

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт Радиоэлектроники и информационных технологий
Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы и сети
(наименование)


Кафедра Вычислительные системы и технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра
(бакалавра, магистра, специалиста)

Студента Кузнецова Георгия Дмитриевича группы 16-В-2
(Ф.И.О.)
на тему Программная система телеприсутствия мобильного робота
(наименование темы работы)

СТУДЕНТ:

 Кузнецов Г.Д.
(подпись) (фамилия, и., о.)

(дата)


КОНСУЛЬТАНТЫ:

1. По _____

(подпись) (фамилия, и., о.)

(дата)

РУКОВОДИТЕЛЬ:

 Гай В.Е.
(подпись) (фамилия, и., о.)
02.07.2020
(дата)

2. По _____

(подпись) (фамилия, и., о.)

(дата)

РЕЦЕНЗЕНТ:

(подпись) (фамилия, и., о.)


(дата)

3. По _____

(подпись) (фамилия, и., о.)

(дата)


ЗАВЕДУЮЩИЙ КАФЕДРОЙ

 Жевнерчук Д.В.
(подпись) (фамилия, и.о.)
02.07.2020
(дата)

ВКР защищена _____
(дата)
протокол № _____
с оценкой _____

МИНОБРАЗОВАНИЯ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Кафедра Вычислительных систем и технологии

УТВЕРЖДАЮ
Зав. кафедрой ВСТ
 Жевнерчук Д.В.
«12» мая 2020 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника
(код и наименование)

студенту Кузнецову Георгию Дмитриевичу группы 16-В-2
(Ф.И.О.)

1. Тема ВКР Программная система телеприсутствия мобильного робота
(утверждена приказом по вузу от 15.04.2020 № 867/5)

2. Срок сдачи студентом законченной работы 02.07.2020

3. Исходные данные к работе данные с камеры (видеопоток); данные с микрофона (аудиопоток); язык разработки – C++; система включает программы для операционной системы GNU/Linux.

4. Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих разработке)

Введение

1. Требование к продукту

2. Анализ требований к разрабатываемому продукту

3. Разработка структуры системы

4. Разработка программных средств

5. Тестирование программного обеспечения

Заключение

Список литературы

Приложение

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

1. Структурная схема разрабатываемой системы

2. Структурные схемы работы программных алгоритмов

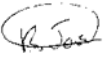
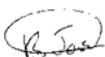
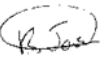
3. Модели обмена данными по сети


4. Изображения тестирования

6. Консультанты по ВКР (с указанием относящихся к ним разделов)

Нормоконтроль Гай В.Е. _____

7. Дата выдачи задания 12.05.2020

Код и содержание Компетенции	Задание	Проектируемый результат	Отметка о выполнении
ПК-1, Способность разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов «человек – электронновычислительная машина»	Разработать структурную схему системы	Структурная схема системы	
ПК-2, Способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	Разработать алгоритмы работы системы, реализовать их на одном из языков программирования с помощью современной среды разработки	Схема алгоритма работы реализована на языке программирования C++, в среде разработки QtCreator	
ПК-3, Способность обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	Выполнить тестирование программ оценить качество работы	Выполнено тестирование программ и оценено качество работы программ	

Руководитель  Гай В.Е.
(подпись)

Задание принял к исполнению 12.05.2020
(дата)

Студент  Кузнецов Г.Д.
(подпись)

**МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)**

АННОТАЦИЯ

к выпускной квалификационной работе

по направлению подготовки (специальности) 09.03.01

Информатика и вычислительная техника

(код и наименование)

студента Кузнецова Георгия Дмитриевича группы 16-B-2
(Ф.И.О.)

по теме Программная система телеприсутствия мобильного робота

Выпускная квалификационная работа выполнена на 129 страницах, содержит 36 рисунков, 0 таблиц, библиографический список из 6 источников, 1 приложение.

Актуальность: Данная работа актуальна, так как телеприсутствие является одной из
необходимых технологий в учебной, военной и промышленной сферах на сегодняшний день.

Объект исследования: Обмен видео и аудио данными, а также данными для манипуляции
роботом.

Предмет исследования: алгоритмы и методы захвата видео и аудио данных, а также их
передачи по сети.

Цель исследования разработка программной системы телеприсутствия мобильного робота.

Задачи исследования: анализ предметной области, разработка алгоритмов, разработка,
прототипа, тестирование.

Методы исследования: теоретический анализ, системный подход, эксперимент.

Структура работы: выпускная квалификационная работа состоит из введения, пяти глав,
заключения, списка литературы и приложения.

Во введении даётся описание проблемы, лежащей в основе данной работы.

В 1 разделе «Требование к продукту». определено назначение разработки, область
применения и технические требования к продукту.

Во 2 разделе «Анализ требований к разрабатываемому продукту» выбирается для
разработки: операционная система, язык программирования, среда разработки, представлен
обзор существующих систем телеприсутствия мобильных роботов, выполняется выбор
подходов к реализации компонентов систем, выбираются протоколы для передачи данных по
сети.

В 3 разделе «Разработка структуры системы» разрабатывается общая структура системы,
разрабатываются методы взаимодействия приложений с сигнальным сервером,
разрабатываются методы передачи команд управления, видео и аудио данных по сети.

В 4 разделе «Разработка программных средств» обзор графического интерфейса
пользователя, представлена разработка программных средств.

В 5 разделе «Тестирование программного обеспечения» поэтапное описание тестирования
готового продукта.


В заключении производятся основные выводы о проделанной работе.

Выводы:

1. Разработана программная система для робота телеприсутствия
2. Проведено тестирование и отладка системы

Рекомендации:





1. Оптимизация исходного кода
2. Будущее развитие программы

 / Кузнецов Г.Д.
подпись студента /расшифровка подписи

«02» июля 2020 г.

Оглавление

Введение.....	5
1. Требование к продукту	6
1.1. Назначение разработки и область применения	6
1.2. Технические требования.....	6
2. Анализ требований к разрабатываемому продукту	7
2.1. Выбор операционной системы.....	7
2.2. Выбор языка программирования	9
2.3. Выбор среды разработки	11
2.4. Обзор существующих систем телеприсутствия роботов	12
2.5. Выбор подходов к реализации компонентов систем	14
2.6. Выбор протоколов для передачи данных по сети.....	15
3. Разработка структуры системы.....	16
3.1. Разработка общей структуры системы.....	16
3.2. Взаимодействие приложений с сигнальным сервером	20
3.3. Передача видео по сети	23
3.4. Передача аудио по сети	25
3.5. Передача сигналов управления по сети	26
4. Разработка программных средств.....	27
5. Тестирование программного обеспечения	31
Заключение.....	41
Список литературы.....	42
Приложение.....	43

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020 (ПЗ)		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Кузнецов Г.Д.		06.07	Программная система телеприсутствия мобильного робота Пояснительная записка	Лит.	Лист
Провер.		Гай В.Е.		06.07			4
						Листов	
Н. контр.		Гай В.Е.		06.07		129	
Утверд.		Жевнерчук Д.В.		06.07		Кафедра «Вычислительные системы и технологии»	

Введение

Задача, на решение которой направлена данная работа – разработка программной системы телеприсутствия мобильного робота для помощи людям с ограниченными возможностями. Система поможет быть в центре событий с помощью удаленного взаимодействия на передвижную платформу и трансляцией видео и аудио сигнала.

В настоящее время уже существуют системы для устройств телеприсутствия. Такие системы разрабатываются для частных закрытых проектов и найти их в свободном доступе практически не представляется возможным. Те же, которые можно найти, не подходят к проекту, над которым выполняется данная работа, так как конечное программное решение должно запускать уникальные скрипты или программы на подвижной платформе.

Данный проект разрабатывался с нуля и в нем необходимо реализовать собственную систему.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						5
Изм	Лист	№ докум.	Подп.	Дата		

1. Требование к продукту

1.1. Назначение разработки и область применения

Данный проект нацелен в основном для узкой аудитории людей, которые по каким-либо причинам не могут посетить такие общественные места как: школы, институты, музеи и многие другие заведения. Главной целью которой является получение информации при помощи зрительного и слухового восприятия, а также где присутствует необходимость в перемещение по объекту, например разные аудитории в институте.

1.2. Технические требования

Требование к разрабатываемому программному обеспечению состоит из следующих пунктов:

- Разработать одно общее программное обеспечение, которое может работать в одном из режимов, режим робота телеприсутствия или клиентский режим.
- Приложения должны обмениваться видеопотоком с веб-камеры через локальную сеть или сеть интернет.
- Приложения должны обмениваться аудиопотоком через локальную сеть или сеть интернет.
- Клиентское программное обеспечение должно передавать желаемые команды управления для перемещения мобильного робота по требованию пользователя, при помощи мыши или клавиатуры.
- В приложение должна быть добавлена возможность отключения видео камеры или микрофона.
- Реализовать соединение в ручном режиме.
- Разработать сигнальный сервер для упрощения подключения пользователей к программному обеспечению мобильного робота в полуавтоматическом режиме.
- Разработанные приложения обязаны запускаться под дистрибутивами ОС GNU/Linux.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						6
Изм	Лист	№ докум.	Подп.	Дата		

2. Анализ требований к разрабатываемому продукту

2.1. Выбор операционной системы

Сейчас существует достаточно большой выбор среди операционных систем, но самыми популярными на сегодняшний день являются такие системы как Windows 7/8/10, MacOS X и GNU/Linux.

Операционные системы Windows являются коммерческим продуктом. Во всех версиях Windows начиная с Windows 95, имеется графический интерфейс. Система очень дружелюбна к пользователю, в последних версиях появилась даже поддержка автоматической установки драйверов устройств из центра обновления Windows, что упрощает процесс настройки компьютера после развертывания ОС. Поэтому системы этого семейства имеют большое распространение у пользователей домашней вычислительной техники.

Windows 10 – это система, которую разрабатывает крупная корпорация Microsoft. Доля используемых данную ОС систему в качестве основной на своих компьютерах среди обычных пользователей, составляет примерно 50%. Младшие версии, такие как Windows 7 и Windows 8, занимают долю примерно 28% для Windows 7 и 4% для Windows 8. Так как проценты используемых предыдущих версий ОС достаточно велики и учитывая даже тот факт, что Windows 7 на текущий момент не поддерживается Microsoft, то из этого следует вывод, что разработка программы под эти версии Windows имеет место быть.

MacOS X или OS X, это операционная система семейства Unix, которая была разработана компанией Apple Inc. Данная ОС является проприетарной, она разработана и сконфигурирована под оборудование, которое производит сама компания. Эту систему достаточно сложно установить на другие ПК без вмешательства сторонних разработчиков. Количество рядовых пользователей составляет примерно 3%.

Так же можно часто встретить упоминание операционной системы GNU/Linux. Она базируется на основе ядра Linux, входит в состав семейства Unix подобных ОС. Данная система имеет очень хорошее распространение среди системных администраторов. На ее базе строится огромное количество серверов. Большая часть дистрибутивов, таких как Debian, OpenSUSE, Fedora, являются свободными, бесплатно распространяются и могут быть установлены любым желающим. Но так как после установки ОС требуется достаточно большое вмешательство в конфигурирование, делает систему не привлекательной для

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						7
Изм.	Лист	№ докум.	Подп.	Дата		

обычного пользователя. Также эти системы славятся высокой безопасностью, при правильной настройке. Особенно если использовать совместно с системой принудительного контроля доступа SE Linux. Но учитывая, что рядовые пользователи не могут заставить работать стабильно дистрибутивы, а большинство ПО таких как: MS Office, Photoshop и многие другие невозможно установить, система используется лишь 2-мя % от общей массы пользователей [4].

Данное программное обеспечение разрабатывается для робота телеприсутствия “ElcBot”. В качестве оборудования, которое используется в прототипе это одноплатный компьютер «raspberry pi 3 model B». На нем в свою очередь возможно развернуть только одну полноценную ОС, это GNU/Linux дистрибутив “Ubuntu server 18.04” с установленным x-org. Исходя из этого была выбрана данная ОС для разработки программного обеспечения.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						8
Изм	Лист	№ докум.	Подп.	Дата		

2.2. Выбор языка программирования

Существует огромное количество языков программирования. самыми популярными можно назвать: C++, Python, Java, а также Swift для MacOS. Рассмотрим языки с конца.

Swift – данный язык годиться только для разработки под ОС от компании Apple. Этот язык хорошо документирован, его очень легко поддерживать, в нем хорошо организована работа с памятью. В связи с тем, что разработка ПО под Apple технику не планируется, то он в этом случае не интересен.

Java – использует виртуальную машину для запуска написанных на ней приложений. Этот инструмент очень мощный, на нем можно реализовать практически любые задачи. Одним из наиболее важных плюсов можно отметить кроссплатформенность написанных приложений. С другой стороны, виртуальная машина java очень затратна по производительности, сильно нагружает центральный процессор и использует большое количество оперативной памяти. Так как ресурсов на одноплатном компьютере достаточно мало, то данный язык не подходит, для реализации проекта.

Python – очень мощный инструмент, присутствует большое количество библиотек. Очень важным моментов является простота разработки ПО и кроссплатформенность написанных программ. Данный язык является интерпретируемым, он имеет хорошую поддержку со стороны разработчиков. В качестве недостатков, особенно стоит отметить, что все приложения, которые были написаны на данном языке выполняются достаточно медленно, а написанный код, проблематично поддерживать из-за синтаксиса языка. Из-за низкой скорости выполнения команд этот язык не желательно использовать для реализации проекта.

C++ – имеет высокую скорость работы. На нем можно реализовать любую задачу, большое количество приложений были написаны именно на этом языке. Так как язык очень древний, у него обширное сообщество, написано множество библиотек. В C++ возможно управлять более детально памятью. Минусом можно назвать сложность в написании программ, так как у разработчиков большая свобода действий. Также часто возникают ошибки, например, утечки памяти, а неправильная работа с указателями приводит к порче памяти. Приложения, которые разработаны при помощи этого инструмента тяжело поддаются портированию под другие операционные системы. Но при использовании фреймворка Qt и их библиотек, проблем при портирование будет значительно меньше.

Framework Qt будет использоваться для захвата и воспроизведения аудио, отображения пользовательского интерфейса и вывода изображения в этом интерфейсе. Также он будет выполнять обработку нажатий клавиш на клавиатуре.

Дополнительно будет использоваться OpenCV, эта библиотека позволит захватывать изображения с web-камеры.

Исходя из всех преимуществ и недостатков, которые были перечислены выше, из-за малого количества ресурсов в одноплатном компьютере, выбор был сделан в пользу C++. Так как правильно написанная программа, будет использовать меньше ресурсов, и работа приложения будет более эффективной, а компьютер будет менее загруженный.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						10
Изм	Лист	№ докум.	Подп.	Дата		

2.3. Выбор среды разработки

Во время создания проекта будет использоваться интегрированная среда разработки (IDE). Она содержит в своем составе ряд программных средств, которые помогают эффективно вести разработку. IDE помогает сократить время написания программного кода и в дальнейшем помогает быстрее найти ошибки при помощи отладочных средств. Дополнительно часто используемым средством является возможность автоматической сборки программного обеспечения.

Таких сред существует большое количество. Рассмотрим самые популярные для написания кода на языке C++ и которые нативны под GNU/Linux. Так как именно этот язык и эта операционная система будут использоваться для создания проекта.

CLion – одна из самых богатых по функционалу сред разработки. Есть возможность установки сторонних плагинов. Данную IDE разрабатывает компания JetBrains. Программа стоит достаточно дорого, а бесплатную версию можно получить, только в том случае если разрабатываемый проект будет с открытым исходным кодом и не будет использоваться в коммерческих целях. Минусом можно назвать только то, что она потребляет большое количество ресурсов компьютера для своей работы.

Eclipse – IDE является свободным с исходным кодом. Данная среда является легковесной, потребляет малое количество ресурсов компьютера, имеется поддержка нескольких языков программирования (в первую очередь java, второстепенно другие языки, в том числе и C++), работает с популярными компиляторами, в среде разработки есть возможность расширения функционала. Минусами можно назвать сложность в первоначальной настройке среды и ее эксплуатации в повседневности.

QtCreator – нацелен на разработку преимущественно на C++. Он имеет множество интересных возможностей. Его плюсами является присутствие редактора форм QtDesigner, хорошо документирован, есть бесплатная версия, поддерживает огромное количество компиляторов, имеется кроссплатформенная компиляция, а также можно установить дополнения, которые позволят разрабатывать программы на других языках программирования.

Так как при написании программы будут использоваться библиотеки Qt, то было решено использовать QtCreator, мощная, удобная и не затратная по ресурсам IDE.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						11
Изм	Лист	№ докум.	Подп.	Дата		

2.4. Обзор существующих систем телеприсутствия роботов

Рассмотрим две самые популярные системы роботов телеприсутствия. Один от отечественного разработчика, другой от зарубежного.

В России на первом месте зарекомендовала себя компания “BotEyes”. Она производит несколько моделей, одна из самых популярных – робот “BotEyes-Pad”. Для управления роботом используется отдельный компьютер, который интегрирован в шасси. Подключение возможно только по Wi-Fi. Для передачи видео и аудио можно использовать любой свой гаджет. В качестве программного обеспечения для видео и аудиопотока рекомендуется использовать всем известную программу “Skype”, разработку которой ведет корпорация Microsoft. По желанию можно установить любую другую. Для управления движением робота используется проприетарное программное обеспечение, которое может быть установлено пользователем на собственное устройство, а также присутствует возможность управления через web-браузер, без установки программы. Данное ПО разрабатывает сама компания “BotEyes”. Минусом такой системы является требование к устройству, с которого будет выполняться управление. На устройстве должна быть функция разделения экрана, например старые версии iPad с устаревшей ОС не подходит для этих целей.



Рисунок 1. Система телеприсутствия BotEyes-Pad

За рубежом самым популярным производителем роботов телеприсутствия стала китайская компания “PadBot”. На текущий момент в продаже имеется робот “PadBot P1”. Данный девайс оснащен LTE модемом, Wi-Fi модулем и 10 дюймовым HD экраном. Клиентское программное обеспечение интереснее чем у отечественного разработчика. Видео и аудио передача, а также управление выполняется из одного приложения. Установка данного программного обеспечения возможна на все современные смартфоны, но приложение под персональные компьютеры отсутствует. Эта программа также является проприетарной, управление возможно только роботами от этого производителя.



Рисунок 2. Система телеприсутствия PadBot P1



Рисунок 3. Программное обеспечение роботов телеприсутствия PadBot

2.5. Выбор подходов к реализации компонентов систем

Для реализации требуемых задач, можно использовать готовый проект, разрабатываемой компанией Google. Его название WebRTC. Это уже полностью бесплатное свободное решение для обмена потоковыми видео и аудио. Работает этот проект исключительно в web-браузере по технологии точка-точка. Главной проблемой при использовании этого проекта заключается в том, что для корректной работы необходимо использовать современный браузер. Вдобавок, есть возможность блокировки ресурсов, где используется WebRTC, так как в данный момент она считается не безопасной технологией. Блокировку производят браузеры, дополнительно могут блокировать и сторонние плагины. Например, блокировщики рекламы. Кроме того, проблему создаст тот факт, что использование различных версий или типов браузеров может привести к некорректной работе.

Другие же проекты попросту недоступны для собственной интеграции, так как являются проприетарным программным обеспечением. Поэтому было решено разработать свою систему, которая будет представлена в виде приложения для персонального компьютера, без использования web-браузеров, так как отсутствует зависимость от сторонних разработчиков. В дальнейшем можно развить проект еще больше и не задумываться о смене основной технологии, потому что система передачи данных будет своя, а проблемы с интеграцией не возникнут.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						14
Изм	Лист	№ докум.	Подп.	Дата		

2.6. Выбор протоколов для передачи данных по сети

Существует два самых популярных протокола транспортного уровня. В зависимости от требуемых задач, необходимо выбрать либо TCP или UDP протокол для обмена данными. Рассмотрим эти протоколы более подробно:

TCP полное название Transmission Control Protocol – нужен для гарантированной доставки сообщения. Перед тем как осуществлять передачу сообщений между двумя узлами создается соединение. Затем пакеты отправляются на удаленный узел. Если пакет в процессе доставки будет утерян, то он будет отправлен вновь. При получении удаленным узлом пакета, тот в свою очередь отправляет подтверждение, что пакет был получен. Таким образом, достоинство этого протокола заключается в том, что данные будут гарантированно доставлены до удаленного узла и их последовательность во время доставки не будет нарушена. Недостатком же можно назвать медленную передачу данных.

UDP полное название User Datagram Protocol – данный протокол не гарантирует доставку пакетов, они могут быть утеряны во время передачи. Его используют в тех случаях, где потеря пакета не приведет к серьезным проблемам. В отличие от TCP предварительная установка соединения не осуществляется. Узел может отправлять пакеты даже если удаленный узел недоступен. Еще одним важным моментом является то, что пакеты могут доходить до удаленного узла в другом порядке. На базе этого протокола, в приложениях можно делать собственные системы исправления ошибок или не делать их вовсе. Все это делает протокол ненадежным. Главным его плюсом является то, что он очень быстрый, так как ожидание проверки доставки пакета не задерживает передачу данных в целом [2].

В данном проекте используются сразу два протокола: TCP и UDP. Первый используется для установки соединения между двумя узлами и обмена команд управления мобильным роботом, а также для взаимодействия приложений с сигнальным сервером. Здесь очень важно, чтобы пакеты доходили до удаленных узлов без потерь, скорость доставки сообщений не имеет значения. Второй протокол используется для обмена видеопотоком и аудиопотоком, так как потеря пакета не критична, а скорость должна быть высокой. Например, если пакет будет потерян, мы просто потеряем небольшой фрагмент кадра или аудио фрейм, что приведет к небольшому искажению полного кадра или прерывание при прослушивании аудиопотока.

3. Разработка структуры системы

3.1. Разработка общей структуры системы

Для реализации поставленных задач, которые обеспечат эффект присутствия, необходимо создать 2 полноценных приложения, исходя из технических требований.

1. Клиентское программное обеспечение с пользовательским интерфейсом

Один экземпляр запущенного приложения может работать в одном из режимов. Это робот телеприсутствия либо клиентский режим. Основная задача данного приложения, обмен данными по сети, такими как видеопоток, аудиопоток и команды управления, между роботом телеприсутствия и персональным компьютером пользователя. Второстепенная задача, возможность обращения ПО к сигнальному серверу. Это необходимо для упрощения установки связи между программами. В этом случае пользователь, использующий приложение в клиентском режиме, не будет указывать IP адрес и порты, которые необходимы для корректной работы программы. Кроме того, представлена функция позволяющая миновать сигнальный сервер. Тогда указывать всю необходимую информацию для обмена данными по сети будет сам пользователь.

2. Сигнальный сервер

Сигнальный сервер требуется для определения и хранения в своей базе IP адресов и используемых портов у удаленных узлов, где запущено приложение в режиме робота телеприсутствия. А также назначение этим программным обеспечениям уникальных идентификаторов (ID), по которым пользователи, использующие приложения в клиентском режиме, могут получать более удобный доступ к роботам телеприсутствия. Предполагается, что сигнальному серверу будет назначен статический IP адрес, а пользовательские устройства, выполняющие задачу роботов телеприсутствия, могут получать динамические адреса.

Обмен данными в этом проекте осуществляется при помощи программного интерфейса использующий адрес и порт, называемый сокет. Как говорилось ранее, в проекте используется протоколы TCP/IP и UDP/IP. Клиентское программное обеспечение, в зависимости от режима работы, выступает либо сервером при использовании режима робота телеприсутствия, либо клиентом, когда программа работает в клиентском режиме. Обмен данными с сигнальным сервером, выполняется также через сокеты.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						16
Изм.	Лист	№ докум.	Подп.	Дата		

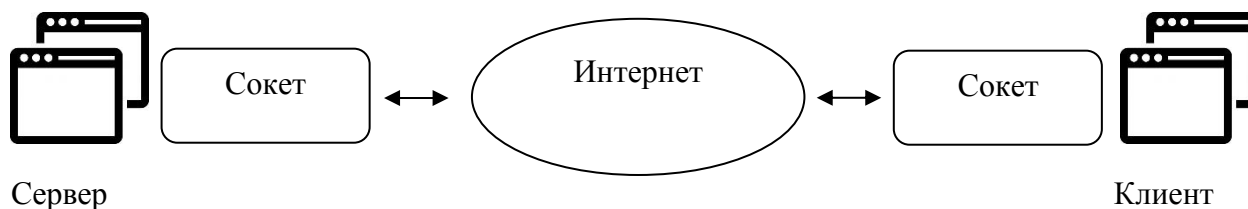


Рисунок 4. Схема обмена данных при помощи сокетов.

В клиентском ПО обмен данными с сервером (мобильный робот), может быть осуществлен одновременно только с одним клиентом (пользователь). Если другой пользователь попытает установить соединение с мобильным роботом, который уже имеет активное соединение, то этот пользователь получит ошибку установки соединения.

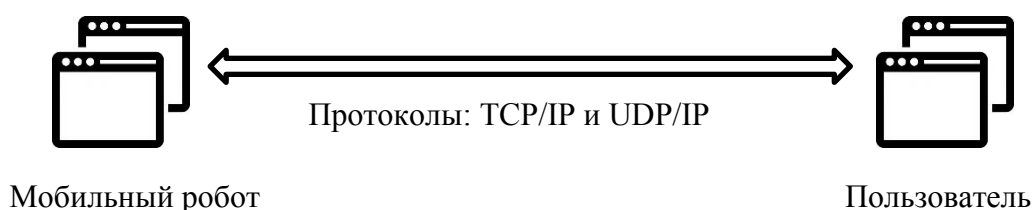


Рисунок 5. Модель связи устройств использующие клиентское ПО.

После инициализации программы, пользователь в настройках приложения может выбрать такой режим, какой он желает использовать на своем устройстве. Затем, в зависимости от его выбора, программа меняет логику работы сетевой части и изменению подвергается графический интерфейс. Рассмотрим отличия более подробно.

При использовании режима робота телеприсутствия, приложение работает как сервер. Если нажать в графическом интерфейсе кнопку «старт», программа перейдет в режим прослушивания сокетов и будет ожидать входящие подключения до тех пор, пока первый клиент не подключится к нему или пользователь не завершит работу самой программы. В клиентском режиме вместо кнопки «старт», будет доступна кнопка «подключение», по нажатию которой, приложение попытается выполнить подключение к серверу через TCP сокет с указанным в настройках IP адреса и порта сервера. При помощи этого сокета определяется доступность устройства, выполняется обмен служебной информацией и идет передача команд управления мобильный роботом. Если канал связи будет налажен, то тогда оба приложения начнут обмен видеопотоком и аудиопотоком при помощи датаграммных сокетов. Если клиент решит разорвать соединение, то сервер опять перейдет в режим прослушивания сокета, а клиент сможет повторно установить соединение с этим же устройством либо с другим. Предусмотрена возможность разрыва соединения со стороны

сервера. После разрыва, программа не будет слушать сокет на входящие соединения до тех пор, пока не нажмут кнопку «Старт». Дополнительно при использовании клиентского режима, в графическом интерфейсе пользователя, появятся кнопки управления движением мобильным роботом и равным образом начнут обрабатываться зарезервированные за кнопками горячие клавиши.

Принцип работы сетевой части программного обеспечения без сигнального сервера, в режимах работа телеприсутствия и клиента, представлен на рисунках 6 и 7.

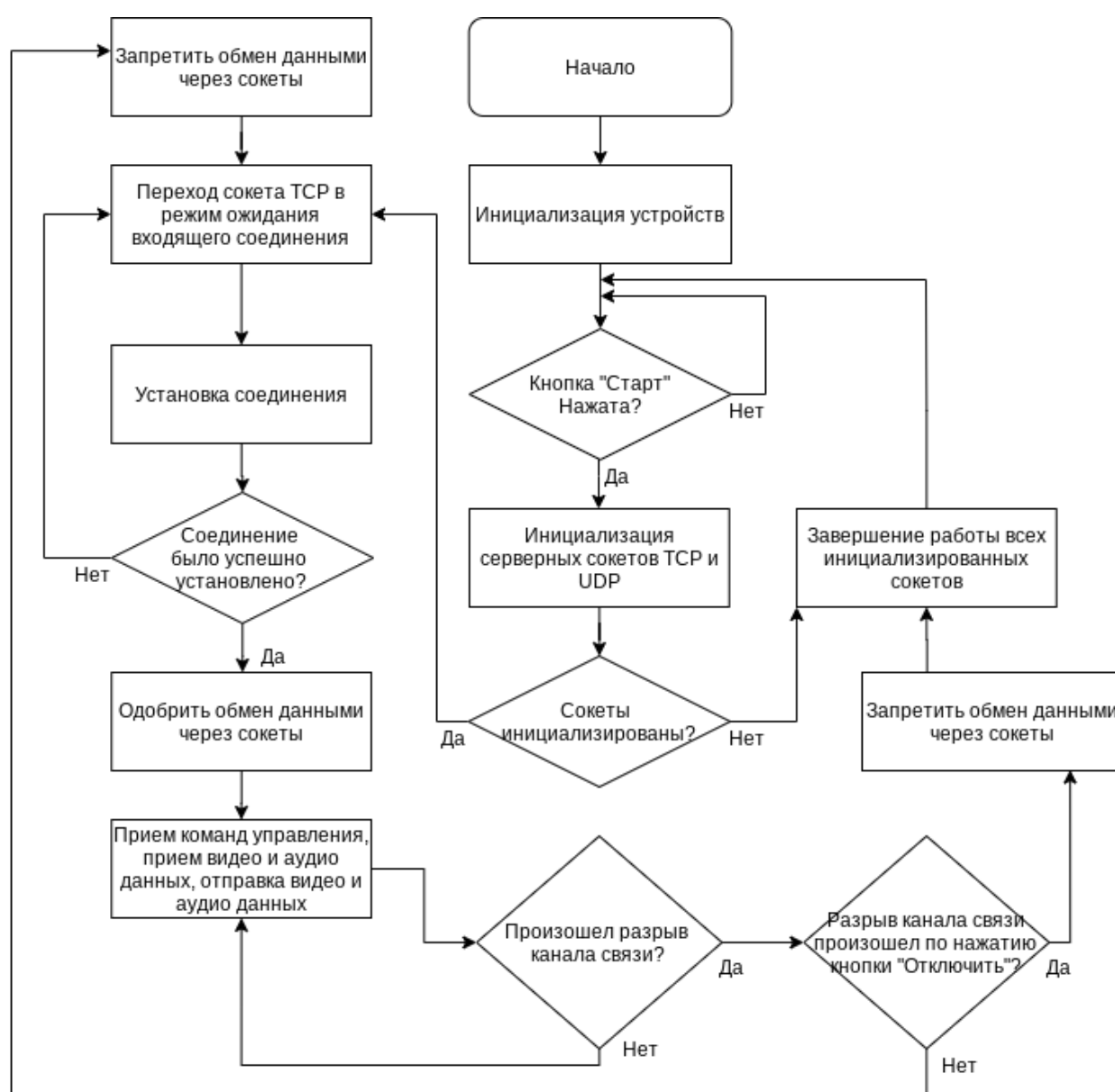


Рисунок 6. Алгоритм серверной части клиентского программного обеспечения в режиме работа телеприсутствия.

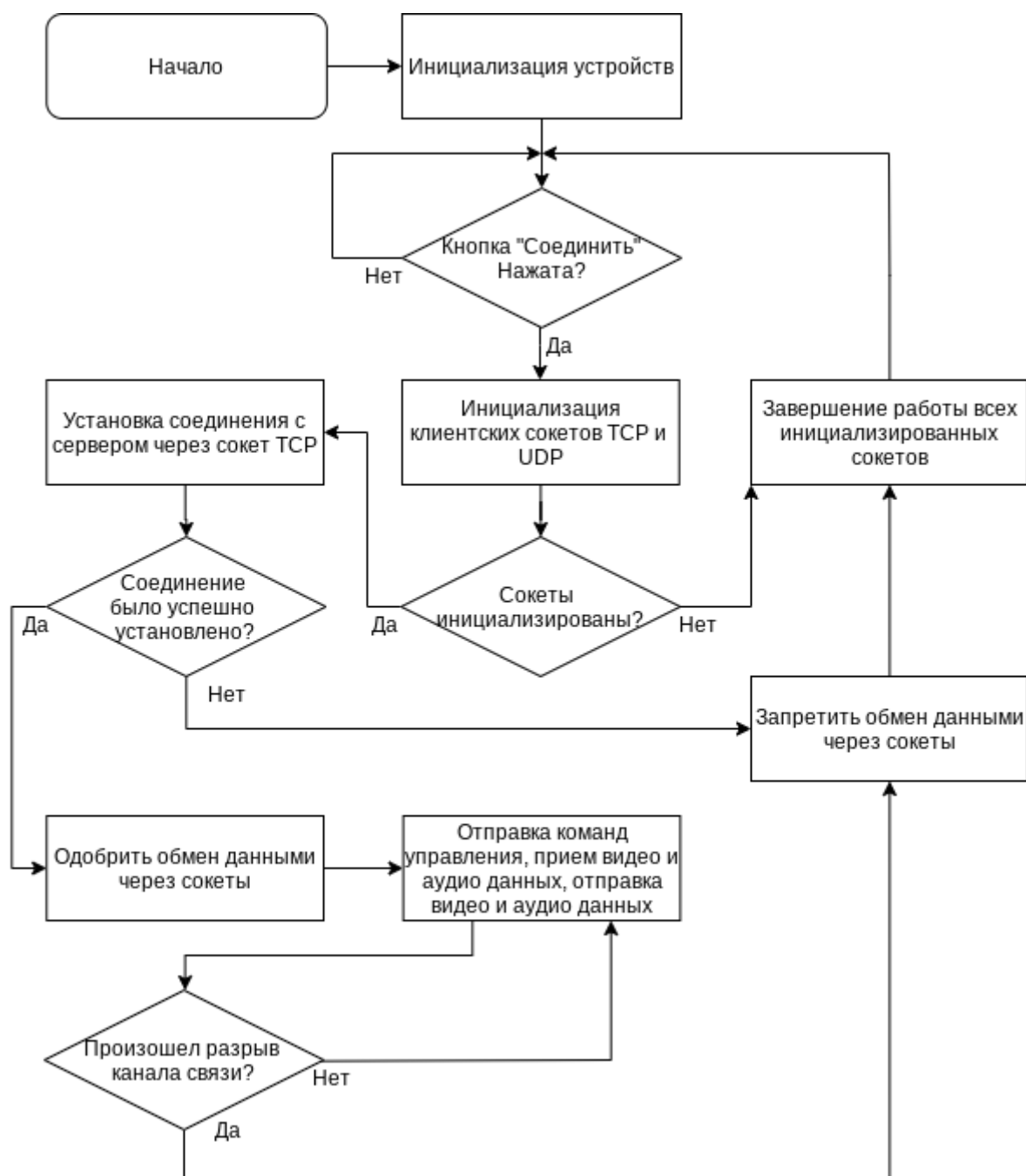


Рисунок 7. Алгоритм сетевой части клиентского программного обеспечения в клиентском режиме.

3.2. Взаимодействие приложений с сигнальным сервером

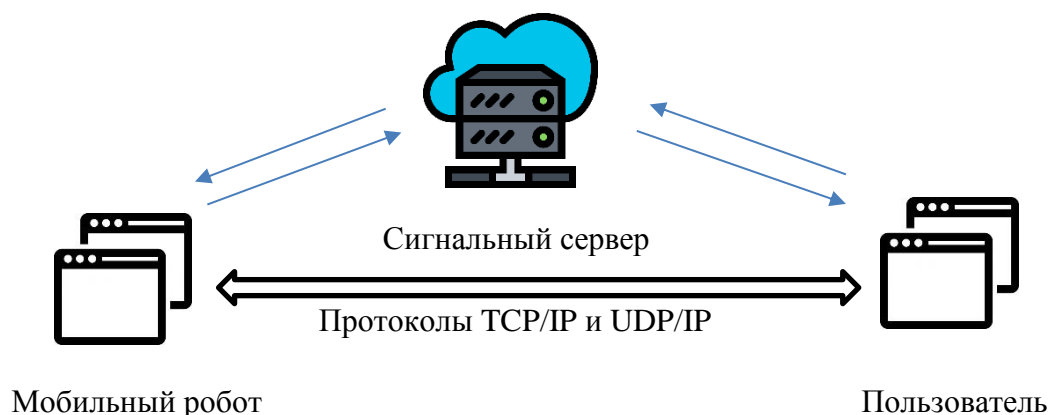


Рисунок 8. Модель связи устройств использующие клиентское ПО совместно с сигнальным сервером.

Клиентское программное обеспечение отправляет на сигнальный сервер запросы в зависимости от выбранного пользователем режима работы программы. Клиентское приложение, работающее в режиме сервера, обрабатывается сигнальным сервером следующим образом:

1. Сервер сообщает свои используемые порты для трансляции видео/аудио/команд управления.
2. Сигнальный сервер записывает полученные данные в свою временную базу хранения данных, случайным образом назначает ID и отправляет его обратно на сервер
3. Сервер получает ID и начинает процедуру прослушивания сокетов, выводит на дисплей ID, по которому пользователь может подключиться к данному устройству.

Данные в пакете, отправляемые с сервера на сигнальный сервер, имеют следующий вид:

0xD0	"9758"	0xEF	"9759"	0xEF	"9758"	0xEF
------	--------	------	--------	------	--------	------

Рисунок 9. Пример данных в пакете, отправляемые на сигнальный сервер с мобильного робота.

Первый байт имеет значение 0xD0, он означает, что далее следуют байты, которые содержат сведения о портах в виде ASCII кодов, разделяются коды байтом 0xEF. Первым идет порт, предназначенный для видео, второй для аудио, последний порт для команд управления. После того как сигнальный сервер получит сообщение от клиентского

программного обеспечения в режиме робота телеприсутствия, он регистрирует его в своей базе и выполнит назначение ID, затем сигнальный сервер отправит обратный пакет, в котором будет содержаться назначенный случайно сгенерированный ID. Содержимое данных в пакете будет выглядеть следующим образом:

0xD0	"000000"
------	----------

Рисунок 10. Пример данных в пакете, отправляемые с сигнального сервера на мобильный робот.

Первый байт имеет код 0xD0 затем в виде ASCII кодов, следует сам ID. После получения пакета от сигнального сервера с ID, программа, работающая в режиме робота телеприсутствия отобразить в окне приложения уникальный идентификатор.

Когда пользователь пытается подключиться к серверу, тот должен узнать его IP и номера портов для входящий соединений. Чтобы упростить задачу, он может установить соединение с программным обеспечением в режиме робота телеприсутствия используя заведомо известный ID. После ввода уникального идентификатора, клиентское ПО в пользовательском режиме обращается к сигнальному серверу. Сигнальный сервер ищет в своей базе зарегистрированное устройство с указанным пользователем ID, затем возвращает IP и порты сервера. Если такого ID не будет в базе, сигнальный сервер вернет ошибку.

Данные в пакете, отправляемые от клиента на сигнальный сервер, выглядят следующим образом:

0xD1	"000000"
------	----------

Рисунок 11. Пример данных в пакете, отправляемые на сигнальный сервер от пользователя.

При запросе первый байт имеет код 0xD1, затем следует введенный ID от пользователя в виде ASCII кодов.

0xD1	"16777343"	0xEF	"9758"	0xEF	"9759"	0xEF	"9758"	0xEF
------	------------	------	--------	------	--------	------	--------	------

Рисунок 12. Пример данных в пакете, отправляемые с сигнального сервера к пользователю.

В обратном ответе первый байт также имеет код 0xD1. Затем следуют: IP записанный целым числом, порты для видео, аудио и команд управления, все они представлены в ASCII и разделяются кодом 0xEF. Адрес, записанный в виде «127.0.0.1» будет представлен как «16777343». В двоичной системе счисления такое число имеет вид «1 0000 0000 0000 0000 0111 1111», затем разбиваем с конца по 8 бит это и будут октеты IP адреса. Если в базе информация не будет найдена или произойдет непредвиденная ошибка, то сигнальный сервер вернет 1 байт с кодом 0xD2. По этому коду, клиентское программное обеспечение в режиме клиента прекратит попытки установки соединения.

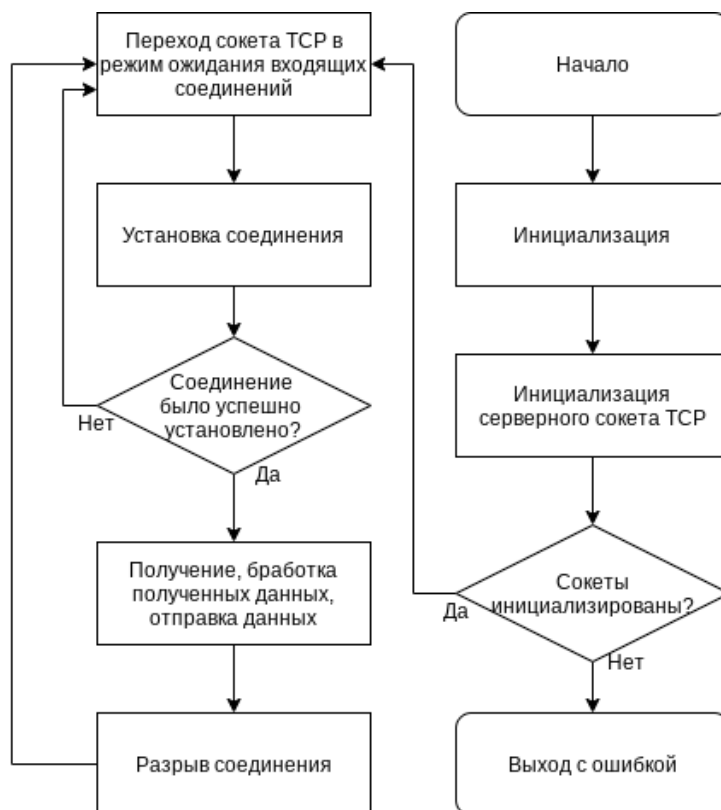


Рисунок 13. Алгоритм сетевой части сигнального сервера.

Если в базе данных при регистрации устройства уже будет существовать запись с тем же IP, то данные из таблицы удалены не будут, а порты и ID будут обновлены.

3.3. Передача видео по сети

Полученный кадр с веб-камеры представляется в виде матриц с несколькими каналами: красного, зеленого и синего цвета, наложение каналов делает изображение цветным. Диапазон значений одного пикселя для каждого отдельного канала варьируется от 0x00 до 0xFF.

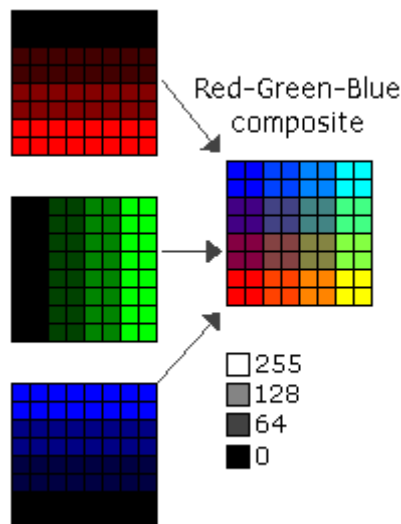


Рисунок 14. Представление матриц цветного изображения.

Чтобы отправить кадр по сети, необходимо его нарезать на строки фиксированной длины, количество этих строк зависит от размера передаваемого пакета. Также необходимо определить ширину и высоту кадра, каждые строки при нарезки получают свой номер пакета. Когда приходит другой кадр, нумерация строк начинается с нуля.

Пакеты с содержимым фрагмента видео кадра имеют следующую структуру:



Рисунок 15. Пример данных в пакете, с фрагментом изображения.

Первые два байта в пакете необходимы для определения номера строки в кадре, затем следуют два байта о высоте кадра, два байта ширины и сами пиксели (по 3 байта), пиксель кодируется 3 байтами информации, так как они цветные. Первый байт в пикселе градация красного цвета, затем байт зеленого и байт синего цвета. В пакет дейтаграммы, данные пикселей записываются до тех пор, пока не закончится установленный по умолчанию размер пакета в байтах.

После того как данные будут сформированы, пакет отправляется на удаленный узел. Количество отправляемых пакетов на 1 кадр зависит от высоты и ширины передаваемого кадра.

При получении пакета удаленным узлом, клиентская программа парсит данные, определяет размер кадра, номер отправляемого пакета и сами данные пикселей. В свою очередь, эти данные накладываются на уже имеющийся у удаленного узла кадр. Разворачиваются они поверх предыдущих данных кадра, учитывается перенос пикселя на новую строку за счет размеров входящего кадра. Также идет учет предыдущего полученного номера строки, и если новый пакет будет иметь номер строки ниже или равный предыдущему, то программа отправит кадр в видеоплеер.

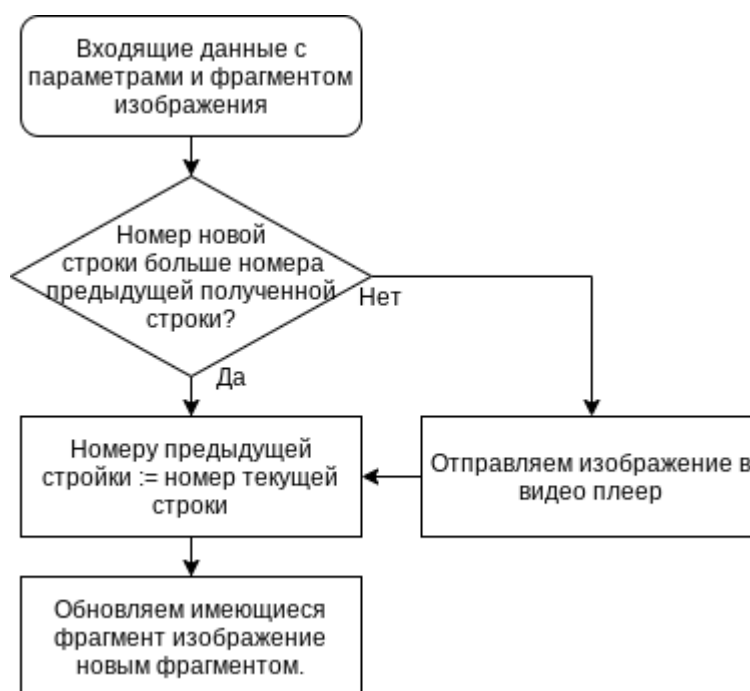


Рисунок 16. Алгоритм обновления полученных фрагментов изображения.

Если пакет каким-либо образом будет утерян при доставке, то программа при восстановлении пропустит недостающую область, смещение изображения не произойдет. Так как при восстановлении учитывается ширина, высота, номер пакета и размер доставляемых пикселей (он фиксирован). Например, нам известно, что в одном пакете может быть 5 пикселей, ширина и высота полного изображения 5x5, если номер строки 0, то запись в матрице начнется с 0-го пикселя, пиксели из другого пакета с номером строки 2, будут записаны с 10 пикселя в полном изображении и так далее.

Количество доставляемых пикселей * номер пакета = начало записи пикселей в матрицу полного изображения.

3.4. Передача аудио по сети

Данные аудио потока упаковываются в пакет как есть, так как они уже представлены в виде байт, упорядочивать их не имеет смысла, важный момент при упаковке данных для отправки на удаленный узел, это определить размер одного пакета, размер одного пакета зависит от частоты дискретизации, от количества каналов (используется всегда один) и продолжительности одного аудио отрывка. Аудио фрагмент был выбран размером в 10ms, при помощи отладчика определил размер аудио данных продолжительностью 10ms и частотой дискретизации 48000 Гц. Один отрывок занимает примерно 3000 байт, после чего было решено использовать размер одного пакета 3129 байт

При получении данных в пакете, приложение записывается полученный аудио фрагмент в временный буфер в той же последовательности, как и получил, затем отправляет их в систему аудио захвата, там в свою очередь идет воспроизведение отрывка.



Рисунок 17. Алгоритмы отправки и приема аудио фрагментов

3.5. Передача сигналов управления по сети

Команды необходимые для управления роботом, приходят от клиента по его требованию, тогда, когда сам пользователь пожелает. Передаваемые данные упаковываются 3-мя байтами.



Рисунок 19. Пример данных в пакете, с командой движения вперед.

Команды кодируются следующим способом: первый байт должен быть кодом 0xF0, второй байт 0xEE

0xF0 — сообщает серверу, что клиент не ожидает обратной связи от сервера, это необходимо на будущее, возможно появятся новые функции.

0xEE — сообщает серверу, что дальше следует 1 байт информации, который отвечает за поведение робота

Последний байт, говорит о конкретной команде, он может принимать следующие значения:

0xA0 — движение вперед

0xA1 — остановка движения

0xA2 — движение назад

0xA3 — поворот налево

0xA4 — поворот направо

Сервер после получения пакета, читает первые два байта и, если они соответствуют всем необходимым требованиям, выводит сообщения в отладочную консоль приложения.

4. Разработка программных средств

В клиентском программном обеспечении взаимодействие с приложением осуществляется при помощи пользовательского интерфейса. После инициализации программы, пользователям отображается окно главного меню, в котором он проводит большинство своего времени. По нажатию кнопки «настройки» появляется форма с параметрами, в которой пользователь может изменять конфигурацию, если в этом будет необходимость. Например, поменять режим работы программного обеспечения (режим клиента или робота телеприсутствия), а также можно изменить IP адрес и порт сигнального сервера или перевести программу в ручное подключение к удаленному узлу.



Рисунок 20. Вид главного меню пользовательского ПО, в разных режимах.

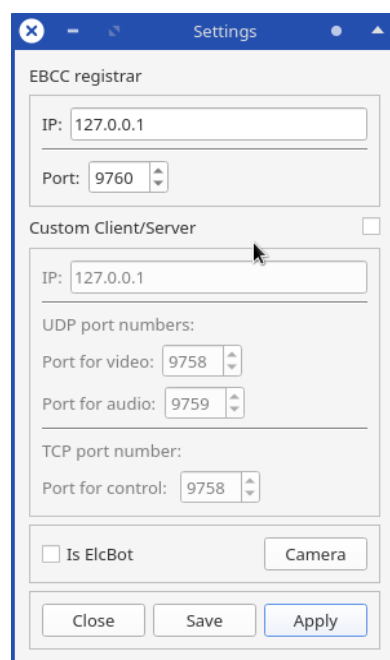


Рисунок 20. Вид формы настроек пользовательского ПО.

Внешний вид главного меню программы меняется незначительно. Изменению подвергается кнопка соединения и в зависимости от режима отсутствует или присутствует

форма с кнопками для удаленного управления. Виды форм описаны в XML файлах «mainwindow.ui», для главного меню и «formsettings.ui», для формы настроек. Эти файлы были сгенерированы автоматически в Qt Design. Инициализация главного окна, обработка событий главного окна (например нажатие кнопок), инициализация формы настроек, отображение двух плееров (локального и удаленного) описаны в исходном файле «mainwindow.cpp» и его заголовочном «mainwindow.h». Назначение основных параметров влияющие на сетевую работу программы, выполняется в исходнике «formsettings.cpp» и заголовочном файле «formsettings.h», а применение выполняется в «mainwindow.cpp». Такие файлы как «videocapture.cpp/videocapture.h», «audiomanager.cpp/audiomanager.h», «controlmanager.cpp/controlmanager.h» и «networktransmission.cpp/networktransmission.h» играют наиболее важную роль в работе программы [3].

В файлах «videocapture.cpp/videocapture.h» описан класс «VideoCapture», он отвечает за инициализацию веб-камеры, захват изображений, передачу изображений в «MainWindow» и систему обмена данными [6]. Так как захват кадра выполняется циклически, то объект класса помещаются в отдельный поток «QThread». Поток запускается по нажатию пользователем кнопки «Старт камеры» и завершается по нажатию «Стоп камера».

Файлы «audiomanager.cpp/audiomanager.h» нужны для захвата аудио фреймов с микрофона и вывода полученных аудио фрагментов на устройства воспроизведения. Объект класса «AudioManager» также выполняются в отдельном потоке. Захват звука начинается сразу после установки соединения с другим устройством. Воспроизведение аудио фрагментов отключить нельзя, только средствами предоставленные операционной системой. Запретить захват звука с микрофона, а также разрешить пользователь может при помощи кнопки «Выключить звук/Включить звук».

Файлы «controlmanager.cpp/controlmanager.h» предназначены для передачи команды пользователя для управление мобильным роботом в систему обмена данными. А также выполняет вывод полученных данных в отладочную консоль от системы обмена данными. В отладочной консоли при получении команды можно посмотреть требуемое действие [5].

Система обмена данными по сети описана в файлах «networktransmission.cpp» и его заголовочном «networktransmission.h». Там можно найти два класса.

Класс «NetworkTransmission» выполняет инициализацию сокетов, в зависимости от выбора пользователем режима требует создать либо серверный сокет, либо клиентский.

Кроме установки соединения, методы в классе обрабатывают входящие данные от «VideoCapture», «AudioManager», также обрабатывает команды от «ControlManager». После вызова функции «startsystemnetworktransmission()», система выполняет соединение при помощи сигнального сервера или без него, проверяет успешность подключения и дает разрешение на обмен данными между двумя узлами. Если соединение не будет установлено, то данные, которые подготовлены к отправке - будут удалены. Функция «sendimage(cv::Mat)» после получения кадра от «VideoCapture» подготавливает данные для пакетов и отправляет сформированные через сокет. Функция «sendaudio(QBuffer *)» переписывает данные из QBuffer в массив байт, затем очищает данные в QBuffer, после чего отправляет массив байт через сокет. «sendcommandcontrol(int)» также формирует отправляемое сообщение для удаленного управления мобильным роботом, затем отправляет его. Функция «stoppingtheconnection()» запрещает обмен данных, закрывает установленное соединение (если такое было) и возвращает систему обмена данных в первоначальный вид.

Класс «NetworkRecv» необходим для ожидания входящих подключений и получения данных через сокет. Объект этого класса выполняется в отдельном потоке. Функция «acceptsock(int, int, int, struct sockaddr_in)» необходима для ожидания входящих подключений, если приложение работает в режиме робота телеприсутствия. Функция «connectsock(int sock, struct sockaddr_in servaddrforcontrol)» наоборот предназначена для подключения к клиентскому приложению в режиме сервера. Функции «recvimage()», «recvaudio()», «recvcontrol()» и «recvregistrardata()» выполняют прием данных через сокет и отправляет их в «NetworkTransmission». Видео кадр будет отправлен в плеер, который описан в «MainWindow.cpp», аудио в объект класса «AudioManager» и команды управления в «ControlManager». Все функции вызываются из объекта класса «NetworkTransmission» [1].

Обработка событий и обмен параметрами между экземплярами классов осуществляются при помощи сигналов и слотов.

Сигнальный сервер не имеет графический интерфейс, работает как консольное приложение. В структуре проекта можно заметить исходные файлы «**database.cpp**»,

«networktransmission.cpp», «datacommunication.cpp», «dataprocessign.cpp» и их заголовочные файлы.

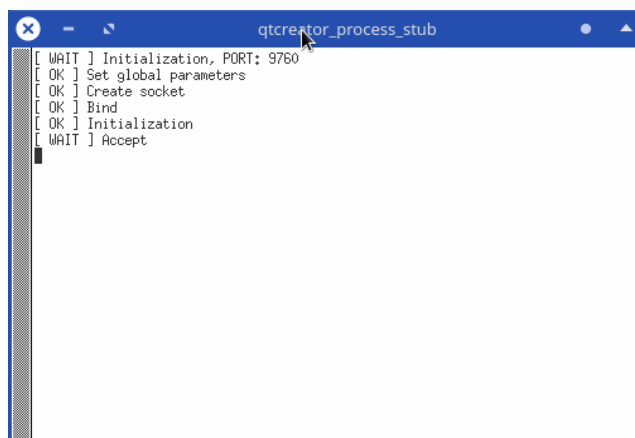


Рисунок 21. Вид консольного приложения сигнального сервера.

В файле «database.cpp/database.h» описана база данных и управление ею. В списке availableElcBotDevice находятся все зарегистрированные устройства с запущенным клиентским приложением в режиме работы телеприсутствия. В списке availableClientDevice записываются все устройства, которые обращались к сигнальному серверу.

```
struct device {
    std::string ip;
    std::string id;

    bool isLock;
};

struct elcbotdevice : public device {
    unsigned int portForVideo;
    unsigned int portForAudio;
    unsigned int portForControl;
};

std::vector<elcbotdevice *> availableElcBotDevice;
std::vector<device *> availableClientDevice;
```

Рисунок 22. Структура базы данных сигнального сервера.

Файлы «networktransmission.cpp/networktransmission.h» предназначены для инициализации сокетов, циклического ожидания клиентов и вызова функций получения данных их обработки и отправки. «datacommunication.cpp» предназначен для получения и отправки сообщений через сокет. «dataprocessign.cpp» выполняет парсинг данных, управление базой данных и формирование данных для отправляемого пакета.

5. Тестирование программного обеспечения

Тестирование проходило в несколько этапов:

- 1) Подключение к удаленному узлу без сигнального сервера
- 2) Подключение к удаленному узлу с сигнальным сервером
- 3) Проверка видео и аудио сигналов
- 4) Проверка доставки сообщений с клиента на серверное ПО

5.1. Подключение к удаленному узлу без сигнального сервера

В данном пункте было проведено множество экспериментов, перечислю одни из самых важных:

- 1) Попытка подключиться к серверному ПО без его запуска

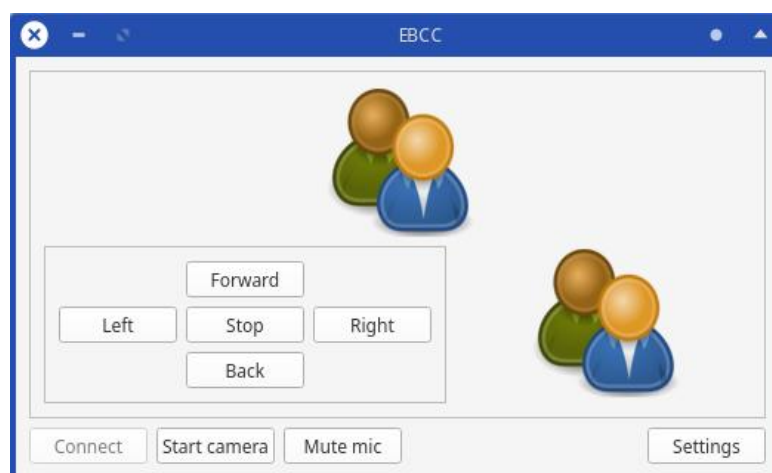


Рисунок 23. Тестирование приложения

Как и ожидалось приложение не установило соединение, через некоторое время кнопка стала доступной, для повторной попытки

- 2) Попытка подключиться к серверному ПО после его запуска

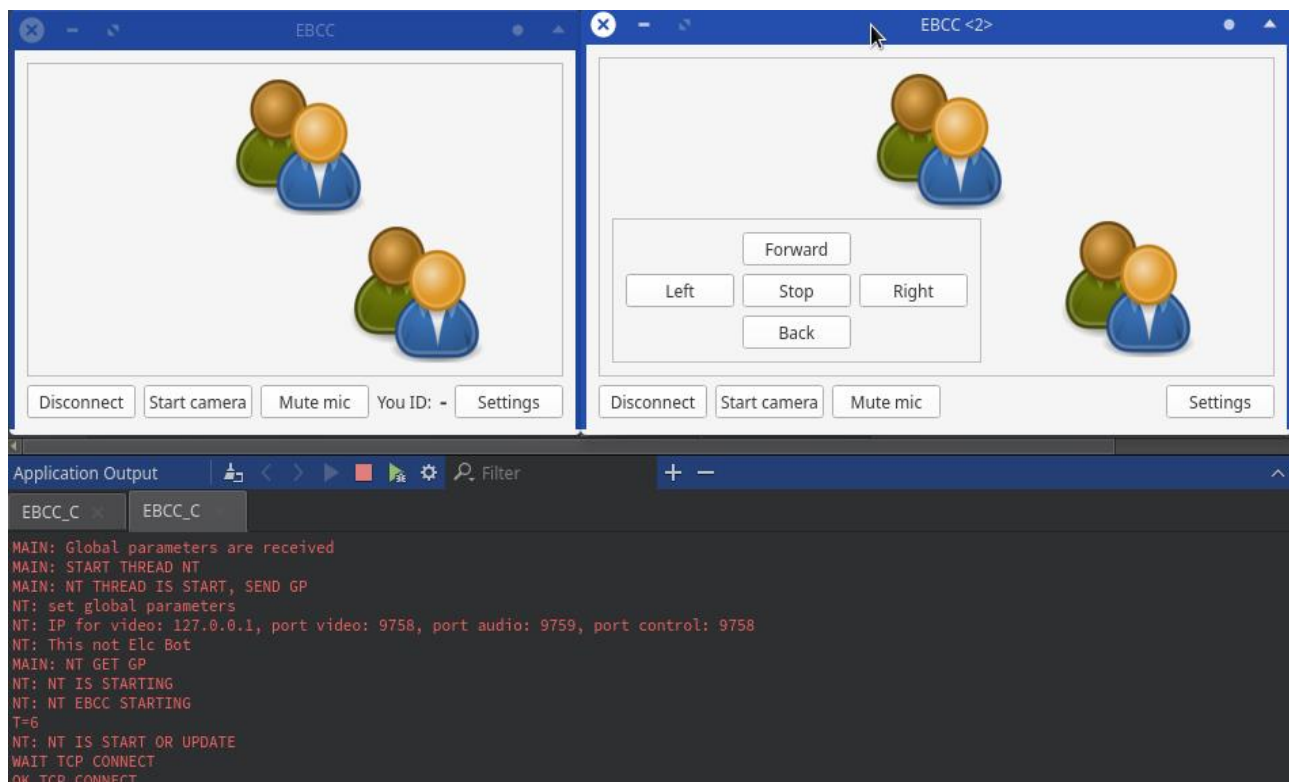


Рисунок 24. Тестирование приложения

Как видим, из рисунка выше, соединение прошло успешно, приложения перешли в режим обмена данных между собой. После закрытия клиентского приложения, серверное начинает слушать новые запросы, поэтому после разрыва соединения, можно повторить попытку подключения.

Если попробовать отключить серверное программное обеспечение, то клиент какое-то время будет обращаться к серверу, но через некоторое время завершит сеанс.

5.2. Подключение к удаленному узлу с сигнальным сервером

Для этого теста добавляется сигнальный сервер, приложения должны обращаться к сигнальному серверу, после чего применять полученные параметры и действовать также как соединение без сигнального сервера.

Пример запуска сигнального сервера:

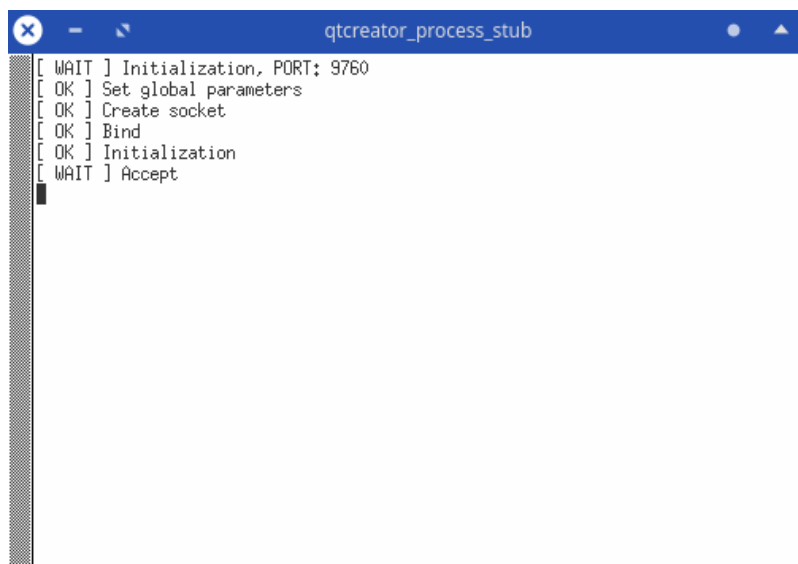


Рисунок 25. Запуск сигнального сервера

Подключение серверного ПО к сигнальному серверу:

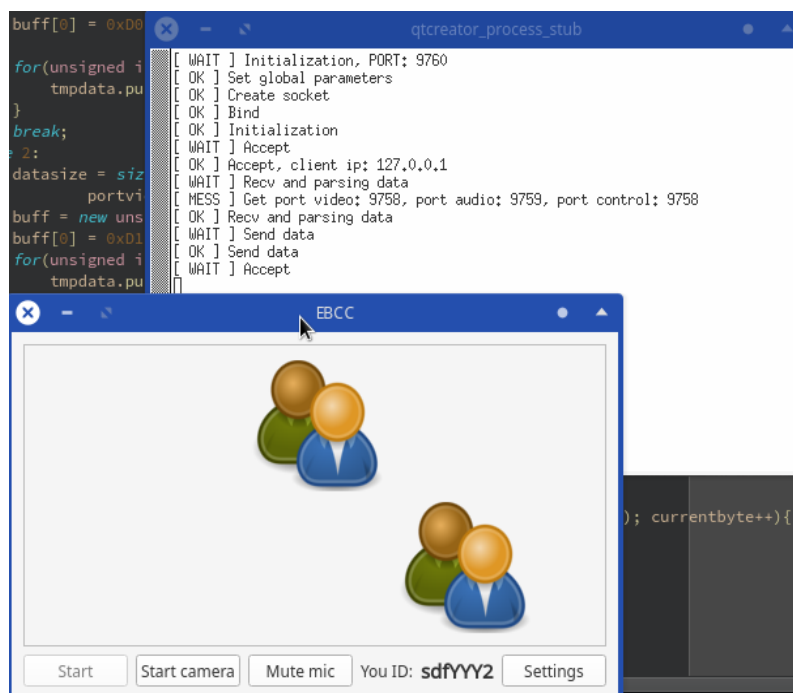


Рисунок 26. Тестирование приложения

Можно заметить, что сигнальный сервер получил данные о клиенте, также сигнальный сервер назначил ID, его можно видеть в окне серверного приложения для робота, сигнальный сервер обработал команду и ждет новые сообщения от других приложений.

Подключение клиентского ПО к сигнальному серверу и получение данных по его запросу.

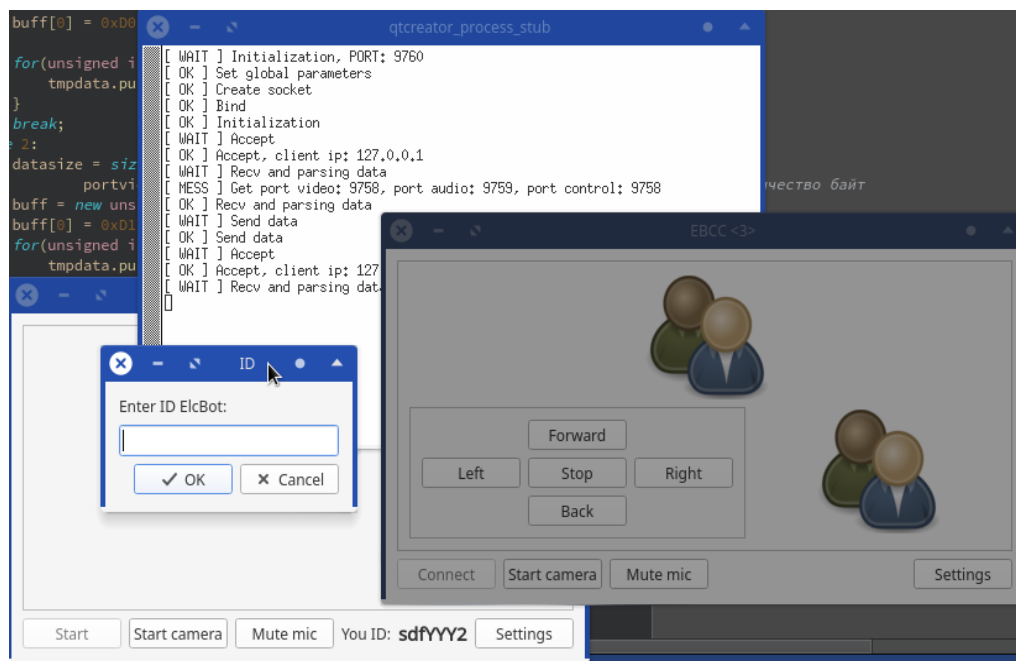


Рисунок 27. Тестирование приложения

Из изображения выше видно, что клиентское ПО подключилось к сигнальному серверу, и прежде, чем отправить пакет, требует ввести требуемый ID, также можно заметить, что серверное ПО для работа еще ожидает входящие подключения.

Попробуем ввести неправильный ID

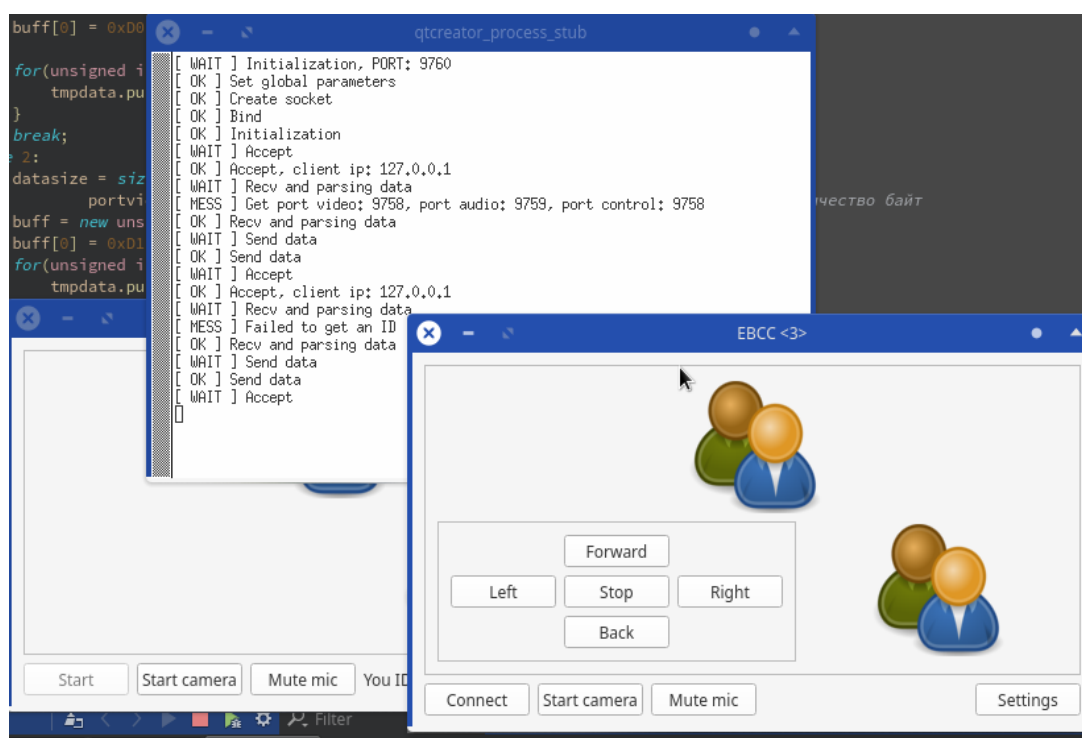


Рисунок 28. Тестирование приложения

Сигнальный сервер не нашел устройство с требуемым ID и отказал в соединении.

Попытка подключиться с правильным ID

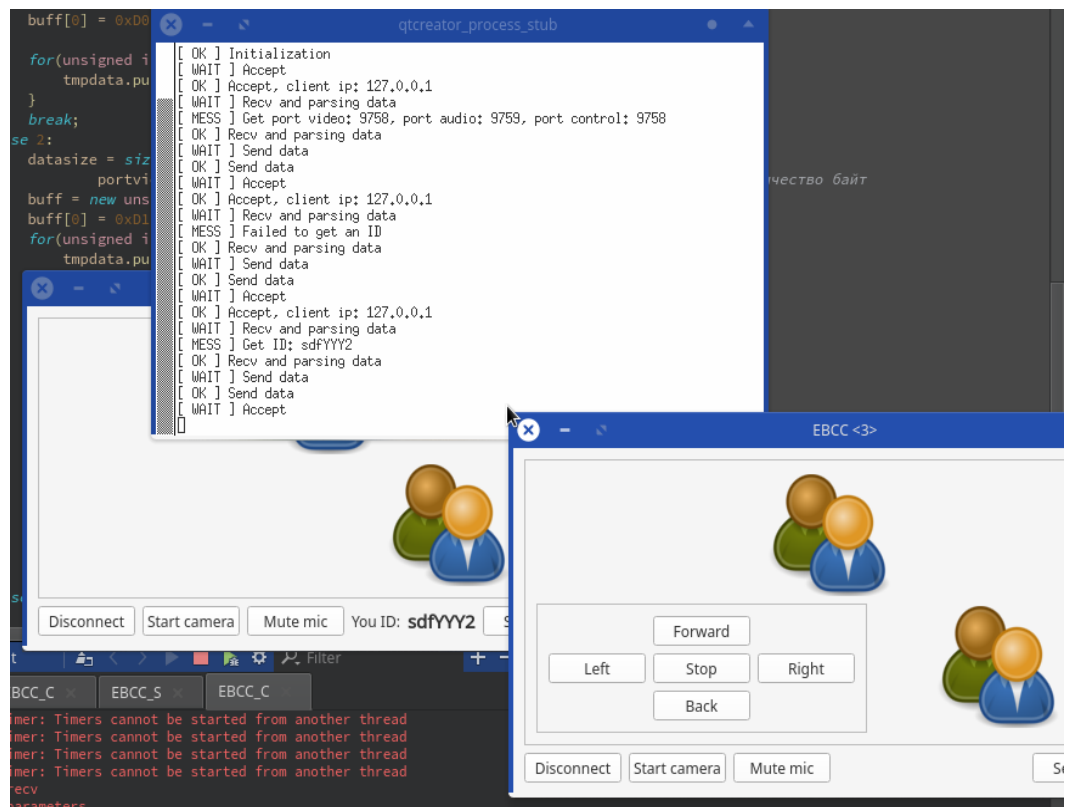


Рисунок 29. Тестирование приложения

Из изображения выше видно, что сигнальный сервер успешно обработал ID, вернул параметры клиенту, тот в свою очередь обратился к серверу для работы

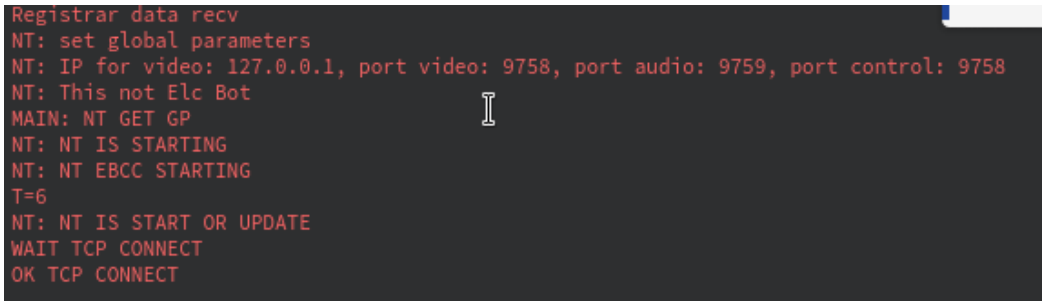


Рисунок 30. Тестирование приложения, логи

Завершение работы сигнального сервера, клиентом и серверным ПО для обрабатываются несколько секунд, после чего приложения автоматически закрываются.

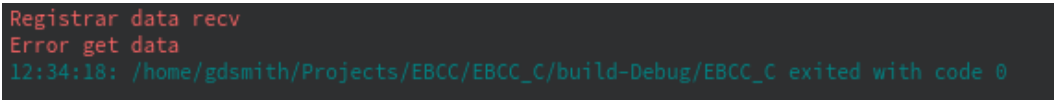


Рисунок 31. Тестирование приложения, логи

5.3. Проверка видео и аудио сигналов

Для проверки видео трансляции, используется камера подключенная к ПК, так как она одна, для начала посмотрим, как работает на одной машине, затем посмотрим работу через локальную сеть.

Трансляция видео сигнала от серверного ПО для работа на клиентское ПО:

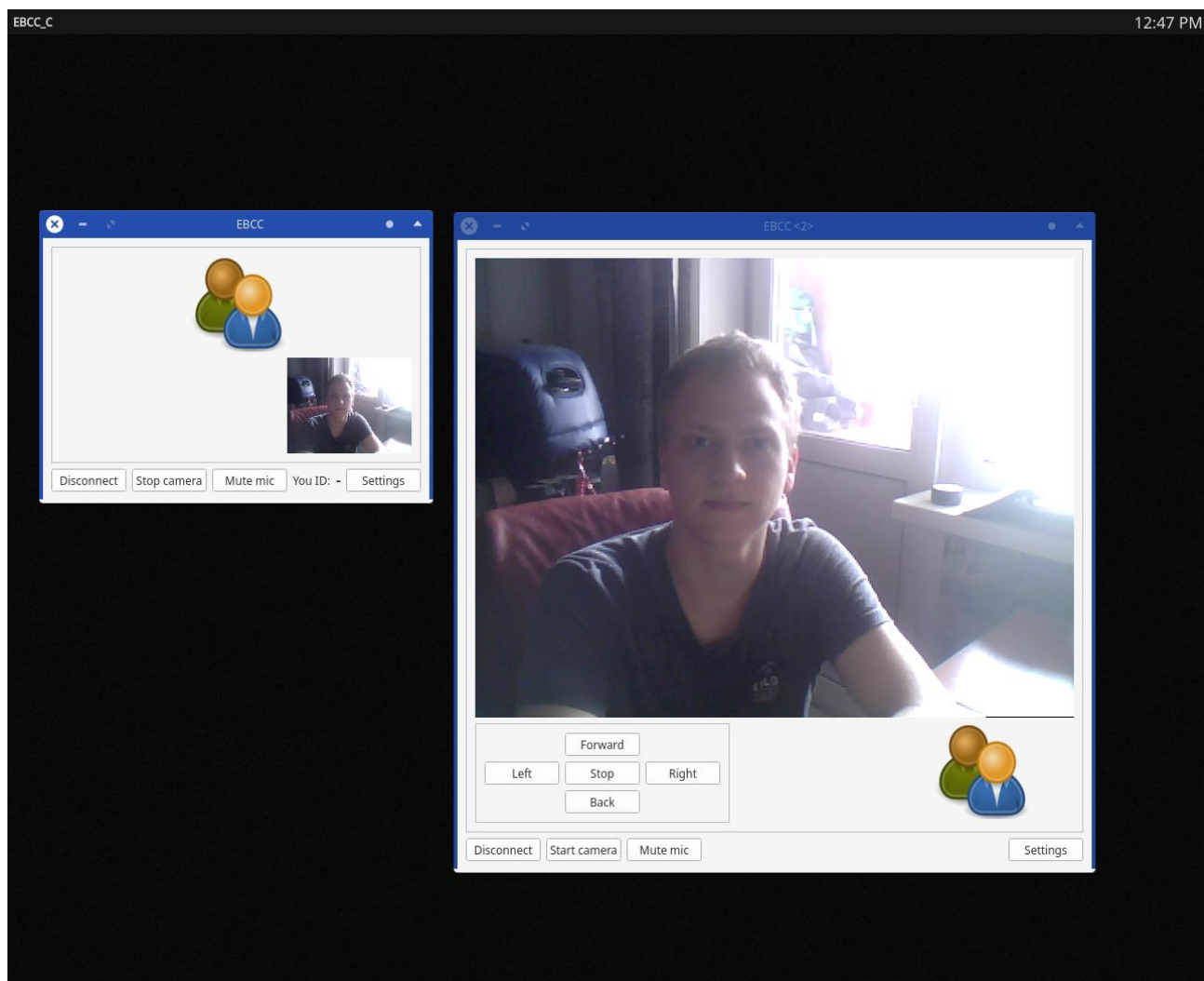


Рисунок 32. Тестирование приложения

Трансляция видео сигнала от клиентского на серверное ПО для работа:

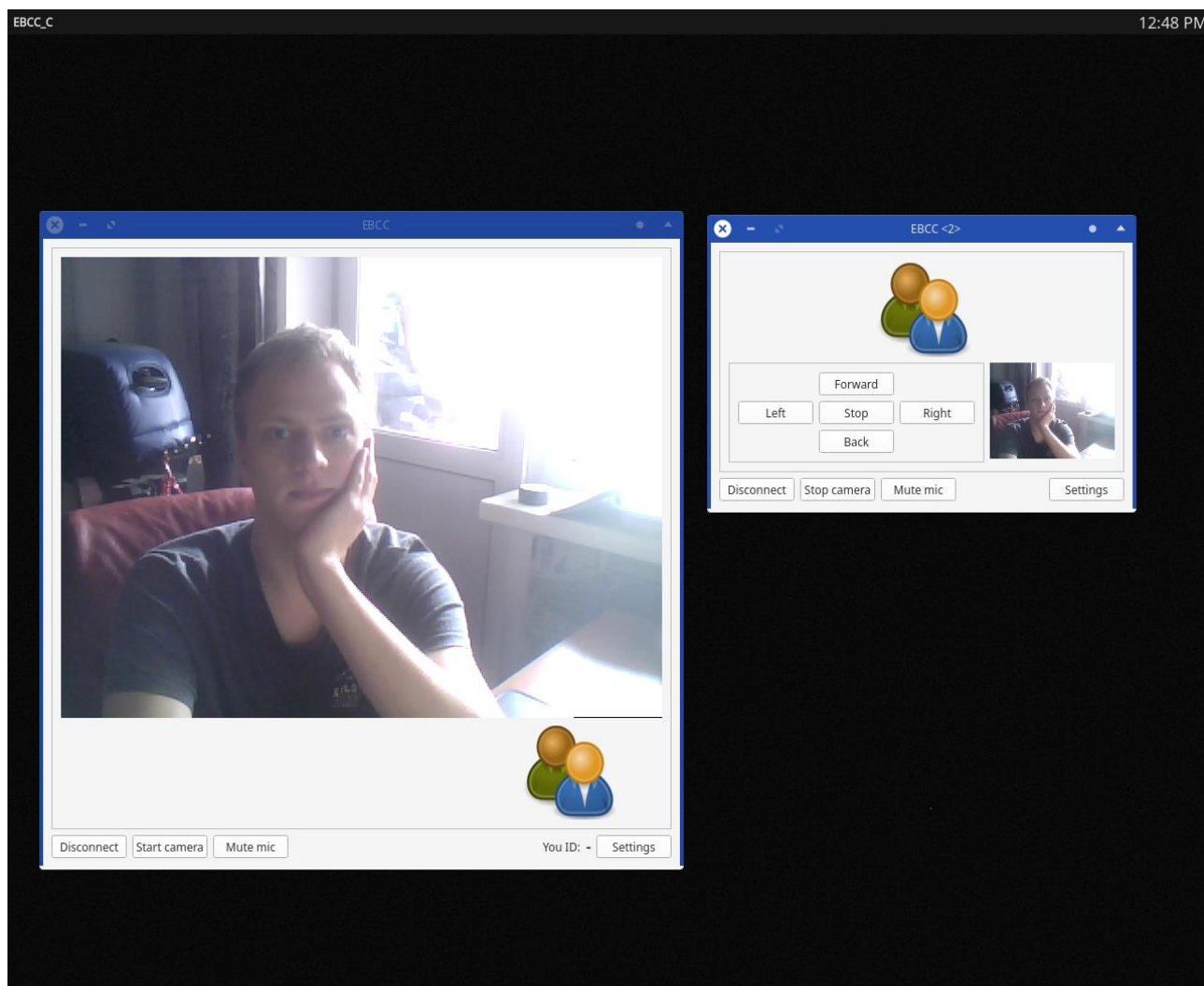


Рисунок 33. Тестирование приложения

Аудио проверяется на слух, увидеть работу аудио и его воспроизведение можно в PulseAudio в графическом интерфейсе KDE Plasma 5

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						37
Изм	Лист	№ докум.	Подп.	Дата		

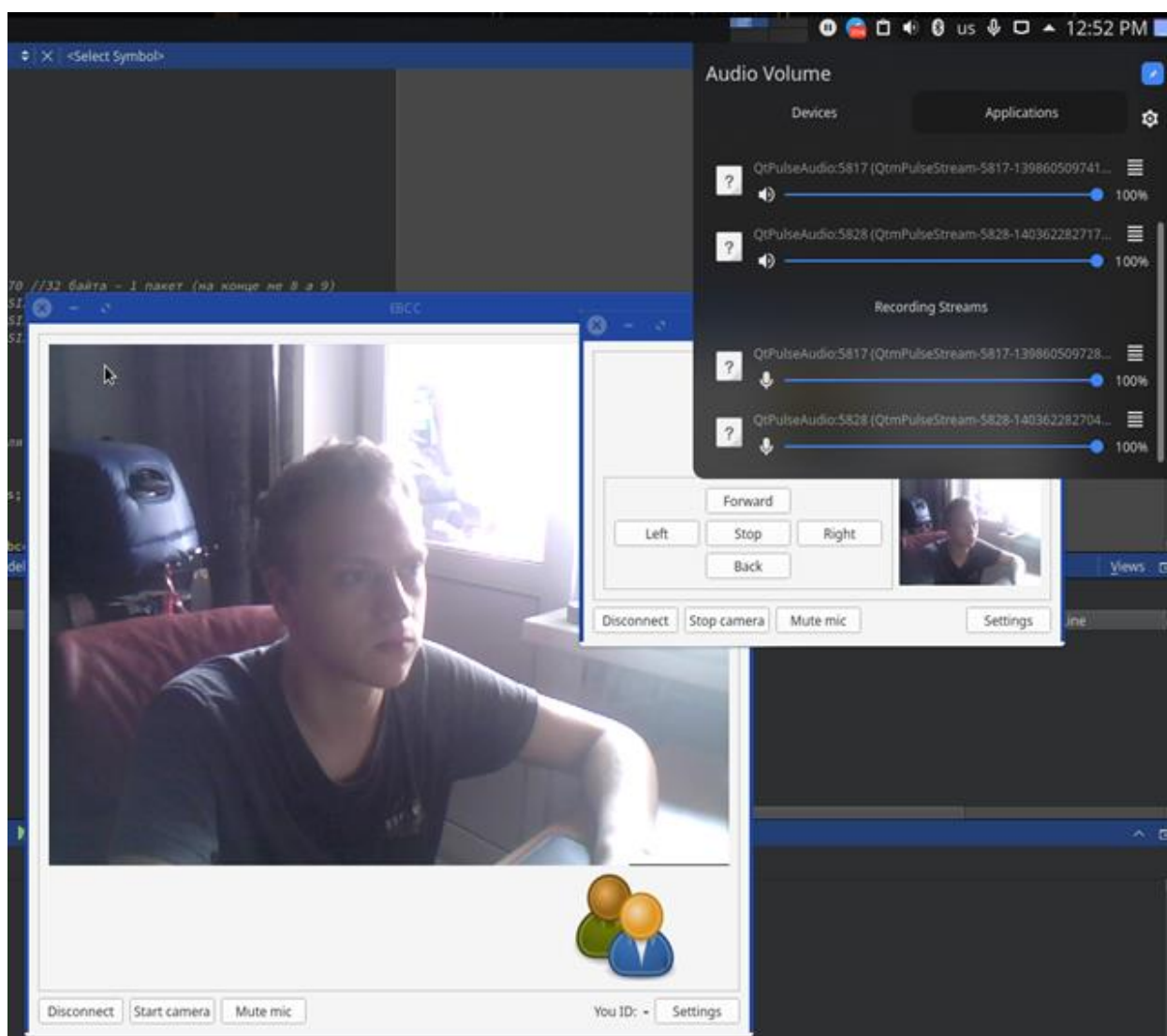


Рисунок 34. Тестирование приложения

Можно заметить 2 устройства воспроизведения и 2 устройства захвата, при прослушивании звук передается четкий с небольшой задержкой.

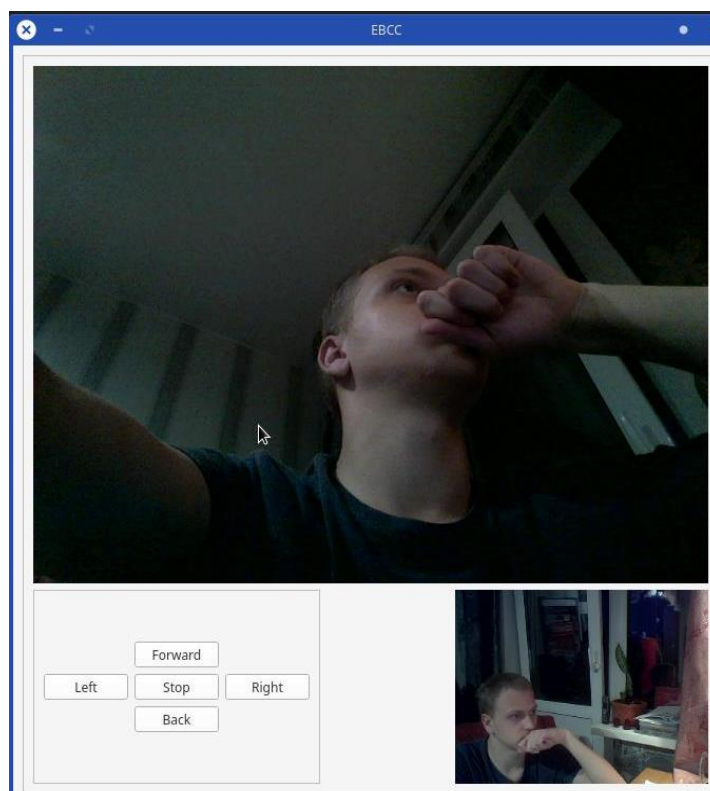


Рисунок 35. Тестирование приложения

Выше продемонстрирована работа приложения с двух устройств, один стационарный ПК, другой ноутбук. На изображении приложение работает на стационарном ПК, ноутбук работает в режиме сервера для робота.

5.4. Проверка доставки сообщений с клиента на серверное ПО

В клиентском приложении после установки соединения нажмем кнопки по очереди «W», «W», «C», «D», «C». в серверном ПО, в отладочной консоли должны увидеть сообщения следующего вида:

Forward

Forward

Stop

Right

Stop

Посмотрим результат работы приложений

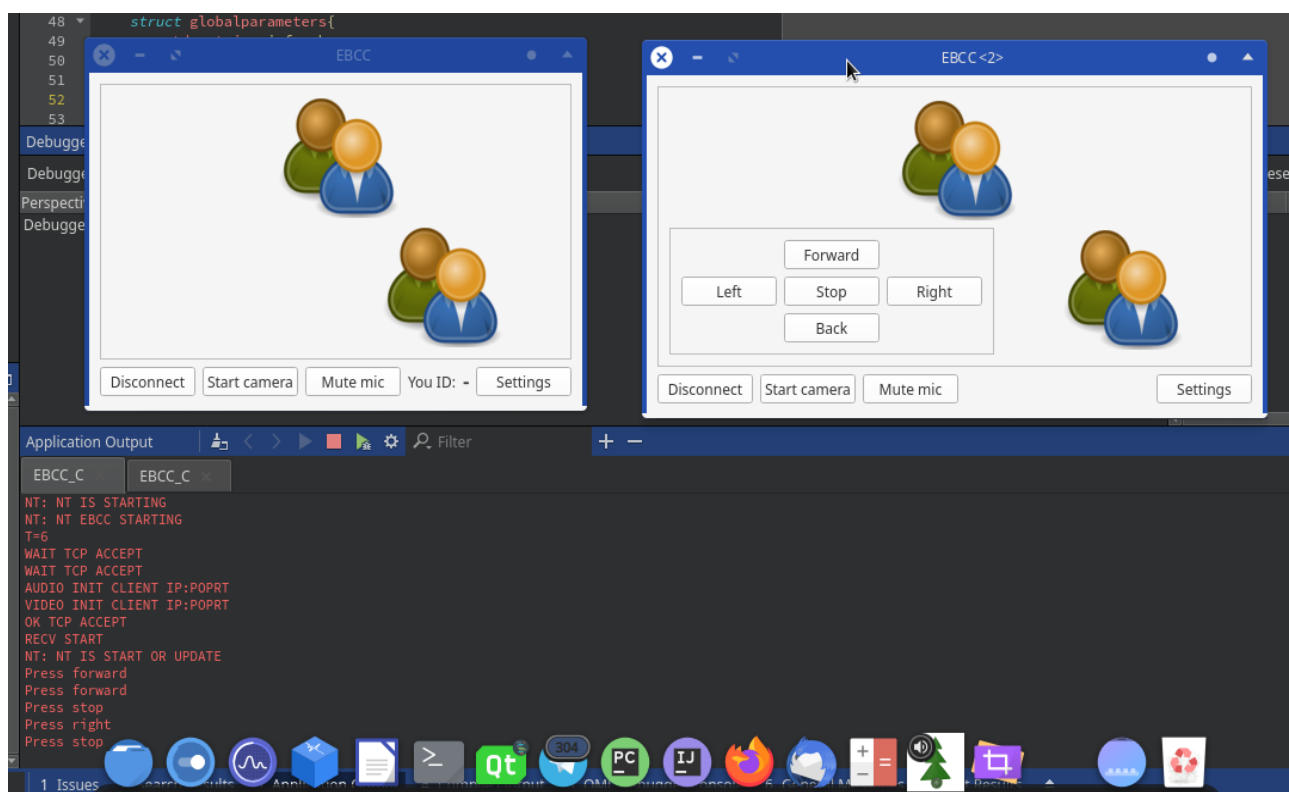


Рисунок 36. Тестирование приложения

Из результатов тестирования, можно сделать вывод, что приложение работает так как и требовалось.

Заключение

Во время выполнения выпускной квалификационной работы было спроектировано программное обеспечение и взаимодействие между программами через сеть. Также разработаны две программы. Первая программа для системы телеприсутствия мобильного робота и клиента, вторая для сигнального сервера. В разработке были применены программные интерфейсы для обмена данными между программами, библиотеки OpenCV и Qt библиотеки. Тестирование показало, что разработанные программы соответствуют требованиям технического задания, а также работают корректно. В данный проект можно дополнительно добавить функционал, тем самым представляется возможность дальнейшего развития.

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						41
Изм	Лист	№ докум.	Подп.	Дата		

Список литературы

1. Frank B. Brokken. C++ Annotations Version 11.3.0 // Published at the University of Groningen ISBN 90 367 0470 7 1994-2019.
2. У. Р. Стивенс, Б. Феннер, Э. М. Рудофф. UNIX Разработка сетевых приложений
3. Макс Шлее. Qt5.10. Профессиональное программирование на C++ // СПб, BHV-СПб, май 2018.
4. Linux [Электронный ресурс]. – <https://ru.wikipedia.org/wiki/Linux>
5. Qt документация по C++ [Электронный ресурс]. – <https://doc.qt.io/qt-5/reference-overview.html>
6. Документация по OpenCV [Электронный ресурс]. – <https://docs.opencv.org/4.3.0/>

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						42
Изм	Лист	№ докум.	Подп.	Дата		

Приложение

Заголовочные файлы

Для клиентского программного обеспечения

mainwindow.h

```
#ifndef MAINWINDOW_H
```

```
#define MAINWINDOW_H
```

```
#include <QMainWindow>
```

```
#include <QImage>
```

```
#include <QString>
```

```
#include <QThread>
```

```
#include <QtGui>
```

```
#include "formsettings.h"
```

```
#include "videocapture.h"
```

```
#include "audiomanager.h"
```

```
#include "controlmanager.h"
```

```
#include "networktransmission.h"
```

```
#define LOCALPLAYERHEIGHT 100
```

```
QT_BEGIN_NAMESPACE
```

```
namespace Ui { class MainWindow; }
```

```
QT_END_NAMESPACE
```

```
class MainWindow : public QMainWindow
```

```
{
```

```
    Q_OBJECT
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						43
Изм.	Лист	№ докум.	Подп.	Дата		

public:

MainWindow(QWidget *parent = nullptr);

~MainWindow();

signals:

void signalopenvideo(int); //сигнал на открытие камеры с требуемым id

void signalopenaudio(); //сигнал на открытие микрофона (потом рассмотрим выбор микрофона)

void signalcloseaudio(); //выключаем микрофон

void signalaudiostart(); //открываем передачу по start mic

void signalaudiomute();

void signalsendcommand(int);

void signalblockimportantsettings(bool); //сигнал блокировки важных настроек

void signalrequestglobalparameters(); //сигнал запроса глобальных параметров сети в настройках

void signalSendGlobalParametersToNetworkTransmission(dp::globalparameters); //этот сигнал отправит параметры сети, сразу как создается поток в NT

void signalstartnetworktransmission();

void stoppingnetworktransmission();

private slots:

void keyPressEvent(QKeyEvent * event);

void pushbuttonconnectclicked(); //Слот для нажатия на кнопку connect

void pushbuttonstartstopclicked(); //Слот для нажатия на кнопку start/stop

void pushbuttonstartstopmicclicked(); //Слот для нажатия на кнопку start/stop

void pushbuttonsettingsclicked(); //Слот для нажатия на кнопку settings

void pushbuttonforforwardclicked(); //Слот для нажатия на кнопку вперед

void pushbuttonforstopclicked(); //Слот для нажатия на кнопку стоп

void pushbuttonforbackclicked(); //Слот для нажатия на кнопку назад

void pushbuttonforleftclicked(); //Слот для нажатия на кнопку лево

void pushbuttonforrightclicked(); //Слот для нажатия на кнопку право

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						44
Изм	Лист	№ докум.	Подп.	Дата		

```

void startvideo(); //Действие на успешное начало потока видео
void startaudio(); //Действие на успешное начало потока аудио

void videorunning(); //Действие на успешное начало захвата потока видео
void audiorunning();
void audiostart();
void videostopped(); //Действие на (успешное/не очень) закрытие захвата потока
ВИДЕО
void audiomute();
void audiostopped();

void getglobalparameters(dp::globalparameters); //получение глобальных параметров
void sendGlobalParametersToNetworkTransmission(); //отправляем глобальные
параметры, сразу после создания потока для networktransmission
void globalparametersinNTreceived(); //действие на успешное получения GP от NT
void networktransferstarted(); //Действие на успешное соединение
void networkdatatransmissionclosed(); //Действие на закрытие соединения или если
соединение не состоялось

void showid(std::string);

void closedsettingsform();

void showlocalimg(cv::Mat img); //Действие на успешное получение локального кадра
void showremoteimg(cv::Mat img); //Действие на успешное получение удаленного
кадра
void showbanneronlocalplayer();
void showbanneronremoteplayer();

private:
    Ui::MainWindow *ui;

    FormSettings * formsettings; //окно настроек

```

```

QThread * videocapturethread; //поток для захвата видео
QThread * audiomanagerthread; //поток для захвата аудио

QThread * networktransmissionthread; //поток для передачи данных по сети

dp::VideoCapture * videocapture; //класа захвата видео
dp::AudioManager * audiomanager; //класа захвата аудио
dp::ControlManager * controlmanager; //класа для управления

dp::NetworkTransmission * networktransmission; //клас передачи данных по сети

QPixmap defaultimg; //изображение по умолчанию
QImage mat2Qimg(const cv::Mat &src); //конвертор cv::Mat в QImage

dp::globalparameters gp; //глобальные параметры для передачи по сети, хранятся тут
пока не нажмут connect

float ratesizelocalheight;

bool isbuttonconnection; //для многофункциональной кнопки связи
bool isbuttonenablecamera; //для многофункциональной кнопки камеры
bool isbuttonenablemic;

};

#endif // MAINWINDOW_H

formsettings.h

#ifndef FORMSETTINGS_H
#define FORMSETTINGS_H
#include <QDialog>
#include <networktransmission.h>

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						46
Изм	Лист	№ докум.	Подп.	Дата		


```

namespace Ui {
class FormSettings;
}

class FormSettings : public QDialog
{
    Q_OBJECT

public:
    explicit FormSettings(QDialog *parent = nullptr);
    ~FormSettings();

signals:
    void signalclosesettingsform(); //сигнал для главного меню, необходим для
реагирования главного окна зн закрытие формы настроек

private:
    Ui::FormSettings *ui;

    dp::globalparameters gp; //структура для пременения сетевых параметров

private slots:
    //слоты на действие кнопок
    void setCustom(int);
    void pushbuttonsaveclicked(); //действие на сохранение
    void pushbuttonapplyclicked(); //действие на закрытие с сохранением
    void pushbuttoncloseclicked(); //действие на закрытие
    void setCheckBox(int);

    void blockimportantsettings(bool);

protected:
    void closeEvent(QCloseEvent *); //обработка события закрытия окна через ОС

signals:
    void signalglobalparameters(dp::globalparameters); //сигнал для передачи параметров
для работы по сети
};

```

```

#endif // FORMSETTINGS_H

videocapture.h

#ifndef VIDEOCAPTURE_H
#define VIDEOCAPTURE_H

#include <QObject>
#include <QImage>

#include "opencv2/opencv.hpp"
#include "opencv2/highgui.hpp"

#include "networktransmission.h"

namespace dp {
    class VideoCapture : public QObject
    {
        Q_OBJECT

    public:
        VideoCapture(dp::NetworkTransmission *);

    signals:
        void signalinterrupt(); //сигнал прерывания

        void signalshowimg(cv::Mat img); //сигнал на передачу кадра для последующей
отрисовки
        void signalcameraisopen(); //сигнал о том что камера открыта

    private slots:
        void open(int id); //открытие камеры
        void start(); //захват кадров с камеры
        void close(); //закрытие камеры и потока

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						48
Изм	Лист	№ докум.	Подп.	Дата		

```

private:
    dp::NetworkTransmission * networktransmission;

    cv::VideoCapture cap;

    bool isStart; //нужно для своевременного выхода из цикла
};
}

#endif // VIDEOCAPTURE_H

audiomanager.h

#ifndef AUDIOCAPTURE_H
#define AUDIOCAPTURE_H

#include <QObject>
#include <QAudioInput>
#include <QAudioOutput>
#include <QBuffer>

#include "networktransmission.h"

namespace dp {
    class AudioManager : public QObject
    {
        Q_OBJECT
    public:
        AudioManager(dp::NetworkTransmission *);
    signals:
        void signalinterrupt(); //сигнал прерывания

        void signalaudioisopen(); //сигнал о том что микрофон открыт
        void signalaudiostart();
        void signalaudiomute();

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						49
Изм	Лист	№ докум.	Подп.	Дата		

```

private slots:
    void open(); //открытие микрофона
    void start(); //захват звука с микрофона
    void mute();
    void close(); //закрытие микрофона и потока
    void datasend();
    void playdata(QBuffer *);

private:
    dp::NetworkTransmission * networktransmission;

    //QByteArray output_bytes;
    QBuffer * output_buffer;
    QBuffer * input_buffer;

    QAudioFormat audioformat;
    QAudioDeviceInfo inputDevice;
    QAudioInput* audio_in = nullptr;
    QAudioOutput* audio_out = nullptr;
    QAudioDeviceInfo outputDevice;

    bool isStart;
};
}

#endif // AUDIOCAPTURE_H

controlmanager.h

#ifndef CONTROLMANAGER_H
#define CONTROLMANAGER_H

#include <QObject>

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						50
Изм	Лист	№ докум.	Подп.	Дата		

```

#include "networktransmission.h"

namespace dp {
    class ControlManager : public QObject
    {
        Q_OBJECT
    public:
        ControlManager(dp::NetworkTransmission *);
    private slots:
        //этими слотами отправляем команды роботу
        void sendcommand(int);
        void startmovement(int);
    private:
        dp::NetworkTransmission * networktransmission;

        void startforward();
        void startstop();
        void startback();
        void startleft();
        void starttright();
    };
}

#endif // CONTROLMANAGER_H

networktransmission.h

#ifndef NETWORKTRANSMISSION_H
#define NETWORKTRANSMISSION_H

//определение платформы для использования нужных библиотек

#define PLATFORM_UNIX    1
#define PLATFORM_WINDOWS 2

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						51
Изм	Лист	№ докум.	Подп.	Дата		

```

#if defined(__linux__)
#define PLATFORM PLATFORM_UNIX
#endif

#include <QObject>
#include <string>

#include "opencv2/opencv.hpp"
#include "opencv2/highgui.hpp"

#include <QInputDialog>
#include <QLineEdit>

#include <QBuffer>

#if PLATFORM == PLATFORM_UNIX

//сеть
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define PACKAGESIZE_V 264//1470 //32 байта - 1 пакет (на конце не 8 а 9) делится на
3 без остатка
#define PACKAGESIZE_A 3129 //SIZE FOR AUDIO ~ 10ms аудиодорожки 44100
#define PACKAGESIZE_C 1024 //SIZE FOR CONTROL
#define PACKAGESIZE_R 1024 //SIZE FOR REGISTRAR

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						52
Изм.	Лист	№ докум.	Подп.	Дата		

```

#endif

namespace dp {

//Глобальные настройки, для общения по сети

struct globalparameters{
    std::string ipforebcs;
    std::string ip;

    unsigned int portforebcs;
    unsigned int portforvideo;
    unsigned int portforaudio;
    unsigned int portforcontrol;

    bool isRegistrarEnable = false;
    bool isElcBot = false;
};

class NetworkRecv : public QObject
{
    Q_OBJECT
public:
    NetworkRecv(bool isElcBot);
    NetworkRecv(int sockfd,struct sockaddr_in servaddr, struct sockaddr_in cliaddr, bool
isElcBot);
    signals:
        void signalsetclientaddress(struct sockaddr_in, int); //сигнал определения адреса
клиента
        void signalsetsock(int); //сигнал для отправки sock
        void signalstartmovement(int);
        void signalrecvregistrardata(unsigned char *, int); //получили какие-то данные от
регистратора
        void signalserverenable();

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						53
Изм	Лист	№ докум.	Подп.	Дата		

```
void signalshowimg(cv::Mat); //отправить полученный кадр на экран
void signalplaydata(QBuffer *); //отправить аудио фрагмент на колонки
void signaldisablecamera(); //сигнал отключения камеры
void signalclosenetworktransmission(); //сигнал закрытия соединения, он еще не
используется
```

```
private slots:
```

```
int recvimage(); //это принимает данные
int recvaudio(); //это принимает данные
int recvcontrol(); //это принимает команды с tcp, не только управления (в
```

```
дальнейшем)
```

```
int recvregistrardata(); //это принимает данные по tcp, с регистратора
```

```
int acceptsock(int, int, int, struct sockaddr_in);
```

```
int connectsock(int, struct sockaddr_in);
```

```
int connectregistrarsock(int, struct sockaddr_in);
```

```
void closeaudiothread();
```

```
private:
```

```
int sockfd; //сокеты
```

```
int sock;
```

```
int registrarsock;
```

```
bool isElcBot; //что за платформа
```

```
struct sockaddr_in servaddr, cliaddr; //адрес сокетов для клиента и сервера
```

```
QBuffer * buff;
```

```
int numbytes; //количество байт
```

```
socklen_t len; //длина адреса
```

```
char buffersizerecvV[PACKAGESIZE_V]; //буфер для передачи данных
```

```
char buffersizerecvA[PACKAGESIZE_A]; //буфер для передачи данных
```

```
char buffersizerecvC[PACKAGESIZE_C]; //буфер для передачи данных
```

```
unsigned char buffersizerecvR[PACKAGESIZE_R]; //буфер для передачи данных
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						54
Изм	Лист	№ докум.	Подп.	Дата		


```

    bool isCamEnable; //определить передается ли данные камеры
};

class NetworkTransmission : public QObject
{
    Q_OBJECT
public:
    NetworkTransmission();

    int sendimage(cv::Mat frame); //сюда придет изображение, которое надо будет
отправить
    int sendaudio(QBuffer *); //сюда придет аудио дорожка, которое надо будет
отправить
    int sendcommandcontrol(int);
    int recv();
    int isreadyaccept(); //для проверки, готова ли принять изображения

signals:
    void signalglobalparametersreceived(); //сигнал об получение и конфигурирование
параметров сети
    void signalconnectionestablished(); //сигнал об успешном соединении
    void signalconnectionstopped(); //сигнал о прекращении связи
    void signalshowimg(cv::Mat); //отправить полученный кадр на экран
    void signalplaydata(QBuffer *); //отправить аудио фрагмент на колонки
    void signalremotedisablecamera();
    void signalcloseaudiothread();
    void signalclosenetworktransmission();

    void signalshowid(std::string);
    void signalstartmovement(int); //сигнал управления роботом

    void signalacceptsock(int, int, int, struct sockaddr_in);
    void signalconnectsock(int, struct sockaddr_in);

```

```
void signalconnectregistrarsock(int, struct sockaddr_in);
```

```
private slots:
```

```
void setglobalparameters(dp::globalparameters gp); //слот для применения  
параметров работы по сети
```

```
void startsystemnetworktransmission(); //инициализация подключения и дальнейшая  
передача данных
```

```
void getclientaddress(struct sockaddr_in, int);
```

```
void getsock(int);
```

```
void tcpserverenable();
```

```
int acceptsock();
```

```
int connectsock();
```

```
void startmovement(int);
```

```
void recvregistrardata(unsigned char *, int);
```

```
void sendimagetoplayer(cv::Mat); //отправить кадр
```

```
void playdata(QBuffer *);
```

```
void disablecamera();
```

```
void stoppingtheconnection(); //Действие на разрыв соединени
```

```
private:
```

```
void connectionestablished(); //это говорит что соединение установлено
```

```
void connectionstopped(); //действие на остановку связи
```

```
//Запрос ID или ip и потры
```

```
int sendDataForRegistrar();
```

```
bool isTCPConnect = false;
```

```
bool isTCPRegistrarConnect = false;
```

```
bool isReadConnect = false; //Статус готовности соединения
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						56
Изм	Лист	№ докум.	Подп.	Дата		

```

bool isReadConnectV = false; //Статус готовности соединения
bool isReadConnectA = false; //Статус готовности соединения
bool isReadConnectC = false; //Статус готовности соединения
int countframe; //Счетчик отправленных кадров

//сеть
//UDP
int sockfdvideo;
int sockfdaudio;
//TCP
int server_fd, sock, opt, addrlenforcontrol;

int registrarsock;

//эти структуры нужны для того, чтобы указать разные порты, для разных данных
(видео/аудио/управление)
//для UDP
struct sockaddr_in servaddrforvideo, cliaddrforvideo;
struct sockaddr_in servaddrforaudio, cliaddrforaudio;
//для TCP
struct sockaddr_in servaddrforcontrol;
struct sockaddr_in servaddrforregistrar;

int numbytes;
socklen_t len;

//для определения сервера или клиента
bool isElcBot;
bool isRegistrar;

NetworkRecv * networkRecvforvideo;
NetworkRecv * networkRecvforaudio;
NetworkRecv * networkRecvforcontrol;
NetworkRecv * networkRecvforregistrar;
QThread * networkrecvforvideothread;

```

```

QThread * networkrecvforaudiothread;
QThread * networkrecvforcontrolthread;
QThread * networkrecvforregistrarthread;

globalparameters gp;

unsigned int imgSize; //размер изображения
unsigned int countpackage; //количество передаваемых пакетов

uchar bufferimgsizesend[PACKAGESIZE_V];
uchar bufferaudiosizesend[PACKAGESIZE_A];
uchar buffercontrolsizesend[PACKAGESIZE_C];
uchar bufferregistrarsizesend[PACKAGESIZE_R];
};
}
#endif // NETWORKTRANSMISSION_H
Для сигнального сервера

```

networktransmission.h

```

#ifndef NETWORKTRANSMISSION_H
#define NETWORKTRANSMISSION_H

```

```

#include <sys/socket.h>
#include <string.h>

```

```

#include <thread>
#include <mutex>

```

```

#include "datacommunication.h"
#include "dataprocessing.h"

```

```

#define DEFAULTPORT 9760

```

```

namespace dp {

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		58

```

class NetworkTransmission
{
public:
    NetworkTransmission(DataBase * database);
    ~NetworkTransmission();

    void initialization();
    void initialization(unsigned int port);
private:
    void setglobalparameters(unsigned int port); //слот для применения параметров
работы по сети
    void startsystemnetworktransmission(); //инициализация подключения и дальнейшая
передача данных
    void stoppingtheconnection(); //Действие на разрыв соединени

    int server_fd, sock, numbytes, remoteip;
    struct sockaddr_in address;

    int opt;
    int addrlen;

    DataCommunication * dataCommunication;
    DataProcessing * dataProcessing;

    unsigned int port;
    unsigned char * buff;
    unsigned int sizebuff;

};
}
#endif // NETWORKTRANSMISSION_H
database.h

#ifndef DATABASE_H

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						59
Изм	Лист	№ докум.	Подп.	Дата		

```

#define DATABASE_H

#include <string>
#include <vector>

namespace dp {
    class DataBase
    {
        struct device {
            std::string ip;
            std::string id;

            bool isLock;
        };

        struct elcbotdevice : public device {
            unsigned int portForVideo;
            unsigned int portForAudio;
            unsigned int portForControl;
        };

    private:
        std::vector<elcbotdevice *> availableElcBotDevice;
        std::vector<device *> availableClientDevice;

    public:
        int getPort();
        void regDev(int);
        std::vector<elcbotdevice *> searchDev(int);
        void remDev(int);
        void clear();
    };
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						60
Изм	Лист	№ докум.	Подп.	Дата		

```

#endif // DATABASE_H

dataprocessing.h

#ifndef DATAPROCESSING_H
#define DATAPROCESSING_H

#include <thread>
#include <mutex>

#include <chrono>

#include <iostream>

#include "database.h"

namespace dp {
    class DataProcessing
    {
    public:
        DataProcessing(DataBase * database);
        ~DataProcessing();

        unsigned char * commandProcessing(int, unsigned char *, unsigned int &numbytes,
unsigned int &);
    private:
        DataBase * dataBase;
    };
}

#endif // DATAPROCESSING_H

datacommunication.h

#ifndef DATACOMMUNICATION_H
#define DATACOMMUNICATION_H

#include <unistd.h>

```

					БКР-ИГТҮ-09.03.01-(16-В-2)-002-2020(ИЗ)	Лист
						61
Изм	Лист	№ докум.	Подп.	Дата		

```

#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>

#define BUFSIZE 1024

namespace dp {
    class DataCommunication
    {
    public:
        DataCommunication(int server_fd, int sock,
                           struct sockaddr_in address, int addrlen);
        ~DataCommunication();

        struct sockaddr_in accept();
        unsigned char * recvData(unsigned int &numbytes);
        int sendData(unsigned char *, unsigned int sizemess);

    private:
        int server_fd, sock, numbytes;
        struct sockaddr_in address;
        int addrlen;

        unsigned char * buff;
    };
}

#endif // DATACOMMUNICATION_H

```

Исходные файлы

Для клиентского программного обеспечения

main.cpp

```
#include "mainwindow.h"
```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						62
Изм	Лист	№ докум.	Подп.	Дата		


```

#include <QApplication>

/* Для справки
 * NT - NetworkTransmission
 * GP - Глобальные параметры сети
 */

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}

```

mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QPainter>

using namespace std;

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new
Ui::MainWindow)
{

    qRegisterMetaType<dp::globalparameters>("dp::globalparameters");
    qRegisterMetaType<std::string>("std::string");
    qRegisterMetaType<cv::Mat>("cv::Mat");
    qRegisterMetaType<sockaddr_in>("sockaddr_in");

    //классы окон
    formsettings = new FormSettings();

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						63
Изм	Лист	№ докум.	Подп.	Дата		

networktransmission = new dp::NetworkTransmission(); //класс для передачи данных по сети

networktransmissionthread = new QThread(); //поток, нужен, для того чтобы соединение и передача данных не вешало UI

videocapture = new dp::VideoCapture(networktransmission); //класс для захвата изображения с камеры

videocapturethread = new QThread(); //поток, нужен, чтобы получение кадров не вешало UI

audiomanager = new dp::AudioManager(networktransmission); //класс для захвата изображения с камеры

audiomanagerthread = new QThread(); //поток, нужен, чтобы получение кадров не вешало UI

controlmanager = new dp::ControlManager(networktransmission);

videocapture->moveToThread(videocapturethread);

audiomanager->moveToThread(audiomanagerthread);

networktransmission->moveToThread(networktransmissionthread);

ui->setupUi(this);

//Тут все параметры которые определяют состояния элементов на момент старта приложения

//фиксированный размер окна

setFixedSize(0, 0);

defaultimg = QPixmap("images/users.png");

showbanneronlocalplayer();

showbanneronremoteplayer();

//если кнопка соединения true, то связь не установлена, в противном случае установлена

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						64
Изм	Лист	№ докум.	Подп.	Дата		

```

isbuttonconnection = true;
isbuttonenablecamera = false;
isbuttonenablemic = false;
//включаем микрофон

ui->pushButtonStartStopMic->setEnabled(false);
//pushbuttonstartstopmicclicked();

//это обработчик кнопок
connect(ui->pushButtonConnect, SIGNAL(clicked()), this,
SLOT(pushbuttonconnectclicked())); //запускаем поток и выполняем соединение по
событию соединиться
connect(ui->pushButtonStartStop, SIGNAL(clicked()), this,
SLOT(pushbuttonstartstopclicked())); //запускаем поток по событию старт или закрываем
connect(ui->pushButtonStartStopMic, SIGNAL(clicked()), this,
SLOT(pushbuttonstartstopmicclicked())); //запускаем поток по событию старт mic или
закрываем
connect(ui->pushButtonSettings, SIGNAL(clicked()), this,
SLOT(pushbuttonsettingsclicked())); //обработка нажатой кнопки настройки
connect(ui->pushButtonForward, SIGNAL(clicked()), this,
SLOT(pushbuttonforforwardclicked())); //обработка нажатой кнопки вперед
connect(ui->pushButtonStop, SIGNAL(clicked()), this,
SLOT(pushbuttonforstopclicked())); //обработка нажатой кнопки стоп
connect(ui->pushButtonBack, SIGNAL(clicked()), this,
SLOT(pushbuttonforbackclicked())); //обработка нажатой кнопки назад
connect(ui->pushButtonLeft, SIGNAL(clicked()), this,
SLOT(pushbuttonforleftclicked())); //обработка нажатой кнопки лево
connect(ui->pushButtonRight, SIGNAL(clicked()), this,
SLOT(pushbuttonforrightclicked())); //обработка нажатой кнопки право

//обработчик старта/стопа камеры и аудио, умеет даже прерываться если выдернуть
кабель (видео)

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						65
Изм.	Лист	№ докум.	Подп.	Дата		

```

connect(videocapturethread, SIGNAL(started()), this, SLOT(startvideo())); //открываем
камеру
connect(audiomanagerthread, SIGNAL(started()), this, SLOT(startaudio())); //открываем
микрофон
connect(this, SIGNAL(signalopenvideo(int)), videocapture, SLOT(open(int)));
connect(this, SIGNAL(signalopenaudio()), audiomanager, SLOT(open()));

connect(videocapture, SIGNAL(signalcameraisopen()), this, SLOT(videorunning()));
//камера открыта, блокируем кнопку старт
connect(videocapture, SIGNAL(signalcameraisopen()), videocapture, SLOT(start()));
//камера открыта, запускаем поток

connect(audiomanager, SIGNAL(signalaudioisopen()), this, SLOT(audiorunning()));
//микрофон открыт, разблокируем кнопку mic

connect(this, SIGNAL(signalaudiostart()), audiomanager, SLOT(start()));
connect(this, SIGNAL(signalaudiomute()), audiomanager, SLOT(mute()));

connect(audiomanager, SIGNAL(signalaudiostart()), this, SLOT(audiostart()));
connect(audiomanager, SIGNAL(signalaudiomute()), this, SLOT(audiomute()));
//connect(audiomanager, SIGNAL(signalaudioisopen()), audiomanager, SLOT(start()));
//микрофон открыт, запускаем поток

connect(videocapturethread, SIGNAL(finished()), this, SLOT(videostopped()));
//определяем закрытый поток и разблокируем кнопку
connect(videocapture, SIGNAL(signalinterrupt()), videocapture, SLOT(close())); //если
приходит прерывание, то захват остановлен, закрываем камеру и поток

connect(this, SIGNAL(signalcloseaudio()), audiomanager, SLOT(close()));
connect(audiomanagerthread, SIGNAL(finished()), this, SLOT(audiostopped()));
//определяем закрытый поток и разблокируем кнопку mic

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		66

```
//connect(audiocapture, SIGNAL(signalinterrupt()), audiocapture, SLOT(close())); //если  
приходит прерывание, то захват остановлен, закрываем микрофон и поток
```

```
//обработка сигналов с формы настроек  
connect(formsettings, SIGNAL(signalclosesettingsform()), this,  
SLOT(closedsettingsform())); //обработка закрытия окна формы  
//получение из настроек глобальных параметров для сети  
connect(formsettings, SIGNAL(signalglobalparameters(dp::globalparameters)),  
this, SLOT(getglobalparameters(dp::globalparameters)));
```

```
connect(this, SIGNAL(signalblockimportantsettings(bool)), formsettings,  
SLOT(blockimportantsettings(bool)));
```

```
//выполним запрос глобальных параметров, для установки значений по умолчанию  
connect(this, SIGNAL(signalrequestglobalparameters()), formsettings,  
SLOT(pushbuttonsaveclicked()));
```

```
emit signalrequestglobalparameters();
```

```
//Группа событий связанная с NT
```

```
//новые параметры будут применены, после повторного соединения
```

```
connect(networktransmissionthread, SIGNAL(started()), this,
```

```
SLOT(sendGlobalParametersToNetworkTransmission())); //после старта потока,  
отправляем параметры по умолчанию
```

```
//отправка по событию в NT
```

```
connect(this, SIGNAL(signalSendGlobalParametersToNetworkTransmission(dp::globalparamete  
rs)),
```

```
networktransmission, SLOT(setglobalparameters(dp::globalparameters)));
```

```
//ожидание подтверждения получения глобальных параметров сети
```

```
connect(networktransmission, SIGNAL(signalglobalparametersreceived()),
```

```
this, SLOT(globalparametersinNTreceived()));
```

```
//запуск установки соединения и сетевой передачи
```

```
connect(this, SIGNAL(signalstartnetworktransmission()),
```

```
networktransmission, SLOT(startsystemnetworktransmission()));
```

```
//ожидание успешного соединения
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						67
Изм.	Лист	№ докум.	Подп.	Дата		

```

connect(networktransmission, SIGNAL(signalconnectionestablished()),
        this, SLOT(networktransferstarted()));
//событие на прерывание сетевой передачи из main
connect(this, SIGNAL(stoppingnetworktransmission()),
        networktransmission, SLOT(stoppingtheconnection()));
//закрыть по требованию самой NT, из-за ошибок в tcp/ip udp/ip
connect(networktransmission, SIGNAL(signalclosenetworktransmission()),
        networktransmission, SLOT(stoppingtheconnection()));
//ожидание правильного закрытия соединения, если было установлено
connect(networktransmissionthread, SIGNAL(finished()),
        this, SLOT(networkdatatransmissionclosed()));

//если придет локальный кадр, то обновим его в окне
connect(videocapture, SIGNAL(signalshowimg(cv::Mat)), this,
SLOT(showlocalimg(cv::Mat)));
//если удаленный кадр собеседника, то обновим его в окне
connect(networktransmission, SIGNAL(signalshowimg(cv::Mat)), this,
SLOT(showremoteimg(cv::Mat)));
//если удаленный собеседник отключил камеру
connect(networktransmission, SIGNAL(signalremotedisablecamera()),
        this, SLOT(showbanneronremoteplayer()));

connect(networktransmission, SIGNAL(signalshowid(std::string)), this,
SLOT(showid(std::string)));

connect(this, SIGNAL(signalcommand(int)), controlmanager,
SLOT(sendcommand(int)));
connect(networktransmission, SIGNAL(signalstartmovement(int)), controlmanager,
SLOT(startmovement(int)));

audiomanagerthread->start(); //инициализация устройств аудио
}

MainWindow::~MainWindow()

```

```

{
    emit signalcloseaudio();

    videocapturethread->requestInterruption(); //отправляем запрос на прерывание
    videocapturethread->wait(10000); //ожидаем завершения потока

    //остановка соединения
    networktransmissionthread->requestInterruption();
    emit stoppingnetworktransmission();
    networktransmissionthread->wait(10000);

    delete videocapture;
    delete videocapturethread;
    delete networktransmission;
    delete networktransmissionthread;
    delete formsettings;
    delete controlmanager;
    delete ui;
}

//Многофункциональная кнопочка)
void MainWindow::pushbuttonconnectclicked(){
    ui->pushButtonConnect->setEnabled(false);
    if(isbuttonconnection){
        qDebug("MAIN: START THREAD NT");
        emit signalblockimportantsettings(true); //блокируем настройки
        networktransmissionthread->start();
    }
    else{
        qDebug("MAIN: STOPPING THREAD NT");
        //запрос на прерывание в потоке NT
        networktransmissionthread->requestInterruption();
        emit stoppingnetworktransmission();
    }
}

```

```

}

//Обработчик нажатия кнопки start/stop cam
void MainWindow::pushbuttonstartstopclicked(){
    ui->pushButtonStartStop->setEnabled(false);

    if(!isbuttonenablecamera){
        videocapturethread->start();
    }
    else{
        videocapturethread->requestInterruption();
    }
}

//Обработчик нажатия кнопки start/stop mic
void MainWindow::pushbuttonstartstopmicclicked(){
    ui->pushButtonStartStopMic->setEnabled(false);

    if(!isbuttonenablemic){
        emit signalaudiostart();
        //audiomanagerthread->start();
    }
    else{
        emit signalaudiomute();
    }
}

void MainWindow::pushbuttonforforwardclicked(){
    emit signalsendcommand(0);
}

void MainWindow::pushbuttonforstopclicked(){
    emit signalsendcommand(1);
}

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						70
Изм	Лист	№ докум.	Подп.	Дата		


```

void MainWindow::pushbuttonforbackclicked(){
    emit signalsendcommand(2);
}

void MainWindow::pushbuttonforleftclicked(){
    emit signalsendcommand(3);
}

void MainWindow::pushbuttonforrightclicked(){
    emit signalsendcommand(4);
}

void MainWindow::keyPressEvent(QKeyEvent * event){
    if(gp.isElcBot){
        return;
    }

    switch (event->key()) {
    case Qt::Key_W:
        pushbuttonforforwardclicked();
        break;
    case Qt::Key_C:
        pushbuttonforstopclicked();
        break;
    case Qt::Key_S:
        pushbuttonforbackclicked();
        break;
    case Qt::Key_A:
        pushbuttonforleftclicked();
        break;
    case Qt::Key_D:
        pushbuttonforrightclicked();
        break;
    }
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ИЗ)	Лист
						71
Изм	Лист	№ докум.	Подп.	Дата		

```

    }
}

//Действие на нажатие кнопки настроек
void MainWindow::pushbuttonsettingsclicked(){
    formsettings->show();
    ui->pushButtonSettings->setEnabled(false);
}

//Действие на успешное начало потока
void MainWindow::startvideo(){
    int devid = 0; //default 0 - камера по умолчанию
    emit signalopenvideo(devid);
}

//Действие на успешное начало потока
void MainWindow::startaudio(){
    //int devid = 0; //default 0 - камера по умолчанию
    emit signalopenaudio();
}

//Действие на успешное начало захвата потока видео
void MainWindow::videorunning(){
    ui->pushButtonStartStop->setText("Stop camera");
    isbuttonenablecamera = true;
    ui->pushButtonStartStop->setEnabled(true);
}

void MainWindow::audiorunning(){
    pushbuttonstartstopmicclicked();
}

void MainWindow::audiostart(){
    ui->pushButtonStartStopMic->setText("Mute mic");
}

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						72
Изм	Лист	№ докум.	Подп.	Дата		

```

    ui->pushButtonStartStopMic->setEnabled(true);
    isbuttonenablemic = true;
}

```

//Действие на (успешное/не очень) закрытие захвата потока видео

```

void MainWindow::videostopped(){
    ui->pushButtonStartStop->setText("Start camera");
    isbuttonenablecamera = false;
    ui->pushButtonStartStop->setEnabled(true);

    showbanneronlocalplayer();
}

```

```

void MainWindow::audiomute(){
    ui->pushButtonStartStopMic->setText("Start mic");
    isbuttonenablemic = false;
    ui->pushButtonStartStopMic->setEnabled(true);
}

```

```

void MainWindow::audiostopped(){
    exit(1);
}

```

//Получение и запись глобальных параметров сети в mainwindows, для дальнейшего ожидания старта потока

```

void MainWindow::getglobalparameters(dp::globalparameters gp){
    this->gp = gp;

    if(gp.isElcBot) {
        ui->frame_2->setVisible(false);
        ui->labelYouId->setVisible(true);
        ui->label_ID->setVisible(true);
        ui->pushButtonConnect->setText("Start");
    }
}

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						73
Изм	Лист	№ докум.	Подп.	Дата		

```

else{
    ui->frame_2->setVisible(true);
    ui->labelYouId->setVisible(false);
    ui->label_ID->setVisible(false);
    ui->pushButtonConnect->setText("Connect");
}

qDebug("MAIN: Global parameters are received");
}

//отправка глобальных параметров сети, как только поток NT будет запущен
void MainWindow::sendGlobalParametersToNetworkTransmission(){
    qDebug("MAIN: NT THREAD IS START, SEND GP");
    emit signalSendGlobalParametersToNetworkTransmission(gp);
}

//Действие на успешное получение глобальных параметров сети системой передачи
данных
void MainWindow::globalparametersinNTreceived(){
    qDebug("MAIN: NT GET GP");

    //запуск системы передачи данных
    emit signalstartnetworktransmission();
}

//Действие на успешный запуск и установку соединения системы сетевой передачи
void MainWindow::networktransferstarted(){
    isbuttonconnection = false;
    ui->pushButtonConnect->setText("Disconnect");
    ui->pushButtonConnect->setEnabled(true);
    emit signalblockimportantsettings(true); //блокируем настройки
}

//Действие на закрытие соединения или если соединение не состоялось

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						74
Изм	Лист	№ докум.	Подп.	Дата		

```

void MainWindow::networkdatatransmissionclosed(){
    //тут делаем все, чтобы можно было создать новое соединение
    qDebug("MAIN: NT THREAD STOP");

    showbanneronremoteplayer();
    ui->label_ID->setText("-");
    isbuttonconnection = true;
    if(gp.isElcBot){
        ui->pushButtonConnect->setText("Start");
    }
    else{
        ui->pushButtonConnect->setText("Connect");
    }
    ui->pushButtonConnect->setEnabled(true);
    emit signalblockimportantsettings(false); //разблокируем настройки
}

void MainWindow::showid(std::string id){
    ui->label_ID->setText(QString::fromStdString(id));
}

//Действие на закрытие формы настроек
void MainWindow::closedsettingsform(){
    ui->pushButtonSettings->setEnabled(true);
}

//Конвертер cv::Mat в QImage
QImage MainWindow::mat2Qimg(const cv::Mat &src) {
    cv::Mat temp;

    cvtColor(src, temp, cv::COLOR_BGR2RGB);

    QImage dest((uchar*) temp.data, temp.cols, temp.rows, temp.step,
    QImage::Format_RGB888);

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		75

```

    QImage dest2(dest);

    dest2.detach();

    return dest2;
}

//Действие на успешное получение локального кадра
void MainWindow::showlocalimg(cv::Mat img){
    ratesizelocalheight = static_cast<float>(LOCALPLAYERHEIGHT) / img.rows;
    cv::resize(img, img, cv::Size(), ratesizelocalheight, ratesizelocalheight);
    ui->localplayer->setPixmap(QPixmap::fromImage(mat2Qimg(img), Qt::AutoColor));
}

//Действие на успешное получение удаленного кадра
void MainWindow::showremoteimg(cv::Mat img){
    ui->remoteplayer->setPixmap(QPixmap::fromImage(mat2Qimg(img), Qt::AutoColor));
}

//показать банер, если удаленный узел отключил камеру
void MainWindow::showbanneronlocalplayer(){
    ui->localplayer->setPixmap(defaultimg);
}

//показать банер, если удаленный узел отключил камеру
void MainWindow::showbanneronremoteplayer(){
    ui->remoteplayer->setPixmap(defaultimg);
}
formsettings.cpp

#include "formsettings.h"
#include "ui_formsettings.h"

FormSettings::FormSettings(QDialog *parent) :

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						76
Изм	Лист	№ докум.	Подп.	Дата		

```

QDialog(parent),
ui(new Ui::FormSettings)
{
    ui->setupUi(this);

    //установка параметров по умолчанию
    //фиксированный размер окна
    setFixedSize(0, 0);

    ui->lineEditIpForEBCCS->setText("127.0.0.1");
    ui->lineEditIpForVideo->setText("127.0.0.1");

    ui->spinBoxPortForEBCCS->setValue(9760);
    ui->spinBoxPortForVideo->setValue(9758);
    ui->spinBoxPortForAudio->setValue(9759);
    ui->spinBoxPortForControl->setValue(9758);

    ui->checkBoxIsCustom->setChecked(false); //по умолчанию регистратор или нет
    ui->checkBox->setChecked(false); //по умолчанию elcbot или нет

    setCustom(ui->checkBoxIsCustom->checkState());

    connect(ui->checkBoxIsCustom,SIGNAL(stateChanged(int)), this,
    SLOT(setCustom(int)));

    connect(ui->pushButtonSave, SIGNAL(clicked()), this, SLOT(pushbuttonsaveclicked()));
    //действие на нажатие кнопки сохранить без закрытия
    connect(ui->pushButtonApply, SIGNAL(clicked()), this,
    SLOT(pushbuttonapplyclicked())); //действие на нажатие кнопки закрыть с сохранением
    connect(ui->pushButtonClose, SIGNAL(clicked()), this,
    SLOT(pushbuttoncloseclicked())); //действие на нажатие кнопки закрыт без сохранения
}

void FormSettings::blockimportantsettings(bool stat){

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						77
Изм	Лист	№ докум.	Подп.	Дата		

```

if(stat){
    ui->checkBox->setEnabled(false);
}
else{
    ui->checkBox->setEnabled(true);
}
}

//действие на сохранение
void FormSettings::pushbuttonsaveclicked(){
    gp.ipforebccs = ui->lineEditIpForEBCCS->text().toUtf8().constData();
    gp.ip = ui->lineEditIpForVideo->text().toUtf8().constData();

    gp.portforebccs = ui->spinBoxPortForEBCCS->value();
    gp.portforvideo = ui->spinBoxPortForVideo->value();
    gp.portforaudio = ui->spinBoxPortForAudio->value();
    gp.portforcontrol = ui->spinBoxPortForControl->value();

    gp.isRegistrarEnable = !ui->checkBoxIsCustom->checkState();

    gp.isElcBot = ui->checkBox->isChecked();

    //отправка сигнала с глобальными параметрами
    emit signalglobalparameters(gp);
}

void FormSettings::setCustom(int stat){
    if(stat){
        ui->frame_EBCC->setEnabled(false);
        ui->frame_CUSTOM->setEnabled(true);
    }
    else{
        ui->frame_EBCC->setEnabled(true);
        ui->frame_CUSTOM->setEnabled(false);
    }
}

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						78
Изм	Лист	№ докум.	Подп.	Дата		


```

    }
}

//действие на закрытие с сохранением
void FormSettings::pushbuttonapplyclicked(){
    pushbuttonsaveclicked(); //В начале сохраним
    pushbuttoncloseclicked(); //затем все закроем
}

//действие на закрытие
void FormSettings::pushbuttoncloseclicked(){
    emit signalclosesettingsform();
    this->close();
}

//действие на закрытие по сигналу ОС
void FormSettings::closeEvent(QCloseEvent *){
    pushbuttoncloseclicked();
}

void FormSettings::setCheckBox(int stat){
    qDebug("MOD CB: %s", std::to_string(stat).c_str());
}

FormSettings::~FormSettings()
{
    delete ui;
}

videocapture.cpp

#include "videocapture.h"
#include <QThread>

using namespace dp;

```

```

VideoCapture::VideoCapture(dp::NetworkTransmission * networktransmission)
{
    //получаем доступ, для обработки сигнал с NT
    this->networktransmission = networktransmission;

    isStart = false;
}

//Инициализация устройства
void VideoCapture::open(int id){
    cap.open(id); //id - номер устройства.

    if(cap.isOpened()){ //Если камера открыта, то отправляем сигнал об успешной
инициализации
        emit signalcameraisopen();
    }
    else{ //Или поток закрываем
        qDebug("Camera not available");
        QThread::currentThread()->quit();
    }
}

//Запускаем цикл захвата кадров, будет выполняться до запроса прерывани
void VideoCapture::start(){
    cv::Mat frame;

    if (!cap.isOpened()) {
        qDebug("Camera not available");
        return;
    }

    isStart = true;

```

```

qDebug("Starting capture video");
while(isStart){
    cap >> frame; //Захват кадра с устройства
    if(frame.empty()){
        QThread::currentThread()->quit();
        isStart = false;
    }
    else{
        if(networktransmission->isreadyaccept()){
            //отправляем кадр, если готовы, потом тут добавить сжатие можно
            networktransmission->sendimage(frame);
            qDebug("SEND");
        }

        emit signalshowimg(frame);

        if (QThread::currentThread()->isInterruptionRequested()){
            qDebug("interrupt");

            emit signalinterrupt();

            isStart = false;
        }
    }
}

//Закрытие камеры и потока
void VideoCapture::close(){
    cap.release();
    QThread::currentThread()->quit();
}

audiomanager.cpp

```

```

#include "audiomanager.h"
#include <QThread>

using namespace dp;

AudioManager::AudioManager(dp::NetworkTransmission * networktransmission)
{
    this->networktransmission = networktransmission;
    connect(networktransmission, SIGNAL(signalplaydata(QBuffer *)), this,
SLOT(playdata(QBuffer *)));

    input_buffer = new QBuffer;
    output_buffer = new QBuffer;
}

//Инициализация устройства
void AudioManager::open(){
    audioformat.setSampleRate(48000);
    audioformat.setChannelCount(1);
    audioformat.setSampleSize(8);
    audioformat.setCodec("audio/pcm");
    audioformat.setByteOrder(QAudioFormat::LittleEndian);
    audioformat.setSampleType(QAudioFormat::UnSignedInt);

    inputDevice = QAudioDeviceInfo::defaultInputDevice();
    outputDevice = QAudioDeviceInfo::defaultOutputDevice();

    //ВОТ ТУТ МОЖНО ОПРЕДЕЛИТЬ УСТРОЙСТВО

    qDebug("%s",QAudioDeviceInfo::availableDevices(QAudio::AudioInput).at(2).deviceName().
toUtf8().constData());

    audio_out = new QAudioOutput(outputDevice, audioformat, this);

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						82
Изм	Лист	№ докум.	Подп.	Дата		

```

input_buffer->open(QIODevice::ReadOnly);
output_buffer->open(QIODevice::WriteOnly);

audio_out->start(input_buffer);

if (inputDevice.isFormatSupported(audioformat))
{
    audio_in = new QAudioInput(inputDevice, audioformat, this);
    audio_in->setNotifyInterval(10);

    emit signalaudioisopen();
}
else{
    qDebug("default format not supported try to use nearest");
    audioformat = inputDevice.nearestFormat(audioformat);
    QThread::currentThread()->quit();
}
}

void AudioManager::datasend(){
    //qDebug("SEND %d",output_buffer->buffer().size());
    if(networktransmission->isreadyaccept()){
        //отправляем аудио кадр, если готов
        networktransmission->sendaudio(output_buffer);
    }
    if(output_buffer != nullptr){
        output_buffer->buffer().clear();
        output_buffer->seek(0);
    }
}

void AudioManager::playdata(QBuffer * buff){
    input_buffer->buffer().remove(0,input_buffer->pos());

    //так байты добавляем

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						83
Изм	Лист	№ докум.	Подп.	Дата		

```

//input_buffer.buffer().append('0');
//qDebug("PLAY %d",output_buffer->buffer().size());
const auto res = input_buffer->seek(0);
assert(res);
input_buffer->buffer().append(buff->buffer());
if(buff != nullptr){
    buff->buffer().clear();
    buff->seek(0);
}
}

//Запускаем цикл захвата кадров, будет выполняться до запроса прерывания
void AudioManager::start(){
    qDebug("Starting capture audio");
    audio_in->start(output_buffer);

    if(audio_in->state()){
        qDebug("Mic not available");
        close();
    }

    connect(audio_in,SIGNAL(notify()), this, SLOT(datasend()));
    emit signalaudiostart();
}

void AudioManager::mute(){
    audio_in->stop();
    if(output_buffer != nullptr){
        output_buffer->buffer().clear();
        output_buffer->seek(0);
    }
    emit signalaudiomute();
}

```

```

//Заккрытие камеры и потока
void AudioManager::close(){
    //закрыть захват звука
    audio_in->stop();
    audio_out->stop();

    if(audio_in != nullptr) delete audio_in;

    if(output_buffer != nullptr){
        output_buffer->buffer().clear();
        output_buffer->seek(0);

        delete output_buffer;
    }
    if(input_buffer != nullptr){
        input_buffer->buffer().clear();
        input_buffer->seek(0);
        //delete input_buffer;
    }

    QThread::currentThread()->quit();
}

controlmanager.cpp

#include "controlmanager.h"

using namespace dp;

ControlManager::ControlManager(dp::NetworkTransmission * networktransmission)
{
    this->networktransmission = networktransmission;
}

//отправка команды движения

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						85
Изм	Лист	№ докум.	Подп.	Дата		

```
void ControlManager::sendcommand(int command){
    networktransmission->sendcommandcontrol(command);
}
```

//для запуска движения, принятое NT

```
void ControlManager::startmovement(int command){
    switch (command) {
    case 0:
        startforward();
        break;
    case 1:
        startstop();
        break;
    case 2:
        startback();
        break;
    case 3:
        startleft();
        break;
    case 4:
        startright();
        break;

    }
}
```

```
void ControlManager::startforward(){
    qDebug("Press forward");
}
```

```
void ControlManager::startstop(){
    qDebug("Press stop");
}
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						86
Изм	Лист	№ докум.	Подп.	Дата		


```
void ControlManager::startback(){
    qDebug("Press back");
}
```

```
void ControlManager::startleft(){
    qDebug("Press left");
}
```

```
void ControlManager::startright(){
    qDebug("Press right");
}
```

networktransmission.cpp

```
#include <QDebug>
#include <QThread>
#include <string>
#include <cmath>
#include <ctime>
```

```
#include "networktransmission.h"
```

```
using namespace dp;
using std::string;
```

```
NetworkRecv::NetworkRecv(bool isElcBot){
    this->isElcBot = isElcBot;
}
```

```
NetworkRecv::NetworkRecv(int sockfd,struct sockaddr_in servaddr,
                           struct sockaddr_in cliaddr, bool isElcBot){
    this->sockfd = sockfd;
    this->servaddr = servaddr;
    this->cliaddr = cliaddr;
    this->isElcBot = isElcBot;
```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ИЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		87

```

    isCamEnable = false;
}

int NetworkRecv::recvimage(){
    int framecount = 0;
    int ptr=6;
    cv::Mat img;
    //new
    unsigned short int heightframe, widthframe, np, ncp;
    unsigned short int numberoldpackage = 0;
    //если новый пакет меньше или равен старому то рисуем кадр
    heightframe = widthframe = np = ncp = 0;
    //unsigned long countemptydata = 0; //пустые данные для рис овки разрыва

    unsigned int currentposy,currentposx;
    currentposx = currentposy = 0;

    while(!QThread::currentThread()->isInterruptRequested()){
        timeval timeout;
        timeout.tv_sec = 0;
        timeout.tv_usec = 0;
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(sockfd, &fds);
        int retval = select(sockfd+1, &fds, NULL, NULL, &timeout);

        if(1 == retval) {
            if(isElcBot){
                len = sizeof(cliaddr);
                if(-1 == (numbytes = recvfrom(sockfd,(unsigned char
*)buffersizerecvV,PACKAGESIZE_V,MSG_WAITALL,(struct sockaddr *) &cliaddr, &len))){
                    perror("recvfrom");
                    emit signalclosenetworktransmission();
                }
            }
        }
    }
}

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						88
Изм	Лист	№ докум.	Подп.	Дата		

```

        return 1;
    }

    //qDebug("Server DATA: %s\n", buffersizerecv);
    //qDebug("RECV DATA");
} //получаем данные для талкателя
else{
    len = sizeof(servaddr);
    if(-1 == (numbytes = recvfrom(sockfd, (unsigned char *)buffersizerecvV,
    PACKAGE_SIZE_V, MSG_WAITALL, (struct sockaddr *) &servaddr, &len))){
        perror("recvfrom");
        emit signalclosenetworktransmission();
        return 1;
    }

    //qDebug("Client DATA: %s\n", buffersizerecv);

}
buffersizerecvV[numbytes] = '\0';
if(!strncmp(buffersizerecvV, "INIT", 4)){
    //йой
    qDebug("VIDEO INIT CLIENT IP:POPRT");
    emit signalsetclientaddress(cliaddr, 0);
}
else{
    np = ((unsigned char)buffersizerecvV[0] << 8) | (unsigned
char)buffersizerecvV[1];
    ncp++;
    heightframe = ((unsigned char)buffersizerecvV[2] << 8) | (unsigned
char)buffersizerecvV[3];
    widthframe = ((unsigned char)buffersizerecvV[4] << 8) | (unsigned
char)buffersizerecvV[5];
    //np--;
    //qDebug("%d" ,np);

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						89
Изм	Лист	№ докум.	Подп.	Дата		

```

if(numberoldpackage >= np){
    //qDebug("NEW FRAME");

    //если изменился размер в процессе, то обновим его
    if(heightframe != img.rows){
        img = cv::Mat::zeros(heightframe,widthframe, CV_8UC3);
    }
    if(widthframe != img.cols){
        img = cv::Mat::zeros(heightframe,widthframe, CV_8UC3);
    }
    //numberoldpackage = 0;
    framecount++;
}
numberoldpackage = np;

//emit signalsetclientaddress(cliaddr);

//если изображения нет, то создадим его
if(img.empty()){
    img = cv::Mat::zeros(heightframe,widthframe, CV_8UC3);
}
currentposy = ((np) * ((PACKAGESIZE_V - 6) / 3)) / widthframe;
currentposx = (((np) * ((PACKAGESIZE_V - 6) / 3))) - (currentposy *
widthframe);

//начинаем с 6 байта, так как первые 6 байт в массиве служебка
ptr = 6;

//Восстановление изображения из массива байт
while(ptr < numbytes){
    if(currentposy < heightframe){
        if(currentposx < widthframe){

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						90
Изм	Лист	№ докум.	Подп.	Дата		

```

img.at<cv::Vec3b>(currentposy,currentposx) =
cv::Vec3b(buffer sizerecvV[ptr+ 0],buffer sizerecvV[ptr+1],buffer sizerecvV[ptr+2]);
ptr+=3;
currentposx++;
}
else{
currentposx = 0;

currentposy++;
}
}
else{
currentposy = currentposx = 0;
break;
}

}
}

```

//что же она делает? Она разрешает отправлять кадры, если они есть, если она будет

//отключена, то видео даже после отправки сигнала отправить кадр, мы не получим

```

isCamEnable = true;
//countemptydata = 0 ;
}
else{ //отключаем камеру, если включена
if(isCamEnable){
isCamEnable = false;
}
//countemptydata = 0 ;
}

```

```

if(framecount >= 1){

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						91
Изм	Лист	№ докум.	Подп.	Дата		

```

        framecount = 0;
        //qDebug("GET COUNT PAC: %d", ncp);
        ncp = 0;
        //qDebug("GET 24 Frame");
        if(!img.empty()) emit signalshowimg(img);
    }

}

/*
//получаем данные для слушателя

emit signalshowimg(frame); - типа как-то так после получения
*/

return 0;
}

int NetworkRecv::recvaudio(){
    buff = new QBuffer;
    buff->open(QIODevice::ReadWrite);

    while(!QThread::currentThread()->isInterruptionRequested()){
        timeval timeout;
        timeout.tv_sec = 0;
        timeout.tv_usec = 0;
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(sockfd, &fds);

        int retval = select(sockfd+1, &fds, NULL, NULL, &timeout);

        if(1 == retval) {
            if(isElcBot){
                len = sizeof(cliaddr);

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		92

```

        if(-1 == (numbytes = recvfrom(sockfd,(unsigned char
*)buffersizerecvA,PACKAGE_SIZE_A,MSG_WAITALL,(struct sockaddr *) &cliaddr, &len))) {
            perror("recvfrom");
            emit signalcloseNetworkTransmission();
            return 1;
        }
        //qDebug("Server DATA: %s\n", buffersizerecv);
        //qDebug("RECV DATA");
    } //получаем данные для талкателя
    else {
        len = sizeof(servaddr);
        if(-1 == (numbytes = recvfrom(sockfd, (unsigned char *)buffersizerecvA,
PACKAGE_SIZE_A, MSG_WAITALL, (struct sockaddr *) &servaddr, &len))) {
            perror("recvfrom");
            emit signalcloseNetworkTransmission();
            return 1;
        }

        //qDebug("Client DATA: %s\n", buffersizerecv);

    }
    buffersizerecvA[numbytes] = '\0';

    if(!strcmp(buffersizerecvA, "INIT", 4)) {
        //йоу
        qDebug("AUDIO INIT CLIENT IP:POPRT");
        emit signalsetClientAddress(cliaddr, 1);
    }
    else {
        //тут обработка пакета аудио
        buff->buffer().append(buffersizerecvA);
        emit signalplaydata(buff);
    }
}

```

```

    }
    return 0;
}

//подключаемся к регистратору
int NetworkRecv::connectregistrarsock(int sock, struct sockaddr_in servaddrforcontrol){
    qDebug("WAIT TCP REGISTRAR CONNECT");
    if (::connect(sock, (struct sockaddr *)&servaddrforcontrol, sizeof(servaddrforcontrol)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    qDebug("OK TCP REGISTRAR CONNECT");
    registrarsock = sock;
    emit signalserverenable();

    return 0;
}

//ожидаем подключения от клиента
int NetworkRecv::acceptsock(int server_fd, int sock, int addrlen, struct sockaddr_in
servaddrforcontrol){

    qDebug("WAIT TCP ACCEPT");

    if ((sock = accept(server_fd, (struct sockaddr *)&servaddrforcontrol,
(socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    qDebug("OK TCP ACCEPT");
    this->sock = sock;
    emit signalsetsock(sock);

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						94
Изм	Лист	№ докум.	Подп.	Дата		


```

    return 0;
}

//подключаеммся к elcbot
int NetworkRecv::connectsock(int sock, struct sockaddr_in servaddrforcontrol){
    qDebug("WAIT TCP CONNECT");
    if (::connect(sock, (struct sockaddr *)&servaddrforcontrol, sizeof(servaddrforcontrol)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    qDebug("OK TCP CONNECT");

    emit signalserverenable();

    return 0;
}

//получаем данные с регистратора
int NetworkRecv::recvregistrardata(){
    qDebug("RECV START");
    int numbytes;
    while(!QThread::currentThread()->isInterruptionRequested()){
        timeval timeout;
        timeout.tv_sec = 0;
        timeout.tv_usec = 0;
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(registrarsock, &fds);
        int retval = select(registrarsock+1, &fds, NULL, NULL, &timeout);
        if(1 == retval) {
            if(-1 == (numbytes = read(registrarsock, buffersizerecvR, PACKAGESIZE_R))){
                perror("recvfrom");
            }
        }
    }
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						95
Изм	Лист	№ докум.	Подп.	Дата		

```

        emit signalclosenetworktransmission();
        return 1;
    }
    buffersizerecvR[numbytes] = '\0';

    emit signalrecvregistrardata(buffersizerecvR, numbytes);
}
}
return 0;
}

//получаем команды от клиента
int NetworkRecv::recvcontrol(){
    qDebug("RECV START");
    int numbytes;
    int commandcontrol = -1;

    while(!QThread::currentThread()->isInterruptionRequested()){
        timeval timeout;
        timeout.tv_sec = 0;
        timeout.tv_usec = 0;
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(sock, &fds);
        int retval = select(sock+1, &fds, NULL, NULL, &timeout);

        if(1 == retval) {
            if(-1 == (numbytes = read( sock , buffersizerecvC, PACKAGE_SIZE_C))){
                perror("recvfrom");
                emit signalclosenetworktransmission();
                return 1;
            }
            buffersizerecvC[numbytes] = '\0';

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						96
Изм	Лист	№ докум.	Подп.	Дата		

```

if(numbytes > 2){
    if(((char)0xF0 == buffersizerecvC[0]) && ((char)0xEE == buffersizerecvC[1])){
        switch (buffersizerecvC[2]) {
            case (char)0xA0: //вперед
                commandcontrol = 0;
                break;
            case (char)0xA1: //стоп
                commandcontrol = 1;
                break;
            case (char)0xA2: //назад
                commandcontrol = 2;
                break;
            case (char)0xA3: //лево
                commandcontrol = 3;
                break;
            case (char)0xA4: //право
                commandcontrol = 4;
                break;
        }
        emit signalstartmovement(commandcontrol);
    }
}

return 0;
}

void NetworkRecv::closeaudiothread(){
    if(buff != nullptr){
        buff->buffer().clear();
        buff->seek(0);

        delete buff;
    }
}

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		97

```

    }
}

NetworkTransmission::NetworkTransmission()
{
    isReadConnect = false;
    isReadConnectV = false;
    isReadConnectA = false;

    isTCPConnect = false;
    isTCPRegistrarConnect = false;

    countframe = 0;
}

//Применение параметров для сети
void NetworkTransmission::setglobalparameters(globalparameters gp){
    registrarsock = 0;
    this->gp = gp;

    if(gp.isRegistrarEnable){
        qDebug("NT: set global parameters for registrar");
        qDebug("NT: IP for registrar: %s, PORT: %d", gp.ipforebcs.c_str(), gp.portforebcs);

        servaddrforregistrar.sin_family = AF_INET;
        servaddrforregistrar.sin_port = htons(gp.portforebcs);

        if(inet_pton(AF_INET, gp.ipforebcs.c_str(), &servaddrforregistrar.sin_addr)<=0)
        {
            printf("\nInvalid servaddrforregistrar/ servaddrforregistrar not supported \n");
            emit signalclosenetworktransmission();
            return;
        }
    }
}

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						98
Изм	Лист	№ докум.	Подп.	Дата		

```

else{
    qDebug("NT: set global parameters");
    qDebug("NT: IP for video: %s, port video: %d, port audio: %d, port control: %d",
        gp.ip.c_str(), gp.portforvideo, gp.portforaudio, gp.portforcontrol);

    sock = 0;

    //тут мы выполняем первоначальную настройку, в зависимости от того кто хост
    memset(&servaddrforvideo, 0, sizeof (servaddrforvideo));
    memset(&servaddrforaudio, 0, sizeof (servaddrforaudio));
    servaddrforcontrol.sin_family = AF_INET;
    servaddrforcontrol.sin_port = htons(gp.portforcontrol);
    if(gp.isElcBot){
        qDebug("NT: This Elc Bot");
        //tcp
        opt = 1;
        addrlenforcontrol = sizeof (servaddrforcontrol);

        servaddrforcontrol.sin_addr.s_addr = INADDR_ANY;

        //udp
        memset(&cliaddrforvideo, 0, sizeof (cliaddrforvideo));
        memset(&cliaddrforaudio, 0, sizeof (cliaddrforaudio));
        //указываем порт, протокол и ip для работы слушателя
        servaddrforvideo.sin_family = AF_INET;
        servaddrforvideo.sin_addr.s_addr = INADDR_ANY;
        servaddrforvideo.sin_port = htons(gp.portforvideo);
        //аудио
        servaddrforaudio.sin_family = AF_INET;
        servaddrforaudio.sin_addr.s_addr = INADDR_ANY;
        servaddrforaudio.sin_port = htons(gp.portforaudio);
    }
    else{
        qDebug("NT: This not Elc Bot");
    }
}

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						99
Изм	Лист	№ докум.	Подп.	Дата		

```

//TCP
// Преобразование адресов IPv4 и IPv6 из текста в двоичную форму
if(inet_pton(AF_INET, gp.ip.c_str(), &servaddrforcontrol.sin_addr)<=0)
{
    printf("\nInvalid servaddrforcontrol/ servaddrforcontrol not supported \n");
    emit signalclosenetworktransmission();
    return;
}

//UDP
//указываем порт, протокол и ip для работы талкателя
servaddrforvideo.sin_family = AF_INET;
servaddrforvideo.sin_port = htons(gp.portforvideo);
servaddrforvideo.sin_addr.s_addr = inet_addr(gp.ip.c_str());
//аудио
servaddrforaudio.sin_family = AF_INET;
servaddrforaudio.sin_port = htons(gp.portforaudio);
servaddrforaudio.sin_addr.s_addr = inet_addr(gp.ip.c_str());
}

//сигнал о получение параметров, говорим main, что можно запускать
}

isElcBot = gp.isElcBot;
isRegistrar = gp.isRegistrarEnable;

emit signalglobalparametersreceived();
}

//получаем адрес клиента, когда тот подключится
void NetworkTransmission::getclientaddress(struct sockaddr_in cliaddr, int uc){
    //тут надо будет определить для видео и аудио
    switch (uc) {
    case 0:
        this->cliaddrforvideo = cliaddr;
        isReadConnectV = true;

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						100
Изм	Лист	№ докум.	Подп.	Дата		

```

        break;
    case 1:
        this->cliaddrforaudio = cliaddr;
        isReadConnectA = true;
        break;
    }
    if(isReadConnectA && isReadConnectV) connectionestablished();
}

//Запуск связи вызывается дважды!
void NetworkTransmission::startsystemnetworktransmission(){
    //тут запускаем сисиему сетевой передачи и начинаем передавать данные,
    //в зависимости от того, кто сервер кто клиент в разговоре
    qDebug("NT: NT IS STARTING");

    //тут две инициализации, одна для подключения и запроса данных от EBCC_S и от
    EBCC_C
    if(isRegistrar){
        qDebug("NT: NT REGISTRAR STARTING");
        networkRecvforregistrar = new NetworkRecv(isElcBot);
        networkrecvforregistrarthread = new QThread();
        networkRecvforregistrar->moveToThread(networkrecvforregistrarthread);

        if ((registrarsock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
            printf("\n Socket creation error \n");
            emit signalclosenetworktransmission();
            return;
        }

        //получаем, информацию что поток запущен и можно получать данные
        connect(networkrecvforregistrarthread, SIGNAL(started()), this, SLOT(connectsock()));
        connect(this, SIGNAL(signalconnectregistrarsock(int, struct sockaddr_in)),
            networkRecvforregistrar, SLOT(connectregistrarsock(int, struct sockaddr_in)));

        //попытка подключиться

```

					ВКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						101
Изм	Лист	№ докум.	Подп.	Дата		

```

        connect(networkRecvforregistrar, SIGNAL(signalserverenable()), this,
        SLOT(tcpserverenable()));

        connect(networkRecvforregistrar, SIGNAL(signalserverenable()),
        networkRecvforregistrar, SLOT(recvregistrardata()));

        //получаем данные от регистратора если есть и дальше их парсим
        connect(networkRecvforregistrar, SIGNAL(signalrecvregistrardata(unsigned char *,
        int)), this, SLOT(recvregistrardata(unsigned char *, int)));

        networkrecvforregistrarthread->start();
    }
    else{
        qDebug("NT: NT EBCC STARTING");
        networkRecvforcontrol = new NetworkRecv(isElcBot);
        networkrecvforcontrolthread = new QThread();
        networkRecvforcontrol->moveToThread(networkrecvforcontrolthread);

        if(isElcBot){
            if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
            {
                perror("socket failed");
                emit signalclosenetworktransmission();
                return;
            }

            // Принудительное подключение сокета к порту
            if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
            &opt, sizeof(opt)))
            {
                perror("setsockopt");
                emit signalclosenetworktransmission();
                return;
            }
        }
    }

```



```

// Принудительное подключение сокета к порту
if (bind(server_fd, (struct sockaddr *)&servaddrforcontrol,
sizeof(servaddrforcontrol))<0)
{
    perror("bind failed");
    emit signalclosenetworktransmission();
    return;
}
if (listen(server_fd, 3) < 0)
{
    perror("listen");
    emit signalclosenetworktransmission();
    return;
}
connect(networkrecvforcontrolthread, SIGNAL(started()), this, SLOT(acceptsock()));
connect(this, SIGNAL(signalacceptsock(int, int, int, struct sockaddr_in)),
networkRecvforcontrol, SLOT(acceptsock(int, int, int, struct sockaddr_in))); //Ожидаем
подключения
}
else{
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        emit signalclosenetworktransmission();
        return;
    }
    connect(this, SIGNAL(signalconnectsock(int, struct sockaddr_in)),
networkRecvforcontrol, SLOT(connectsock(int, struct sockaddr_in))); //попытка подключиться
    connect(networkrecvforcontrolthread, SIGNAL(started()), this,
SLOT(connectsock()));
}
connect(networkRecvforcontrol, SIGNAL(signalserverenable()), this,
SLOT(tcpserverenable()));

```

```

        connect(networkRecvforcontrol, SIGNAL(signalsetsock(int)), this,
        SLOT(getsock(int)));
        connect(networkRecvforcontrol, SIGNAL(signalsetsock(int)), networkRecvforcontrol,
        SLOT(recvcontrol()));
        connect(networkRecvforcontrol, SIGNAL(signalstartmovement(int)), this,
        SLOT(startmovement(int)));
        networkrecvforcontrolthread->start();

//создаём дескриптор файла для UDP
if ((sockfdvideo = socket(AF_INET, SOCK_DGRAM, 0)) == - 1) {
    perror("listener: socket");
    emit signalclosenetworktransmission();
    return;
}

if ((sockfdaudio = socket(AF_INET, SOCK_DGRAM, 0)) == - 1) {
    perror("listener: socket");
    emit signalclosenetworktransmission();
    return;
}

int size_write = 9 * 1024 * 1024;

if(setsockopt(sockfdvideo,SOL_SOCKET,SO_SNDBUF,&size_write,sizeof(size_write)) == -
1){
    emit signalclosenetworktransmission();
    return;
}

size_write = 9 * 1024 * 1024;

if(setsockopt(sockfdaudio,SOL_SOCKET,SO_RCVBUF,&size_write,sizeof(size_write)) == -
1){

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		104

```

        emit signalclosenetworktransmission();
        return;
    }

    //Во время загрузки проверяем, elcbot ли это, для того
    //чтобы определить сервер, который будет ждать входящих подключений
    if(isElcBot){
        //наносим на порт
        if (bind(sockfdvideo,(const struct sockaddr *)&servaddrforvideo,
sizeof(servaddrforvideo)) == - 1) {
            //если ошибка, все закрываем, ну не смогли мы открыть порт
            perror("listener: bind");
            emit signalclosenetworktransmission();
            return;
        }

        if (bind(sockfdaudio,(const struct sockaddr *)&servaddrforaudio,
sizeof(servaddrforaudio)) == - 1) {
            //если ошибка, все закрываем, ну не смогли мы открыть порт
            perror("listener: bind");
            emit signalclosenetworktransmission();
            return;
        }
    }

    //Создаем отдельные потоки для приема сообщений видео.
    networkRecvforvideo = new NetworkRecv(sockfdvideo, servaddrforvideo,
cliaddrforvideo, isElcBot);

    networkrecvforvideothread = new QThread();
    networkRecvforvideo->moveToThread(networkrecvforvideothread);

    networkRecvforaudio = new NetworkRecv(sockfdaudio, servaddrforaudio,
cliaddrforaudio, isElcBot);

```

```

networkrecvforaudiothread = new QThread();
networkRecvforaudio->moveToThread(networkrecvforaudiothread);

connect(this, SIGNAL(signalcloseaudiothread()), networkRecvforaudio,
SLOT(closeaudiothread()));

if(isElcBot) {
    //Говорим серверу, чтоб тот пакетики принимал
    //тут у нас ожидается ip клиента для сервера, без него стартовать не будем
    //timeout - Важно
    connect(networkRecvforvideo, SIGNAL(signalsetclientaddress(struct sockaddr_in,
int)), this, SLOT(getclientaddress(struct sockaddr_in, int)));
    connect(networkRecvforaudio, SIGNAL(signalsetclientaddress(struct sockaddr_in,
int)), this, SLOT(getclientaddress(struct sockaddr_in, int)));
    recv();
}
else{
    std::string str = "INIT";

    if(-1 == sendto(sockfdvideo, (const char *)str.c_str(), strlen(str.c_str()),
MSG_CONFIRM, (const struct sockaddr *) &servaddrforvideo,
sizeof(servaddrforvideo))){
        perror("talker: sendto");
        emit signalclosenetworktransmission();
        return;
    }
    if(-1 == sendto(sockfdaudio, (const char *)str.c_str(), strlen(str.c_str()),
MSG_CONFIRM, (const struct sockaddr *) &servaddrforaudio,
sizeof(servaddrforaudio))){
        perror("talker: sendto");
        emit signalclosenetworktransmission();
        return;
    }
    //ну а здесь у нас прием пакетов клиентом, ну комментировать нечего

```

```

        recv();
        connectionestablished();
    }
}

void NetworkTransmission::recvregistrardata(unsigned char * data, int sizedata){
    qDebug("Registrar data recv");
    //globalparameters gp;
    string id = "";
    unsigned char mode = 0xFE;
    unsigned short int porttype = 0;
    string ip, portvideo, portaudio, portcontrol;

    gp.isRegistrarEnable = false;
    gp.isElcBot = isElcBot;
    if(sizedata > 0){
        mode = data[0];
        switch (mode) {
            case 0xD0:
                for(int currentbyte = 1; currentbyte < sizedata; currentbyte++){
                    id.push_back(data[currentbyte]);
                }
                //запускаем подключения к elcbot, если пришла команда что надо подключаться
                //выводим ID и тд, все что нужно
                emit signalshowid(id);
                break;
            case 0xD1:
                porttype = 0;
                for(int currentbyte = 1; currentbyte < sizedata; currentbyte++){
                    if(0xEF == data[currentbyte]){
                        porttype++;
                        currentbyte++;
                    }
                }

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						107
Изм	Лист	№ докум.	Подп.	Дата		

```

        if(0 == porttype){
            ip.push_back(data[currentbyte]);
        }
        else if(1 == porttype){
            portvideo.push_back(data[currentbyte]);
        }
        else if(2 == porttype){
            portaudio.push_back(data[currentbyte]);
        }
        else if(3 == porttype){
            portcontrol.push_back(data[currentbyte]);
        }
    }
    struct in_addr remoteip;
    remoteip.s_addr = stoi(ip);

    gp.ip = inet_ntoa(remoteip);
    gp.portforvideo = stoi(portvideo);
    gp.portforaudio = stoi(portaudio);
    gp.portforcontrol = stoi(portcontrol);
    break;
case 0xD2:
    //разрыв
    //emit signalclosenetworktransmission();
    mode = 0xFE;
    break;
}

//qDebug("data: %s, %i",data,sizedata);
//временное решение
if(mode != 0xFE) setglobalparameters(gp);
}
else{

```

					БКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						108
Изм	Лист	№ докум.	Подп.	Дата		

```

        qDebug("Error get data");
    }
}

void NetworkTransmission::startmovement(int command){
    emit signalstartmovement(command);
}

void NetworkTransmission::tcpserverenable(){
    if(isRegistrar){
        isTCPRegistrarConnect = true;
        //отправляем данные регистратору
        sendDataForRegistrar();
    }
    else{
        isTCPConnect = true;
    }
}

void NetworkTransmission::getsock(int sock){
    this->sock = sock;
    tcpserverenable();
}

int NetworkTransmission::acceptsock(){
    emit signalacceptsock(server_fd, sock, addrlenforcontrol, servaddrforcontrol);
    return 0;
}

int NetworkTransmission::connectsock(){
    if(isRegistrar){
        emit signalconnectregistrarsock(registrarsock, servaddrforregistrar);
    }
    else{

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						109
Изм	Лист	№ докум.	Подп.	Дата		

```

        emit signalconnectsock(sock, servaddrforcontrol);
    }
    return 0;
}

```

//Действие на остановку соединения, например, когда был принудительный запрос от пользователя или потеря соединения

```
void NetworkTransmission::stoppingtheconnection(){
```

```
    //тут делаем все то, что надо для остановки
```

```
    isReadConnect = false;
```

```
    isReadConnectV = false;
```

```
    isReadConnectA = false;
```

```
    isTCPConnect = false;
```

```
    isTCPRegistrarConnect = false;
```

```
    //закрываем сокет
```

```
    close(sockfdvideo);
```

```
    close(sockfdaudio);
```

```
    close(server_fd);
```

```
    close(registrarsock);
```

```
    close(sock);
```

```
    //Говорим, что все готово и закрываем соединение
```

```
    connectionstopped();
```

```
}
```

//Действие на успешное соединение

```
void NetworkTransmission::connectionestablished(){
```

```
    qDebug("NT: NT IS START OR UPDATE");
```

```
    //ставим это когда будем готовы принимать картинки и отправлять
```

```
    countframe = 0; //счетчик для отправленных
```

```
    isReadConnect = true;
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						110
Изм	Лист	№ докум.	Подп.	Дата		


```

        //отправка сигнала успешного соединения
        emit signalconnectionestablished();
    }

    //Для службы видеозахвата, проверяет доступна ли отправка изображений
    int NetworkTransmission::isreadyaccept(){
        return isReadConnect;
    }

    //тут для запуска потока получения пакетов
    int NetworkTransmission::recv(){
        qDebug("T=%d",QThread::idealThreadCount());
        networkrecvforvideothread->start();
        networkrecvforaudiothread->start();
        connect(networkrecvforvideothread, SIGNAL(started()), networkRecvforvideo,
        SLOT(recvimage())); //открываем камеру
        connect(networkrecvforaudiothread, SIGNAL(started()), networkRecvforaudio,
        SLOT(recvaudio())); //открываем прием звука

        connect(networkRecvforvideo, SIGNAL(signaldisablecamera()), this,
        SLOT(disablecamera()));
        //возможно надо принимать команду отключения микрофона
        //рисуем кадр на экране
        connect(networkRecvforvideo, SIGNAL(signalshowimg(cv::Mat)), this,
        SLOT(sendimagetoplayer(cv::Mat)));

        connect(networkRecvforaudio, SIGNAL(signalplaydata(QBuffer *)), this,
        SLOT(playdata(QBuffer *)));
        return 0;
    }

    void NetworkTransmission::sendimagetoplayer(cv::Mat frame){
        emit signalshowimg(frame); //ВОТ ОНА! РИСОВКА КАДРА!
    }

```

```

}

void NetworkTransmission::playdata(QBuffer * buff){
    emit signalplaydata(buff);
}

void NetworkTransmission::disablecamera(){
    //если удаленно отключили камеру, вешаем банер
    emit signalremotedisablecamera();
}

//функция отправки изображения на удаленный ресурс
int NetworkTransmission::sendimage(cv::Mat frame){
    //send send send send
    //stoppingtheconnection();
    //что-то новое, неизведанное
    imgSize = frame.total() * frame.elemSize(); //количество байт изображения
    countpackage = ceil(static_cast<float>(imgSize) / PACKAGESIZE_V);

    int height, width;

    unsigned short int heightframe, widthframe, ncp;

    height = width = ncp = 0;

    heightframe = frame.rows;
    widthframe = frame.cols;

    //ширина и высота записанная в 4 разных байтах
    uchar hf, hs, wf, ws, npf, nps;

    //делим
    hf = ((unsigned short)heightframe >> 8);

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						112
Изм.	Лист	№ докум.	Подп.	Дата		

```

hs = ((unsigned short)heightframe >> 0);
wf = ((unsigned short)widthframe >> 8);
ws = ((unsigned short)widthframe >> 0);

//количество пакетов с служебкой и данными картинки в 3 канала, для передачи 1
изображения,
//учитывается на 1 пакет: 6 байт служебной информации и сами данные, размер
//пакета можно менять в define, а может потом и дам возможность менять в
настройках
countpackage = ceil( static_cast<float>(countpackage * 6 + imgSize) /
PACKAGESIZE_V);

//номер пикселя для буфера
int ptr = 6;
for(unsigned int numberpackage = 0; numberpackage <= countpackage;
numberpackage++){

    if(ptr >= PACKAGESIZE_V){
        //самая опасная группировка
        if(isElcBot){ //отправляем данные от слушателя
            len = sizeof(cliaddrforvideo);
            if(isReadConnect){
                if(-1 == (sendto(sockfdvideo, (unsigned char *) bufferingsizesend, ptr,
MSG_CONFIRM, (const struct sockaddr *) &cliaddrforvideo, len))){
                    perror("talker: sendto");
                    emit signalclosenetworktransmission();
                    return 1;
                }
            }
        }
        else{ //отправляем данные от талкателя
            len = sizeof(servaddrforvideo);
            if(isReadConnect){

```

```

        if(-1 == sendto(sockfdvideo, (unsigned char *) bufferimgsizesend, ptr,
MSG_CONFIRM, (const struct sockaddr *) &servaddrforvideo, len)){
            perror("talker: sendto");
            emit signalclosenetworktransmission();
            return 1;
        }
    }
}

ptr = 6;
ncp++;
}

npf = ((unsigned short)numberpackage >> 8);
nps = ((unsigned short)numberpackage >> 0);

//поменять надо будет, не нравится мне такой стиль
bufferimgsizesend[0] = npf;
bufferimgsizesend[1] = nps;
bufferimgsizesend[2] = hf;
bufferimgsizesend[3] = hs;
bufferimgsizesend[4] = wf;
bufferimgsizesend[5] = ws;

while((ptr < PACKAGESIZE_V) && (height < frame.rows)){
    while((ptr < PACKAGESIZE_V) && (width < frame.cols)){
        //бежим по полной картинке, и заполняем буфер для пакета
        bufferimgsizesend[ptr + 0] = frame.at<cv::Vec3b>(height,width).val[0];
        bufferimgsizesend[ptr + 1] = frame.at<cv::Vec3b>(height,width).val[1];
        bufferimgsizesend[ptr + 2] = frame.at<cv::Vec3b>(height,width).val[2];
        ptr += 3;
        width++;
    }
}

```

```

        if(ptr < PACKAGE_SIZE_V) {
            height++;
            width = 0;
        }
    }
}

countframe++;

//qDebug("SEND NCP: %d", ncp);

if(countframe >= 24){
    qDebug("SEND 24 Frame");
    countframe = 0;
}
return 0;
}

int NetworkTransmission::sendaudio(QBuffer * buff){
    if(isElcBot){ //отправляем данные от слушателя
        len = sizeof(cliaddrforaudio);
        if(isReadConnect){
            if(-1 == (sendto(sockfdaudio, (unsigned char *) buff->buffer().data(), buff->buffer().size(), MSG_CONFIRM, (const struct sockaddr *) &cliaddrforaudio, len))){
                perror("talker: sendto");
                emit signalclosenetworktransmission();
                return 1;
            }
        }
    }
    else{ //отправляем данные от талкателя
        len = sizeof(servaddrforaudio);
        if(isReadConnect){
            if(-1 == sendto(sockfdaudio, (unsigned char *) buff->buffer().data(), buff->buffer().size(), MSG_CONFIRM, (const struct sockaddr *) &servaddrforaudio, len))){

```

```

        perror("talker: sendto");
        emit signalcloseNetworkTransmission();
        return 1;
    }
}

return 0;
}

```

//отправка ip порта и тд регистратору

```
int NetworkTransmission::sendDataForRegistrar(){
```

```
    //здесь в зависимости от eclbot не eclbot формирование запроса
```

```
    uchar command;
```

```
    string id;
```

```
    unsigned char * data = nullptr;
```

```
    unsigned int datasize = 0;
```

```
    if(isElcBot){
```

```
        command = 0xD0;
```

```
        string portvideo = std::to_string(gp.portforvideo);
```

```
        string portaudio = std::to_string(gp.portforaudio);
```

```
        string portcontrol = std::to_string(gp.portforcontrol);
```

```
        datasize = sizeof (unsigned char) / sizeof (unsigned char) * 4 + portvideo.size() //4
```

количество EF и +1 байт действия

```
        + portaudio.size() + portcontrol.size(); //3 - это количество портов
```

```
        data = new unsigned char[datasize];
```

```
        data[0] = command;
```

```
        std::vector<unsigned char> tmpdataport;
```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						116
Изм	Лист	№ докум.	Подп.	Дата		

```

for(unsigned int portsize = 0; portsize < portvideo.size(); portsize++){
    tmpdataport.push_back(portvideo.at(portsize));
}
tmpdataport.push_back(0xEF);
for(unsigned int portsize = 0; portsize < portaudio.size(); portsize++){
    tmpdataport.push_back(portaudio.at(portsize));
}
tmpdataport.push_back(0xEF);
for(unsigned int portsize = 0; portsize < portcontrol.size(); portsize++){
    tmpdataport.push_back(portcontrol.at(portsize));
}
tmpdataport.push_back(0xEF);

for(unsigned int currentbyte = 1; currentbyte < datasize; currentbyte++){
    data[currentbyte] = tmpdataport.at(currentbyte-1);
}

tmpdataport.clear();
qDebug("DEBUG: UINT SIZE: %lu", sizeof (unsigned int));
}
else{
    bool bOk;
    QString str = QDialog::getText( 0, "ID", "Enter ID ElcBot:",
                                   QLineEdit::Normal, "", &bOk);

    if (!bOk) {
        // Была нажата кнопка Cancel
        command = 0xD2;
        datasize = sizeof (unsigned char) / sizeof (unsigned char);
        data = new unsigned char[datasize];
        data[0] = command;
    }
    else{ //id
        command = 0xD1;
        datasize = sizeof (unsigned char) / sizeof (unsigned char) + str.size();

```

```

        data = new unsigned char[datasize];
        data[0] = command;
        id = str.toUtf8().constData();
        for(unsigned int idsize = 1; idsize < datasize; idsize++){
            data[idsize] = id.at(idsize - 1);
        }
    }
}

if(isTCPRegistrarConnect){
    if(-1 == (send(registrarsock, data, datasize, 0 ))){
        perror("talker: sendto");
        emit signalclosenetworktransmission();
        return 1;
    }
    if(data != nullptr){
        datasize = 0;
        delete [] data;
    }
}
return 0;
}

int NetworkTransmission::sendcommandcontrol(int command){
    char data[3] = {0};

    data[0] = 0xF0; //говори о том что, данные без обратной связи
    data[1] = 0xEE; //говорит о том, что следующий байт - данные управления роботом

    if(isTCPConnect){
        switch (command) {
            case 0: //вперед
                data[2] = 0xA0;
                break;

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						118
Изм	Лист	№ докум.	Подп.	Дата		


```

case 1: //стоп
    data[2] = 0xA1;
    break;
case 2: //назад
    data[2] = 0xA2;
    break;
case 3: //лево
    data[2] = 0xA3;
    break;
case 4: //право
    data[2] = 0xA4;
    break;
}

if(-1 == (send(sock, data, strlen(data), 0 ))){
    perror("talker: sendto");
    emit signalcloseNetworkTransmission();
    return 1;
}
}
return 0;
}

//Действие на остановку соединения
void NetworkTransmission::connectionstopped(){
    //Завершаем поток, для возможного повторного создания потока
    //QThread::currentThread()->requestInterruption();

    if(networkRecvforvideo != nullptr){
        networkrecvforvideothread->requestInterruption(); //отправляем запрос на
        прерывание
        networkrecvforvideothread->quit();
        networkrecvforvideothread->wait(5000); //ожидаем завершения потока
    }
}

```

```

        delete networkrecvforvideothread;
        delete networkRecvforvideo;
    }
    if(networkRecvforaudio != nullptr){
        emit signalcloseaudiothread();
        networkrecvforaudiothread->requestInterruption(); //отправляем запрос на
прерывание
        networkrecvforaudiothread->quit();
        networkrecvforaudiothread->wait(5000); //ожидаем завершения потока

        delete networkrecvforaudiothread;
        delete networkRecvforaudio;
    }

    QThread::currentThread()->quit();
    //говорим main, что остановка соединения успешна завершена.
    //emit signalconnectionstopped();
    qDebug("NT: STOP NT");
}

```

Для сигнального сервера

main.cpp

```

#include "networktransmission.h"
#include "database.h"

int main(int argc, char const *argv[])
{
    dp::DataBase * dataBase;
    dp::NetworkTransmission * networkTransmission;

    dataBase = new dp::DataBase();
    networkTransmission = new dp::NetworkTransmission(dataBase);
}

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						120
Изм	Лист	№ докум.	Подп.	Дата		

```

networkTransmission->initialization();

delete networkTransmission;
dataBase->clear();
delete dataBase;
return 0;
}

networktransmission.cpp

#include "networktransmission.h"
#include <iostream>
#include <arpa/inet.h>

using namespace dp;

NetworkTransmission::NetworkTransmission(DataBase * database)
{
    dataProcessing = new DataProcessing(database);
}

NetworkTransmission::~NetworkTransmission(){
    stoppingtheconnection();

    if(dataCommunication != nullptr){
        delete dataCommunication;
    }
    delete dataProcessing;
}

void NetworkTransmission::initialization(){
    initialization(DEFAULTPORT);
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ИЗ)	Лист
						121
Изм	Лист	№ докум.	Подп.	Дата		

```

void NetworkTransmission::initialization(unsigned int port){
    std::cout << "[ WAIT ] Initialization, PORT: " << port << std::endl;
    setglobalparameters(port);
    startsystemnetworktransmission();
}

void NetworkTransmission::setglobalparameters(unsigned int port){
    opt = 1;
    addrlen = sizeof(address);

    this->port = port;
    std::cout << "[ OK ] Set global parameters" << std::endl;

}

void NetworkTransmission::startsystemnetworktransmission(){
    // Создание дескриптора файла сокета
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    std::cout << "[ OK ] Create socket" << std::endl;

    // Принудительное подключение сокета к порту
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                    &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		122

```

address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( port );

// Принудительное подключение сокета к порту
if (bind(server_fd, (struct sockaddr *)&address,
        sizeof(address))<0)
{
    perror("bind failed");
    exit(EXIT_FAILURE);
}

std::cout << "[ OK ] Bind " << std::endl;

if (listen(server_fd, 3) < 0)
{
    perror("listen");
    exit(EXIT_FAILURE);
}

dataCommunication = new dp::DataCommunication(server_fd, sock, address, addrlen);
std::cout << "[ OK ] Initialization" << std::endl;

struct in_addr ca;
unsigned int numbytes;
while(true){
    buff = nullptr;
    std::cout << "[ WAIT ] Accept" << std::endl;
    remoteip = dataCommunication->accept().sin_addr.s_addr;
    ca.s_addr = remoteip;
    numbytes = 0;
    std::cout << "[ OK ] Accept, client ip: " << inet_ntoa(ca) << std::endl;
    sizebuff = 0;
    std::cout << "[ WAIT ] Recv and parsing data" << std::endl;
    buff = dataProcessing->commandProcessing(remoteip, dataCommunication-
>recvData(numbytes), numbytes, sizebuff);
    std::cout << "[ OK ] Recv and parsing data" << std::endl;
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		123

```

std::cout << "[ WAIT ] Send data" << std::endl;
dataCommunication->sendData(buff, sizebuff);
std::cout << "[ OK ] Send data" << std::endl;
if(buff != nullptr){
    delete[] buff;
}
}
}

```

```

void NetworkTransmission::stoppingtheconnection(){
    close(server_fd);
    close(sock);
}

```

dataprocessing.cpp

```

#include "dataprocessing.h"
#include <string>
#include <vector>

```

```

using namespace dp;
using namespace std;

```

```

DataProcessing::DataProcessing(DataBase * dataBase)
{
    this->dataBase = dataBase;
}
DataProcessing::~DataProcessing(){

}

```

```

unsigned char * DataProcessing::commandProcessing(int ip, unsigned char * getbuff,
unsigned int &numbytes, unsigned int &sizebuff){
    sizebuff = 2;
    //printf("DATA: %s\n",getbuff);
}

```

					BKP-ИГТУ-09.03.01-(16-B-2)-002-2020(ИЗ)	Лист
						124
Изм	Лист	№ докум.	Подп.	Дата		

```

unsigned int datasize;
unsigned short int mode = 0;
unsigned short int porttype = 0;
string strip;
string portvideo, portaudio, portcontrol;
string id = "";
std::vector<char> tmpdata;
if(numbytes > 0){
    switch (getbuff[0]) { //в этом месте парсеры
    case 0xD0: //тут elcbot требует регистрации
        //регистрируем
        mode = 1;
        porttype = 0;

        for(unsigned int currentbyte = 1; currentbyte < numbytes; currentbyte++){
            if(0xEF == getbuff[currentbyte]){
                porttype++;
                currentbyte++;
            }

            if(0 == porttype){
                portvideo.push_back(getbuff[currentbyte]);
            }
            else if(1 == porttype){
                portaudio.push_back(getbuff[currentbyte]);
            }
            else if(2 == porttype){
                portcontrol.push_back(getbuff[currentbyte]);
            }
        }

        printf("[ MESS ] Get port video: %s, port audio: %s, port control: %s\n",
portvideo.c_str(), portaudio.c_str(), portcontrol.c_str());

        //portvideo portaudio portcontrol - обрабатываем и даем ID

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						125
Изм	Лист	№ докум.	Подп.	Дата		

```

        //назначаем ID
        //добавить запись ip в БД
        id = "sdfYYY2"; //тут заполнение

        break;
case 0xD1: //просто ID приходит
    for(unsigned int currentbyte = 1; currentbyte < numbytes; currentbyte++){
        id.push_back(getbuff[currentbyte]);
    }
    //id -работаем с данными
    printf("[ MESS ] Get ID: %s\n", id.c_str());
    //ищем и даем ports ip
    mode = 2;
    strip = to_string(ip); //тут ip из бд
    portvideo = "9758"; //тут заполнение
    portaudio = "9759"; //тут заполнение
    portcontrol = "9758"; //тут заполнение
    break;
case 0xD2:
    //ну тут ничего не делаем
    mode = 3; //тут просто отправка обратно D2
    printf("[ MESS ] Failed to get an ID\n");
    break;
    }
}
else{
    return nullptr;
}

unsigned char * buff = nullptr;

switch (mode) {
case 1:

```



```

        datasize = sizeof(unsigned int) / sizeof(unsigned int) + id.size(); //тут считаем
количество байт

        buff = new unsigned char[datasize]; //ID RANDOM GENERATION для elcbot
        buff[0] = 0xD0; //тут надо придумать команды

        for(unsigned int currentbyte = 0; currentbyte < id.size(); currentbyte++){
            tmpdata.push_back(id.at(currentbyte));
        }
        break;
case 2:
        datasize = sizeof(unsigned int) / sizeof(unsigned int) * 5 + strip.size() +
        portvideo.size() + portaudio.size() + portcontrol.size(); //тут считаем количество
байт

        buff = new unsigned char[datasize]; //IP PORTS от зависимости ID
        buff[0] = 0xD1;
        for(unsigned int currentbyte = 0; currentbyte < strip.size(); currentbyte++){
            tmpdata.push_back(strip.at(currentbyte));
        }
        tmpdata.push_back(0xEF);
        for(unsigned int currentbyte = 0; currentbyte < portvideo.size(); currentbyte++){
            tmpdata.push_back(portvideo.at(currentbyte));
        }
        tmpdata.push_back(0xEF);
        for(unsigned int currentbyte = 0; currentbyte < portaudio.size(); currentbyte++){
            tmpdata.push_back(portaudio.at(currentbyte));
        }
        tmpdata.push_back(0xEF);
        for(unsigned int currentbyte = 0; currentbyte < portcontrol.size(); currentbyte++){
            tmpdata.push_back(portcontrol.at(currentbyte));
        }
        tmpdata.push_back(0xEF);

        break;
case 3:

```

					ВКР-НГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						127
Изм	Лист	№ докум.	Подп.	Дата		

```

        datasize = sizeof(unsigned int) / sizeof(unsigned int);
        buff = new unsigned char[datasize]; //Прервать - уже
        buff[0] = 0xD2;
        break;
    }
    if(buff == nullptr){
        return nullptr;
    }
    for(unsigned int currentbyte = 1; currentbyte < datasize; currentbyte++){
        buff[currentbyte] = tmpdata.at(currentbyte-1);
    }
    sizebuff = datasize;
    tmpdata.clear();

    return buff;
}

```

datacommunication.cpp

```

#include "datacommunication.h"
#include <vector>
using namespace dp;

DataCommunication::DataCommunication(int server_fd, int sock,
                                     struct sockaddr_in address, int addrlen)
{
    this->server_fd = server_fd;
    this->sock = sock;
    this->address = address;
    this->addrlen = addrlen;
    buff = new unsigned char[BUFSIZE];
}

DataCommunication::~DataCommunication(){
    delete[] buff;
}

```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						128
Изм	Лист	№ докум.	Подп.	Дата		

```
}
```

```
struct sockaddr_in DataCommunication::accept(){  
    if ((sock = ::accept(server_fd, (struct sockaddr *)&address,  
        (socklen_t*)&addrlen))<0)  
    {  
        perror("accept");  
        exit(EXIT_FAILURE);  
    }  
    return address;  
}
```

```
unsigned char * DataCommunication::recvData(unsigned int &numbytes){  
    numbytes = read( sock , buff, BUFSIZE);  
    buff[numbytes] = '\0';  
  
    return buff;  
}
```

```
int DataCommunication::sendData(unsigned char * buff, unsigned int sizemess){  
    if(-1 == send(sock , buff , sizemess , 0 )){  
        return 1;  
    }  
    return 0;  
}
```

					БКР-ИГТУ-09.03.01-(16-В-2)-002-2020(ПЗ)	Лист
						129
Изм	Лист	№ докум.	Подп.	Дата		