Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Компьютерные системы и сети»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

Система мониторинга парковочных мест

Студент ___ИУ6(81)___	_____	___А.В. Кирьяненко__
(Группа)	(Подпись, дата)	(И.О.Фамилия)
Руководитель ВКР	_____	_____А.Ю. Попов____
	(Подпись, дата)	(И.О.Фамилия)
Консультант	_____	_____
	(Подпись, дата)	(И.О.Фамилия)
Нормоконтролер	_____	_____Ю.И. Бауман____
	(Подпись, дата)	(И.О.Фамилия)

2018 г.

АННОТАЦИЯ

Дипломный проект посвящен разработке системы мониторинга парковочных мест. При выполнении дипломного проекта, было проведено исследование IoT технологий в сфере транспортной инфраструктуры. В рамках исследования был проведен анализ существующих систем. На основании этого был спроектирован недорогой относительно конкурентов аппаратно-программный прототип системы мониторинга парковочных мест. Основное назначение системы заключается в определении наличия свободных парковочных мест и отображении этих данных пользователям. Данная система позволит пользователям (водителям) в режиме реального времени получать актуальную информацию о состоянии парковочных мест.

ABSTRACT

The diploma project is devoted to the development of a monitoring system for parking places. When carrying out the diploma project, was conducted research IoT technologies in the field of transport infrastructure. As part of the research, existing systems were analyzed. Based on this, the hardware-software prototype of a monitoring system for parking places was designed inexpensive with respect to competitors. The main purpose of the system is to determine the availability of parking spaces and display data for users. This system allows users (drivers) to receive up-to-date information about status of parking places in real time.

РЕФЕРАТ

Записка 70 с., 30 рис., 18 табл., 7 лист., 15 источников, 6 прил.

МКРОКОТРОЛЛЕР, МК-СИСТЕМА, ОПРОС ДАТЧИКОВ, ЭЛЕКТРОПРИБОР, БЕСПРОВОНАЯ СЕТЬ, МОНИТОРИНГ, ПАРКОВОЧНЫЕ МЕСТА, IOT, ИНТЕРНЕТ ВЕЩЕЙ, РАНСПОРТ, ТРАНСПОРТНАЯ ИНФРАСТРУКТУРА, MQTT

Разработан проект «Система мониторинга парковочных мест». Основное назначение системы заключается в определении наличия свободных парковочных мест и отображении этих данных пользователям. Данная система позволит пользователям (водителям) в режиме реального времени получать актуальную информацию о состоянии парковочных мест. Данная система состоит из: «МК-подсистемы регистрации свободных парковочных мест», вычислительного хаба и сервера.

«МК-подсистема регистрации свободных парковочных мест» фиксирует наличие свободных парковочных мест. К микроконтроллеру подключаются сенсоры, каждый из которых следит за своим парковочным местом. Через радиомодуль RFM95W информация о состоянии парковочных мест (свободно / занято) передается в вычислительный хаб, для последующей обработки. Реализована функция оплаты парковочного места, с отображением на дисплее и подтверждением оплаты с клавиатуры.

Основное назначение вычислительного хаба, состоит в осуществлении взаимодействия сервера, датчиков «МК-подсистема регистрации свободных парковочных мест» и других компонентов системы.

Сервер осуществляет сбор данных с датчиков «МК-подсистема регистрации свободных парковочных», управление всеми компонентами системы и вывод данных пользователям об актуальном состоянии парковочных мест

Материалы по дипломному проекту представлены в виде графической части, приложения с отлаженным программным кодом и расчетно-пояснительной записки.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БС – беспроводная сеть

БД – база данных

МК – микроконтроллер

МКПМ – модуль контроля парковочного места

МКПРСПМ – МК-подсистема регистрации свободных парковочных мест

ОС – операционная система

ПМ – парковочное место

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

СМПМ – система мониторинга парковочных мест

СУБД – система управления базами данных

ТС – транспортное средство

УГО – условно графическое обозначение.

EEPROM – энергонезависимая перезаписываемая память

IoT – internet of things

ITS – intelligent transportation system

LoRa – Long Range, метод модуляции в беспроводных сетях

MQTT – message queue telemetry transport

PPCM – parking place control module

RoR – Ruby on Rails

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дипломный проект «Система мониторинга парковочных мест» выполняется на основании учебного плана кафедры ИУ6 с исходными данными, заданными в техническом задании. Целью работы является проектирование системы мониторинга парковочных мест.

Формирование специализированных зон для стоянки автотранспорта началось почти одновременно с появлением первых автомобилей. Число машин стремительно растет и для решения возникнувшей проблемы ограниченности парковочных мест начали внедрять современные технологии.

Основным направлением развития являются «умные» датчики парковки. Такие датчики встраиваются в дорожное полотно на места парковок и отслеживают свободно ли машиноместо над ними, отдавая данные в общую систему. Используя сеть таких датчиков, формируется карта парковок, состояния которых рассылается пользователям на улицах с помощью специальных табло или мобильного приложения.

Такая система позволяет решить следующие задачи:

- сокращение очередей и времени поиска свободного места на парковке;
- организация оптимального движения транспортных средств и пешеходов на парковке;
- визуальное оповещение о количестве свободных мест в реальном времени с помощью информационных табло или на веб-портале;
- оптимизация работы парковочного пространства.

В ходе данной работы была спроектирована «Система мониторинга парковочных мест». Основное назначение системы заключается в определении наличия свободных парковочных мест и отображении этих данных пользователям. Данная система позволит пользователям (водителям) в режиме реального времени получать актуальную информацию о состоянии парковочных мест. Данная система состоит из: «МК-подсистемы регистрации свободных парковочных мест», вычислительного хаба и сервера.

1 Исследование IoT технологий в сфере транспортной инфраструктуры

1.1 Интернет вещей

1.1.1 Базовые принципы IoT

Интернет вещей базируется на 3-х базовых принципах:

1) повсеместно распространенную коммуникационную инфраструктуру,
2) глобальную идентификацию любого объекта,
3) возможность любого объекта отправлять и принимать данные через персональную сеть или через сеть Интернет.

Наиболее важными отличиями Интернета вещей от существующего интернета людей являются:

- фокус на вещах, а не на человеке;
- существенно большее число подключенных объектов;
- существенно меньшие размеры объектов и невысокие скорости;
- фокус на считывании данных, а не на коммуникациях;
- потребность формирования новой инфраструктуры и новых стандартов.

Официальное определение Интернета вещей приведено в Рекомендации МСЭ-TY.2060 [1], согласно которому IoT – глобальная инфраструктура информационного общества, обеспечивающая передовые услуги за счет организации связи между вещами (физическими или виртуальными) на основе существующих и развивающихся совместимых информационных и коммуникационных технологий. Под «вещами» (things) здесь понимается физический объект или же виртуальный объект (например, мультимедийный контент или прикладная программа), которые идентифицированы и связаны через сеть. МСЭ-Т использует понятие «устройство» (device) – часть оборудования с обязательным требованием по коммуникации и необязательным возможностью по сбору, обработке и хранению данных.

Любой узел сети интернет-вещей предоставляет свой сервис, оказывающий некую услугу поставки данных. Также узел может принимать команды от других узлов. Таким образом, все интернет-вещи могут взаимодействовать друг с другом и решать совместные вычислительные задачи. [2]

1.1.2 Архитектура IoT

Архитектура интернет вещей во многом схожа с четырехслойной архитектурой NGN. Она включает четыре функциональных уровня (Рисунок 1):

1) Приложения
2) Сервисный
3) Шлюзы и сети
4) Сеть датчиков

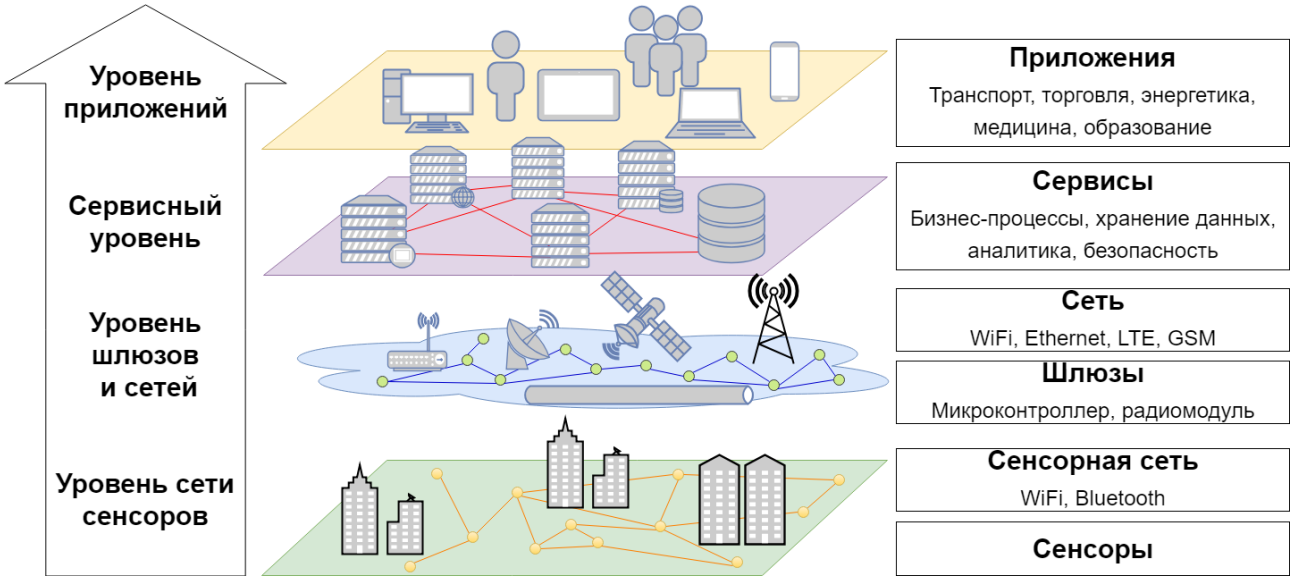

Рисунок 1 – Архитектура IoT

Далее снизу-вверх идёт описание каждого уровня IoT архитектуры [2].

1.1.2.1 Уровень сенсоров и сенсорных сетей.

Нижний уровень архитектуры IoT состоит из «умных» (smart) объектов, интегрированных с сенсорами. Сенсоры обеспечивают сбор и обработку данных в реальном времени. Существуют различные типы сенсоров для соответствующих задач, например, для измерения температуры, давления, скорости, местоположения и др. Как правило, сенсоры требуют соединения с агрегатором сенсоров (шлюзом), которые реализуются с использованием локальной сети (LAN, Local Area Network), таких как Ethernet и Wi-Fi или персональной сети (PAN, Personal Area Network), таких как Bluetooth и ультраширокополосной беспроводной связи на малых расстояниях (UWB, Ultra-Wide Band). Для сенсоров, не требующих подключения к агрегатору, их связь с серверами/приложениями может предоставляться с использованием глобальных беспроводных сетей WAN, таких как GSM, GPRS и LTE.

Сенсоры с низким энергопотреблением и низкой скоростью передачи данных, образуют беспроводные сенсорные сети (WSN, Wireless Sensor Network). WSN набирают все большую популярность, т. к. они могут содержать огромное количество сенсоров и охватывают большие площади.

1.1.2.2 Уровень шлюзов и сетей.

Огромный объем данных, создаваемых на нижнем уровне IoT множеством сенсоров, требует надежной и высокопроизводительной проводной или беспроводной сетевой инфраструктуры в качестве транспортной среды. Для реализации широкого спектра услуг и приложений в IoT необходимо обеспечить совместную работу множества сетей различных технологий и протоколов доступа. Эти сети должны обеспечивать требуемые значения качества передачи информации, и прежде всего по задержке пропускной способности и безопасности. Этот уровень состоит из сетевой инфраструктуры, реализуемой с помощью интеграции разнородных сетей в единую сетевую платформу. Абстрактный сетевой уровень в IoT позволяет через соответствующие шлюзы нескольким клиентам использовать ресурсы в одной сети независимо и совместно без ущерба для конфиденциальности, безопасности и производительности.

1.1.2.3 Сервисный уровень

Сервисный уровень содержит набор информационных услуг для автоматизации технологических и бизнес операции в IoT: поддержка операционной и бизнес деятельности различной аналитической обработки информации, хранения данных, обеспечения информационной безопасности, управления бизнес-правилами, управления бизнес-процессами и др.

1.1.2.4 Уровень приложений

На верхнем уровне архитектуры IoT существуют различные типы приложений для соответствующих сфер деятельности (транспорт, торговля, энергетика, медицина, образование и др.).

1.2 Протокол MQTT

MQTT (Message Queue Telemetry Transport) – упрощенный протокол сетевого уровня для обмена сообщениями между устройствами. Этот протокол работает поверх стека TCP/IP и разработан для преодоления проблем, связанных с подключением быстро растущего числа датчиков, микрокомпьютеров, приводов, телефонов, планшетов. В настоящее время MQTT является наиболее распространенным протоколом для организации IoT инфраструктуры. [3]

MQTT организован по принципу издатель/подписчик (publisher/subscriber): издатель (устройства типа publishers) является отправителем сообщения, которое публикуется в централизованном сервисе (брокере сообщений), а подписчик (устройства типа subscriber) получает сообщение из брокера. Для использования брокера MQTT необходимо пройти процедуру подписки на определенные темы публикуемых сообщений (Рисунок 2).

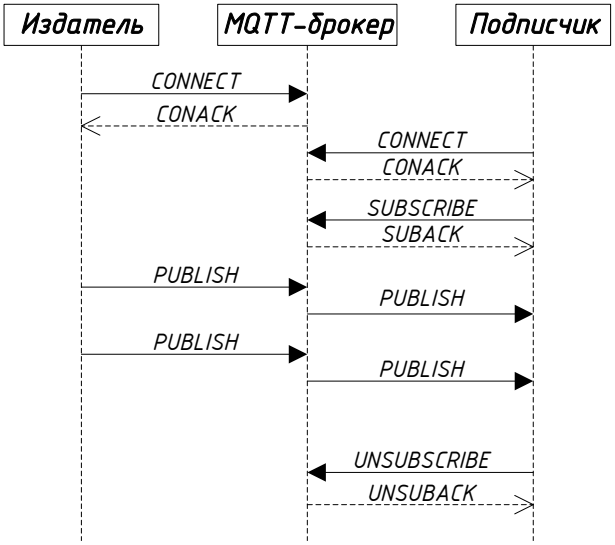

Рисунок 2 – Диаграмма последовательностей MQTT протокола

В процессе работы издатель передает данные с указанной темой сообщения, которое сопоставляется брокером с темами подписчиков. В случае совпадения, каждый из них получает экземпляр данных.

Таблица 1 – Сравнение протоколов MQTT и HTTPS

		3G		**WiFi**	
		HTTP	**MQTT**	**HTTP**	**MQTT**
Приём	**Сообщения / час**	1,708	160,278	3,628	263,314
	% батареи / сообщение	0.01709	0.0001	0,00095	0,00002
Отдача	**Сообщения / час**	1,926	21,685	5,229	23,184
	% батареи / сообщение	0,00975	0,00082	0,00104	0,00016

Небольшие накладные расходы, связанные с хорошо продуманным форматом заголовков, позволяют эффективно применять этот протокол для решений Интернета вещей. В таблице 1 показаны результаты экспериментального сравнения протоколов MQTT и HTTP [3]. Эксперименты показывают, что данный протокол обладает небольшими накладными расходами на стороне устройства, что позволяет сократить расход энергии аккумуляторной батареи и увеличивает количество передаваемых в единицу времени сообщений.

1.3 Архитектора сетей LoRa

Модуляция LoRa [4] определяет физический уровень, который может использоваться в сетях с различной архитектурой – mesh-сети, звезда, точка-точка и другие. Благодаря своей высокой чувствительности (-148dbm) LoRa идеально подходит к устройствам с требованиями низкого потребления электроэнергии и высокой устойчивости связи на больших расстояниях до 20 км.

LoRaWAN – это открытый протокол для высокоёмких (до 1 000 000 устройств в одной сети) сетей с большим радиусом действия и низким энергопотреблением. Сеть LoRaWAN состоит из следующих элементов (Рисунок 3): конечное устройство, шлюзы, сетевой сервер и сервер приложений.

Конечное устройство – предназначено для осуществления управляющих или измерительных функций. Содержит набор необходимых датчиков и управляющих элементов.

Шлюз – устройство, принимающее данные от конечных устройств с помощью радиоканала и передающее их в транзитную сеть (например, Ethernet или WiFi). Шлюз и конечные устройства образуют сетевую топологию типа звезда. Обычно данное устройство содержит многоканальные приёмопередатчики для обработки сигналов в нескольких каналах одновременно или даже, нескольких сигналов в одном канале. Соответственно, несколько таких устройств обеспечивает зону радиопокрытия сети и двунаправленную передачу данных между конечными устройствами и сервером.

Сетевой сервер - предназначен для управления сетью.

Сервер приложений - может удаленно контролировать работу конечных устройств и собирать необходимые данные с них.

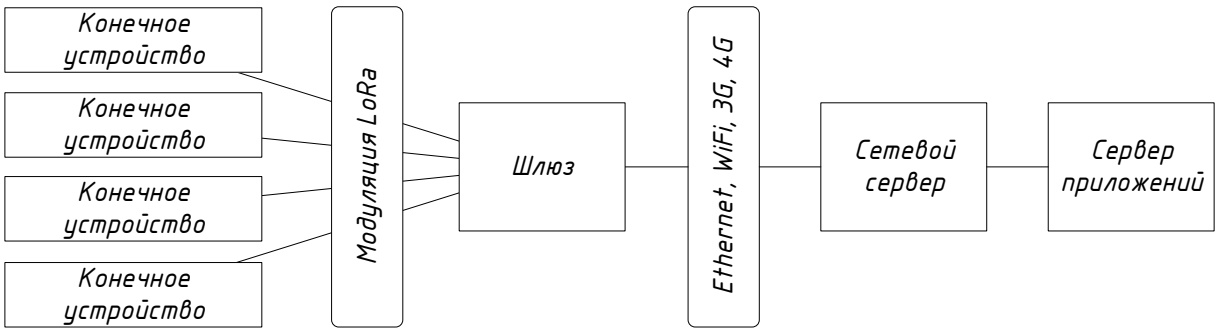

Рисунок 3 – Архитектура сети LoRaWAN

Преимущества LoRaWAN:

- Большая дальность передачи радиосигнала достигает 10 — 15 км.
- Низкое энергопотребление у конечных устройств, благодаря минимальным затратам энергии на передачу небольшого пакета данных.

- Высокая проникающая способность радиосигнала в городской застройке.
- Высокая масштабируемость сети на больших территориях.
- Отсутствие необходимости получения частотного разрешения и платы за радиочастотный спектр, из-за использования нелицензируемых частот (ISM band).

Недостатки LoRaWAN:

- Низкая пропускная способность, составляет от нескольких сотен бит/с до нескольких десятков кбит/с.
- Задержка передачи данных от датчика до конечного приложения, связанная с временем передачи радиосигнала, может достигать от нескольких секунд до нескольких десятков секунд.
- Отсутствие единого стандарта, который определяет физический слой и управление доступом к среде для беспроводных LPWAN-сетей.
- Риски зашумленности спектра нелицензированного диапазона частот.
- Проприетарная технология модуляции LoRa, «закрытая» патентом Semetech.
- Ограничение мощности сигнала.

1.4 Умный транспорт

Интеллектуальные транспортные системы ITS на основе IoT технологий дают возможность реализовывать автоматическое взаимодействие между объектами инфраструктуры и ТС V2I (Vehicle to Infrastructure) или между различными ТС V2V (Vehicle to Vehicle). Системы V2V реализовывают обмен данными по беспроводной связи между ТС на дистанции до нескольких сот метров. Системы V2I реализовывают обмен между ТС и центрами управления дорожным движением. Информация, переданная объектами инфраструктуры, отправляется в общую систему и передаётся близлежащим ТС. Технологии обеих групп могут существенно повысить безопасность и эффективность транспорта.

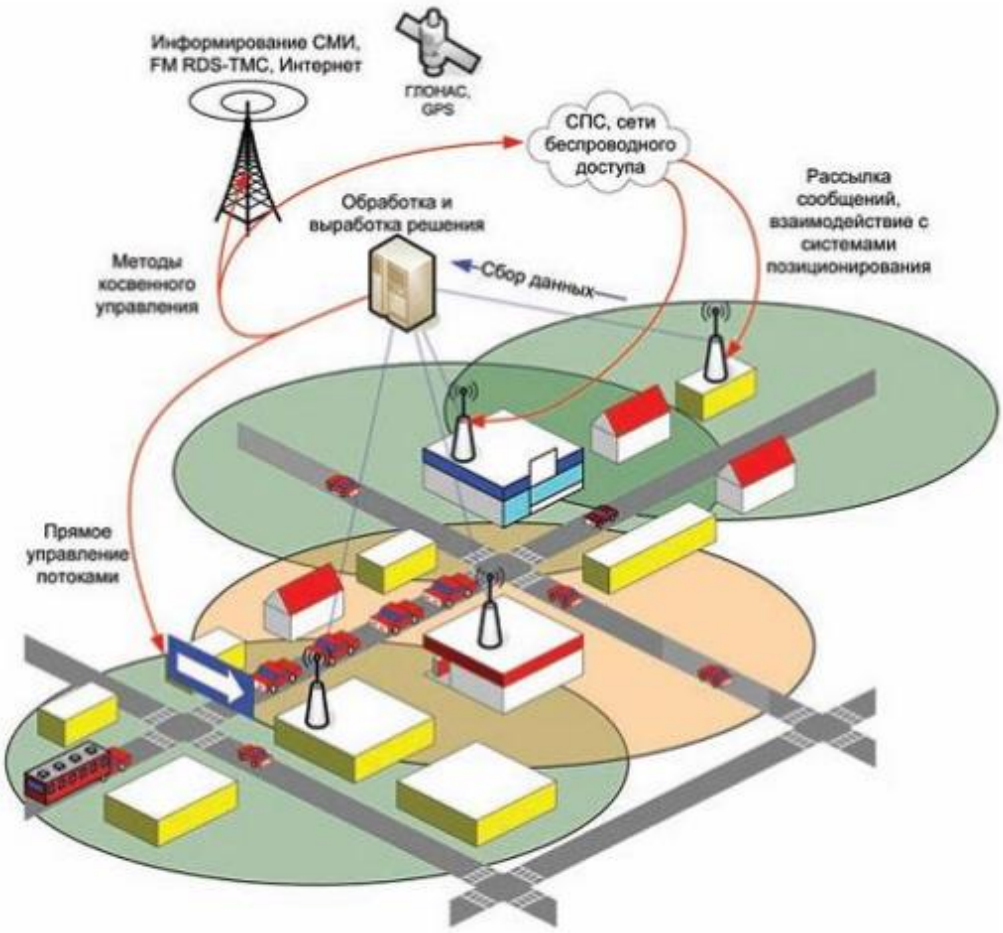

Рисунок 4 – Система интеллектуального управления транспортом

Как пример использования IoT технологий в населенных пунктах можно привести систему управления авто трафиком (Рисунок 4), которая, анализируя пропускную способность дорог, самостоятельно управляет трафиком с помощью перенастройки светофоров и в реальном времени публикует информацию о текущем состоянии, которая доступна другим устройствам и сервисам, например, ГЛОНАСС/GPS-навигатор, мобильный телефон или специализированные веб-сайты.

Применение IoT технологий в транспортной сфере позволяет отслеживать оповещения об опасных ситуациях, а также перенаправлять маршруты движения в режиме реального времени. Помимо этого, благодаря установленным на улицах датчикам появится возможность публикации информации об загруженности дорог и автострад. Например, среди таких «умных» транспортных систем IoT можно упомянуть:

- системы предотвращения столкновений;
- системы «боковой поддержки», указывающие водителю на пересечение дорожных полос или опасные маневры;
- системы ночного видения;
- системы автоматического управления автомобилем и движения в группах автомобилей

- системы мониторинга авто трафика
- системы мониторинга парковочных мест
- системы, контролирующие состояние водителя (например, не позволяющие ему заснуть);
- системы превентивного реагирования на аварийную ситуацию (в частности, системы, которые осуществляют предварительное натягивание ремней безопасности перед неизбежным столкновением).

Система информирования водителей при помощи встраиваемых в машины устройств VICS (Vehicle Information and Communication System) [5] собирает информацию с сенсоров, установленных на объектах дорожной инфраструктуры (дорожном полотне, камерах наблюдения и пр.), с использованием «машин-зондов» (мобильных пунктов наблюдения за дорожным движением), а также с уже установленных бортовых систем, собирающие данные о скорости движения транспортного потока. Эти данные системой VICS обрабатывается и рассылается по бортовым навигационным системам. Пользователи системы могут получать информацию в различных видах - в виде текста, простой графики, карт. Бортовые системы динамически обрабатывают данные и предлагают водителю оптимальный маршрут.

1.5 Обзор существующих СМПМ

Система мониторинга парковочных мест предоставляет в режиме реального времени все необходимые данные о работоспособности парковок, наличии свободных мест участникам дорожного движения (водителям и операторам), позволяет организовать контроль времени парковки автомобилей. В данном направлении сейчас активно ведутся разработки. Например, сейчас работают такие сервисы как: «Яндекс. Парковки» и «Парковки Москвы». Рассмотри их более подробно.

«Яндекс. Парковки» [6] – это мобильное приложение, которое показывает на карте, свободные парковочные места, помогает построить удобный маршрут до ближайшего из них и узнать о степени загруженности парковок. Также имеется возможность оплаты на городские и некоторые коммерческие парковки. Данный сервис работает только на территории Москвы.

«Московский паркинг» [7] – это сервис, который содержит базу парковок при торговых центрах, вокзалах и аэропортах, а также платных и бесплатных стоянок Москвы. Проект призван справиться с проблемой «хаотичного паркования» на улицах Москвы и создать возможность для комфортного передвижения пешеходов, средств общественного транспорта и автомобилей. Имеется возможность оплаты парковки через мобильное приложение.

Как правило СМПМ содержит подсистему фотовидеофиксации ТС и подсистему мониторинга занятости парковочного пространства, при этом система также содержит центральный сервер, отвечающий за синхронизацию и обрабатывание данных от обеих подсистем. Подсистема фотовидеофиксации включает, как минимум, одну видеокамеру, модуль памяти и сервер обработки фотовидеоинформации, соединенный каналами передачи данных с центральным сервером системы. А подсистема мониторинга занятости парковочного пространства включает минимум один детектор занятости парковочного места и сервер мониторинга занятости ПМ, соединенный каналами передачи данных с центральным сервером системы. [8]

Известна система определения занятости парковочных мест (патент RU 103120 U1, МПК: E04H 6/00) [9], включающая по меньшей мере одну камеру, установленную рядом с парковочными местами и средство отображения информации о занятости парковочных мест. Отличается тем, что в качестве камеры используют стереокамеру, установленную с возможностью получения стереоизображения зоны ее наблюдения, включающей несколько парковочных мест. При этом камера снабжена вычислительным модулем, выполненным с возможностью определения занятости парковочных мест по стереоизображению указанной зоны наблюдения и передачи сигнала на средство отображения информации о занятости парковочных мест.

Минусом системы является ее ограниченная функциональность и потребность в установки в зоне парковки большого количества стереокамер, что существенно увеличивает стоимость установки и обслуживания, кроме того потребуются дополнительные технические средства для обрабатывания огромного объема видеоинформации от множества стереокамер. Это делает низкоэффективным использование такой системы в масштабах крупного города с большим количеством парковочных мест.

1.6 Выбор сенсора присутствия автомобиля

Датчик присутствия — электронный прибор, регистрирующий бесконтактными методами объекты определенного класса на территории своего контроля. В зависимости от результатов регистрации он может коммутировать электрические импульсы, по сигналам которых другие устройства выполняют различного рода действия. [10]

По принципу действия различают датчики присутствия:

- ультразвуковые: барьерные, диффузионные;
- фотоэлектрические;
- емкостные;

- датчики нагрузки;
- комбинированные.

Рассмотрим подробно каждый из видов, определим области их применения, оценим достоинства и недостатки.

1.6.1 Ультразвуковые датчики присутствия

Ультразвуковые сенсоры присутствия испускают и принимают волны, не улавливаемые человеческим ухом (частотой порядка 200 кГц).

Возможны два режима работы:

- **Барьерный**: между датчиками, расположенными друг напротив друга, проходит ультразвуковая волна. Она не попадет в приемник, если в зоне действия появится посторонний предмет (барьер).
- **Диффузионный**: с использованием датчика, который испускает волну, а затем улавливает её, отраженную от объекта, оказавшегося на пути луча.

В обоих случаях при появлении постороннего предмета коммутируется сигнал, передаваемый на исполняющие устройства.

Преимущества:

- обнаружение прозрачных объектов;
- невосприимчивость к световым вспышкам и бликам;
- работоспособность в сложных условиях (туман, пыль, пар).

Недостатки:

- низкая дальность (верхний порог) фиксации;
- ненадежность регистрации объектов из мягких материалов;
- наличие “слепой зоны” (нижнего порога обнаружения).

Примеры использования ультразвуковых датчиков: парковочные системы современных автомобилей, подсчет количества единиц готовой продукции на конвейере.

1.6.2 Фотоэлектрические датчики присутствия

Фотоэлектрические датчики работают по схожей с ультразвуковой схемы. Отличие заключается в использовании оптического излучения вместо ультразвукового.

Преимущества:

- высокий порог фиксации (до 150 метров у барьерных датчиков);

- быстродействие;
- отсутствие слепой зоны.

Недостатки:

- невозможность регистрации прозрачных объектов;
- сбои в условиях тумана, пыли, при проявлении световых вспышек и бликов.

Фотоэлектрические датчики используются для контроля за упаковочными и производственными линиями, проверки уровня наполнения прозрачной тары, предотвращения несанкционированного доступа на закрытые территории, остановки промышленного оборудования при попадании человека в опасные зоны.

1.6.3 Ёмкостные датчики присутствия

Конструктивно представляют собой цилиндрические или плоскопараллельные конденсаторы. При появлении объекта в зоне действия изменяется их диэлектрическая проницаемость, а значит и емкость, что вызывает срабатывание.

Преимущества:

- низкая инерционность;
- высокий порог чувствительности.

Недостаток — вероятность сбоев в работе под влиянием внешних электромагнитных полей.

Приборы применяются для контроля за заполнением резервуаров жидкостями и сыпучими материалами, как счетчики единиц готовой продукции и элементы противоугонных систем автомобилей.

1.6.4 Датчики нагрузки

Это конвертеры, преобразовывающие механическое усилие в электрический ток. Конструктивно датчик представляет собой тензорезистор в виде тонкой проволоки, зигзагообразно закрепленный на эластичной подложке. Как упругий элемент используется ткань, резина, полимерная пленка. Под действием силы проводник деформируется, сопротивление его меняется, что генерирует электрический сигнал.

Преимущества:

- малая толщина, обеспечивающая скрытую установку;
- легкость монтажа.

Недостатки:

- необходимость использования усилителя сигнала;
- подверженность многократно повторяющимся механическим нагрузкам, что приводит к выходу из строя;
- снижение чувствительности при перепадах температур

1.7 Выводы

Согласно проведенному исследованию IoT технологии идеально подходят для реализаций в проектах в сфере транспортной инфраструктуры. Это позволит:

- развить новые подходы по планированию и управлению транспортных потоков;
- повысить эффективность системы хранения ТС в городе;
- анализировать огромные массивы данных, и на их основании делать города более удобными, экономичными и прекрасными.

Внедрение высоких технологий в паркинг городов поможет:

- сократить очередь и время поиска свободного места на парковке;
- организации оптимального движения ТС и пешеходов;
- оптимизации работы парковочного пространства.

Были рассмотрены существующие СМПМ (п.п. 1.5.), основными недостатками которых являются: ограниченная область действия, высокая стоимость внедрения.

На основании этого принято решение проектирования системы мониторинга парковочных мест. Данная система должна состоять из «МК-подсистемы регистрации свободных парковочных мест», вычислительного хаба и центрального сервера.

2 Анализ требований к СМПМ

2.1 Разработка диаграммы вариантов использований

Основное назначение СМПМ заключается в определении наличия свободных парковочных мест и отображении этих данных пользователям. Данная система позволяет пользователям (водителям) в режиме реального времени получать актуальную информацию о состоянии парковочных мест.

Для уточнения требований к функционированию системы необходимо определить варианты её использования при помощи диаграммы вариантов использования. Система

взаимодействует с двумя видами действующих лиц: пользователь (водитель) и владелец стоянки. Соответствующие диаграммы для этих лиц показаны на рисунках 5 и 6.

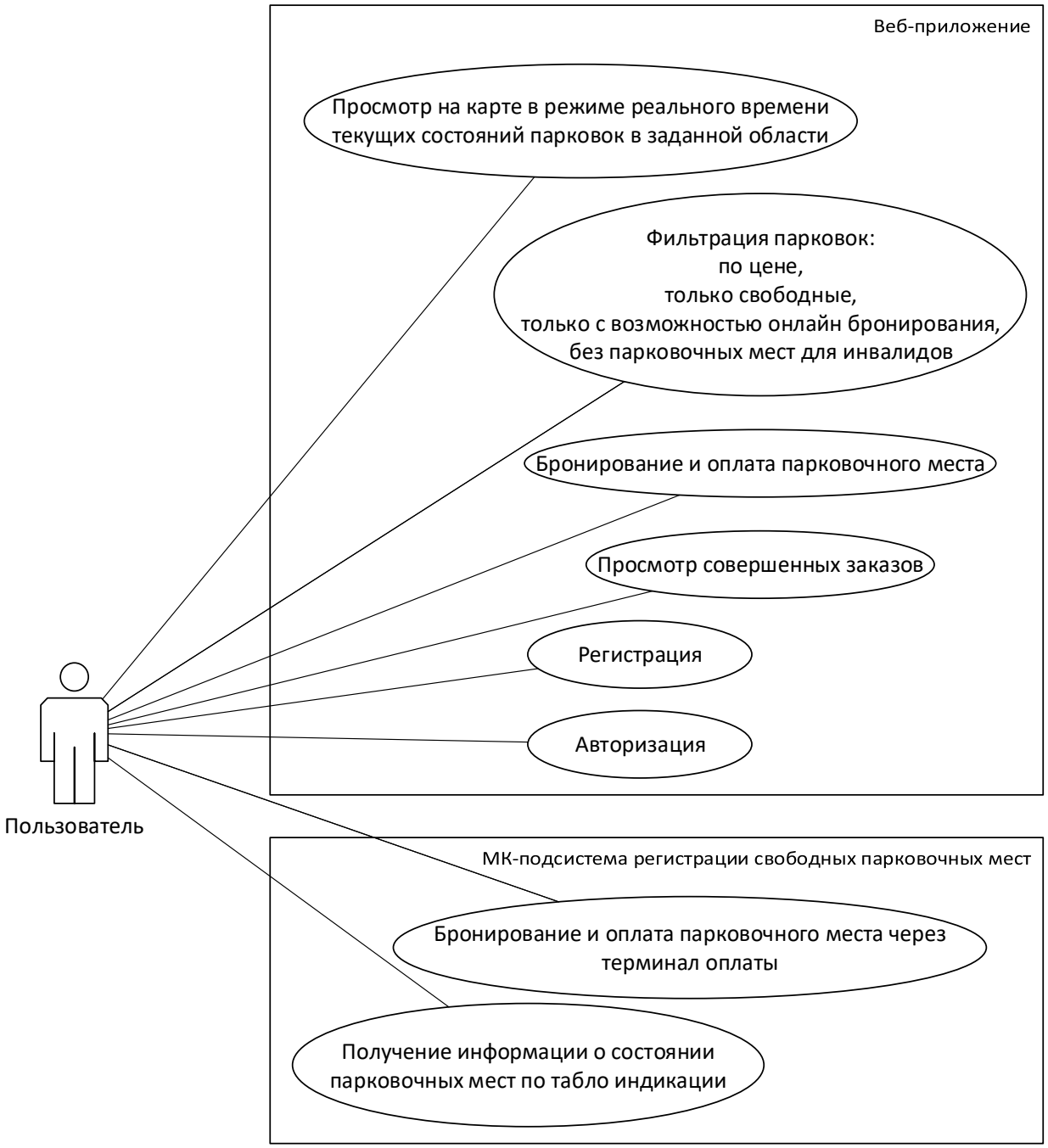

Рисунок 5 – Диаграмма вариантов использований для пользователя

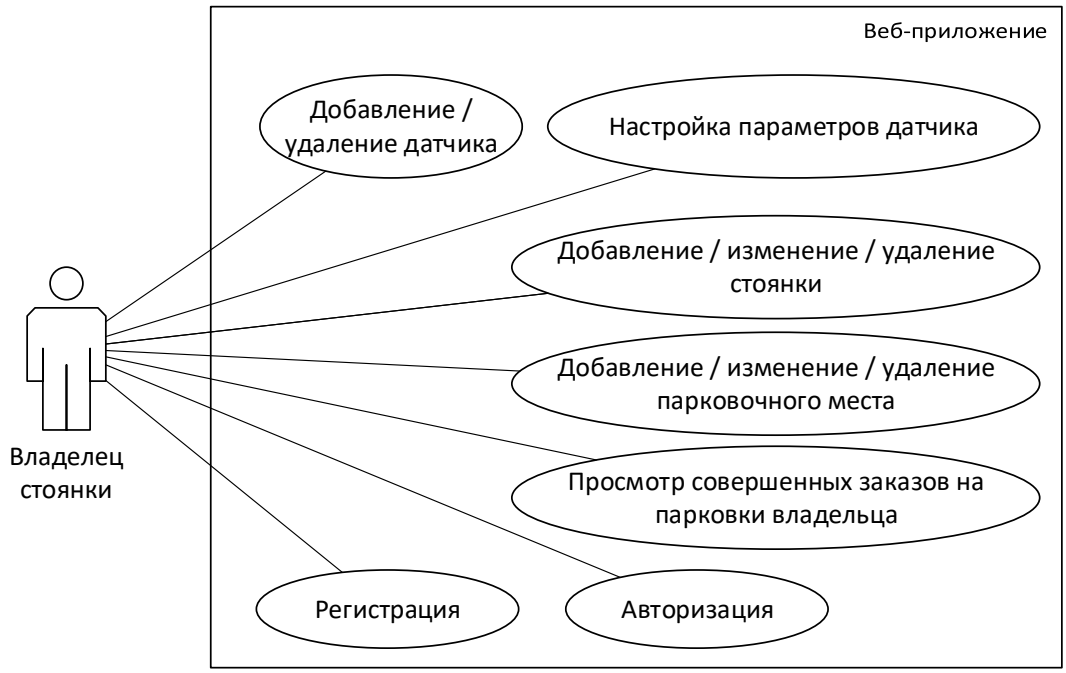

Рисунок 6 – Диаграмма вариантов использований для владельца стоянки

2.2 Описание и структура СМПМ

На рисунке 7 представлена структурная схема системы мониторинга парковочных мест. Система мониторинга парковочных мест состоит из следующих блоков:

- МК-подсистема регистрирования свободных парковочных мест (МКПРСПМ),
- вычислительный хаб,
- сервер.

Датчики МКПРСПМ служат для фиксации наличия свободных парковочных мест и оплаты. МКПРСПМ формирует пакеты с информацией о свободных парковочных местах и передаёт их на вычислительный хаб по беспроводной сети LoRaWAN (Описание системы сообщений МКПРСПМ п.п. 3.8.).

Вычислительный хаб, представленный в виде микрокомпьютера Raspberry Pi, необходим для осуществления взаимодействия сервера, МКПРСПМ и других компонентов системы. Хаб выполняет приём пакетов, переданных с МКПРСПМ, формирует пакеты MQTT и передает их на сервер по сети интернет. Обратно от сервера хаб получает команды управления МКПРСПМ, которые также передаются по сети LoRaWAN, на МК-подсистемы.

Пользователь с мобильного устройства на платформе Android с установленным приложением сможет посмотреть карту с указанием местоположений свободных парковочных мест. Мобильное приложение получает данные о парковочных местах с помощью API-сервисов, предоставляемых сервером.

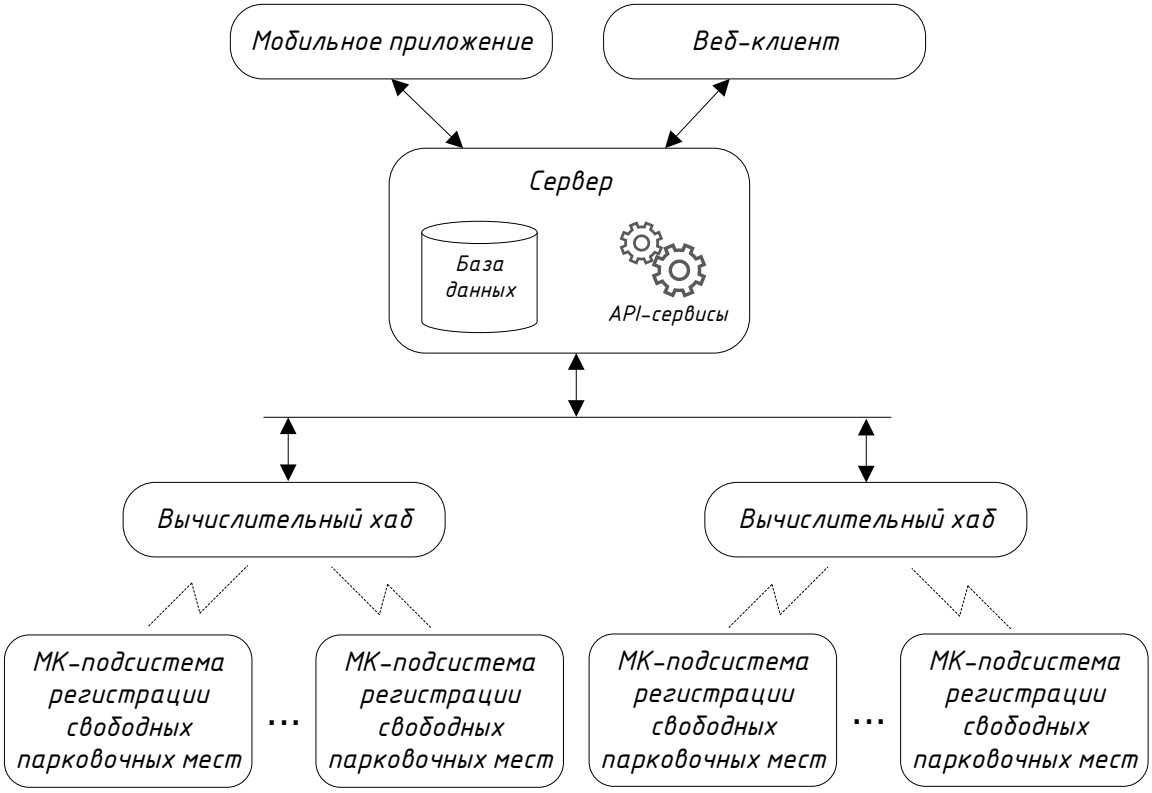

Рисунок 7 – Структурная схема системы мониторинга парковочных мест

Веб-клиент, предоставляет карту со свободными парковочными местами. Также для владельцев стоянок, предоставляется возможность для управления за парковочными местами: изменение тарифного плана, настройка параметров МКПСРСПМ.

2.3 Описание функциональной схемы СМПМ

На рисунке 8 показана функциональная схема СМПМ. Система работает следующим образом. В МКПРСПМ сигнал о состоянии парковочного места приходит от сенсора присутствия автомобиля, после чего МК ATMEGA328 формирует пакеты и отправляет их через радиомодуль RFM95 по сети LoRaWAN на вычислительный хаб. Также по сети LoRaWAN МКРСППМ может принять команды для изменения настроечных параметров.

Вычислительный хаб служит для осуществления взаимодействия сервера, МКПРСПМ и других компонентов системы. Хаб через радиомодуль RFM95 принимает пакеты, переданные с МКПРСПМ, формирует MQTT пакеты и передает их на сервер и на другие компоненты системы. Данные внутри MQTT пакетов передаются в формате JSON. Обратно от сервера хаб получает команды управления МКПРСПМ, которые также передаются по сети LoRaWAN, на МК-подсистемы. ПО для вычислительного хаба написано с использованием фреймворка Qt. На вычислительном хабе может быть установлено серверное ПО, тем самым хаб сформирует локальную сеть, внутри которой сможет работать как сервер. MQTT – это

легкий протокол для передачи данных, который идеально подходит для взаимодействия большого количества IoT устройств. Таким образом, к вычислительному хабу возможно подключить множество других устройств (например, табло индикации количества свободных парковочных мест), позволяющие взаимодействовать по протоколу MQTT.

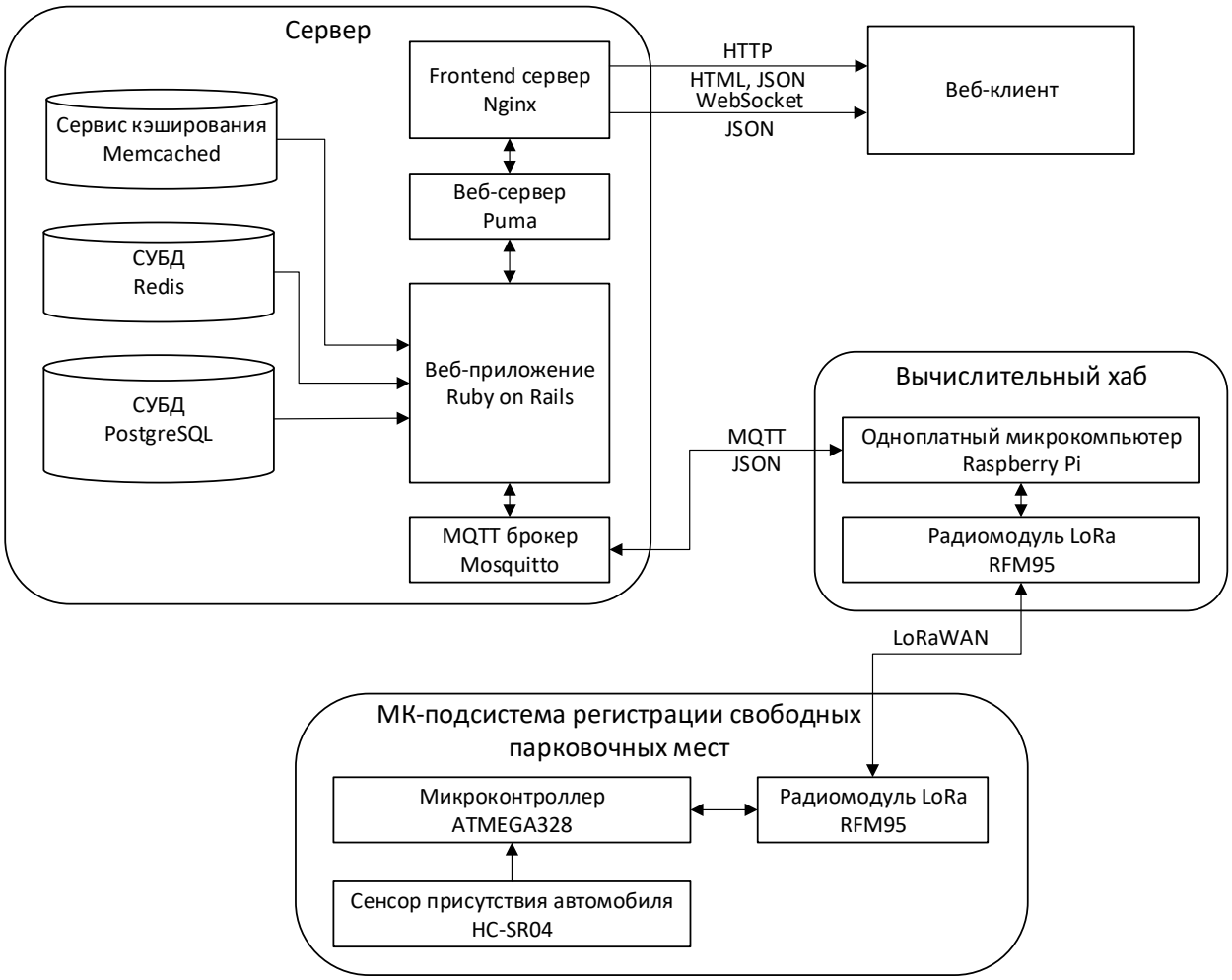

Рисунок 8 – Функциональная схема СМПМ

Сервер взаимодействует с вычислительным хабом по протоколу MQTT. Для этого на сервере установлен MQTT брокер Mosquitto. Приложение на фреймворке Ruby on Rails принимает данные о состоянии ПМ через MQTT брокер и сохраняет в БД PostgreSQL. Также сервер может управлять датчиками МКПРСПМ, отправляя команды по протоколу MQTT на вычислительный хаб. Сервер Nginx является Frontend сервером, он осуществляет раздачу статики и проксирование запросов на веб-сервер Puma, который уже взаимодействует с веб-приложением на Ruby on Rails. Веб-приложение для кэширования использует сервис кэширования Memcached. Для работы Action Cable, который осуществляет взаимодействие с веб-клиентами по WebSocket, необходима «key - value» хранилище данных Redis. Чтобы веб-клиент в режиме реального времени получал данные о состоянии парковочных мест, данные о парковочных местах в формате JSON периодически отправляются по протоколу WebSocket.

3 Разработка МКПРСПМ

3.1 Анализ требований к разрабатываемому устройству

Согласно техническому заданию, необходимо разработать МК-подсистему регистрации свободных парковочных мест, выполненный на основе микроконтроллера ATMEGA328. Данный регистратор фиксирует наличие свободных парковочных мест. К микроконтроллеру подключаются сенсоры, каждый из которых следит за своим парковочным местом. Через радиомодуль информация о состоянии парковочных мест (свободно / занято) передается в вычислительный хаб Raspberry Pi для последующей обработки. Должна быть реализована функция оплаты парковочного места.

Проанализировав задание, можно заключить, что разрабатываемая система должна представлять собой аппаратно-программный модуль, содержащий непосредственно микроконтроллер ATMEGA328, запрограммированный исполняющей программой. К микроконтроллеру через расширители портов PCF8574 подключаются сенсоры, следящие за парковочными местами. Также, расширитель портов PCF8574 используется для вывода информации о состоянии закрепленного за ним парковочного места. Для коммуникации с вычислительным хабом Raspberry Pi используется радиомодуль LoRa. Для реализации функции оплаты парковочного места используется терминал оплаты, состоящий из дисплея и клавиатуры. Работа с терминалом построена в диалоговом режиме.

3.2 Разработка структурной схемы МКПРСПМ

МКПРСПМ служит для фиксации наличия свободных парковочных мест и оплаты. Данная подсистема состоит из следующих элементов (Рисунок 9):

- микроконтроллер ATMEGA328,
- модули контроля парковочного места (МКПМ),
- терминал оплаты,
- радиомодуль RFM95W.
- Часы реального времени DS3231

Каждое парковочное место оборудовано МКПМ, состоящем из сенсора, который определяет свободно ли парковочное место, и расширителем портов PCF8574, для подсоединения сенсора к МК. Выводами «Free» и «Booked» служат для отображения информации о текущем состоянии данного парковочного места. Логическая «1» на выводе

«Free» означает, что данное парковочное место свободно, а «0» – занято. Логическая «1» на выводе «Booked» означает, что данное место забронировано, а «0» – нет.

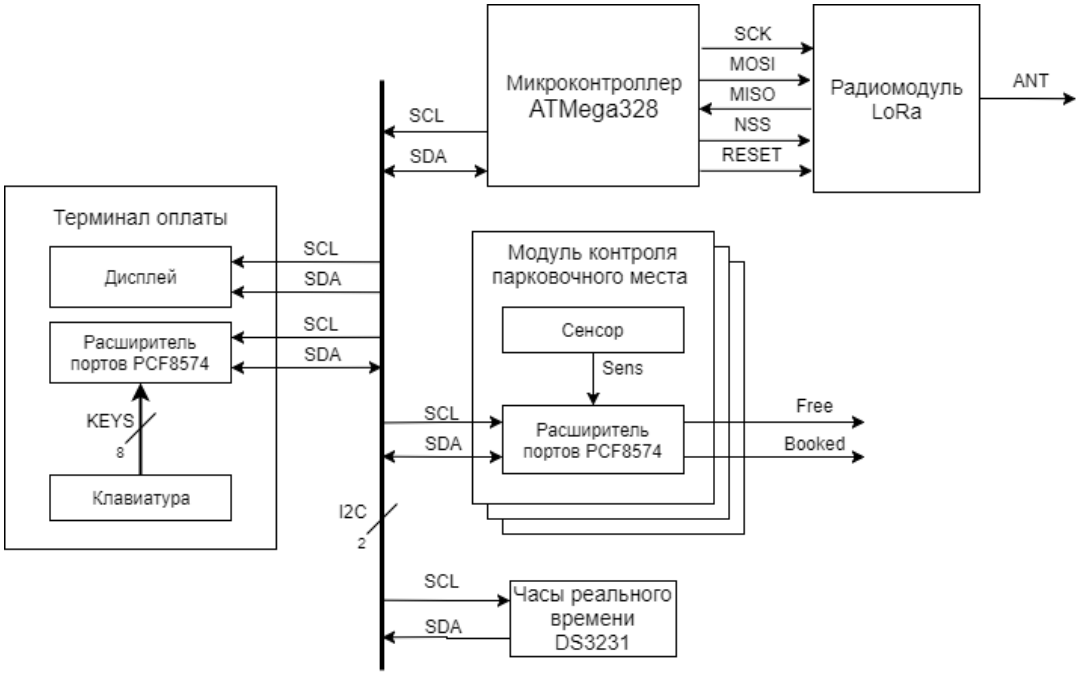

Рисунок 9 – Структурная схема МКПРСПМ

Микроконтроллер ATMEGA328 с помощью модулей МКПМ считывает информацию о состоянии парковочных мест и устанавливает на индикаторах МКПМ соответствующие состояния. Далее микроконтроллер формирует пакеты с информацией о свободных парковочных местах для передачи на вычислительный хаб по беспроводной сети LoRaWAN.

Радиомодуль LoRa осуществляет приём / передачу данных по сети LoRaWAN. Данный модуль подключён к МК по шине SPI.

Терминал оплаты осуществляет оплату парковочного места по заданному тарифу. Он состоит из дисплея, для отображения информации для пользователя, клавиатуры для ввода данных и расширителем портов PCF8574 для соединения клавиатуры с МК по шине I2C.

Часы реального времени DS3231 необходимы для постоянного учёта хронометрических данных (дата, время) на микроконтроллере. Это позволяет изменять тариф для оплаты парковочного места в зависимости от времени суток. А также позволяет отображать текущее время на дисплее. Установка заданного времени на часах происходит при приходе соответствующего сообщения «Изменить время» (Описание системы сообщений МКПРСПМ п.п. 3.8.).

3.3 Разработка функциональной схемы МКПМ

Каждое парковочное место оборудованы МКПМ, который определяет свободно ли данное парковочное место и выводит информацию о состоянии этого парковочного места.

Была разработана функциональная схема МКПМ, представленная на рисунке Е.5. Данная схема описывает функциональный блок «РРСМ» (Рисунок 10), который используется при построении функциональной схемы МКПРСПМ (п.п. 3.4.).

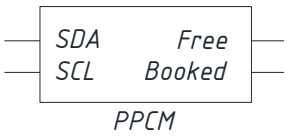

Рисунок 10 – УГО функционального блока РРСМ

Ниже на рисунке 11, показана функциональная схема МКПМ.

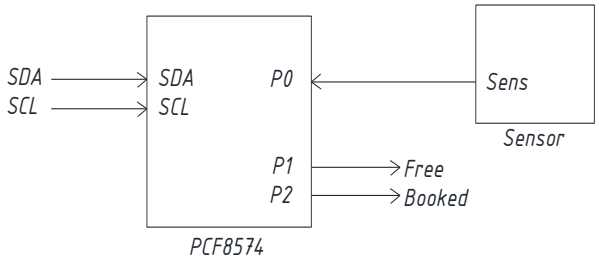

Рисунок 11 – Функциональная схема МКПМ

Схема состоит из двух блоков:

- Сенсор, который определяет свободно ли данное парковочное место.
- Расширитель портов PCF8574, осуществляющий связь сенсора с МК, а также через выводы P1 и P2 выводит информацию о состоянии парковочного места. Расширитель подключен к МК по шине I2C. К выводу P0 подключается сенсор. Выход P1 выводит сигнал «Free», логическая «1» которого означает, что данное парковочное место свободно, а «0» – занято. Выход P2 выводит сигнал «Booked», логическая «1» которого означает, что данное парковочное место забронировано, а «0» – нет.

3.4 Разработка функциональной схемы МКПРСПМ

Исходя из результатов анализа требований (п.п. 3.1.), можно выделить следующие функциональные блоки МКПРСПМ:

- микроконтроллер ATMEGA328;
- радиомодуль LoRa BEE;
- USB-UART драйвер ATMEGA16U2
- модулей МКПМ (PPCM) в количестве до 15 штук, устанавливаемых на каждое парковочное место;
- клавиатура;
- расширитель портов PCF8574 для подключения клавиатуры к микроконтроллеру;
- дисплей;
- часы реального времени DS3231.

В результате проведенного анализа, была спроектирована функциональная схема взаимодействия элементов, приведенная на рисунке Е.6.

Разберем структуру взаимодействия элементов на функциональной схеме. На каждое парковочное место устанавливается МКПМ. Микроконтроллер ATMEGA328 с помощью модулей МКПМ, подключенных к МК по шине I2C, считывает информацию о состоянии парковочных мест. После обработки данных, микроконтроллер изменяет состояние парковочного места (свободно / занято), установив выводы МКПМ в соответствующие состояния. Далее, микроконтроллер формирует пакет с информацией о состоянии парковочного места и отправляет его через шину SPI на радиомодуль LoRa BEE, который, в последствии, передаёт на вычислительный хаб Raspberry Pi, для последующей обработки.

Для осуществления оплаты заданного парковочного места используется терминал оплаты, состоящий из дисплея, матричной клавиатуры и расширителя портов PCF8574 для соединения клавиатуры с МК по шине I2C. Дисплей подключается к МК по шине I2C. В диалоговом режиме на нём отображается информация о текущем состоянии оплаты. МК через расширитель портов опрашивает клавиатуру и определяет нажатые клавиши. Для опроса клавиатуры подаются на выходы расширителя, подключенные к столбцам клавиатуры, сканирующие импульсы, а на входах, подключенных к строкам, определяется, какая клавиша из соответствующего столбца нажата. Далее МК соответствующе обрабатывает нажатые клавиши и отображает результат обработки на дисплей.

3.5 Разработка принципиальной схемы

3.5.1 Выбор элементной базы

3.5.1.1 Микроконтроллер ATMEGA328

В данной работе решено было использовать микроконтроллер семейства AVR, так как он: широко распространен, имеет развитую систему команд, удовлетворяет всем современным технологическим требованиям. Данное семейство восьмиразрядных микроконтроллеров имеет обширное количество различных моделей. Было решено использовать ATMEGA328 как широко распространенный и недорогой микроконтроллер, позволяющий реализовать все поставленные задачи. УГО данного МК показано на рисунке 12.

Рисунок 12 – УГО ATMEGA328

Описание выводов микроконтроллера приведено в таблице 1.

Таблица 2 – Описание выводов МК ATMEGA328

№ вывода	Мнемоника	Назначение вывода
7	VCC	Питание +5В
8	GND	Земля
17	MOSI	Вход ведомого. Служит для передачи данных от ведущего устройства ведомому
18	MISO	Выход ведомого. Служит для передачи данных от ведомого устройства ведущему
19	SCK	Последовательный тактовый сигнал
3-6, 11-19	PD1-PD6, PB0-PB5	Цифровые входы/выходы
23-28	PC0-PC5	Аналоговые входы/выходы
1	PC6	Перезапуск МК
2	PD0	Чтение (разъем для программатора)
9	PB6	Подключение внешнего резонатора (вход)
10	PB7	Подключение внешнего резонатора (выход)

Микроконтроллер ATMEGA328 обладает следующими особенностями и техническими характеристиками [11]:

1) расширенная архитектура RISC:
   - мощная система команд со 131 инструкцией, большинство из которых выполняются за один машинный цикл,
   - 32 восьмиразрядных регистра общего назначения,
   - производительность до 20 млн. оп./с на частоте 20МГц,
   - встроенное умножение за 2 цикла;
2) высокая износостойкость энергонезависимых сегментов памяти:
   - 32 КБ встроенной самопрограммируемой flash-памяти,
   - 1 КБ энергонезависимой памяти (EEPROM),
   - 2 КБ статической памяти (SRAM),
   - 10000/100000 циклов перезаписи для Flash/EEPROM,
   - хранение данных в течение 20 лет при температуре 25°C,
3) системы ввода-вывода: 23 программируемые линии ввода-вывода;
4) рабочее напряжение: от 1.8 до 5.5 В;
5) рабочая температура: от -40° до 85° С;
6) рабочая частота (4.5 - 5.5В): 20 МГц;
7) энергопотребление (1МГц, 1.8 В, 25° С):
   - активный режим: 0.2 мА,
   - режим пониженного энергопотребления: 0.1 мкА.

3.5.1.2 Радиочастотный приёмопередатчика RFM95W

RFM95W — модуль радиочастотного приёмопередатчика, работающий на частоте 868/915 МГц Работа приёмопередатчика основана на технологии LoRa, которая позволяет добиться широкополосной передачи данных при крайне низком энергопотреблении, а также высокой помехоустойчивости. УГО RFM95W представлено на рисунке 13, а описание выводов – в таблице 3.

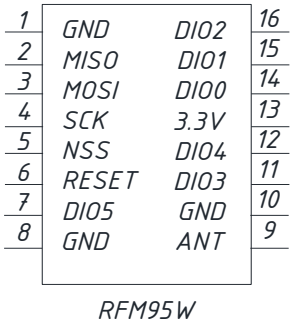

Рисунок 13 – УГО приёмопередатчика RFM95W

Таблица 3 – Описание выводов RFM95W

№ Вывода	Мнемоника	Описание
1, 8, 10	GND	Земля
2	MISO	SPI выход последовательной передачи данных
3	MOSI	SPI вход последовательного приема данных
4	SCK	SPI вход синхронизации приема данных
5	NSS	SPI вход выбора подчиненного
6	RESET	Вход сброса
7, 11, 12, 14, 15, 16	DIO5, DIO3, DIO4, DIO0, DIO1, DIO2	Цифровой вход / выход конфигурации
9	ANT	Вход / выход на антенну
13	3,3V	Питание 3,3В

Технические характеристики:

- Максимальный размер канала связи 168 дБ.
- Режим повышенной мощности, +20дБм/100мВт.
- Стандартный режим +14 дБм.
- Программируемая скорость передачи данных до 300 кбит / с.
- Высокая чувствительность: до -148 дБм.
- Низкий ток потребления пи приёме 10,3 мА.
- Полностью интегрированный синтезатор с разрешением 61 Гц.
- FSK, GFSK, MSK, GMSK, LoRaTM и OOK модуляции.
- 127 дБ динамический диапазон RSSI.
- Пакетный движок до 256 байтов с CRC.
- Встроенный датчик температуры и индикатор разряда батареи.
- Размер модуля: 16x16 мм

3.5.1.3 Расширитель портов PCF8574

PCF8574 [12] – кремневая ИС, выполненная по CMOS-технологии. Посредством двухлинейной двунаправленной шины I2C она обеспечивает дистанционное расширение порта ввода - вывода общего назначения в большинстве серий микроконтроллеров. Устройство состоит из 8-битного квазипорта ввода-вывода и интерфейса I2C-шины. PCF8574 имеет низкий потребляемый ток. В устройстве установлена линия прерывания (INT), которая может быть подключена к логике прерывания микроконтроллера. Посылая сигнал прерывания по этой линии, дистанционный ввод - вывод сообщает микроконтроллеру о поступающих на его порты данных, без необходимости поддерживать связь через I2C-шину. Это значит, что PCF8574 может оставаться простым "подчиненным" устройством. На рисунке 14 приведено УГО модификаций PCF8574P и PCF8574AP.

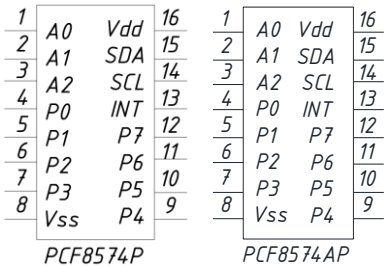

Рисунок 14 – УГО расширителя портов PCF8574P и PCF8574AP

В таблице 4 приведено описание выводов PCF8574.

Таблица 4 – Описание выводов PCF8574P / PCF8574AP

№ Вывода	Мнемоника	Описание
1 – 3	A0 – A2	Адресные входы
4 – 7, 9 – 12	P0 – P3, P4 – P7	Квази-двунаправленный ввод / вывод
8	Vss	Земля
13	INT	Вывод прерывания
14	SCL	Последовательная линия синхронизации
15	SDA	Последовательная линия данных
16	Vdd	Питание 5 В

У данной микросхемы PCF8574 можно задавать адрес, под которым она будет обозначаться на шине I2C. Всего имеется 8 вариантов адресации т.е. можно подключить 8 модулей, что в сумме позволяет нарастить 64 входа/выхода, используя 2 вывода контроллера. Для изменения адресации необходимо установить на выводах A0, A1, A2 потенциал, соответствующий логическому «0» или «1».

Модификация PCF8574AP отличается от PCF8574P только базовым адресом: у PCF8574P – 0x20, а у PCF8574AP – 0x38. Сделано это для того, чтобы к одной шине можно было подключить до 16-ти расширителей с разными адресами — максимум 8 штук PCF8574 и 8 штук PCF8574A.

Технические характеристики:

- Рабочий режим питания - от 2.5 до 6 В
- Низкий ток покоя – максимум 10 мА
- I^2C-шина для расширителя параллельного порта
- Выход прерывания с открытым стоком
- Дистанционный 8-битный расширитель ввода – вывода I^2C-шины
- Адресация на 3 вывода аппаратных адресов для использования до 8 устройств (до 16 устройств при использовании PCF8574A)

В данной работе каждый расширитель портов закреплен за своим парковочным местом. Они используются для считывания данных с сенсоров и для вывода информации о состоянии парковочного места. Был выбран расширитель портов PCF8574, представленный в форм-факторе PCF8574P и PCF8574AP.

3.5.1.4 Сравнение сенсоров присутствия автомобиля

Ниже в таблице 5 приведено сравнение сенсоров присутствия автомобиля:

Таблица 5 – Сравнение сенсоров присутствия автомобиля

	HC-SR04	**E18-D80NK**	**US-015**	**JSN-SR04T**
Тип	Ультразвук-овой	Фотоэлектр-ический	Ультразвук-овой	Ультразвук-овой
Обнаружение прозрачных объектов	+	-	+	+
Невосприимчивость к бликам	+	-	+	+
Работоспособность в условиях недостаточной видимости	+	-	+	+
Дальность обнаружения	500 см	80 см	700 см	450 см
Нижний порог обнаружения	3 см	3 см	2 см	25 см
Цена	90 руб	280 руб	160 руб	730 руб

В результате проведенного сравнения (Таблица 5) В качестве сенсора, следящего за состоянием парковочного места, был выбран ультразвуковой датчик расстояния HC-SR04, т.к. данный ультразвуковой датчик имеет следующие преимущества:

- обнаружение прозрачных объектов;
- невосприимчивость к световым вспышкам и бликам;
- работоспособность в сложных условиях (туман, пыль, пар);
- низкая цена.

Подробное описание датчика HC-SR04 смотри п.п. 3.5.1.5.

3.5.1.5 Ультразвуковой датчик расстояния HC-SR04

Бесконтактный направленный датчик HC-SR04 [13], используя ультразвуковые волны, измеряет расстояние до объекта. На плате модуля размещены пьезоизлучатель ультразвука и воспринимающий отраженную волну микрофон. В отличие от инфракрасных дальномеров на ультразвуковой датчик HC-SR04 не влияют источники света или цвет препятствия. Внешний вид данного датчика показан на рисунке 15. На рисунке 16 представлена УГО датчика расстояния HC-SR04.

Рисунок 15 – Внешний вид ультразвукового датчика расстояния HC-SR04

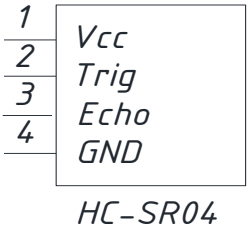

Рисунок 16 – УГО ультразвукового датчика расстояния HC-SR04

Ниже в таблице 6 приводится описание выводов данного датчика.

Таблица 6 – Описание выводов ультразвукового датчика расстояния HC-SR04

№ Вывода	Мнемоника	Описание
1	VCC	Питание +5В
2	GND	Земля
3	Trig	Цифровой вход для включения измерения
4	Echo	Цифровой выход. После завершения измерения, на этот выход будет подана логическая единица на время, пропорциональное расстоянию до объекта

Технические характеристики:

- Напряжение питания 5 В
- Ток потребления в режимах
  - ожидания до 2 мА
  - работы 15 мА
- Частота ультразвука 40 кГц
- Угол обзора 15 градусов
- Измеряемое расстояние
  - от 0,03 до 0,6 м с разрешающей способностью 3 мм
  - от 0,6 до 5 м погрешность увеличивается

Принцип работы:

1) Подается импульс продолжительностью 10 мкс, на вывод Trig.
2) Внутри дальномера входной импульс преобразуется в 8 импульсов частотой 40 КГц и посылается вперед через "Т глазик"
3) Дойдя до препятствия, посланные импульсы отражаются и принимаются "R глазиком" (Рисунок 17). Получаем выходной сигнал на выводе Echo.
4) Непосредственно на стороне контроллера переводим полученный сигнал в расстояние.

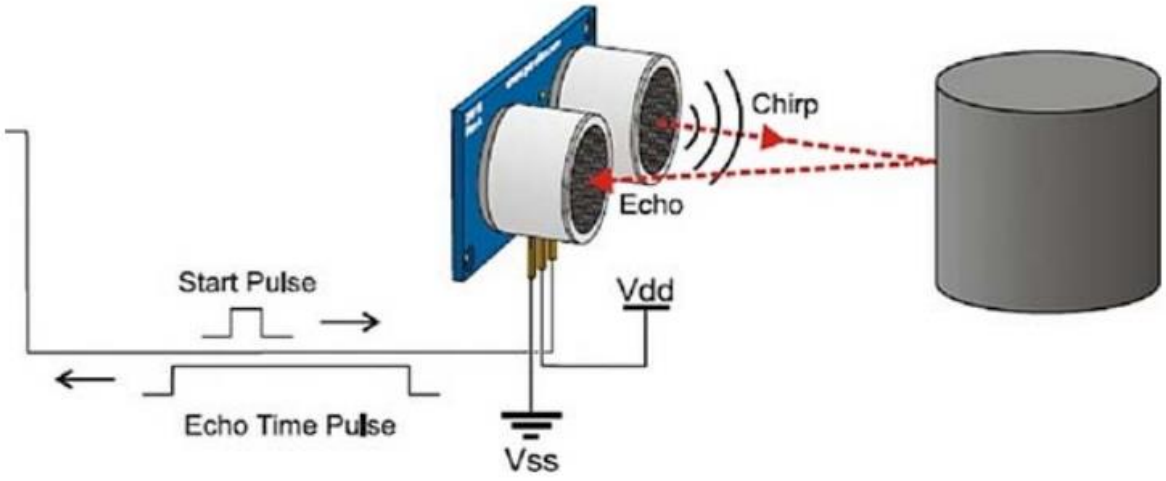

Рисунок 17 – Принцип работы ультразвукового датчика расстояния HC-SR04

3.5.1.6 Часы реального времени DS3231SN

Для постоянного учёта хронометрических данных (дата, время) на микроконтроллере используются часы реального времени DS3231SN. Внешний вид модуля часов реального времени представлен на рисунке 18.

Рисунок 18 – Внешний вид модуля часов реального времени DS3231SN

На рисунке 19 показано УГО для часов реального времени DS3231SN. В таблице 7 приводится описание выводов часов.

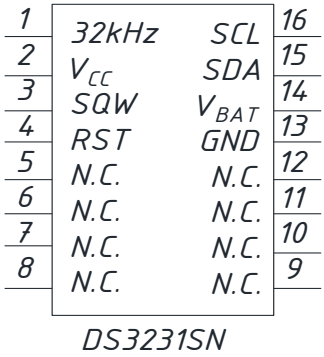

Рисунок 19 – УГО часов реального времени DS3231SN

Таблица 7 – Описание выводов часов реального времени DS3231SN

№ Вывода	Мнемоника	Описание
1	32kHz	Тактовый вывод с частотой 32кГц
2	V_{CC}	Питание +5В
3	SQW	Программируемый выход Square-Wave сигнала
4	RST	Сброс
5 – 12	N.C.	Не соединения
13	GND	Земля
14	V_{BAT}	Питание от батареи 3,3В
15	SDA	Последовательная линия данных
16	SCL	Последовательная линия синхронизации

Технические характеристики:

- Рабочая температура -40°C ~ 85°C
- Напряжение питания, батареи 2,3 V ~ 5,5 V
- Напряжение питания 2,3 V ~ 5,5 V
- Интерфейс подключения I²C

Микросхема DS3231 представляет собой высокоточные часы реального времени RTC, которая обладает встроенным кварцевым генератором с температурной компенсацией, благодаря чему уход времени составляет всего ±2 минуты за год. Дополнительно реализована функция будильника, также имеется выход прерываний.

Микросхема использует интерфейс передачи данных I2C. Адрес микросхемы (7 бит) на шине I2C равен 1101000.

3.5.2 Описание принципиальной схемы МКПМ

Были спроектированы принципиальные схемы МКПМ на базе PCF8574P и PCF8574AP, которые представлены на рисунках Е.8 и Д10 соответственно. Данные схемы соответственно описывают блоки PPCM и PPCM_A, УГО которых изображены на рисунке 20. Данные модули отличаются от друг драга – модификацией расширителя порта PCF8574, на которой они построены. Модуль PPCM построен на базе расширителя порта PCF8574P, а PPCM_A – на базе PCF8574AP. Функциональная схема данных модулей представлена на рисунке Е.5 и была описана в п.п. 0. В таблице 8 приведено описание выводов МКПМ.

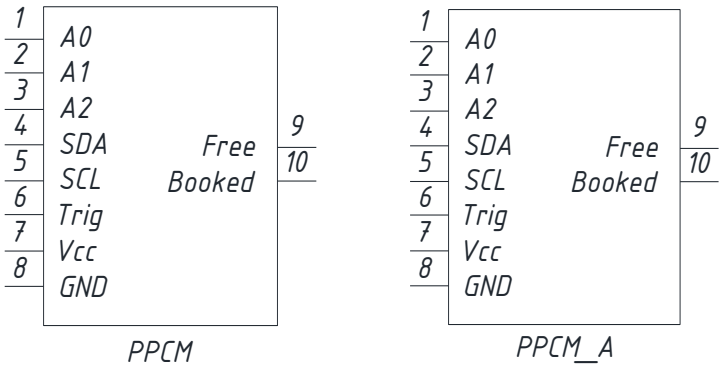

Рисунок 20 – УГО модулей PPCM и PPCM_A

Таблица 8 – Описание выводов модулей PPCM / PPCM_A

№ Вывода	Мнемоника	Описание
1 – 3	A0 – A2	Адресные входы
4	SDA	Последовательная линия данных
5	SCL	Последовательная линия синхронизации
6	Trig	Цифровой вход для включения измерения
7	V_{CC}	Питание +5В
8	GND	Земля
9	Free	Выводит сигнал «Free» логическая «1» означает, что данное парковочное место свободно, а «0» – занято
10	Booked	Выводит сигнал «Booked» логическая «1» означает, что данное парковочное место забронировано, а «0» – нет

За каждым парковочным местом закреплен свой МКПМ, состоящий из расширителя портов PCF8574 (DD1) и сенсора HC-SR04 (DD2). Расширитель портов осуществляет связь МК с сенсором и выводит информацию о состоянии парковочного места. К выводам P1 и P2 расширителя соответственно подключены светодиоды VD1 и VD2, которые наглядно показывают состояние данного парковочного места. Данный расширитель портов подключены к МК по шине I2C. PCF8574 имеет 3 адресных входа, таким образом, можно подключить до 8 таких устройств, следовательно, МКПРСПМ может обслуживать до 8 парковочных мест. Модификация PCF8574AP имеет другой базовый адрес, следовательно, параллельно можно подключить еще 8 устройств PCF8574AP, тогда разрабатываемое устройство сможет обслуживать до 15 парковочных мест + 1 расширитель для подключения клавиатуры к МК. В таблице 9 приведено описание ролей выводов расширителя портов PCF8574 (DD2).

Таблица 9 – Роли выводов расширителя портов PCF8574P / PCF8574AP

Контакт	Роль
1, 2, 3 – A0, A1, A2	Адресные входы
4 – P0	Выход на вывод Trig сенсора HC-SR04 для включения измерения
5 – P1	Выводится логическая «1», в случае если парковочное место свободно, иначе – «0»
6 – P2	Выводится логическая «1», в случае если парковочное место забронировано, иначе – «0»
8 – Vss	Земля
14 – SCL	Последовательная линия синхронизации
15 - SDA	Последовательная линия данных
16 – Vdd	Питание +5В

Сенсор HC-SR04 (DD2) осуществляет мониторинг занятости данного парковочного места. Вход Trig сенсора соединен с выводом P0 расширителя портов PCF8574 (DD1). Для включения измерения сенсора МК подаёт сигнал на вывод P0 соответствующего расширителя портов. После завершения измерения на выходе Echo сенсора HC-SR04 будет подана логическая единица на время, пропорциональное расстоянию до объекта. Выход Echo сенсора соединён с МК через вход PD3. В таблице 10 приведено описание ролей выводов данного сенсора.

Таблица 10 – Роли выводов сенсора HC-SR04

Контакт	Роль
1 – VCC	Питание +5В
2 – GND	Земля
3 – Echo	Цифровой вход для включения измерения.
4 – Trig	Цифровой выход, после завершения измерения, будет подана логическая единица на время, пропорциональное расстоянию до ТС

Расширители портов PCF8574 (DD1) выдают очень низкий ток 100 мкА, поэтому были добавлены MOSFET-транзисторы (VT1, VT2) для усиления выходных сигналов «Free» и «Booked». В цепь затвора транзисторов были добавлены подтягивающие резисторы (R3, R5), чтобы гарантированно удерживать высокий уровень на затворе мосфета при отсутствии сигнала низкого уровня от расширителя (DD1). Это исключает самопроизвольное выключение транзистора. В разрыв цепи затвора также установлены резисторы (R1, R2) номиналом 100 Ом, для предотвращения кратковременных выбросов тока и защиты вывода PCF8574.

3.5.3 Описание принципиальной схемы МКПРСПМ

На основе описанной элементной базы (п.п. 3.5.) была разработана принципиальная схема МКПРСПМ для 15 парковочных мест, которая доступна для ознакомления на рисунке Е.13.

Модуль RFM95W (DD7), осуществляющий передачу и приём данных по сети LoRa, подключен к МК ATMEGA328P-PU (DD8) шине SPI. В таблице 11 приведено описание подключения модуля RFM95W к МК.

Таблица 11 – Подключение модуля RFM95W к МК ATMEGA328P

Контакт RFM95W	**Контакт МК**	**Роль**
1, 8, 10 – GND	–	Земля
2 – MISO	18 – PB4 (MISO)	Приём данных от модуля на МК
3 – MOSI	17 – PB3 (MOSI)	Передача данных от МК на модуль
4 – SCK	19 – PB5 (SCK)	Синхронизация передачи данных
5 – NSS	16 – PB2 (SS)	Выбор подчиненного устройства
6 – RESET	15 – PB1	Перезапуск модуля
9 - ANT	–	Вывод на антенну
13 – 3,3V	–	Питание +3,3В
14 – DIO0	4 – PD2 (INT0)	Возникновение прерывание на МК при успешном приёме / передачи пакета по сети LoRa

Микроконтроллер Arduino Uno R3 содержит встроенный драйвер USART-USB, выполненный на ATMEGA16U2 (DD2), необходимый для передачи и приёма данных на ПЭВМ по последовательному интерфейсу. Таким образом, с помощью данного модуля можно, как альтернатива модулю RFM95, осуществлять приём / передачу данных.

Для работы шины I2C необходима внешняя подтяжка к питанию, состоящая из двух резисторов. В данном проекте такими резисторами являются R14 и R15, номинал которых равен 4,7КОм.

За каждым парковочным местом закреплен свой модуль МКПМ (DD10 – DD24), которые разделяются на две модификации PPCM (DD11 – DD17) и PPCM_A (DD10, DD18 – DD24). Данные модули подключаются к МК по шине I2C. PPCM отличается от PPCM_A только базовым адресом: у PPCM – 0x20, а у PPCM_A – 0x38. Таким образом, можно подключить до 15 таких устройств, что позволит МКПРСПМ обслуживать 15 парковочных мест. В таблице 12 приведено описание подключения модулей МКПМ к МК.

Таблица 12 – Подключение модуля PPCM / PPCM_A к МК ATMEGA328P

Контакт МКПМ	Контакт МК	Роль
1, 2, 3 – A0, A1, A2	–	Адресные входы
4 – SDA	27 – PC4	Последовательная линия данных
5 – SCL	28 – PC5	Последовательная линия синхронизации
6 – Trig	5 – PD3	Цифровой вход для включения измерения
7 – V_{CC}	–	Питание +5В
8 – GND	–	Земля
9 – Free	–	Выводит сигнал «Free» логическая «1» означает, что данное парковочное место свободно, а «0» – занято
10 – Booked	–	Выводит сигнал «Booked» логическая «1» означает, что данное парковочное место забронировано, а «0» – нет

Часы реального времени DS3231SN (DD6), используются для учета хронометрических параметров. Они подключаются к МК по шине I2C. Чтобы при отключении питания МКПРСПМ время на часах не сбрасывалось, часы дополнительно питаются от гальванического элемента G1, напряжение питания которого равно +3,3В. В таблице 13 описывается подключение часов к МК.

Таблица 13 – Подключение часов реального времени DS3231SN к МК ATMEGA328P

Контакт DS3231SN	Контакт МК	Роль
2 – V_{CC}	–	Питание +5В
13 – GND	–	Земля
14 – V_{BAT}	–	Питание от батареи +3,3В
15 – SDA	27 – PC4	Последовательная линия данных
16 – SCL	28 – PC5	Последовательная линия синхронизации

Клавиатура AK-1604-N-WWB (DD5) подключается к МК посредством расширителя портов PCF8574P (DD4). Строки клавиатуры (ROW0 – ROW3) подключаются к выводам расширителя P0 – P3, а столбцы (COL0 – COL3) – к выводам P4 – P7. Таким образом, для подключения клавиатуры используются не 8 выводов МК, а 2, т.е. линии шины I2C. В таблице 14 приводится описание подключения данного расширителя PCF8574P (DD4) к микроконтроллеру.

Таблица 14 – Подключение расширителя портов PCF8574P, отвечающего за клавиатуру, к МК ATMEGA328P

Контакт PCF8574P	Контакт МК	Роль
1, 2, 3 – A0, A1, A2	–	Адресные входы
4, 5, 6, 7 – P0, P1, P2, P3	–	Выводы для подключения строк клавиатуры
8 – Vss	–	Земля
9, 10, 11, 12 – P4, P5, P6, P7	–	Выводы для подключения столбцов клавиатуры
14 – SCL	28 – PC5	Последовательная линия синхронизации
15 – SDA	27 – PC4	Последовательная линия данных
7 – V_{DD}	–	Питание +5В

Дисплей WAVGAT_128x64 (DD9) подключается к МК по шине I2C. Ниже в таблице приведено описание подключения этого дисплея к МК.

Таблица 15 – Подключение дисплея WAVGAT_128x64 к МК ATMEGA328P

Контакт WAVGAT_128x64	Контакт МК	Роль
1 – GND	–	Земля
2 – V_{CC}	–	Питание +5В
3 – SCL	28 – PC5	Последовательная линия синхронизации
4 – SDA	27 – PC4	Последовательная линия данных

В качестве драйвера USB-UART используется микроконтроллер ATMEGA16U2 (DD2), обеспечивающий связь микроконтроллера с USB-портом ПЭВМ. При подключении к ПК Arduino UNO определяется как виртуальный COM-порт. Заводская прошивка ATMEGA16U2 использует стандартные драйвера USB-COM — установка внешних драйверов не требуется, что делает принцип работы микросхемы удобным в эксплуатации. В таблице 12 приведено описание подключения драйвера к МК.

Таблица 16 – Описание выводов ATMEGA16U2

Контакт ATMEGA16U2	Мнемоника	Назначение вывода
4 – VCC	–	Питание +5В
3 – GND	–	Земля
32 – AVCC	–	Питание для всех аналоговых устройств
8 – PD2	2 – PD0	Приём данных на МК
8 – PD3	3 – PD1	Передача данных от МК
30 – D-	–	Порт USB с отрицательной обратной связью
29 - D+	–	Порт USB с положительной обратной связью
28 – UGND	–	USB земля
27 – UCAP	–	USB регулятор выходного напряжения питания
1 – XTAL1	–	Подключение внешнего резонатора (вход)
2 – XTAL2	–	Подключение внешнего резонатора (выход)

USB-разъем, помимо предоставления интерфейса для последовательной передачи данных к ПЭВМ, позволяет использовать компьютер, как источник питания. Для токовой защиты установлен самовосстанавливающийся предохранитель MF-MSMF050-2 (F1), защищающий порт USB компьютера от токов короткого замыкания и сверхтоков. Хотя практически все компьютеры имеют подобную защиту, тем не менее, данный предохранитель обеспечивает дополнительный барьер. Предохранитель срабатывает при прохождении тока более 500 мА через USB порт и размыкает цепь до тех пор, пока нормальные значения токов не будут восстановлены.

3.6 Расчет потребляемой мощности

Суммарная мощность, которую потребляет устройство – есть сумма мощностей элементов, из которых это устройство состоит. Мощность – произведение напряжения питания и тока, который протекает через микросхему.

Для расчета потребляемой мощности используем информацию из документации используемых микросхем и других элементов. Информация о токах и соответственно мощности, потребляемых микросхемами, приведена в таблице 17.

Таблица 17 – Расчет мощности компонентов

Наименование микросхемы	Ток потребления, мА	Напряжение питания	Потребляемая мощность, мВт
ATMEGA328P	16	5	80
ATMEGA16U2	50	5	250
LoRa BEE	10,3	3,3	40
PCF8574P	80	5	400
HC-SR04	15	5	75
DS3231SN	3	5	15
WAVGAT_128x64	16	5	80

Используя информацию из таблицы, рассчитаем суммарную потребляемую мощность, для МКПРСПМ обслуживающего одно парковочное место:

$$P = 80 + 250 + 40 + 400 + 75 + 400 + 15 + 80 = \mathbf{1340}\ \text{мВт}$$

Для МКПРСПМ обслуживающего 15 парковочных мест, потребляющая мощность будет составлять:

$$P = 80 + 250 + 40 + (400 + 75) * 15 + 400 + 15 + 80 = \mathbf{8710}\ \text{мВт}$$

3.7 Описание разработанного программного обеспечения

3.7.1 Описание схемы алгоритма

Для реализации указанных в задании функций было разработано программное обеспечение, обеспечивающее взаимодействие элементов микроконтроллерной системы и обработку поступающих данных.

Схема алгоритма основной программы представлена на рисунке 21. Первоначально выполняется настройка и инициализация всех используемых элементов, после чего программа переходит в бесконечный цикл опроса.

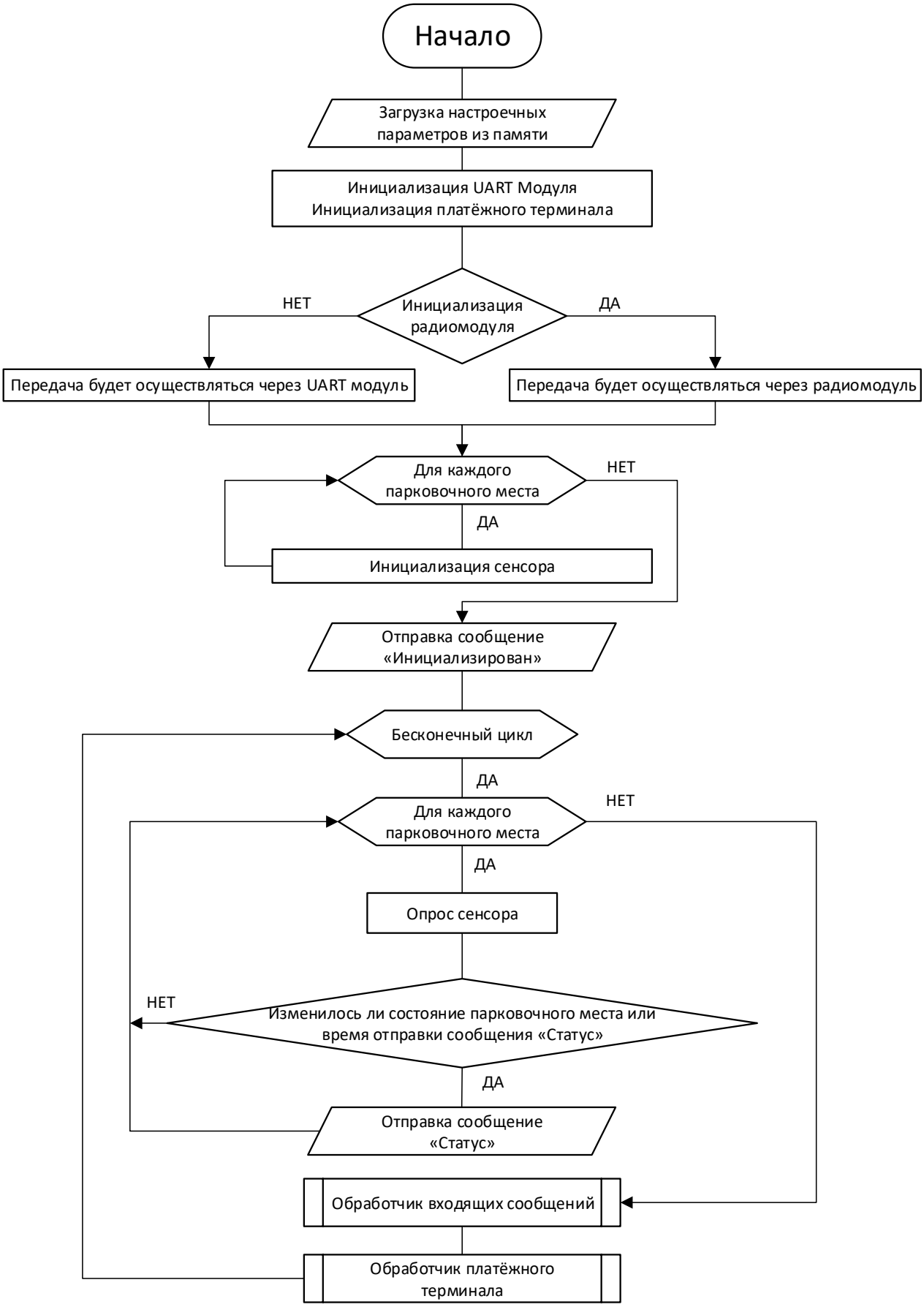

Рисунок 21 – Схема алгоритма основной программы

Блок «Обработчик входящих сообщений» является составным для компактности основной блок схемы. Блок схема этой процедуры представлена на рисунке 22.

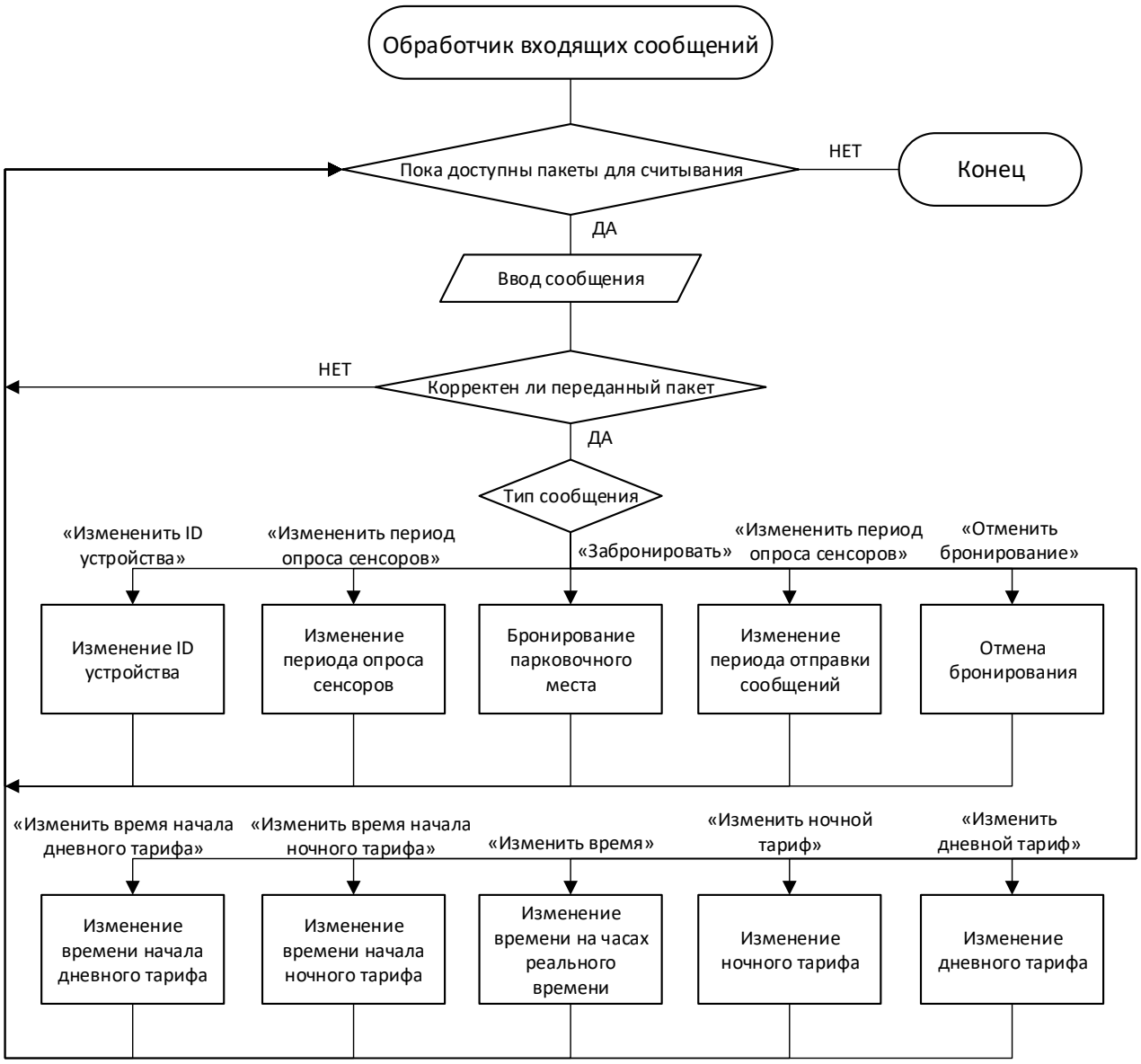

Рисунок 22 – Схема алгоритма процедуры «Обработчик входящих сообщений»

Ниже на рисунке 23 представлена блок схема процедуры «Обработчик платёжного терминала».

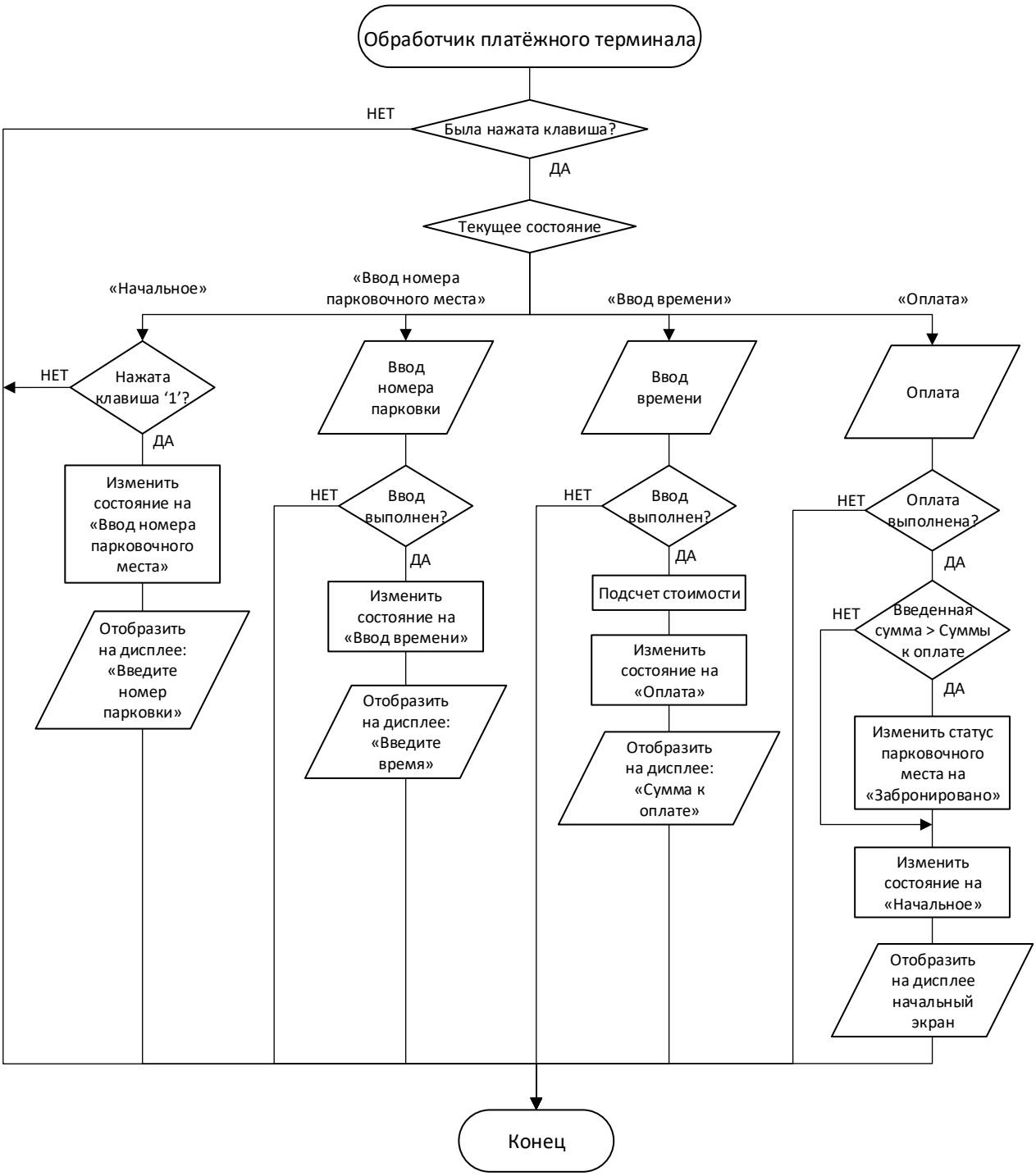

Рисунок 23 – Схема алгоритма процедуры «Обработчик платёжного терминала»

3.7.2 Описание диаграммы классов

При разработке программы применялся преимущественно объектный подход, так как данный подход является наиболее удобным для дальнейшей разработки и поддержки проекта. На рисунке Е.14 представлена диаграмма классов разработанной программы. Как можно видеть из диаграммы классов было реализовано два паттерна «Стратегия» и один «Одиночка».

Паттерн «Стратегия» позволяет выбирать алгоритм путём определения соответствующего класса. Класс «ReceiverTransmitter» является интерфейсом для классов «RadioModule» и «SerialModule». Это позволило в основной программе реализовать процедуру выбора драйвера для приёма и передачи данных. В случае отсутствия радиомодуля LoRa, то в качестве драйвера будет выбран драйвер USB-UART.

При создании объекта класса «ReceiverTransmitter» в конструктор передаётся обработчик входящих сообщений – объект, принадлежащий абстрактному классу «AbstractReceiveMessageHandler». Это позволяет задавать различные обработчики входящих сообщений. Например, операция по изменению ID устройства является небезопасной, и было принято решение запретить выполнять данную операцию при приёме сообщений по беспроводной связи, однако при подключении по UART интерфейсу операция разрешена.

Паттерн «Одиночка» – шаблон проектирования, гарантирующий, что в однопроцессном приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру. «Одиночкой» является класс «Parameters», который содержит в себе настроечные параметры, и осуществляющий запись параметров память EEPROM. Определив этот класс «Одиночкой», позволило иметь доступ к единственному экземпляру данного класса из любого места программы.

3.8 Описание системы сообщений МКПРСПМ

Реализована система сообщений для приёма и передачи информации между МКПРСПМ и вычислительным хабом. Каждое принимаемое и отправляемое сообщение состоит из заголовка и тела сообщения.

Структура заголовка:

1) 1 байт – Тип сообщения
2) 4 байта – Идентификатор МКПРСПМ

Далее рассматривается структура тела сообщения для каждого типа сообщений, отправляемых с МКПРСПМ:

- Инициализация МКПРСПМ:
    1) 2 байта – Период опроса датчиков
    2) 2 байта – Период отправки сообщений о состоянии парковочного места
    3) 2 байта – Дневной тариф
    4) 2 байта – Ночной тариф
    5) 4 байта – Время начала дневного тарифа
    6) 4 байта – Время начала ночного тарифа
- Состояние парковочного места:
    1) 1 байт – Идентификатор парковочного места
    2) 1 байт – Булевое значение свободно ли парковочное место
- Оплата парковочного места:
    1) 1 байт – Идентификатор парковочного места
    3) 4 байта – Время бронирования
    4) 2 байта – Внесенная сумма
    5) 2 байта – Итоговая стоимость

Структура тела сообщения для основных типов принимаемых сообщений:

- Изменить идентификатор МКПРСПМ.
    1) 4 байта – Новый идентификатор МКПРСПМ
- Изменить период отправки сообщений о состоянии парковочного места:
    1) 2 байт – Период отправки сообщений о состоянии парковочного места
- Изменить время на часах реального времени:
    1) 4 байта – Время
- Изменить дневной тариф
    1) 2 байта – Стоимость 1 часа в рублях
- Изменить параметры настройки
    1) 2 байта – Период опроса датчиков
    2) 2 байта – Период отправки сообщений о состоянии парковочного места
    3) 2 байта – Дневной тариф
    4) 2 байта – Ночной тариф
    5) 4 байта – Время начала дневного тарифа
    6) 4 байта – Время начала ночного тарифа

- Бронирование парковочного места
    1) 1 байт – Идентификатор парковочного места
    2) 2 байта – Время бронирования
- Отмена бронирования парковочного места
    1) 1 байт – Идентификатор парковочного места

При отправке сообщения через UART интерфейс перед заголовком добавляется еще 1 байт – длина сообщения.

4 Разработка вычислительного хаба

4.1 Анализ требований

Согласно анализу технического задания, консольное приложение вычислительного хаба должен реализовывать следующий функционал:

- прием пакетов данных, полученные от датчиков МКПРСПМ по сети LoRaWAN или через последовательный порт UART и формирование MQTT-пакетов в формате JSON и передача их на сервер и на другие устройства СМПМ;
- приём MQTT пакетов данных в формате JSON, полученные от сервера и передача их на МКПРСПМ;
- конфигурационный файл для настройки параметров ПО вычислительного хаба.

Для реализации ПО вычислительного хаба использовались библиотеки фреймворка Qt, а также библиотека RadioHead для работы с радиомодулем RFM95. Основное преимущество использования Qt фреймворка – является то, что он является кроссплатформенным, т.е. разработанное ПО можно будет запускать на различных устройствах и ОС. Таким образом данное ПО можно будет установить не только на микрокомпьютере Raspberry Pi, но и на обычном ПК для тестирования или для непосредственного использования данной системы. Также в Qt имеются такие удобные библиотеки как:

- «*QMqttClient*» – интерфейс для взаимодействия с MQTT брокером (появилась только в Qt 5.10);
- «*QSerialPort*» – интерфейс для взаимодействия с последовательными портами;
- «*QSettings*» – класс для реализации постоянных платформонезависимых настроек приложения.

4.2 Подключение радиомодуля RFM95

Радиомодуль RFM95 (п.п. 3.5.1.2. Радиочастотный приёмопередатчика RFM95W) используется для связи вычислительного хаба с датчиками МКПРСПМ по сети LoRaWAN. Если вычислительный хаб представлен в виде микрокомпьютера Raspberry Pi 3 Model B, то в таблице 18 приведено подключение радиомодуля к данному микрокомпьютеру:

Таблица 18 – Подключение модуля RFM95W к Raspberry Pi 3 Model B

Контакт RFM95W	**Контакт Raspberry Pi 3 Model B**	**Роль**
1, 8, 10 – GND	20 – GND	Земля
2 – MISO	21 – SPI0 MISO	Приём данных от модуля на Raspberry
3 – MOSI	19 – SPI0 MOSI	Передача данных от Raspberry на модуль
4 – SCK	23 – SPI0 SCLK	Синхронизация передачи данных
5 – NSS	24 – SPI0 CE0	Выбор подчиненного устройства
9 - ANT	–	Вывод на антенну
13 – 3,3V	17 – 3,3V	Питание +3,3В
14 – DIO0	22 – GPIO25	Возникновение прерывание на Raspberry при успешном приёме / передачи пакета по сети LoRa

Ниже на рисунке 24 показано подключение модуля LoRa BEE на базе радиочастотного приёмопередатчика RFM95W к микрокомпьютеру Raspberry Pi 3 Model B:

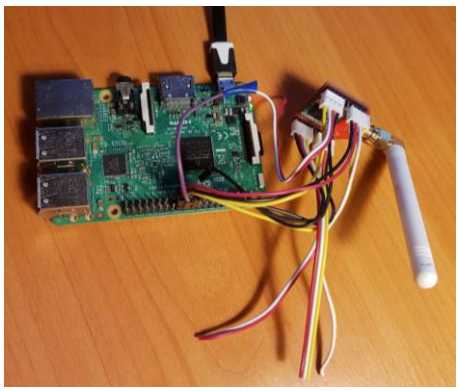

Рисунок 24 – Внешний вид подключения модуля LoRa BEE к Raspberry Pi 3 Model B

4.3 Описание диаграммы классов

При проектировании ПО для вычислительного хаба использовался объектно-ориентированный подход, т.к. при таком подходе ускоряется время разработки и программный продукт легко изменять и добавлять новый функционал. На рисунке Е.15 показана разработанная диаграмма классов.

«*Worker*» является главным классом, отвечающий за инициализацию, а также периодическим опросом драйвера (класс «*Driver*») в методе «*run()*» для приёма данных с радиомодуля LoRa или с последовательного порта. С помощью «*QTimer*» с заданной периодичностью вызывается метод «*run()*».

Абстрактный класс «*Driver*» является базовым классом для «*LoRaConnection*» и «*SerialConnection*» и является интерфейсом для взаимодействия с датчиками МКПРСПМ. Потомки данного класса реализовывают подключение и приём-передачу данных с МКПРСПМ, определяя методы «*send()*», «*available()*» и «*recv()*». Таким образом, реализуется паттерн «Стратегия», тем самым внутри «*Worker*» в зависимости от настроек выбирается способ подключения с датчиками МКПРСПМ: радиомодуль LoRa или последовательный порт UART. Принимая пакет, драйвер определяет тип сообщения и отправляет его на соответствующий обработчик.

Класс «*Server*» реализует интерфейс взаимодействия с сервером по протоколу MQTT. При инициализации он подключается к MQTT брокеру сервера и подписывается на соответствующие темы. Приняв сообщение от сервера, экземпляр класса по подписке определяет соответствующий обработчик.

Абстрактный класс «*AbstractReceiveMessageHandler»* является обработчиком входящих сообщений от датчиков МКПРСПМ и сервера. Данный класс реализует потомок «*ReceiveMessageHandler».*

4.4 Описание файла настроек

При первом запуске приложения генерируется файл настроек «*settings.ino*», в котором можно поменять способ подключения к датчикам МКПРСПМ (LoRa или UART), перечислить список идентификаторов датчиков, настроить подключение к серверам и т.д. Ниже (Листинг 1) приводится пример настроечного файла «*settings.ino*» с комментариями, в которых описан каждый параметр:

Листинг 1 – Файл «settings.ino»

```
[General]
# Способ подключения к датчикам МКПРСПМ (lora, serial)
driver=lora
# Частота радиомодуля при подключении по lora
frequency=868
# Timeout при отправки данных по lora
lora_timeout=1000
# Последовательный порт при подключении serial
serial_port=ttyUSB0
# Baud rate для последовотельного порта
baud_rate=9600
```

```
# Список идентификаторов датчиков МКПРСПМ
sensors=1, 2
# Список серверов и папраметров подключения к ним
# host - хост MQTT брокер сервера
# mqtt_port - порт MQTT брокер сервера
# mqtt_username - логин от MQTT брокер сервера
# mqtt_password - пароль от MQTT брокер сервера
# login - логин владельца парковки
# password - пароль владельца парковки
servers="[\
      {\
            \"host\":\"0.0.0.0\",\
            \"mqtt_port\":1883,\
            \"mqtt_username\":\"\",\
            \"mqtt_password\":\"\",\
            \"login\":\"user1@mail.ru\",\
            \"password\":\"pwd123456\"\
      }, {\
            \"host\":\"m23.cloudmqtt.com\",\
            \"mqtt_port\":16325,\
            \"mqtt_username\":\"mqtt_username\",\
            \"mqtt_password\":\"Pwd123456\",\
            \"login\":\"user1@gmail.com\",\
            \"password\":\"123456\"\
      }\
]"
# Частота опроса датчиков
idle_period=300
```

Обратите внимание, что параметр «*servers*» описывается в виде массива, таким образом, можно подключить несколько MQTT брокер серверов, и от каждого из них хаб будет принимать команды и отправлять каждому данные от датчиков МКПРСПМ.

5 Разработка сервера

5.1 Анализ задания и выбор технологий

Как показал анализ технического задания, ПО на серверной стороне должен реализовывать следующий функционал:

- отображение на карте местоположения стоянок и парковочных мест;
- выдача актуальных данных о ПМ;
- бронирование парковочного места;
- добавление датчиков МКПРСПМ;
- удалённая настройка датчиков МКПРСПМ;
- регистрирование стоянок и парковочных мест с указанием местоположения на карте;
- изменение данных о стоянках и парковочных местах;

- приём MQTT пакетов данных о состоянии ПМ, полученные от вычислительного хаба;
- сохранение данных об изменении состоянии ПМ в БД;
- регистрация и авторизация пользователей.

Для создания данного ПО для серверной части был использован полноценный, многоуровневый фреймворк, использующий базы данных, Ruby on Rails, который основан на архитектуре Модель-Представление-Контроллер (Model-View-Controller, MVC). Обработка запросов и выдача данных в контроллерах, предметная область, отраженная в базе данных, – для всего этого Rails предоставляет однородную среду разработки на Ruby. Основным преимуществом языка программирования Ruby и фреймворка Ruby on Rails считается скорость разработки. Практика показывает, что скорость разработки проектов на RoR увеличивается на 30 – 40 процентов по отношению к любому другому языку программирования или фреймворку. В первую очередь прирост скорости разработки определяется обширным набором готовых к работе штатных инструментов RoR, колоссальным набором готовых решений в сообществе, языком Ruby и простоте программирования на нем. Кроме того, в отличие от других фреймворков, в составе RoR есть отличные средства автоматизированного тестирования, что ускоряет переход проекта от стадии «программа написана» к стадии «программа работает без ошибок». Так же следует отметить, что Ruby on Rails обеспечивает лучшую безопасность проекта. При использовании инструментов RoR исключены SQL-инъекции и XSS-атаки, все входные параметры экранируется по умолчанию, выводимые переменные в шаблонах также экранируются.

Согласно анализу требований, на серверной стороне используется следующее ПО:

- Разработанное веб-приложение на фреймворке Ruby on Rails.
- Frontend-сервер Nginx, который осуществляет раздачу статических файлов и проксирование запросов на веб-сервер Puma, взаимодействующий с веб-приложением на Ruby on Rails.
- MQTT-брокер Mosquitto необходим для реализации MQTT протокола. На брокера приходят данные с вычислительных хабов, а веб-приложение их считывает.
- Сервис кэширования Memcached используется веб-приложением для кэширования данных.
- СУБД PostgreSQL используется для хранения данных.
- СУБД Redis необходим для работы Action Cable, позволяющий взаимодействовать с веб-клиентами по WebSocket.

- Фреймворк Bootstrap v4.1 – это инструментарий с открытым исходным кодом для разработки с помощью HTML, CSS и JS. С помощью него создаётся адаптивный и интерактивный веб-дизайн.
- Google Maps APIs для реализации карты парковок.

5.2 Анализ способа хранения данных

Существует несколько технологий позволяющих хранить и упорядочивать информацию. Существует множество систем управления базой данных (СУБД), проанализировав их, я пришёл к выводу, что наиболее лучшим вариантом будет использование PostgreSQL, т.к. оно обладает следующими преимуществами: поддержка БД неограниченного размера, мощные и надёжные механизмы транзакций и репликации, расширяемая система встроенных языков программирования, наследование, легкая расширяемость. Также для работы с картографическими данными имеется расширение PostGIS.

На рисунке Е.16 показана разработанная схема базы данных. В таблице «users» находятся зарегистрированные пользователи системы.

В таблице «parkings» хранятся стоянки. В поле «area» находится массив координат полигона, который образует территория стоянки. Это необходимо для отрисовки на карте синего полигона, обозначающего территорию стоянки (Рисунок 25). Поле «parking_places_count» добавлено с целью оптимизации выдачи количества парковочных мест, принадлежащих данной стоянке.

Таблица «sensors» хранит добавленные датчики регистрации свободных парковочных мест. При добавлении нового датчика для него генерируется уникальный идентификатор «id», который необходимо будет указать самому физическому датчику.

В таблице «parking_places» находятся парковочные места. Поля «booked», «free» и «connected» отвечают за текущее состояние ПМ. Поле «can_book» отвечает за возможность онлайн бронирования этого ПМ.

В таблице «parking_states» хранится история изменения состояний парковочного места. Таблица служит для сбора статистических данных. При приходе одинокого состояния с прошлым новая запись не добавляется, а обновляется поле «updated_at».

Таблица «orders» служит для хранения совершенных заказов пользователей на бронирование парковочного места. В поле «cost» хранится суммарная стоимость заказа, в поле «payment» – внесенная сумма пользователем. Поле «booked_time» хранит время в секундах на сколько было забронировано ПМ, а «end_time» – время окончания брони. При добавлении

нового заказа поле «active» по умолчанию устанавливается в значение TRUE, что означает, что заказ в данный момент активен, а после окончании брони поле «active» устанавливается в значение FALSE.

5.3 Формы интерфейса

Для верстки страниц веб-приложения применялся фреймворк Bootstrap v4.0. Данный фреймворк имеет множество стилей для множества компонентов, с помощью которых можно быстро реализовать адаптивный, интерактивный дизайн проекта. Таким образом, используя данный фреймворк обеспечивается правильное отображение веб-страниц на различных устройствах.

Также для реализации карты, отображающей парковки, использовался Google Maps APIs. Данное API очень удобное, имеет хорошую документацию и позволяет реализовывать интерактивные карты, которые хорошо будут отображаться на различных устройствах.

На рисунке 25 показана главная странице «/», на которой пользователю предоставляется карта с указанием местоположений парковок в данной области. На карте синими областями показана территория стоянки, а красными маркерами – парковочные места. В правой части находится кнопка «Меню» при нажатии на которую выдвигается боковая панель с элементами управления для фильтрации содержимого карты. Более подробнее о формах интерфейсов смотри приложение Б.

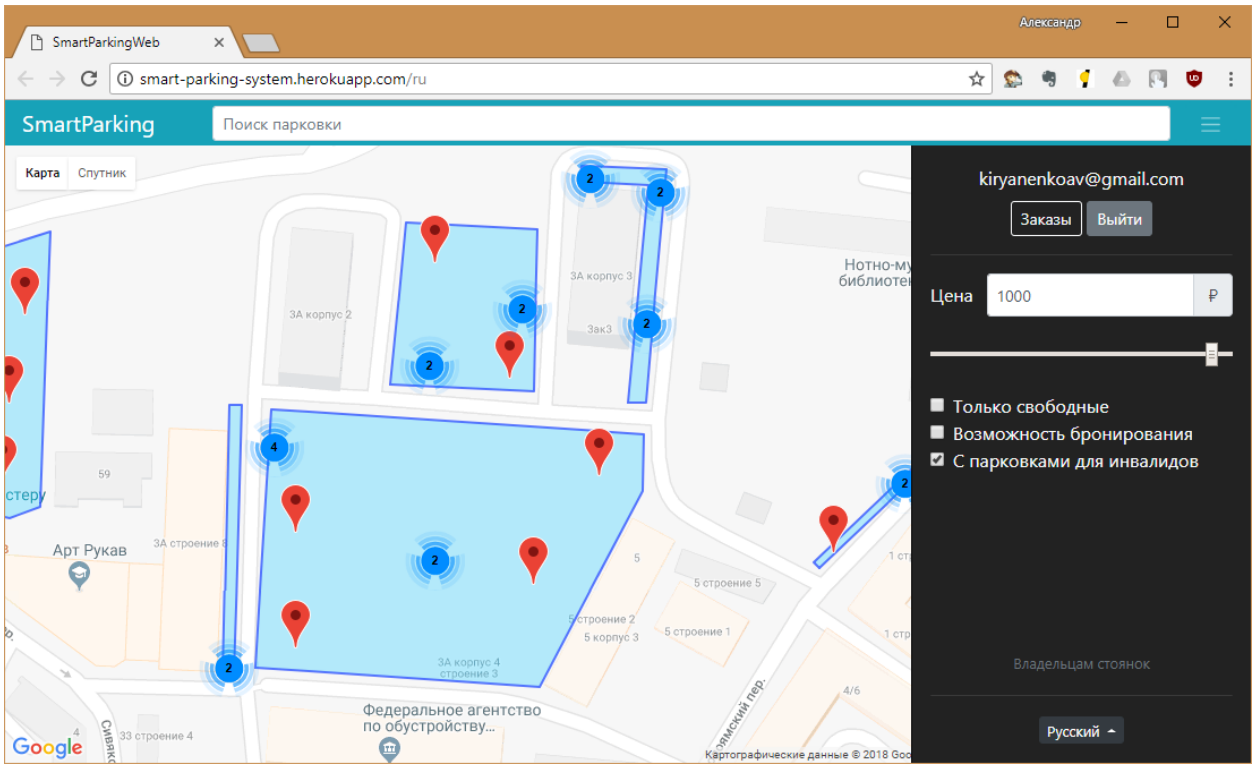

Рисунок 25 – Главная страница

5.4 Алгоритм выдачи парковочных мест

В Ruby on Rails механизм Action Cable с легкостью интегрирует WebSockets с остальными частями веб-приложения. Он позволяет писать функциональность реального времени и является производительным и масштабируемым. Он представляет полный стек, включая клиентский фреймворк на JavaScript и серверный фреймворк на Ruby. Этот механизм использовался для реализации выдачи в реальном времени состояний парковочных мест на главной странице «/» веб-приложения.

Веб-клиент на главной странице «/» создаёт WebSocket соединение и подключается к каналу «*MapChannel*» и для выдачи парковочных мест в заданной области передаёт следующие параметры запроса: координаты местоположения, радиус и список фильтров поиска. Сервер нового клиента подписывает на соответствующий поток (stream), по которому периодически высылаются состояния парковочных в заданной области.

В БД для оптимизации поиска в таблицах на всех полях, по которым осуществляется поиск были навешаны соответствующие BTREE индексы. А также для пространственных координат в таблицах парковочных мест и стоянок были созданы GIST индексы.

Также для оптимизации производительности карта была поделена на взаимно пересекающиеся квадраты. Число точек, по которым могут быть построены квадраты, ограничено и размеры самих квадрат также ограничены и определяются, исходя из масштаба, определяемого по переданному радиусу. Таким образом, количество таких квадратов ограничено и, следовательно, исчезает уникальность в запросах пользователей, клиентов можно подписывать на общие потоки и появляется возможность кэширования запросов.

Масштаб s вычисляется по формуле:

$$s = ceil\left(\log_2 \frac{2 * r_{\text{исх}}}{a_{min}}\right), \quad (1)$$

где, $r_{\text{исх}}$ – радиус из параметра запроса:

a_{min} – константа, минимальная сторона квадрата, равная 0,01.

Сторона квадрата a определяется формулой:

$$a = a_{min} * 2^s, \quad (2)$$

Для расчета координат центра квадрата потребуются промежуточное вычисление x_a и y_a – координаты с шагом равным стороне:

$$\begin{cases} x_a = floor\left(\dfrac{x_{\text{исх}}}{0{,}5 * a}\right) \\ y_a = floor\left(\dfrac{y_{\text{исх}}}{0{,}5 * a}\right) \end{cases}, \tag{3}$$

где, $x_{\text{исх}}$ и $y_{\text{исх}}$ – координаты местоположения из параметра запроса.

Далее находятся координаты четырёх квадратов $x_{\text{кв}}$, $x'_{\text{кв}}$ и $y_{\text{кв}}$, $y'_{\text{кв}}$ вокруг исходных координат $x_{\text{исх}}$ и $y_{\text{исх}}$. И требуется определить координаты x и y квадрата ближайшего к исходным координатам:

$$\begin{cases} x = \left[\begin{array}{l} x_{\text{кв}} = \dfrac{a * x_a}{2} \\ x'_{\text{кв}} = \dfrac{a * (x_a + 1)}{2} \end{array}\right. \\ y = \left[\begin{array}{l} y_{\text{кв}} = \dfrac{a * y_a}{2} \\ y'_{\text{кв}} = \dfrac{a * (y_a + 1)}{2} \end{array}\right. \end{cases} \tag{4}$$

На рисунке 26 приводится схема алгоритма подключения нового клиента к каналу «*MapChannel*» для периодической выдачи в реальном времени актуальных данных о состоянии парковочных мест.

Было проведено нагрузочное тестирование для данного алгоритма: при заполненной БД (13582 записи) RPS составил 846 запросов в секунду (более подробнее см. Приложение В).

После формирования подписки клиента заданный квадрат добавляется в список обслуживаемых квадратов. Отдельный поток в бесконечном цикле для каждого квадрата получает данные о состоянии парковочных мест и рассылает их в соответствующем потоке (stream). В листинге 2 приводится класса «MapService», который осуществляет данное действие.

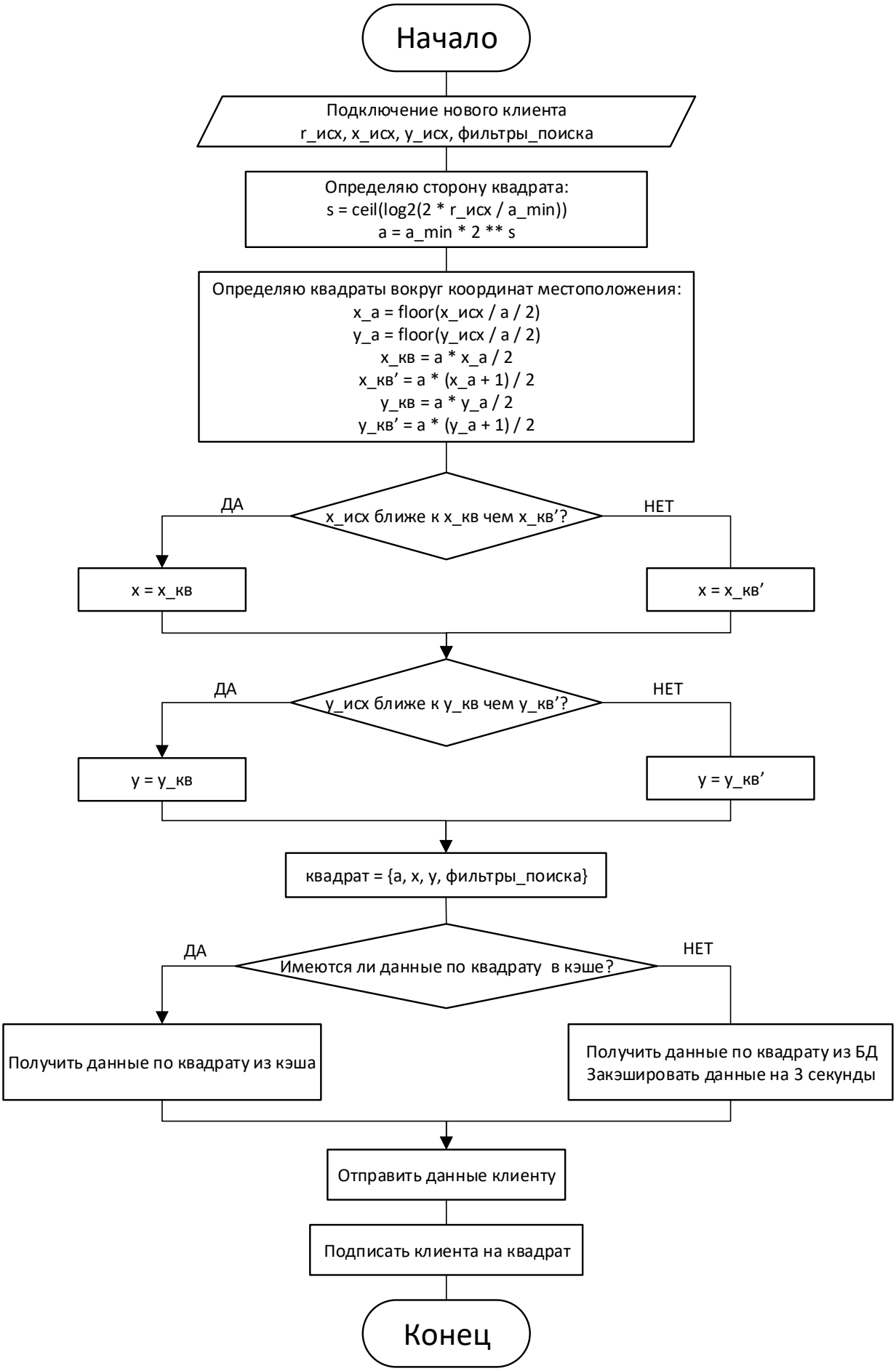

Рисунок 26 – Схема алгоритма подключения нового клиента к каналу «MapChannel»

Листинг 2 – Файл «/app/services/map_service.rb»

```
require 'singleton'

class MapService
  include Singleton

  def initialize
    @map_clients = Hash.new
    @squares = Hash.new
    @squares_m = Mutex.new
    run
  end

  def add_square(square)
    @squares_m.synchronize do
      if @squares.has_key? square.stream
        sq = @squares[square.stream]
        sq[:count] += 1
      else
        @squares[square.stream] = {square: square, count: 1}
      end
    end
  end

  def remove_square(square)
    @squares_m.synchronize do
      sq = @squares[square.stream]
      sq[:count] -= 1
      if sq[:count] <= 0
        @squares.delete square.stream
      end
    end
  end

  private
  def run
    Thread.new do
      loop do
        before = Time.now

        begin
          values = []
          @squares_m.synchronize { values = @squares.values.dup }
          values.each do |value|
            value[:square].broadcast
          end
          ParkingPlace.unset_changed
        rescue Exception => e
          Rails.logger.error e.message
          Rails.logger.error e.backtrace
        end

        sleep_time = Rails.configuration.map_sending_period - (Time.now -
before)
        sleep sleep_time if sleep_time > 0
      end
    end
  end
end
```

5.5 Автоматизация развертывания серверного ПО

Для автоматизации развертывания ПО на боевых серверах используется среда виртуализации Docker, а также Docker-compose – инструмент для запуска многоконтейнерных приложений. Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux -систему, а также предоставляет среду по управлению контейнерами. Таким образом, для разворачивания серверного ПО достаточно чтобы на сервере с ОС семейства Linux были установлены среда Docker и инструмент Docker-compose.

Ниже (Листинг 3) представлен Dockerfile – файл с инструкцией по разворачиванию Docker-контейнера веб-приложения Ruby on Rails:

Листинг 3 – Файл «Dockerfile»

```
FROM ruby:2.3
MAINTAINER Kiryanenko Alexander <KiryanenkoAV@gmail.com>

RUN apt-get update &&\
    apt-get install -y \
        build-essential \
        nodejs \
        npm \
        postgresql-contrib \
        git &&\
    npm install yarn -g

RUN mkdir /app
WORKDIR /app

COPY Gemfile ./
RUN bundle install --jobs 20

COPY . .

# Setting env up
ENV RAILS_ENV 'production'
ENV RAKE_ENV 'production'
ENV SECRET_KEY_BASE $(rake secret)

RUN bundle exec rake assets:precompile

EXPOSE 3000

CMD rails db:migrate && bundle exec puma -C config/puma.rb
```

В файле «docker-compose.yml» (Листинг 4) описываются запускаемые контейнеры: веб-приложения, СУБД PostgreSQL, СУБД Redis, MQTT-брокера Mosquitto, сервис Memcached и сервера Nginx.

Листинг 4 – Файл «docker-compose.yml»

```
version: '2'
services:
  postgres:
    image: mdillon/postgis:9.6-alpine
    environment:
      POSTGRES_USER: smartparking
      POSTGRES_PASSWORD: 123456
    ports:
      - 5432:5432
    volumes:
      - postgres:/var/lib/postgresql/data
    restart: always

  redis:
    image: redis
    command: redis-server
    ports:
      - 6379:6379
    volumes:
      - redis:/data
    restart: always

  mqtt:
    image: ansi/mosquitto
    ports:
      - 1883:1883
    restart: always

  memcached:
      image: memcached:alpine
      ports:
        - 11211:11211

  web:
    depends_on:
      - postgres
      - redis
      - mqtt
      - memcached
    volumes:
      - public:/app/public
      - tmp:/tmp
    build: .
    ports:
      - 3000:3000
    env_file: .env
    environment:
      RAILS_ENV: production
      REDIS_URL: redis://redis:6379/1
      MQTT_URI: mqtt://mqtt
      DB_HOST: postgres
    links:
      - postgres
      - redis
      - mqtt
      - memcached
    restart: always

  nginx:
    image: nginx:alpine
    volumes_from:
      - web
```

```
    volumes:
      - ./config/nginx.conf.erb:/etc/nginx/nginx.conf
      - nginx_logs:/etc/nginx/logs/nginx
    depends_on:
      - web
    ports:
      - 80:80
    links:
      - web
    restart: always

volumes:
  redis:
  postgres:
  public:
  tmp:
  nginx_logs:
```

Для развертывания серверного ПО достаточно вбить следующую команду:

```
$ docker-compose up
```

5.6 Балансировка нагрузки

Балансировка нагрузки необходима в случаях, когда один-единственный сервер не справляется с возложенными на него задачами в силу возросшего количества запросов. В Nginx роль балансировщика реализует модуль HttpUpstreamModule. С помощью облачной платформы Google Cloud была реализована инфраструктура (Рисунок 27), состоящая из сервера для балансировки нагрузки между 3-мя backend-серверами, на которых находится само веб-приложение.

Технические характеристики виртуальных машин:

- Количество ядер: 1
- ОЗУ: 3,75 Гб

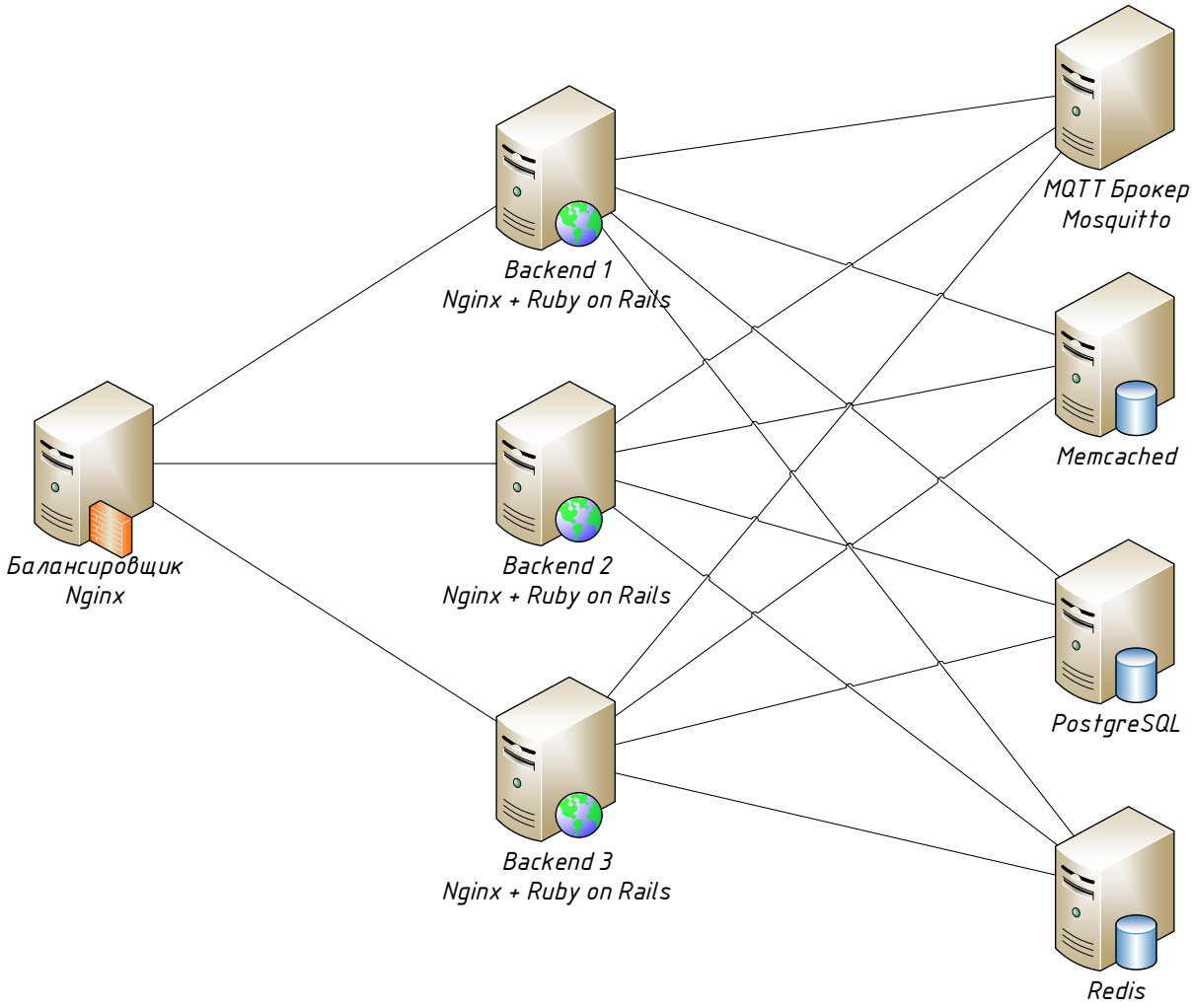

Рисунок 27 – Принципиальная схема сети

Ниже (Листинг 5) показан конфигурационный файл Nginx для балансировщика нагрузки:

Листинг 5 – Файл «nginx-balancer.conf»

```
upstream backend {
  ip_hash;
  server 10.142.0.4 max_fails=3 fail_timeout=10s;
  server 10.142.0.5 max_fails=3 fail_timeout=10s;
  server 10.142.0.6 max_fails=3 fail_timeout=10s;
}

server {
  listen 80;
  server_name 0.0.0.0;
  root /app/public;
  allow all;

  location /nginx_status {
    stub_status;
  }

  location /assets/ {
    error_page 404 = @store;
    expires max;
  }
```

```
  location @store {
    proxy_store on;
    proxy_store_access user:rw group:rw all:r;
    proxy_pass http://backend;
    proxy_set_header X-Real-IP  $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto http;
    proxy_set_header Host $http_host;
  }

  location / {
    proxy_pass http://backend;
    proxy_next_upstream error timeout invalid_header http_502;
    proxy_set_header X-Real-IP  $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto http;
    proxy_set_header Host $http_host;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_redirect off;
  }
}
```

С помощью мониторингового сервиса Datadog была продемонстрирована балансировка нагрузки с симуляцией падения одного backend-сервера во время нагрузочного тестирования (рисунки 28, 29). Как можно видеть при падении одного из backend-серверов (фиолетовый график) балансировщик (синий график) распределяет нагрузку между оставшимися backend-серверами (зеленый, желтый график).

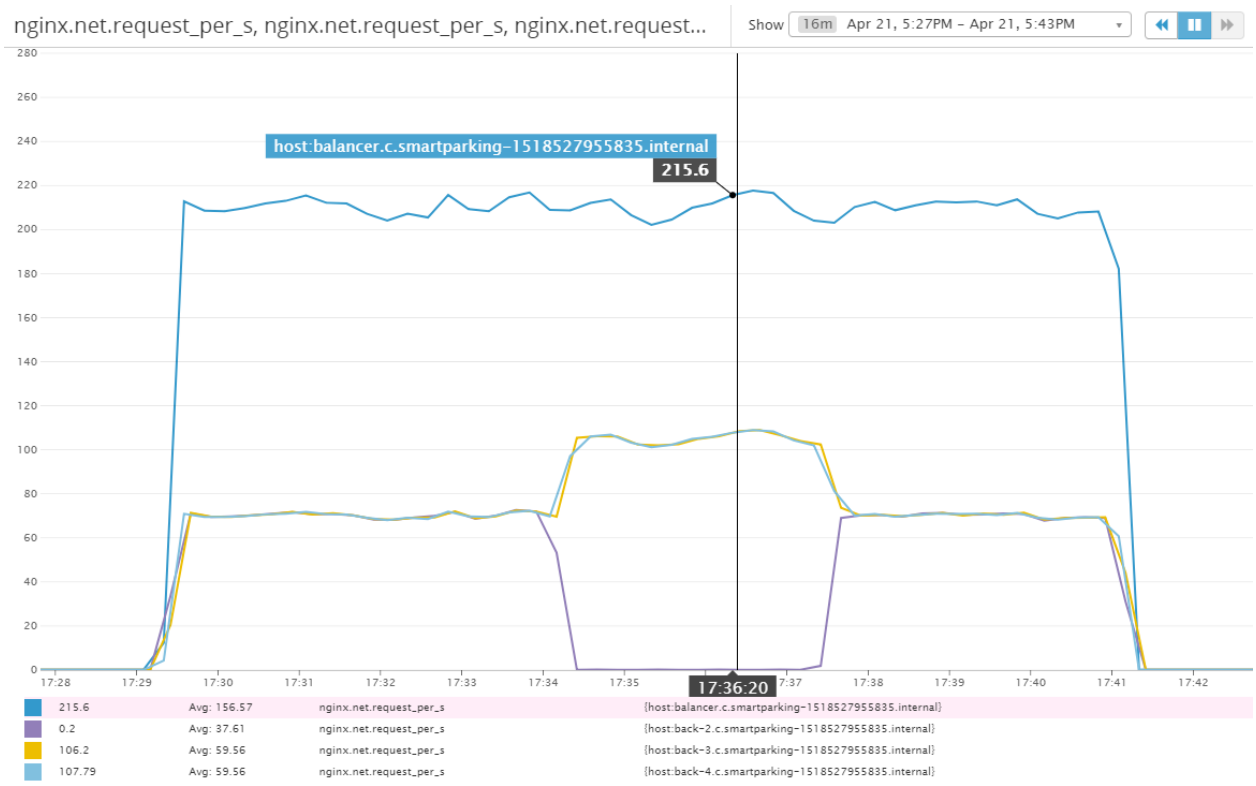

Рисунок 28 – График RPS на сервере балансировщика (синий) и backend-серверах

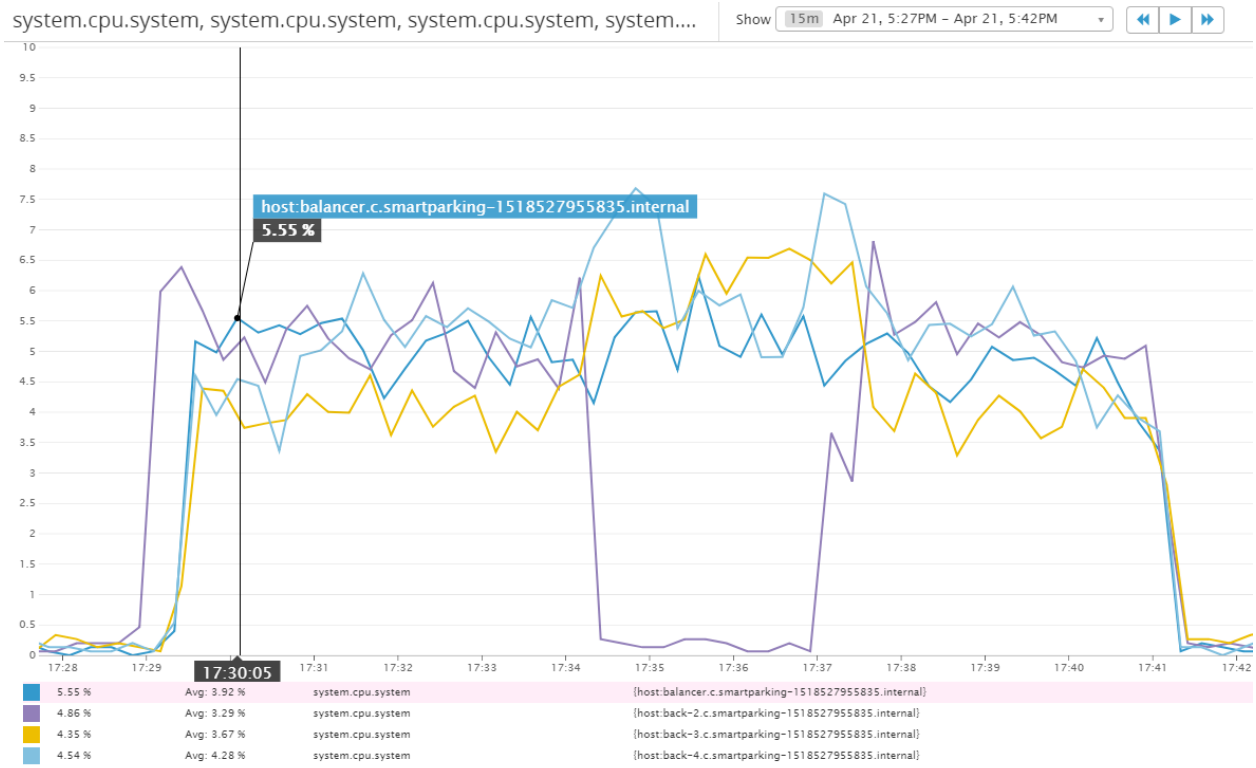

Рисунок 29 – График нагрузки CPU на сервере балансировщика (синий) и backend-серверах

6 Обеспечение безопасности в СМПМ

6.1 Обеспечение защищённой передачи данных между узлами

TLS — [14] криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет. TLS использует асимметричное шифрование для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности сообщений для сохранения целостности сообщений. TLS даёт возможность клиент-серверным приложениям осуществлять связь в сети таким образом, что нельзя производить прослушивание пакетов и осуществить несанкционированный доступ.

Таким образом, для обеспечения безопасности при передаче между узлами системы необходимо использовать протоколы, основанные на протоколе TLS. Так при передаче MQTT пакетов между вычислительным хабом и сервером следует использовать протокол MQTTS. А осуществление взаимодействия между веб-клиентом и сервером – по защищенному протоколу HTTPS.

6.2 Защита от межсайтового скриптинга (XSS)

Межсайтовый скриптинг (XSS) – вредоносная атака, которая внедряет на стороне клиента исполняемый код. Ruby on Rails предоставляет методы для защиты от этих атак [15]. Атаки XSS работают подобным образом: злоумышленник встраивает некоторый код, веб-приложение сохраняет его и отображает на странице, после чего представляет его жертве. XSS может украсть Cookie, угнать сессию, перенаправить жертву на фальшивый веб-сайт, отобразить рекламу, полезную злоумышленнику, изменить элементы на веб-странице, чтобы получить конфиденциальную информацию или установить вредоносное программное обеспечение, используя дыры в веб-браузере.

Для защиты от XSS осуществляется фильтрация ввода с помощью белого списка, а не черного. Фильтрация белым списком устанавливает допустимые значения, остальные значения недопустимы. Черные списки всегда не законченные.

Также для защиты экранируется весь вывод в приложении, особенно при отображении пользовательского ввода, который не был отфильтрован при вводе. Используется встроенный метод *escapeHTML()* (или его псевдоним *h()*), чтобы заменить введенные символы HTML *&, ", <, >* их неинтерпретируемыми представителями в HTML *(&, ", < и >)*.

6.3 Защита от SQL-инъекции

Цель атаки в форме SQL-инъекции – сделать запросы к базе данных, манипулируя с параметрами приложения. Атака может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах.

В Ruby on Rails есть встроенный фильтр для специальных символов SQL, который экранирует *' , "* , символ *NULL* и разрыв строчки. Использование ORM автоматически применяет эту контрмеру. В фрагментах SQL, например в фрагментах условий *where("...")*, это применяется вручную. Вместо передачи строки в опцию conditions, передается массив или хэш, чтобы экранировать испорченные строки. [15]

6.4 Защита от межсайтовой подделки запрос (CSRF)

Межсайтовая подделка запроса (CSRF, Cross-Site Request Forgery) – метод атаки, которая работает через включение вредоносного кода или ссылки на страницу, которая обращается к веб-приложению, на котором предполагается, что пользователь

аутентифицирован. Если сессия для того веб-приложения не истекла, злоумышленник сможет выполнить несанкционированные команды.

Причина CSRF кроется в том, что браузеры не понимают, как различить, было ли действие явно совершено пользователем (как, скажем, нажатие кнопки на форме или переход по ссылке) или пользователь неумышленно выполнил это действие (например, при посещении *bad.com*, ресурсом был отправлен запрос на *good.com/some_action*, в то время как пользователь уже был авторизован на *good.com*).

Для защиты от CSRF атаки необходимо надлежащим образом использовать GET и POST (DELETE, PUT и PATCH должны использоваться как POST) запросы. А также использовать токен безопасности в не-GET-запросах.

Веб-приложение СМПМ построено согласно REST принципам, таким образом, GET запросы являются безопасными: они предназначены только для получения информации и не должны изменять состояние сервера. Для защиты от остальных подделанных запросов, Ruby on Rails представляет [15] обязательный токен безопасности, который знает наш сайт, но не знают остальные. Rails автоматически включает токен безопасности во все формы и Ajax запросы. Если токен безопасности не будет соответствовать ожидаемому, то будет вызвано исключение.

Необходимо отметить, что уязвимости межсайтового скриптинга (XSS) обходят все защиты от CSRF. XSS дает злоумышленнику доступ ко всем элементам на странице, поэтому он может прочитать токен безопасности CSRF из формы или непосредственно подтвердить форму (п.п. 6.2. Защита от межсайтового скриптинга).

7 Тестирование СМПМ

Для проверки требований, предъявленных в техническом задании, в разработанной системе мониторинга парковочных мест были протестированы все компоненты системы, а также было проведено комплексное тестирование системы (Приложение В). Для тестирования МКПРСПМ был собран отладочный макет, внешний вид которого изображен на рисунке 30.

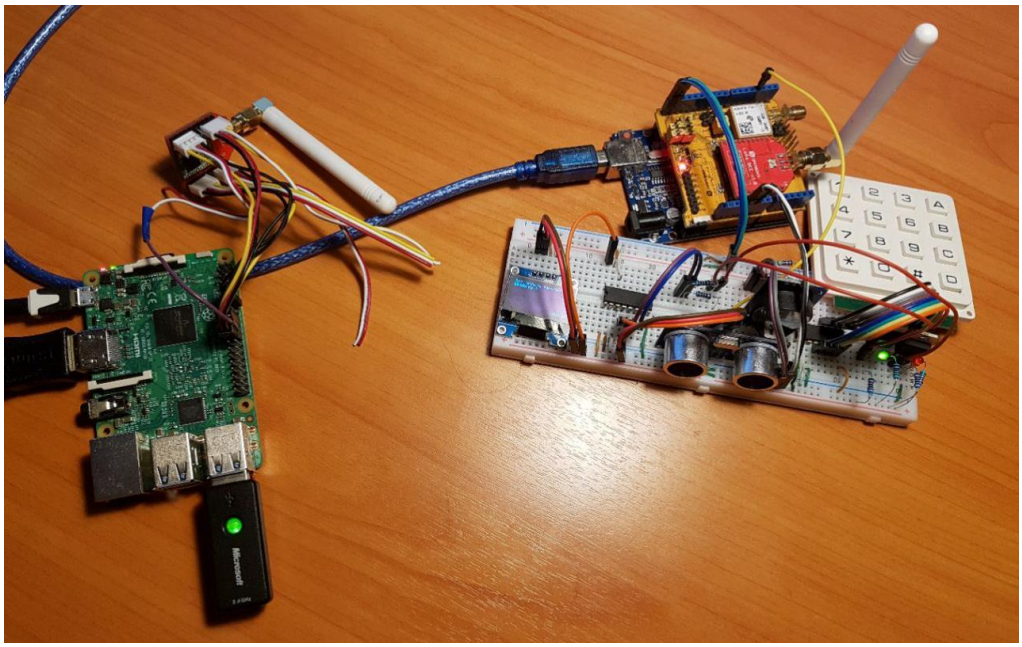

Рисунок 30 – Внешний вид отладочного макета

Также для проверки и анализа данных работоспособности при большой нагрузки было проведено нагрузочное тестирование. Так, например, RPS для выдачи парковочных мест (п.п. 5.4) при заполненной БД (13582 записи) составила 846 запросов в секунду (более подробнее см. в Приложение В).

Для обеспечения качества в ходе дальнейшей разработки для веб-приложения были написаны 60 автоматизированных теста, покрывающие весь функционал. Из них 14 модульных теста, 6 интеграционных теста и 40 функциональных тестов контроллеров.

Результатом проведенного тестирования было установлено, что были выполнены все предъявляемые требования к разработанному прототипу системы мониторинга парковочных мест.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы бакалавра был спроектирован и реализован аппаратно-программный прототип «Системы мониторинга парковочных мест». Такая система для водителей будет в режиме реального времени предоставлять актуальные данные о состоянии парковочных мест. Данная система состоит из МК-подсистемы регистрации свободных парковочных мест, вычислительного хаба и центрального сервера.

Программное обеспечение для компонентов системы написано преимущественно в объектном стиле, который обеспечивает интеграцию и удобство для дальнейшей разработки командой разработчиков.

В ходе выполнения работы было проведено исследование IoTтехнологий в транспортной инфраструктуре, а также анализ существующих систем. Разработанная система экономически более выгодна относительно конкурентных систем.

Все компоненты системы были отлажены и протестированы. Для разработанного серверного ПО были написаны автоматизированные тесты для обеспечения качества в ходе дальнейшей разработки. Также был проведен анализ безопасности системы. Таким образом, разработанная система соответствует всем требованиям, заявленных в техническом задании.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Рекомендации МСЭ-TY.2060. Обзор интернета вещей.

2 Интернет вещей: учебное пособие [текст] / Росляков А.В., Ваняшин С.В., Гребешков А.Ю. – Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015 – 136 с.

3 Открытый студенческий конкурс (хакатон) по-быстрому прототипированию решений Интернета вещей [Электронный ресурс] – URL: https://bmstu-hackathon.github.io/2016/ (дата обращения 12.11.2017).

4 Что такое LORA? [Электронный ресурс]. – URL: http://lo-ra.ru/lora/ (дата обращения 10.11.2017).

5 «Умные» среды, «умные» системы, «умные» производства // серия докладов (зеленых книг) в рамках проекта «Промышленный и технологический форсайт Российской Федерации» – Москва, Санкт-Петербург, 2012. 63 с.

6 Яндекс.Парковки [Электронный ресурс] – URL: https://yandex.ru/support/parking/ (дата обращения 12.11.2017).

7 Московский паркинг [Электронный ресурс] – URL: http://parking.mos.ru (дата обращения 12.11.2017).

8 Патент РФ № 130334, 20.07.2013

9 Патент РФ № 103120, 27.03.2011

10 Датчики присутствия. [Электронный ресурс]. – URL: https://hran.im/umnyiy-dom/datchiki/datchiki-prisutstviya.html (дата обращения 10.11.2017).

11 Документация на микроконтроллер ATMEGA328P [Электронный ресурс]. – URL: http://www.atmel.com/ru/ru/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATMEGA328-328P_Datasheet.pdf (дата обращения 10.11.2017).

12 PCF8574. Дистанционный 8-битный расширитель ввода – вывода I2C-шины. [Электронный ресурс]. – URL: http://www.gaw.ru/html.cgi/txt/ic/Philips/interfaces/iic/pcf8574.htm (дата обращения 10.11.2017).

13 Ультразвуковой датчик HC-SR04. [Электронный ресурс]. – URL: https://arduino-kit.ru/userfiles/image/HC-SR04%20_.pdf (дата обращения 04.11.2017).

14 TLS. [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/TLS (дата обращения 04.04.2018).

15 Безопасность приложений на Rails. [Электронный ресурс]. – URL: http://rusrails.ru/ruby-on-rails-security-guide (дата обращения 04.04.2018).