Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия

«____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Программа для экспресс идентификации и диагностики устройств станции спутниковой связи по сети Ethernet
тема

Руководитель	_____ подпись, дата	ведущий инженер, канд. техн. наук должность, ученая степень	В.А. Хабаров инициалы, фамилия
Выпускник	_____ подпись, дата		А.А. Жвакин инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2019

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия

«____» _____ 2019 г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы
бакалаврской работы, дипломного проекта, дипломной работы, магистерской диссертации

Студенту Жвакину Алексею Андреевичу
фамилия, имя, отчество

Группа КИ15-07Б Направление (специальность) 09.03.01
номер код

«Информатика и вычислительная техника»
наименование

Тема выпускной квалификационной работы Программа для экспресс идентификации и диагностики устройств станции спутниковой связи по сети Ethernet

Утверждена приказом по университету № 7516/с от 29 мая 2019

Руководитель ВКР В.А. Хабаров, ведущий инженер, АО «НПП «Радиосвязь»
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР: задание на бакалаврскую работу

Перечень разделов для ВКР: Анализ задания на выпускную квалификационную работу; Разработка; Тестирование; Руководство пользователя

Перечень графического материала: презентация доклада выступления

Руководитель ВКР _____ В.А. Хабаров
подпись инициалы и фамилия

Задание принял к исполнению _____ А.А. Жвакин
подпись инициалы и фамилия

«__» _____ 2019 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Программа для экспресс идентификации и диагностики устройств станции спутниковой связи по сети Ethernet» содержит 47 страниц текста, 27 иллюстраций, 7 использованных источников и 3 приложения.

ПОИСК, ИДЕНТИФИКАЦИЯ, СТАНЦИЯ СПУТНИКОВОЙ СВЯЗИ, ПОДСИСТЕМА АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ, МНОГОПОТОЧНОСТЬ, UDP, ETHERNET, WINDOWS, LINUX.

Цель работы: разработать программу, позволяющую проводить экспресс идентификацию и диагностику устройств станции спутниковой связи. Во время выполнения задания на выпускную квалификационную работу был проведен анализ предметной области и составлен план работы. Были рассмотрены различные способы взаимодействия по сетевым протоколам, а также методы реализации кроссплатформенности. Для ускорения проведения поиска активных устройств в подсети была использована многопоточность. В результате была разработана программа и протестирована на программах-имитаторах реальных приборов станции спутниковой связи.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Появление технологии спутниковой связи стало одним из самых значимых и глобальных технических прорывов в истории человечества. Спутниковая связь обеспечила возможность удаленной передачи информации на расстояния недостижимые для обычных наземных ретрансляторов радиосигналов. Данная технология является развитием традиционной радиорелейной связи путём вынесения ретранслятора на очень большую высоту. Так как максимальная зона его видимости в этом случае — почти половина Земного шара, то необходимость в цепочке ретрансляторов отпадает — в большинстве случаев достаточно и одного.

Для успешного функционирования системы спутниковой связи на земле чаще всего в неподвижном состоянии находятся Станции спутниковой связи (ССС), которые состоят из множества различных устройств, с которыми Подсистема автоматизированного управления (ПАУ) обменивается данными по сети Ethernet. Обмен с устройствами реализован с помощью различных сетевых протоколов (TCP/UDP/SNMP/TELNET и др). Помимо вариативности сетевых протоколов, у устройств могут быть разные прикладные протоколы обмена данными (формат сообщений которыми обмениваются ПАУ и устройство).

Целью выпускной квалификационной работы является разработка программы, позволяющей проводить экспресс идентификацию и диагностику устройств станции спутниковой связи.

1 Анализ задания на выпускную квалификационную работу

Станции связи, с которыми будет взаимодействовать разрабатываемая в рамках выпускной квалификационной работы программа, состоят из определенного набора приборов, которые можно разделить на несколько функциональных групп:

- антенны и навигационное оборудование;
- усилители мощности;
- конвертеры промежуточной частоты;
- модемное оборудование: модулятор/демодулятор и кодер/декодер, именуемые приемо-передающим комплектом (ППК);
- оконечное оборудование;
- вспомогательное оборудование.

Схема оборудования станции представлена на рисунке 1.

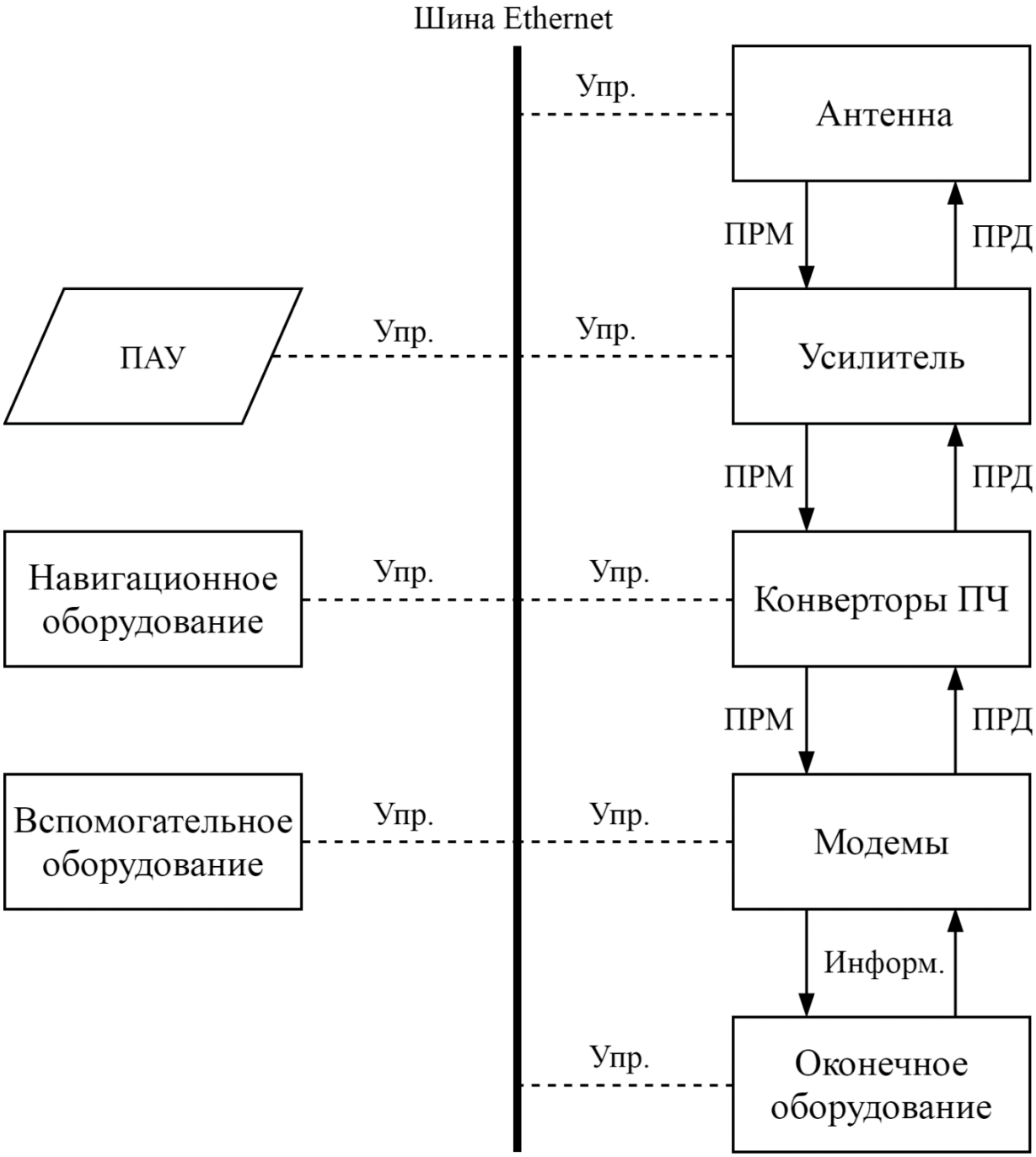

Рисунок 1 – Схема станции спутниковой связи

Взаимодействие с каждым из приборов происходит по определенному информационному протоколу обмена (Приложение А). Обмен осуществляется в режиме запрос-ответ, т.е. формируется определенного вида

сообщение в виде последовательности байт и отправляется в прибор на известный IP-адрес и порт. Прибор получив такой запрос обрабатывает его, формирует ответ и отправляет его на IP-адрес и порт, с которого был инициирован запрос. Каждый прибор имеет ID идентификатор, он используется при запросе к прибору и при ответе прибор также его сообщает. Программа должна отправлять запросы на найденные IP адреса, полученные из модуля «Поиск», и анализировать ответы.

Для разработки программного обеспечения было решено использовать кроссплатформенный фреймворк Qt(C++) [2]. Исходя из цели выпускной квалификационной работы был составлен план работы:

- получить список сетевых адаптеров и их интерфейсов;
- получить список активных устройств;
- идентифицировать их как приборы.

2 Разработка

Исходя из плана работы, составленного в предыдущем пункте принято решение о разбиении программы на модули:

- модуль Поиск: получение списка сетевых адаптеров и их интерфейсов, а также поиск активных устройств в локальной сети;
- модуль Идентификация: распознавание полученных устройств как приборов ССС.

2.1 Знакомство с библиотекой Network

Библиотека «Network», разработанная АО «НПП «Радиосвязь», реализует различное взаимодействие между сетевыми устройствами посредством протоколов UDP, TCP, COM, SNMP, Telnet и другими. Воспользуемся приложением «Network», использующим данную библиотеку, и протестируем работу с протоколами TCP и UDP.

UDP (англ. User Datagram Protocol — протокол пользовательских датаграмм) — один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных [4].

TCP (англ. Transmission Control Protocol — протокол управления передачей) — один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных. Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи [5].

С помощью библиотеки «Network» были проведены тесты обмена сообщениями между TCP-сервером и TCP-клиентом, а также между двумя UDP-адаптерами.

Скриншоты результатов теста TCP протокола с помощью клиента и сервера находятся на рисунках 2, 3, 4 и 5.

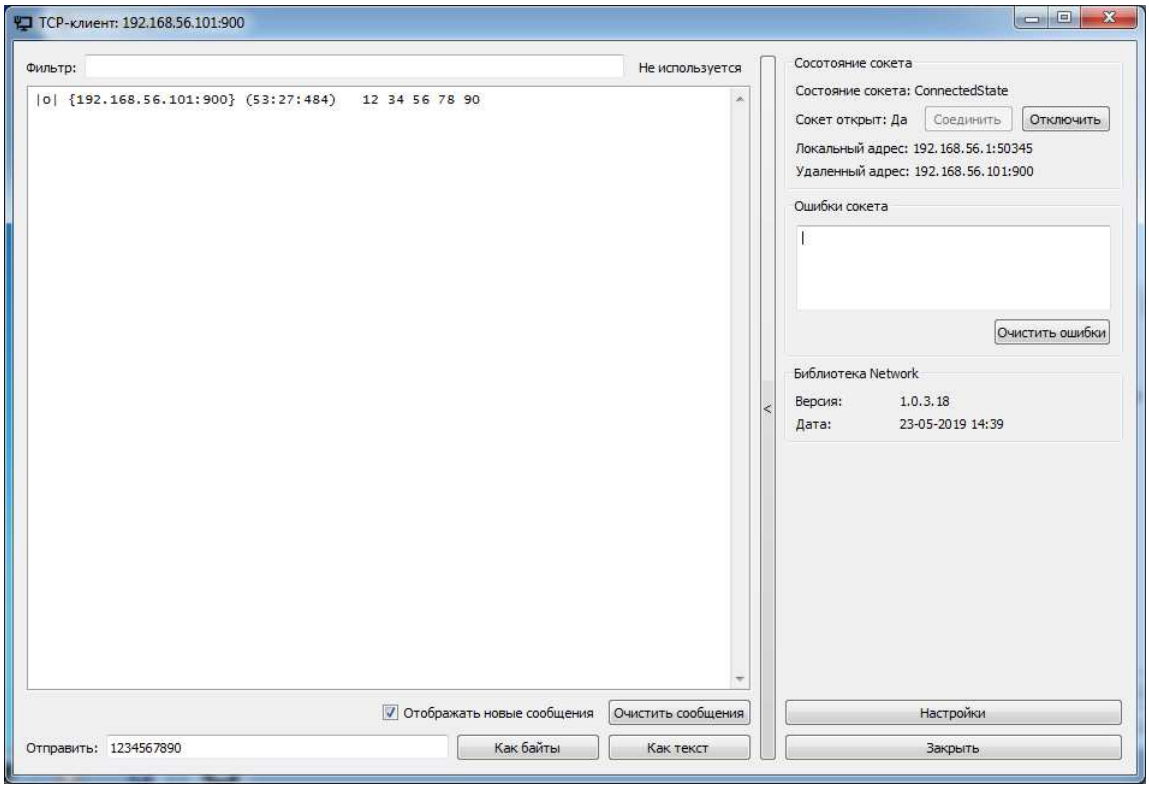

Рисунок 2 – Клиент отправляет байты Серверу

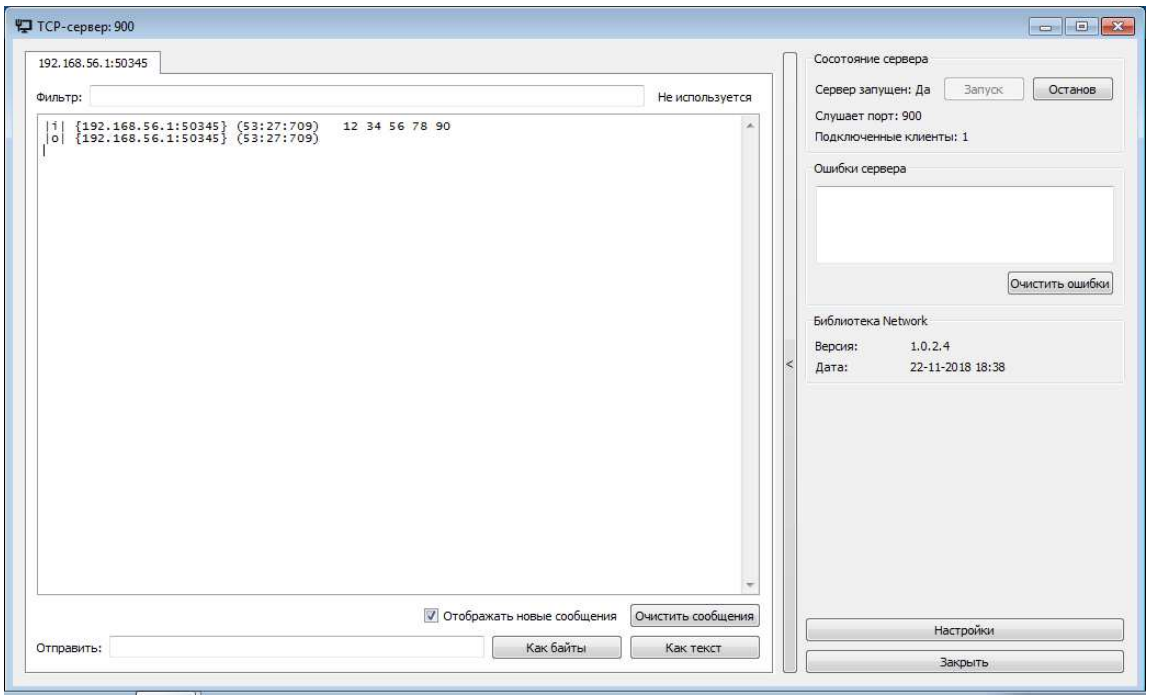

Рисунок 3 – Сервер принимает байты от Клиента

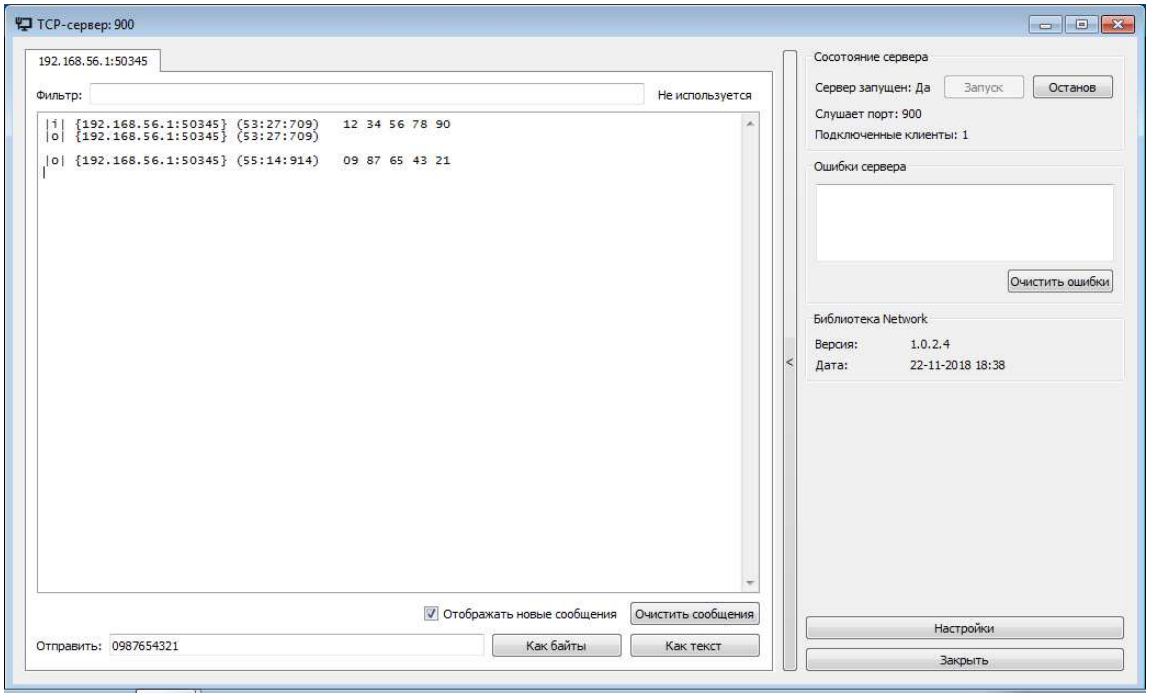

Рисунок 4 – Сервер отправляет байты Клиенту

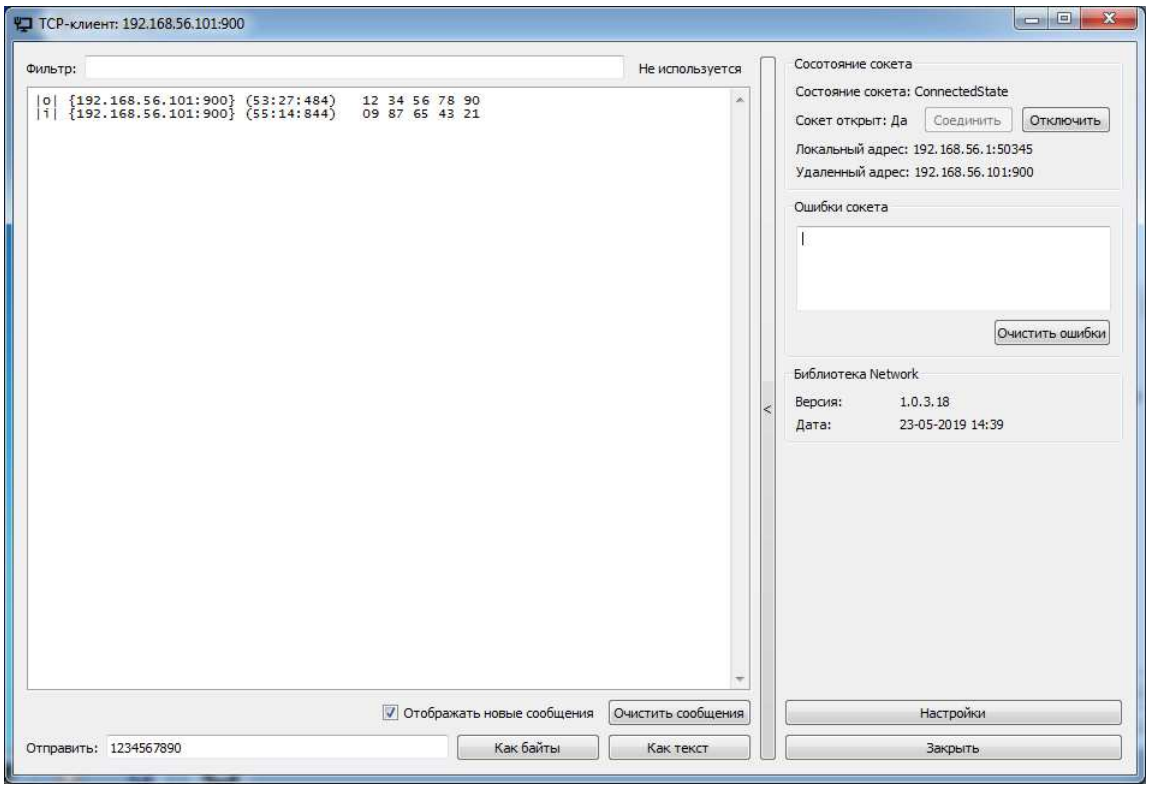

Рисунок 5 – Клиент принимает байты от Сервера

Для проведения тестирования взаимодействия по протоколу UDP воспользуемся двумя UDP-адаптерами. Результаты теста протокола представлены на рисунках 6 и 7.

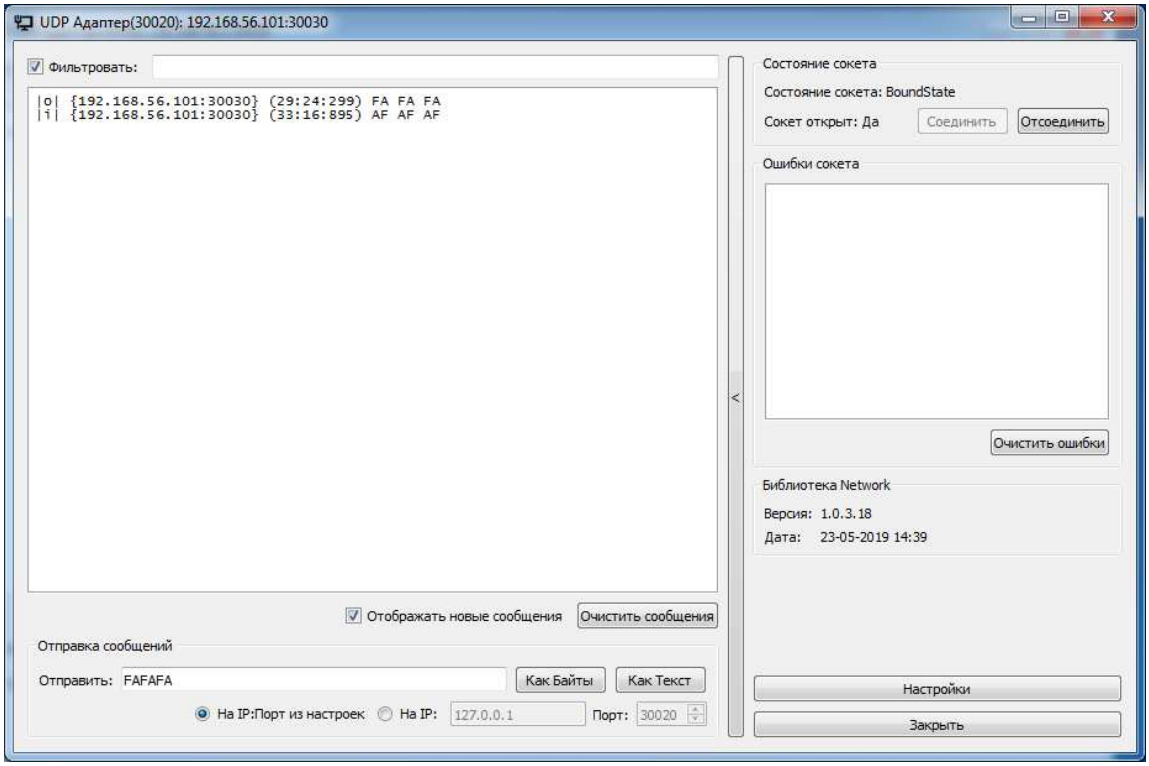

Рисунок 6 – UDP-адаптер#1 отправляет байты и принимает ответ от UDP-адаптера#2

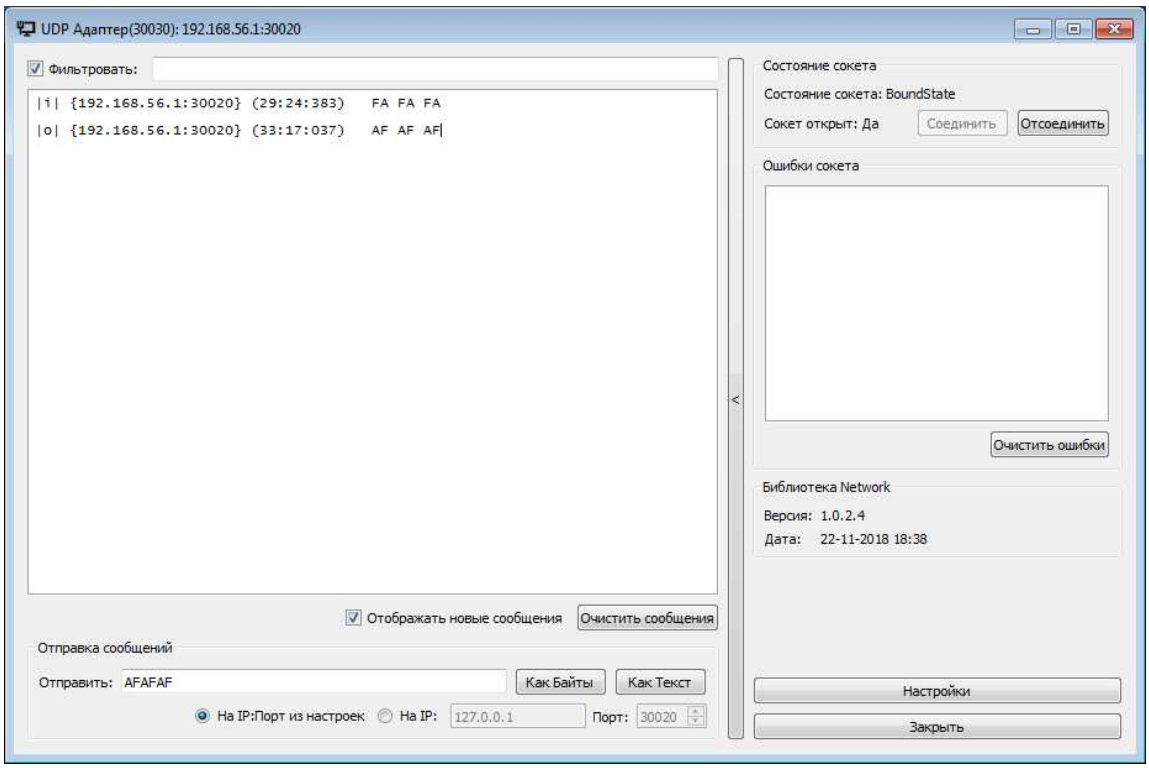

Рисунок 7 – UDP-адаптер#1 принимает байты и отправляет обратно UDP-адаптеру#2

2.2 Модуль поиска

Модуль отвечает за получение списка адаптеров компьютера, интерфейсов на каждом из адаптеров, а также проведение поиска активных устройств с использованием Echo-запросов по сети Ethernet.

2.2.1 NetworkTester

Класс *NetworkTester* отвечает за получение и хранение списка сетевых адаптеров компьютера. Его атрибуты и методы приведены на рисунке 8.

Для решения этой задачи воспользуемся специальным API: The Internet Protocol Helper (IP Helper) [3]. Данное API работает под операционными системами семейства Windows и позволяет получать и изменять параметры конфигурации сети для локального компьютера.

Воспользуемся библиотекой «iphlpapi.h», а именно функцией «GetAdapterInfo» и получим список всех сетевых адаптеров компьютера, содержащий их названия и MAC-адреса. Нужное взаимодействие с параметрами сетевой конфигурации реализовано в методе *NetworkTester::refreshAdapters()*, код которого приведен в листинге программы в Приложении Б.

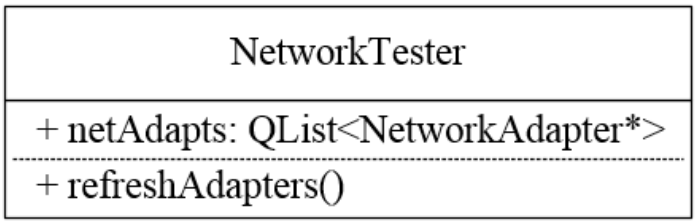

Рисунок 8 – Атрибуты и методы класса NetworkTester

2.2.2 NetworkAdapter

Класс *NetworkAdpater* реализует получение и хранение списка сетевых интерфейсов компьютера. Диаграмма класса представлена на рисунке 9.

Среди встроенных средств фреймворка Qt есть класс *QNetworkInterface*, один из методов которого позволяет получить список сетевых интерфейсов, а именно метод *NetworkAdpater::allInterfaces()*. Каждый сетевой интерфейс имеет имя, IP-адрес с маской и MAC-адрес. Этот метод возвращает список из объектов класса сетевых интерфейсов. Теперь в нашем распоряжении находятся все существующие на данном компьютере сетевые адаптеры и сетевые интерфейсы.

Ввиду того, что каждый сетевой адаптер может иметь как один, так и несколько сетевых интерфейсов необходимо сопоставить полученные сетевые интерфейсы каждому из адаптеров. Исходя из того, что полученные сетевые адаптеры имеют MAC-адреса, также, как и сетевые интерфейсы, возможно их сопоставление. Решением задач по получению сетевых интерфейсов, а также сопоставлению их сетевым адаптерам занят метод *NetworkAdapter::refreshSubnets(QString macAddress)*, код которого приведен в Приложении Б.

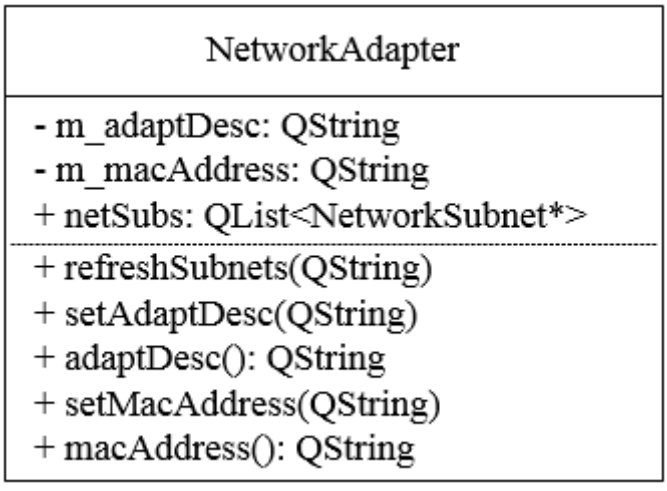

Рисунок 9 – Атрибуты и методы класса NetworkAdapter

2.2.3 NetworkSubnet

Класс *NetworkSubnet* создан для проведения поиска активных устройств, подключенных к локальной сети Ethernet. Атрибуты и методы класса изображены на рисунке 10.

Для быстрого и эффективного нахождения всех существующих сетевых устройств в подсети применен протокол межсетевых управляющих сообщений ICMP, входящий в стек протоколов TCP/IP. В данном случае для решения текущей задачи использован API: IP Helper для формирования echo-запросов [3], отправляемых на каждый из адресов подсети. Для ускорения процесса нахождения сетевых устройств применено многопоточное программирование с использованием классов *QFuture* и *QtConcurrent*.

За проведение echo-запросов отвечают следующие методы:

- NetworkSubnet ::ping(QHostAddress ip) – пинг одиночного адреса;

- NetworkSubnet ::pingTest(QHostAddress ip, QHostAddress netmask) – пинг диапазона устройств заданный IP-адресом и маской.

Листинг методов представлен в Приложении Б.

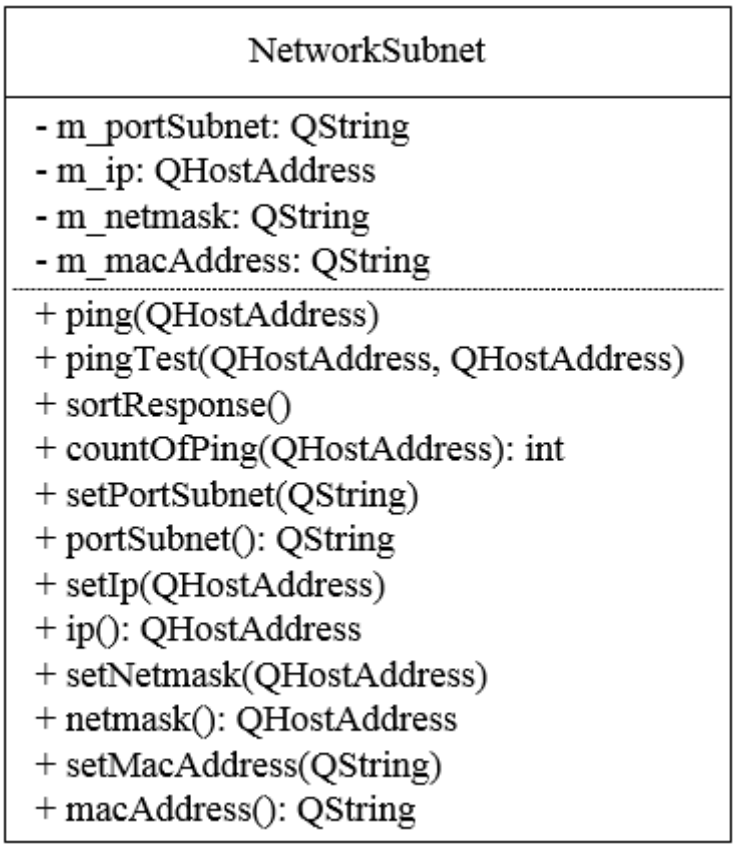

Рисунок 10 – Атрибуты и методы класса NetworkSubnet

2.2.4 Результат разработки модуля поиска

В результате работы были созданы 3 класса, которые в совокупности реализуют:

- получение сетевых адаптеров компьютера;

- получение сетевых интерфейсов компьютера и их сопоставление с сетевыми адаптерами;

- проведение echo-запросов и получение списка ответивших на запросы устройств.

Диаграмма реализованных классов приведена на рисунке 11.

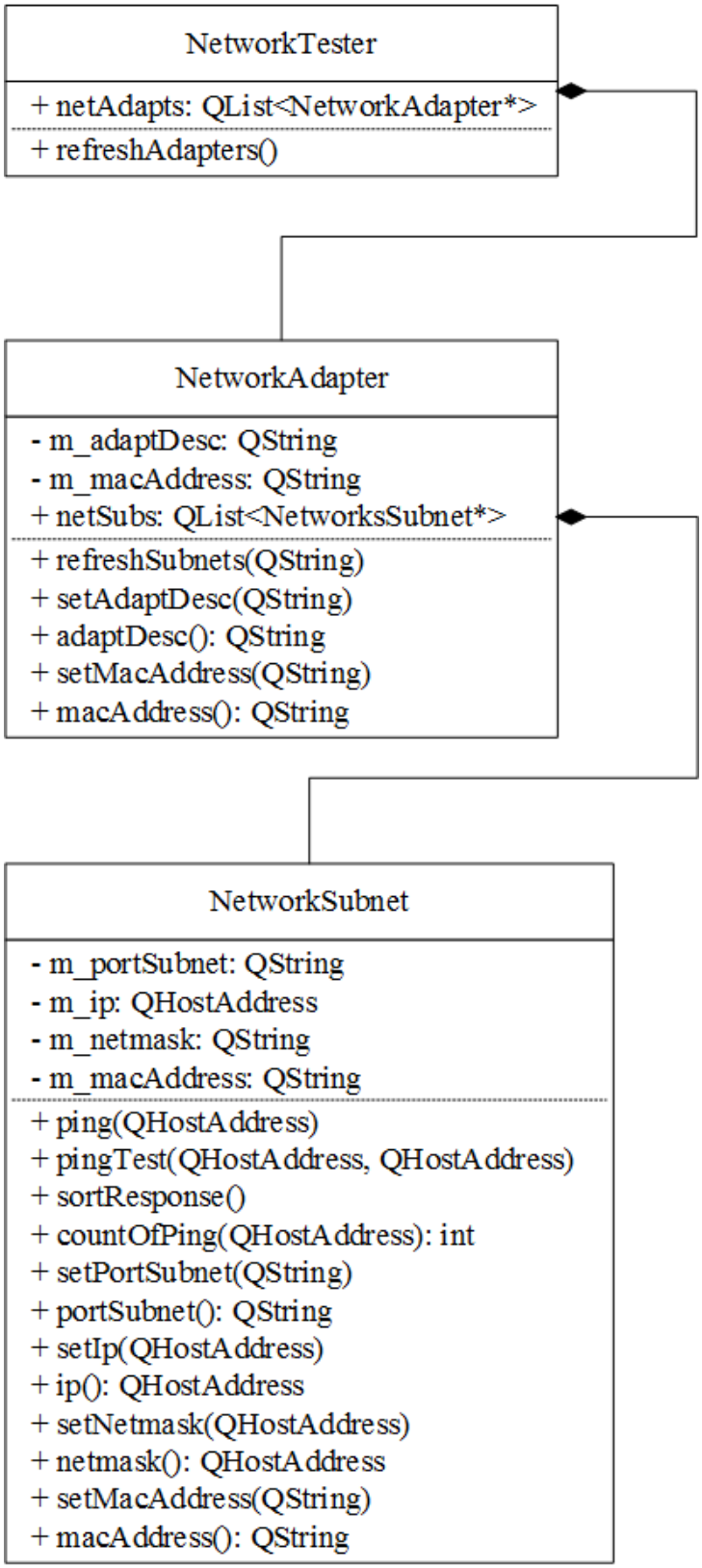

Рисунок 11 – Диаграмма классов NetworkTester, NetworkAdapter и NetworkSubnet

2.3 Модуль идентификации

Взаимодействие между ПАУ и приборами ССС осуществляется по протоколу прикладного уровня путем отправки запросов, формат запросного сообщения приведен в Приложении А.

Каждый тип приборов, входящих в состав ССС имеет уникальный ID, по которому к нему можно обращаться: запрашивать или устанавливать данные регистров. Ввиду того, что для каждого типа прибора требуется свое сообщение запроса было принято решение о разработке классов, управляющих созданием и отправкой запросов, а также приемом и анализом ответов от приборов.

2.3.1 PackageManager

Для того, чтобы иметь возможность получения и легкого редактирования посылок для пользователя принято решение о хранении запросов и сопутствующей им информации в виде Json-файла, пример которого представлен в Приложении В. Это позволит в будущем добавлять посылки для новых приборов без вмешательства в код программы.

Структура файла состоит из словаря с единственным ключом «packages» со списком значений, которые являют собой посылки. Каждая из посылок включает в себя:

- имя прибора, для которого она назначена;
- сообщение запроса, являющееся массивом байтов;
- список из пар значений «№ байта»: «Ожидаемое значение».

Класс *PackageManager* (рисунок 12) с помощью метода *readJsonFile(QString)* получает и сохраняет посылки, которые были считаны из Json-файла, имя которого передавалось как аргумент в методе. Для хранения посылок в памяти во время выполнения программы принято решение о создании вспомогательных классов, речь о которых пойдет далее.

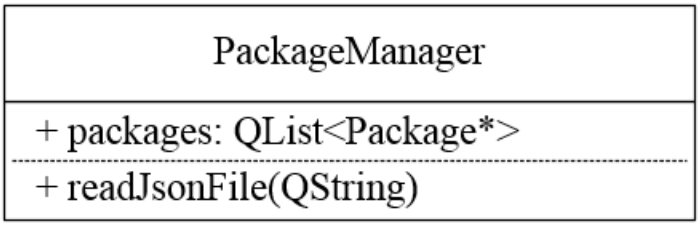

Рисунок 12 – Атрибуты и методы класса PackageManager

2.3.2 ByteAccord

Для однозначной идентификации аппаратуры станции спутниковой связи требуется сравнить определенные байты в полученной от прибора посылке с заранее заданными для проверки байтами. Байты, подлежащие проверке, а также их ожидаемое и необходимое для успешной идентификации устройства значение следует хранить в памяти в виде пар «Позиция Байта»: «Значение». Позиция байта задается типом числовых данных int, а его значение с помощью unsigned char. Установка, хранение и их получение реализовано в классе ByteAccord (рисунок 13).

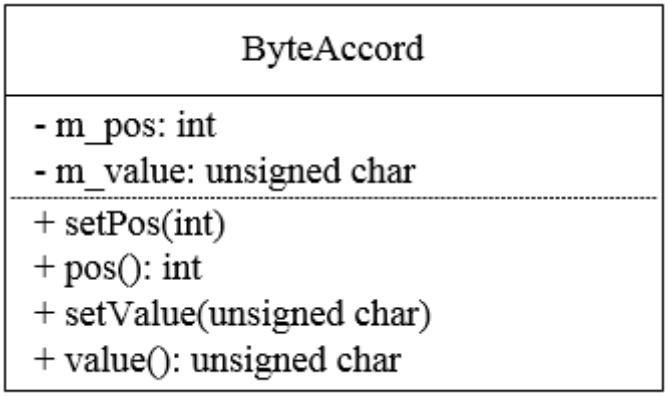

Рисунок 13 – Атрибуты и методы класса ByteAccord

2.3.3 Package

Каждая из посылок содержит в себе название прибора, для идентификации которого используется данная посылка, само сообщение-запрос для отправки на прибор и массив из пар значений «Позиция Байта»: «Значение». Для хранения названия и запроса в классе *Package* выделены два поля типа QString, а также массив из ByteAccord. Диаграмма класса представлена на рисунке 14.

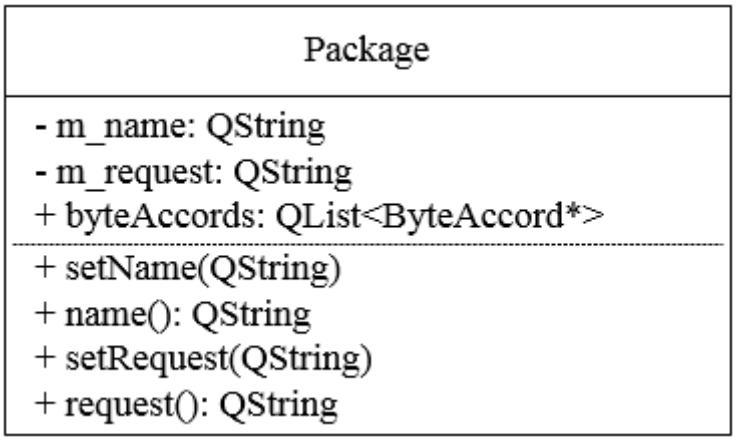

Рисунок 14 – Атрибуты и методы класса Package

2.3.4 PackageSender

В связи с тем, что на приборах станции спутниковой связи подняты Udp-сервера, данный класс *PackageSender* (рисунок 15) был спроектирован как средство управления Udp-адаптером. Для взаимодействия с Udp-адаптером используется библиотека «Network», а именно класс *UdpAdapter*. *PackageSender* ответственен за отправку запросов на IP-адреса приборов, полученных ранее с помощью echo-запросов ICMP. Он же анализирует пришедшие на клиентскую машину ответы приборов.

Для использования этого класса следует установить список посылок, полученных *PackageManager*`ом через метод *setPackages(QList<Packages*>)*,

а также передать список из IP-адресов устройств, которые будут опрашиваться посредством метода *sendRequest()*. В результате работы метода будет получен список из IP-адресов и их соответствия определенным типам приборов.

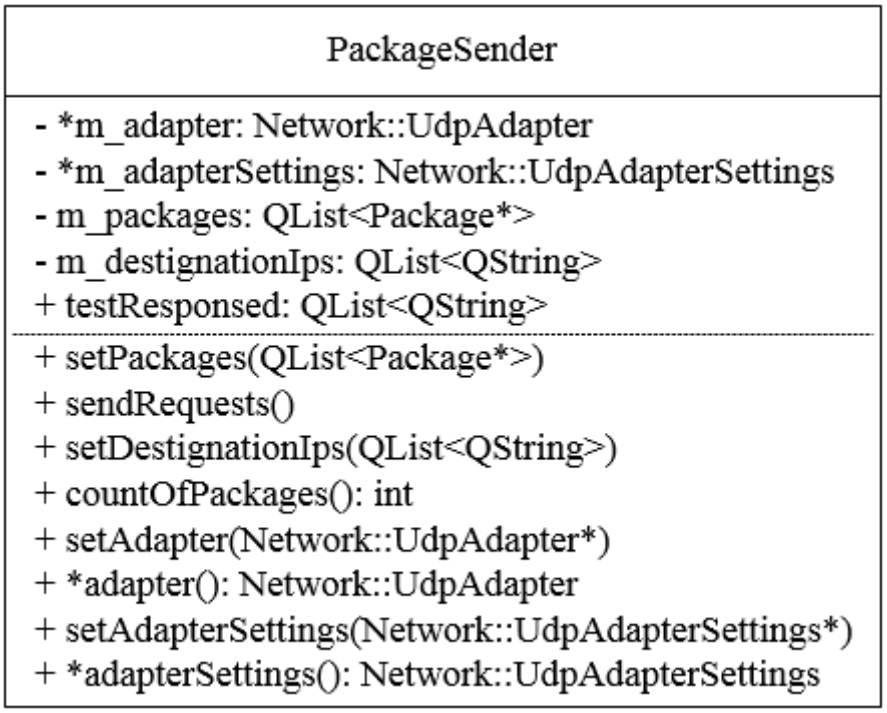

Рисунок 15 – Атрибуты и методы класса PackageSender

2.3.5 Результат разработки модуля идентификации

В результате работы были созданы 4 класса, которые в совокупности реализуют:

- получение посылок из Json-файла и их хранение;

- отправку посылок на заранее полученные из модуля «Поиск» IP-адреса;

- анализ полученных ответов от устройств станции спутниковой связи;

- проведение соответствия между IP-адресами и типами приборов;

- получение списка идентифицированных приборов в виде пар «IP-адрес»: «Прибор».

Диаграмма реализованных классов представлена на рисунке 16.

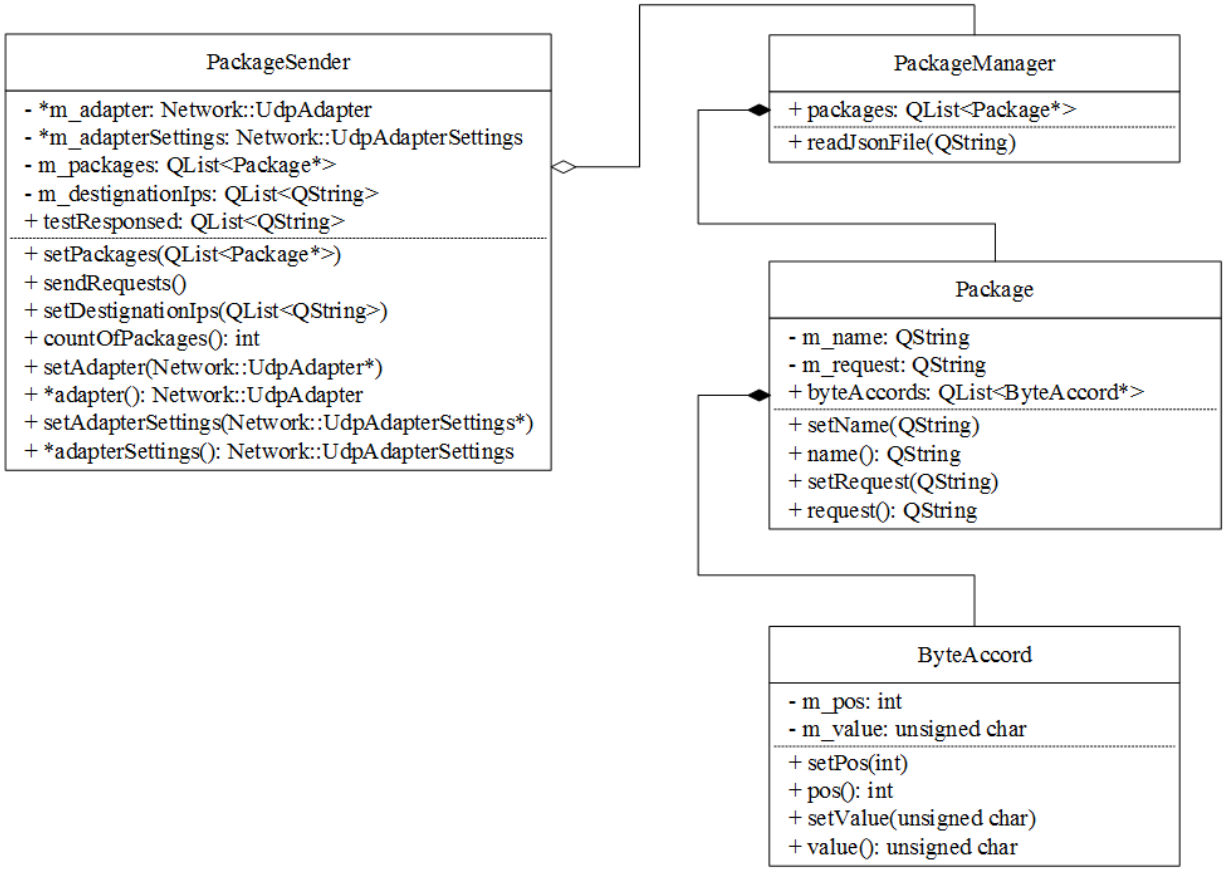

Рисунок 16 – Диаграмма классов PackageSender, PackageManager, Package, ByteAccord

2.4 Кроссплатформенная реализация

Ввиду того, что язык C++ и фреймворк Qt являются кроссплатформенными, существует возможность перекомпиляции программы «NetworkTester» под семейство операционных систем Linux.

Но во время разработки приложения, а именно функций классов для получения списка сетевых адаптеров компьютера, а также проведения ICMP echo-запросов было использовано платформо-зависимое API Windows. В текущем формате без изменения кода перекомпиляция под Linux невозможна.

Необходимо изменить исходный код нескольких методов, а именно методов *ping(QHostAddress)* класса *NetworkSubnet* и *refreshAdapters()* класса *NetworkTester*. Для того, чтобы не иметь два разных проекта с версиями кода для Windows и Linux используются директивы препроцессора #ifdef и #endif. При передаче в качестве аргумента для #ifdef директивы, содержащей в себе название целевой операционной системы, компилятору будет понятно какой из частей кода использовать при компиляции программы.

Метод *ping(QHostAddress)* использует для работы библиотеки "winsock2.h", "iphlpapi.h" и "icmpapi.h". Данные библиотеки созданы под ОС Windows и не поддерживаются ОС Linux. В реализации метода для Linux воспользуемся системной утилитой «Ping». Эта утилита принимает на вход такие параметры как IP-адрес, количество пакетов, интервал между отправкой пакетов, время жизни пакета TTL и другие. Чтобы выполнить команду ping программными средствами фреймворка Qt используется класс *QProcess* с методом *exec(QString program, QStringList arguments)*. В виде аргумента *program* передается строка «ping», а для *arduments* – IP-адрес и количество пакетов. Реализация полученного метода представлена в Приложении Б.

Метод *refreshAdapters()* также использует библиотеку «IP Helper», поэтому воспользуемся методом *allInterfaces()* класса *QNetworkInterface*. С

помощью данного метода получим список, состоящий из имен интерфейсов и связанных с ними MAC-адресами. Листинг кода метода *refreshAdapters()* представлен в Приложении Б.

3 Тестирование

Тестирование программы «NetworkTester» проводится с использованием виртуальной машины. Виртуальная машина будет играть роль реального прибора аппаратуры станции спутниковой связи с помощью заранее предоставленных АО «НПП «Радиосвязь» программ-имитаторов.

3.1 Версия для Windows

В первую очередь необходимо проверить функционал, отвечающий за сканирование подсети на наличие устройств в сети (рисунок 17).

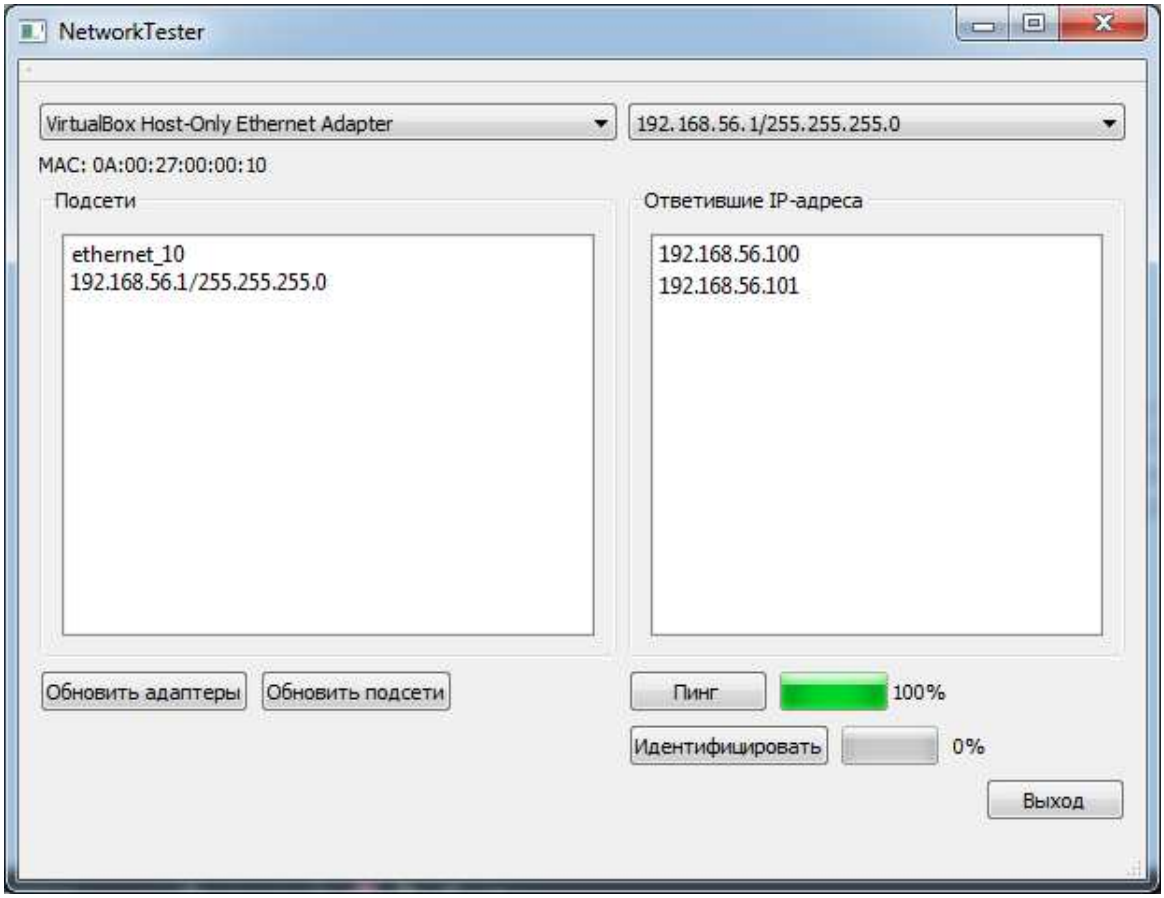

Рисунок 17 – Тест пинга на версии для Windows

Тест пройден успешно: все устройства, находящиеся в подсети виртуального адаптера найдены.

Для проверки корректности функционирования модуля «Идентификация» требуется поднять виртуальную машину с запущенной и настроенной программой-имитатором для работы в качестве реально существующего прибора, который будет отвечать на запросы по Udp-протоколу.

Тесты с прибором К-ПРД-6 представлены на рисунках 18 и 19.

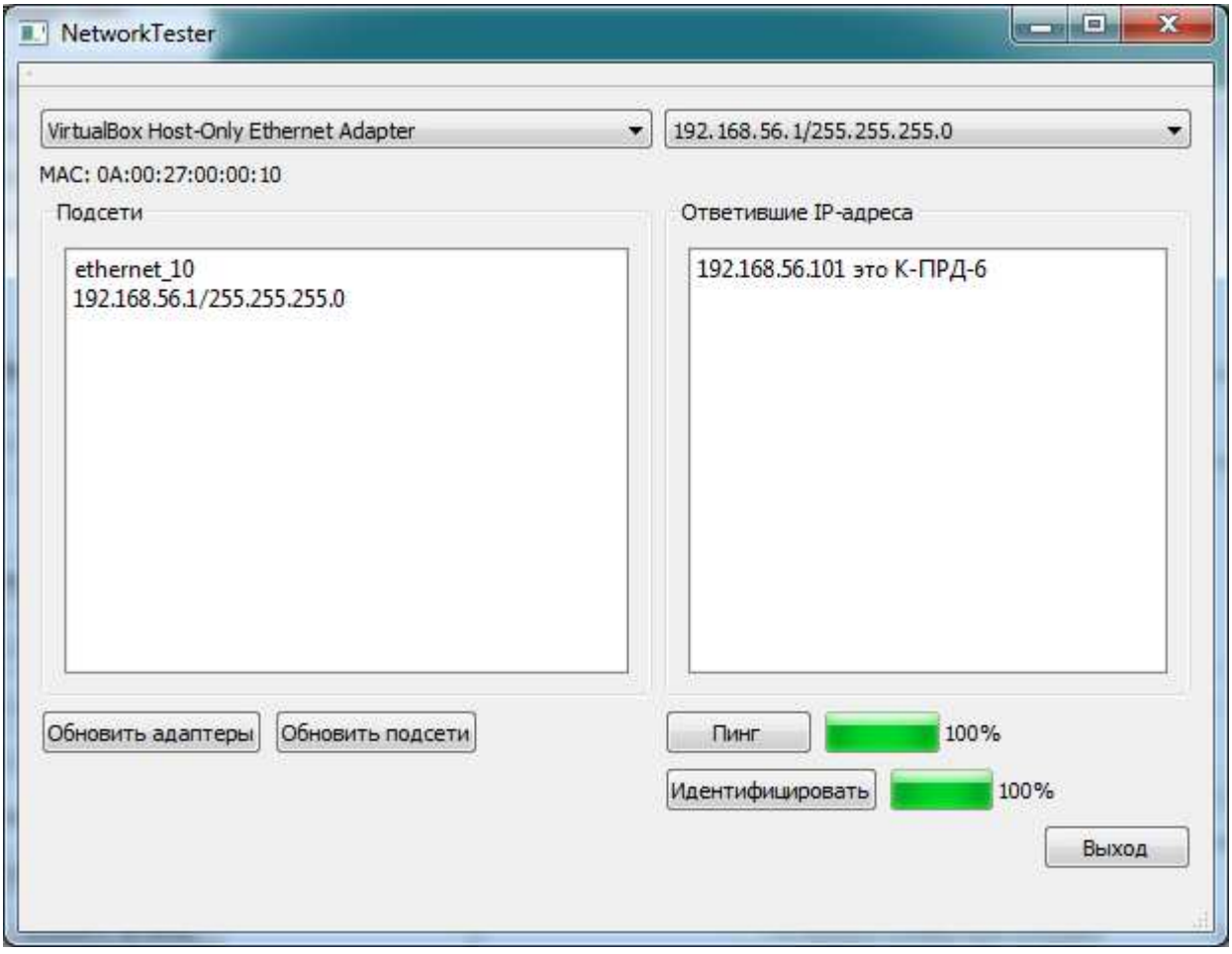

Рисунок 18 – Идентификация прибора К-ПРД-6

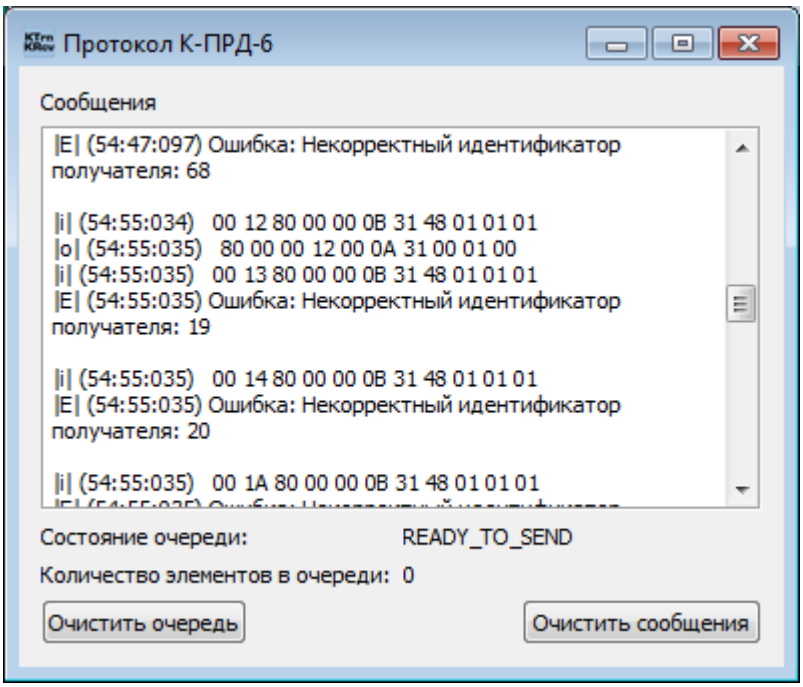

Рисунок 19 – Полученные и отправленные сообщения для К-ПРД-6

На рисунке 19 можно наблюдать что на одно из сообщений был дан ответ. Этот ответ был проанализирован и из результатов анализа следует, что сообщение «00128000000B3148010101» предназначалось для идентификации прибора К-ПРД-6. Тест пройден успешно.

Протестируем программу на другом приборе (рисунки 20, 21). В этот раз поднимем на виртуальной машине имитатор прибора К-ПРМ-7НК. Проведем идентификацию еще раз.

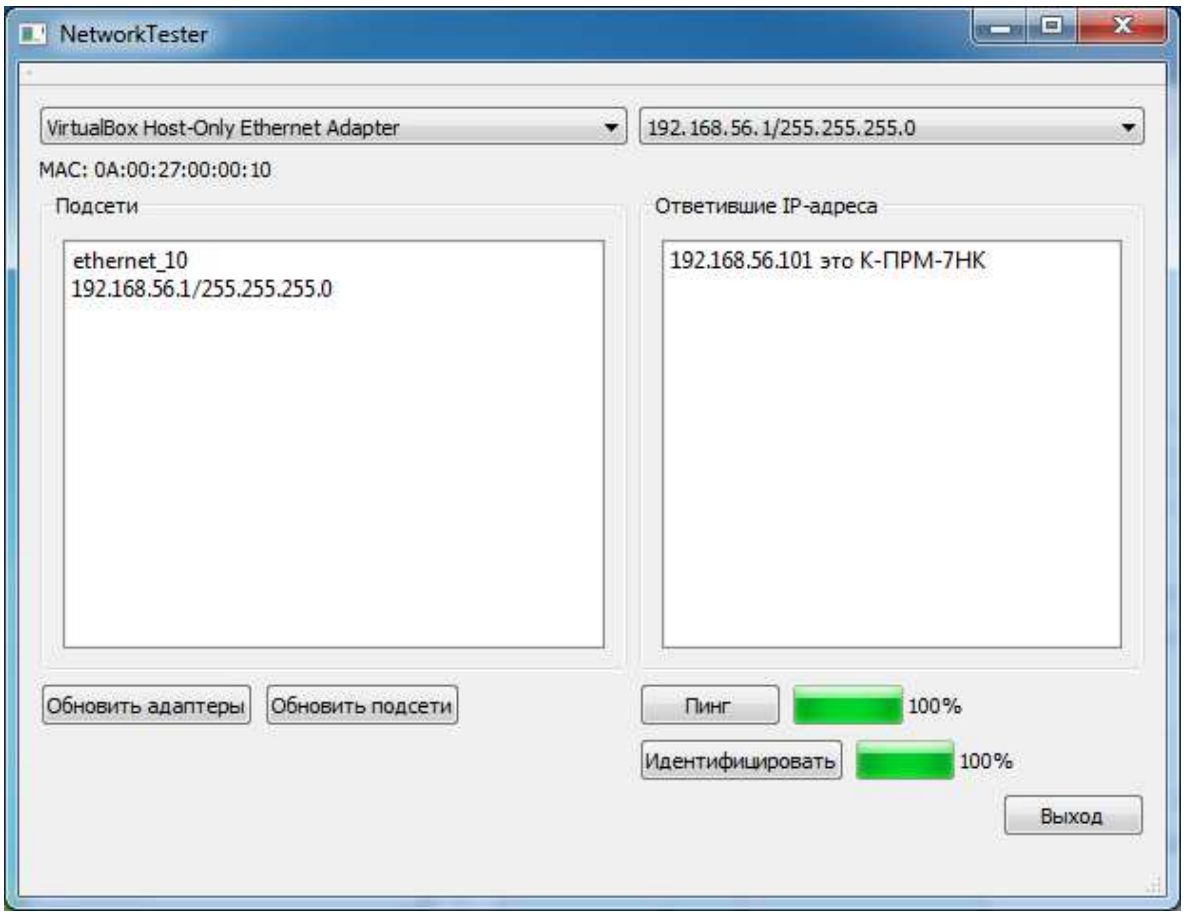

Рисунок 20 – Идентификация прибора К-ПРМ-7НК

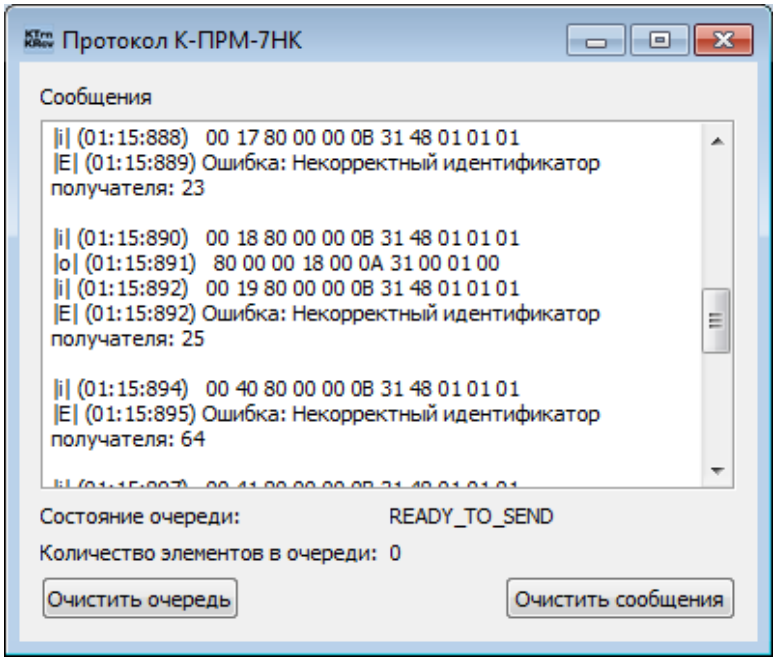

Рисунок 21 – Полученные и отправленные сообщения для К-ПРМ-7НК

Из рисунка 21 следует, что прибор отправил ответ на одно из сообщений. Прибор К-ПРМ-7 идентифицирован успешно – тест пройден.

3.2 Версия для Linux

Повторим последовательной действий, описанную в пункте 4.1, но в роли реальной аппаратуры сымитируем другие приборы. Проведем поиск активных устройств в подсети посредством пинга (рисунок 22).

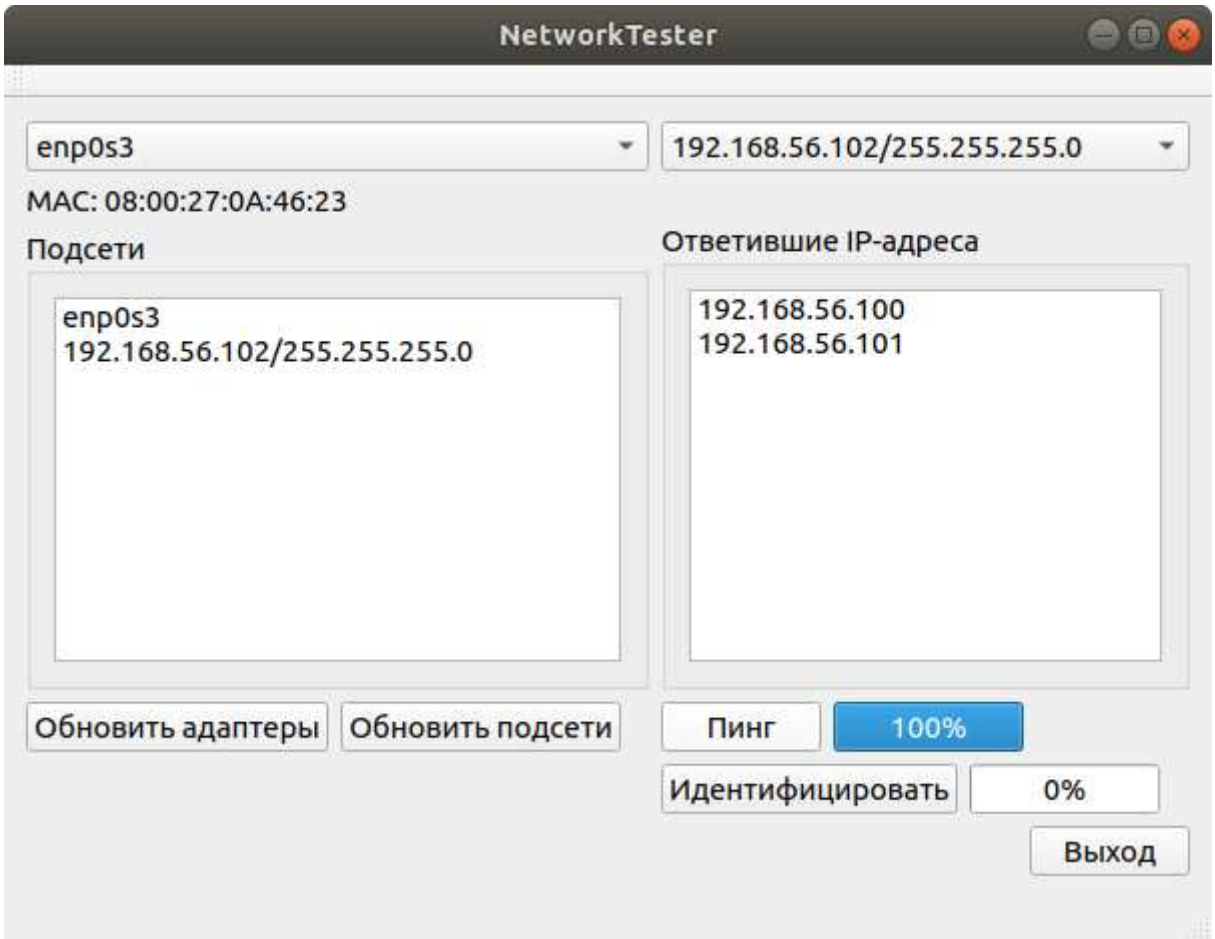

Рисунок 22 – Тест пинга на версии для Linux

В ходе проведения echo-запросов получены верные результаты. Тест пройден успешно.

В роли прибора для тестирования модуля «Идентификация» выступит К-ПРД-45. Скриншоты тестирования идентификации прибора приведены на рисунках 23 и 24.

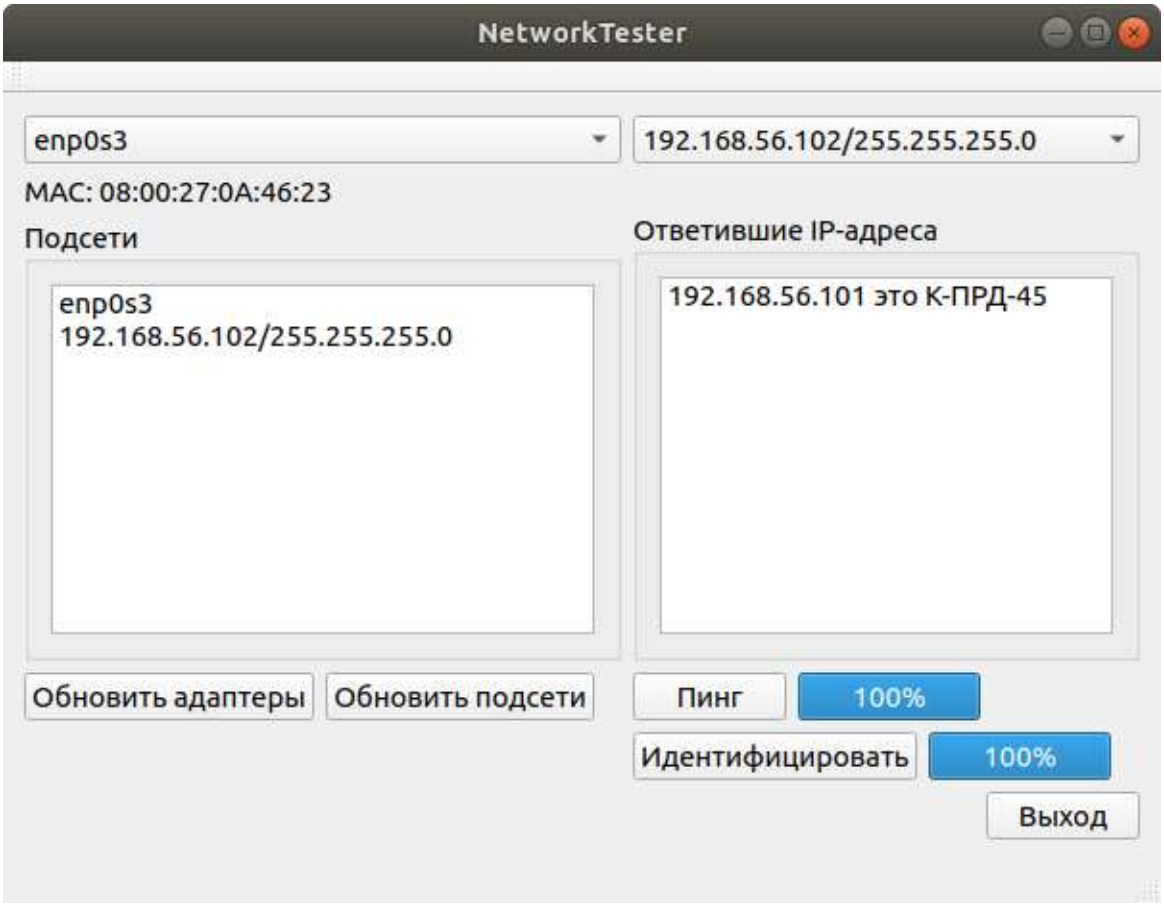

Рисунок 23 – Идентификация прибора К-ПРД-45

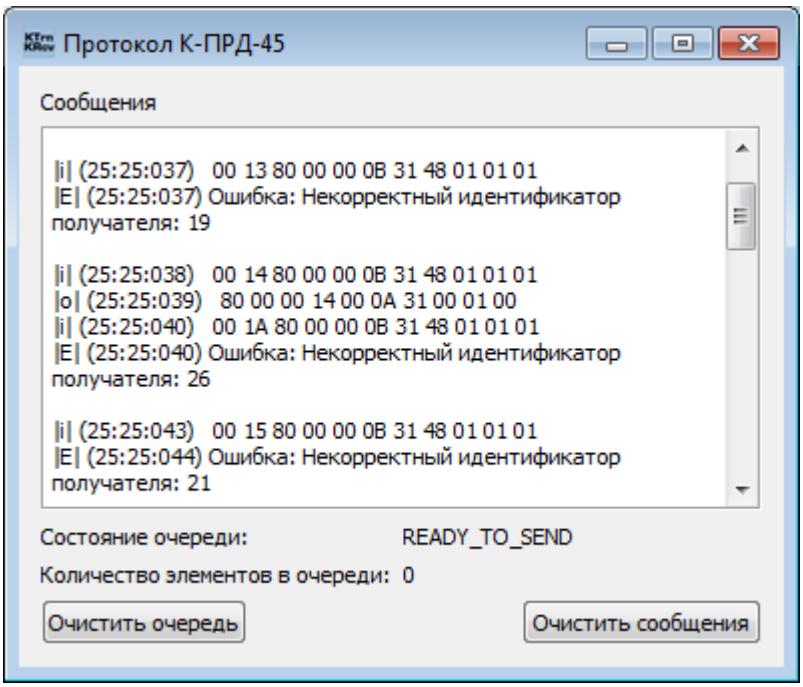

Рисунок 24 – Полученные и отправленные сообщения для К-ПРД-45

Прибор К-ПРД-45 ответил на запрос и был идентифицирован. Тест пройден успешно.

Протестируем еще с одним приборов, в этот раз это будет УМ44-5. Тестирование представлено на рисунках 25, 26.

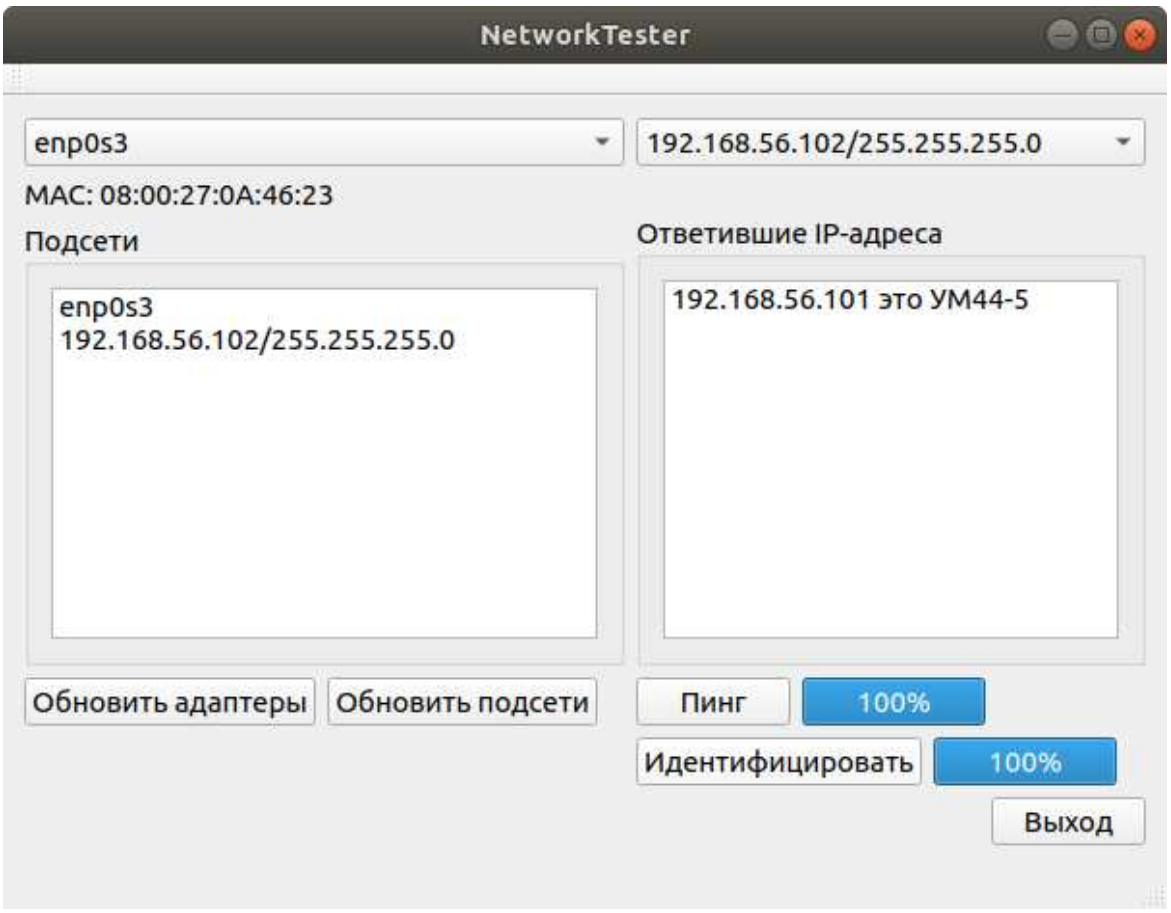

Рисунок 25 – Идентификация прибора УМ44-5

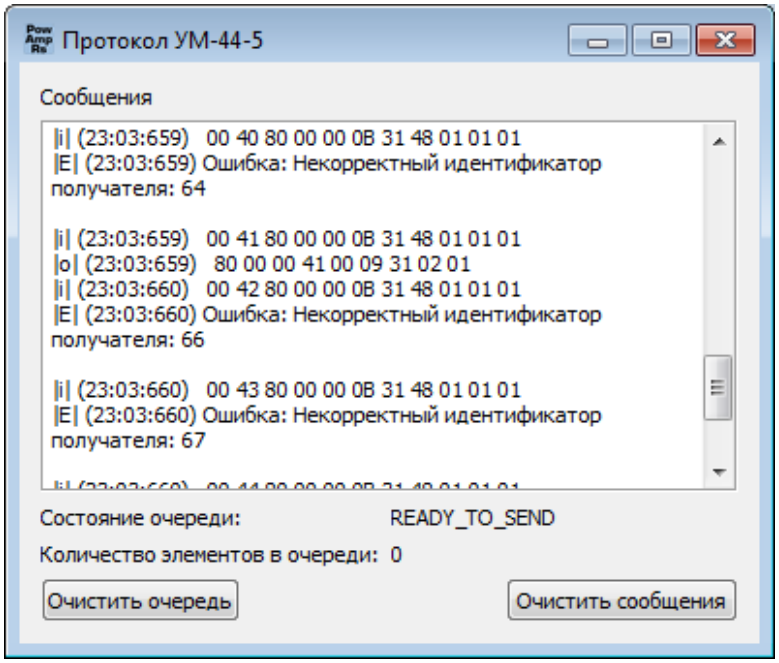

Рисунок 26 – Полученные и отправленные сообщения для УМ44-5

Исходя из рисунков выше определяем, что тест идентификации пройден успешно.

4 Руководство пользователя

При запуске программы «NetworkTester» перед пользователем возникает окно, представленное на рисунке 27.

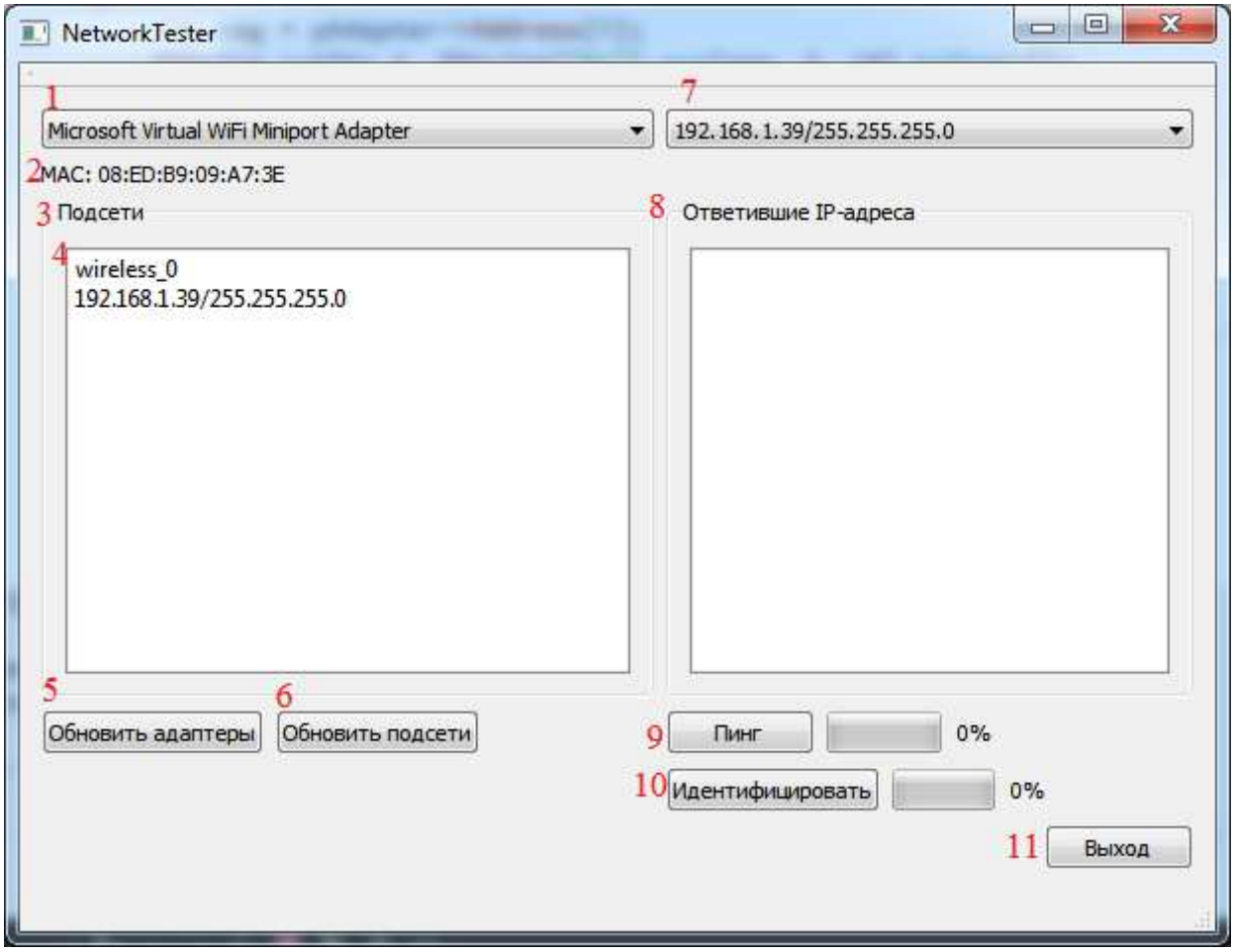

Рисунок 27 – Gui программы «NetworkTester»

Рассмотрим элементы по отдельности.

1. ComboBox с помощью которого производится выбор сетевого адаптера;

2. В данной строке отображен MAC-адрес текущего выбранного адаптера;

3. GroupBox со списком подсетей для выбранного адаптера;

4. Пример элемента подсети: имя интерфейса, IP-адресс/Маска;

5. Кнопка «Обновить адаптеры»: произвести заново поиск сетевых адаптеров компьютера;

6. Кнопка «Обновить подсети»: произвести заново поиск сетевых интерфейсов компьютера;

7. ComboBox используется для выбора интерфейса требуемого для пинга и идентификации аппаратуры станции спутниковой связи;

8. В данном ListWidget выводятся найденные во время проведения операции «пинг» устройства, которые отображаются в виде IP-адресов;

9. Кнопка запуска операции «пинг» с визуализацией процесса в виде прогресс-бара;

10. Кнопка запуска операции «идентификация» с визуализацией процесса в виде прогресс-бара;

11. Кнопка выхода из приложения.

Для проведения поиска и идентификации пользователю нужно выбрать сетевой адаптер, сетевой интерфейс с которого будут проведены необходимые операции. Сначала требуется определить наличие активных устройств в подсети с помощью кнопки «Пинг», потом нажатием на кнопку «Идентификация» получить и вывести список идентифицированных устройств.

ЗАКЛЮЧЕНИЕ

Во время работы были реализованы методы получения списка сетевых адаптеров и проведения echo-запросов кроссплатформенно для ОС Windows и Linux. Метод, осуществляющий echo-запросы, выполняется с использованием многопоточности. Для обмена сообщениями между ПАУ и приборами станции спутниковой связи была подключена в проект стороння библиотека «Network», для использования которой были изучены принципы и механика взаимодействия по сетевым протоколам UDP и TCP. Хранение списка приборов с необходимыми для их идентификации запросами и сопутствующей информацией происходит с использованием JSON файла, что дает возможность наращивания функционала (таблица новых приборов) за счет дополнения JSON файла, которая не требует вмешательства в код программы и перекомпиляции. Для проведения тестирования программы были использованы технологии виртуализации, с помощью которых были подняты имитаторы реально существующих приборов.

Итогом работы является создание кроссплатформенной программы для экспресс идентификации и диагностики аппаратуры станции спутниковой связи по сети Ethernet.

СПИСОК СОКРАЩЕНИЙ

АО «НПП «Радиосвязь» – акционерное общество «Научно-производственное предприятие «Радиосвязь»

ВКР – Выпускная Квалификационная Работа

ЛСВ – Локальная Вычислительная Сеть

ОС – Операционная Система

ПАУ – Подсистема Автоматизированного Управления

ССС – Станция Спутниковой Связи

GUI – Graphical User Interface

ICMP – Internet Control Message Protocol

JSON – JavaScript Object Notation

SNMP – Simple Network Management Protocol

TCP – Transmission Control Protocol

TELNET – teletype network

TCP/IP – Transmission Control Protocol/Internet Protocol

UDP – User Datagram Protocol

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 "Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности".

2. Саммерфилд, М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++ / М. Саммерфилд. – Санкт-Петербург: Символ-Плюс, 2011. – 552 с.

3. IP Helper | Microsoft Docs [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/en-us/windows/desktop/api/_iphlp/

4. UDP – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/UDP

5. Transmission Control Protocol – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Transmission_Control_Protocol

6. Дуглас Камер. Сети TCP/IP, том 1. Принципы, протоколы и структура = Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. — Москва: «Вильямс», 2003. — 880 с.

7. Дмитрий Кетов. Linux. Внутреннее устройство / Анна Кузьмина. — Санкт-Петербург: БХВ, 2017. — 320 с. — ISBN 978-5-9775-3580-9.

ПРИЛОЖЕНИЕ А

Протокол ЛВС

Формат сообщений, которыми обмениваются ПАУ и аппаратура станции спутниковой связи представлен в виде таблицы 1.

Таблица 1 – Формат сообщений ЛВС

№ байта	Наименование поля	Описание поля
1,2	ИДП	Идентификатор получателя сообщения «кому»
3,4	ИДО	Идентификатор отправителя сообщения «от кого»
5,6	ДЛС	Длина сообщения (длина всего пакета, включая заголовочную часть)
7	ВС	Вид сообщения
8	КОП	Код операции/ошибки
9	НП	Номер пакета/блока
10…n	БПИ	Блок прикладной информации

Поле ВС принимает значения:

- 30h – служебная команда;
- 31h – сообщение с блоком прикладной информации;
- E5h – уровень сигнала.

Поле КОП определяет инициализируемую операцию:

- 44h – запись данных;
- 48h – чтение данных;
- 00h – запрос от ПАУ обработан;
- 02h – неверный код операции от ПАУ;
- 03h – неформатное сообщение от ПАУ.

ПРИЛОЖЕНИЕ Б

Листинг фрагментов кода программы

Выдержка из NetworkTester.cpp

```
void NetworkTester::refreshAdapters()
{
   netAdapts.clear();

#ifdef Q_OS_WIN
   setlocale(LC_ALL, "Russian");
   PIP_ADAPTER_INFO pAdapterInfo;
   pAdapterInfo = (IP_ADAPTER_INFO *) malloc(sizeof(IP_ADAPTER_INFO));
   ULONG buflen = sizeof(IP_ADAPTER_INFO);

   if (GetAdaptersInfo(pAdapterInfo, &buflen) == ERROR_BUFFER_OVERFLOW)
   {
      free(pAdapterInfo);
      pAdapterInfo = (IP_ADAPTER_INFO *) malloc(buflen);
   }

   if (GetAdaptersInfo(pAdapterInfo, &buflen) == NO_ERROR)
   {
      PIP_ADAPTER_INFO pAdapter = pAdapterInfo;

      while (pAdapter)
      {
         QString MACAddress;

         for (int i = 0; i < 6; i++)
         {
            int seg = pAdapter->Address[i];
            QString cutStr =  QString("%1").arg(seg, 2, 16).toUpper();
```

```
                cutStr.replace(" ", "0");

                MACAddress.append(cutStr);
                MACAddress.append(":");
            }

            MACAddress.chop(1);
            NetworkAdapter *adapt = new NetworkAdapter(this, pAdapter->Description,
MACAddress);
            netAdapts.append(adapt);
            pAdapter = pAdapter->Next;
        }
    }

#endif

#ifdef Q_OS_LINUX

    QNetworkInterface netInterface;
    QList<QNetworkInterface> IpList = netInterface.allInterfaces();

    for (int i = 0; i < IpList.size(); i++)
    {
        qDebug() << "Interface " << i << ":" << IpList.at(i).humanReadableName() << "; "
                 << IpList.at(i).name() << "; " << IpList.at(i).hardwareAddress();
        NetworkAdapter *adapt = new NetworkAdapter(this, IpList.at(i).name(),
                                                   IpList.at(i).hardwareAddress());
        netAdapts.append(adapt);
    }
#endif
}
```

Выдержка из NetworkAdapter.cpp

```
void NetworkAdapter::refreshSubnets(QString macAddress)
{
    QList<QNetworkInterface> interList = QNetworkInterface::allInterfaces();
    netSubs.clear();

    for (auto interface : interList)
    {
        if (/*(interface.type() == QNetworkInterface::InterfaceType::Ethernet
            || interface.type() == QNetworkInterface::InterfaceType::Wifi)
            &&*/ interface.flags().testFlag(QNetworkInterface::IsUp))
        {
            for (auto entrie : interface.addressEntries())
            {
                auto protocol = entrie.ip().protocol();
                auto hardwareAddr = interface.hardwareAddress();

                if (protocol == QAbstractSocket::IPv4Protocol && hardwareAddr ==
macAddress)
                {
                    NetworkSubnet *ns = new NetworkSubnet(this, interface, entrie);
                    netSubs.append(ns);

                    qDebug() <<
                            "Обновление списка сабнетов. Добавление элемента";
                    qDebug() << entrie.ip();
                    qDebug() << interface.name() << entrie.ip() << entrie.netmask() <<
interface.hardwareAddress();
                }
            }
        }
    }
```

Выдержка из NetworkSubnet.cpp

```
void NetworkSubnet::ping(QHostAddress ip)
{
#ifdef Q_OS_WIN
   // Объявляем переменные
   HANDLE hIcmpFile;                  // Обработчик
   unsigned long ipaddr = INADDR_NONE;     // Адрес назначения
   DWORD dwRetVal = 0;                // Количество ответов
   char SendData[32] = "Data Buffer";      // Буффер отсылаемых данных
   LPVOID ReplyBuffer = NULL;              // Буффер ответов
   DWORD ReplySize = 0;               // Размер буффера ответов

   hIcmpFile = IcmpCreateFile();   // Создаём обработчик
   ipaddr = inet_addr(ip.toString().toStdString().c_str());

   // Выделяем память под буффер ответов
   ReplySize = sizeof(ICMP_ECHO_REPLY) + sizeof(SendData);
   ReplyBuffer = (VOID *) malloc(ReplySize);

   // Вызываем функцию ICMP эхо запроса
   dwRetVal = IcmpSendEcho(hIcmpFile, ipaddr, SendData, sizeof(SendData),
                     NULL, ReplyBuffer, ReplySize, 1000);

   if (dwRetVal > 0)
   {
      respondingHosts.append(ip);
   }
#endif

#ifdef Q_OS_LINUX

   QStringList params;
   params << "-c" << "1";
```

```
    params << ip.toString();

    int exitCode = QProcess::execute("ping", params);

    if (exitCode == 0)
    {
        respondingHosts.append(ip);
    }

#endif

    emit execStatusIncrease();
    qDebug() << ip << " is pinged";
}

void NetworkSubnet::pingTest(QHostAddress ip, QHostAddress netmask)
{
    respondingHosts.clear();
    quint32 ipInt = ip.toIPv4Address();
    quint32 maskInt = netmask.toIPv4Address();

    quint32 startAdress = ipInt & maskInt;
    QHostAddress destignationIp = QHostAddress(startAdress);

    int count = countOfPing(netmask);

    QThreadPool *pool = new QThreadPool();
    pool->setMaxThreadCount(255);

    QFuture <void> future;

    int countPerOnce = 1;
```

//Разное количество одновременно возможных пингов обусловлено особенностями ОС. Linux не выдерживает одновременное создание большого количества Pipe'ов

```
#ifdef Q_OS_WIN
   countPerOnce = 255;
#endif

#ifdef Q_OS_LINUX
   countPerOnce = 85;
#endif

   while (true)
   {
      if (count > countPerOnce)
      {
         count = count - countPerOnce;

         for (int i = 0; i < countPerOnce; i++)
         {
            startAdress += 1;

            if (startAdress == ipInt)
            {
               continue;
            }

            destignationIp = QHostAddress(startAdress);
            future = QtConcurrent::run(pool, this, &NetworkSubnet::ping, QHostAddress(destignationIp));
         }

         future.waitForFinished();
         continue;
      }
```

```
            if (count <= countPerOnce)
            {
                for (int i = 0; i < count; i++)
                {
                    startAdress += 1;

                    if (startAdress == ipInt)
                    {
                        continue;
                    }

                    destignationIp = QHostAddress(startAdress);
                    future = QtConcurrent::run(pool, this, &NetworkSubnet::ping,
QHostAddress(destignationIp));
                }

                future.waitForFinished();
                break;

            }
        }

        future.waitForFinished();
        qDebug() << "Дождались всех потоков";
        sortResponse();

        delete pool;
        qDebug() << "Список ip, которые ответили на запрос: \n" <<
                respondingHosts << "\n";

        emit pingTestFinish();
    }
```

ПРИЛОЖЕНИЕ В

Пример элемента посылки в Packages.json


```
"name": "К-ПРД-6",
"request": "00128000000B3148010101",
"response":
[
        {
                "byte": 2,
                "value": "00"
        },
        {
                "byte": 3,
                "value": "12"
        },
        {
                "byte": 7,
                "value": "00"
        }
]
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
«28» 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Программа для экспресс идентификации и диагностики устройств станции спутниковой связи по сети Ethernet
тема

Руководитель	26.06.19 подпись, дата	ведущий инженер, канд. техн. наук должность, ученая степень	В.А. Хабаров инициалы, фамилия
Выпускник	26.06.19 подпись, дата		А.А. Жвакин инициалы, фамилия
Нормоконтролер	27.06.19 подпись, дата	доцент, канд. техн. наук должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2019