройки»,  «Опции» «Current settings», «Settings» | Робот озвучивает текст, содержащий показания текущих настроек модуля интерактивного взаимодействия. Форма ответа: «Текущие настройки – Звук :<x>, Микрофон: <x>, Распознавание: <x>, Мониторинг: <x>, Режим: <x>», где: <x> - включенность опции. |

**3.3 Описание функциональности прошивки Arduino Uno**

Взаимодействие объектов классов прошивки отражено на чертеже ГУИР.400201.040 РР.5.

Класс ArduinoInteractionManager (см. таблицы 3.4 и 3.5) выполняет функции обработки текущего состояния микроконтроллера объектов классов SerialIntercommunicator и MovementController.

Таблица 3.4 – Описание методов класса ArduinoInteractionManager

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| begin |  | void | Создание инстанций классов SerialIntercommunicator и MovementController. |
| run |  | void | Выполнение одиночной итерации функции run для объектов serialcomm и movement. |

Таблица 3.5 – Описание полей класса ArduinoInteractionManager

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| serialcomm | SerialIntercommunicator | private | Обработчик команд, поступающих через serial port. |
| movement | MovementController | private | Контроллер для обработки ситуаций столкновения. |

Класс BatteryController представлет собой обработчик показаний заряда батареи «Мультизадачного робота», получаемые из класса BatteryChargeSensor. Описание класса приведено в таблицах 3.6 и 3.7.

Таблица 3.6 – Описание методов класса BatteryController

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| run |  | byte | Циклический опрос состояния объекта класса BatteryChargeSensor и дальнейшая интерполяция полученной величины относительно графика разряда литий-ионных батарей. |

Таблица 3.7 – Описание полей класса BatteryController

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| sensor | BatteryChargeSensor | private | Объект для считывания уровня заряда батареи. |

Класс BatteryChargeMeasurer считывает показания с аналогового пина микроконтроллера и сохраняет до следующего считывания. Описание класса приведено в таблицах 3.8 и 3.9.

Таблица 3.8 – Описание методов класса BatteryChargeMeasurer

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| BatteryChargeMeasurer | int voltPin | void | Конструктор класса, принимающий номер аналогового пина для снятия показаний. |
| init | int voltPin | void | Инициализация переменной mVoltPin, хранящей номер пина для считывания показаний. |
| getCurrentState |  | float | Возвращает значение показания уровня напряжение, сохраненного в поле voltage. |
| read |  | float | Считывает показание уровня напряжения аналогового пина в поле voltage. Возвращает полученное значение. |

Таблица 3.9 – Описание полей класса BatteryChargeMeasurer

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mVoltPint | int | private | Номер аналогового пина. |
| voltage | float | private | Показание уровня напряжение на аналоговом пине. |

SaveStateSensor – абстрактный класс, описывающий поведение датчика, сохраняющего состояние после снятия показаний (см. таблицы 3.10 и 3.11).

Таблица 3.10 – Описание методов класса SaveStateSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| getCurrentState |  | int | Возвращает значение переменной state. |
| read |  | void | Чистая виртуальная функций снятия показаний датчика. |

Таблица 3.11 – Описание полей класса SaveStateSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| state | int | private | Показание датчика. |

Класс InfraSensor наследуется от абстрактного класса SaveStateSensor и реализует методы для снятия показаний с инфракрасных датчиков. Описание класса приведено в таблицах 3.12 и 3.13.

Таблица 3.12 – Описание методов класса InfraSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| init | int centerPin, int leftPin, int rightPin, int backPin | void | Инициализация переменных mCenterPin, mLeftPin, mRightPin, mBackPin, хранящих номера пинов для считывания показаний. |
| read |  | void | Метод для снятия показаний с инфракрасных датчиков. Показания записываются в поле state, наследуемое из родительского класса. |

Таблица 3.13 – Описание полей класса InfraSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mCenterPin | int | private | Номер аналогового пина для центрального датчика. |
| mLeftPin | int | private | Номер аналогового пина для левого датчика. |
| mRightPin | int | private | Номер аналогового пина для правого датчика. |
| mBackPin | int | private | Номер аналогового пина для заднего датчика. |

Класс PhotoSensitiveSensor наследуется от абстрактного класса SaveStateSensor и реализует методы для снятия показаний с датчика освещенности. Описание класса приведено в таблицах 3.14 и 3.15.

Таблица 3.14 – Описание методов класса PhotoSensitiveSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| init | int sensorPin | void | Инициализация переменной, хранящей номер пина для считывания показаний. |
| read |  | void | Метод для снятия показаний с датчика освещенности. Показания записываются в поле state, наследуемое из родительского класса. |
| normalizeData |  | void | Метод для получения показаний датчика, нормализованных относительно среднего уровня освещенности. |

Таблица 3.15 – Описание полей класса PhotoSensitiveSensor

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mSensorPin | int | private | Номер аналогового пина датчика освещенности. |

Класс UltraSound наследуется от абстрактного класса SaveStateSensor и предоставляет реализацию методов для считывания показаний с датчика освещенности. Описание класса приведено в таблицах 3.16 и 3.17.

Таблица 3.16 – Описание методов класса UltraSound

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| init | int sensorPin, int echoPin | void | Инициализация полей mTrigPin и mEchoPin, хранящей номера пинов для считывания показаний. |
| read |  | void | Метод для снятия показаний с ультразвукового датчика. Показания записываются в поле state, наследуемое из родительского класса. |

Таблица 3.17 – Описание полей класса UltraSound

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mTrigPin | int | private | Номер цифрового пина , принимающего на вход импульс для ультразвукового датчика. |
| mEchoPin | int | private | Номер цифрового пина, возвращающего эхо импульса, посланного на ультразвуковой датчик. |
| mDuration | long | private | Длительность возврата импульс. |

InteractionCommand – базовый абстрактный класс для команд, выполняемых объектами классов SerialIntercommunicator и MovementController (см. таблицы 3.18 и 3.19).

Таблица 3.18 – Описание методов класса InteractionCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| execute |  | void | Чисто виртуальная функция выполнения команды. |

Класс MotorMovementCommand наследуется от абстрактного класса InteractionCommand и реализует выполнение команды перемещения, путем подачи скорости движения и направления действия. В классе используются объекты из библиотеки AFMotor для управления скоростью моторов. Описание класса приведено в таблицах 3.19 и 3.20.

Таблица 3.19 – Описание методов класса MotorMovementCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| execute |  | void | Выполнение команды перемещения. Направление и длительность перемещения определяются значениями полей currentDirection и mTimeDelay. |
| init | int direction, long timeDelay | MotorMovementCommand \* | Инциализация полей currentDirection и mTimeDelay. Возвращает указатель на объект. |
| Private методы | | | |
| stop |  | void | Остановка моторов |
| forward |  | void | Установка моторов на движение моторов вперед |
| back |  | void | Установка моторов на движение моторов назад |
| left |  | void | Установка моторов на поворот робота против часовой стрелки |
| right |  | void | Установка моторов на поворот робота по часовой стрелке. |
| goForward | long time | void | Движение вперед заданный промежуток времени. |

Продолжение таблицы 3.19

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| goBack | long time | void | Движение назад заданный промежуток времени. |
| turnLeft | long time | void | Поворот налево заданный промежуток времени. |
| turnRight | long time | void | Поворот направо заданный промежуток времени. |

Таблица 3.20 – Описание полей класса MotorMovementCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| running | static boolean | private | Индикатор движения. |
| currentDirection | static int | private | Выбранное направление движения. |
| speed | static int | private | Скорость движения моторов |
| mTimeDelay | long | private | Время движения по выбранному направлению. По умолчанию равно нулю. |

Класс ServoMovementCommand наследуется от абстрактного класса InteractionCommand и реализует выполнение команды поворота сервоприводов, путем установки длительности и величины ШИМ импульса через библиотеку Adafruit\_PWMServoDriver (см. таблицы 3.21 и 3.22).

Таблица 3.21 – Описание методов класса ServoMovementCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| execute |  | void | Выполнение команды перемещения. |
| init | int servoNum, int deg | ServoMovementCommand \* | Инциализация полей mServoNum и mDeg. Возвращает указатель на объект. |

Продолжение таблицы 3.21

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Private методы | | | |
| servoVertical | int deg | void | Подача ШИМ импульса для поворотов по вертикали на переданное количество градусов. |
| servoHorizontal | int deg | void | Подача ШИМ импульса для поворотов по горизонтали, на переданное количество градусов. |

Таблица 3.22 – Описание полей класса ServoMovementCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mDeg | int | private | Угол поворота. |
| mServoNum | int | private | Номер сервопривода для выполнения команды. |

Класс LocationAnalysisCommand наследуется от абстрактного класса InteractionCommand и реализует выполнение команды сбора показаний датчиков и формирование строки результата, формат которой указана в таблице 3.2 в строке команды observe (см. таблицы 3.23 и 3.24).

Таблица 3.23 – Описание методов класса LocationAnalysisCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| execute |  | void | Выполнение команды снятия показаний с датчиков. Показания датчиков хранятся в соответствующих полях класса. |
| init | int servoNum, int deg | LocationAnalysisCommand \* | Инициализация объектов датчиков. Возвращает указатель на объект. |
| getResultString |  | char \* | Формирует и возвращает строку результат снятия показаний с датчиков. |

Таблица 3.24 – Описание полей класса LocationAnalysisCommand

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| USSensor | UltraSoundSensor | private | Объект для считывания и хранения показаний ультразвукового датчика. |
| PSSensor | PhotoSensitiveSensor | private | Объект для считывания и хранения показаний датчика освещенности. |
| IRSensor | InfraSensor | private | Объект для считывания и хранения показаний инфракрасных датчиков. |
| mLocationState | String | private | Строка результата собранных показаний датчиков. |

Класс MovementController контролирует показания датчиков и обрабатывает ситуации столкновения. Описание класса приведено в таблицах 3.25 и 3.26.

Таблица 3.25 – Описание методов класса MovementController

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| run |  | void | Обработка показаний датчиков и дальнейший вызов команд для предотвращения и избегания столкновений. |
| checkUltraSoundSensor |  | boolean | Проверка показаний ультразвукового датчика на минимальное расстояние для движения вперед. |
| checkInfraSensors | boolean allowForwardMovement | int | Проверяет показания инфракрасных сенсоров и возвращает значение обнаруженных препятствий, с учетом разрешенного или запрещенного движения вперед. |

Таблица 3.26 – Описание полей класса MovementController

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| servoCommand | ServoMovementCommand | private | Команда для поворота сервоприводов. |
| motorCommand | MotorMovementCommand | private | Команда для перемещения. |
| USSensor | UltraSoundSensor | private | Объект для считывания и хранения показаний ультразвукового датчика. |
| IRSensor | InfraSensor | private | Объект для считывания и хранения показаний инфракрасных датчиков. |
| PSSensor | PhotoSensitiveSensor | private | Объект для обработки показаний датчика освещенности. |

Класс SerialIntercommunicator осуществляет прием, оправку и выполнение команд, согласно архитектуре набора команд, представленных в таблице 3.1. SerialIntercommunicator выполняет работу с классом Serial. Описание класса приведено в таблицах 3.27 и 3.28.

Таблица 3.27 – Описание методов класса SerialIntercommunicator

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| run |  | void | Обработка показаний датчиков и дальнейший вызов команд для предотвращения и избегания столкновений. |
| read |  | void | Чтение из последовательного порта в mCommand и mParameter. |
| write | char \* data | void | Запись строки в последовательный порт. |
| write | byte data | void | Запись байта в последовательный порт. |

Таблица 3.28 – Описание полей класса SerialIntercommunicator

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| 1 | 2 | 3 | 4 |
| servoCommand | ServoMovementCommand | private | Команда для поворота сервоприводов. |
| motorCommand | MotorMovementCommand | private | Команда для перемещения. |

Продолжение таблицы 3.28

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| locationCommand | LocationAnalysisCommand | private | Команда для снятия показаний датчиков. |
| batteryController | BatteryController | private | Объект для считывания процента заряда батарей. |
| mCommand | String | private | Строка для хранения команды, считанной с последовательного порта. |
| mParameter | String | private | Строка для хранения параметра команды. |

**3.4 Описание функциональности мобильного приложения**

Последовательность взаимодействия объектов классов мобильного приложения при подключении к Bluetooth серверу отражена на чертеже ГУИР.400201.040 РР.4.

Класс CommunicationTask унаследован от AsyncTask и обеспечивает подключение и разрыв соединения между мобильным устройством и Bluetooth сервером модуля интерактивного взаимодействия. Подключение к серверу осуществляется в отдельном потоке. В случае разрыва сети в момент передачи или чтения данных применяется метод void reconnect(). Описание класса приведено в таблицах 3.29 и 3.30.

Таблица 3.29 – Описание методов класса CommunicationTask

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Protected методы | | | |
| onPreExecute |  | void | Выполнение действия до исполнения асинхронной задачи. Отображает диалог с прогрессом выполнения подключения. |
| onPostExecute | Void result | void | Выполнение действия после исполнения задачи. Скрывает диалог с прогрессом выполнения. Если подключение не было успешным – закрывает окно главного меню и возвращает пользователя на экран выбора устройства для подключения. |

Продолжение таблицы 3.29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | 4 |
| doInBackground | Void … devices | Void | | Метод для подключения в Bluetooth устройства, выполняется в отдельном от UI потоке. |
| Public методы | | | | |
| write | String s | void | Запись строки в Bluetooth сокет. | |
| write | Byte b | void | Запись байта в Bluetooth сокет. | |
| read |  | int | Вычитка из Bluetooth сокета. Производит очистку входного потока в случае неуспешного выполнения операций считывания. | |
| read | byte[] buffer | int | Чтения данных из входного потока в буфер байт. | |
| read | byte[] buffer, int off, int len | int | Чтение данных из входного потока в буфер байт, заданной длинной и с заданным отступом для буфера. | |
| isAvailable |  | int | Метод для проверки наличия в сокете данных для чтения. | |
| isConnected |  | boolean | Метод проверки подключения. | |
| disconnect |  | void | Отключения от Bluetooth устройства. | |

Окончание таблицы 3.29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | | 2 | 3 | 4 |
| Private методы | | | | |
| reconnect |  | |  | Переподключение к Bluetooth устройству. |
| popup | String | | s | Отображение всплывающего сообщения с текстом, заданным в строке s. |

Таблица 3.30 – Описание полей класса CommunicationTask

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| deviceUUID | UUID | private static final | Уникальный UUID для RFCOMM (SPP) соединения. |
| mConnected | boolean | private | Статус подключения. |
| mBluetoothAdapter | BluetoothAdapter | private | Объект Bluetooth адаптера мобильного устройства. |
| mBluetoothSocket | BluetoothSocket | private | Клиентский сокет соединения. |
| mAddress | String | private | Адрес устройства для подключения. |
| mProgressDialog | ProgressDialog | private | Диалог с прогрессом выполнения подключения. |

Класс CommunicationActivity – абстрактный класс, который наследуется от AppCompatActivity, и реализует запуск асинхронной задачи объекта класса CommunicationTask, а также контролирует отключение соединения. Описание класса приведено в таблицах 3.31 и 3.32.

Таблица 3.31 – Описание методов класса CommunicationActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| onDestroy |  | void | Метод, вызываемый при уничтожении окна. Выполняет закрытие Bluetooth соединения и освобождение ресурсов. |

Продолжение таблицы 3.31

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| onBackPressed | Void … devices | Void | Вызов кнопки назад. Выполняет закрытие Bluetooth соединения и освобождение ресурсов. |
| Private методы | | | |
| initializeBluetoothConnection | Void result | void | Выполнение асинхронной задачи CommunicationTask и инициализация полученного подключения в классе ClientApplication.. |

Таблица 3.32 – Описание полей класса CommunicationActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mBluetoothConnection | CommunicationsTask | final private | Текущее Bluetooth соединение. |
| mDeviceAddress | String | private | Адрес устройства для подключения. Передается из DeviceListActivity через Intent. |

Класс ClientApplication наследуется от класса Application и реализует паттерн одиночка для обеспечения наличие единственной инстанции приложения. Описание класса приведено в таблицах 3.33 и 3.334.

Таблица 3.33 – Описание методов класса ClientApplication

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| onCreate |  | void | Создание единственной инстанции класса. Создает новый объект класс Settings для последующей инициализации. |

Продолжение таблицы 3.33

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| savePreferences | String key, int value | void | Сохранение значения ключа в хранилище Preferences мобильного устройства. |
| Private методы | | | |
| loadPreferences | String key | int | Загрузка значения по ключу из хранилища Preferences мобильного устройства. |
| loadRoboSettingsFromPreferences |  | void | Загрузка настроек в mRoboSettings из Preferences. |
| saveRoboSettingsInPreferences |  | void | Загрузка настроек из mRoboSettings в Preferences. |

Таблица 3.34 – Описание полей класса ClientApplication

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mBluetoothConnection | CommunicationsTask | protected | Текущее Bluetooth соединение. |
| sInstance: | ClientApplication | static private | Статический объект для обращения к классу ClientApplication. |
| mRoboSettings | Settings | private | Настройки работы модуля интерактивного взаимодействия. |
| mRoboSettingsLoaded | boolean | private | Индикатор инициализации настроек из Preferences для ленивой загрузки. |

Класс Settings является инкапсуляцией полей объекта настроек модуля интерактивного взаимодействия. Данный класс предназначен для расшифровки полученной от сервера JSON строки, содержащей текущие настройки робота, и сохранения объекта в хранилище Preferences во время выключения мобильного приложения (см. таблицу 3.35).

Таблица 3.35 – Описание полей класса Settings

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| 1 | 2 | 3 | 4 |

Продолжение таблицы 3.35

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Sound | Integer | package private | Настройки для включения или отключения звукового сопровождения. |
| mic | Integer | package private | Настройки для включения или отключения работа микрофонов. |
| facerec | Integer | package private | Настройки для включения или отключения распознавания лиц. |
| submon | Integer | package private | Включение или отключение мониторинга пользователей. |
| opmode | Integer | package private | Выбор режима работы робота – Standby или Standalone. Механизм работы режимов описан далее в пункте «Руководство пользователя». |
| battery | Integer | package private | Уровень заряда батареи. |

Класс MainMenuActivity наследуется от класса CommunicationActivity и представляет собой окно главного меню приложения. Кроме отображения для пользователя списка меню и уровня заряда батареи контролирует статус работы модуля интерактивного взаимодействия. Описание класса приведено в таблицах 3.36 и 3.37.

Таблица 3.36 – Описание методов класса MainMenuActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Protected методы | | | |
| onCreate | Bundle savedInstanceState | void | Создание окна. Генерация списка настроек и инициализация слушателей для перехода в пункты меню. |
| onPause |  | void | Обработчик перехода приложения в режим Paused. Приостанавливает обработчик получения обновлений от подключенного устройства. |

Продолжение таблицы 3.36

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| onResume |  | void | Обработчик возобновления работы приложения. Производит переподключение соединения и возобновляет работу обработчик обновлений. |
| Public методы | | | |
| onDestroy |  | void | Метод, вызываемый при уничтожении окна. Выполняет закрытие Bluetooth соединения и освобождение ресурсов. |
| showPopupWindowClick |  | void | Метод для отображения диалога с краткими характеристика приложения и устройства по нажатию на пункт дополнительного меню «About Robo». |
| Private методы | | | |
| restartRoboCommand |  | void | Метод для отправки команды рестарта системы робота. |
| updateRoboSettings |  | Void | Обработчик получения обновлений от BluetoothServer части модуля интерактивного взаимодействия. Опрашивает модуль с определенным периодом для стабильного поддержания соединения и получения точных показаний уровня заряда батареи. |

Таблица 3.37 – Описание полей класса MainMenuActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mMainLayout | LinearLayout | private | Контейнер окна, содержащий все элементы меню. Используется для отображения диалога «About Robo». |
| mUpdateHandler | Handler | private final | Обработчик для получения обновлений от модуля интерактивного взаимодействия. Выполняется в отдельном потоке с заданной периодичностью. Непосредственно взаимодействует с классом ClientApplication для проверки стабильного соединения и обновления настроек во всем приложении. |
| mLocalContext | Context | private final | Контекст текущего окна. |

Класс MainMenuAdapter является адаптером для элементов типа MainMenuItem, отображаемых в списке на экране выбора устройства для подключения. Описание класса приведено в таблицах 3.38 и 3.39.

Таблица 3.38 – Описание методов класса MainMenuAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| getItem | int position | MainMenuItem | Получение элемента адаптера по индексу. |
| getDropDownView | int position, View convertView,ViewGroup parent | View | Получение строки меню для DropDown отображения. |
| getView | int position, View convertView,ViewGroup parent | View | Получение строки меню для отображения. |

Таблица 3.39 – Описание полей класса MainMenuAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mInflater: | ListView | private | Контейнер для генерации расположения элементов. |
| mSubjects | ArrayList<MainMenuItem> | private | Список объектов, содержащихся в списке. |

Класс MainMenuItem является инкапсуляцией полей объекта списка меню, отображаемых на главном экране меню, таких как – int icon (иконка элемента меню) и String text (текст элемента меню).

Класс DeviceListActivity представляет собой окно выбора Bluetooth устройств для подключения. Описание класса приведено в таблицах 3.40 и 3.41.

Таблица 3.40 – Описание методов класса DeviceListActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Protected методы | | | |
| onCreate | Bundle savedInstanceState | void | Метод для создания окна. Проверяет содержимое Intenta, полученного от MainMenuActivity для выхода из приложения по нажатию на кнопку Exit в главном меню. Если нет – отображает окно для включения. |
| Private методы | | | |
| setListPairedDevices |  | void | Генерирует список устройств и проверяет, включен ли Bluetooth на мобильном устройстве. |

Таблица 3.41– Описание полей класса DeviceListActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mDeviceList: | ListView | private | UI элемент списка устройств для подключения. |
| mDeviceAdapter | DeviceAdapter | final private | Адаптер для отображения элементов BluetoothDevice в mDeviceList. |
| mBluetoothAdapter | BluetoothAdapter | final private | Bluetooth адаптер мобильного устройства. |

Класс DeviceAdapter является адаптером для элементов типа BluetoothDevice, отображаемых на экране выбора устройства для подключения. Описание класса приведено в таблицах 3.42 и 3.43.

Таблица 3.42 – Описание методов класса DeviceAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| getItem | int position | BluetoothDevice | Получение элемента адаптера по индексу. |
| getDropDownView | int position, View convertView,ViewGroup parent | View | Получение строки меню для DropDown отображения. |
| getView | int position, View convertView,ViewGroup parent | View | Получение строки меню для отображения в неразвернутом виде. |

Таблица 3.43 – Описание полей класса DeviceAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mInflater: | ListView | private | Контейнер для генерации расположения элементов. |
| mDevices | ArrayList<BluetoothDevice> | private | Список объектов, содержащихся в списке. |

Класс CameraOptionActivity наследуется от класса BaseActivity и описывает экран настроек работы процессов модуля интерактивного взаимодействия, связанных с камерой робота. Описание класса приведено в таблицах 3.44 и 3.45.

Таблица 3.44 – Описание методов класса CameraOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Protected методы | | | |
| onCreate | Bundle savedInstanceState | void | Метод создания окна класса. Инициализирует прослушиватели для нажатия кнопок Ok для диалогов удаления и добавления субъектов распознавания. Подписывается на сообщения от окна главного меню. |
| Public методы | | | |
| setImageViewWithByteArray | ImageView view, byte[] data | void | Статическая функция установки байтового представления изображения в объект ImageView. |
| deleteSubject | int numberOfSubject | void | Функция для удаления субъекта распознавания. |
| addNewSubject |  | void | Функция добавления нового субъекта распознавания. |
| onDestroy |  | void | Уничтожение окна. Отключает получение broadcast сообщений от главного окна. |
| Private методы | | | |
| setFaceRecConfig | boolean enable | boolean | Включение или отключение распознавания лиц. |
| setSubjectMonConfig | boolean enable | boolean | Включение или отключение распознавания лиц. |

Таблица 3.45 – Описание полей класса CameraOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| dialogRemoveClickListener | OnClickListener | private final | Прослушиватель события нажатия на кнопку Ок диалога перед удаление субъекта распознавания. |
| dialogAddClickListener | OnClickListener | private final | Прослушиватель события нажатия на кнопку Ок диалога перед добавлением субъекта распознавания. |
| mFacerec | boolean | private | Состояние переключателя «Enable face recognition». |
| mSubjectmon | boolean | private | Состояние переключателя «Enable subject monitoring». |
| mMessageReceiver | BroadcastReceiver | private | Получатель broadcast сообщений от окна главного меню. |
| mSpinner | SpinnerDropDown | private | Выпадающий список субъектов распознавания. |
| mBluetoothConnection | CommunicationsTask | private | Текущее Bluetooth соединение. |
| mSubjectsAdapter | FaceRecognitionAdapter | private | Адаптер для отображения элементов. Используется для прослушивания вызова метода getDropDownList для начала загрузки изображений. |

Класс SpinnerDropDown наследуется от класса Spinner, предназначен для обработки событий открытия и закрытия выпадающего списка. Описание класса приведено в таблицах 3.46 и 3.47.

Таблица 3.46 – Описание методов класса SpinnerDropDown

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |

Продолжение таблицы 3.46

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| performClick |  | boolean | Метод выполнения нажатия на список для раскрытия. |
| onWindowFocusChanged | boolean hasFocus | void | Обработка смены фокуса окна. |
| setSpinnerEventsListener | OnSpinnerEventsListener SpinnerEventsListener | void | Установка прослушивателей события списка. |
| performClosedEvent |  | void | Метод для события закрытия списка. |
| hasBeenOpened |  | boolean | Флаг открытия списка. |

Таблица 3.47 – Описание полей класса SpinnerDropDown

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mOpenStarted: | boolean | private | Флаг открытия списка. |

Класс FaceRecognitionAdapter является адаптером для элементов типа FaceRecognition, отображаемых в выпадающем списке распознаваемых субъектов на экране настроек работы камеры (см. таблицы 3.48 и 3.49).

Таблица 3.48 – Описание методов класса FaceRecognitionAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| setOwner | Spinner widget | void | Установка списка, в который будут приходить события нажатия на элементы адаптера. |
| setOnItemLongClickListener | OnItemLongClick-Listener listener | Void | Установка прослушивателя события долгого нажатия на элемент списка. |

Продолжение таблицы 3.48

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| setAdapterEventsListener | OnAdapterEvents-Listener EventsListener | void | Установка прослушивателя для событий адаптера. |
| getItem | int position | FaceRecognition | Получение элемента адаптера по индексу. |
| getDropDownView | int position, View convertView,ViewGroup parent | View | Получение строки меню для DropDown отображения. |
| getView | int position, View convertView,ViewGroup parent | View | Получение строки меню для отображения в неразвернутом виде. |
| areAllItemsEnabled |  | boolean | Проверка доступности всех элементов списка. |

Таблица 3.49 – Описание полей класса FaceRecognitionAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mInflater: | ListView | private | Контейнер для генерации расположения элементов. |
| mSubjects | ArrayList<BluetoothDevice> | private | Список объектов, содержащихся в списке. |

Класс FaceRecognition представляет собой содержимое элемента выпадающего списка субъектов распознавания модуля интерактивного взаимодействия, отображаемых на экране настроек работы камеры. FaceRecognition содержит такие поля, как String name – название, отображаемый около изображения субъекта, Bitmap bitmap – изображение формата BMP с субъектом распознавания, Integer number – уникальный номер субъекта распознавания, присвоенный интерактивным модулем.

Класс ImageManager представляет собой обработчик запросов на загрузку изображений из модуля интерактивного взаимодействия. Включает в себя функционал для обработки загрузки изображения сразу несколькими потоками и сохранению полученных изображений в кэш. Описание класса приведено в таблицах 3.50 и 3.51.

Таблица 3.50 – Описание методов класса ImageManager

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| getInstance: |  | ImageManager | Получение инстанции класса. |
| stopAllDownloads |  | void | Остановка всех загрузок. |
| stopDownload | ImageTask downloaderTask, Integer imageNumber | View | Остановка загрузки изображения. |
| startDownload | ImageView imageView,  Integer imageNumber,  boolean cacheFlag | ImageTask | Начало загрузки изображения. |
| handleImageState | ImageTask photoTask, int state | void | Обработка состояния загрузки изображения. |
| Package private методы | | | |
| recycleTask | ImageTask downloadTask | void | Освобожение ресурсов, выделенных на объект ImageTask. |

Таблица 3.51 – Описание полей класса ImageManager

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| 1 | 2 | 3 | 4 |
| sDownloadInProgress | boolean | public static | Флаг работы по загрузке изображений |
| DOWNLOAD\_FAILED | int | package private static | Константа для статуса ошибки загрузки. |
| DOWNLOAD\_STARTED | int | package private static | Константа для статуса начала загрузки. |
| DOWNLOAD\_COMPLETE | int | package private static | Константа для статуса успешного завершения загрузки. |
| IMAGE\_CACHE\_SIZE | int | private static | Размер очереди кэша для сохранения изображений после загрузки. |
| KEEP\_ALIVE\_TIME | int | private static | Время keep\_alive таймера подключения. |

Продолжение таблицы 3.51

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| TIME\_UNIT | TimeUnit | private static | Время timeout подключения. |
| POOL\_SIZE | int | private static | Размер пула потоков. |
| sInstance | ImageManager | private static | Статическая инстанция класса. |
| mImageCacheStorage | LruCache | private | Кэш для хранения изображений. |
| mDownloadQueue | BlockingQueue | private | Очередь потоков для загрузки изображений. |
| mDownloadPool | ThreadPoolExecutor | private | Пул потоков. |
| mImageTaskWorkQueue | Queue<ImageTask> | private | Хранилище потоков загрузки. |
| mHandler | Handler | private | Обработчик состояний потоков. |

Класс ImageTask управляет потоками загрузок изображений класса ImageDownloadRunnable, предоставляет хранилище для исполняемой задачи и загружаемых буферов изображений (см. таблицы 3.52 и 3.53).

Таблица 3.52 – Описание методов класса ImageTask

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Public методы | | | |
| initializeDownloaderTask | ImageManager imageManager,  mageView imageView,  Integer imageNumber,  boolean cacheFlag | void | Инициализация задачи для начала загрузки изображения . |
| setDownloadThread | Thread currentThread | View | Установка потока для обработки задачи загрузки. |
| handleDownloadState | int state | void | Обработчик состояния загрузки. |
| Package private методы | | | |

Продолжение таблицы 3.52

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| getDownloadRunnable |  | Runnable | Получение потока загрузки. |
| recycle |  | void | Освобождение ресурсов, выделенных под задачу загрузки. |
| isCacheStorageEnabled |  | boolean | Проверка поддержки кэш хранилища. |
| handleState | int state | void | Обработчик состояния потока. |

Таблица 3.53 – Описание полей класса ImageTask

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mThreadThis: | Thread | package private | Текущий поток исполнения задачи. |
| mImageBuffer | byte[] | package private | Буфер для хранения получаемого изображения. |
| mImageManager | ImageManager | private | Объект класса ImageManager для сообщения об окончании выполнения задачи. |
| mDownloadRunnable | Runnable | private | Интерфейс загрузки для потока исполнения задачи. |
| mImageWeakRef | WeakReference | private | Ссылка на объект ImageView для установки изображения и проверки доступности элемента в текущий момент жизненного цикла приложения в целях избегания утечек памяти. |
| mCacheEnabled | boolean | private | Флаг включения записи в кэш. |
| mImageNumber | int | private | Номер изображения для загрузки. |

Класс ImageDownloadRunnable наследуется от класса Runnable и предоставляет интерфейс для загрузки изображений субъектов, хранящихся на микроконтроллере «Мультизадачного робота», по существующему Bluetooth соединению. Описание класса приведено в таблицах 3.54 и 3.55.

Таблица 3.54 – Описание методов класса ImageDownloadRunnable

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| run |  | void | Выполняет загрузку изображения по Bluetooth и сохраняет полученный буфер через объект mImageTask. |
| Package private методы | | | |
| ImageDownloadRunnable | TaskRunnableDownload-Methods imageTask |  | Конструктор класса. Инициализирует объект интерфейса Runnable для выполнения метода run. |

Таблица 3.55 – Описание полей класса ImageDownloadRunnable

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| STATE\_FAILED: | int | package private static | Константа для статуса ошибки выполнения загрузки. |
| STATE\_STARTED | int | package private static | Константа для статуса начала выполнения. |
| STATE\_COMPLETED | int | package private static | Константа для статуса завершения загрузки. |
| READ\_SIZE | int | private static | Размер порции данных для считывания. |
| mImageTask | TaskRunnableDownloadMethods | package private static | Методы для установки буфера, записи данных, считывания данных изображения из Bluetooth сокета. |

BaseActivity – базовый класс для всех подразделов главного меню мобильного приложения. Он обеспечивает анимацию плавного перехода между экранами. Класс унаследован от AppCompatActivity. Описание класса приведено в таблицах 3.56 и 3.57.

Таблица 3.56 – Описание методов класса BaseActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Public методы | | | |
| finish |  | void | Переопределение метода finish() класса AppCompatActivity. Вызывает метод установки плавной анимации закрытия окна. |
| startActivity | Intent intent | void | Переопределение метода startActivity () класса AppCompatActivity. Вызывает метод установки плавной анимации открытия окна. |
| Protected методы | | | |
| overridePendingTransitionEnter |  | void | Установка плавной анимации перехода при открытии нового окна. |
| overridePendingTransitionExit |  | void | Установка плавной анимации перехода при закрытии окна. |

Класс FindMyRobo унаследован от класса BaseActivity и описывает окно включения поиска устройства (см. таблицы 3.57 и 3.58).

Таблица 3.57 – Описание методов класса FindMyRobo

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| 1 | 2 | 3 | 4 |
| Protected методы | | | |

Продолжение таблицы 3.57

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| setStartSearchParameter | boolean isOn | void | Передача команды поиска робота по Bluetooth соединению. |
| setLogoVisibility | boolean isVisible | Void | Установка видимости логотипа программы на фоне виджета загрузки. |

Таблица 3.58 – Описание полей классов

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mLogo: | ImageView | package private | Логотип приложения |

Класс OperationModeOptionActivity наследуется от класса BaseActivity. Представляет собой окно настроек режима функционирования робота. Описание класса приведено в таблицах 3.59 и 3.60.

Таблица 3.59 – Описание методов класса OperationModeOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Private методы | | | |
| setOperationModeConfig | int mode | void | Установка текущего режима работы робота. Механизм работы режимов описан далее в пункте «Руководство пользователя». |

Таблица 3.60 – Описание полей класса OperationModeOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| 1 | 2 | 3 | 4 |
| radioButtonClickListener: | OnClickListener | package private | Прослушиватель нажатия на радиокнопки. |

Продолжение таблицы 3.60

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| mCurrentMode | Int | private | Текущий установленных режим работы. |
| mMessageReceiver | BroadcastReceiver | private | Получатель broadcast сообщений от окна главного меню для обновления настроек и проверки подключения. |

Класс SoundMicOptionActivity предоставляет доступ к настройке работы звука и микрофона. Унаследован от класса BaseActivity. Описание класса приведено в таблицах 3.61 и 3.62.

Таблица 3.61 – Описание методов класса SoundMicOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Private методы | | | |
| setSoundConfig | boolean enable | boolean | Установка включения или отключение воспроизведения звукового сопровождения. |
| setMicConfig | boolean enable | boolean | Установка включения или отключение работы микрофонов. |

Таблица 3.62 – Описание полей класса SoundMicOptionActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| mSoundEnabled: | boolean | private | Состояние переключателя «Enable sound». |
| mMicEnabled | boolean | private | Состояние переключателя «Enable microphone». |
| mMessageReceiver | BroadcastReceiver | private | Получатель broadcast сообщений от окна главного меню для обновления настроек и проверки подключения. |

Класс WiFiSettingsActivity представляет собой окно для настройки подключения Wi-Fi соединения робота. Унаследован от класса BaseActivity. Описание класса приведено в таблицах 3.63 и 3.64.

Таблица 3.63 – Описание методов класса WiFiSettingsActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Принимаемые значения | Возвращаемые значения | Описание |
| Protected методы | | | |
| onCreate | Bundle savedInstanceState | void | Создает окно. Устанавливает прослушиватель для кнопки «START CONFIGURATION». Сбрасывает флаг sWifiSettingInProgress в состояние false. |
| Private методы | | | |
| writeOutput | String text | void | Запись текста в поле mMessageTextView. |
| clearOutput |  | void | Очистка тестового поля mMessageTextView . |
| waitReceivedMessage | InputStream mInputStream, long timeout | void | Метод ожидания сообщений от модуля интерактивного взаимодействия. Используется для синхронизации между командами отправки SSID и PSK. |

Таблица 3.64 – Описание полей класса WiFiSettingsActivity

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Доступ | Описание |
| 1 | 2 | 3 | 4 |
| sWifiSettingInProgress: | boolean | public static | Флаг выполнения установки настроек Wi-Fi. |
| mInitWifiRunnable | Runnable | package private | Поток выполнения настройки Wi-Fi. |
| mSsidTextView | TextView | private | Поле ввода SSID Wi-Fi сети. |
| mPskTextView | TextView | private | Поле ввода PSK Wi-Fi сети. |

Продолжение таблицы 3.64

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| mMessageTextView | TextView | private | Текстовое поле сообщений от модуля. |
| mReadBlockSize | byte | private | Размер блока чтения данных из сокета. |
| readBufferPosition | Int | private | Текущая позиция считывания данных в буфер. |

**3.5 Результаты функционального проектирования**

В рамках функционального тестирования была разработана архитектура набора команд движения для USB-Serial соединения и команд настроек для Bluetooth соединения, структура классов и последовательности взаимодействия мобильного приложения и прошивки Arduino Uno, структура модулей и последовательность взаимодействия процессов модуля интерактивного взаимодействия.

Разработанные архитектуры набора команд позволяют упростить и унифицировать передачу данных между элементами системы. Текущая структура позволяет в дальнейшем при доработке проекта добавлять новые команды, не усложняя схемы взаимодействия между модулями.

Структура классов прошивки Arduino Uno была разработана в соответствие с принципом расширяемости кода. Текущую реализацию можно применять как готовую библиотеку для взаимодействия с Arduino Uno по протоколу RS-232, так и для разработки новых программ на основе готовых функциональных блоков.

Структура модуля интерактивного взаимодействия обеспечивает разделение функциональности между процессами для параллельного выполнения задач, что положительно сказывается на скорости обработки данных.

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

В данном разделе будут рассмотрены наиболее важные алгоритмы, реализованные в модуле интерактивного взаимодействия, мобильном приложении и прошивке Arduino Uno.

**4.1 Алгоритмы модуля интерактивного взаимодействия**

**4.1.1 Алгоритм предобработки изображений**

Модуль FacePreProcessing обеспечивает предобработку изображений, содержащих лица. Алгоритм функции def PreProcessing\_Origin(im, w, h)представлен на чертеже ГУИР.400201.040 ПД.1.

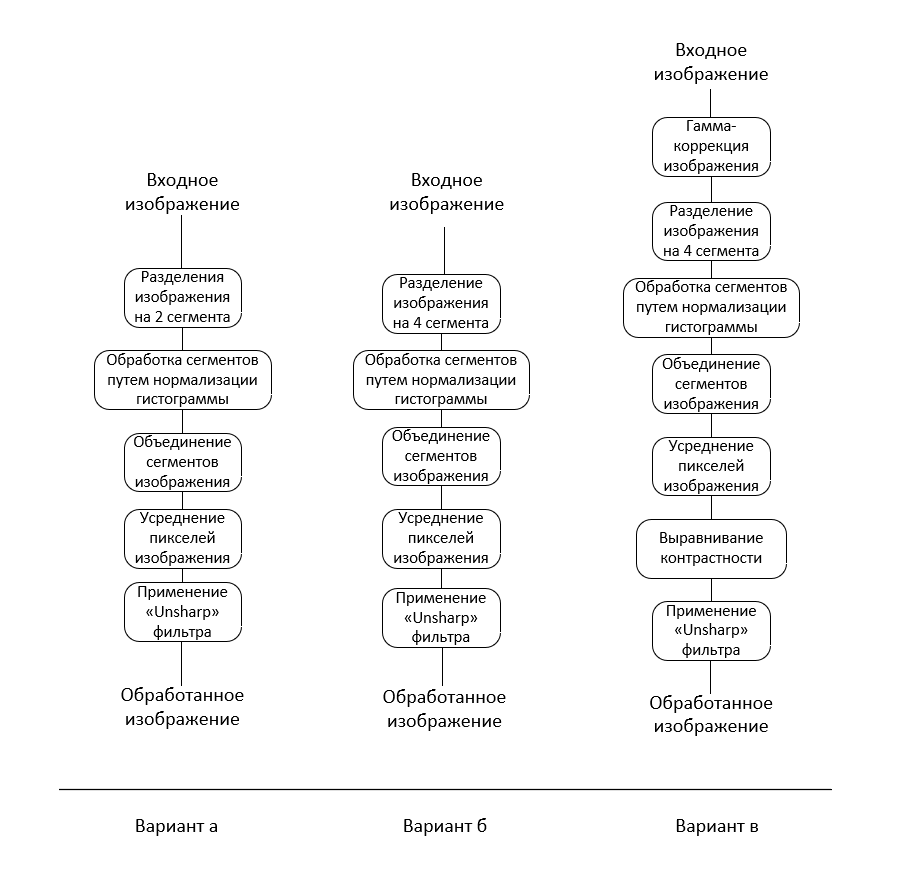


Рисунок 4.1 – Алгоритмы нормализации освещенности. Вариант а - базовый алгоритм, вариант б – модификация варианта а для разных углов освещенности, вариант в – применяемый в работе модифицированный алгоритм.

Данный алгоритм позволяет повысить уровень распознавания пользователя. На этапе функционального проектирования было проанализировано несколько подходов к обработке изображения для выравнивания освещенности. Этапы обработки изображений для разных вариантов алгоритма представлены на рисунке 4.1. В качестве базы для реализованного в модуле интерактивного взаимодействия варианта предобработки изображения был использован подход, описанный в работе Illumination Normalization Preprocessing for face recognition [24], схематично изображенный на вариантах а и б рисунка 4.1.

В целях улучшения результатов предобработки для каждого изображения, поступившего в функцию def PreProcessing\_Origin(im, w, h), производится гамма-коррекция. В зависимости от общего уровня освещенности изображения коррекция производится в двух направлениях: уменьшение яркость средних тонов, если уровень освещенности более 60 процентов, и повышения, если освещенность ниже 30 процентов. Изменение гаммы, в отличие от отдельного изменения яркости всего изображения в целом, позволяет выровнять общий уровень яркости изображения без потери контуров лица в сильно затемнённых или засвеченных областях.

Основу алгоритма нормализации освещенности составляет процедура нормализации гистограммы изображения. Для того, чтобы выравнивание освещенности между участками лица происходило более эффективно, изображение в реализованном алгоритме разделяется на 4 сегмента (см. рисунок 4.2). По сравнению с вариантом а алгоритма нормализации, представленном на рисунке 4.1, деление изображения на 4 части более рационально для применения в режиме реального времени, так как угол освещения лица в конкретный момент времени невозможно предсказать и затемнение лица может быть неравномерным относительно вертикали или горизонтали.

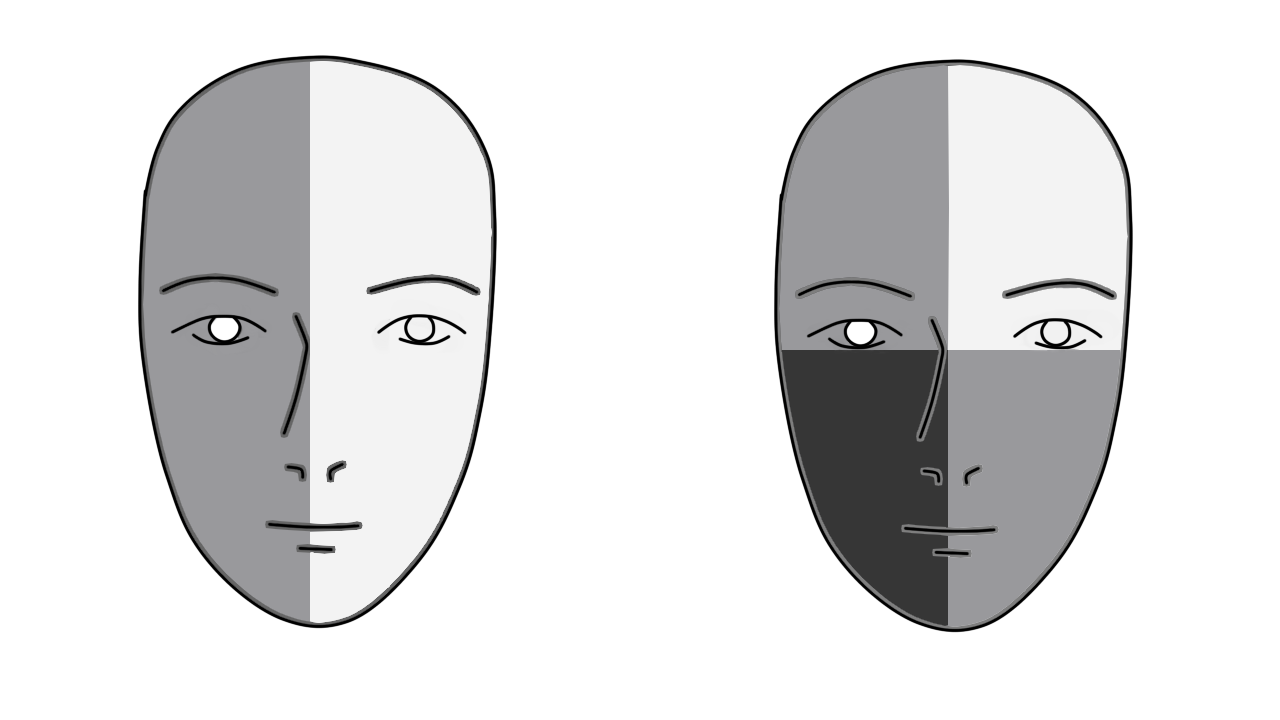


Рисунок 4.2 – Деление изображения на сегменты, относительно угла падения света на лицо



Рисунок 4.3 – Изображения без применения алгоритма предобработки. Нумерация слева направо, начиная с единицы.

Гистограмма изображения номер два, изображенного на рисунке 4.3, представлена на рисунке 4.4. Как видно на рисунке, распределение светлых и темных участком неравномерное, также есть заметный перекос в сторону темных оттенков.

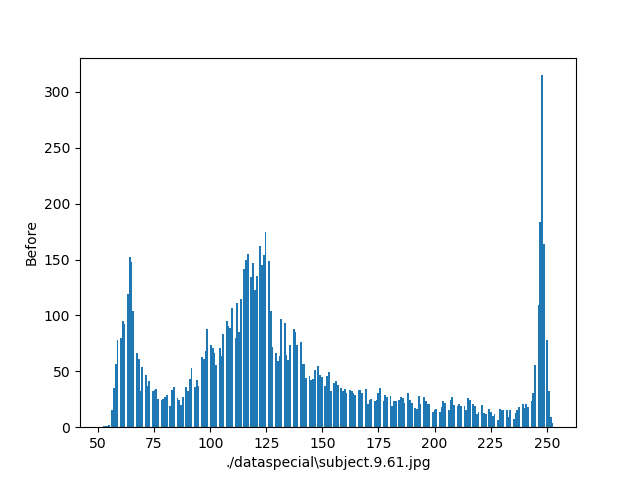


Рисунок 4.4 – Гистограмма изображения лица с камеры «Мультизадачного робота»

Для уменьшения влияния высокочастотных шумов, полученных как в результате нормализации гистограммы, так и склеивания сегментов изображения, применяется усреднение значений пикселей изображения. Величина значения устанавливается равной среднему арифметическому значения выбранного пикселя и 8 соседних.

Дальнейшим этапом алгоритма является нормализация контрастности. В данном варианте предобработки изображений используется мин-макс нормализация контрастности. Нижнее и верхнее значения входного изображения создаются так, чтобы охватить весь динамический диапазон, в нашем случае – значения от 0 до 255, так как обрабатывается серое изображение. Другими словами, производится нормирование значений пикселей относительно диапазона значений серого изображения по формуле

где ‒ значение пикселя выходного изображения;

‒ исходное значение пикселя входного изображения;

‒ минимальное значение среди всех пикселей изображения;

‒ максимальное значение среди всех пикселей изображение.

Окончательным этапом является применение фильтра «Unsharp», также называемого нерезкое маскирование, для повышения резкости изображения.

После выполнения предобработки гистограмма изображения выглядит, как показано на рисунке 4.5. Как видно из изображения, уменьшилась количественная разница между темными и светлыми областями изображения.

Пример изображений, прошедших предобработку, представлен на рисунке 4.6.

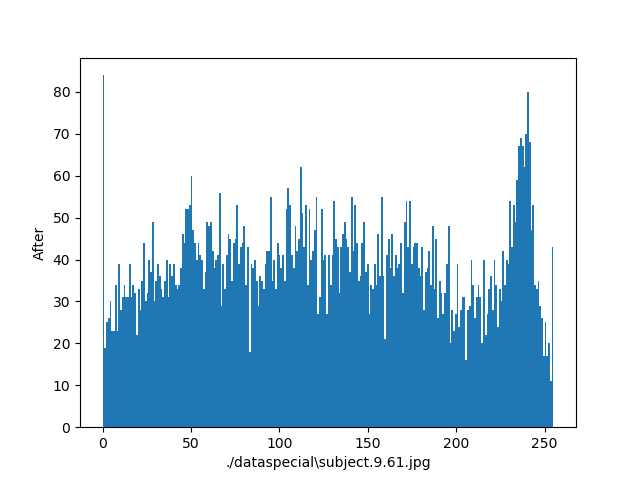


Рисунок 4.5 – Гистограмма изображения лица с камеры «Мультизадачного робота»

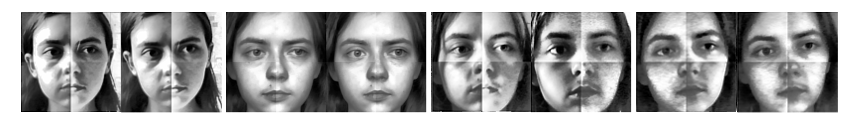


Рисунок 4.6 – Изображения, получаемые с применением предобработки

Таким образом, в результате применения алгоритма предобработки изображений существенно повышается уровень распознавания пользователей, что было подтверждено результатами тестирования модуля FaceRecognitionProcessing на изображениях с рисунков 4.3 и 4.6. Результаты распознавания без предобработки и с пременением алгоритма представлены на рисунках 4.7 и 4.8 соответсвенно.

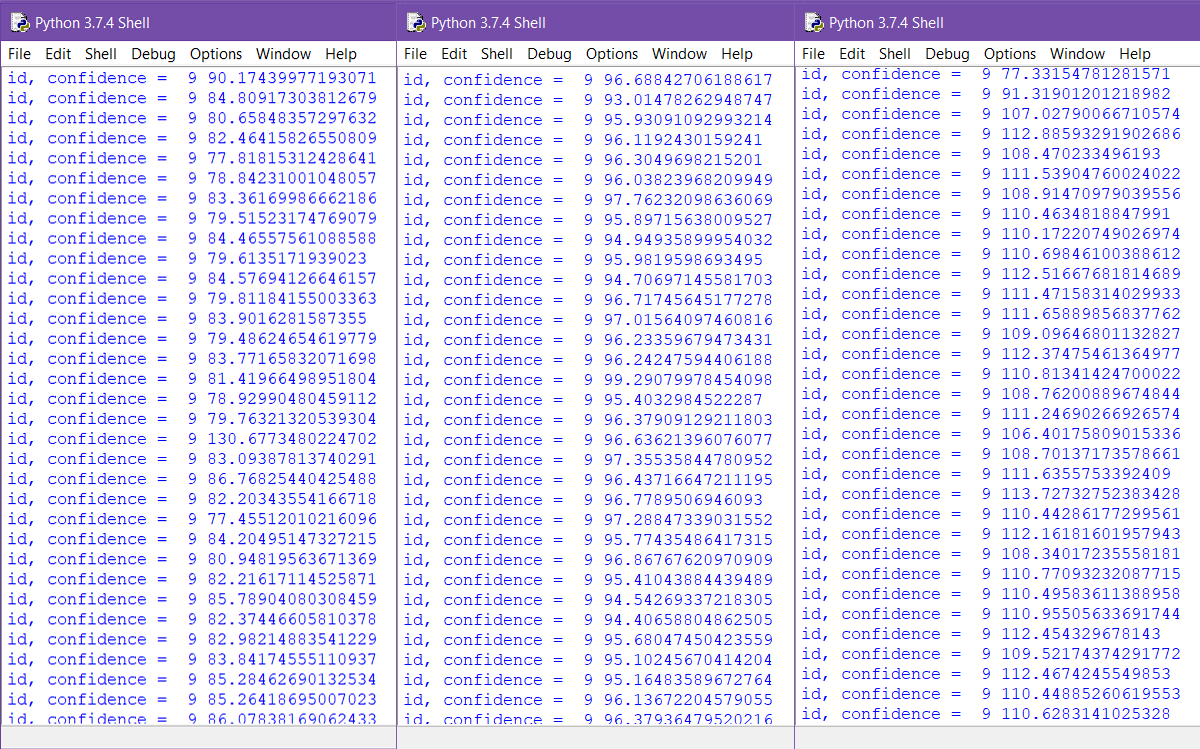


Рисунок 4.7 – Распознавание лиц без предобработки. Слева направо: равномерное освещение, боковое освещение, сильно затемненное помещение.

На изображении 4.7 отображены результаты выполнения функции FaceRecognitionProcessing.recognizer.predict(image), определяющей id потенциально распознанного пользователя и степень уверенности распознавателя. Чем ниже показатель confidence, тем больше уровень совпадения между гистограммами субъекта с индексом id и изображения обнаруженного лица. Пороговое значение уверенности распознавателя для модуля FaceRecognitionProcessing равняется 95. Такое величина была выбрана исходя из показателей распознавания с применением предобработки в затемненном помещении и распределению значений показателя confidence для алгоритма классификации LBPH [18].

Таким образом, при смене освещения на боковое (см. рисунок 4.7), классификатор определяет изображение лица как распознанное менее, чем в половине случаев, а при обработке в затемненном помещении уверенность распознавания переходит пороговое значение.

В отличие от результатов распознавания без предобработки, на изображении 4.8 средний показатель confidence не превышает 50 для равномерного и бокового освещения, что является хорошим уровнем распознавания для локальных бинарных шаблонов гистограмм [18]. В случае затемненного помещения результат также в среднем не превышает поговое значение confidence.



Рисунок 4.8 – Распознавание лиц с применением предобработки. Слева направо: равномерное освещение, боковое освещение, сильно затемненное помещение.

**4.1.2 Алгоритм добавления субъекта распознавания**

Процесс InteractionProcessing включает в себя несколько файлов на языке Python, описывающие алгоритмы распознавания лица и перемещения в пространстве. Функция def Create\_Dataset(sound\_enable) находится в модуле FaceDatasetCreate, предназначенном для добавления и удаления субъектов распознавания из датасета. Алгоритм данной функции представлен на чертеже ГУИР.400201.040 ПД.2.

Рассмотрим алгоритм функции def Create\_Dataset(sound\_enable) по шагам.

Шаг 1. Начало алгоритма. Получаем в качестве параметра функции переменную sound\_enable.

Шаг 2. Объявляем переменную cam и присваиваем ей значение результата выполнения функции cv2.VideoCapture(0), создающей объект для обращения к камере устройства.

Шаг 3. Вызываем функцию cam.set с параметрами 3 и 1280 для установки ширины изображения камеры.

Шаг 4. Вызываем функцию cam.set с параметрами 4 и 720 для установки высоты изображения камеры.

Шаг 5. Загружаем каскады Хаара для детектирования лиц при помощи функции cv2.CascadeClassifier ('haarcascade\_frontalface\_default.xml').

Шаг 6. Сохраняем в переменную id результат выполнения вложенной функции получения индекса нового пользователя Get\_Next\_Id().

Шаг 7. Если звуковое сопровождения не включено, то переходим к шагу 9.

Шаг 8. Вызываем функцию os.system с параметром «mpg321 -a plughw:1,0 ./Sounds/blip.mp3» для воспроизведения звука начала съемки изображений нового субъекта.

Шаг 9. Объявляем переменную count и присваиваем ей значение 0.

Шаг 10. Объявляем цикл while с выходным условием count < 100. Если условие не выполняется, то переходим к шагу 25.

Шаг 11. Вызываем функцию cam.read() для создания снимка с камеры и сохраняем результат выполнения в переменные ret и img

Шаг 12. Вызываем функцию cv2.flip с параметрами img и -1 для отражения изображения по вертикали. Результат выполнения функции записываем в переменную img.

Шаг 13. Объявляем переменную gray и присваиваем ей значение результата выполнения функции cv2.cvtColor с параметрами img и cv2.COLOR\_BGR2GRAY для перевода изображения img в серое.

Шаг 14. Объявляем переменную faces и присваиваем ей значение результата выполнения функции face\_detector.detectMultiScale() с параметрами gray, 1.3 и 5 для обнаружения координат лиц на изображении при помощи каскадов Хаара.

Шаг 15. Объявляем цикл for по координатам лиц (x,y,w,h) в списке faces. Если список завершился, то переходим к шагу 22.

Шаг 16. Выполняем функцию изменения размера изображения cv2.resize с параметрами gray, [y:y+h,x:x+w],(350,350) и сохраняем результат выполнения в локальную переменную resized\_image

Шаг 17. Выполняем функцию предобработки изображения лица PreProcessing\_Origin модуля FacePreProcessing с параметрами resized\_image, 350, 350 и сохраняем результат выполнения в локальную переменную resized\_image.

Шаг 18. Объявляем переменную path и присваиваем ей значение результата конкатенации строк «data/subject.», str(id), «.», str(count)и «.bmp».

Шаг 19. Выполняем функцию записи изображения на sd-карту cv2.imwrite с параметрами path и resized\_image.

Шаг 20. Если величина переменной count равна 100 – переходим к шагу 22.

Шаг 21. Инкрементируем величину переменной count, переходим к шагу 15.

Шаг 22. Если звуковое сопровождения не включено, то переходим к шагу 24.

Шаг 23. Вызываем функцию os.system с параметром «mpg321 -a plughw:1,0 ./Sounds/blip.mp3» для воспроизведения звука начала съемки изображений нового субъекта.

Шаг 24. Переходим к шагу 10.

Шаг 25. Вызываем функцию cam.release().

Шаг 26. Завершение алгоритма.

**4.1.3 Алгоритм поиска**

Функция def Search() является частью модуля InteractionProcessing и выполняет обнаружение лиц на изображении для последующего мониторинга.

Шаг 1. Начало алгоритма. Объявляем переменные count и detect со значение 0 и переменную result со значением False.

Шаг 2. Если значение переменной face\_recognition, содержащей настройку разрешения распознавания лица, равно 0, то переходим к шагу 16.

Шаг 3. Объявляем цикл while с выходным условием count <20. Если условие не выполняется, то переходим к шагу 17.

Шаг 4. Выполняем функцию поиска лица на изображении FaceRecognitionProcessing.detect.Find\_Face(), сохраняем результаты ее выполнения в локальные переменные found и image.

Шаг 5. Если значение переменной found не равно True, то переходим к шагу 16.

Шаг 6. Выполняем функцию распознавания лица FaceRecognitionProcessing.recognizer.predict, в качестве параметра передаем переменную image, сохраняем результаты ее выполнения в локальные переменные id и confidence.

Шаг 7. Если значение переменной confidence больше или равно 95, то переходим к шагу 16.

Шаг 8. Присваиваем переменной result значение True.

Шаг 9. Если значение переменной detected меньше или равно 10, то переходим к шагу 15.

Шаг 10. Если значение переменной movement\_enable, содержащей настройку разрешения перемещения, не равно 1 и значение переменной subject\_monitoring, содержащей настройку разрешения мониторинга, не равно 1, то переходим к шагу 14.

Шаг 11. Если значение переменной sound\_enable, содержащей настройку разрешения звукового сопровождения, не равно 1, то переходим к шагу 13.

Шаг 12. Выполняем функцию os.system с параметром «mpg321 -a plughw:1,0 ./Sounds/blip.mp3» для звукового сопровождения мониторинга пользователя.

Шаг 13. Выполняем функцию мониторинга пользователя MovementProcessing.Monitor\_user с параметрами ix, iy, iw, ih из модуля FaceRecognitionProcessing.detect.

Шаг 14. Присваиваем переменным count и detected значение 0.

Шаг 15. Инкрементируем величину переменной detected. Переходим к шагу 3.

Шаг 16. Инкрементируем величину переменной count. Переходим к шагу 3.

Шаг 17. Возвращаем в результате выполнения функции значение переменной result.

Шаг 18. Завершение алгоритма.

**4.1.4 Алгоритм обработки команд процессом BluetoothServer**

Процесс BluetoothServer, выполняемый на Raspberry Pi, базируется на одноименном модуле на языке Python. Так как BluetoothServer представляет собой сервис в операционной системе микрокомпьютера, то основной алгоритм выполняется в бесконечном цикле. Поэтому будем рассматривать только одиночную итерацию выполнения обработки.

Шаг 1. Начало алгоритма. Считываем данные из буфера приемника клиентского сокета client\_sock в размере 1024 байта в переменную data.

Шаг 2. Если длина массива байт из переменной data равно нулю, то переходим на шаг 1.

Шаг 3. Если длина массива байт из переменной data не равна 1024 байта, то переходим к шагу 5.

Шаг 4. Устанавливаем в переменную isBufferFull значение True, переходим к шагу 6.

Шаг 5. Устанавливаем в переменнуюisBufferFull значение False.

Шаг 6. Объединяем глобальный буфер partial, содержащий последнюю недочитанную команду, с массивом data, преобразуем полученный массив в список по разделителю «$$» и записываем в локальную переменную commands.

Шаг 7. Объявляем локальную переменную number и присваиваем ей значение длины списка commands.

Шаг 8. Если переменная isBufferFull не равна True, то переходим к шагу 11.

Шаг 9. Записываем в буфер partial последний элемент списка commands.

Шаг 10. Декрементируем значение переменной number, переходим к шагу 12.

Шаг 11. Очищаем буфер partial.

Шаг 12. Объявляем цикл for по переменной i с выходным условие i < number, если выходное условие не выполняется, то переходим к шагу 18.

Шаг 13. Создаем локальную переменную command и присваиваем ей значение i-го элемента списка commands.

Шаг 14. Объявляем переменную parts и записываем в нее результат выполнения функции split с параметром «/» для переменной command.

Шаг 15. Создаем переменную count и присваиваем ей длину списка из переменной parts.

Шаг 16. Выполняем функцию parseCommand модуля BluetoothServer с параметрами command, parts и count.

Шаг 17. Переходим на шаг 12.

Шаг 18. Завершение алгоритма.

**4.2 Алгоритмы мобильного приложения**

Наиболее сложным алгоритмом мобильного приложения является загрузка субъектов распознавания, так как изображения загружаются с микрокомпьютера в реальном времени и по мере загрузки отображаются в выпадающем списке пользовательского интерфейса. Загрузка каждого изображения выделена в отдельный поток, чтобы не замедлять процесс работы с приложением и не блокировать поток графического интерфейса до завершения загрузки изображения.

**4.2.1 Алгоритм старта загрузки изображений**

static ImageTask startDownload(ImageView imageView, Integer imageNumber, boolean cacheFlag) является статической функцией класса ImageManager, предназначена для начала загрузки и установки загруженных изображений субъектов распознавания в выпадающий список.

Шаг 1. Начало алгоритма. Получаем в качестве параметра функции переменную imageView типа ImageView, imageNumber типа Integer, cacheFlag типа boolean.

Шаг 2. Объявляем переменную downloadTask типа ImageTask и присваиваем ей значение результата выполнения функции получения задачи из пула выполнения poll() инстанции класса ImageManager.

Шаг 3. Если значение переменной downloadTask не равно null, то переходим к шагу 5.

Шаг 4. Выделяем память под инстанцию класса ImageTask. В качестве параметра конструктора передаем результат выполнения функции получения текущего Bluetooth соединения getCurrentBluetoothConnection() класса ClientApplication. Сохраняем полученный объект в переменную downloadTask.

Шаг 5. Вызываем функцию инициализации задачи initializaDownloadTask для переменной downloadTask и передаем в качестве параметров инстанцию класса ImageManager, переменные imageView, imageNumber, cacheFlag.

Шаг 6. Устанавливаем буфер загрузки для задачи downloadTask при помощи метода setByteBuffer с параметрами, полученными в результате выполнения функции get очереди mImageCache инстанции класса ImageManager для номера изображения imageNumber.

Шаг 7. Если результат выполнения функции getByteBuffer переменной downloadTask не равен null, то переходим к шагу 9.

Шаг 8. Вызываем функцию execute пула потоков загрузки mDownloadThreadPool для потока задачи downloadTask, переходим к шагу 10.

Шаг 9. Вызываем функцию handleState класса ImageManager с параметрами downloadTask и DOWNLOAD\_COMPLETE для обработки завершения выполнения загрузки изображения.

Шаг 10. Возвращаем объект переменной downloadTask.

Шаг 11. Завершение алгоритма.

**4.3 Алгоритмы прошивки Arduino Uno**

В качестве наиболее интересных алгоритмов прошивки Arduino Uno были выбраны реализации функций обработки ситуаций столкновения и обработки команд от Raspberry, так как их выполнение представляет собой ключевую задачу микроконтроллера, обеспечивая движение робота в пространстве.

**4.3.1 Алгоритм обработки ситуаций столкновения**

void run() является функцией класса MovementController, предназначена для обработки и предотвращения ситуаций столкновения робота с объектами на пути перемещения. Алгоритм данной функции представлен на чертеже ГУИР.400201.040 ПД.3.

Подробнее рассмотрим алгоритм функции void run() по шагам.

Шаг 1. Начало алгоритма. Если результат выполнения статической функции isRunning() класса MotorMovementCommand не равен true, то переходим к шагу 34.

Шаг 2. Объявляем переменную lightLevel типа int и присваиваем ей значение результата выполнения функции считывания показаний read для переменной датчика освещенности PSSensor класса MovementController.

Шаг 3. Если величина lightLevel меньше 150, то переходим к шагу 5.

Шаг 4. Выполняем функцию установки скорости моторов setSpeed класса MotorMovementCommand с параметром, равным результату выполнения функции normalizeData переменной PSSensor, переходим к шагу 6.

Шаг 5. Выполняем функцию установки скорости моторов setDefaultSpeed класса MotorMovementCommand.

Шаг 6. Объявляем переменную barrierInForward типа boolean и присваиваем ей значение результата выполнения функции проверки минимальной дистанции для движения checkUltraSoundSensor().

Шаг 7. Объявляем переменную forbiddenDirection типа int и присваиваем ей значение результата выполнения функции проверки разрешенных направлений движения checkInfraSensors с параметром barrierInForward.

Шаг 8. Если величина forbiddenDirection равна 0, то переходим к шагу 34.

Шаг 9. Если величина forbiddenDirection равна константной величине FORBID\_FORWARD, то переходим к шагу 12.

Шаг 10. Если величина forbiddenDirection равна константной величине FORBID\_LEFT, то переходим к шагу 12.

Шаг 11. Если величина forbiddenDirection равна константной величине FORBID\_FORWARD\_LEFT, то переходим к шагу 12, иначе – к шагу 16.

Шаг 12. Если результат выполнения статической функции getCurrentDirection() класса MotorMovementCommand не меньше константной величины MOVE\_BACK, то переходим к шагу 34.

Шаг 13. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_BACK и 500. Вызываем функцию выполнения команды.

Шаг 14. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_RIGHT и 200. Вызываем функцию выполнения команды.

Шаг 15. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_FORWARD. Вызываем функцию выполнения команды. Переходим к шагу 34.

Шаг 16. Если величина forbiddenDirection равна константной величине FORBID\_RIGHT, то переходим к шагу 18.

Шаг 17. Если величина forbiddenDirection равна константной величине FORBID\_FORWARD\_RIGHT, то переходим к шагу 18, иначе – к шагу 22.

Шаг 18. Если результат выполнения статической функции getCurrentDirection() класса MotorMovementCommand не меньше константной величины MOVE\_BACK, то переходим к шагу 34.

Шаг 19. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_BACK и 500. Вызываем функцию выполнения команды.

Шаг 20. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_RIGHT и 200. Вызываем функцию выполнения команды.

Шаг 21. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_FORWARD. Вызываем функцию выполнения команды. Переходим к шагу 34.

Шаг 22. Если величина forbiddenDirection равна константной величине FORBID\_LEFT\_RIGHT, то переходим к шагу 24.

Шаг 23. Если величина forbiddenDirection равна константной величине FORBID\_FORWARD\_LEFT\_RIGHT, то переходим к шагу 24, иначе – к шагу 28.

Шаг 24. Если результат выполнения статической функции getCurrentDirection() класса MotorMovementCommand не меньше константной величины MOVE\_BACK, то переходим к шагу 34.

Шаг 25. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_BACK и 500. Вызываем функцию выполнения команды.

Шаг 26. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_LEFT и 200. Вызываем функцию выполнения команды.

Шаг 27. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_FORWARD. Вызываем функцию выполнения команды. Переходим к шагу 34.

Шаг 28. Если величина forbiddenDirection не равна константной величине FORBID\_ALL, то переходим к шагу 30.

Шаг 29. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_STOP. Вызываем функцию выполнения команды. Переходим к шагу 34.

Шаг 30. Если результат выполнения статической функции getCurrentDirection() класса MotorMovementCommand не равен константной величины MOVE\_BACK, то переходим к шагу 34.

Шаг 31. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_FORWARD и 500. Вызываем функцию выполнения команды.

Шаг 32. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_LEFT и 500. Вызываем функцию выполнения команды.

Шаг 33. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_BACK. Вызываем функцию выполнения команды.

Шаг 34. Завершение алгоритма.

**4.3.2 Алгоритм обработки команд класса SerialIntercommunicator**

void run() является функцией класса SerialIntercommunicator, предназначена для обработки команд , полученных через последовательный порт.

Шаг 1. Начало алгоритма. Если результат выполнения функции read() класса SerialIntercommunicator не равен true, то переходим к шагу 20.

Шаг 2. Если значение первого элемента строки command класса SerialIntercommunicator не равна «v», то переходим к шагу 4.

Шаг 3. Вызываем функцию инициализации переменной servoCommand с параметрами 0 и результатом преобразования строки dist из класса SerialIntercommunicator к типу int. Вызываем функцию выполнения команды.

Шаг 4. Если значение первого элемента строки command из класса SerialIntercommunicator не равна «h», то переходим к шагу 6.

Шаг 5. Вызываем функцию инициализации переменной servoCommand с параметрами 1 и результатом преобразования строки dist из класса SerialIntercommunicator к типу int. Вызываем функцию выполнения команды.

Шаг 6. Если значение первого элемента строки command класса SerialIntercommunicator не равна «s», то переходим к шагу 8.

Шаг 7. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_STOP. Вызываем функцию выполнения команды.

Шаг 8. Если значение первого элемента строки command класса SerialIntercommunicator не равна «f», то переходим к шагу 10.

Шаг 9. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_FORWARD. Вызываем функцию выполнения команды.

Шаг 10. Если значение первого элемента строки command класса SerialIntercommunicator не равна «b», то переходим к шагу 12.

Шаг 11. Вызываем функцию инициализации переменной motorCommand с параметром MOVE\_BACK. Вызываем функцию выполнения команды.

Шаг 12. Если значение первого элемента строки command из класса SerialIntercommunicator не равна «l», то переходим к шагу 14.

Шаг 13. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_LEFT и результатом преобразования строки dist из класса SerialIntercommunicator к типу int. Вызываем функцию выполнения команды.

Шаг 14. Если значение первого элемента строки command из класса SerialIntercommunicator не равна «r», то переходим к шагу 16.

Шаг 15. Вызываем функцию инициализации переменной motorCommand с параметрами MOVE\_RIGHT и результатом преобразования строки dist из класса SerialIntercommunicator к типу int. Вызываем функцию выполнения команды.

Шаг 16. Если значение первого элемента строки command класса SerialIntercommunicator не равна «o», то переходим к шагу 18.

Шаг 17. Вызываем функцию выполнения команды переменной locationCommand. Выполняем функцию записи в последовательный порт write с параметром, полученным в результате вызова метода преобразования результата команды в строку getResultString для переменной locationCommand.

Шаг 18. Если значение первого элемента строки command класса SerialIntercommunicator не равна «t», то переходим к шагу 20.

Шаг 19. Вызываем функцию записи в последовательный порт write из класса SerialInterCommunicator с параметром, полученным в результате вызова метода выполнения вычислений уровня заряда run() переменной batteryController

Шаг 20. Завершение алгоритма.

**4.4 Результаты разработки программных модулей**

В рамках разработки программных моделей представлены и описаны наиболее интересные разработанные алгоритмы программного обеспечения. Алгоритм предобработки и алгоритм обработки ситуаций столкновения являются одними из наиболее важных частей модулей. Их выполнение определяет скорость работы системы и успешное выполнение поставленных задач.

Применение алгоритма предобработки позволило повысить эффективность выполнения распознавания пользователей, позволяя выполнять корректный мониторинг субъектов в помещениях с разным уровнем освещенности.

Алгоритм обработки ситуаций столкновения обеспечивает безопасное автоматизированное перемещение робота в пространстве. В зависимости от расстояния до предмета производиться объезд препятствий или смена направления движения.

В свою очередь алгоритм поиска демонстрирует логику взаимодействия робота с пользователем, включающую в себя обнаружение лица в видеопотоке, распознавание и мониторинг пользователей.

1. **ПРОГРАММА И МЕТОДЫ ИСПЫТАНИЙ**

В данном диплом проекте основное направление методики испытаний – проверка корректности и исправности работы аппаратной платформы «Мультизадачного робота». Это подразумевает под собой наличие тестов, определяющие исправность работы в целом и позволяющие детектировать конкретные проблемы.

**5.1 Исправная работа программной части мультизадачного робота**

Предусловие: Установлено устойчивое соединение робота с сетью Интернет посредством Wi-Fi. Мобильное устройство подключено по Bluetooth к мультизадачному роботу. Настройки робота по умолчанию: режим функционирование standalone (перемещение разрешено и запись звука с ближнего микрофона), звуковое сопровождение включено, запись звука с микрофонов включена, распознавание лиц включено, мониторинг пользователя включен.

Цель проверки: Подтверждение правильного исполнения программного обеспечения в случае отсутствия технических неполадок.

Результаты тестирования исправного выполнения программной части «Мультизадачного робота» отображены в таблицах 5.1 и 5.2.

**5.1.1 Модуль интерактивного взаимодействия**

Таблица 5.1 – Тестирование работы модуля интерактивного взаимодействия

|  |  |  |  |
| --- | --- | --- | --- |
| Действия | Ожидание | Результат |  |
| 1 | 2 | 3 |  |
| Настройки по умолчанию | | | |
| Перед камерой робота не обнаружено лицо для распознавания. | Робот автоматически последовательно поворачивает голову в нескольких направлениях для осмотра всего поля зрения робота. | Результат совпадает с ожиданием. |  |
| Во всем поле зрения робота не обнаружено лицо для распознавания. | Робот автоматически перемещается в помещении в одном из 4х направлений движения: вперед, влево, вправо, назад. Выбор направления движения зависит от предыдущей итерации движения. | Результат совпадает с ожиданием. |  |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| В поле зрения робота обнаружено лицо распознаваемого пользователя. | Робот мониторит положение лица распознанного пользователя. Производится звуковое сопровождения распознавания пользователя. При перемещении лица из центральной области робот поворачивает головную часть в сторону пользователя. | Результат совпадает с ожиданием. |  |
| При перемещении робота обнаружено крупное препятствие в менее чем 30 см от ультразвукового датчика. | Производится смена направления движения. | Результат совпадает с ожиданием. |  |
| При перемещении робота обнаружено мелкое препятствие менее, чем в 10 см от инфракрасного датчика. | Производится остановка и объезд препятствия путем выбора безопасного направления движения на основе показаний инфракрасных датчиков. | Результат совпадает с ожиданием. |  |
| Пользователь произносит одну из следующих фраз: «Привет, Робо», «Привет», «Hi, Robo», «Hi» или «Hello». | Робот воспроизводит сгенерированный ответ, согласно списку команд из таблицы. | Результат совпадает с ожиданием. |  |
| Пользователь произносит одну из следующих фраз: «Скажи время», «Текущее время», «Current time», «Tell time». | Робот воспроизводит сгенерированный ответ, согласно списку команд из таблицы. | Результат совпадает с ожиданием. |  |
| Пользователь произносит одну из следующих фраз: «Робо, как дела?», «Как дела?», «How are you?». | Робот воспроизводит сгенерированный ответ, согласно списку команд из таблицы. | Результат совпадает с ожиданием. |  |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| Пользователь произносит одну из следующих фраз: «Скажи настройки», «Текущие настройки», «Опции» «Current settings», «Settings». | Робот воспроизводит сгенерированный ответ, согласно списку команд из таблицы. | Результат совпадает с ожиданием. |  |
| Изменение настроек камеры | | | |
| В мобильном приложении было включено распознавание лиц и отключен мониторинг субъекта. В поле зрения робота обнаружено лицо распознаваемого пользователя. | Робот остается в неподвижном положении. Производится звуковое сопровождения распознавания пользователя. При значительном перемещении лица из центральной области робот перестает фиксировать обнаружение пользователя и производит автоматическое передвижение головной части корпуса для повторного обнаружения пользователя. | Результат совпадает с ожиданием. |  |
| В мобильном приложении было отключено распознавание лиц. В поле зрения робота обнаружено лицо распознаваемого пользователя. | Робот автоматически перемещается в помещении в одном из 4х направлений движения: вперед, влево, вправо, назад. Выбор направления движения зависит от предыдущей итерации движения. | Результат совпадает с ожиданием. |  |
| Изменение настроек звука и микрофона | | | |
| В поле зрения робота обнаружено лицо распознаваемого пользователя. В мобильном приложении отключен звук для робота. | Робот мониторит положение лица распознанного пользователя. Звуковое сопровождение не производится. При перемещении лица из центральной области робот поворачивает головную часть в сторону пользователя. | Результат совпадает с ожиданием. | |
| В поле зрения робота обнаружено лицо распознаваемого пользователя. В мобильном приложении включено распознавание лиц, отключен звук и мониторинг субъекта. | Робот остается в неподвижном положении. Звуковое сопровождение не производится. При значительном перемещении лица из центральной области робот перестает фиксировать обнаружение пользователя и производит автоматическое передвижение головной части корпуса для повторного обнаружения пользователя. | Результат совпадает с ожиданием. | |

Окончание таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| В мобильном приложении была вызвана команда рестарт, звук для робота отключен. | Звуковое сопровождение отключения отсутствует. | Результат совпадает с ожиданием. |
| Пользователь произносит любую фразу из списка поддерживаемых (см. таблицу). В мобильном приложении отключен звук для робота. | Робот не производит запись звука, звуковой ответ отсутствует. | Результат совпадает с ожиданием. |
| Пользователь произносит любую фразу из списка поддерживаемых (см. таблицу). В мобильном приложении отключены микрофоны робота. | Робот не производит запись звука, звуковой ответ отсутствует. | Результат совпадает с ожиданием. |
| Изменение режима функционирования | | |
| В мобильном приложении выбран режим «Standby». | Робот не производит движения, независимо от попадания в поле зрения робота лица пользователя или фиксирования препятствий датчиками. | Результат совпадает с ожиданием. |
| Пользователь произносит любую фразу из списка поддерживаемых (см. таблицу) на расстоянии более 3 метров от робота. В мобильном приложении выбран режим «Standby». | Запись звука производится дальним микрофоном. Робот корректно распознает команды. | Результат совпадает с ожиданием. |

**5.1.2 Мобильное приложение**

Таблица 5.2 – Тестирование работы мобильного приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Действия | Ожидание | Результат |  |
| 1 | 2 | 3 |  |
| Нажатие на элемент «Restart Robo» списка опций экрана меню. | «Мультизадачный робот» начнет перезагрузку. Мобильное приложение вернется к экрану выбора подключенного Bluetooth устройства. | Результат совпадает с ожиданием. |  |

Продолжение таблицы 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| Выбор «Мультизадачного робота» из списка подключенных Bluetooth устройств при входе в приложение. | Отобразиться экран меню с диалогом загрузки. Диалог загрузки завершиться, появиться надпись «Connected» и отобразится экран главного меню. | Результат совпадает с ожиданием. |  |
| Нажатие на элемент «About Robo» списка опций экрана меню. | На экране появиться диалоговое окно с краткой информацией о разработанном приложении. | Результат совпадает с ожиданием. |  |
| Нажатие на элемент «Disconnect» списка опций экрана меню. | Мобильное приложение вернется к экрану выбора Bluetooth устройства. Отобразится надпись «Disconnected». | Результат совпадает с ожиданием. |  |
| Нажатие на элемент «Exit» списка опций экрана меню. | Отобразится надпись «Disconnected». Приложение будет закрыто. | Результат совпадает с ожиданием. |  |
| Выбран элемент меню «Camera». | Откроется окно настроек камеры. | Результат совпадает с ожиданием. |  |
| Выбран элемент меню «Sound and mic». | Откроется окно настроек звука и микрофона. | Результат совпадает с ожиданием. |  |
| Выбран элемент меню «Wi-Fi connection». | Откроется окно установки Wi-Fi соединения. | Результат совпадает с ожиданием. |  |
| Выбран элемент меню «Operation mode». | Откроется окно выбора режима функционирования. | Результат совпадает с ожиданием. |  |
| Выбран элемент меню «Find my Robo». | Откроется окно поиска робота. | Результат совпадает с ожиданием. |  |
| На экране настроек камеры нажата кнопка со значком «+». | Появляется диалог с предупреждением о приостановке работы робота и добавлению нового субъекта распознавания. | Результат совпадает с ожиданием. |  |
| В окне диалога с предупреждением о добавлении нового субъекта распознавания был выбран вариант «Ок». | Диалог закрывается. Робот приостанавливает работу, ожидает попадание в поле зрения камеры лица и делает 100 снимков. Попадание в поле зрения робота пользователя сопровождается звуковым сигналом. По выполнению заданного количества снимков робот переобучает классификатор, производит звуковой сигнал и продолжает выполнение работы. Выпадающий список закрывается. | Результат совпадает с ожиданием. |  |

Продолжение таблицы 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| В окне диалога с предупреждением о добавлении нового субъекта распознавания был выбран вариант «Cancel». | Диалог закрывается. | Результат совпадает с ожиданием. |  |
| На экране настроек камеры нажат переключатель «Enable face recognition». | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Результат совпадает с ожиданием. |  |
| На экране настроек камеры нажат переключатель «Enable subject monitoring». | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Результат совпадает с ожиданием. |  |
| На экране настроек камеры нажат выпадающий список Saved subjects. | Появляется список элементов с номерами и картинками распознаваемых субъектов. | Результат совпадает с ожиданием. |  |
| Произведено долгое нажатие на элемент выпадающего списка | Появляется диалог с предупреждением о приостановке работы робота и удалению выбранного субъекта распознавания. | Результат совпадает с ожиданием. |  |
| В окне диалога с предупреждением об удалении выбранного субъекта распознавания был выбран вариант «Ок». | Диалог закрывается. Робот приостанавливает работ и выполняет удаление изображений субъекта по номеру. После выполнения удаления робот переобучает классификатор, производит звуковой сигнал и продолжает выполнение работы. Производится удаление выбранного субъекта из списка распознаваемых субъектов. | Результат совпадает с ожиданием. |  |
| В окне диалога с предупреждением о добавлении нового субъекта распознавания был выбран вариант «Cancel». | Диалог закрывается. | Результат совпадает с ожиданием. |  |
| На экране настроек звука и микрофона нажат переключатель «Enable sound». | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Результат совпадает с ожиданием. |  |

Окончание таблицы 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| На экране настроек звука и микрофона нажат переключатель «Enable microphone». | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Результат совпадает с ожиданием. |  |
| На экране настроек режима функционирования произведено переключение радиокнопки. | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Результат совпадает с ожиданием. |  |
| На экране «Wi-Fi connection» были оставлены пустыми поля SSID и PSK для существующей cети. Нажата кнопка «START CONFIGURATION». | Появляется диалог с надпись «Please enter SSID and PSK for proper Wi-Fi connection». | Результат совпадает с ожиданием. |  |
| На экране «Wi-Fi connection» были введены корректные поля SSID и PSK для существующей и доступной Wi-Fi cети. Нажата кнопка «START CONFIGURATION». | Мобильное приложение отправляет настройки сети и начинает получать сообщения от робота, отображаемые в поле «Messages». По завершению установки подключения к новой Wi-Fi в поле «Messages» появляется сообщение «Status succeeded». | Результат совпадает с ожиданием. |  |
| На экране «FindMyRobo» кнопка включения поиска была переведена в положение «On». | Робот начинает издавать громкий периодический звук. | Результат совпадает с ожиданием. |  |
| На экране «FindMyRobo» кнопка включения поиска была переведена в положение «Off». | Робот прекращает издавать громкий периодический звук. | Результат совпадает с ожиданием. |  |

**5.2 Проблемы с Bluetooth соединением**

Предусловие: В процессе работы мобильного приложения произошел разрыв Bluetooth соединения.

Цель проверки: установить наличие проблем с Bluetooth соединение в системе.

Результаты тестирование на наличие проблем с Bluetooth соединением представлены в таблице 5.3.

Таблица 5.3 – Обнаружение проблем с Bluetooth соединением

|  |  |  |  |
| --- | --- | --- | --- |
| Действия | Ожидание | Результат |  |
| 1 | 2 | 3 |  |
| Пользователь применяет новые настройки для звука, микрофона, распознавания лиц, мониторинга субъектов или режима функционирования. | Переключатель меняет положение. Появляется всплывающее сообщение «Settings successfully set». Приложение продолжает поддерживать соединение с роботом. | Появляется диалоговое окно прогресса установки подключения. Мобильно приложение пытается подключиться к «Мультизадачному роботу». Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. | с |
| На экране настроек камеры нажат выпадающий список Saved subjects. | Появляется список элементов с номерами и картинками распознаваемых субъектов. | Появляется диалоговое окно прогресса установки подключения. Мобильно приложение пытается подключиться к «Мультизадачному роботу». Экран настроек камеры закрывается. Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. |  |
| До падения соединения в мобильном приложении был открыт список субъектов распознавания. В окне диалога с предупреждением о добавлении нового субъекта распознавания был выбран вариант «Ок». | Диалог закрывается. Робот приостанавливает работу, ожидает попадания в поле зрения камеры лица и делает 100 снимков. Снимки пользователя сопровождаются звуковым сигналом. По выполнению заданного количества снимков робот переобучает классификатор, производит звуковой сигнал и продолжает выполнение работы. Выпадающий список закрывается. | Экран настроек камеры закрывается. Появляется диалоговое окно прогресса установки подключения. Мобильно приложение пытается подключиться к «Мультизадачному роботу». Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. |  |

Продолжение таблицы 5.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| До падения соединения в мобильном приложении был открыт список субъектов распознавания. В окне диалога с предупреждением об удалении выбранного субъекта распознавания был выбран вариант «Ок». | Диалог закрывается. Робот приостанавливает работ и выполняет удаление изображений субъекта по номеру. После выполнения удаления робот переобучает классификатор, производит звуковой сигнал и продолжает выполнение работы. Производится удаление выбранного субъекта из списка распознаваемых субъектов. | Экран настроек камеры закрывается. Появляется диалоговое окно прогресса установки подключения. Мобильно приложение пытается подключиться к «Мультизадачному роботу». Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. |  |
| На экране «Wi-Fi connection» были введены корректные поля SSID и PSK для существующей cети. Нажата кнопка «START CONFIGURATION». | Мобильное приложение отправляет настройки сети и начинает получать сообщения от робота, отображаемые в поле «Messages». По завершению установки подключения робота к новой Wi-Fi сети в поле «Messages» появляется сообщение «Status succeeded». | Экран «Wi-Fi connection» закрывается. Появляется диалоговое окно прогресса установки подключения. Мобильно приложение пытается подключиться к «Мультизадачному роботу». Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. |  |
| На экране «FindMyRobo» кнопка включения поиска была переведена из положения «Off» в положение «On». | Робот начинает издавать громкий периодический звук. | Экран «FindMyRobo» закрывается. Появляется диалоговое окно прогресса установки подключения. Мобильное приложение пытается подключиться к «Мультизадачному роботу». Главный экран меню закрывается, происходит переход на экран списка устройств для подключения. |  |

**5.3 Проблемы с соединением по RS-232**

Предусловие: Мобильное устройство подключено по Bluetooth к мультизадачному роботу. Настройки робота по умолчанию: режим функционирование standalone (перемещение разрешено, запись звука осуществляется с ближнего микрофона), звуковое сопровождение включено, запись звука с микрофонов включена, распознавание лиц включено, мониторинг субъекта распознавания включен. Соединение Arduino Uno и Raspberry Pi по протоколу RS-232 было разорвано по причине аппаратных или программных неполадок.

Цель проверки: установить наличие проблем с соединением Arduino Uno и Raspberry.

Результаты тестирования на наличие проблем с соединением по RS-232 отображены в таблице 5.4.

Таблица 5.4 – Обнаружение проблем с соединением по RS-232

|  |  |  |  |
| --- | --- | --- | --- |
| Действия | Ожидание | Результат |  |
| 1 | 2 | 3 |  |
| Перед роботом не обнаружено лицо для распознавания. | Робот автоматически последовательно поворачивает голову в нескольких направлениях для осмотра всего поля зрения робота. | Робот не поворачивает голову в нескольких направлениях для осмотра всего поля зрения робота. |  |
| Во всем поле зрения робота не обнаружено лицо для распознавания. | Робот автоматически перемещается в помещении в одном из 4х направлений движения: вперед, влево, вправо, назад. Выбор направления движения зависит от предыдущей итерации движения. | Робот не перемещается в помещении. |  |
| В поле зрения робота обнаружено лицо распознаваемого пользователя. | Робот мониторит положение лица распознанного пользователя. Производится звуковое сопровождения распознавания пользователя. При перемещении лица из центральной области робот поворачивает головную часть в сторону пользователя. | Робот не производит поворотов головной части. Производится звуковое сопровождения распознавания пользователя. |  |

Продолжение таблицы 5.4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 |  |
| В мобильном приложении выбран режим «Standby». | Робот не производит движения, независимо от попадания в поле зрения робота лица пользователя или фиксирования препятствий датчиками. | Результат совпадает с ожиданием. |  |
| Робот перемещается в помещении. С мобильного устройства приходит команда смены режима функционирования на standby. | Робот останавливает перемещение в пространстве. | Робот не останавливает перемещение в пространстве. |  |

**5.4 Выводы по результатам тестирования программного обеспечения**

По результатам проведения тестирования для проверки корректности и исправности функционирования «Мультизадачного робота» была подтверждена исправная работа программного обеспечения. Робот функционирует согласно поставленным задачам и реагирует на аппаратные неполадки в соответствии с ожидаемым поведением. Для обнаружения неполадок, связанных с аппаратной частью, были предусмотрены соответствующие планы для ручного тестирования.

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

В данном разделе будет приведена информация для конечного пользователя, входящая в комплект поставляемого программного продукта.

**6.1 Развертывание программной части на аппаратной платформе**

**6.1.1 Требования к аппаратной составляющей платформы**

Для благополучного запуска программного обеспечения аппаратная часть «Мультизадачного робота», обязательно включающая в себя подключенные и исправно работающие компоненты, описанные в дипломном проекте «Мультизадачный робот. Аппаратная часть», должна удовлетворять следующим требованиям:

* микрокомпьютер Raspberry Pi 4 Model B;
* ОЗУ 2Гб и более;
* не менее 10 процентов свободного места на SD карте;
* подключение Arduino Uno через кабель, поддерживающий стандарт USB 2.0 и выше.

**6.1.2 Требования к программному обеспечению аппаратной платформы**

Для благополучного запуска программы микрокомпьютер Raspberry должен удовлетворять следующим требованиям:

* версия ядра операционной системы Raspbian не ниже 4.19;
* наличие установленных драйверов последней версии для seeed-respeaker 2-mics pi hat;
* установленный пакет mpg321 для воспроизведения звукового сопровождения;
* замена словаря пакета pocketsphinx на файл pronounciation-dictionary.dict, содержащий урезанное число воспроизводимых команд для повышения точности офлайн распознавания и поставляемый вместе с модулем интерактивного взаимодействия;
* наличие списка дополнительных предустановленных библиотек языка Python, представленного далее в таблице 6.1.

Таблица 6.1 – Работа аппаратной платформы

|  |  |
| --- | --- |
| Пакет | Минимальная устанавливаемая версия |
| 1 | 2 |
| Flask | 1.0.2 |
| proxy | 1.3.1 |
| numpy | 1.16.2 |
| opencv-contrib-python | 4.1.0.25 |

Продолжение таблицы 6.1

|  |  |
| --- | --- |
| 1 | 2 |
| Picamera | 1.13 |
| piglow | 1.2.5 |
| Pillow | 5.4.1 |
| pocketsphinx | 0.1.15 |
| psutil | 5.5.1 |
| py-espeak-ng | 0.1.8 |
| PyBluez | 0.23 |
| PyGObject | 3.30.4 |
| pyOpenSSL | 19.0.0 |
| pyserial | 3.4 |
| python-apt | 1.8.4.1 |
| SpeechRecognition | 3.8.1 |
| urllib3 | 1.24.1 |
| wifi | 0.3.8 |

**6.1.3 Установка модуля интерактивного взаимодействия**

Для установки модуля интерактивного взаимодействия на микрокомпьютер необходимо предварительно распаковать архив Robo.zip в корень домашней папки пользователей /home/. Далее необходимо выполнить команду sudo ./install из директории распакованного архива.

Файл install представляет собой bash скрипт для установки сервисов в системе и создания директории для изображений субъектов распознавания:

#!/bin/bash

# make directory for user's photo

mkdir data

systemctl daemon-reload

cp interactionprocessor.service /etc/systemd/system/

systemctl start interactionprocessorservice

systemctl enable interactionprocessor.service

cp soundrecognition.service /etc/systemd/system/

systemctl start soundrecognition.service

systemctl enable soundrecognition.service

cp bluetoothserver.service /etc/systemd/system/

systemctl start bluetoothserver.service

systemctl enable bluetoothserver.service

Также был предусмотрен деинсталлятор uninstall, вызываемый командой sudo ./uninstall в каталоге модуля:

#!/bin/bash

systemctl stop bluetoothserver.service

systemctl stop interactionprocessorservice

systemctl stop soundrecognition.service

# remove directory for user's photo and all files

rm -r data

cd /etc/systemd/system/

systemctl daemon-reload

systemctl disable bluetoothserver.service

rm bluetoothserver.service

systemctl disable interactionprocessor.service

rm interactionprocessor.service

systemctl disable soundrecognition.service

rm soundrecognition.service

После окончания работы инсталлятора модуль интерактивного взаимодействия готов к работе. Для включения работы системы необходимо перезагрузить микрокомпьютер.

**6.1.4 Установка прошивки Arduino Uno**

Перед загрузкой прошивки в микроконтроллер необходимо предварительно распаковать собранную библиотеку Robo\_library.zip с исходными файлами в папку libraries каталога Arduino со скетчами.

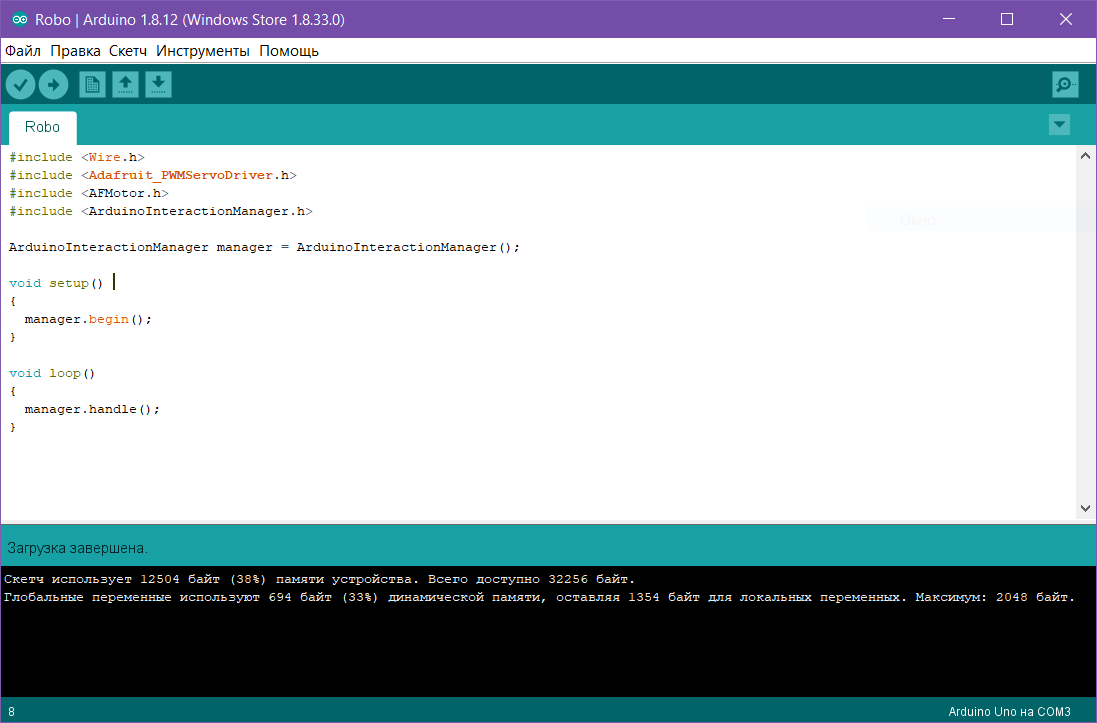


Рисунок 6.1 – Результат загрузки прошивки Robo.ino

Файл скетча прошивки Robo.ino необходимо открыть в Arduino IDE, после этого подключить к компьютеру микроконтроллер и нажать на кнопку «Загрузка». Дождаться надписи «Загрузка завершена» в окне вывода, продемонстрированного на рисунке 6.1.

**6.2 Руководство пользователя для мобильного приложения**

Для установки пользователями мобильного приложения «Robo Assistant» был собран apk файл. Для запуска процесса установки пользователю достаточно кликнуть по файлу robo\_assistant.apk и следовать инструкциям инсталлятора. Мобильное приложение не требует предварительной установки сторонних компонентов.

**6.2.1 Минимальные системные требования**

Для оптимальной работы программного обеспечения мобильное устройство должно удовлетворять следующим требованиям

* операционная система Android 8.0 и выше;
* ОЗУ 1Гб и более;
* свободное место в ПЗУ не менее 10мб;
* наличие Bluetooth.

**6.2.2 Рекомендуемые системные требования**

Для благополучного запуска программы компьютер должен удовлетворять следующим требованиям:

* операционная система Android 8.0 и выше;
* ОЗУ 1Гб и более;
* более 10 процентов свободного места в ПЗУ;
* наличие Bluetooth.

**6.2.3 Обзор интерфейса и функциональных возможностей приложения**

После успешной установки пользователем мобильного приложения на рабочем столе или в списке приложений устройства, в зависимости от графической оболочки операционной системы Android, появиться значок приложения Robo Assistant, продемонстрированный на рисунке 6.2.

По нажатию на иконку приложения пользователь попадает на экран выбора устройства для подключения (см. рисунок 6.2). Если в данный момент на мобильном устройстве не включен Bluetooth, то появиться ссылка для перехода в настройки для его включения.

В списке устройств на рисунке 6.2 отображается имя подключенного устройства и его MAC-адрес. По нажатию на элемент списка, в нашем случае Robo является именем робота, происходит подключение к выбранному устройству. Если в списке устройств не значится Robo, то необходимо подключить данное устройства в настройках Bluetooth.

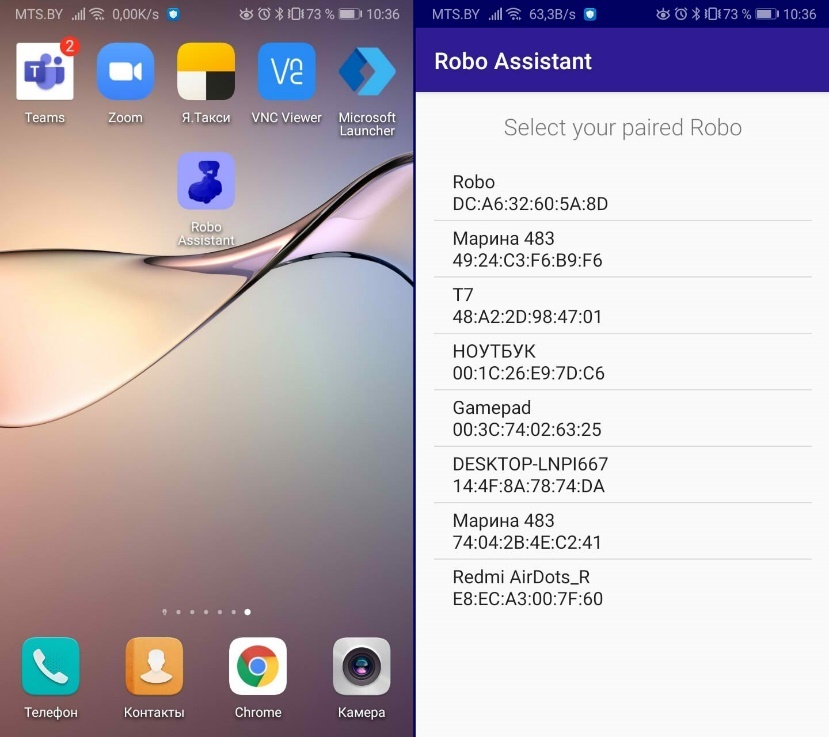
****

Рисунок 6.2 – Иконка приложения на рабочем столе с графической оболочкой EMUI версии 9.1.0 (слева) и список устройств для подключения (справа)

Далее открывается экран главного меню (см. рисунок 6.3) и появляется диалог прогресса подключения. Если робот в данный момент выключен, недоступен по причине неполадок аппаратной части или большого удаления от мобильного устройства, то экран главного меню закрывается и происходит возврат к списку подключаемых устройств. Появляется всплывающее сообщение «disconnected». Если подключение прошло успешно, то диалог прогресса подключения закрывается и появляется всплывающее сообщение «сonnected» на главном экране.

Далее пользователь получает доступ к меню мобильного приложения, отображенном на рисунке 6.3. В меню отображается список с областями настроек для перехода на соответствующий экран и статус уровня заряда батареи робота. Все настройки были вынесены на отдельные экраны для повышения удобства использования мобильного приложения

По нажатию на значок опций на главном экране появляется выпадающее меню с дополнительными функциями, такими как «Restart Robo», производящая перезагрузку операционной системы микрокомпьютера «Мультизадачного робота», «About Robo» – вызов диалогового окна с кратким описанием мобильного приложения и функционала робота, и функциями отключения от устройства «Disconnect» и выхода из приложения «Exit». Функции отключения от устройства и выход из приложения дублируют функционал аппаратных кнопок назад на мобильном устройстве.

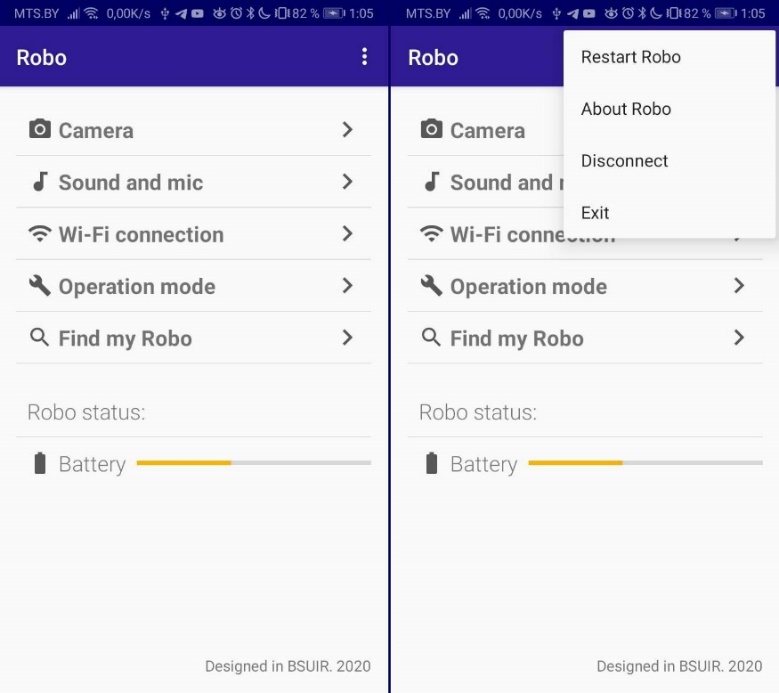
****

Рисунок 6.3 – Главное меню приложения

Список настроек содержит как текстовое, так и графическое обозначение области применения настроек.

Чтобы попасть на экран настроек камеры пользователь нажимает на элемент списка «Camera» со значком камеры.

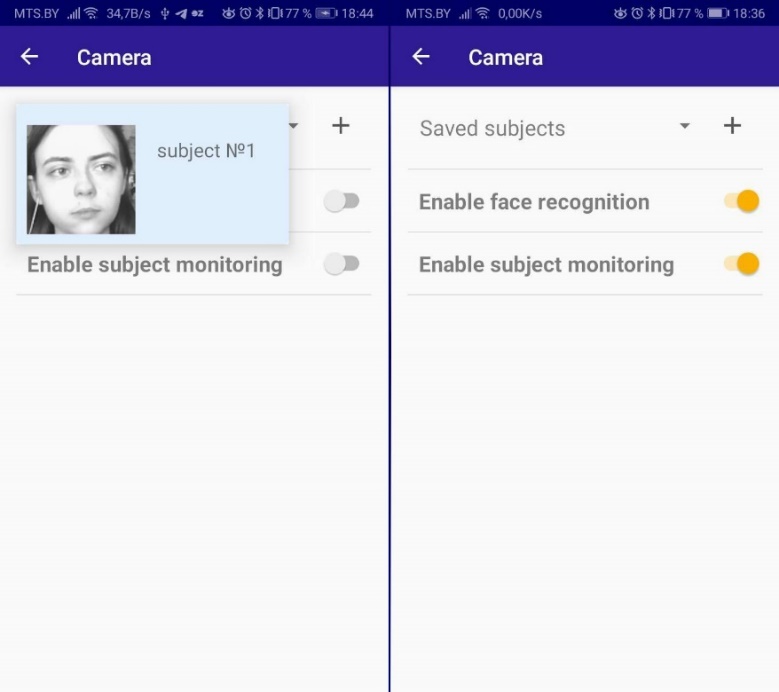
****

Рисунок 6.4 – Экран настроек камеры

На экране настроек камеры, изображенном на рисунке 6.4, пользователь имеет возможность настроить включение или отключение распознавания лиц и мониторинга субъектов для «Мультизадачного робота». Также возможно добавление нового субъекта распознавания при нажатии на кнопку со значком «+», просмотр текущих субъектов распознавания по нажатию на выпадающий список «Saved subjects» и удаление субъектов по долгому нажатию на элемент выпадающего списка.

Чтобы попасть на экран настроек звука и микрофона пользователь нажимает на элемент списка «Sound and mic» со значком ноты.

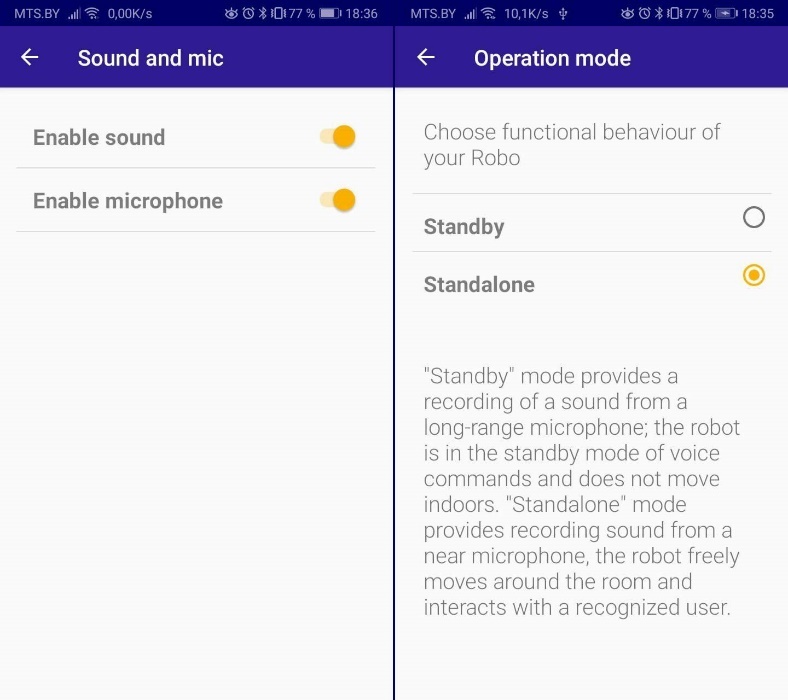
****

Рисунок 6.5 – Экран настроек звука и микрофона (слева) и экран настроек режима функционирования (справа)

На экране настроек звука, изображенном на рисунке 6.5 слева, пользователь имеет возможность настроить включение или отключение звукового сопровождения робота, а также микрофона. Необходимо учитывать, что отключение звукового сопровождения также вызовет отключение распознавания речи.

Чтобы попасть на экран настроек режима функционирования пользователь нажимает на элемент списка «Operation mode» со значком гаечного ключа.

На экране настроек режима функционирования, изображенном на рисунке 6.5 справа, пользователь имеет возможность выбрать режим функционирования «Мультизадачного робота». Режим «Standby» выключает автоматизированное перемещение робота и обеспечивает запись звука с микрофона дальнего действия. Режим «Standalone» предоставляет возможность автоматизированного перемещения и запись звука с ближнего микрофона.

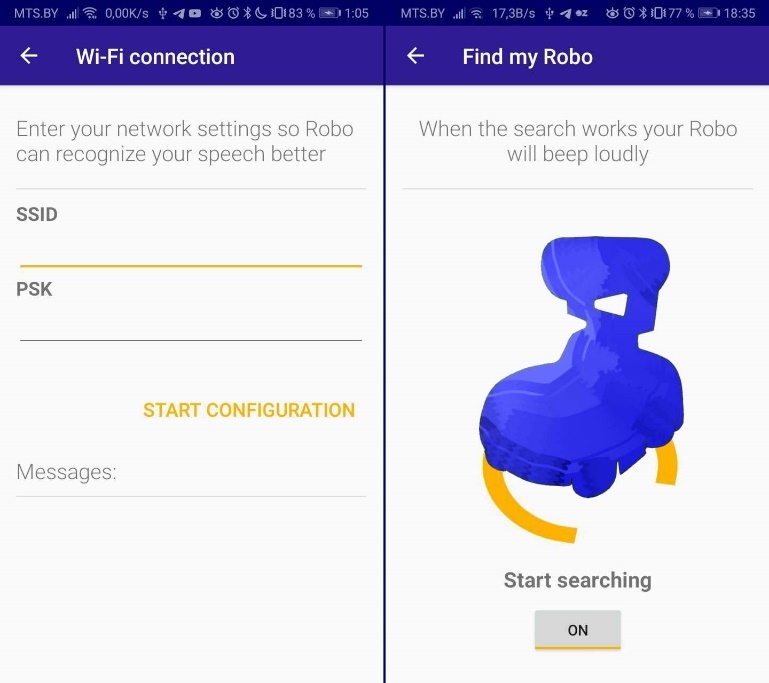
****

Рисунок 6.6 – Экран настройки Wi-Fi соединения (слева) и экран поиска робота (справа)

Чтобы попасть на экран настроек Wi-Fi подключения «Мультизадачного робота» (см. рисунок 6.6) пользователь нажимает на элемент списка «Wi-Fi connection» со значком Wi-Fi. На этом экране пользователь может ввести SSID и PSK желаемой Wi-Fi сети для подключения робота к ней. Чтобы начать конфигурацию сети пользователю необходимо нажать на кнопку «START CONFIGURATION». По мере выполнения настройки пользователь будет наблюдать сообщения от робота со статусами промежуточных этапов в поле «Messages».

Чтобы попасть на экран поиска робота (см. рисунок 6.6) пользователь нажимает на элемент списка «Find my Robo» со значком лупы. На данном экране пользователь имеет возможность нажать на двухпозиционную кнопку под надписью «Start searching», чтобы включить или выключить громкое звуковое сопровождение робота для его поиска. Во время включения поиска на экране отображается прогресс бар и логотип приложения, также изображенные на рисунке 6.6.

1. **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОЙ ЧАСТИ В СОСТАВЕ КОЛЛЕКТИВНОГО ПРОЕКТА «МУЛЬТИЗАДАЧНЫЙ РОБОТ»**
   1. **Характеристика программной части мультизадачного робота**

Программная часть мультизадачного робота представляет собой функционально законченный модуль в составе коллективного проекта «Мультизадачный робот», который предназначен для использования в качестве интеллектуальной системы мониторинга помещений с функциями голосового ассистента для использования в офисных и жилых помещениях, а также образовательных учреждениях.

К основным функциональным возможностям программной части относятся:

* обучение классификатора лиц на основе данных видеопотока;
* распознавание и классифицирование лиц на изображениях с видеопотока;
* слежение камеры за распознанным объектом в реальном времени;
* распознавание и обработка голосовых команд пользователя;
* перемещение робота в пространстве с избеганием препятствий;
* настройка режима работы распознавателей лица и речи при помощи мобильного приложения;
* настройка подключения робота к сети Wi-Fi при помощи мобильного приложения.

Отличительной особенностью разрабатываемого решения является то, что сбор данных и обучение классификатора для распознавания лиц происходят на аппаратной платформе робота в реальном времени и не зависят от вычислительных ресурсов иных устройств. Это позволяет сделать робота автоматизированным и способным работать без присутствия пользователя, реализуя концепцию устройства для «умного дома» [2].

Персональная настройка робота обеспечивается мобильным приложением посредством Bluetooth. Следовательно, нет необходимости использовать для подключения сеть Интернет или кабели для передачи данных. Данное решение не только позволяет упростить работу с «Мультизадачным роботом» для конечного пользователя, но и удешевляет распространение и внедрение разработки благодаря стремительно увеличивающейся доле пользователей мобильных устройств [1].

Таким образом, актуальность данной разработки обеспечивается широким функционалом и автономностью робота, позволяя упростить бытовые задачи пользователя и заменить сразу несколько аналогичных устройств, таких как настольный голосовой ассистент и система мониторинга помещений.

«Мультизадачный робот» имеет большой потенциал не только для персонального использования, но и для применения в сфере образования. Программное обеспечение проекта предоставляет базу для обучения учащихся и студентов разработке встроенного и системного программного обеспечения для микроконтроллеров семейства Arduino и микрокомпьютеров фирмы Raspberry Pi Foundation.

Разработка программной части мультизадачного робота будет осуществляться сотрудниками УО «Белорусский государственный университет информатики и радиоэлектроники» по индивидуальному заказу кафедры ЭВМ для последующего внедрения в научно-исследовательской лаборатории.

**7.2 Расчет инвестиций в разработку программного средства**

Расчет основной заработной платы (Зо) исполнителей осуществляется по следующей формуле

’

где Кпр – коэффициент премий;

*n* – категория исполнителей, занятых разработкой программного cредства;

З*чi*  – часовая заработная плата исполнителя *i*-й категории, р.;

𝑡*i* – трудоемкость работ, выполняемых исполнителем *i*-й категории, определяется исходя из сложности разработки программного обеспечения и объема выполняемых им функций, ч.

Список категорий исполнителей, их месячная и часовая заработная плата, а также трудоемкость работ представлены в таблице 7.1.

Согласно формуле 7.1 был произведен расчет основной заработной платы с учетом коэффициента премий, равного 1,5.

Таблица 7.1 – Расчет затрат на основную заработную плату команды

разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоемкость работ, ч. | Итого, р. |
| 1 | 2 | 3 | 4 | 5 |
| Менеджер проекта | 1200,00 | 7,14 | 200 | 1428,00 |
| Программист | 1000,00 | 5,95 | 600 | 3570,00 |
| Тестировщик | 800,00 | 4,76 | 150 | 714,00 |

Продолжение таблицы 7.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Дизайнер | 900,00 | 5,36 | 85 | 455,60 |
| Итого | | | | 6167,60 |
| Премия | | | | 9251,40 |
| Всего затраты на основную заработную плату разработчиков | | | | 15419,00 |

Дополнительная заработной платы разработчиков определяется по формуле

где Нд ‒ норматив дополнительной заработной платы.

Подставив значения соответствующих величин в формулу (7.2), получим

Для расчета отчислений на социальные нужды воспользуемся следующей формулой

где Нсоц ‒ ставка отчислений в ФСЗН и Белгосстрах.

В соответствии с действующим законодательством на момент проведения расчетов по формуле (3.3), в Республике Беларусь ставка отчислений в ФСЗН и Белгосстрах составляла 34,6 %. Соответственно, подставив рассчитанные значения, получим денежную сумму для отчислений на социальные нужды.

Для определения суммы прочих расходов воспользуемся формулой

где Нпр ‒ норматив прочих расходов, %.

Определим норму расходов равной 30 %. Подставим данную величину в формулу (7.4), и получим что

Вычислим общую суммы затрат на разработку программной части по следующей формуле:

Зр = Зо + Зд + Рсоц +  Рпр (7.5)

Подставляя рассчитанные суммы заработных плат и отчислений на социальные нужды в формулу (3.5), получим, что

Зр = + + + = 27455,07 р.

Плановая прибыль, включаемая в цену программного части, будет определена по формуле

где Рпс – рентабельность затрат на разработку программного части.

Рассчитаем денежную сумму плановой прибыли по формуле (7.6), получим

Для определения отпускной цены программного средства воспользуемся формулой

Цпс = Зр + Ппс , (7.7)

Подставив денежную сумму плановой прибыли и общую сумму затрат на разработку в формулу (7.7) получим, что

Цпс = 27455,07 +8236,52= 35691,59 р.

На основе проведенных математических расчетов, сформируем отпускную цену разрабатываемого программного средства на основе затрат (см. таблицу 7.2).

Таблица 7.2 – Формирование цены программного средства на основе затрат

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Расчет по формуле  (в таблице) | Значение, р. |
| 1. Основная заработная плата разработчиков | (см. таблицу 3.1) | 15419,00 |
| 2. Дополнительная заработная плата разработчиков |  | 1541,90 |
| 3. Отчисления на социальные нужды |  | 5868,47 |
| 4. Прочие расходы |  | 4625,70 |
| 5. Общая сумма затрат на разработку | Зр =Зо + Зд ++, (3.5) | 27455,07 |
| 6. Плановая прибыль, включаемая в цену программного средства |  | 8236,52 |
| 7. Отпускная цена программного средства | Цпс = Зр +Ппс , (3.7) | 35691,59 |

Таким образом был выполнен расчет инвестиций на разработку программной части мультизадачного робота по индивидуальному заказу.

**7.3 Расчет результата от разработки и использования**

**программного средства для организации-разработчика**

Программное средство будет реализовываться организацией разработчиком по отпускной цене, сформированной на основе затрат на разработку (см. таблицу 3.2). Экономический эффект, полученный организацией-разработчиком, в виде прироста чистой прибыли от его разработки, определяется по формуле

где Ппс ‒ прибыль, включаемая в цену программного средства, р.;

Нп – налог на прибыль, %.

В настоящий момент ставка налога на прибыль в Республике Беларусь составляет 18%. Прирост чистой прибыли по формуле (3.8) будет равен

**7.4 Расчет показателей экономической эффективности разработки и использования программного средства для организации разработчика**

Оценка экономической эффективности разработки будет осуществлена при помощи расчета простой нормы прибыли (рентабельности затрат на разработку программного средства) по формуле

где ∆Пч ‒ прирост чистой прибыли, полученной от разработки программного средства организацией-разработчиком по индивидуальному заказу, р.;

Зр ‒ затраты на разработку программного средства организацией разработчиком, р.

Вычислим оценку экономической эффективности, подставив значения в формулу (7.9), таким образом получаем

**ЗАключение**

Данное устройство имеет большой потенциал к применению в реальной жизни. Робота можно использовать для удаленного мониторинга состояния помещения, при длительном отъезде из дома, в качестве интерактивной охранной системы или просто как развлечение для детей или всей семьи. Централизованно управляющую несколькими роботами систему на основе разработанной модели можно использовать для перевозки грузов в условиях производства или для прокладки кабеля в шахте, своими размерами не позволяющей работать там человеку.

Благодаря системному подходу к проектированию возможно дальнейшее улучшение и расширение функциональности программного обеспечения «Мультизадачного робота».

К преимуществам данного проекта относятся:

* сочетание функционала голосового ассистента и системы мониторинга помещений;
* слежение камеры за распознанным объектом в реальном времени;
* возможность настройки распознавателей лица и речи через мобильное приложение;
* автоматизированное перемещение робота в пространстве с избеганием препятствий.

Недостатками данной разработки являются:

* отсутствие распознавания эмоций на изображении лица;
* отсутствие SDK для пользовательской разработки.

Дальнейшие варианты развития и улучшения функциональности:

* добавление новых голосовых команд для управления роботом;
* распознавание эмоций лица пользователя;
* реализация web-клиента для подключения к роботу по Wi-Fi сети.

**СПИСОК ИСПОЛЬЗованных ИСТОЧНИКОВ**

1. Sberbank Robotics Laboratory, Аналитический обзор мирового рынка робототехники 2019 [Электронный ресурс] / Лаборатория робототехники Сбербанка. – Электронные данные. – Режим доступа: https://www.sberbank.ru/common/img/uploaded/pdf/sberbank\_robotics\_review\_2019\_17.07.2019\_m.pdf–Дата доступа: 23.03.2020.
2. РБК Тренды. Будущее за роботами: 11 трендов развития робототехники в ближайшие годы [Электронный ресурс]. Режим доступа: https://www.rbc.ru/trends/innovation/5d6feaba9a79479e9bfce47e–Дата доступа: 23.03.2020.
3. Ideaing. Here Are The Best Robot Personal Assistants for Your Home in 2020 [Электронный ресурс]. Режим доступа: https://www.rbc.ru/trends/innovation/5d6feaba9a79479e9bfce47e–Дата доступа: 23.03.2020.
4. Spykeeworld. Best Home Security Robot 2019 – Review & Buyer’s Guide [Электронный ресурс]. Режим доступа: https://www.spykeeworld.com/best-home-security-robot/–Дата доступа: 23.03.2020.
5. Robohunter. Мобильный робот xTurion для умного дома [Электронный̆ ресурс]. Режим доступа: https://robo-hunter.com/resume/mobilnii-robot-xturion-dlya-umnogo-doma/–Дата доступа 10.02.2020.
6. BUDDY The Emotional Robot [Электронный ресурс]. Режим доступа: https://www.spykeeworld.com/best-home-security-robot/–Дата доступа: 23.03.2020.
7. Robohunter. Робот Buddy [Электронный ресурс]. Режим доступа: https://robo-hunter.com/resume/robot-buddy/–Дата доступа: 23.03.2020.
8. Arduino [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.arduino.cc/en/main/–Дата доступа: 23.03.2020.
9. Arduino IDE [Электронный̆ ресурс]. – Электронные данные. – Режим доступа: https://www.arduino.cc/en/Main/Software–Дата доступа 25.03.2020.
10. Raspbian [Электронный̆ ресурс]. – Электронные данные. – Режим доступа: https://www.raspberrypi.org/downloads/raspbian/–Дата доступа 25.03.2020.
11. Engadget. So you got a Raspberry Pi: now what? [Электронный ресурс]. Режим доступа: https://www.engadget.com/2012-09-04-raspberry-pi-getting-started-guide-how-to.html–Дата доступа: 23.03.2020.
12. SparkFun Electronics ®. Raspberry Pi Python IDE Comparison [Электронный ресурс]. Режим доступа: https://www.sparkfun.com/news/2706–Дата доступа: 23.03.2020.
13. Wikipedia. Python [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Python–Дата доступа: 23.03.2020.
14. Can you help me. Защита от распознавания лиц [Электронный ресурс]. Режим доступа: https://can-you-help.me/lajfxak-texnologiya-i-budushhee/zashhita-ot-raspoznavaniya-licz.html–Дата доступа: 23.03.2020.
15. Информационный портал habr. Применение локальных бинарных шаблонов к решению задачи распознавания лиц [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/193658/–Дата доступа: 28.03.2020.
16. Kadir, K. A Comparative Study Between LBP and Haar-Like Features for Face Detection Using OpenCV [Электронный ресурс]/ Kadir, K.; Kamaruddin, M.K.; Nasir, H.; Safie, S.I.; Bakti, Z.A.K.// In Proceedings of the International Conference on Engineering Technology and Technopreneuship, Kuala Lumpur, Malaysia; pp. 335–339, 26–28 August 2014 – Режим доступа: https://www.researchgate.net/publication/308836179\_A\_comparative\_study\_between\_LBP\_and\_Haar-like\_features\_for\_Face\_Detection\_using\_OpenCV–Дата доступа: 28.03.2020.
17. AlternativeTo. AWS Machine Learning [Электронный ресурс]. Режим доступа: https://alternativeto.net/software/aws-machine-learning/–Дата доступа: 23.03.2020.
18. Bah, S.M An improved face recognition algorithm and its application in attendance management system [Электронный ресурс] / Bah, S.M; Ming, F. // Array 2019, 5, 100014 – Режим доступа: https://www.researchgate.net/publication/308836179\_A\_comparative\_study\_between\_LBP\_and\_Haar-like\_features\_for\_Face\_Detection\_using\_OpenCV–Дата доступа: 28.03.2020.
19. Real Python. The Ultimate Guide To Speech Recognition With Python [Электронный ресурс]. Режим доступа: https://realpython.com/python-speech-recognition/#picking-a-python-speech-recognition-package/–Дата доступа: 23.03.2020.
20. Nordic APIs. 5 Best Speech-to-Text APIs [Электронный ресурс]. Режим доступа: https://nordicapis.com/5-best-speech-to-text-apis/–Дата доступа: 23.03.2020.
21. Python Package Index. SpeechRecognition 3.8.1 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://pypi.org/project/SpeechRecognition/–Дата доступа: 23.03.2020.
22. Opencv.org. About OpenCV [Электронный ресурс]. Режим доступа: https://opencv.org/about.html–Дата доступа: 25.03.2020.
23. Информационный портал habr. Распознаем лица на фото с помощью Python и OpenCV [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/301096/–Дата доступа: 28.03.2020.
24. Sharif M. Illumination Normalization Preprocessing for face recognition [Электронный ресурс]/ Sharif M., Mohsin S., Jamal M. J. and Raza //M., IEEE International Conference on Environmental Science and Information Application Technology (ESIAT), pp. 44-47, 2010 – Режим доступа: https://www.researchgate.net/publication/224173583\_Illumination\_normalization\_preprocessing\_for\_face\_recognition–Дата доступа: 28.03.2020.
25. The Robotics Back-End. Raspberry Pi Arduino Serial Communication – Everything You Need To Know [Электронный ресурс]. Режим доступа: https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/–Дата доступа: 23.03.2020.
26. PySerial [Электронный ресурс]. – Электронные данные. – Режим доступа: https://pyserial.readthedocs.io/en/latest/#–Дата доступа: 23.03.2020.
27. Scribles. Setting Up Bluetooth Serial Port Profile on Raspberry Pi using sdptool [Электронный ресурс]. Режим доступа: https://scribles.net/setting-up-bluetooth-serial-port-profile-on-raspberry-pi/–Дата доступа: 23.03.2020.
28. Android [Электронный ресурс]. – Электронные данные. – Режим доступа: https://developer.android.com/–Дата доступа: 23.03.2020.
29. Slant. What are the best IDEs for Android development? [Электронный ресурс]. Режим доступа: https://www.slant.co/topics/1321/~best-ides-for-android-development–Дата доступа: 23.03.2020.
30. Android Studio. Build a UI with Layout Editor [Электронный ресурс]. Режим доступа: https://developer.android.com/studio/write/layout-editor–Дата доступа: 23.03.2020.
31. Material Design for Android [Электронный ресурс]. Режим доступа: https://developer.android.com/guide/topics/ui/look-and-feel–Дата доступа: 23.03.2020.
32. Wikipedia. RFCOMM [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/RFCOMM–Дата доступа: 23.03.2020.
33. Android. Bluetooth overview [Электронный ресурс]. Режим доступа: https://developer.android.com/guide/topics/connectivity/bluetooth–Дата доступа: 23.03.2020.
34. Adafruit PCA9685 PWM Servo Driver Library [Электронный ресурс]. – Электронные данные. –Режим доступа: https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library–Дата доступа: 23.03.2020.
35. Random Nerd Tutorials. Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino [Электронный ресурс]. Режим доступа: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/–Дата доступа: 23.03.2020.

**ПРИЛОЖЕНИЕ А**

***(обязательное)***

**Листинг кода**

BluetoothServer.py

#

# includes parts of rfcomm-server.py

#

import signal

import os

import time

import subprocess

import json

from PIL import Image

from bluetooth import \*

import numpy as np

from io import StringIO, BytesIO

import select

import socket

import ParseSettings as settings

from wifi import Cell, Scheme

socket.setdefaulttimeout(60)

wpa\_supplicant\_conf = "/etc/wpa\_supplicant/wpa\_supplicant.conf"

sudo\_mode = "sudo "

FIFO\_Sound = 'tosound'

FIFO\_Interaction = 'tointeraction'

def send\_command\_to\_sound(command):

setting = settings.get\_setting("settings.ini","PIDS","Sound")

if setting != "0" :

os.kill(int(setting), signal.SIGUSR1)

with open(FIFO\_Sound, "w" ) as fifo:

print("FIFO opened")

fifo.write(command)

def send\_command\_to\_interaction(command):

setting = settings.get\_setting("settings.ini","PIDS","Interaction")

if setting != "0":

os.kill(int(setting), signal.SIGUSR1)

with open(FIFO\_Interaction, "w" ) as fifo:

print("FIFO opened")

fifo.write(command)

def wifi\_connect(ssid, psk):

cmd = 'wpa\_passphrase {ssid} {psk} | sudo tee -a {conf} > /dev/null'.format(

ssid=str(ssid).replace('!', '\!'),

psk=str(psk).replace('!', '\!'),

conf=wpa\_supplicant\_conf

)

str\_return = ""

str\_return = os.system(cmd)

cmd = sudo\_mode + 'wpa\_cli -i wlan0 reconfigure'

str\_return = os.system(cmd)

time.sleep(10)

cmd = 'iwconfig wlan0'

str\_return = os.system(cmd)

cmd = 'ifconfig wlan0'

str\_return = os.system(cmd)

p = subprocess.Popen(['hostname', '-I'], stdout=subprocess.PIPE,

stderr=subprocess.PIPE)

out, err = p.communicate()

if out:

ip\_address = out

else:

ip\_address = "<Not Set>"

return ip\_address

def ssid\_discovered():

Cells = Cell.all('wlan0')

# for test purpose only

# print(Cells)

# for current in Cells:

# print('\n key ')

# print(current)

# wifi\_info += current.ssid + "\n"

wait\_ssid ="/wait\_ssid/"

return wait\_ssid

def handle\_client(client\_sock) :

# get ssid

client\_sock.send(ssid\_discovered())

ssid = client\_sock.recv(1024)

if ssid == '' :

return

# get psk

client\_sock.send("/wait\_psk/")

psk = client\_sock.recv(1024)

if psk == '' :

return

ip\_address = wifi\_connect(ssid, psk)

print ('ip address: ' + str(ip\_address))

client\_sock.send('status:' + str(ip\_address))

return

def root():

return optionsStatus()

def optionsStatus():

status = {}

status['sound'] = settings.get\_setting("settings.ini","Settings","Sound")

status['mic'] = settings.get\_setting("settings.ini","Settings","Sound")

status['facerec'] = settings.get\_setting("settings.ini","Settings","Sound")

status['submon'] = settings.get\_setting("settings.ini","Settings","Sound")

status['opmode'] = settings.get\_setting("settings.ini","Settings","Sound")

status['battery'] = settings.get\_setting("settings.ini","Settings","Battery")

# JSON encode and transmit response

response = json.dumps(status)

client\_sock.send(response)

def count\_subjects():

path = './data'

imagePaths = [os.path.join(path,f) for f in os.listdir(path)]

ids = []

for imagePath in imagePaths:

id = int(os.path.split(imagePath)[-1].split(".")[1])

ids.append(id)

print (ids)

responce = {}

responce['subjects'] = ids

response = json.dumps(status)

client\_sock.send(response)

# add serial port service

subprocess.call(['sudo', 'sdptool', 'add', 'SP'])

# simple-agent.py script is a part of pybluez modules examples

# check out by:

# sudo apt-get install python-module-pybluez

# wget -O simple-agent http://git.kernel.org/?p=bluetooth/bluez.git;a=blob\_plain;f=test/simple-agent;hb=HEAD

# ./simple-agent hci0 <remote address>

subprocess.call(['python3','simple-agent.py'])

# signal main loop start

signalhandler = False

try:

while True:

server\_sock = BluetoothSocket(RFCOMM)

server\_sock.bind(("", PORT\_ANY))

server\_sock.listen(1)

port = server\_sock.getsockname()[1]

uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

advertise\_service( server\_sock, "roboserver-bt",

service\_id = uuid,

service\_classes = [uuid, SERIAL\_PORT\_CLASS],

profiles = [SERIAL\_PORT\_PROFILE] )

# signal server startup

print ("Server socket created")

# wait for connection

client\_sock, client\_info = server\_sock.accept()

#for test purpose only

#client\_sock.settimeout(None)

print ("Client connected")

# receiver loop

try:

partial = ''

while True:

try:

dat = client\_sock.recv(1024)

except socket.timeout as e:

err = e.args[0]

# timeout exception

if err == 'timed out':

sleep(1)

print ('recv timed out, retry later')

continue

else:

print (e)

sys.exit(1)

print("Recieve some data")

data = dat.decode("utf-8")

if len(data) == 0:

break

elif len(data) == 1024:

isBufferFull = True

else:

isBufferFull = False

# save partial received commands

commands = (partial + data).split('$$')

number = len(commands)

if isBufferFull:

partial = commands[number-1]

number -= 1

else:

partial = ''

# loop through commands

for i in range(number):

# parse the command

command = commands[i]

print(" Command = " + command)

parts = command.split('/')

count = len(parts)

def parseCommand(command, parts, count):

if command == '':

continue

elif command == '/':

root()

elif command == '/status/':

optionsStatus()

elif command == '/subjects/':

count\_subjects()

elif command == '/add/':

send\_command\_to\_interaction("add")

elif command[0:8] == '/remove/' and count == 3:

send\_command\_to\_interaction("remove " + parts[2])

elif command == '/restart/':

os.system('reboot')

elif command == '/find/':

send\_command\_to\_sound("find")

elif command == '/wifi\_connect/':

handle\_client(client\_sock)

elif command[0:8] == '/opmode/' and count == 3:

enable = parts[2]

settings.update\_setting("settings.ini","Settings","Operation mode", enable)

soundEn = settings.get\_setting("settings.ini","Settings","Operation mode")

send\_command\_to\_interaction("opmode " + soundEn)

elif command[0:8] == '/submon/' and count == 3:

enable = parts[2]

settings.update\_setting("settings.ini","Settings","Subject monitoring", enable)

soundEn = settings.get\_setting("settings.ini","Settings","Subject monitoring")

send\_command\_to\_interaction("submon " + soundEn)

elif command[0:9] == '/facerec/' and count == 3:

enable = parts[2]

settings.update\_setting("settings.ini","Settings","Face recognition", enable)

soundEn = settings.get\_setting("settings.ini","Settings","Face recognition")

send\_command\_to\_interaction("facerec " + soundEn)

elif command[0:5] == '/mic/' and count == 3:

enable = parts[2]

settings.update\_setting("settings.ini","Settings","Mic", enable)

soundEn = settings.get\_setting("settings.ini","Settings","Mic")

send\_command\_to\_interaction("mic " + soundEn)

elif command[0:7] == '/sound/' and count == 3:

enable = parts[2]

settings.update\_setting("settings.ini","Settings","Sound", enable)

soundEn = settings.get\_setting("settings.ini","Settings","Sound")

send\_command\_to\_interaction("sound " + soundEn)

elif command[0:7] == '/photo/':

number\_ = 1

if count == 3:

number\_ = parts[2]

print("Count = " + str(count) + " part[2] = " + str(parts[2]) + " number = " + str(number))

imagePath = '/home/raspberry/data/subject.' + str(number\_) + '.1.jpg'

with open(imagePath, mode='rb') as file:

fileContent = file.read()

size = len(fileContent)

print("{" + str(size) + "}")

client\_sock.settimeout(5.0)

client\_sock.sendall(struct.pack(">L", size) + fileContent)

client\_sock.settimeout(None)

print("Sent photo done")

else:

print('unrecognized command')

parseCommand(command, parts, count)

# receiver loop try

except IOError as err:

print("IOError: {0}".format(err))

pass

# connection lost

client\_sock.close()

server\_sock.close()

# signal disconnect

# main loop try

except KeyboardInterrupt:

client\_sock.close()

server\_sock.close()

FaceDatasetCreate.py

import cv2

import numpy

import os

import FacePreProcessing as fp

# Path for face image database

path = './data'

def Get\_Paths():

return [os.path.join(path,f) for f in sorted(os.listdir(path))]

def Get\_Id(imagePath):

return int(os.path.split(imagePath)[-1].split(".")[1])

def Get\_Next\_Id():

imagePaths = [os.path.join(path,f) for f in sorted(os.listdir(path))]

cur\_id = 0

id = 0

for imagePath in imagePaths:

print ('\n Path = ' + imagePath)

cur\_id = int(os.path.split(imagePath)[-1].split(".")[1])

print('Subject id = ' + os.path.split(imagePath)[-1].split(".")[1])

if cur\_id == id +1:

id = cur\_id

elif cur\_id > (id + 1):

return id + 1

return id + 1

def Delete\_Subject(num):

imagePaths = [os.path.join(path,f) for f in sorted(os.listdir(path))]

cur\_id = 0

id = 0

i = 0

for imagePath in imagePaths:

cur\_id = int(os.path.split(imagePath)[-1].split(".")[1])

if cur\_id == num:

os.remove(imagePath)

return

def Create\_Dataset(sound\_enable):

print("\n Create face dataset ...")

cam = cv2.VideoCapture(0)

cam.set(3, 640) # set video width

cam.set(4, 480) # set video height

face\_detector = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

id = Get\_Next\_Id()

if sound\_enable == 1:

os.system('mpg321 -a plughw:1,0 ./Sounds/blip.mp3')

count = 0

while(count < 100):

ret, img = cam.read()

img = cv2.flip(img, -1) # flip video image vertically

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_detector.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:

resized\_image = cv2.resize(gray[y:y+h,x:x+w],(350,350))

resized\_image = fp.PreProcessing\_Origin(resized\_image, 350, 350)

path = "data/subject." + str(id) + '.' + str(count) + ".bmp"

cv2.imwrite(path, resized\_image)

if count == 100:

break;

count += 1

print(count)

if sound\_enable == 1:

os.system('mpg321 -a plughw:1,0 ./Sounds/blip.mp3')

cam.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == '\_\_main\_\_':

Create\_Dataset()

FaceDetection.py

import cv2

import FacePreProcessing as fp

# iniciate id counter

id = 0

# Initialize and start realtime video capture

cam = cv2.VideoCapture(0)

cam.set(3, 640) # set video widht

cam.set(4, 480) # set video height

# Define min window size to be recognized as a face

minW = 0.1 \* cam.get(3)

minH = 0.1 \* cam.get(4)

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath);

ix = 0

iy = 0

iw = 0

ih = 0

def Release\_Cam():

global cam

cam.release()

def Reset\_Cam():

global cam

cam = cv2.VideoCapture(0)

cam.set(3, 640) # set video widht

cam.set(4, 480) # set video height

def Find\_Face():

global ix

global iy

global iw

global ih

ret, img = cam.read()

img = cv2.flip(img, -1) # flip video image vertically

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

found = False

faces = faceCascade.detectMultiScale(

gray,

scaleFactor=1.2,

minNeighbors=5,

minSize=(int(minW), int(minH))

)

for (x, y, w, h) in faces:

resized\_image = cv2.resize(gray[y:y + h, x:x + w], (350, 350))

resized\_image = fp.PreProcessing\_Origin(resized\_image, 350, 350)

ix = x

iy = y

iw = w

ih = h

return True, resized\_image

return False, 0

FacePreProcessing.py

import cv2

import os

import numpy as np

def adjust\_gamma(image, gamma=1.0):

invGamma = 1.0 / gamma

table = np.array([((i / 255.0) \*\* invGamma) \* 255

for i in np.arange(0, 256)]).astype("uint8")

return cv2.LUT(image, table)

def histeq(im,nbr\_bins=256):

#get image histogram

imhist,bins = np.histogram(im.flatten(),nbr\_bins,normed=True)

cdf = imhist.cumsum() #cumulative distribution function

cdf = 255 \* cdf / cdf[-1] #normalize

im2 = np.interp(im.flatten(),bins[:-1],cdf)

return im2.reshape(im.shape), cdf

def PreProcessing\_Origin(im, w, h):

max\_brightness = 0.7

min\_brightness = 0.3

brightness = np.sum(im) / (255 \* w \* h)

ratio = brightness / max\_brightness

if ratio >= 1:

print("Image bright enough")

im = cv2.adjust\_gamma(im,1/ratio)

ration = brightness / min\_brightness

if ratio < 1:

print("Image not bright enough")

im = cv2.adjust\_gamma(im,1/ratio)

vertSeg = h//2

horSeg = w//2

X1 = im[0:vertSeg ,0:horSeg]

X2 = im[0:vertSeg ,horSeg:w]

X3 = im[vertSeg:h ,0:horSeg]

X4 = im[vertSeg:h ,horSeg:w]

X1 = np.array(X1, dtype = np.uint8)

X2 = np.array(X2, dtype = np.uint8)

X3 = np.array(X3, dtype = np.uint8)

X4 = np.array(X4, dtype = np.uint8)

X1 = histeq(X1)[0]

X2 = histeq(X2)[0]

X3 = histeq(X3)[0]

X4 = histeq(X4)[0]

up = np.concatenate((X1, X2), axis=1)

down = np.concatenate((X3, X4), axis=1)

X = np.concatenate((up, down), axis=0)

Y2 = np.array(X, dtype = np.float64)

Y3 = Y2

for i in range(1,h-2):

for j in range(1, w-2):

summ = (Y2[i,j] + Y2[i,j+1] + Y2[i,j-1] + Y2[i+1,j] + Y2[i-1,j] + Y2[i+1,j+1] + Y2[i+1,j-1] + Y2[i-1,j+1] + Y2[i-1,j-1])

Y3[i,j] = summ/9

Y3 = ((Y3 - Y3.min()) / (Y3.max()-Y3.min())) \* 255

data = Y3.round().astype(np.uint8)

data = unsharp\_mask(data)

return data

def unsharp\_mask(image, kernel\_size=(5, 5), sigma=1.0, amount=1.0, threshold=0):

blurred = cv2.GaussianBlur(image, kernel\_size, sigma)

sharpened = float(amount + 1) \* image - float(amount) \* blurred

sharpened = np.maximum(sharpened, np.zeros(sharpened.shape))

sharpened = np.minimum(sharpened, 255 \* np.ones(sharpened.shape))

sharpened = sharpened.round().astype(np.uint8)

if threshold > 0:

low\_contrast\_mask = np.absolute(image - blurred) < threshold

np.copyto(sharpened, image, where=low\_contrast\_mask)

return sharpened

def PreProcessing\_Basic(im, w, h):

X1 = im[0:h ,0:w//2]

X2 = im[0:h ,w//2:w]

X1 = cv2.equalizeHist(X1)

X2 = cv2.equalizeHist(X2)

X1 = np.array(X1, dtype = np.uint8)

X2 = np.array(X2, dtype = np.uint8)

X = np.concatenate((X1, X2), axis=1)

Y2 = np.array(X, dtype = np.float64)

Y3 = Y2

for i in range(1,h):

for j in range(1, w):

Y3[i,j] = (Y2[i,j+1] + Y2[i,j-1] + Y2[i+1,j] + Y2[i-1,j] + Y2[i+1,j+1] + Y2[i+1,j-1] + Y2[i-1,j+1] + Y2[i-1,j-1])/8

data = Y3.astype(np.uint8)

return data

FaceRecognitionProcessing.py

import FaceDatasetCreate as fd

import FaceRecognitionTrain as ft

import FaceDetection as detect

import FacePreProcessing as preproc

import cv2

import time

import os

import struct

import sys

import math

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('trainer/trainer.yml')

def Add\_User(sound\_enable = True):

print('add user')

detect.Release\_Cam()

fd.Create\_Dataset(sound\_enable)

ft.Train\_LBPH()

detect.Reset\_Cam()

recognizer.read('trainer/trainer.yml')

def Delete\_User(number):

fd.Delete\_Subject(number)

ft.Train\_LBPH()

recognizer.read('trainer/trainer.yml')

FaceRecognitionTrain.py

import cv2

import numpy as np

from PIL import Image

import os

# Path for face image database

path = './data'

def Train\_LBPH():

recognizer = cv2.face.LBPHFaceRecognizer\_create()

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml");

def getImagesAndLabels(path):

imagePaths = [os.path.join(path,f) for f in os.listdir(path)]

faceSamples=[]

ids = []

i = 0

for imagePath in imagePaths:

PIL\_img = Image.open(imagePath).convert('L') # convert it to grayscale

img\_numpy = np.array(PIL\_img,'uint8')

id = int(os.path.split(imagePath)[-1].split(".")[1])

faces = detector.detectMultiScale(img\_numpy)

for (x,y,w,h) in faces:

faceSamples.append(img\_numpy[y:y+h,x:x+w])

ids.append(id)

return faceSamples,ids

faces,ids = getImagesAndLabels(path)

recognizer.train(faces, np.array(ids))

recognizer.write('trainer/trainer.yml')

print("\n {0} subjects was trained".format(len(np.unique(ids))))

InteractionProcessor.py

import MovementProcessing

import FaceRecognitionProcessing

import os

import errno

import signal

import time

import cv2

import ParseSettings as ps

face\_recognition = 1

subject\_monitoring = 1

sound\_enable = 1

movement\_enable = 1

FIFO = 'tointeraction'

try:

os.mkfifo(FIFO)

except OSError as oe:

if oe.errno != errno.EEXIST:

raise

def runCommand(command):

global face\_recognition

temp = face\_recognition

print(command)

commandParam = command.split()

print (commandParam)

if command == "add":

face\_recognition = 0

MovementProcessing.Initial\_State()

FaceRecognitionProcessing.Add\_User()

face\_recognition = temp

elif commandParam[0] == 'remove':

face\_recognition = 0

MovementProcessing.Initial\_State()

FaceRecognitionProcessing.Delete\_User(commandParam[1])

face\_recognition = temp

elif commandParam[0] == 'facerec':

face\_recognition = int(commandParam[1])

elif commandParam[0] == 'submon':

global subject\_monitoring

subject\_monitoring = int(commandParam[1])

elif commandParam[0] == 'sound':

global sound\_enable

sound\_enable = int(commandParam[1])

elif commandParam[0] == 'opmode':

global movement\_enable

movement\_enable = int(commandParam[1])

if movement\_enable == 0:

print(' ')

MovementProcessing.Initial\_State()

def receiveSignal(signalNumber, frame):

print('Received:', signalNumber)

print("Opening FIFO...")

with open(FIFO, "r" ) as fifo:

print("FIFO opened")

for line in fifo:

runCommand(line)

return

def loadSetting():

face\_recognition = int(ps.get\_setting("settings.ini","Settings","Face recognition"))

subject\_monitoring = int(ps.get\_setting("settings.ini","Settings","Subject monitoring"))

sound\_enable = int(ps.get\_setting("settings.ini","Settings","Sound"))

movement\_enable = int(ps.get\_setting("settings.ini","Settings","Operation mode"))

def Search():

count = 0

detected = 0

result = False

print ('face\_recognition = ' + str(face\_recognition))

if face\_recognition == 0:

return result

while (count < 30):

found, image = FaceRecognitionProcessing.detect.Find\_Face()

if found :

id, confidence = FaceRecognitionProcessing.recognizer.predict(image)

if(confidence < 95):

result = True

print('see you')

if(detected > 10):

if movement\_enable == 1 and subject\_monitoring == 1:

if sound\_enable == 1:

os.system('mpg321 -a plughw:1,0 ./Sounds/blip.mp3')

MovementProcessing.Monitor\_User(FaceRecognitionProcessing.detect.ix,

FaceRecognitionProcessing.detect.iy,

FaceRecognitionProcessing.detect.iw,

FaceRecognitionProcessing.detect.ih)

count = 0

detected = 0

detected += 1

else:

count+=1

else :

count+=1

print('no face')

return result

ps.update\_setting("settings.ini", "PIDS", "Interaction", str(os.getpid()))

signal.signal(signal.SIGUSR1, receiveSignal)

print('Signal handler set')

loadSetting()

print('Process settings loaded')

MovementProcessing.Initial\_State()

while True:

if not Search():

print ('user not found')

if movement\_enable == 1:

if sound\_enable == 1:

os.system('mpg321 -a plughw:1,0 ./Sounds/blip2.mp3')

MovementProcessing.Movement\_Interaction()

time.sleep(1)

k = cv2.waitKey(100) & 0xff

if k == 27:

break;

MovementProcessing.py

import time

import serial

import os

import struct

import sys

import math

import SerialIntercomm as si

head\_turns = 0

prev\_direction = 0

def Initial\_State():

print('initial state')

global head\_turns

global prev\_direction

si.Send\_Command\_Vertical\_Servo(0)

si.Send\_Command\_Horizontal\_Servo(40)

si.Send\_Command\_Stop()

head\_turns = 0

prev\_direction = 0

return

def Monitor\_User(x, y, w, h):

print(':',x,y,w,h)

Find\_User(-1)

if((x + w/2 < 300) | (x+ w/2 > 340)):

distX = ((x + w/2) \* 85) / 640

si.Send\_Command\_Horizontal\_Servo(distX)

print("h" + str(distX) + "\n")

if((y + h/2 < 200) | (y + h/2 > 280)):

distY = ((y + h/2) \* 40) / 480

si.Send\_Command\_Vertical\_Servo(distY)

print("v" + str(distY) + "\n")

def Find\_User(direction):

if (direction == -1):

si.Send\_Command\_Stop()

elif(direction == 0):

si.Send\_Command\_Forward()

elif(direction == 1):

si.Send\_Command\_Left()

elif(direction == 2):

si.Send\_Command\_Right()

else:

si.Send\_Command\_Back()

def Turn\_Around\_Head(turn):

if(turn == 0):

si.Send\_Command\_Turn\_Up()

elif(turn == 1):

si.Send\_Command\_Turn\_Down()

elif (turn == 2):

si.Send\_Command\_Turn\_Left()

else:

si.Send\_Command\_Turn\_Right()

#prev\_direction = 0 - Forward , 1 - Left, 2 - Right, 3 - back

def Movement\_Interaction():

global head\_turns

global prev\_direction

# load status arduino

if(head\_turns < 4):

Turn\_Around\_Head(head\_turns)

head\_turns+=1

else:

Find\_User(prev\_direction)

if(prev\_direction == 3):

prev\_direction = 0

else:

prev\_direction+=1

head\_turns = 0

time.sleep(1)

#exitflag = False

ParseSettings.py

import configparser

import os

path = "settings.ini"

def create\_config(path):

"""

Create a config file

"""

config = configparser.ConfigParser()

config.add\_section("Settings")

config.set("Settings", "Face recognition", "1")

config.set("Settings", "Subject monitoring", "1")

config.set("Settings", "Sound", "1")

config.set("Settings", "Mic", "1")

config.set("Settings", "Operation mode", "0")

config.set("Settings", "Battery", "100")

config.add\_section("PIDS")

config.set("PIDS", "Interaction", "0")

config.set("PIDS", "Sound", "0")

with open(path, "w") as config\_file:

config.write(config\_file)

def get\_config(path):

"""

Returns the config object

"""

if not os.path.exists(path):

create\_config(path)

config = configparser.ConfigParser()

config.read(path)

return config

def get\_setting\_str():

"""

Print out a setting

"""

config = get\_config(path)

value = config.get("Settings", "Sound")

msg = "Звук установлен в {value}".format(

value=value

)

value = config.get("Settings", "Mic")

msg = msg + "Микрофон установлен в {value}".format(

value=value

)

value = config.get("Settings", "Face recognition")

msg = msg + "Распознавание установлено в {value}".format(

value=value

)

value = config.get("Settings", "Subject monitoring")

msg = msg + "Мониторинг установлен в {value}".format(

value=value

)

value = config.get("Settings", "Operation mode")

msg = msg + "Режим установлен в {value}".format(

value=value

)

print(msg)

return msg

def get\_setting(path, section, setting):

"""

Print out a setting

"""

config = get\_config(path)

value = config.get(section, setting)

msg = "{section} {setting} is {value}".format(

section=section, setting=setting, value=value

)

print(msg)

return value

def update\_setting(path, section, setting, value):

config = get\_config(path)

config.set(section, setting, value)

with open(path, "w") as config\_file:

config.write(config\_file)

def delete\_setting(path, section, setting):

config = get\_config(path)

config.remove\_option(section, setting)

with open(path, "w") as config\_file:

config.write(config\_file)

if \_\_name\_\_ == "\_\_main\_\_":

path = "settings.ini"

SerialIntercomm.py

import time

import serial

import os

import struct

import sys

import math

ser = serial.Serial(

port='/dev/ttyACM0',

baudrate=250000,

parity=serial.PARITY\_NONE,

stopbits=serial.STOPBITS\_ONE,

bytesize=serial.EIGHTBITS

)

def Send\_Command\_Stop():

ser.write(str.encode("s" + "\n"))

def Send\_Command\_Forward():

print('finding you forward...')

ser.write(str.encode("f" + "\n"))

ser.write(str.encode("s" + "\n"))

def Send\_Command\_Left():

print('finding you left...')

ser.write(str.encode("l1000" + "\n"))

ser.write(str.encode("f" + "\n"))

ser.write(str.encode("s" + "\n"))

def Send\_Command\_Right():

print('finding you right...')

ser.write(str.encode("r1000" + "\n"))

ser.write(str.encode("f" + "\n"))

ser.write(str.encode("s" + "\n"))

def Send\_Command\_Back():

print('finding you back...')

ser.write(str.encode("b" + "\n"))

ser.write(str.encode("s" + "\n"))

def Send\_Command\_Turn\_Up():

print('turn up...')

ser.write(str.encode("v5" + "\n"))

def Send\_Command\_Turn\_Down():

print('turn down...')

ser.write(str.encode("v40" + "\n"))

def Send\_Command\_Turn\_Left():

print('turn left...')

ser.write(str.encode("h5" + "\n"))

def Send\_Command\_Turn\_Right():

print('turn right...')

ser.write(str.encode("h85" + "\n"))

def Send\_Command\_Horizontal\_Servo(distX):

ser.write(str.encode("h" + str(distX) + "\n"))

def Send\_Command\_Vertical\_Servo(distY):

ser.write(str.encode("v" + str(distY) + "\n"))

def Send\_Command\_Forward\_Time(timeDeltaSmall):

ser.write("f" + "\n")

time.sleep(timeDeltaSmall)

ser.write("s" + "\n")

time.sleep(1)

def Send\_Command\_Back\_Time(timeDeltaBig):

ser.write("b" + "\n")

time.sleep(timeDeltaBig)

ser.write("s" + "\n")

time.sleep(1)

SoundAnalyzer.py

#! /usr/bin/env python

#-\*-coding:utf-8-\*-

import parseSettings as ps

import os

import speech\_recognition as sr

import SoundPlayer as sp

def Google():

r = sr.Recognizer()

with sr.WavFile("voice.wav") as source: # use "test.wav" as the audio source

audio = r.record(source) # extract audio data from the file

t = 0

try:

t = r.recognize\_google(audio,language = "ru-RU", key = "AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw")

print(t) # recognize speech using Google Speech Recognition

except LookupError: # speech is unintelligible

print("Could not understand audio")

return False

except sr.UnknownValueError:

print("Google could not understand audio")

return False

#pass

except sr.RequestError as e:

print("Google error; {0}".format(e))

return False

if t == 0:

print("Error")

elif t==("Привет") or t==("привет") or t==("Hi") or t==("Hello"): #Hi

sp.Play\_Hello()

elif t==("Скажи время") or t==("Текущее время") or t==("Current time") or t==("Tell time"):

sp.Play\_Current\_Time()

elif t==("Как дела") or t==("How are you"):

sp.Play\_HRU()

elif t==("Скажи настройки") or t==("Текущие настройки") or t==("Current settings"):

sp.Play\_Current\_Settings()

elif t.find("Привет") != -1 or t.find("привет") != -1 or t.find("Hi") != -1 or t.find("Hello") != -1: #Hi

sp.Play\_Hello()

elif t.find("Время") != -1 or t.find("время") != -1:

sp.Play\_Current\_Time()

elif t.find("Как дела") != -1 or t.find("как дела") != -1 or t.find("How are you") != -1 or t.find("how are you") != -1:

sp.Play\_HRU()

elif t.find("Настройки") != -1 or t.find("настройки") != -1 or t.find("Опции") != -1 or t.find("опции") != -1:

sp.Play\_Current\_Settings()

else:

return False

return True

def Spinx():

r = sr.Recognizer()

with sr.WavFile("voice.wav") as source:

audio = r.record(source)

t = 0

try:

t = r.recognize\_sphinx(audio)

print(t)

except LookupError:

print("Could not understand audio")

except sr.UnknownValueError:

print("Sphinx could not understand audio")

except sr.RequestError as e:

print("Sphinx error; {0}".format(e))

if t==("hi"):

sp.Play\_Hello()

elif t==("time"):

sp.Play\_Current\_Time()

elif t==("how are you"):

sp.Play\_HRU()

elif t==("settings"):

sp.Play\_Current\_Settings()

def Start\_Record():

#test purpose

r = sr.Recognizer()

with sr.WavFile("robo.wav") as source:

audio = r.record(source)

t = 0

try:

t = r.recognize\_sphinx(audio)

print(t)

except LookupError:

print("Could not understand audio")

return False

except sr.UnknownValueError:

print("Sphinx could not understand audio")

return False

except sr.RequestError as e:

print("Sphinx error; {0}".format(e))

return False

if t.find("robo") != -1 or t.find("hi") != -1 or t.find("hello") != -1 or t.find("ok") != -1 or t.find("okay") != -1:

return True

if \_\_name\_\_ == '\_\_main\_\_':

Start\_Record()

SoundPlayer.py

import os

import datetime

import parseSettings as ps

HRU = ["Все хорошо", "Пора бы отдохнуть","А у тебя как?"]

def Play\_Current\_Time():

currentDT = datetime.datetime.now()

time = 'espeak-ng --stdout -vru+f5 -p 50 -g 10ms -s 160 "Текущее время ' + str(currentDT.hour) + ' часов и ' + str(currentDT.minute) + ' минут" | aplay -D plughw:1,0'

os.system(time)

def Play\_Current\_Settings():

settings = 'espeak-ng --stdout -vru+f5 -p 50 -g 10ms -s 160 "Текущие настройки ' + ps.get\_setting\_str() + ' " | aplay -D plughw:1,0'

pid = os.system(settings)

def Play\_Hello():

os.system('espeak-ng --stdout -vru+f5 -p 80 -g 20ms -s 160 "Привет" | aplay -D plughw:1,0')

def Play\_HRU():

num = random.randint(0,2)

phrase = 'espeak-ng --stdout -vru+f5 -p 50 -g 10ms -s 160 "' + HRU[num] + '" | aplay -D plughw:1,0'

SoundRecognition.py

import SoundAnalyzer as sa

import ParseSettings as ps

import subprocess

import os

import errno

import signal

FIFO = 'tosound'

sound\_enable = 1

mic\_enable = 1

microphone = 'plughw:2,0'

find = False

microphone\_device\_1 = 'plughw:2,0'

microphone\_device\_0 = 'plughw:1,0'

try:

os.mkfifo(FIFO)

except OSError as oe:

if oe.errno != errno.EEXIST:

raise

def runCommand(command):

print(command)

commandParam = command.split()

if command[0:5] == "sound":

global sound\_enable

sound\_enable = int(commandParam[1])

elif command[0:3] == "mic":

global mic\_enable

mic\_enable = int(commandParam[1])

elif command[0:6] == "opmode":

global microphone

if int(commandParam[1]) == 0:

microphone = microphone\_device\_0

else:

microphone = microphone\_device\_1

elif command == "find":

global find

if not find:

find = True

else:

find = False

def receiveSignal(signalNumber, frame):

print('Received:', signalNumber)

print("Opening FIFO...")

with open(FIFO, "r" ) as fifo:

print("FIFO opened")

for line in fifo:

runCommand(line)

return

def loadSetting():

mic\_enable = int(ps.get\_setting("settings.ini","Settings","Mic"))

sound\_enable = int(ps.get\_setting("settings.ini","Settings","Sound"))

if int(ps.get\_setting("settings.ini","Settings","Operation mode")) == 0:

microphone = microphone\_device\_0

else:

microphone = microphone\_device\_1

ps.update\_setting("settings.ini", "PIDS", "Sound", str(os.getpid()))

signal.signal(signal.SIGUSR1, receiveSignal)

loadSetting()

#dictionary

#/home/<user>/.local/lib/python3.7/site-packages/speech\_recognition/pocketsphinx-data/en-US

while True:

if find:

out = subprocess.Popen(['aplay','-D', microphone\_device\_0 ,'/home/Robo/Sounds/play.wav'],

stderr=subprocess.STDOUT,

shell=False)

out.wait()

continue

if sound\_enable == 0 or mic\_enable == 0:

continue

out = subprocess.Popen(['arecord','-f', 'dat', '-d', '4', '-D', microphone ,'voice.wav'],

stderr=subprocess.STDOUT,

shell=False)

out.wait()

if sa.Google() == False:

print("Try sphinx\n")

sa.Spinx()

os.remove('voice.wav')

**ПРИЛОЖЕНИЕ Б**

***(обязательное)***

**Спецификация программного дипломного проекта**

**ПРИЛОЖЕНИЕ В**

***(обязательное)***

**Ведомость документов**