Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С ПРОГРАММНЫМ ПАКЕТОМ ВЫСОКОТОЧНОГО ПОЗИЦИОНИРОВАНИЯ RTKLIB»

Автор Кузнецов Андрей Андреевич _____
(фамилия, имя, отчество) (подпись)

Направление подготовки (специальность) 09.04.01 –
(код, наименование)
Информатика и вычислительная техника

Квалификация магистр
(бакалавр, магистр)

Руководитель Соснин Владимир Валерьевич, к.т.н., доцент _____
(Фамилия, И., О., ученое звание, степень) (подпись)

К защите допустить

Зав.кафедрой Муромцев Дмитрий Ильич, к.т.н., доцент _____
(Фамилия, И., О., ученое звание, степень) (подпись)

« _____ » _____ 20 ____ г.

Санкт-Петербург, 2018 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. В настоящее время сложно представить жизнь без спутниковой навигации – данная технология стала неотъемлемой частью деятельности огромного числа людей. Спутниковые системы позволяют определить улицу или дом, где находится человек, или же просто помочь в ориентировании на незнакомой местности. Но использование систем навигации не ограничивается только бытовым применением – данная технология активно применяется для решения задач автоматизации сельскохозяйственных работ, топографических съёмок, а также во множестве других областей.

Точность современных приёмников, установленных в смартфонах или автомобильных навигаторах, в зависимости от условий, при которых осуществлялось определение местоположения, варьируется от трёх до пяти метров [?]. Подобный результат подходит для повседневного применения, например, для ориентации по городу. Однако же, для решения задач более сложных, чем перечисленные выше, необходимы гораздо более точные данные, которые получают, используя технологию *дифференциальной GPS*. Данное решение подразумевает использование сложных алгоритмов, а стоимость представленных на рынке устройств, позволяющих производить подобные расчёты, может превышать 10000 долларов США [1; 2].

Для тех, кому по тем или иным причинам дорогостоящее оборудование недоступно, решением может стать RTKLIB [3] – проект с открытым исходным кодом, реализующий алгоритмы высокоточного позиционирования и поддерживающий работу с общедоступными одночастотными приёмниками. Однако, распространению данного пакета программ мешает неудобство его использования: для управления и мониторинга требуется наличие стационарного компьютера или ноутбука, а программы RTKLIB имеют множество режимов работы, настроек и конфигурационных файлов, что, соответственно, повышает общий порог вхождения.

Объектом исследования является программный пакет высокоточного позиционирования RTKLIB.

Предметом исследования является процесс взаимодействия пользователя с программными компонентами RTKLIB.

Целью исследования является создание приложения, позволяющего взаимодействовать с RTKLIB через веб-браузер. Под взаимодействием понимается возможность наблюдать за состоянием RTK-системы, изменять настройки компонентов RTKLIB, производить геодезические изыскания и сбор данных, а также работать с накопленными файлами логов данных.

Для достижения цели исследования был сформулирован следующий ряд **задач**:

– изучить состав и возможности программного комплекса RTKLIB;

– произвести анализ существующих веб-приложений, предназначенных для работы устройствами, у которых отсутствую органы управления;

– осуществить проектирование и разработку приложения;

– произвести тестирование приложения.

Средствами разработки в представленной работе являются: языки программирования Python и JavaScript для реализации серверной (англ. *back-end*) и клиентской (англ. *front-end*) частей приложения соответственно, открытые JavaScript-библиотеки D3.js, OpenLayers, JavaScript-фреймворк Vue.js. Для организации обмена данными серверной и клиентской частей приложения в реальном времени используются библиотека Socket.IO, принцип работы которой основывается на протоколе WebSocket.

Методологической основой работы послужила гибкая методология разработки (англ. *Agile software development*), ориентированная на итеративный процесс создания программного продукта и учитывающая возможность динамического формирования требований.

Новизна работы обусловлена отсутствием в настоящее время каких-

либо программных продуктов с открытым API, основанных на RTKLIB и позволяющих работать с геодезическим оборудованием через веб-браузер.

Результатом данной работы является рабочая версия приложения, в которой реализованы все необходимые функции, перечисленные в постановке цели исследования. Также была создана и выложена в открытый доступ пользовательская документация, поясняющая основные моменты работы с приложением. Открытый API находится в стадии разработки.

Апробация результатов работы. Наличие документации позволило осуществить открытое тестирование приложения пользователями и, как результат, получить отзывы, сообщения об ошибках и пожелания к функциональности.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Особенности работы GPS-приёмников

Каждый GPS-приёмник определяет свои координаты, основываясь на расстояниях до спутников, с которых он получает сигналы. Данные расстояния вычисляются из времени, которое требуется радиосигналам для прохождения от космических аппаратов до приёмника.

Для установления позиции приёмнику необходимо получать сигналы минимум от четырёх спутников. Каждый из этих сигналов может быть искажён при прохождении через слои атмосферы или при отражении от различных наземных объектов (рис. 1.1) – данные явления вызывают появление ошибок и задержек, что отрицательно сказывается на точности позиционирования.

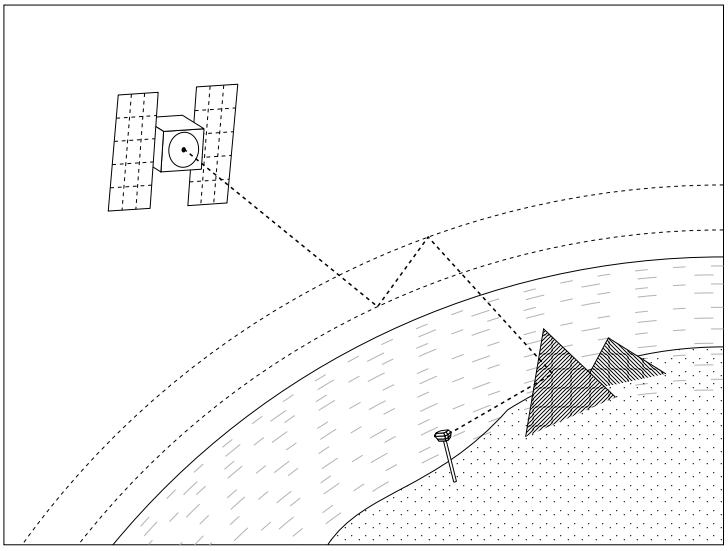


Рисунок 1.1 – Источники ошибок: атмосферные задержки и переотражение сигнала

Важную роль в решении проблемы, описанной выше, играет масштабность системы GPS. Расстояние между наземными объектами и космическими спутниками так велико, что многие расстояния на земле становятся незначительными. Иными словами, если разместить два приёмника на расстоя-

нии нескольких сотен километров друг от друга, то сигналы, которые они будут получать со спутников, будут проходить практически через одну и ту же часть атмосферы, что позволит считать ошибки на обоих приёмниках одинаковыми (рис. 1.2).

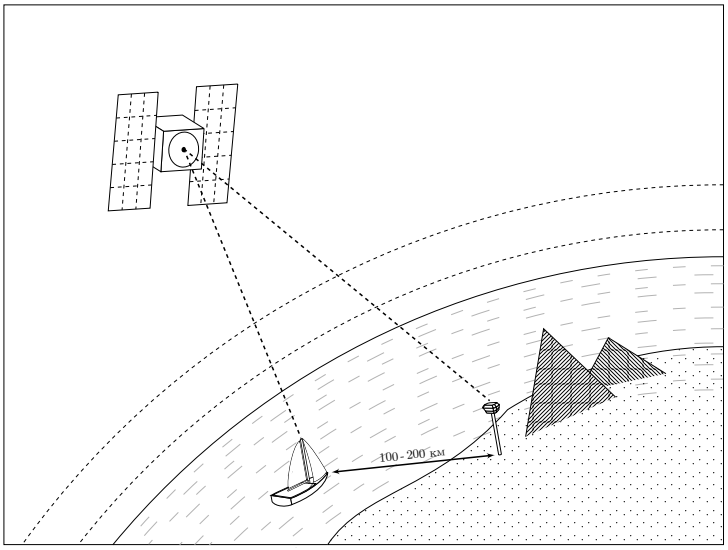


Рисунок 1.2 – Два приёмника получают сигналы с одного и того же спутника

Описанный выше принцип не является лишь теоретическими рассуждениями – данный способ устранения ошибок применяется на практике и является основой *дифференциальной GPS*.

1.2 Дифференциальная GPS и кинематика реального времени

1.2.1 Дифференциальная GPS

Дифференциальная GPS (англ. *Differential Global Positioning System, DGPS*) – система, предназначенная для повышения точности сигналов GPS. Принцип работы данной системы заключается в измерении и учёте разницы между рассчитанной и закодированной псевдодальностями до спутников [4].

Важнейшей особенность DGPS является использование двух приёмников при проведении измерений:

– **База** (англ. *base*) – стационарный приёмник, который находится в точке с заранее рассчитанной координатой. База транслирует данные о разнице

между информацией о позиции, полученной со спутника, и закодированными данными о своём местонахождении.

– **Ровер** (англ. *rover*) – приёмник, с помощью которого производятся какие-либо измерения. Используя данные, полученные с базы, ровер учитывает влияние внешних факторов на расчёт координаты, тем самым получая более точную информацию о своём местонахождении.

Таким образом, работа дифференциальной GPS основана на следующем принципе: считая искажения спутниковых сигналов одинаковыми для близлежащих приёмников, мы получаем возможность вносить поправки в получаемое решение, улучшая результаты измерений.

1.2.2 Кинематика реального времени

Кинематика реального времени (англ. *Real Time Kinematic, RTK*) – режим работы, при котором приём и применение поправок с базы происходят в реальном времени, что позволяет получать результат практически сразу. Важнейшей особенностью данного режима является тот факт, что для обеспечения работы необходима постоянная связь между ровером и базой.

RTK представляется крайне удобным инструментом для задач, решение которых подразумевает использование высокоточного позиционирования. Однако, алгоритмы дифференциальной GPS, обеспечивающие работу RTK, весьма сложны, и производители приёмников предлагают решение в виде закрытого, проприетарного программного обеспечения, встроенного в их продукты. Стоимость приёмников, поддерживающих дифференциальный режим работы, достаточно высока и может превышать 10000 долларов США.

Высокая стоимость оборудования является причиной того, что использование технологии дифференциальной GPS распространено только в таких специфических областях деятельности, как геодезия, земельный кадастр и т.п. Однако, для тех людей, которым по каким-либо причинам недоступны дорогостоящие устройства, решением может стать RTKLIB – программный ком-

плекс, реализующий вышеупомянутые алгоритмы для стандартных, общедоступных GPS-приёмников.

1.3 Программный пакет RTKLIB

RTKLIB – программный пакет с открытым исходным кодом, предназначенный для осуществления стандартного и высокоточного позиционирования с помощью ГНСС (глобальных навигационных спутниковых систем). В состав пакета входит библиотека функций и ряд приложений, использующих данную библиотеку.

Создателем RTKLIB является Томодзи Такасу, профессор Токийского Университета Морских Наук и Технологий. Существенный вклад в кодовую базу проекта был внесён Мишелем Баваро [3; 5], итальянским инженером с огромным опытом работы в сфере спутниковой навигации.

Начиная с релиза первой версии RTKLIB, состоявшегося в 2007 году, проект активно развивается и на данный момент представляет собой систему из восьми приложений, некоторые из которых имеют как графический, так и консольный интерфейс.

Функции компонентов RTKLIB подробно описаны в прилагающейся к проекту официальной документации. Ниже будут рассмотрены наиболее важные для данной работы возможности программного комплекса.

1.3.1 Поддерживаемые спутниковые системы

RTKLIB поддерживает работу с шестью основными системами спутниковой навигации:

- GPS;
- GLONASS;
- Galileo;
- QZSS;
- BeiDou;

– SBAS.

Основной системой, позволяющей осуществлять высокоточное позиционирование является GPS. Остальные системы, как правило, служат источником дополнительных данных, которые помогают опровергнуть или подтвердить получаемое решение.

1.3.2 Режимы работы

Программный комплекс RTKLIB поддерживает несколько режимов работы, каждый из которых предназначен для решения различных навигационных задач. При выборе того или иного режима процесс работы с приёмником может измениться коренным образом. Рассмотрим особенности и назначение основных режимов работы комплекса:

– **Single**, или **Single point positioning** (рис. 1.3). Базовый режим работы, при использовании которого можно получить координаты со стандартной точностью. В режиме Single местоположение определяется с помощью данных одного приёмника – база в данном режиме не используется.

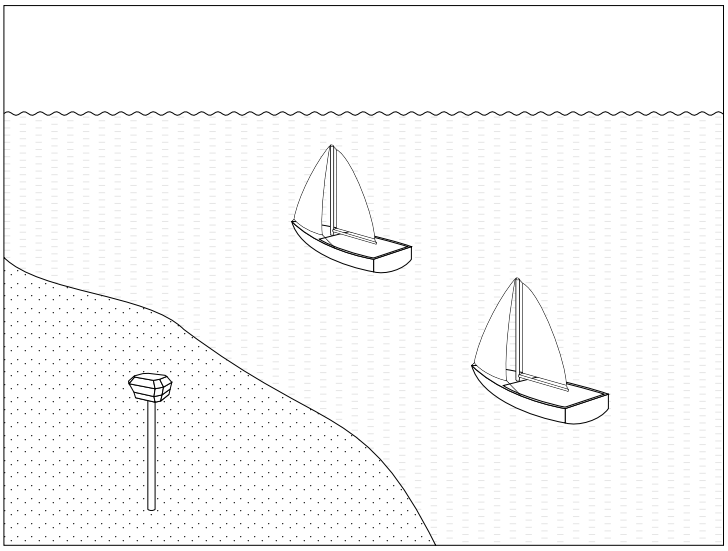


Рисунок 1.3 – Режим Single. Корабли, исполняющие роли роверов, не используют поправки с базы, которая установлена на берегу

При использовании режима Single RTKLIB производит расчёт координаты, используя данные о спутниках навигационной системы. Однако, сами при-

ёмники, как правило, берут на себя задачу расчёта местоположения, т.к. они изначально оптимизированы для получения single-решения.

RTKLIB в режиме Single используется только на приёмниках, которые поддерживают функцию выдачи необработанных (или «сырых») данных: фазовых измерений, псевдодальностей и т.д. При работе в данном режиме RTKLIB создаёт файлы логов «сырых» данных, которые далее можно будет использовать для постобработки (англ. *post-processing*).

– **Static** и **Kinematic** (рис. 1.4). При данных настройках RTKLIB позволяет перейти от постобработки накопленных логов к измерениям в режиме реально времени (RTK). В режимах Static и Kinematic данные ровера используются вместе данными базовой станции, что позволит определять позицию ровера гораздо точнее.

Отличием между режимами Static и Kinematic является подвижность ровера. Если при работе в режиме Static ровер считается неподвижным, то в режиме Kinematic ровер находится в движении.

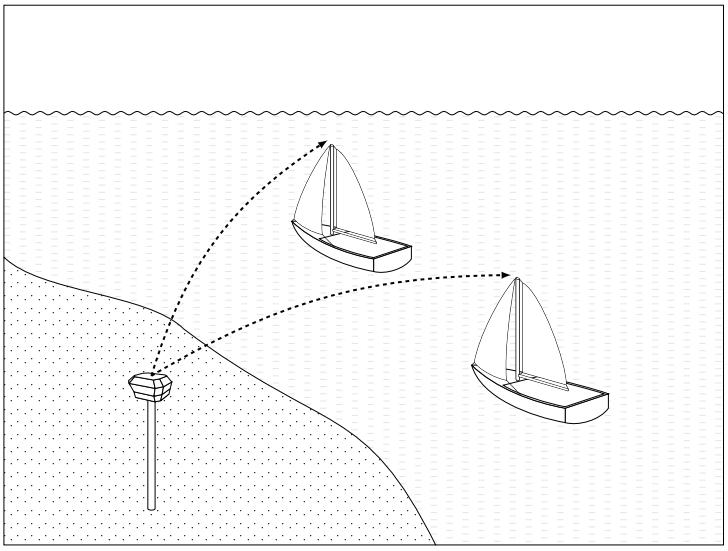


Рисунок 1.4 – Режимы Static и Kinematic. Корабли, исполняющие роли роверов, используют поправки с базы, которая установлена на берегу

Как и в случае Single, в рассматриваемых режимах RTKLIB позволяет записывать логи необработанных данных ровера, однако важно отметить, что па-

раллельно с этим можно также вести лог поправок, получаемых с базы. Сбор всех этих данных может быть крайне полезен для определения и решения проблем связи между двумя приёмниками.

– **Moving-Baseline** (рис. 1.5). Данный режим схож с режимом Kinematic, однако имеет одно ключевое отличие – движущуюся базу. Moving-Baseline не предназначен для улучшения точности абсолютного позиционирования, однако относительно друг друга база и ровер будут получать точные координаты – работая в этом режиме, можно получить аналог GPS компаса.

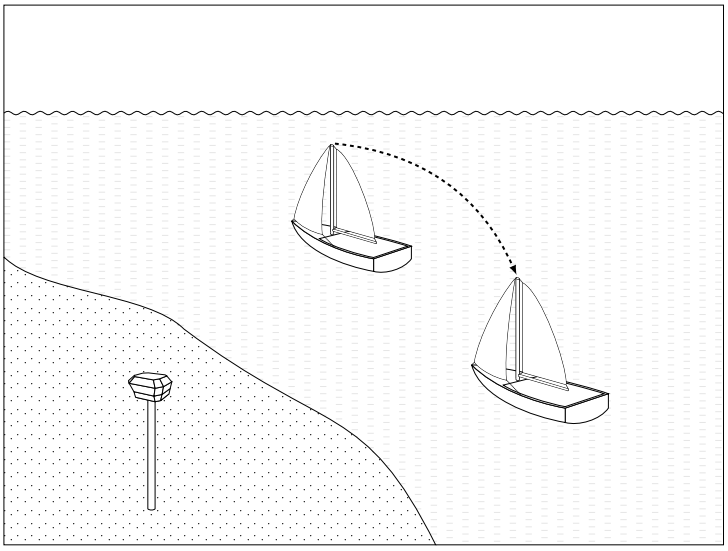


Рисунок 1.5 – Режим Moving-Baseline. Один из кораблей выполняет роль базы и передаёт поправки на корабль-ровер; оба корабля находятся в движении

– **Precise point positioning**, или **PPP**. Режимы работы (PPP-Static, PPP-Kinematic и PPP-Fixed), которые используются для подготовке к работе в режиме RTK.

Среди перечисленных выше PPP режимов интерес для данной работы представляет PPP-Static, который используется для определения позиции будущей базы. Суть работы RTKLIB при таких настройках заключается в накоплении и усреднении информации о текущей позиции приёмника. Проработав таким образом в одной точке несколько часов, приёмник может с хорошей

степенью точности определить координаты своего местоположения и выдать данные для настройки базы.

1.3.3 Поддерживаемые форматы данных

RTKLIB поддерживает множество форматов хранения данных, используемых при работе с ГНСС. Набор поддерживаемых форматов включает в себя как независимые от специфических устройств форматы (стандартные форматы), так и форматы, используемые для выдачи координат лишь GPS-приёмниками определённых производителей (проприетарные форматы).

Стандартные форматы сообщений:

- RINEX (сокр. англ. *Receiver Independent Exchange Format*);
- RTCM (сокр. англ. *Radio Technical Commission for Maritime Services*);
- NMEA (сокр. англ. *National Marine Electronics Association*).

Примеры поддерживаемых проприетарных форматов сообщений:

- UBX (приёмники u-blox);
- NovAtel;
- Hemisphere;
- JAVAD;
- Furuno;
- NVS.

1.3.4 Программы, входящие в состав RTKLIB

Для понимания внутреннего устройства и получения представления о функциональности различных компонентов RTKLIB, следует рассмотреть программы, входящие в состав данного программного пакета. Последние версии RTKLIB – 2.4.2 и 2.4.3 beta – включают в себя восемь приложений, отвечающих за различные функции комплекса (таблица 1.1).

В рамках данной работы рассматриваются три приложения RTKLIB, отвечающие за ключевые функции, необходимые при решении задач навигации и позиционирования: RTKNAVI, STRSVR и RTKCONV.

Таблица 1.1 – Программы, входящие в состав RTKLIB

№	Назначение	GUI*	CUI**
1	Запуск приложений RTKLIB	RTKLAUNCH	-
2	Позиционирование в режиме реального времени	RTKNAVI	RTKRCV
3	Мультиплексор потоков данных	STRSVR	STR2STR
4	Постобработка данных	RTKPOST	RNX2RTKP
5	Конвертер данных	RTKCONV	CONVBIN
6	Графическое отображение полученных данных	RTKPLOT	-
7	Скачивание данных ГНСС	RTKGET	-
8	NTRIP браузер	SRCTBLBROWS	-

* Графическое приложение
** Консольное приложение

– **RTKNAVI** – программа, обеспечивающая работу приёмника в режиме RTK. RTKNAVI выполняет все задачи, возлагаемые на ровер:

1) приём необработанных данных с GPS-приёмника;
2) приём поправок с базы;
3) ведение логов полученных данных;
4) обработка данных с помощью RTK алгоритмов;
5) выдача полученного решения в указанном формате.

Также, с помощью RTKNAVI пользователь получает наглядную информацию о качестве текущего решения и о сигналах, получаемых ровером и базой со спутников.

RTKNAVI имеет консольный аналог – приложение **RTKRCV**. Текстовое версия поддерживает все функции графического приложения, включая вывод информации о решении, спутниках и текущих координатах.

Важным отличием двух вариантов приложения является способ установки настроек – если RTKRCV имеет отдельное окно с графическими элементами

управления, то RTKRCV настраивается только с помощью конфигурационных файлов, работа с которыми требует от пользователя определённых навыков.

– **STRSVR** (в консольная версии – **STR2STR**) является мультиплексором потоков данных, который позволяет создать до трёх каналов передачи связи, в которые будет направлен вывод подключённого к компьютеру GPS-приёмника.

STRSVR предоставляет широкий набор возможностей по перенаправлению данных:

1) передача через COM-порты;
2) передача через TCP сокеты;
3) передача через NTRIP-серверы;
4) запись в файл.

Также STRSVR обеспечивает (для каждого из выходных потоков) конвертацию входящих данных в указанный пользователем формат. Только STRSVR может осуществлять выдачу информации в легковесном формате RTCM3, что необходимо для обеспечения работы в режиме базы.

– **RTKCONV**, или его консольный аналог **CONVBIN** – конвертер данных, получаемых со спутников. Одно из самых важных применений данной программы – конвертация файлов логов из форматов, привязанных к конкретным приёмникам, в независимый формат RINEX.

Использование RTKCONV может быть обусловлено необходимостью проведения дальнейшего анализа и постобработки данных с использованием специального программного обеспечения.

1.4 Основные проблемы использования RTKLIB

RTKLIB предоставляет широчайший спектр возможностей, а исходный код данного проекта является открытым. Однако, данный комплекс распространён достаточно слабо и не является широко известным. Данный факт свя-

зан отнюдь не с низким качеством результатов работы RTKLIB, а, в первую очередь, с удобством использования данного программного продукта.

Графические версии приложений RTKLIB возможно запустить только в среде операционной системы семейства Windows. Однако, достаточно проблематично обеспечить запуск графических Windows-приложений, к примеру, на беспилотном летательном аппарате или на ГНСС-приёмнике. Но полностью раскрыть потенциал RTK можно именно при интеграции программного обеспечения в технику. Данный факт делает консольные версии программ RTKLIB более привлекательным вариантом.

Входящие в состав пакета текстовые приложения не требуют больших вычислительных ресурсов и могут быть запущены на огромном количестве платформ под управлением GNU/Linux. Но важно понимать, что для работы с консольными приложениями необходимо постоянно поддерживать работу одного или нескольких терминалов, а также иметь компьютер с клавиатурой и доступом к данным приёмника.

Вне зависимости от того, используются ли графические или консольные версии приложений, невозможно использовать обширные возможности RTKLIB без знания особенностей процесса конфигурирования данного комплекса. Огромное количество настроек вкупе с различными режимами работы могут вызвать трудности при освоении RTKLIB новыми пользователями, а разнообразные конфигурационные файлы и неочевидные с первого взгляда решения в дизайне графических настроек способны запутать даже опытных пользователей.

Таким образом, можно выделить следующий ряд трудностей, которые могут возникнуть при использовании RTKLIB:

– работа с комплексом возможна только при наличии компьютера с клавиатурой;

– графические приложения, входящие в состав пакета, можно запу-

стить только при использовании операционной системы семейства Windows;

– конфигурационные файлы консольных версий программ имеют достаточно сложный синтаксис;

– большое количество доступных настроек может затруднить освоение комплекса новыми пользователями.

В качестве решения проблем, перечисленных выше, предлагается создать веб-приложение, с помощью которого можно было бы осуществлять контроль над работой RTKLIB, используя практически любое устройство с веб-браузером. Работа подобного приложения будет обеспечиваться с помощью дополнительного программного обеспечения, добавленного на вычислительный модуль, на котором запущен RTKLIB. Осуществляя с помощью такого программного обеспечения обмен командами и информацией между веб-интерфейсом и RTKRCV, можно осуществлять работу с RTKLIB, используя смартфон или планшет.

1.5 Обзор существующих веб-приложений, предназначенных для работы с устройствами без органов управления

В настоящее время существует достаточно большое количество электроники, единственным способом управления которой является специальный веб-интерфейс. Примером подобных устройств служат маршрутизаторы и одноплатные компьютеры.

В рамках проведённого обзора были рассмотрены популярные операционные системы, позволяющие осуществлять управление устройством, на которое они установлены, через веб-браузер:

– **OpenWrt** – дистрибутив GNU/Linux, предназначенный для установки на маршрутизаторы.

– **Windows 10 IoT Core** – версия операционной системы Windows, созданная специально для встраиваемых решений.

1.5.1 OpenWrt

Работа с маршрутизаторами, находящимися под управлением OpenWrt, производится через специальный веб-интерфейс под названием **LuCI**. Данный интерфейс представляет собой одностраничное веб-приложение, доступное через браузер любого устройства, подключённого к сети маршрутизатора. LuCI позволяет пользователю:

– произвести настройку маршрутизатора;

– просмотреть информацию о маршрутизаторе и установленном программном обеспечении;

– просмотреть информацию о подключённых устройствах и трафике.

В интерфейсе LuCI доступ к различным страницам и настройкам осуществляется через меню в верхней части веб-страницы (рис. 1.6). Все пункты верхнего меню, за исключением кнопки выхода из системы, представляют собой выпадающие меню, позволяющие перейти на одну из страниц соответствующей категории.

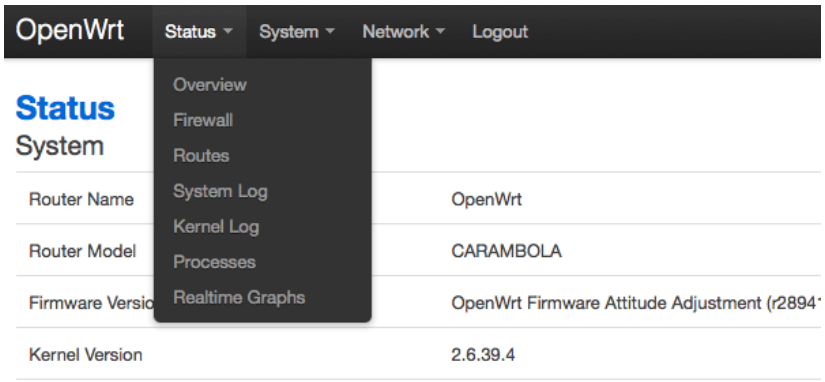


Рисунок 1.6 – Главное меню LuCI

Благодаря разнообразным веб-элементам и возможностям JavaScript в LuCI пользователь получает возможность удобно редактировать различные настройки маршрутизатора с помощью интерактивных форм (рис. 1.7), что позволяет производить конфигурацию устройства, используя практически любой компьютер или смартфон.

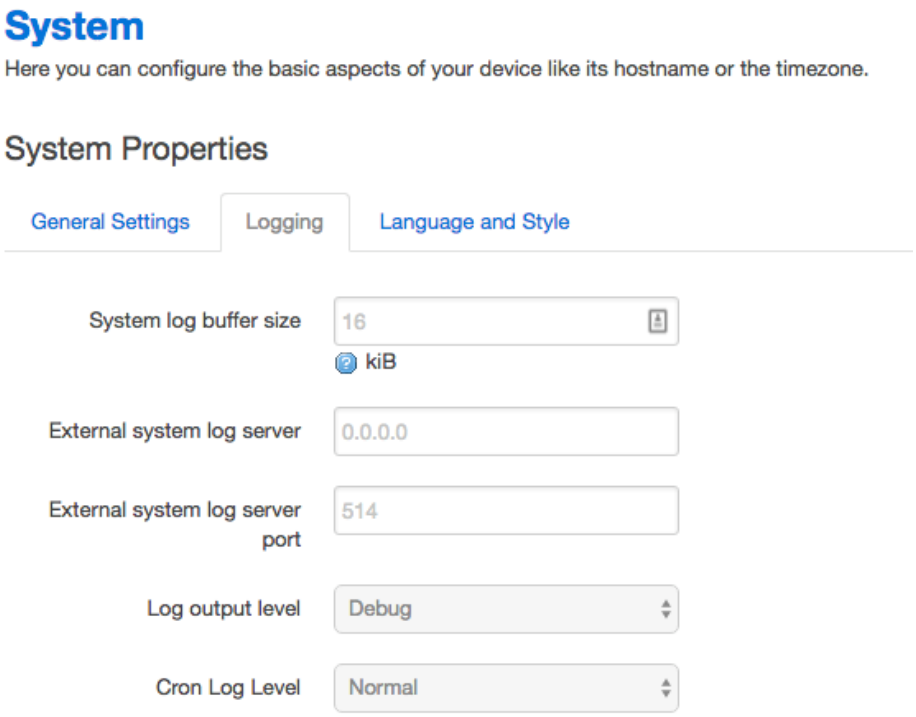


Рисунок 1.7 – LuCI: Форма настройки системных логов

Также, OpenWrt демонстрирует прекрасный пример применения современных веб-технологий для визуализации данных. Через интерфейс LuCI пользователь может наблюдать за текущей загрузкой сети с помощью специального графика (рис. 1.8).

1.5.2 Windows 10 IoT Core

Windows 10 IoT Core позволяет использовать операционную систему Windows на таких устройствах, как Raspberry PI или MinnowBoard. Так как подобные встраиваемые решения зачастую не имеют экрана и клавиатуры, взаимодействие пользователя с системой осуществляется с помощью веб-страницы (рис. 1.9).

Различные функции, предоставляемые системой, доступны пользователю через специальное боковое меню (рис. 1.10). При переключении между пунктами меню веб-приложение отображает страницы с различными панелями управления.

Богатые возможности веб-технологий позволяют отобразить в браузе-

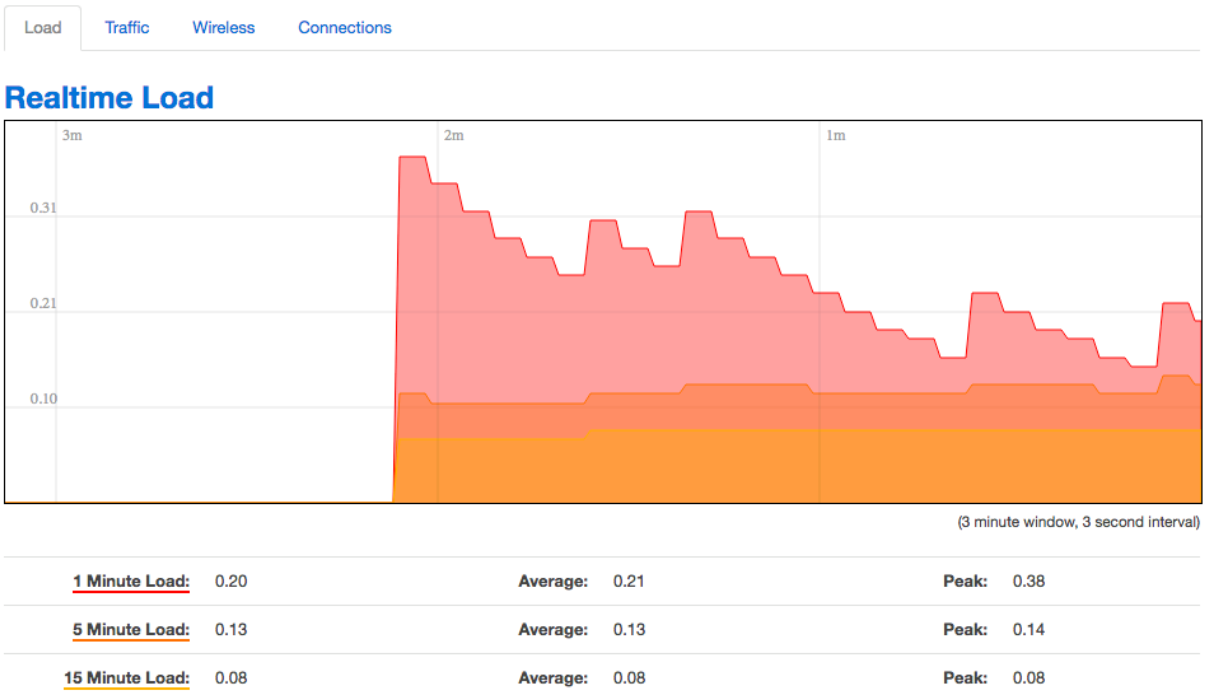


Рисунок 1.8 – LuCI: График загрузки сети

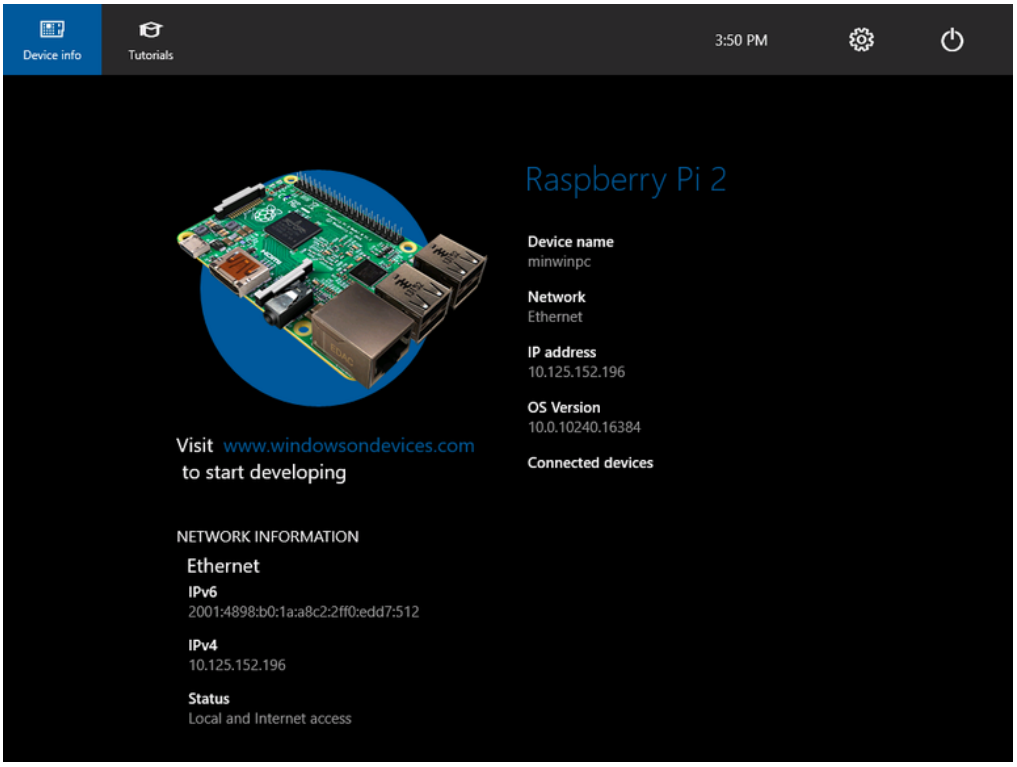


Рисунок 1.9 – Главная страница Windows 10 IoT Core

ре привычные для операционной системы Windows утилиты: менеджер установленных приложений, диспетчер задач, панель настройки беспроводных

подключений, менеджер подключённых устройств (рис. 1.10) и т.д.

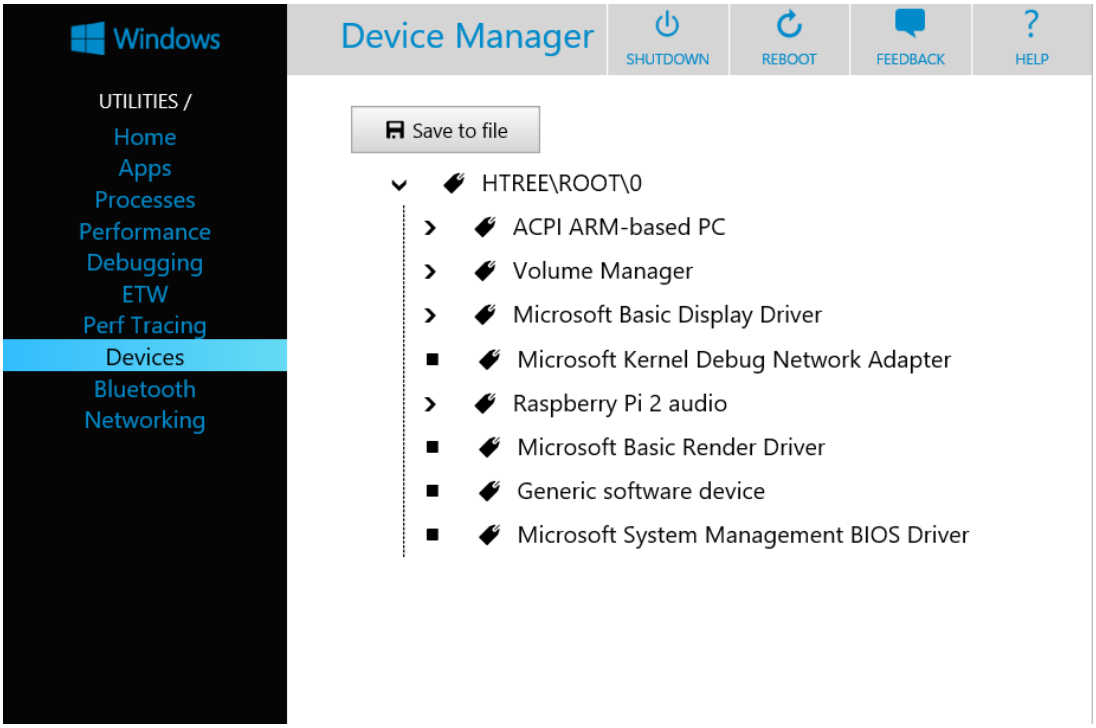


Рисунок 1.10 – Windows 10 IoT Core: Менеджер подключённых устройств

1.6 Выводы по разделу 1

На основании проведённого обзора и анализа, были сделаны следующие выводы:

– Программный комплекс RTKLIB является адекватной альтернативой для проприетарного программного обеспечения ГНСС-приёмников, однако его распространению мешают высокий порог вхождения пользователя, отсутствие кроссплатформенных графических версий приложений и сложность использования во встраиваемых решениях.

– Возможным решением задачи обеспечения взаимодействия с RTKLIB, встроенного в GNSS-приёмник, является создание веб-приложения, доступного через браузер практически любого устройства, при условии нахождения в одной сети с приёмником.

– Одностраничные приложения являются наиболее удачным решением при создании веб-интерфейсов для устройств без органов управления. От-

сутствие долгой загрузки при переключении между различными представлениями и получение необходимых данных с помощью асинхронных запросов благоприятно влияют на опыт пользователя (англ. *User experience, UX*).

Проведённый обзор показал распространённость использования веб-приложений для управления встраиваемыми решения, что ещё раз подчёркивает актуальность предлагаемой работы. При учёте сделанных наблюдений и выводов становится возможно создание браузерного приложения, которое существенно облегчит использование GNSS-приёмника, работа которого основана на использовании RTKLIB.

2 ОБЗОР ПЛАТФОРМЫ ДЛЯ РАЗРАБОТКИ И ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

2.1 Платформа для разработки

Разработка приложения будет осуществляться на платформе продуктов компании Emlid [6]: ГНСС модуля Reach [7] и ГНСС приёмника Reach RS [8]. Основой данных устройств являются вычислительный модуль Intel Edison и плата, на которую установлен ГНСС модулю компании u-blox. Intel Edison работает под управлением GNU/Linux, что позволяет использовать многочисленные средства разработки, доступные для дистрибутивов данной операционной системы.

Устройства Reach были анонсированы в 2015 году, а их разработка была финансирована краудфандинговой кампанией [?]. Первые модули поставлялись пользователям с предустановленным веб-интерфейсом, созданным с помощью библиотек jQuery [?] и jQuery Mobile [?].

Данный веб-интерфейс был необходим для ознакомления пользователей с возможностями ГНСС-модуля, однако использование устаревшей технологии веб-разработки и отзывы пользователей, сообщающие о недостаточной функциональности приложения, вызвали необходимость создания принципиально нового решения.

При разработке приложения на платформе указанных выше устройств важно учитывать следующий факт: несмотря на то, что Reach и Reach RS созданы на базе одного и того же вычислительного модуля и используют одинаковые приёмники u-blox, имеется ряд существенных различий в аппаратном обеспечении данных устройств [7; 8]. Различия Reach и Reach RS, которые необходимо учесть при создании веб-приложения, указаны в таблице 2.1.

Наличие (отсутствие) тех или иных возможностей у Reach или Reach RS отразится на клиентской части разрабатываемого решения в виде специфи-

Таблица 2.1 – Различия Reach и Reach RS

Техническая/функциональная особенность	**Reach**	**Reach RS**
Встроенная батарея	нет	да
Встроенная антенна	нет	да
Встроенное радио	нет	да
Возможность управления фотокамерой	да	нет

ческих форм и элементов интерфейса, отображение которых будет зависеть от того, на каком из устройств запущено приложение.

2.2 Общая архитектура приложения

2.2.1 Задачи и требования

Для создания общей архитектуры приложения необходимо определить наиболее важные идеи, задачи и требования, предъявляемые к разрабатываемому продукту.

Важно отметить, что в следующем далее списке присутствуют пункты, которые касаются лишь разработчиков программных компонентов, отвечающих за взаимодействие с RTKLIB, системными утилитами и различными аппаратными компонентами Reach и Reach RS. Данные модули и детали их разработки выходят за рамки рассматриваемой работы, но являются чрезвычайно важными для понимания общей структуры приложения.

Разрабатываемое приложение должно:

– **Осуществлять запуск приложений RTKLIB в управляемых контейнерах**. Надёжным и простым способом организации взаимодействия с приложениями RTKLIB является их запуск в управляемых контейнерах. При данном подходе необходимые утилиты будут работать так же, как если бы пользователь запускал их в терминале.

Используя подобный подход, становится возможно организовать взаимодействие веб-приложения с необходимыми программами таким образом, что:

1) не требуется вмешательство в исходный код RTKLIB;

2) облегчается поддержка совместимости с новыми версиями программного комплекса;

3) разрабатываемый продукт остаётся независим от изменений в кодовой базе RTKLIB.

– **Иметь клиентскую часть, представленную в виде одностраничного приложения**. Основываясь на выводах, сделанных в разделе 1, было принято решение реализовать клиентскую часть разрабатываемого решения в виде одностраничного приложения.

Как показал проведённый обзор, данный подход в настоящее время является одним из самых популярных решений для создания веб-интерфейсов, предназначенных для управления такими устройствами, как Reach или Reach RS.

– **Поддерживать передачу данных по протоколу WebSocket**. Для создания отзывчивого интерфейса веб-приложения и обеспечения лучшего пользовательского опыта бо̲льшую часть взаимодействий клиентской части с сервером следует организовать с помощью асинхронных запросов и сообщений.

Наиболее удачным решение для разрабатываемого приложения является протокол WebSocket. Создавая на клиентской части веб-приложения слушателей определённых событий, становится возможно организовать отображение различной информации в режиме реального времени без постоянных опросов сервера.

2.2.2 Основные модули

Перечислим основные модули приложения (рис. 2.1):

а) Серверная часть

1) WebSocket сервер

2) Модуль взаимодействия с RTKLIB

3) Демоны и сервисы для взаимодействия с аппаратными компонентами устройства

б) Клиентская часть

1) WebSocket клиент

2) JavaScript-приложение

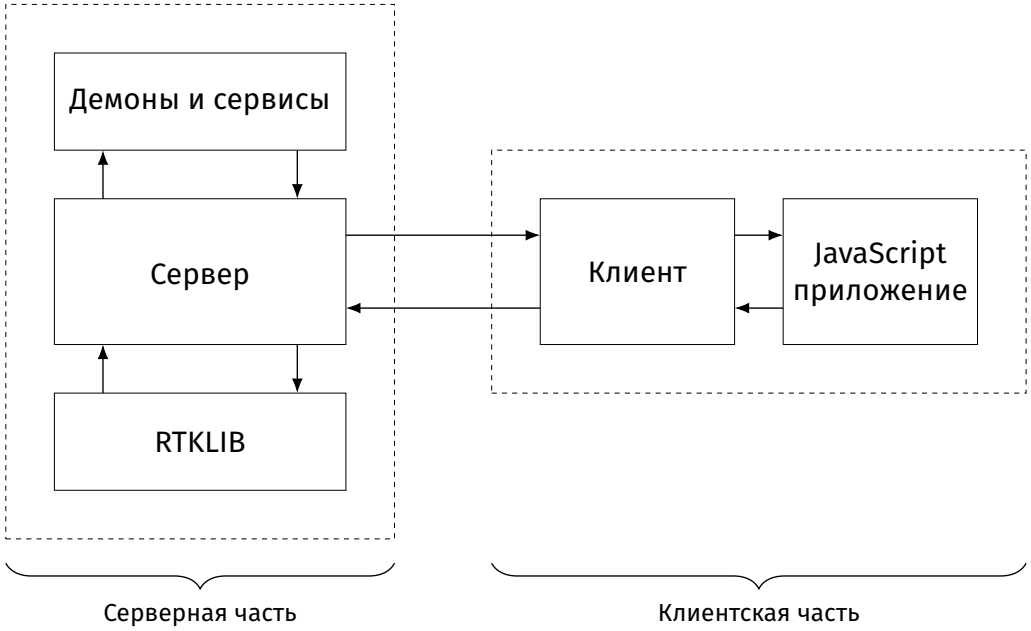


Рисунок 2.1 – Общая архитектура приложения

2.3 Выбор инструментов разработки

2.3.1 Серверная часть приложения

В рамках разработки описанного приложения выбор инструментов для написания серверной части приложения зависит не только от возможностей веб-фреймворков, доступных для того или иного языка программирования, но и от задач, описанных в подразделе 2.2. Также, важно помнить, что разрабатываемое приложение предназначено для запуска на устройствах с ограниченными вычислительными ресурсами.

Разработчиками серверной части приложения был проведён обзор высокоуровневых языков программирования, часто используемых для разра-

ботки веб-приложений. Основными из рассмотренных вариантов были языки Java, Python и Ruby.

На основании различных оценок, детали формирования которых выходят за рамки рассматриваемой работы, основным языком для серверной части приложения был выбран Python.

Одним из основных инструментов при написании серверной части приложения был выбран веб-фреймворк Flask [?]. Запуск приложений RTKLIB в управляемых контейнерах осуществляется с помощью модуля rexpect [?].

2.3.2 Клиентская часть приложения

2.3.2.1 Основные средства

Основу клиентской части приложения составляют: язык гипертекстовой разметки HTML, CSS-стили и скрипты на языке JavaScript.

Для наиболее быстрой и удобной разработки одностраничного приложения было решено использовать специализированную библиотеку или фреймворк. На момент проектирования приложения одними из самых популярных [?] инструментов для создания одностраничных приложений являлись React [?], Angular 2 [?], Vue.js [?] и Knockout [?].

Для анализа и сравнения вышеперечисленных библиотек и фреймворков были изучены:

- официальные документации;
- исходные коды простейших приложений и их компонентов;
- результаты синтетических тестов [?].

По результатам изучения возможных решений для разработки приложения был выбран фреймворк Vue.js. Данный выбор также был обусловлен наличием у автора данной работы опыта разработки с использованием фреймворка AngularJS, который является одним из источников вдохновения для создателей Vue.js [?].

Также, вместе с Vue.js было решено использовать библиотеку-расши-

рение для данного фреймворка – Vuex [?]. Данная библиотека позволяет создать централизованное хранилище данных, доступ к которому будут иметь все компоненты приложения. Одной из важнейших особенностей работы с таким хранилищем является способность компонентов Vue.js реактивно обновлять своё состояние при изменении тех или иных данных в Vuex.

2.3.2.2 Работа с картами

В качестве решения для задачи отображения карт и всевозможной информации на картах был выбран проект OpenLayers [?]. Данная библиотека предоставляет широкие возможности для работы с картами:

– выбор источника графических тайлов карты;

– управление отображением элементов управления картой;

– модификация способов взаимодействия пользователя с картой;

– координатная сетка и шкала масштаба, доступные по умолчанию;

– возможность отрисовки произвольных фигур и изображений на различных слоях карты.

2.3.2.3 WebSockets

Для работы с WebSocket была выбрана библиотека Socket.IO [?]. Данный выбор обусловлен не только популярностью библиотеки, но и наличием полной поддержки Socket.IO на серверной части приложения – для веб-фреймворка Flask существует расширение Flask-SocketIO, позволяющее серверу обмениваться сообщениями с любыми клиентами Socket.IO.

2.4 Детализированная архитектура приложения

С учётом инструментов, выбранных в предыдущем подразделе, уточним архитектуру приложения, представленную ранее (рис. 2.1). На рисунке 2.2 изображена детализированная архитектура разрабатываемого приложения.

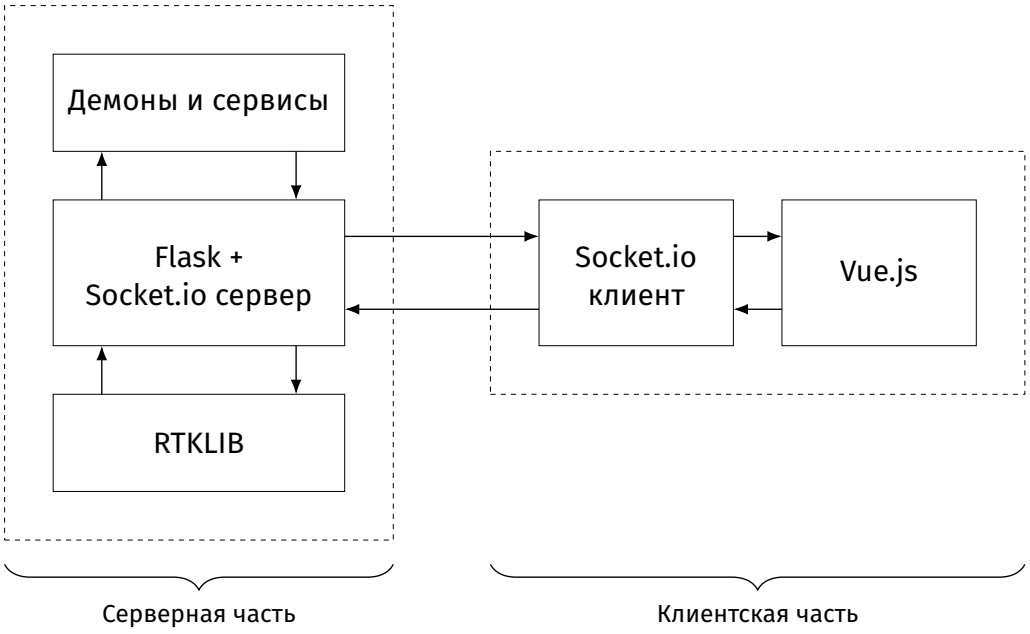


Рисунок 2.2 – Детализированная архитектура приложения

2.4.1 Подробная архитектура клиентской части приложения

Клиентская часть разрабатываемого приложения представляет из себя совокупность отдельных программных модулей и глобального хранилища, изменения данных в котором вызывают реактивное обновление модулей, имеющих соответствующие слушатели событий.

Модули приложения можно условно разделить на две группы:

– модули, являющиеся *моделями представления* для соответствующих экранов приложения;

– модули общего назначения, предназначенные для обработки событий, приходящих от сервера, или содержащие различные утилиты.

На рисунке 2.3 изображены все основные компоненты приложения и их связи с глобальным хранилищем. К модулям общего назначения относятся только модуль обработки событий и модуль работы с картами, остальные компоненты написаны с помощью Vue.js и связаны с представлениями приложения.

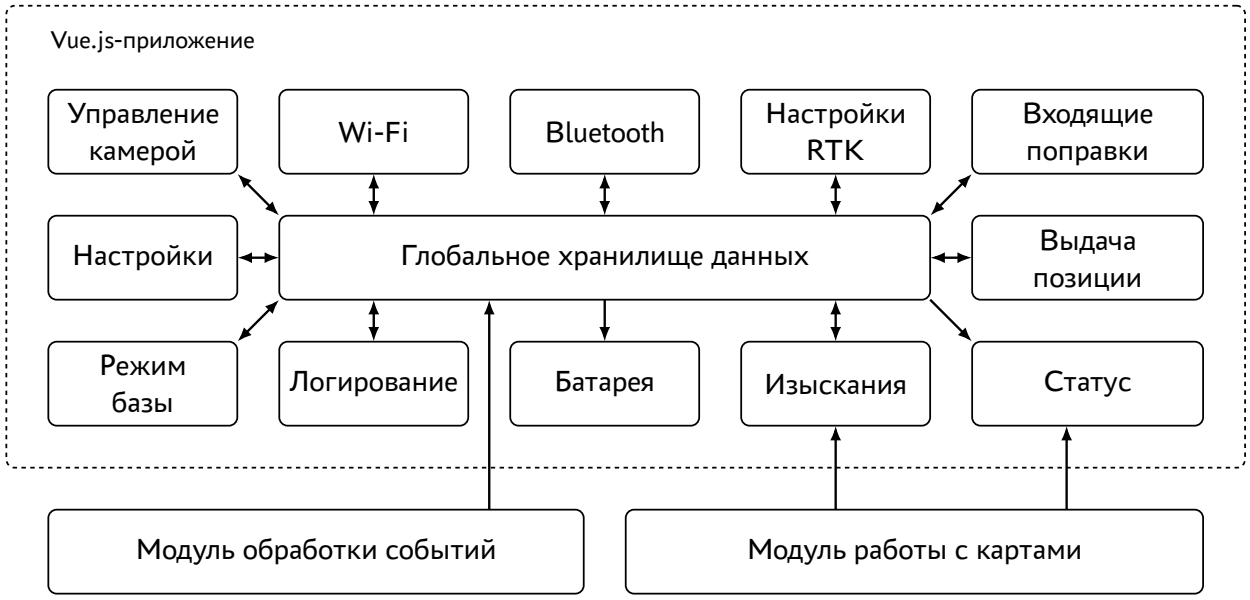


Рисунок 2.3 – Архитектура клиентской части приложения

2.5 Описание модулей приложения и основные требования, предъявляемые к ним

2.5.1 Статус

Данный модуль служит для обработки, форматирования и отображения основной информации, связанной с работой приёмника:

– значения отношений сигнал/шум (англ. *Signal-to-noise ratio, SNR*) для принимаемых со спутников сигналов (для ровера и базы);

– режим работы RTKLIB;

– качество получаемого решения;

– текущие координаты ровера;

– скорость движения ровера;

– параметры RTK;

– координаты базы.

Также, модуль должен осуществлять сбор и хранение истории изменения координат ровера для отображения данной информации на интерактивной карте.

2.5.2 Изыскания

Модуль, реализующий инструменты для проведения полевых работ и предоставляющий пользователю следующие возможности:

- создание и удаление проектов;
- экспорт проектов в различных форматах;
- сбор точек;
- просмотр данных проекта на интерактивной карте.

2.5.2.1 Создание проекта

При создании нового проекта пользователь должен:

- указать название проекта;
- указать имя автора проекта;
- добавить описание проекта (опционально);
- ввести высоту антенны по умолчанию;
- создать до трёх правил для автоматического сбора точек.

Правила для автоматического сбора точек представляют собой условия, при которых устройство автоматически будет принимать результаты усреднения координат. Параметры правил:

- статус решения (Single, Float или Fix);
- минимальное время сбора (от одной секунды до одного часа);
- максимальное допустимое значение СКО;
- максимальное допустимое значение DOP.

2.5.2.2 Экспорт проекта

Экспорт проектов должен быть доступен в следующих форматах:

- GeoJSON;
- ESRI Shapefile;
- DXF;
- CSV;

– DroneDeploy CSV.

2.5.3 Настройки RTK

Модуль приложения, необходимый для настройки параметров RTK и используемых ГНСС.

Настраиваемые параметры RTK:

– режим позиционирования (Single, Static или Kinematic);

– метод разрешения фазовой неоднозначности;

– угол отсечки спутников (от 0 до 30 градусов);

– маска для отношения сигнал/шум (от 0 до 40);

– максимальные возможные горизонтальные и вертикальные ускорения ГНСС-антенны (от 0 до 10 м/с^2).

Настройки используемых ГНСС:

– выбор ГНСС из списка, представленного в пункте 1.3.1;

– выбор частоты обновления навигационных данных (1, 5, 10 или 14 Гц).

При настройке необходимо соблюдать следующие ограничения:

а) получение данных GPS невозможно отключить;

б) невозможно одновременно включить системы ГЛОНАСС и BeiDou;

в) невозможно настроить получение данных от ГЛОНАСС и BeiDou при частоте обновления выше 5 Гц.

Ограничения, указанные в пунктах б) и в) списка выше, обусловлены аппаратными ограничениями приёмника u-blox, установленного в устройствах Reach и Reach RS.

2.5.4 Входящие поправки

Данный модуль необходим для настройки получения ровером поправок с базы. В список задач, решаемых данным компонентом, входит выбор способа получения поправок и настройка значений, специфичных для каждого из них.

Способы получения поправок и элементы их конфигурации:

- последовательный порт
    1) устройство (UART, USB-to-PC, USB-OTG);
    2) скорость передачи (бод);
    3) формат данных.
- NTRIP
    1) адрес;
    2) порт;
    3) имя пользователя (опционально);
    4) пароль (опционально);
    5) точка подключения;
    6) формат данных.
- TCP соединение
    1) роль (сервер, клиент);
    2) адрес;
    3) порт;
    4) формат данных.
- Bluetooth
    1) формат данных.
- LoRa
    1) частота;
    2) выходная мощность;
    3) пропускная способность;
    4) формат данных.

2.5.5 Выдача позиции

Модуль, позволяющий настроить до двух потоков данных, содержащих информацию о текущей позиции устройства. Как и в случае с настройкой входящих поправок, рассматриваемый модуль предоставляет инструменты для выбора способа передачи данных и их конфигурации.

Способы передачи данных и элементы их конфигурации:

– последовательный порт

1) устройство (UART, USB-to-PC, USB-OTG);
2) скорость передачи (бод);
3) формат данных.

– TCP соединение

1) роль (сервер, клиент);
2) адрес;
3) порт;
4) формат данных.

– Bluetooth

1) формат данных.

2.5.6 Режим базы

Модуль, предназначенный для настройки параметров, специфичных для устройства, работающего в режиме базы. Содержит в себе инструменты для:

– настройки потока данных с поправками (способы и их настройки);

– выбора типов передаваемых RTCM3-сообщений и частоты их передачи;

– установки точных координат будущей базы (автоматически и вручную).

2.5.6.1 Передача данных

Способы передачи данных и элементы их конфигурации:

– последовательный порт

1) устройство (UART, USB-to-PC, USB-OTG);
2) скорость передачи (бод).

– NTRIP

1) адрес;
2) порт;
3) имя пользователя и пароль (опционально);

4) точка подключения.

– TCP соединение

1) роль (сервер, клиент);
2) адрес;
3) порт.

– Bluetooth (не имеет настроек)

– LoRa

1) частота;
2) выходная мощность;
3) пропускная способность.

2.5.6.2 Передаваемые сообщения

Доступные для передачи типы RTCM3-сообщений: 1002, 1006, 1008, 1019, 1020, 1010, 1097, 1107, 1117, 1127. Описание типов RTCM-сообщений представлено в [?] и [?]

Каждый из перечисленных типов сообщений может передаваться с частотой 0.1, 0.5, 1, 2, 5 или 10 Гц.

При использовании LoRa, максимальный объём передаваемых RTCM3-сообщений должен быть ограничен в зависимости от выбранной пропускной способности.

2.5.6.3 Координаты базы

Установку координат будущей базы возможно осуществить двумя способами:

– ручной ввод (WGS84 или ECEF координаты);

– автоматическое усреднение (при статусе решения Single, Float, Fix).

Время автоматического усреднения позиции может составлять от 6 секунд до 30 минут.

2.5.7 Логирование

Данный модуль необходим для управления записью и историей логов данных. Модуль решает следующие задачи:

– включение (выключение) записи и выбор формата логов определённого типа;

– получение списка доступных логов;

– предоставление пользователю инструментов для удаления и скачивания файлов логов.

Типы логов и доступные для них форматы:

– Лог необработанных данных (англ. *raw data*). Доступные форматы:

1) UBX;
2) RINEX (версии 2.10-2.12 и 3.00-3.03).

– Лог позиций устройства. Доступные форматы:

1) LLH;
2) XYZ;
3) ENU;
4) NMEA;
5) ERB.

– Лог поправок, полученных с базы. Доступные форматы:

1) RTCM3;
2) RINEX (версии 2.10-2.12 и 3.00-3.03).

2.5.8 Управление камерой

Модуль, позволяющая произвести настройку управления камерой для осуществления аэрофотосъёмки. Данная возможность доступна только для устройств Reach.

Основными задачи модулю являются:

– включение (отключение) подачи сигнала камере;

– управление параметрами широтно-импульсно модулированного управляющего сигнала;

– выдача временно́й отметки последнего срабатывания камеры.

Параметры управляющего сигнала:

– период импульсов (от 1 до 30 секунд);

– длительность импульса (от 5 до 100 миллисекунд);

– полярность сигнала.

2.5.9 Wi-Fi

Модуль управления сетевыми подключениями, который решает следующие задачи:

– перевод устройства в режим беспроводной точки доступа;

– управление списком сохранённых (известных устройству) Wi-Fi-сетей;

– предоставление пользователю интерфейса для подключения устройства к Wi-Fi-сетям;

– вывод информации о текущем подключении.

2.5.10 Bluetooth

Данный модуль необходим для управления подключениями к различным устройствам с помощью Bluetooth. Задачи, решаемые модулем:

– включение (отключение) Bluetooth;

– включение (отключение) возможности обнаружения другими устройствами;

– установка PIN-кода (от 1 до 6 цифр);

– сопряжение (англ. *pairing*) с другими устройствами;

– управление списком сопряжённых устройств.

2.5.11 Настройки

Модуль, содержащий инструменты для общей настройки устройства и различные утилитарные возможности.

Общие настройки:

– выбор канал получения обновлений (канал со стабильными версиями или канал с ранним доступом к новым возможностям);

– изменение имени устройства;

– настройка поведения при заполнении памяти.

Утилиты:

– генерирование отчётов о состоянии устройства (в виде текста или архива);

– полный сброс настроек;

– включение (выключение) «ночного» режима работы (при включении данного режима устройство отключает светодиодные индикаторы).

2.5.12 Батарея

Модуль, предназначенный для вывода информации о текущем состоянии встроенного аккумулятора на устройствах Reach RS.

Отображаемая информация:

– напряжение (вольт);

– сила тока (ампер);

– уровень заряда батареи (проценты);

– температура датчика STC (градусы Цельсия);

– ограничение силы тока USB (миллиампер);

– ограничение силы тока разъёма питания (миллиампер);

– USB OTG кабель (подключён/отключён);

– текстовый статус зарядки батареи;

– температура датчика LTC (градусы Цельсия).

2.5.13 Модуль обработки событий

Модуль служит для регистрации слушателей событий WebSocket, содержащих данные которые:

– необходимы сразу для нескольких (или всех) модулей приложения;

– влияют на глобальное состояние приложения.

Данные, получаемые модулем с помощью событий WebSocket:

– координаты ровера и базы;

– скорость перемещения ровера;

– параметры RTK-системы;

– качество получаемого решения;

– данные о видимых спутниках (отдельно для ровера и базы);

– информация о входящих и исходящих потоках данных;

– IP-адрес устройства;

– информация о потере (восстановлении) соединения с устройством;

– информация об обновлении разрешённого диапазона частот.

2.5.14 Модуль работы с картами

Данный модуль представляет собой обёртку над компонентами библиотеки OpenLayers. Наличие подобного компонента в приложении обусловлено необходимостью быстро и удобно создавать интерактивные карты для различных экранов приложения.

Модуль должен предоставлять конструктор для создания нового экземпляра карты OpenLayers. Единственным необходимым параметром конструктора должен быть уникальный идентификатор DOM-элемента, в котором будет отображена карта. В качестве дополнительного параметра конструктор должен принимать объект конфигурации, который позволит:

– включить (отключить) отображение координатной сетки;

– настроить отображение элементов управления картой, добавленных по умолчанию;

– включить (отключить) различные типы взаимодействий с картой.

2.5.15 Общие замечания и ограничения, накладываемые модули, предназначенные для конфигурации устройства

2.5.15.1 Ограничение, связанные с использованием портов при создании TCP-серверов

– для избежания конфликтов с системными службами, запрещено использовать порты с номерами меньше 3000;

– необходимо предотвращать попытки создания двух и более серверов, использующих один и тот же порт.

2.5.15.2 Ограничение, связанные с использованием Bluetooth

Одновременно для ввода и вывода Bluetooth могут использовать не более одной службы соответственно. Использование Bluetooth для ввода (вывода) данных одновременно двумя или более службами невозможен из-за аппаратных ограничений.

2.5.15.3 Ограничение, связанные с использованием LoRa

– в соответствии с таблицей 2.1, передача (приём) данных с помощью LoRa доступен только для устройств Reach RS;

– невозможно использовать LoRa для получения и отправки данных одновременно;

– аппаратное обеспечение Reach RS позволяет использовать LoRa на частотах от 862 до 1020 МГц, однако необходимо принять во внимание возможные дополнительные законодательные ограничения, накладываемые на разрешённый диапазон частот.

2.6 Сопоставление модулей приложения с представлениями пользовательского интерфейса

На основе наблюдений, сделанных в подразделе 1.5, было принято решение разнести многочисленные функции приложения по отдельным вкладкам. Подобное разделение панелей индикаторов (англ. *dashboard*), всевоз-

можных форм настроек и прочих элементов управления призвано сделать интерфейс приложения интуитивным, а процесс работы пользователя с устройством – комфортным.

Разделение интерфейса на вкладки, по сути, является отображением модульной структуры приложения, описанной в подразделах 2.4 и 2.5. Был сформирован следующий список вкладок, содержащих элементы управления, относящиеся только к конкретной части приложения:

– Статус (англ. *Status*);

– Изыскания (англ. *Survey*);

– Настройки RTK (англ. *RTK settings*);

– Входящие поправки (англ. *Correction input*);

– Выдача позиции (англ. *Position output*);

– Режим базы (англ. *Base mode*);

– Логирование (англ. *Logging*);

– Управление камерой (англ. *Camera control*);

– Wi-Fi;

– Bluetooth;

– Настройки (англ. *Settings*).

2.7 Выводы по разделу 2

– Проведён обзор платформы для разработки приложения. По результатам обзора используемых устройств были выявлены основные технические особенности, которые необходимо было учесть при разработке клиентского приложения.

– На основании ключевых требований к итоговому приложению и с учётом особенностей программного обеспечения используемых устройств, была создана архитектура проекта, а также были выбраны средства для его реализации.

– Была произведена детализация архитектуры клиентской части приложения, в результате чего были выделены её ключевые модули.

– Были сформулированы требования к основным модулям веб-приложения.

– Был сформулированы основные идеи, лежащие в основе построения пользовательского интерфейса разрабатываемого приложения.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Подготовка окружения для разработки

Основой разрабатываемого приложения является JavaScript-фреймворк Vue.js. Данный фреймворк позволяет начать работу над проектом максимально просто – достаточно добавить на веб-страницу JavaScript-файл с кодом Vue.js, используя HTML-тег *script*.

Однако, для обеспечения удобства разработки и возможности использования новейших возможностей языка JavaScript было решено использовать пакетный менеджер NPM [9] вместе с компилятором Babel [10] и утилитой Webpack [11], предназначенной для сборки веб-приложений.

3.1.1 Пакетный менеджер NPM

NPM (аббр. *Node Package Manager*) – менеджер пакетов, входящий в состав программной платформы Node.js [12]. NPM существенно упрощает установку компонентов, необходимых для работы или сборки приложения.

При работе с данным пакетным менеджером, особый интерес представляет файл **package.json**. Данный файл содержит информацию о разрабатываемом приложении: название, версия, описание и т.д (см. листинг 3.1). Но наиболее важным содержимым файла package.json являются зависимости – список имён и версий пакетов, требующихся для работы приложения.

3.1.2 Babel

Babel – транспилер (англ. *transpiler*), транслирующий код JavaScript стандартов ES2015 и новее [13] в код более ранних версий JavaScript.

Применение данного инструмента при разработке проекта позволяет использовать возможности JavaScript, представленные в новейших стандартах языка, не теряя при этом совместимость приложения со старыми версиями веб-браузеров.

Листинг 3.1 – Пример содержимого файла package.json

```
{
  "name": "ReachView",
  "version": "1.0.0",
  "description": "Reach/Reach RS controller app",
  "author": "Emlid Ltd <info@emlid.com>",
  "private": true,
  "scripts": {
    "dev": "node build/dev-server.js",
    "build": "node build/build.js",
    "unit": "karma start test/unit/karma.conf.js --single-run",
    ...
  },
  "dependencies": {
    ...
    "babel-core": "^6.22.1",
    "babel-eslint": "8.1.0",
    "d3": "^4.10.2",
    "eslint": "4.11.0",
    "openlayers": "^4.6.4",
    "socket.io-client": "1.4.5",
    "vue": "2.5.7",
    "vue-i18n": "^7.0.5",
    "vue-loader": "12.2.2",
    "vuex": "^3.0.1",
    "webpack": "2.6.1",
    ...
  },
  "devDependencies": {
    ...
    "chai": "^4.1.2",
    "express": "^4.14.1",
    "karma": "^1.4.1",
    "mocha": "^3.2.0",
    "sinon": "^2.1.0",
    ...
  }
}
```

3.1.3 Webpack

Webpack – система сборки для JavaScript-приложений, предназначенная, в первую очередь, для генерирования статических ресурсов на основе JavaScript-модулей и их зависимостей.

Одним из основных преимуществ Webpack является его способность работать с практически любыми типами ресурсов. Данная возможность обеспечивается дополнительно устанавливаемых *загрузчиков* (англ. *loaders*), которые, к примеру, позволяют:

– производить компиляцию JavaScript-файлов с помощью Babel;

– осуществлять статический анализ кода с помощью ESLint [14];

– минифицировать и обфусцировать код приложения;

– обрабатывать файлы с расширением «.vue» (однофайловые компоненты Vue.js);

– производить трансляцию стилей, описанных на языке SCSS [15], в CSS.

Стоит также отметить, что с помощью Webpack можно существенно облегчить разработку веб-приложения. Благодаря специальным расширениям становится возможно запустить локальный HTTP-сервер, позволяющий просматривать и отлаживать разрабатываемое приложение в браузере компьютера, на котором ведётся разработка.

3.2 Структура проекта

Исходный код клиентской части приложения является частью Python-проекта, написанного с использованием веб-фреймворка Flask. Подобная структура проекта позволяет держать весь код приложения в одном репозитории и облегчает сборку Python-приложения, предназначенного для установки на устройства Reach и Reach RS.

Код клиентского веб-приложения находится в директории *static-src*. Структура данной директории подробно описана на рисунке 3.1.

Рисунок 3.1 – Структура проекта

3.3 Глобальное хранилище данных приложения. Однонаправленный поток данных

Vuex – библиотека-расширение для Vue.js приложений, позволяющая создавать глобальное хранилище данных, доступное для всех модулей и Vue-компонентов, входящих в состав проекта. Подобное хранилище необходимо не только для обмена данными между компонентами – основной его задачей является управление состоянием представлений приложения.

Идеи, лежащие в основе данной библиотеки, унаследованы от архитектуры Flux [16]. Vuex предоставляет шаблонный подход к управлению состояниями компонентов приложения, основанный на *однонаправленном потоке данных*.

3.3.1 Однонаправленный поток данных

Взаимный обмен данными и событиями между моделями и представлениями при наличии большого числа компонентов является потенциальным источником ошибок. Асинхронные изменения и побочные эффекты могут существенно усложнить разработку и отладку, а также нарушить работу приложения.

Однонаправленный поток данных в простейшем виде представлен на рисунке 3.2.

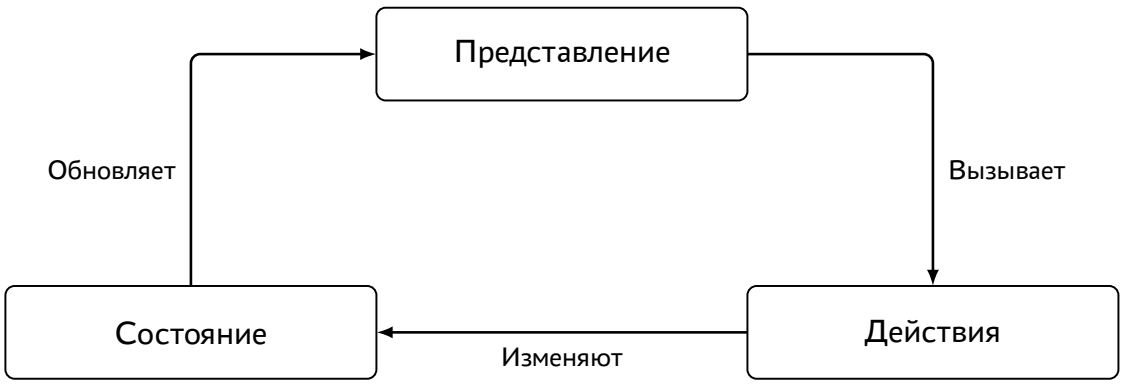


Рисунок 3.2 – Однонаправленный поток данных

Подход, описанный на рисунке 3.2, прекрасно подходит для управления представлением одного компонента. Однако, при появлении нескольких компонентов, зависящих от одного и того же состояния, структура приложения может существенно усложниться.

Vuex решает проблему, описанную выше, вводя в структуру приложения глобальное хранилище данных, являющееся состоянием-«одиночкой» (англ. *singleton*), которое обновляет все необходимые представления с помощью реактивных обновлений.

Для избежания неконтролируемых изменений состояния в Vuex введено следующее ограничение: изменить состояние можно только с помощью синхронных транзакций, называемых *мутациями*.

Асинхронные операции, результаты выполнения которых изменяют состояние, в Vuex получили название *«действия»*. Внутри функций-обработчиков действий становится возможно, к примеру, совершить асинхронный запрос к серверу, а при получении результата сделать одну или более синхронных мутаций состояния.

Схема организации однонаправленного потока данных при использовании Vuex изображена на рисунке 3.3.

3.3.2 Модули хранилища данных

Глобальные данные о состоянии приложения по умолчанию хранятся в одном и том же объекте. Рост числа модулей приложения неизбежно приводит к тому, что структура данного объекта существенно усложняется, а именование мутаций и действий может вызывать затруднения из-за требования к уникальности имён сущностей Vuex.

Для решения описанных выше проблем Vuex предоставляет инструменты для разделения хранилища на *модули*. Модули представляют из себя самостоятельные части глобального состояния, каждая из которых может содержать объект с данными, мутации и т.д. При использовании данного подхо-

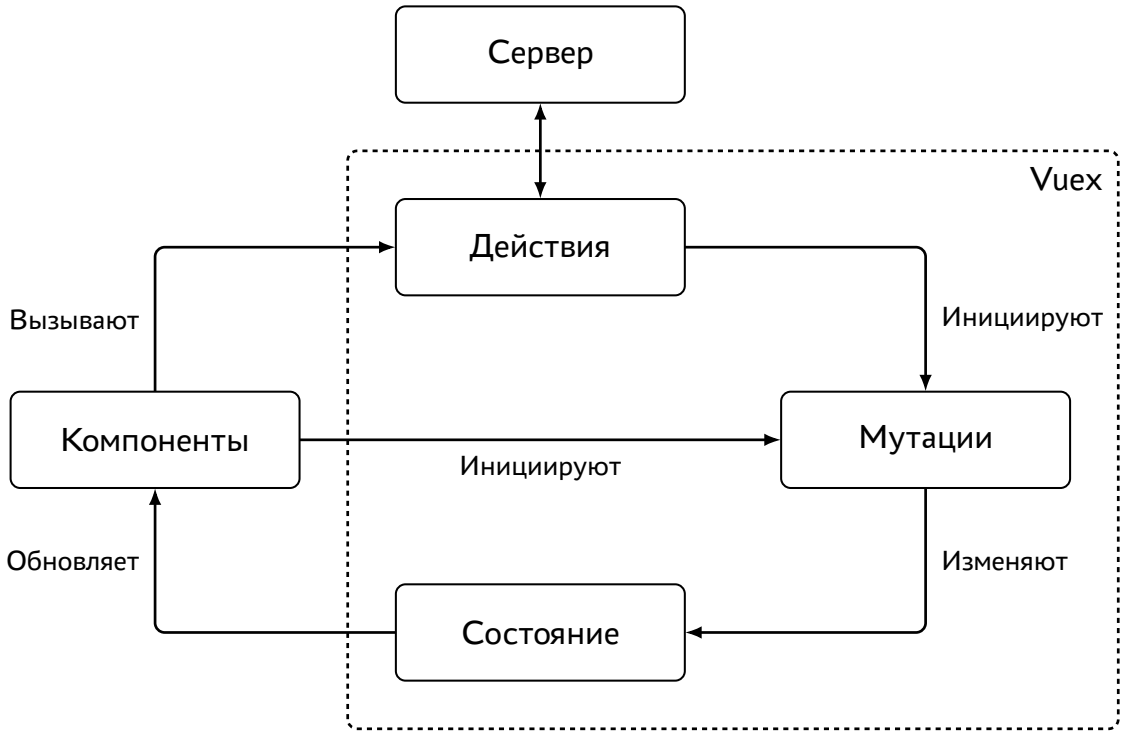


Рисунок 3.3 – Однонаправленный поток данных при использовании Vuex

да возможные проблемы с именованием решаются благодаря тому, что каждый модуль Vuex может иметь собственное пространство имён.

Хранилище Vuex, используемое в разрабатываемом приложении, было разделено на модули, соответствующие важнейшим подсистемам программного обеспечения устройств Reach и Reach RS:

– информация о текущей версии приложения, а также об устройстве и его конфигурация (модули *device* и *settings*);

– позиционирование и режим RTK (модуль *status*);

– входящие и исходящие потоки данных (модуль *streams*);

– беспроводные соединения (модуль *wireless*);

– изыскания (модуль *survey*);

– состояние компонентов графического интерфейса (модуль *ui*).

Схема модулей глобального хранилища изображена на рисунке 3.4.

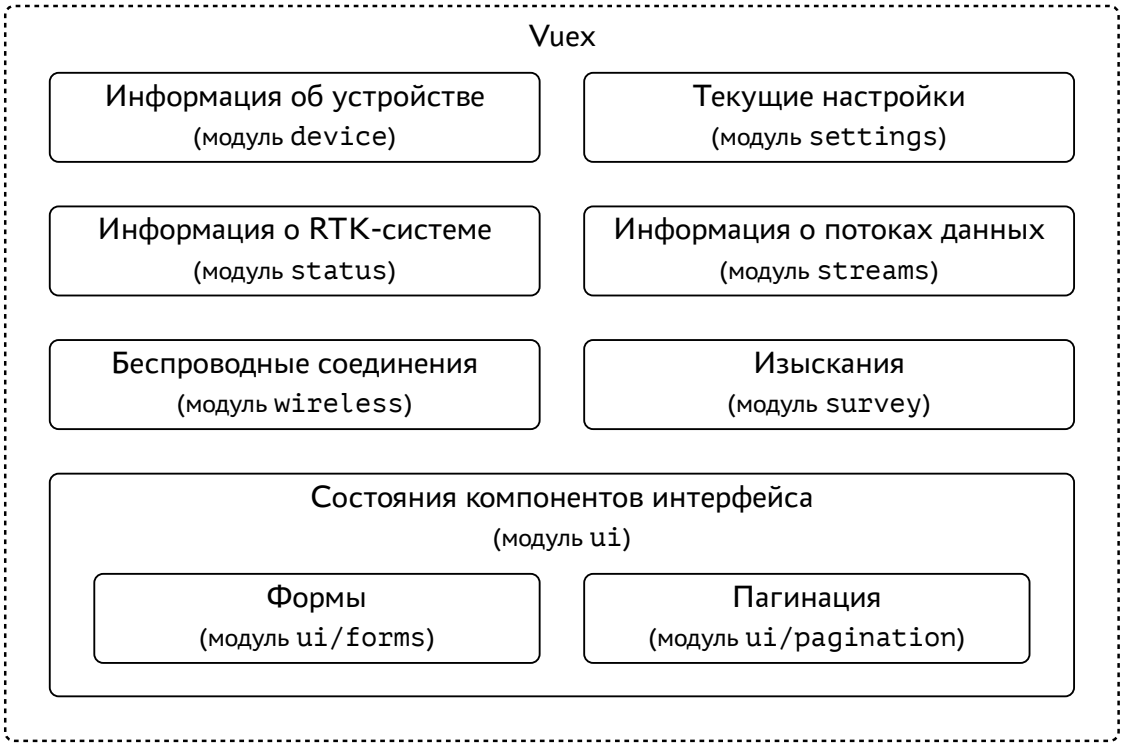


Рисунок 3.4 – Модульная структура хранилища Vuex

3.4 Модули и компоненты приложения

Далее описана структура и организация модулей и компонентов разрабатываемого веб-приложения. Первые два пункта посвящены модулям общего назначения и вспомогательным компонентам, которые необходимы для создания моделей представления, описанных в подразделе 2.5. Сами же модели представления описаны в третьем пункте текущего подраздела.

3.4.1 Модули общего назначения

К модулям общего назначения относятся модуль работы с картами и модуль обработки событий. Рассмотрим каждый из их подробнее.

3.4.1.1 Модуль работы с картами

Модуль работы с картами представляет из себя обёртку (англ. *wrapper*) над частью библиотеки OpenLayers, используемой в разрабатываемом приложении для отображения карт и различной информации на них. Код модуля содержится в файле *olWrapper.js*.

Данный модуль предназначен для повышения удобства создания однотипных карт – для добавления карты на страницу достаточно импортировать модуль в код приложения и вызвать соответствующие функции.

Импорт необходимых модулей OpenLayers, единожды осуществляемый в модуле работы с картами (см. листинг 3.2), исключает повторение существенного количества строк кода.

Листинг 3.2 – olWrapper.js: импорт необходимых модулей OpenLayers

```
import OLControlFullscreen from 'ol/control/fullscreen';
import OLControlScaleLine from 'ol/control/scaleline';
import OLFeature from 'ol/feature';
import OLGeomLineString from 'ol/geom/linestring';
import OLGeomPoint from 'ol/geom/point';
import OLGraticule from 'ol/graticule';
import OLMap from 'ol/map';
import OLStyle from 'ol/style/style';
import OLStyleCircle from 'ol/style/circle';
import OLStyleFill from 'ol/style/fill';
import OLStyleStroke from 'ol/style/stroke';
import OLTile from 'ol/layer/tile';
import OLVectorLayer from 'ol/layer/vector';
import OLVectorSource from 'ol/source/vector';
import OLView from 'ol/view';
import OSMSource from 'ol/source/osm';
import olControl from 'ol/control';
import olProj from 'ol/proj';
...
```

3.4.1.2 Модуль обработки событий

Модуль обработки событий (*eventsModule.js*) выполняет ряд ключевых задач, необходимых для работы приложения:

- инициализация WebSocket-подключения;
- обработка разрыва (восстановления) соединения с сервером;
- оповещение пользователя о различных событиях.
- инициализация модулей Vuex путём вызова их действий.

В листинге 3.3 представлено содержимое файла eventsModule.js (полный листинг файла см. в приложении А). По умолчанию модуль экспортирует функцию, выполнение которой зарегистрирует несколько слушателей событий Socket.IO и вызовет действие *«activate»* (*activate* с англ. – «активировать») в четырёх модулях Vuex.

Действия с названием «activate» присутствуют во всех модулях Vuex, кроме модулей, отвечающих за хранение состояния компонентов интерфейса (см. рис. 3.4). «Активация» модуля Vuex подразумевает под собой:

– установку значений по умолчанию в объекте состояния (при необходимости);

– регистрацию слушателей определённых широковещательных сообщений сервера;

– отправку на сервер запросов на получение данных;

– создание таймеров для периодических опросов сервера.

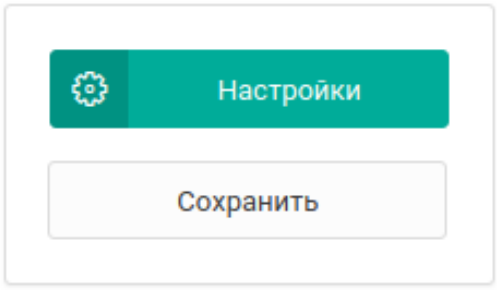
Листинг 3.3 – Модуль обработки событий

```
1  import io from 'socket.io-client';
2  import store from '@/store/index';
3
4  export const socket = process.env.NODE_ENV === 'production' ? io() : io(IP);
5
6  export default () => {
7    let lostConnectionNoty;
8
9    socket.on('connect', () => { ... });
10   socket.on('reconnect', () => { ... });
11   socket.on('disconnect', () => { ... });
12   socket.on('reachview upgrade version', msg => { ... });
13
14   store.dispatch('device/activate');
15   store.dispatch('settings/activate');
16   store.dispatch('status/activate');
17   store.dispatch('streams/activate');
18 };
```

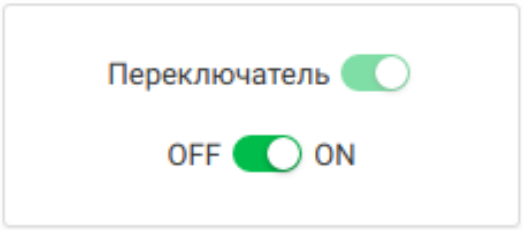
Также модуль обработки событий экспортирует объект *socket*, импортируя который, любой компонент приложения сможет осуществлять общение с сервером через WebSocket-события.

3.4.2 Вспомогательные компоненты

Вспомогательные компоненты представляют из себя набор блоков и элементов интерфейса, необходимых для создания всевозможных представлений приложения. Данные компоненты включают в себя кнопки, переключатели, панели, индикаторы выполнения и т.д. (см. рис. 3.5).



(а)



(б)

Рисунок 3.5 – Пример вспомогательных компонентов:
(а) - кнопки, (б) - переключатели

Каждый вспомогательный элемент является Vue-компонентом, что даёт возможность управлять внешним видом, содержимым и поведением каждого из них. К примеру, компонент «переключатель» (англ. *toggle*) кроме обязательного параметра, являющегося булевой переменной, также позволяет настроить свой текст, цвет и доступность для изменения состояния (см. листинг 3.4).

Листинг 3.4 – Настраиваемые свойства компонента «переключатель»

```
<template>
  ...
</template>

<script>
  export default {
    props: {
      // Текст
      title: {
        type: String,
        default: null
      },
      // Значение
      value: {
        type: Boolean,
        required: true
      },
      // Цвет
      color: {
        type: String,
        default: 'success',
        validator(color) { ... }
      },
      // Доступность для изменения
      disabled: {
        type: Boolean,
        default: false
      }
    },
    computed: { ... },
    methods: { ... }
  };
</script>

<style scoped lang='scss'>
  ...
</style>
```

3.4.3 Модели представления и представления

3.4.3.1 Статус

«Статус» является представлением, отображаемым по умолчанию. Основная задача данного представления – вывод информации об RTK-системе, сигналах спутников и позиции приёмника.

«Статус» зависит от данных, содержащихся в модуле status хранилища Vuex. Модуль status содержит всю необходимую для вывода информацию, обновляемую в режиме реального времени благодаря реактивности Vuex и зарегистрированным слушателям широковещательных сообщений сервера.

На рисунке 3.6 изображена страница «Статус», отображающая состояние ровера, получающего поправки с базы. Набор элементов страницы и их отображение меняется, в зависимости от наличия (отсутствия) тех или иных данных, а также от режима работы RTKLIB. Так, например, столбчатая диаграмма значений соотношений сигнал/шум для принимаемых со спутников сигналов примет вид, изображённый на рисунке 3.7, при отсутствии данных о спутниках, видимых базе. Другим примером может служить скрытие информации о позиции базы и RTK-параметрах при переводе RTKLIB в режим Single (см. пункт 1.3.2).

3.4.3.2 Изыскания

«Изыскания» – раздел приложения, с помощью которого происходит основная часть работы с устройством. Используя данный интерфейс пользователь может производить геодезические изыскания – сбор точек на местности, с разделением их на проекты.

В соответствии с требованиями, перечисленными в пункте 2.5.2, был разработан модуль приложения, основанный на сущностях и прецедентах, описанных на рисунках 3.8 и 3.9 соответственно.

Модуль «Изыскания» состоит из множества отдельных представлений, переключаясь между которыми, пользователь осуществляет разнообразные

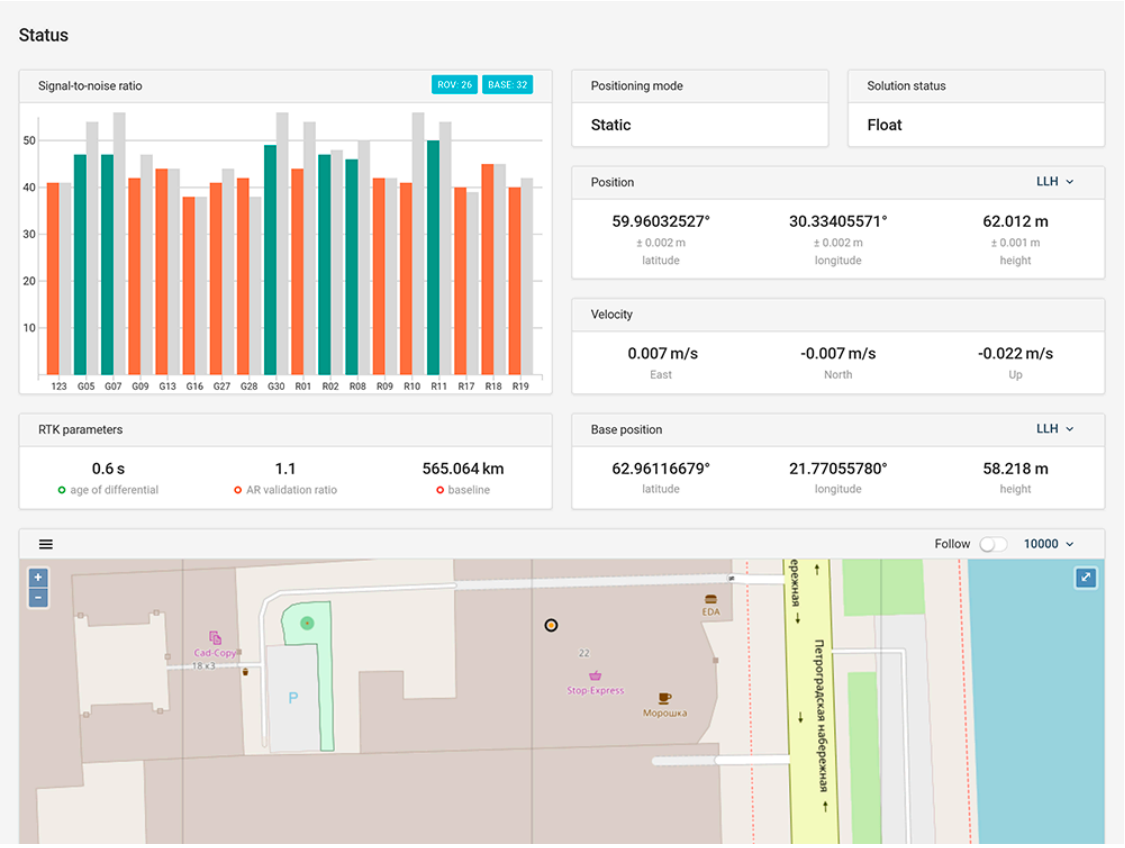


Рисунок 3.6 – Страница «Статус»

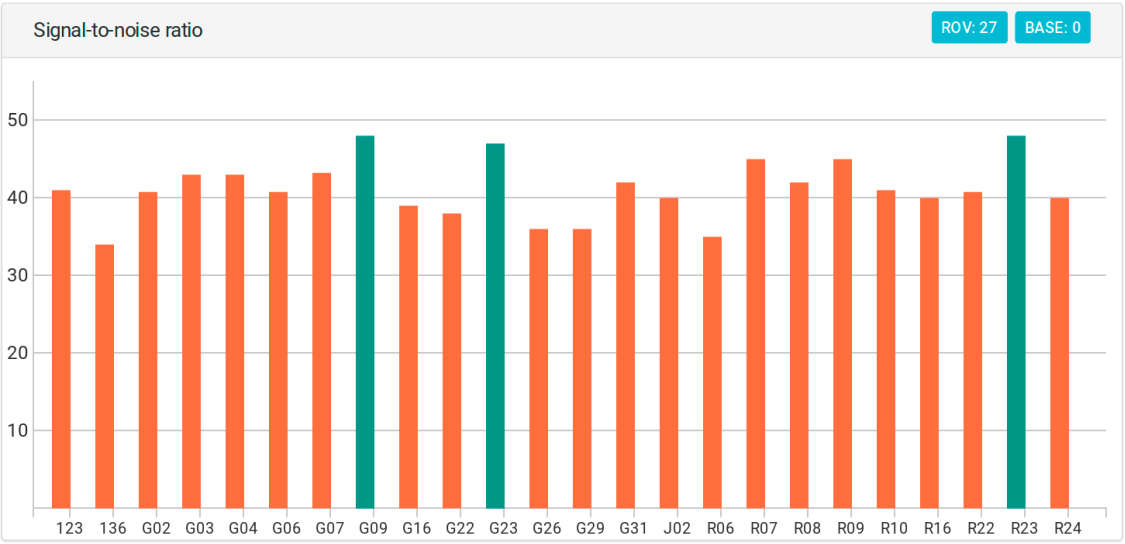


Рисунок 3.7 – Отсутствие данных о спутниках, видимых базе

действия с собранными данными. Одной из основных задач при разработке данного модуля было чёткое описание возможных состояний и условий перехода между ними. На рисунке 3.10 представлена диаграмма состояний модуля «Изыскания».

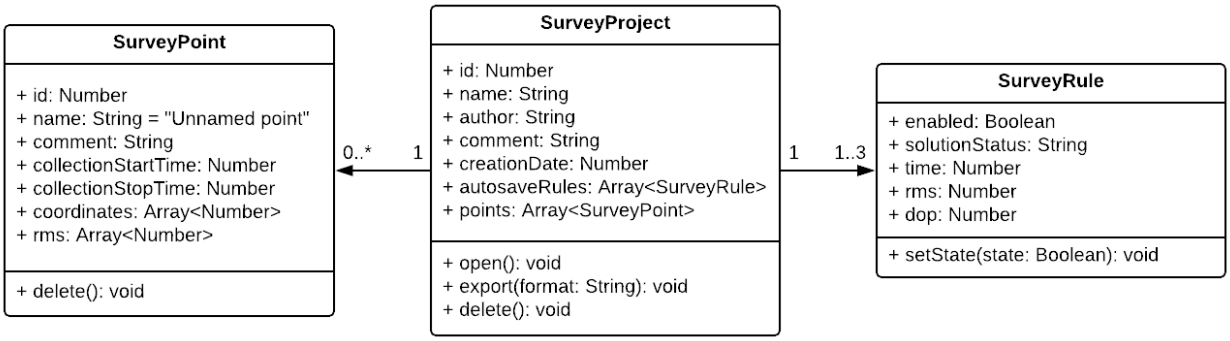


Рисунок 3.8 – Диаграмма классов модуля «Изыскания»

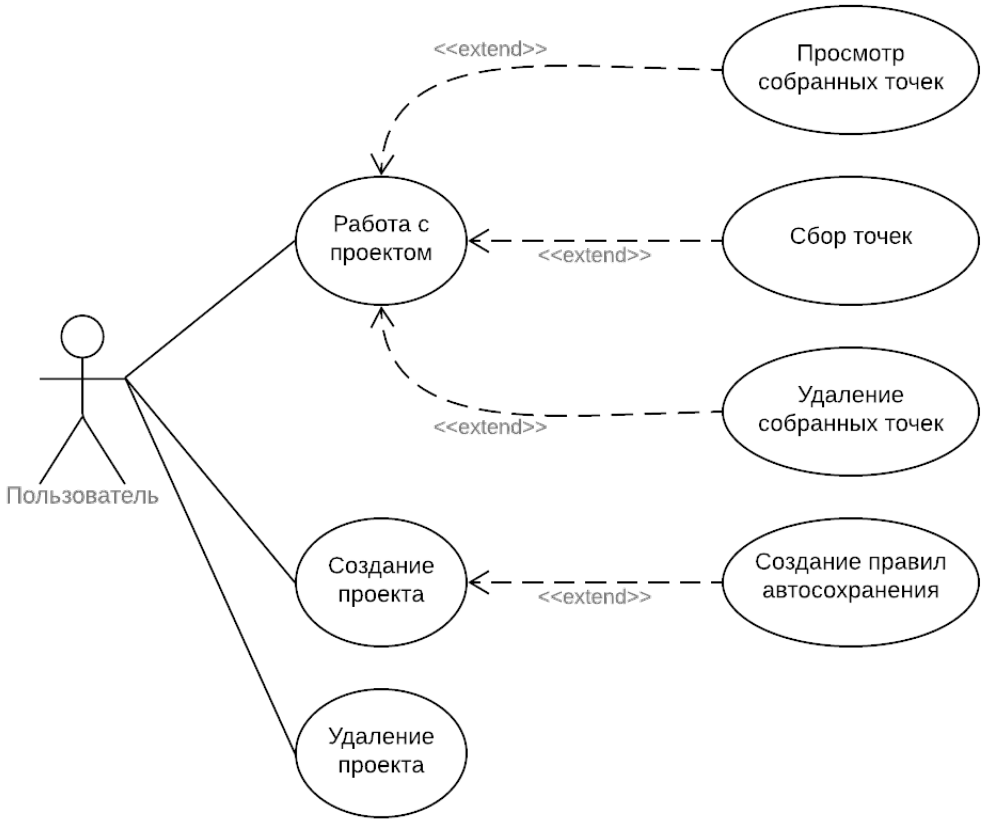


Рисунок 3.9 – Диаграмма прецедентов модуля «Изыскания»

3.4.3.3 Модули настройки параметров работы устройства

К модулям, предназначенным для настройки параметров устройства, относятся «Настройки RTK» и «Управление камерой». Представления данных модулей (рисунки 3.11 и 3.12) реализованы в виде форм настроек, отображающих требования описанные в пункте 2.5.2.

Алгоритм применения новых настроек с помощью форм вкладок «Настройки RTK» и «Управление камерой» представлен на рисунке 3.13. В ал-

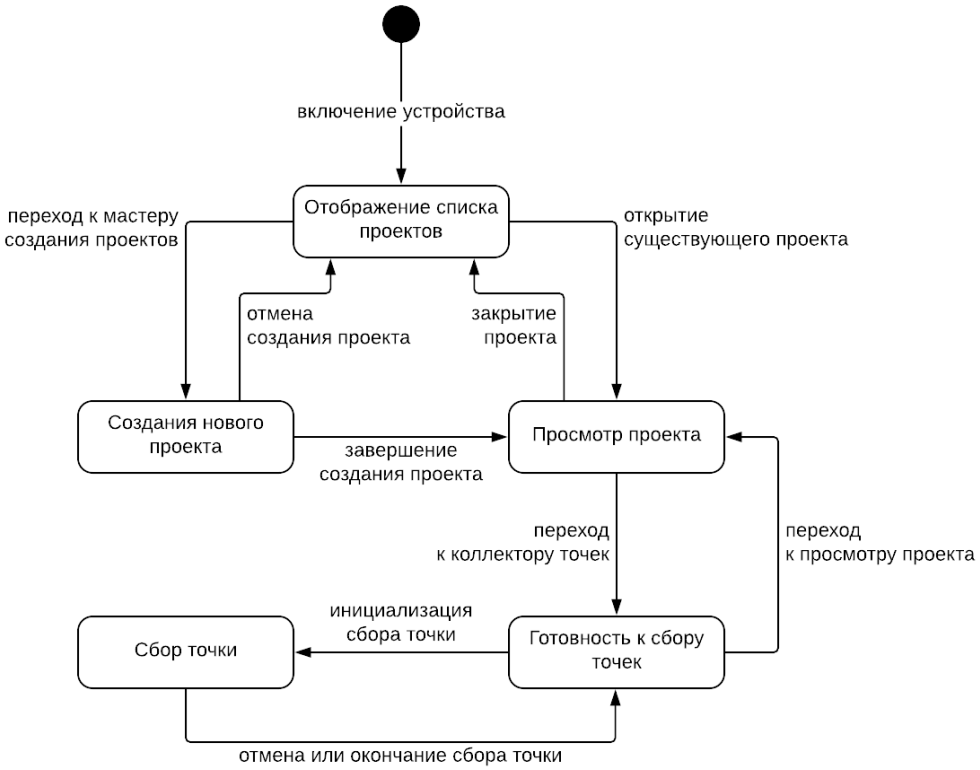


Рисунок 3.10 – Диаграмма состояний модуля «Изыскания»

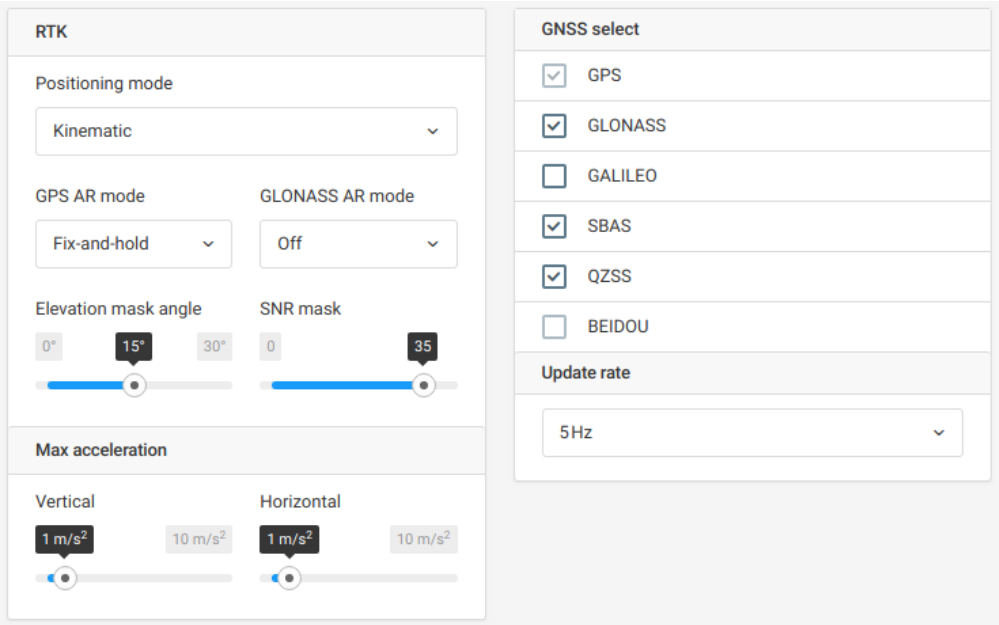


Рисунок 3.11 – Формы вкладки «Настройки RTK»

горитме отсутствует этап валидации применяемой конфигурации, т.к. формы рассматриваемых вкладок не содержат полей, подразумевающих ввод произвольных пользовательских значений.

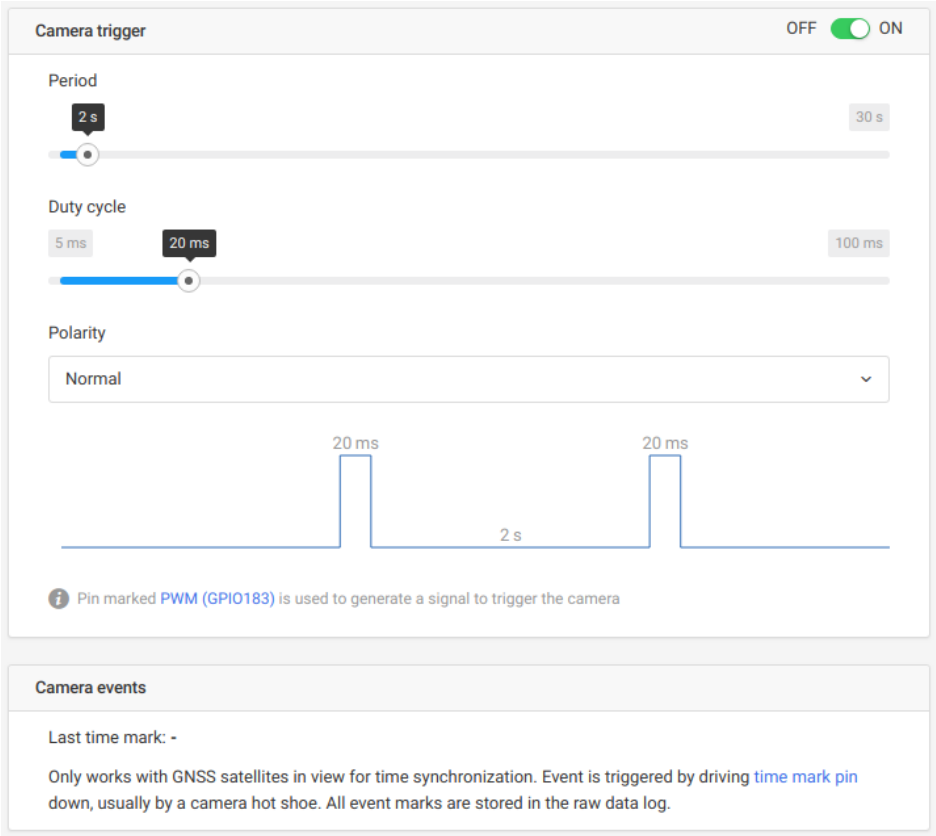


Рисунок 3.12 – Формы вкладки «Управление камерой»

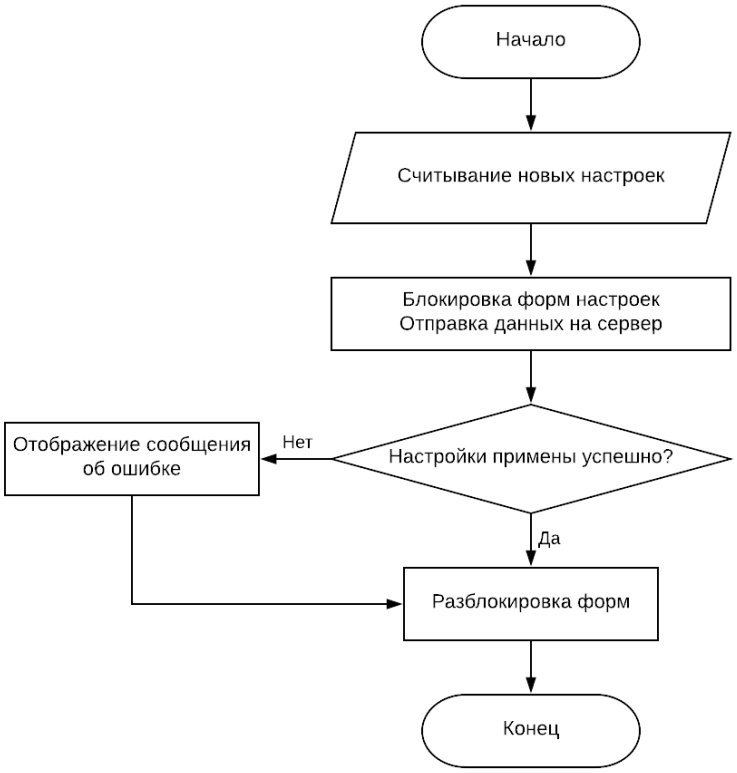


Рисунок 3.13 – Схема алгоритма применения настроек с помощью графической формы

3.4.3.4 Модули настройки входящих и исходящих потоков данных

За настройку входящих и исходящих потоков данных отвечают модули «Входящие поправки», «Выдача позиции» и «Режим базы» (рисунки 3.14, 3.15 и 3.16 соответственно). Как и в случае с компонентами, описанными в подпункте 3.4.3.3, представления рассматриваемых модулей содержат исключительно формы настроек.

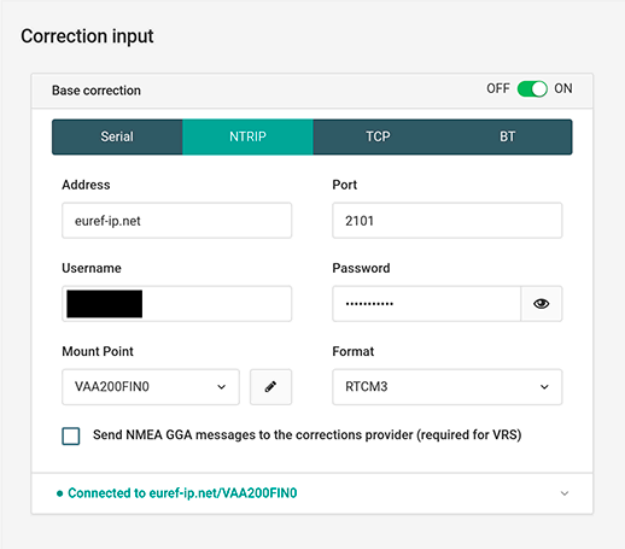


Рисунок 3.14 – Формы вкладки «Входящие поправки»

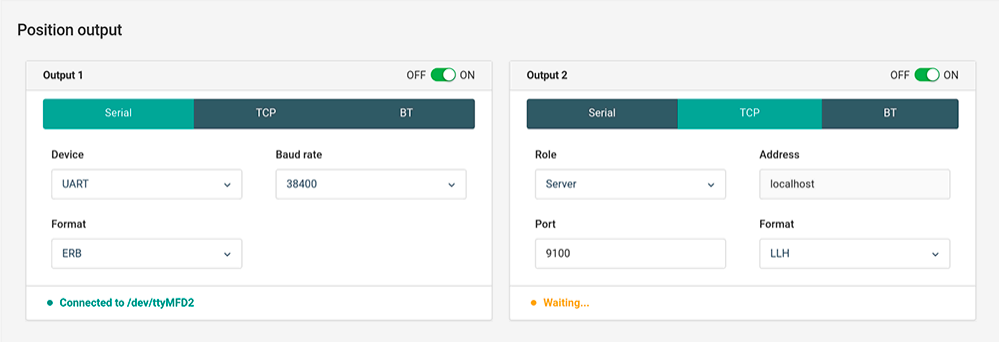


Рисунок 3.15 – Формы вкладки «Выдача позиции»

Отличительными особенностями форм представлений модулей настройки потоков данных являются:

– необходимость валидации значений, введённых пользователем;

– необходимость проверки наличия возможных конфликтов, описанных в пункте 2.5.15.

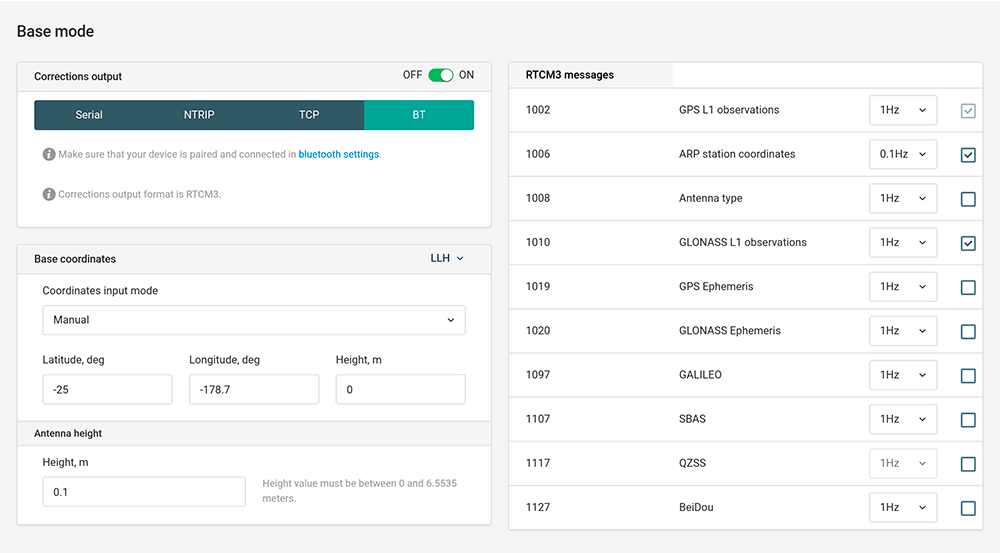


Рисунок 3.16 – Формы вкладки «Режим базы»

С учётом описанных особенностей, алгоритм применения настроек потоков данных принимает вид, представленный на рисунке 3.17.

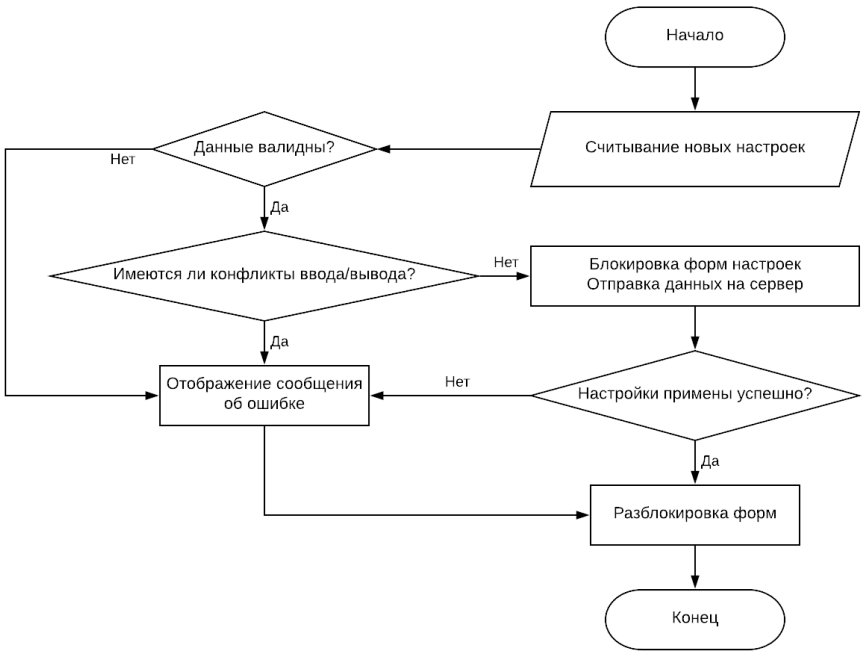


Рисунок 3.17 – Схема алгоритма применения настроек потоков данных с помощью графической формы

3.4.3.5 Модули настройки беспроводных соединений

Модули настройки беспроводных соединений – «Wi-Fi» и «Bluetooth» (рисунки 3.18 и 3.19) – отвечают за подключение приёмника к сетям Wi-Fi и за его сопряжение другими устройствами.

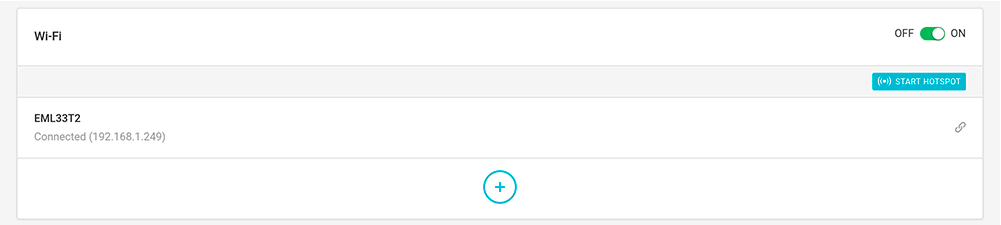


Рисунок 3.18 – Вкладка «Wi-Fi»

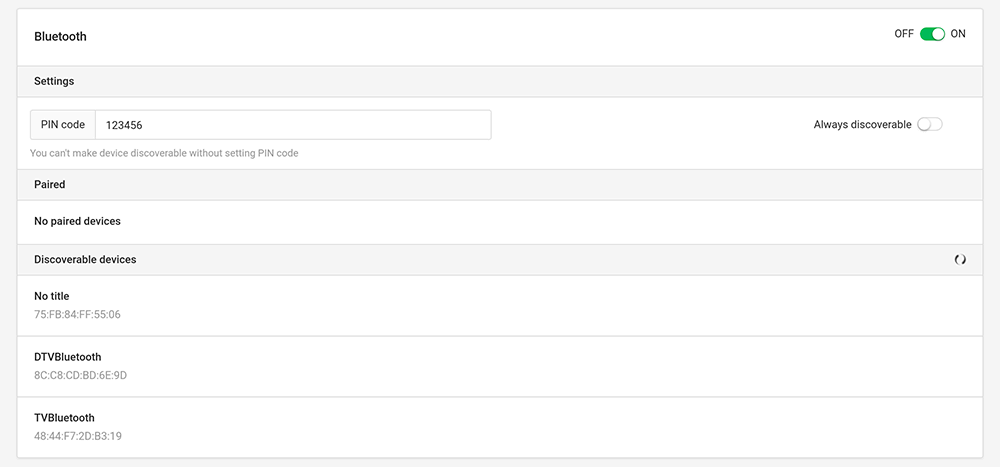


Рисунок 3.19 – Вкладка «Bluetooth»

Модули реализованы в соответствии с требованиями, описанными в пункте 2.5.2. Диаграммы вариантов использования вкладок «Wi-Fi» и «Bluetooth» представлены на рисунках 3.20 и 3.21 соответственно.

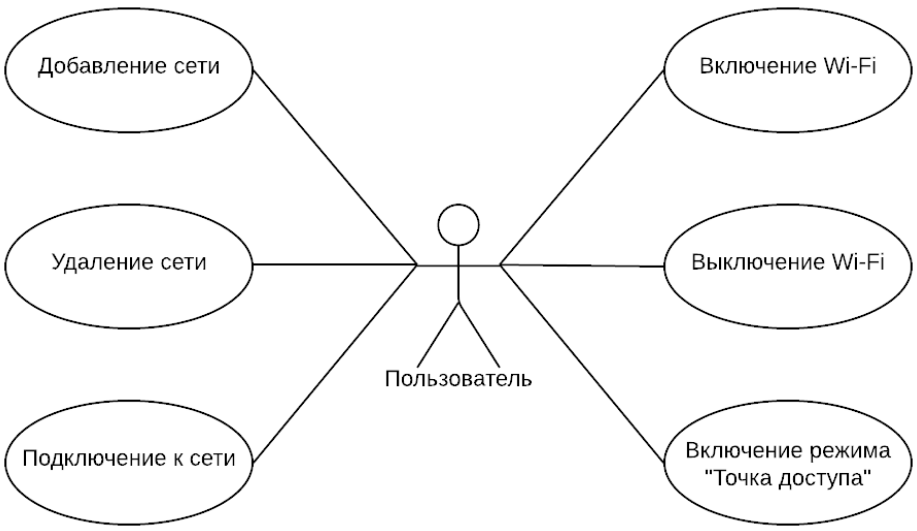


Рисунок 3.20 – Диаграмма прецедентов модуля «Wi-Fi»

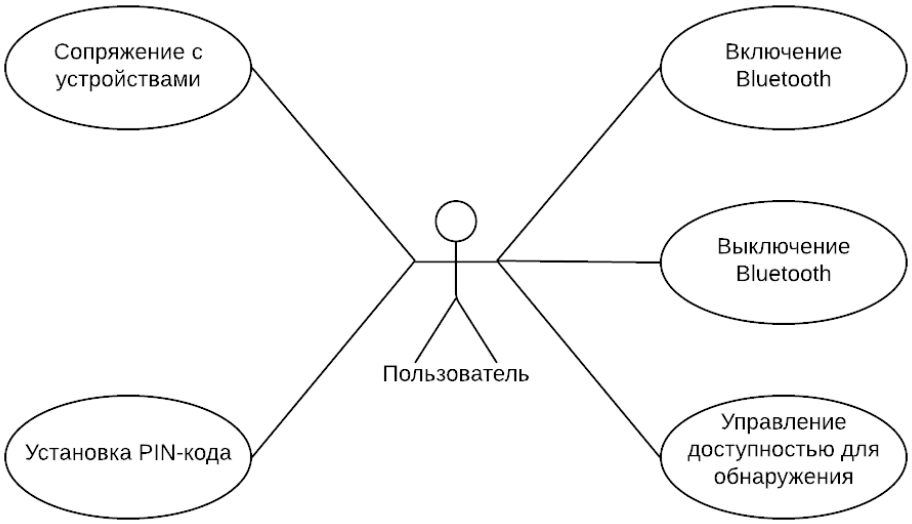


Рисунок 3.21 – Диаграмма прецедентов модуля «Bluetooth»

3.4.3.6 Логирование

«Логирование» – модуль приложения, с помощью которого пользователь получает доступ к управлению логами данных приёмника.

Представление модуля (см. рис. 3.22) состоит из трёх панелей для управления процессом записи логов (доступны логи данных трёх типов) и списка доступных логов, разделённых по дате начала записи.

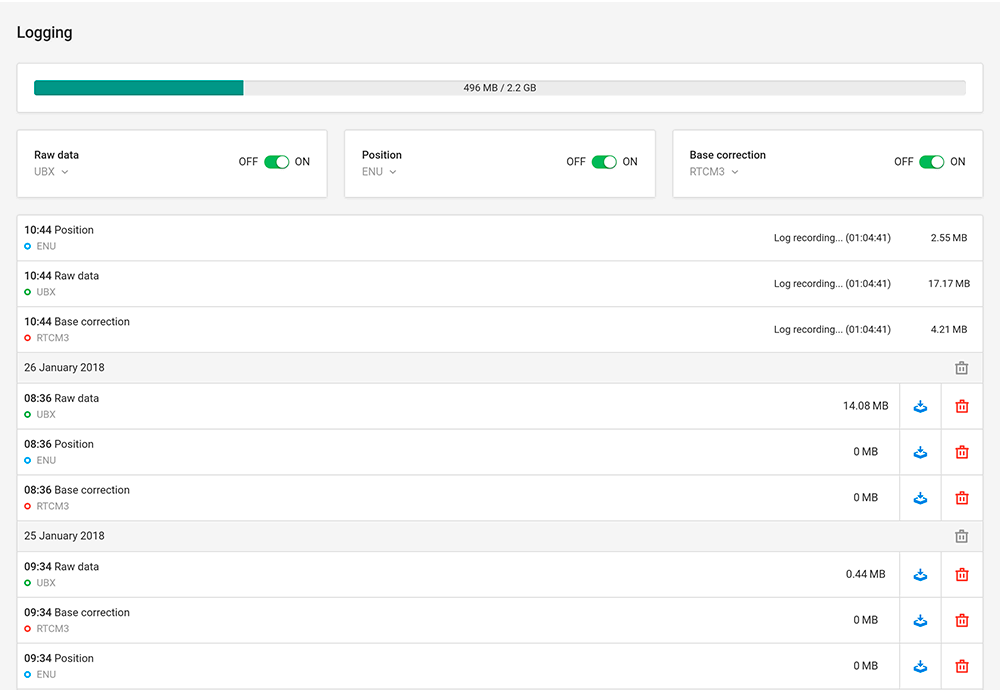


Рисунок 3.22 – Вкладка «Логирование»

Диаграмма вариантов использования вкладки «Логирование» представлена на рисунке 3.23.

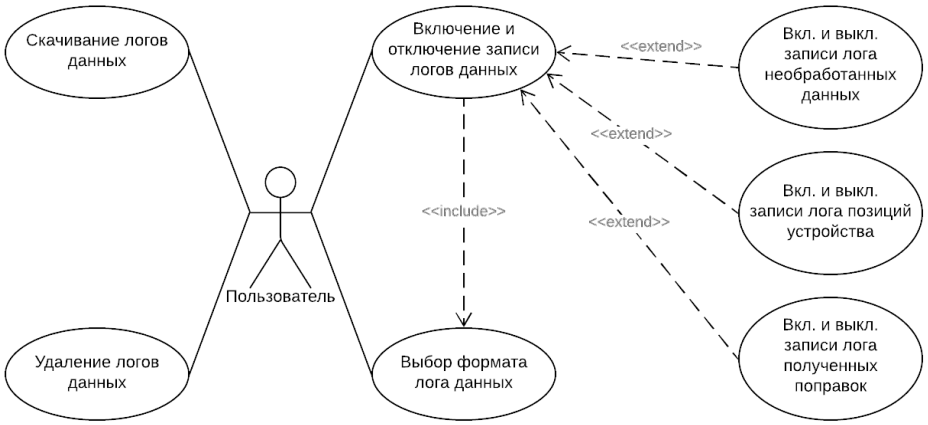


Рисунок 3.23 – Диаграмма прецедентов модуля «Логирование»

3.4.3.7 Общие настройки

Модуль предоставляет пользователю доступ к ряду общих настроек утилитарных функций приёмника (см. пункт 2.5.11). Содержимое вкладки «Общие настройки» представлено на рисунке 3.24.

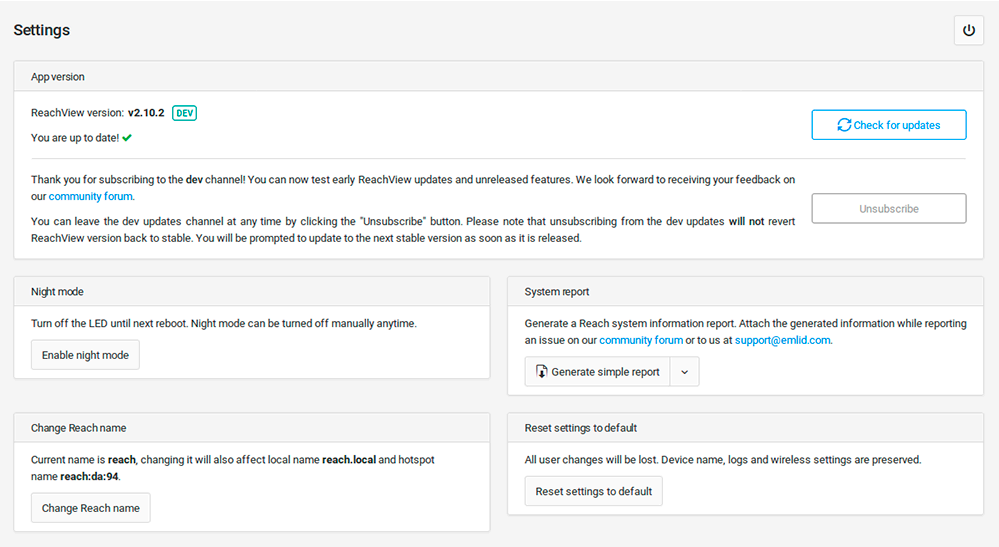


Рисунок 3.24 – Вкладка «Общие настройки»

Возможности, доступные во вкладке «Общие настройки», описаны с помощью диаграммы прецедентов, изображённой на рисунке 3.25.

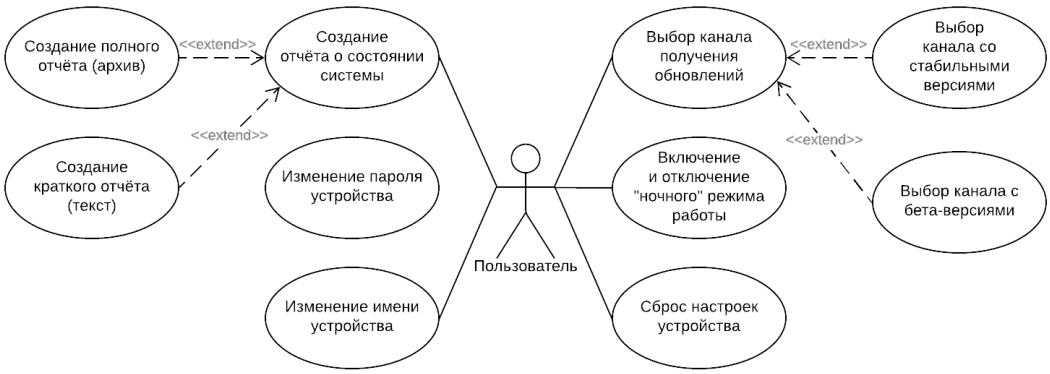


Рисунок 3.25 – Диаграмма прецедентов модуля «Общие настройки»

3.5 Тестирование приложение

3.5.1 Модульное тестирование

В ходе разработки исходный код приложения был покрыт модульными тестами. Для написания тестов были использован следующие инструменты:

– **Karma** [?] – утилита для запуска JavaScript-тестов;

– **Mocha** [?] – тестовый фреймворк для JavaScript-приложений;

– **Istanbul** [?] – утилита для анализа покрытия исходного кода JavaScript-приложений модульными тестами.

В листинге 3.5 представлен пример объявления тестовых случаев (англ. *test case*) и тестовых наборов (англ. *test suite*). Полная версия тестов листинга 3.5 представлена в приложении Б.

Листинг 3.5 – Настраиваемые свойства компонента «переключатель»

```
describe('Toggle.vue', () => {
  describe('Render behavior', () => {
    beforeEach(() => { ... });

    it('Should render correct title', (done) => { ... });

    it('Should render correct color', (done) => { ... });
  });

  describe('Click behavior', () => {
    beforeEach(() => { ... });
```

```
12
13      it('Should toggle active class on click', (done) => { ... });
14
15      it('Should not toggle active class on click when disabled', (done) => {
16        ...
17      });
18    });
19  });
```

На рисунке 3.26 продемонстрирован пример вывода Karma, работающей с Istanbul.

```
$ npm run test

...

Executed 148 of 154 (skipped 6) SUCCESS (6.701 secs / 5.888 secs)
TOTAL: 148 SUCCESS

=========================== Coverage summary ===========================
Statements : 58.99% ( 1109/1880 )
Branches   : 51.66% ( 606/1173 )
Functions  : 47.92% ( 115/240 )
Lines      : 59.58% ( 1088/1826 )
========================================================================
```

Рисунок 3.26 – Запуск тестов с помощью утилиты Karma

3.5.2 Функциональное тестирование

Функциональное тестирование приложения проводилось в ручном и автоматизированном режимах. Для автоматизации тестов были использованы язык программирования Python и библиотека Selenuim [?], позволяющая автоматизировать управление веб-браузерами (см. листинг 3.6).

Благодаря использованию тестового фреймворка pytest, предназначенного для написание тестов на Python, был автоматизирован процесс тестирования приложения в нескольких браузерах. Вызов графического сервера Xvfb [?] из кода Python-тестов позволил добавить возможность проверки приложения на различных разрешениях экрана.

Листинг 3.6 – Тестирование с помощью pytest и Selenium

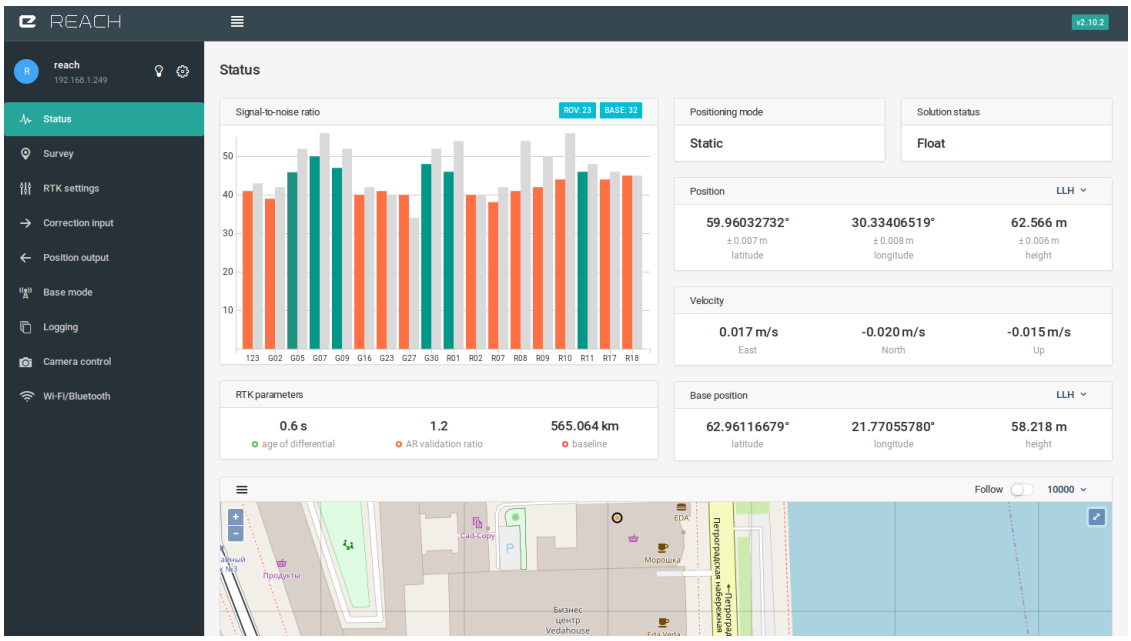
```
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

from tools.locators import Locators as locators
from tools.screenshot import screenshot


@screenshot
def test_check_battery_icon(browser, device):
    if device == 'Reach':
        WebDriverWait(browser, 0).until_not(
            EC.visibility_of_any_elements_located(locators.battery_icon)
        )
    elif device == 'ReachRS':
        WebDriverWait(browser, 0).until(
            EC.visibility_of_any_elements_located(locators.battery_icon)
        )
```

3.6 Выводы по разделу 3

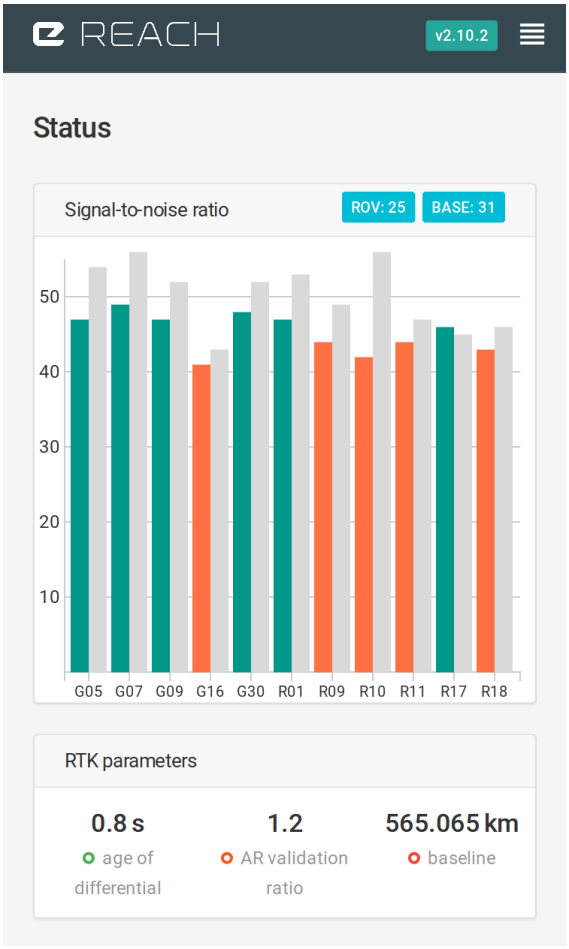
– Исходя из выбранных инструментов, было создано окружение для разработки, которое позволило использовать новейшие веб-технологии при создании приложения.

– Было создано кроссбраузерное, адаптивное веб-приложение для управления устройствами Reach и Reach RS, которое соответствует всем заявленным требованиям (см. рис. 3.27).

– Для разработанного приложения были созданы комплексы модульных и функциональных тестов, позволяющие автоматизировать проверку работы как отдельных компонентов, так и всего приложения в целом.



(а)



(б)

Рисунок 3.27 – Адаптация приложения к размеру дисплея устройства (на примере страницы «Статус»): (а) - экран ноутбука, (б) - экран смартфона

4 АПРОБАЦИЯ РЕЗУЛЬТАТОВ РАЗРАБОТКИ

4.1 Установка приложения на модули и приёмники

Разработанный пользовательский веб-интерфейс в составе Python-приложения был установлен на несколько устройств Reach и Reach RS. Автором работы совместно с разработчиками серверной части приложения было произведено ручное тестирование всех представлений и форм, доступных через веб-приложение.

Работа приложения была протестирована на различных устройствах, работающих под управлением ОС семейства Windows, macOS, Linux, а также на нескольких версиях мобильных систем Android и iOS.

Была проверена доступность приложения при подключении приёмников к Wi-Fi-сетям в качестве клиента, а также при их работе в режиме точки доступа.

4.2 Полевые испытания устройств

Разработанный веб-интерфейс был протестирован при работе устройств под открытым небом. Проверено поведение приложения при:

– отключении и подключении антенны во время работы устройства (для модулей Reach);

– потере и восстановлении соединения с базой;

– намеренном добавлении помех к сигналам, получаемым устройством со спутников.

При тестировании устройств под открытым небом особое внимание было уделено проверке раздела «Изыскания». Был произведён ручной сбор точек на местности, а также осуществлена проверка работы приложения при использовании правил автоматического сбора точек.

4.3 Бета-версии приложения

С ноября 2016 года было начато распространение бета-версии разработанного веб-приложения. Установка данного приложения подразумевала перепрошивку устройства.

Благодаря инициативной группе опытных геодезистов, являющихся пользователями устройств Reach, были собраны отчёты о найденных ошибках и пожелания по доработке приложения.

В марте 2017 года приложение стало основной рабочей версией веб-интерфейса для Reach и Reach RS.

4.4 Выводы по разделу 4

– По окончании процесса разработки все функции веб-приложения были протестированы автором работы и сотрудниками компании Emlid.

– Работа приложения была протестирована в веб-браузерах нескольких устройств, работающих под управлением различных операционных систем.

– Были проведены испытания приложения при работе с устройствами под открытым небом.

– Бета-версии приложения были протестированы инициативной группой пользователей, имеющих опыт работы с профессиональным геодезическим оборудованием.

– Разработанное приложение было успешно внедрено в качестве основной рабочей версии веб-интерфейса устройств Reach и Reach RS.

ЗАКЛЮЧЕНИЕ

В рамках проведённой работы были получены следующие результаты:

– Проведён обзор областей применения высокоточного позиционирования. Были рассмотрены проблемы доступности профессионального геодезического оборудования и распространённости открытого программного комплекса высокоточного позиционирования RTKLIB.

– Проведён анализ применения веб-интерфейсов для взаимодействия с устройствами без органов управления.

– Сформулированы требования к созданию веб-приложения для взаимодействия с устройствами Reach и Reach RS компании Emlid, работающими под под управление программного обеспечения, работающего на основе пакета RTKLIB.

– Создана детализированная архитектура приложения, на основе которой было разработано веб-приложение, полностью удовлетворяющее всем заявленным требованиям.

– Для приложения созданы комплексы модульных и функциональных текстов, позволяющие автоматизировать проверку работоспособности приложения.

– Проведена апробация результатов работы. Устройства, с установленным на них разработанным веб-приложением, были протестированы сотрудниками компании Emlid, а также инициативной группой опытных геодезистов, пользующихся устройствами Reach и Reach RS.

– На основе полученных в ходе апробации отчётах об ошибках и пожеланиях произведены доработки приложения. Разработанный продукт был успешно внедрён в качестве основной рабочей версии веб-интерфейса для устройств Reach и Reach RS.

Автор планирует продолжать работу над проектом и развивать его, добавляя новые функции и исправляя возможные ошибки, которые могут быть найдены во время эксплуатации приложения.

СПИСОК ТЕРМИНОВ

IRR: Internal Rate of Return

NPV: Net Present Value

PPP: Purchasing Power Parity

Библиографический список

1. Каталог GNSS-систем [Электронный ресурс] // Официальный интернет-магазин компании ГЕООПТИК. 2018. - Режим доступа: https://www.geooptic.ru/catalog/gnss-priemniki/ (дата обращения: 26.04.2018).

2. JAVAD GNSS [Электронный ресурс] // Официальный интернет-магазин компании JAVAD. 2018. - Режим доступа: https://www.javad.com/jgnss/products/receivers/ (дата обращения: 26.04.2018).

3. Takasu T. RTKLIB: An Open Source Program Package for GNSS Positioning [Электронный ресурс] // RTKLIB support information. 2015. - Режим доступа: http://www.rtklib.com/ (дата обращения: 01.02.2017).

4. Trimble GPS Tutorial [Электронный ресурс] // Официальный сайт компании Trimble. 2017. - Режим доступа: http://www.trimble.com/gps_tutorial/ (дата обращения: 01.02.2017).

5. Bavaro M. Michele's GNSS blog [Электронный ресурс] // Личный блог Мишеля Баваро. 2017. - Режим доступа: http://michelebavaro.blogspot.ru/ (дата обращения: 15.02.2017).

6. Emlid Ltd [Электронный ресурс] // Официальный сайт компании Emlid. 2017. - Режим доступа: https://emlid.com/ (дата обращения: 01.02.2017).

7. Emlid Ltd – Reach [Электронный ресурс] // Официальный сайт компании Emlid. Описание ГНСС модуля Reach. 2017. - Режим доступа: https://emlid.com/reach/ (дата обращения: 01.02.2017).

8. Emlid Ltd – Reach RS [Электронный ресурс] // Официальный сайт компании Emlid. Описание ГНСС приёмника Reach RS. 2017. - Режим доступа: https://emlid.com/reachrs/ (дата обращения: 01.02.2017).

9. NPM – the package manager for Node.js [Электронный ресурс] // Официальный сайт проекта NPM. 2018. - Режим доступа: https://www.npmjs.com/ (дата обращения: 01.02.2017).

10. Babel – The compiler for writing next generation JavaScript [Электронный ресурс] // Официальный сайт проекта Babel. 2018. - Режим доступа: https://babeljs.io/ (дата обращения: 01.02.2017).

11. Webpack module bundler [Электронный ресурс] // Официальный сайт проекта Webpack. 2018. - Режим доступа: https://webpack.js.org/ (дата обращения: 01.02.2017).

12. Node.js [Электронный ресурс] // Официальный сайт проекта Node.js. 2018. - Режим доступа: https://nodejs.org/ (дата обращения: 01.02.2017).

13. ECMAScript Language Specification [Электронный ресурс] // Спецификация ECMAScript. 2018. - Режим доступа: https://tc39.github.io/ecma262/ (дата обращения: 01.02.2017).

14. ESLint - Pluggable JavaScript linter [Электронный ресурс] // Официальный сайт проекта ESLint. 2018. - Режим доступа: https://eslint.org/ (дата обращения: 01.02.2017).

15. Sass: Syntactically Awesome Style Sheets [Электронный ресурс] // Официальный сайт проекта Sass. 2018. - Режим доступа: https://sass-lang.com/ (дата обращения: 01.02.2017).

16. Flux – Application Architecture for Building User Interfaces [Электронный ресурс] // Официальный сайт проекта Flux. 2015. - Режим доступа: http://facebook.github.io/flux/ (дата обращения: 01.02.2017).

Приложение А

Листинг А.1 – eventsModule.js: Модуль обработки событий

```
import io from 'socket.io-client';
import noty from 'noty';
import store from '@/store/index';

export const socket = process.env.NODE_ENV === 'production'
  ? io()
  : io(IP);

export default () => {
  let lostConnectionNoty;

  socket.on('connect', () => {
    socket.emit('browser connected');
  });

  socket.on('reconnect', () => {
    if (typeof lostConnectionNoty !== 'undefined') {
      lostConnectionNoty.close();
    }
  });

  socket.on('disconnect', () => {
    lostConnectionNoty = noty({
      width: 200,
      text: `Lost connection with Reach. Please check your network,
             then try refreshing the page`,
      type: 'error',
      dismissQueue: true,
      closeWith: false,
      timeout: false,
      theme: 'defaultTheme',
      layout: 'topRight',
      callback: {
        onClose: () => {
          noty({
            width: 200,
```

```
            text: 'Reach reconnected!',
            type: 'success',
            dismissQueue: true,
            closeWith: ['click'],
            timeout: 4000,
            theme: 'defaultTheme',
            layout: 'topRight'
          });
        }
      }
    });
  });

  socket.on('reachview upgrade version', msg => {
    if (msg['upgrade available']) {
      const availableVersion = msg['available version'].split('-')[0];

      noty({
        width: 200,
        text: `ReachView ${availableVersion} is available!`,
        type: 'success',
        dismissQueue: true,
        closeWith: ['click'],
        timeout: 4000,
        theme: 'defaultTheme',
        layout: 'topRight'
      })
    }
  });

  store.dispatch('device/activate');
  store.dispatch('settings/activate');
  store.dispatch('status/activate');
  store.dispatch('streams/activate');
};
```

Приложение Б

Листинг Б.1 – Тесты для компонента «переключатель»

```
import Toggle from '@/components/common/Toggle';
import { expect } from 'chai';
import i18n from '@/modules/localizationModule';
import { mount } from 'avoriaz';

describe('Toggle.vue', () => {
  let wrapper;

  describe('Render behavior', () => {
    beforeEach(() => {
      wrapper = mount(Toggle, {
        propsData: {
          title: 'default title',
          value: true,
          color: 'primary'
        },
        i18n
      });
    });

    it('Should render correct title', (done) => {
      const readLabelText = () => wrapper
        .first('.toggle__label').text().trim();

      expect(readLabelText()).to.equal('default title');
      wrapper.setProps({ title: 'new title' });
      expect(readLabelText()).to.equal('new title');

      done();
    });

    it('Should render correct color', (done) => {
      const checkClass = (color) =>
        wrapper.first('.toggle__handle').hasClass(`toggle__handle—${color}`);

      expect(checkClass('primary')).to.be.true;
```

```
37      wrapper.setProps({ color: 'danger' });
38      expect(checkClass('danger')).to.be.true;
39
40      done();
41    });
42  });
43
44  describe('Click behavior', () => {
45    const checkClass = () => wrapper.hasClass('toggle--checked');
46    const findSwitchElem = () => wrapper.first('.toggle__switch');
47
48    beforeEach(() => {
49      wrapper = mount({
50        data() {
51          return {
52            model: true,
53            disabled: false
54          };
55        },
56
57        components: { Toggle },
58        template: '<toggle v-model="model" :disabled="disabled"/>'
59      }, {
60        i18n
61      });
62    });
63
64    it('Should toggle active class on click', (done) => {
65      expect(checkClass()).to.be.true;
66      findSwitchElem().trigger('click');
67
68      wrapper.vm.$nextTick(() => {
69        expect(checkClass()).to.be.false;
70        done();
71      });
72    });
73
74    it('Should not toggle active class on click when disabled', (done) => {
75      expect(checkClass()).to.be.true;
76      wrapper.setData({ disabled: true });
```

```
77        findSwitchElem().trigger('click');
78
79        wrapper.vm.$nextTick(() => {
80          expect(checkClass()).to.be.true;
81          done();
82        });
83      });
84    });
85  });
```