МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ ПОЗИЦИОНИРОВАНИЯ ВНУТРИ ПРОИЗВОДСТВЕННЫХ ПОМЕЩЕНИЙ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.308/249 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____П.О. Шабуров
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____И.А. Рябухин
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М.А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Д.В. Топольский
«__»_____2022 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Рябухину Игорю Андреевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Разработка серверной части системы позиционирования внутри производственных помещений» утверждена приказом по университету от 12 декабря 2021 г. № 308/141.

2. Срок сдачи студентом законченной работы: 1 июня 2022 г.

3. Исходные данные к работе:

Обеспечить основной функционал приложения:

– установление связи с аппаратной частью системы позиционирования посредством определенных интерфейсов;

– установление связи с клиентской частью системы позиционирования;

– настройка аппаратной части системы позиционирования;

– возможность единовременной работы с несколькими клиентскими приложениями.

4. Перечень подлежащих разработке вопросов:

– анализ существующих программных продуктов, используемых для позиционирования объектов внутри помещений;

– анализ современных технологий, решений и методов для реализации серверного приложения, разработка структуры программного продукта;

– реализация собственного программного обеспечения, обеспечивающего взаимодействие с аппаратной и клиентской частями системы позиционирования внутри помещений;

– тестирование и отладка реализованного программного продукта согласно выбранным методологиям и критериям качества.

5. Дата выдачи задания: 1 декабря 2021 г.

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*И.А. Рябухин*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы (анализ существующих программных продуктов и решений, используемых для решения задач позиционирования объектов внутри помещений)	07.02.2022	
Разработка модели, проектирование (анализ современных технологических решений в разработке серверного программного обеспечения, проектирование структуры приложения)	07.03.2022	
Реализация системы (разработка модулей API, модулей сетевого взаимодействия, модулей взаимодействия с аппаратной частью системы позиционирования внутри помещений)	04.04.2022	
Тестирование, отладка, эксперименты (отладка реализованного программного обеспечения согласно выбранным методам и стандартам тестирования)	10.05.2022	
Компоновка текста работы и сдача на нормоконтроль	26.05.2022	
Подготовка презентации и доклада	03.06.2022	

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*И.А. Рябухин*/

АННОТАЦИЯ

И.А. Рябухин. Разработка серверной части системы позиционирования внутри производственных помещений. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 41 с., 10 ил., библиогр. список – 16 наим.

В рамках выполнения выпускной квалификационной работы пройдены все этапы проектирования и разработки серверной части программного обеспечения для системы позиционирования внутри производственных помещений. Проанализированы недостатки имеющегося на предприятии программного обеспечения. На основании сравнительного анализа выбраны технологические решения для реализации серверного приложения. Выбран шаблон проектирования архитектуры приложения и определена структура серверного приложения. На основе описанной архитектуры с использованием выбранных технологических решений была разработана серверная часть программного обеспечения системы позиционирования. После разработки было произведено тестирование программного обеспечения согласно выбранным методологиям.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

У предприятия ООО «ДСТ-УРАЛ» (далее предприятие) [1] возникла потребность в системе позиционирования внутри помещений. Целью внедрения данной системы в бизнес-процессы предприятия является оптимизация производственных и логистических процессов [2].

В сфере позиционирования объектов внутри помещений количество различных доступных систем крайне ограничено. В основном системы используют технологии Bluetooth, Bluetooth Low Energy (BLE), Wi-Fi. Ввиду технических ограничений данные системы не имеют высокой точности позиционирования, что не позволяет использовать их для отслеживания перемещений транспортных средств в пределах производственных или же складских помещений. Заказчиком была выбрана система «Marvelmind» [3].

Программное обеспечение, поставляемое вместе с данным аппаратным комплексом, не удовлетворяло всем функциональным требованиям заказчика, в следствие чего предприятием было составлено техническое задание, на основе которого была спроектирована и реализована серверная часть программно-аппаратного комплекса.

Целью всей выпускной квалификационной работы является проектирование и реализация программного продукта, соответствующего функциональным требованиям, предоставленных предприятием. Создание серверного приложения должно обеспечить оптимизацию производственного процесса на этапах транспортировки грузов: запчастей, материалов, инструмента. Для корректной и полноценной работы всей системы требуется спроектировать программный интерфейс приложения (англ. Application Programming Interface, далее API) [4], описывающий все возможные методы и типы данных, которые может запросить клиентская часть системы. Также

требуется описать взаимодействие с аппаратной частью системы позиционирования посредством проводного последовательного интерфейса UART [11] и беспроводного интерфейса стандарта IEEE 802.11 (далее Wi-Fi) [5]. Программа должна реализовывать все функции, доступные контроллеру системы позиционирования и описанные в открытой документации для разработчиков [6].

Перечень данных функций включает в себя:

– получение координат в миллиметрах;

– настройка главного контроллера системы (далее модем);

– настройка маяков;

– опрос аппаратной части о текущем состоянии, наличии ошибок.

Программа должна обрабатывать координаты в случае их наличия: переводить в формат, запрошенный клиентским программным обеспечением и проверять на корректность, после чего осуществлять передачу координат посредством беспроводного интерфейса Wi-Fi.

В данной работе были поставлены следующие задачи:

– при наличии аналогов проанализировать их функционал;

– выбрать язык программирования, среду и методы разработки;

– на основе технического задания предприятия спроектировать программное обеспечение, описав модули;

– реализовать функциональную версию программного обеспечения;

– провести тестирование и отладку программного продукта.

В первой главе представлен обзор аналогов и анализ технологических решений для реализации программного обеспечения. Во второй главе описаны требования к реализуемому приложению. Третья глава содержит в себе проектирование архитектуры и описание используемых алгоритмов. В четвертой главе содержится реализация программного продукта. В пятой главе описано тестирование разработанного программного обеспечения.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Предприятием была выбрана система позиционирования «Marvelmind», использующая ультразвуковые передатчики и приемники для установления координат отслеживаемого объекта. Данная система была выбрана на основании критериев точности, обеспечиваемой использованием ультразвуковых технологий и невысокой стоимости относительно аналогичных аппаратных комплексов позиционирования. Однако минусом данного продукта является его программное обеспечение, функционал которого не покрывает потребности предприятия в полной мере.

Основная задача, поставленная предприятием, заключается в разработке собственного программного обеспечения, которое сможет заменить проприетарный программный комплекс, поставляемый в комплекте с выбранной системой.

Проприетарное программное обеспечение не удовлетворяет заказчика по следующим критериям:

– невозможность интеграции программного продукта в процессы производства;

– отсутствие разделения на серверную и клиентские части;

– нестабильная работа программного обеспечения;

– невозможность одновременного запуска нескольких экземпляров программного обеспечения;

– исходный код программного продукта недоступны, что делает невозможным редактуру приложения для нужд предприятия;

– неудобный интерфейс клиентской программы.

Основной причиной разработки нового программного обеспечения является требование предприятия к последующей интеграции данной системы в общие производственные и логистические процессы. Этот критерий так же объясняет модульную структуру нового программного обеспечения, которая позволит легко и эффективно интегрировать систему в существующий программный комплекс, используемый на предприятии. Приложение, реализованное в ходе выполнения выпускной квалификационной работы, является собственностью предприятия, что позволяет предприятию использовать его в любых целях и вносить любые изменения.

1.2 Анализ основных технологических решений

Для реализации программного обеспечения требуется выбрать язык программирования, который позволит реализовать запрошенный функционал в полной мере. Одним из важнейших критериев является продолжительная поддержка данного языка программирования, что гарантирует предприятию стабильную работу программного обеспечения. Также выбранный язык должен иметь удобную, функциональную среду разработки. Соблюдение данного критерия позволяет сэкономить время на этапе разработки программного проудкта, а также повысить покрытие разработанного программного обеспечения тестами. Кроссплатформенность в данном случае не является обязательным критерием, однако возможность запуска приложения на различных системах приветствуется заказчиком.

Для сравнения и выбора были взяты следующие языки программирования:

– Python;
– C++;
– C#.

Python – язык программирования, относящийся к категории языков высокого уровня [7]. Данный язык программирования имеет динамическую строгую типизацию, автоматическое управление памятью и полностью объектно-ориентированную структуру, так как в данном языке всё является объектами. Код программ, написанных на языке Python, отличается емкостью, компактностью, лёгкой читаемостью и минималистичным синтаксисом, что облегчает чтение кода и уменьшает затраты времени на последующее сопровождение и рефакторинг программы. Благодаря интерпретируемости, данный язык программирования позволяет создавать кроссплатформенные приложения.

Интерпретатор CPython – эталонная реализация языка Python, является стандартом языка. Данный интерпретатор распространяется под свободной лицензией, позволяющей использовать его без каких-либо ограничений при создании и использовании любых приложений, в том числе и проприетарных.

Однако из-за особенностей всех интерпретаторов языка Python, скорость выполнения программ крайне низкая в сравнении с программами, написанными с использованием других языков программирования. Данный язык имеет динамический характер, что безусловно ведёт к дополнительным расходам мощностей при выполнении программ. Данная особенность является ключевой при выборе языка программирования, т.к. программа, отвечающая за систему позиционирования объектов должна выполняться максимально эффективно по времени. При написании особо важных блоков кода принято использовать языки программирования низкого уровня, что очень сильно усложняет и замедляет разработку программы и её последующее сопровождение. Данная интеграция низкоуровневых языков опирается на сторонние библиотеки, что усложняет коммерческое использование и снижает последующую стабильность работы программы.

Самой популярной и полнофункциональной средой разработки для создания программ на данном языке является PyCharm IDE. При коммерческом применении данная среда разработки является платной.

Таким образом, преимуществами данного языка программирования высокого уровня являются:

– компактный и легко читаемый код;

– большой встроенный функционал;

– кроссплатформенность итоговых программ;

– высокая скорость написания программ.

Недостатками Python соответственно можно назвать:

– сравнительно низкая скорость выполнения итоговых программ;

– сравнительно малая эффективность выполнения программ;

– необходимость интегрировать блоки кода низкого уровня.

C++ – компилируемый язык программирования, обладающий статической типизацией [8]. Данный язык поддерживает обширное количество парадигм программирования: объектно-ориентированную, процедурную, обобщенную. Язык C++ имеет обширное количество учебной литературы, примеров программ, что позволяет в полной мере изучить все аспекты языка программирования и создавать качественные программы. Однако синтаксис данного языка сравнительно неудобен и узок, что сужает спектр применимости языка и усложняет разработку. Также минусом является необходимость контролировать обращения программы к памяти, так как это может привести к ошибкам.

Оптимальной средой разработки для данного языка является MS Visual Studio 2019 [9]. Данная среда разработки имеет широкий функционал для написания, отладки и финальной сборки программ. Она получила широкое распространение и имеет объемную официальную документацию,

предоставленную в разных языках. Visual Studio является платной для коммерческого использования.

Положительными сторонами данного языка программирования являются:

– сочетание свойств языков как низкого, так и высокого уровня;
– сравнительно высокая скорость выполнения итоговых программ;
– сравнительно низкое потребление памяти и процессорного времени;
– высокая популярность, обширная поддержка;
– кроссплатформенность итоговой программы.

Негативными сторонами применения данного языка являются;

– низкая скорость написания программ;
– объемность и низкая читаемость кода;
– усложненное покрытие готового кода тестами;
– наличие большого количества известных уязвимостей.

C# – объектно-ориентированный язык программирования, разработанный компанией Microsoft [10]. Данный язык является основным при написании приложений для платформ Microsoft .NET Framework, .NET Core и .NET5. Имея C-подобный синтаксис, данный язык идеологически близок к Java. C# имеет статическую типизацию, поддерживает многие парадигмы программирования. Список его особенностей и поддерживаемых функций широк: полиморфизм, делегаты, события, перегрузка операторов и многое другое. Код, написанный на данном языке, имеет средний объем и высокую читаемость. Платформа .NET5, являющаяся симбиозом двух предшественников .NET Framework и .NET Core, является кроссплатформенной и позволяет запускать приложения на различных системах, как Windows, так и Unix. Производительность приложений, запущенных на платформе .NET5, является оптимальной, значительно выигрывая у Python и немногим проигрывая C++. C# позволяет реализовать

сложные модульные конструкции приложений, интегрировать в них различные сервисы, такие как сетевые модули, облачные вычисления, периферийные модули. Совмещая объектно-ориентированную и событийную парадигмы, можно создать оптимальную по времени выполнения программу, которая будет безопасной и защищенной от различных ошибок памяти, внешних злоумышленных воздействий. Программы, написанные на языке C#, выполняются не прямо на самой машине, а внутри CLR (англ. Common Language Runtime, общеязыковая исполняющая среда). Данная особенность предоставляет языку те возможности, которые недоступны классическим языкам программирования. Для примера – автоматическая сборка мусора. Это автоматическая форма управления памятью, отвечающая за периодическое освобождение памяти, удаляя ненужные объекты. В таких языках как C++ данная обязанность возложена на программиста, который должен вручную прописывать все функции очистки памяти, иначе возможны ошибки и утечки. Поэтому язык C# вкупе с CLR и платформой .NET5 облегчают обеспечение безопасной и стабильной работы программ, исключая вектор атак на память, используемую программой.

Преимуществами языка C# можно назвать:

– относительную легкость в написании кода;

– емкость, компактность, читаемость кода;

– широкое покрытие кода различными тестами;

– кроссплатформенность;

– скорость и эффективность выполнения программы;

– наличие широкого спектра различной учебной литературы, в том числе официальной документации Microsoft [10].

Таким образом, для разработки и реализации серверного программного обеспечения был выбран язык высокого уровня C# вкупе с платформой .NET5.

Данная связка является актуальной, будет иметь длительную поддержку и проста в реализации.

В качестве среды разработки была выбрана Microsoft Visual Studio 2019, которая является основной и единственной официальной средой разработки для языка C# и платформы .NET5. Функционал Visual Studio объемен и позволяет провести весь цикл разработки, реализации и выпуска продукта. Среда имеет встроенное покрытие кода тестами, интеллектуальные методы анализа программ и автоматическое обнаружение разного рода ошибок. На официальном сайте Microsoft представлено большое количество информации и учебной литературы, позволяющей полностью изучить и использовать функционал данного инструмента для разработки. Данная среда платна при коммерческом использовании, однако для написания выпускной квалификационной работы была использована студенческая версия. Впоследствии при работе на предприятии будет использована корпоративная коммерческая версия.

1.3 Вывод

В качестве основного языка программирования для решения поставленных задач был выбран язык высокого уровня C#. Широкий функционал и легко читаемый код позволит интегрировать созданное программное обеспечение в процессы предприятия с минимальными затратами финансов и времени.

В качестве среды разработки была выбрана Microsoft Visual Studio 2019, позволяющая полностью использовать все инструменты C#, облегчить и ускорить разработку приложения.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Предприятием были установлены следующие функциональные требования к разрабатываемому программному обеспечению:

– возможность одновременного подключения нескольких клиентских приложений;

– возможность установления соединения с аппаратной частью системы позиционирования посредством двух каналов связи: Wi-Fi и UART;

– возможность настройки аппаратной части системы позиционирования согласно официальной документации системы позиционирования;

– вывод всех полученных и отправленных пакетов данных посредством консоли для целей отладки и контроля за функционированием системы;

– получение и обработка всех возможных пакетов данных, отправленных аппаратной частью системы, согласно документации системы;

– вывод данных о текущих координатах всех отслеживаемых объектов и стационарных маяков посредством консоли;

– возможность получения информации о версии прошивок модема и подключенных к нему маяков;

– передача полученных координат в клиентские программы посредством сетевого протокола UDP;

– возможность сохранения ошибок программы, ошибок сетевого взаимодействия и ошибок аппаратной части системы позиционирования в отдельный файл;

– хранение всех настроек программы в конфигурационном файле, находящемся на сервере.

2.2 Нефункциональные требования

Разработанное приложение должно соответствовать следующим системным требованиям:

– иметь совместимость с платформами семейств Windows и Unix;

– код программы должен быть написан на языке программирования C#;

– приложение должно быть разработано с использованием унифицированной платформы .NET5;

– программа должна иметь модульную структуру для последующей интеграции в информационно-вычислительную систему производства;

– все вычисления должны локальный характер.

3 ПРОЕКТИРОВАНИЕ

3.1 Проектирование структуры программного продукта

Для реализации программного продукта была выбрана объектно-ориентированная парадигма (далее ООП). Данный выбор позволяет изолировать модули друг от друга, ограничив доступ к некоторым данным извне, при этом сохраняя полное функциональное взаимодействие. Обобщенная архитектура подобного решения представлена на рисунке 1.

Данная структура программы содержит в себе:

– классы, позволяющие разбивать программу на модули;

– объекты данных классов, т.е. экземпляры;

– параметры объектов класса, которые включают в себя свойства, методы, события.

События в данной структуре позволяют связать модули посредством сообщений, инициируемых произошедшим событием и принимаемых обработчиком событий.

Рисунок 1 – Объектно-ориентированная парадигма

Для упрощения и типизации передачи и обработки данных, в программе будут описаны структуры данных. Структурой является совокупность переменных, которые объединены под одним именем. Объявление структуры приводит к образованию шаблона, который впоследствии используется для создания объектов структуры. Такая типизация данных позволяет упростить разработку, а также сделать более эффективную программу, избегая лишних переменных.

Благодаря использованию ООП при создании программы, была достигнута требуемая модульность итогового программного продукта. Код программы содержит следующие модули:

– модуль описания маяков;

– модуль описания модема;

– модуль сетевого взаимодействия;

– модуль описания возможных пакетов данных;

– основной модуль программы;

– модуль сериализации и десериализации сетевых данных;

– модуль связи с аппаратной частью посредством UART;

– модуль записи данных об ошибках и статусах программы.

Данные модули связаны между собой посредством событий и наследования. Каждый модуль является отдельным блоком кода, имеющим модификаторы доступа и не зависящим от остальных модулей. Так, например, если в модуле описания модема произойдёт какая-либо ошибка, то программа сохранит отчёт об ошибке, оповестив пользователей, но не завершит свою работу, т.к. отказоустойчивость программы обеспечена основным модулем программы. При совершении ошибок в модулях, функционал программы может быть частично утрачен, однако программа не завершит свою работу, тем самым не произойдёт отказ в обслуживании.

Также ООП обладает одним из важнейших параметров – инкапсуляцией данных. Это, например, позволяет защитить данные, используемые во время выполнения программы и не допустить стороннего взаимодействия со всеми модулями в целом.

Модуль сериализации и десериализации сетевых данных является модулем API, реализующим все необходимые методы формирования и чтения сетевых данных для получения или отправки посредством протокола UDP. В данном модуле используется возможность языка C# иметь тип данных dynamic. Элементы, типизированные как dynamic, поддерживают любые операции, а допустимые действия с объектами или выражениями определяются компилятором уже во время работы программы. Таким образом происходит упрощение разработки модуля API, который требует точного установления типа входных данных для последующей их записи в новый экземпляр соответствующего класса.

В модуле сетевого взаимодействия описана работа с протоколом передачи данных UDP. Использованы стандартные объекты UdpClient и IPEndPoint, принадлежащие классу System.Net. Связь с модулями, участвующими в сетевом обмене реализована посредством делегата и соответствующего события, оповещающих о получении каких-либо данных на прослушиваемый порт.

При создании модуля описания возможных пакетов данных, была использована документация к аппаратной части системы позиционирования «Marvelmind». В данном классе описаны все возможные пакеты данных, которые передают и принимают информацию от аппаратной части: настройки маяков, настройки модема, координаты маяков, статусы маяков, сообщения об ошибках.

3.2 Алгоритмы решения

Модульность, обеспеченная разделением программы на классы и объекты позволяет гибко настраивать каждый класс. В процессе разработки структуры программы после этапа разделения программы на отдельные модули, было решено выделить основные классы:

– класс Beacon, отвечающий за описание параметров маяков;

– класс Modem, отвечающий за описание параметров модема;

– класс Network, отвечающий за работу с протоколом UDP;

– абстрактный класс IncomingPacket, описывающий пакеты, принимаемые со стороны модема посредством UART;

– абстрактный класс RequestPacket, описывающий пакеты, отправляемые модему посредством UART;

– класс Program, являющийся основным классом и содержащий точку входа в программу;

– класс SerializerAPI, содержащий методы сериализации и десериализации абстрактных данных;

– класс SerialPortConnection, отвечающий за работу с модемом посредством интерфейса UART.

Основным классом программы является класс Program, внутри которого содержится статичный метод Main. После запуска программы данный метод является основным потоком, в котором происходит объявление, создание и инициализация всех остальных классов, отражающих модули. Внутри себя данный класс имеет глобальные объекты, представленные в листинге 1.

Листинг 1 – Глобальные объекты класса Program

```
#region Public
public static byte[] ReceiveBuffer { get; set; }
#endregion
```

```
#region Private
private static SerialPortConnection Connection { get; set; }
private static Modem Supermodem { get; set; }
private static Network ModemUdpStream { get; set; }
private static Network ClientUdpStream { get; set; }
private static SerializerAPI Serializer { get; set; }
#endregion
```

Помимо основного класса Program, серверное приложение также содержит другие классы, описывающие объекты и взаимодействие с ними. Пример двух публичных классов, описывающих все доступные параметры модема представлен в листинге 2.

Листинг 2 – Объекты классов Modem, ModemSettings

```
public class ModemSettings
{
    public byte MajorVersion { get; set; }
    public byte MinorVersion { get; set; }
    public byte BaseBeaconID { get; set; }
    public byte SecondBaseBeaconID { get; set; }
    public bool HighResolutionMode { get; set; }
    public bool MovementFiltering { get; set; }
    public bool PowerSaveMode { get; set; }
}

public class Modem
{
    #region Private
    private const byte ModemAddress = 0xff;
    #endregion

    #region Public
    public List<Beacon> Beacons { get; set; }
    public List<Submap> Submaps { get; set; }
    public ModemSettings Settings { get; set; }
}
```

В листинге 3 представлены объекты, содержащиеся в классах Beacon и BeaconSettings. Данные объекты описывают маяки и их параметры соответственно. Класс Beacon также используется в качестве шаблона для сериализации и десериализации данных, участвующих в передаче информации между серверным и клиентскими приложениями соответственно.

Листинг 3 – Объекты классов Beacon, BeaconSettings

```
public class BeaconSettings
{
    public byte ID { get; set; }
    public bool isHedge { get; set; }
    public bool isAwake { get; set; }
    public string FirmwareVersion { get; set; }
    public int UartBaud { get; set; }
    public int RadioProfile { get; set; }
    public int RadioBand { get; set; }
    public int Submap { get; set; }
    public bool Exists { get; set; }
}
[Serializable]
public class Beacon
{
    public CoordinatesStructure_mm Coordinates_mm { get; set;
     }
    public BeaconSettings Settings { get; set; }
    [JsonIgnore]
    public string Name { get { return $"Beacon { Settings.ID
}"; } }
}
```

В качестве формата данных для совершения обмена информацией между серверным и клиентскими приложениями был выбран текстовый формат обмена данными JSON. Плюсами формата JSON является легкая читаемость как человеком, так и компьютером, высокая степень эффективности и автоматизации конвертации данных в сообщение и обратной конвертации сообщения в данные.

В листинге 4 представлен пример JSON сообщения, генерируемого программой при сериализации экземпляра класса типа Beacon.

Листинг 4 – Формат JSON сообщения

```
127.0.0.1 : {
  "MethodType": "Beacon",
  "Contents": {
    "Coordinates_mm": {
      "X": 0,
      "Y": 0,
      "Z": 0
    },
    "Settings": {
```

```
      "ID": 3,
      "isHedge": false,
      "isAwake": false,
      "FirmwareVersion": "7.01",
      "UartBaud": 115200,
      "RadioProfile": 8,
      "RadioBand": 12,
      "Submap": 0,
      "Exists": true
    }
  }
}
```

Именно поле MethodType позволяет серверному и клиентскому приложениям корректно осуществлять динамическую типизацию, упрощая разработку и сопровождение приложения. Динамическая типизация также позволит интегрировать данный модуль API в любую систему предприятия, реализованную на базе C# .NET5.

Для реализации сетевого взаимодействия был создан класс Network, глобальные объекты которого приведены в листинге 5.

Листинг 5 – Объекты класса Network

```
#region Public
public delegate void UdpDataReceivedHandler(object sender,
DataReceivedEventArgs e);
public event UdpDataReceivedHandler OnDataReceived;
public ConcurrentDictionary<IPEndPoint, Pipe> EndpointsPipes;
public bool AllowDebug = false;
#endregion
#region Private
private bool Run = false;
private UdpClient sender;
#endregion
```

Листинг 5 показывает, что в модуле сетевого взаимодействия присутствует событие, позволяющее передавать полученные данные в текстовом формате в обработчик. Впоследствии обработчик десериализует эти данные.

4 РЕАЛИЗАЦИЯ

После запуска консольного приложения первоначально требуется ввести логин и пароль локального администратора. Данный шаг требуется для защиты всей системы позиционирования от несанкционированного доступа. Логин и пароль перед сохранением проходят этап хэширования, что гарантирует их защищенность от каких-либо утечек. Окно ввода логина и пароля представлено на рисунке 2.

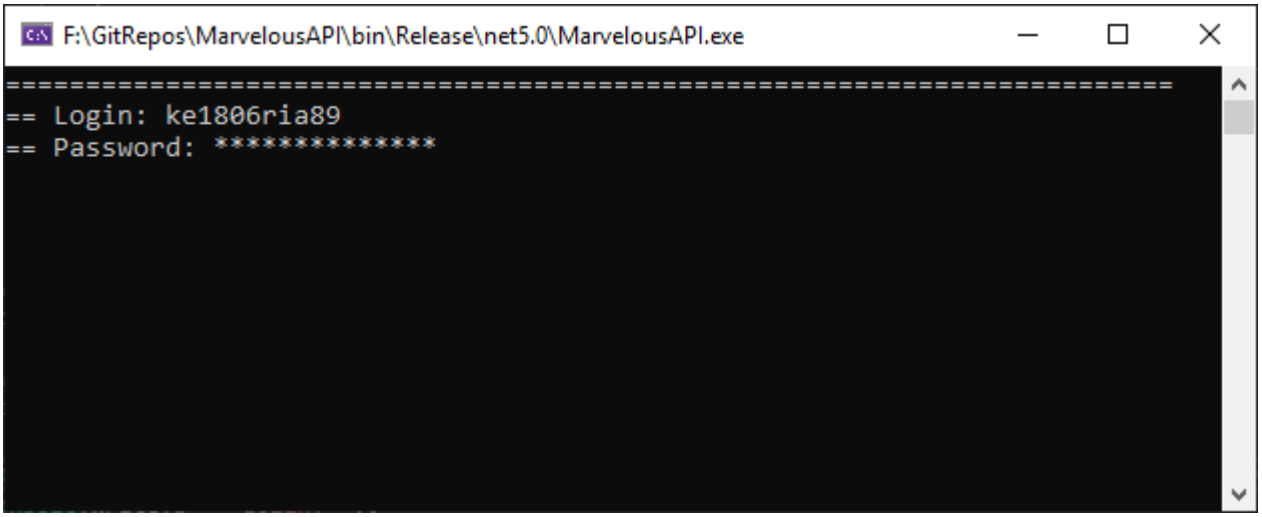


Рисунок 2 – Окно авторизации пользователя

Для реализации защиты пароля от перехвата был использован класс System.Security.SecureString, который является стандартным для C#. Код, отвечающий за скрытый ввод пароля и его последующую инкапсуляцию представлен в листинге 6.

Листинг 6 – Функция безопасного ввода пароля

```
private static SecureString GetPassword()
{
   SecureString password = new();
   while (true)
   {
       ConsoleKeyInfo keyPressed = Console.ReadKey(true);
       if (keyPressed.Key == ConsoleKey.Enter) break;
       else if (keyPressed.Key == ConsoleKey.Backspace)
       {
```

```
            if (password.Length > 0)
            {
                password.RemoveAt(password.Length - 1);
                Console.Write("\b \b");
            }
        }
        else if (!char.IsControl(keyPressed.KeyChar))
        {
            password.AppendChar(keyPressed.KeyChar);
            Console.Write("*");
        }
    }
    return SecureWithHash(password);
}
```

После ввода данных для аутентификации, пользователь попадает в меню выбора метода подключения к модему. Согласно техническому заданию предприятия, было реализовано два метода установления связи с модемом: посредством Wi-Fi и посредством UART. Данные методы различны по функционалу и не являются взаимозаменяемыми. Для первоначальной настройки модема используется интерфейс UART, а для последующего получения координат объектов используется беспроводной интерфейс Wi-Fi. Меню выбора представлено на рисунке 3.

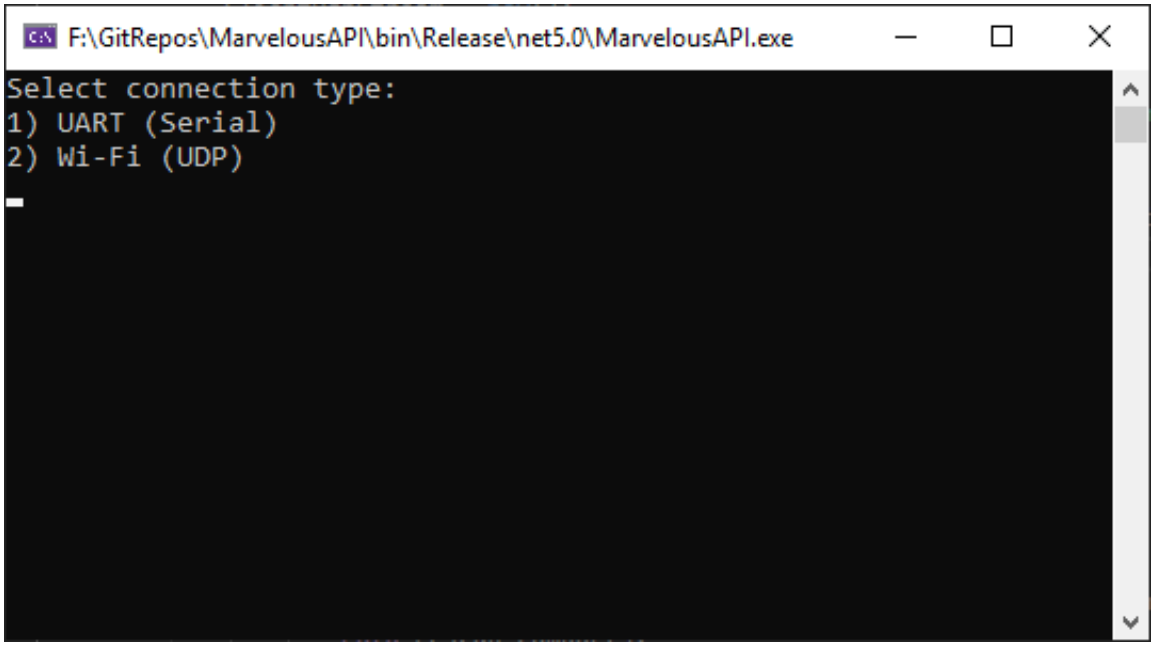


Рисунок 3 – Меню выбора метода подключения

После выбора желаемого метода подключения, осуществляется инициализация соответствующего интерфейса. Код инициализации для интерфейса UART представлен в листинге 7, для интерфейса Wi-Fi представлен в листинге 8.

Листинг 7 – Инициализация интерфейса UART

```
if (Connection != null) Connection = null;
Connection = new();
Console.Clear();
Console.WriteLine("Select serial port:");
List<string> comportnames = Connection.Scan();
if (comportnames.Count == 0)
{
    DrawMessage("No COM ports availaible! Press any key to
retry...", ref choice);
    goto rescan_comports;
}
for (int i = 0; i < comportnames.Count; i++)
Console.WriteLine($"{ i + 1 }: { comportnames[i] }");
choice = Convert.ToInt32(Console.ReadKey().KeyChar - '0');
if (choice < 0 || choice > comportnames.Count) goto
rescan_comports;
if (choice == 0) return;
else
{
    Connection.Open(comportnames[choice - 1]);
    Connection.Port.DataReceived += new
SerialDataReceivedEventHandler(SerialDataReceived);
    Connection.Port.DiscardInBuffer();
    Connection.Port.DiscardOutBuffer();
}
```

Листинг 8 – Инициализация интерфейса Wi-Fi

```
ClientUdpStream = new();
Serializer = new();
Supermodem = new();
ClientUdpStream.OnDataReceived += new
Network.UdpDataReceivedHandler(ClientUdpDataReceived);
ClientUdpStream.Start(40000);
ModemUdpStream = new();
ModemUdpStream.Start(49100);
ModemUdpStream.OnDataReceived += new
Network.UdpDataReceivedHandler(ModemUdpDataReceived);
```

На рисунке 4 представлено меню, доступное пользователю при подключении к модему посредством проводного интерфейса UART. Оно содержит весь функционал, описанный предприятием в техническом задании и доступный для реализации согласно технической документации «Marvelmind».

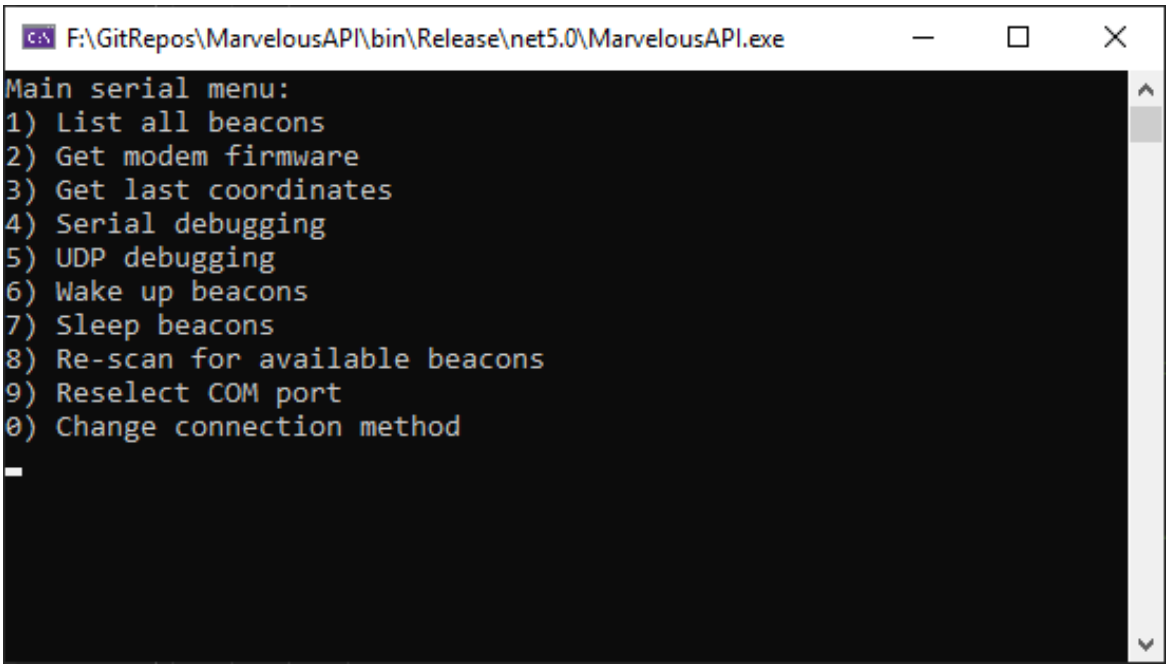


Рисунок 4 – Главное меню при подключении посредством UART

На рисунке 5 показан вариант меню, который доступен при выборе подключения посредством Wi-Fi.

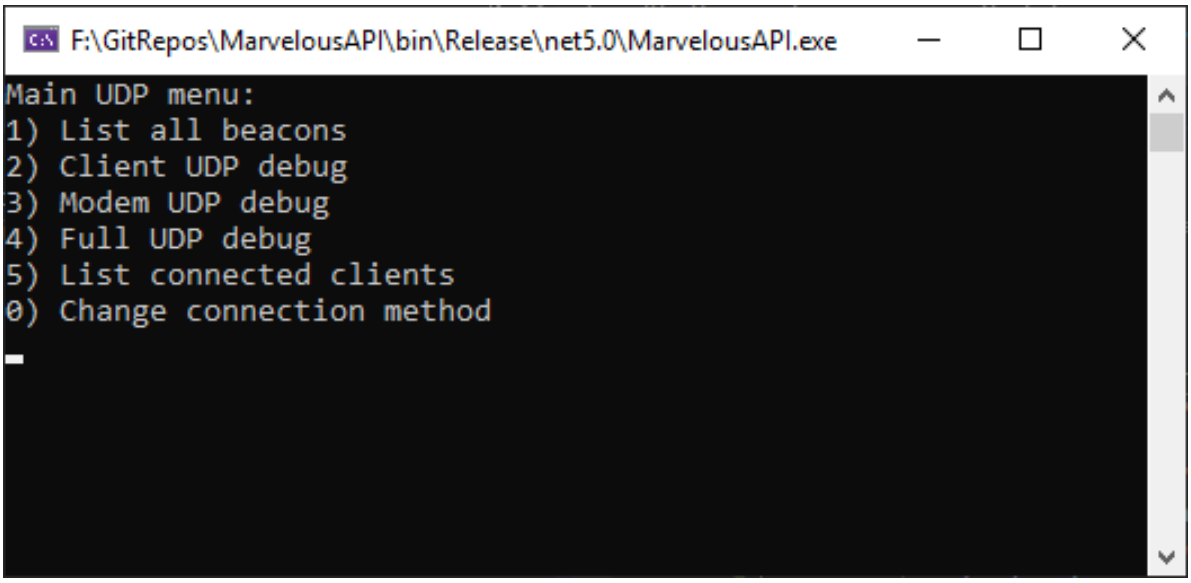


Рисунок 5 – Главное меню при подключении посредством Wi-Fi

Выбор нужного пункта в меню совершается путем однократного нажатия на цифру, соответствующую этому пункту. После выбора происходит либо вызов какого-либо действия, либо переход в следующее подменю.

Одним из основных функциональных требований к приложению является необходимость получения данных программой как со стороны модема, так и со стороны клиентских приложений. В листинге 9 представлен метод ClientUdpDataReceived. Он отвечает за обработку полученных данных и является частью событийной парадигмы в приложении.

Листинг 9 – Методы ClientUdpDataReceived

```
public static void ClientUdpDataReceived(object sender,
Network.DataReceivedEventArgs e)
{
    ClientUdpStream.DebugWriteLine($"{e.RemoteIP.Address} :
{e.Message}");
    dynamic data = Serializer.Deserialize(e.Message);
    if (data?.GetType().ToString() == "MarvelousAPI.Beacon")
    {
        foreach (PropertyInfo propertyInfo in
Type.GetType("MarvelousAPI.BeaconSettings").GetProperties())
        {
            try
            {
                object value =
propertyInfo.GetValue(data.Settings);
                if (value != null)
propertyInfo.SetValue(Supermodem.Beacons[Supermodem.Beacons.Fi
ndIndex(x => x.Settings.ID ==
((Beacon)data).Settings.ID)].Settings, value);
            }
            catch (Exception) { }
        }
    }
}
```

Листинг 10 содержит код метода ModemUdpDataReceived, отвечающего за обработку данных, полученных от модема. В ходе обработки происходит сортировка пакетов по типам инкапсулированных в них данных. Данный метод

также является событием, происходящем при появлении пакетов UDP на прослушиваемом порту.

Листинг 10 – Методы ModemUdpDataReceived

```
public static void ModemUdpDataReceived(object sender,
Network.DataReceivedEventArgs e)
{
    int packgeType = e.Message[3];
    if (packgeType == 0x0011)
     {
        int id = (byte)e.Message[0];
        int x = (byte)e.Message[12];
        int y = (byte)e.Message[16];
        int z = (byte)e.Message[12];
        Supermodem.Beacons[id].Settings.Exists = true;
        Supermodem.Beacons[id].Settings.isAwake = true;
        Supermodem.Beacons[id].Settings.isHedge = true;
        Supermodem.Beacons[id].Coordinates_mm = new
CoordinatesStructure_mm { X = x, Y = y, Z = z};
        foreach (KeyValuePair<IPEndPoint, Pipe> endPoint in
ClientUdpStream.EndpointsPipes)
ClientUdpStream.Send(Serializer.Serialize(Supermodem.Beacons[(
byte)e.Message[0]]), endPoint.Key);
ModemUdpStream.DebugWriteLine(Serializer.Serialize(Supermodem.
Beacons[(byte)e.Message[0]]));
    }
    else if (packgeType == 0x0012)
     {
        for (int i = 0; i < (byte)e.Message[5]; i++)
          {
            int id = (byte)e.Message[6 + i * 14 + 0];
            int x = (byte)e.Message[6 + i * 14 + 4];
            int y = (byte)e.Message[6 + i * 14 + 8];
            int z = (byte)e.Message[6 + i * 14 + 12];
            Supermodem.Beacons[id].Settings.Exists = true;
            Supermodem.Beacons[id].Settings.isAwake = true;
            Supermodem.Beacons[id].Settings.isHedge = false;
            Supermodem.Beacons[id].Coordinates_mm = new
CoordinatesStructure_mm { X = x, Y = y, Z = z };
            foreach (KeyValuePair<IPEndPoint, Pipe> endPoint
in ClientUdpStream.EndpointsPipes)
ClientUdpStream.Send(Serializer.Serialize(Supermodem.Beacons[i
d]), endPoint.Key);
        }
     }
}
```

Данные методы вызываются при получении данных на указанный в программе порт. Происходит анализ данных, сортировка и вызов соответствующих методов обработки. Методы класса SerializerAPI, отвечающие за сериализацию и десериализацию приведены в листинге 10.

Листинг 11 – Методы Serialize и Deserizlize

```
public string Serialize(dynamic instance)
{
    string typeName = instance?.GetType().ToString();
    Type typeArgument = Type.GetType(typeName);
    Type genericClass = typeof(MethodWrapper<>);
    Type constructedClass =
genericClass.MakeGenericType(typeArgument);
    dynamic created =
Activator.CreateInstance(constructedClass);
    created.Contents = instance;
    JsonSerializerOptions options = new() { WriteIndented =
true };
    return JsonSerializer.Serialize(created, options);
}

public dynamic Deserialize(string json)
{
    MethodWrapper deserializedMethod =
JsonSerializer.Deserialize<MethodWrapper>(json);
    return
JsonSerializer.Deserialize(Convert.ToString(deserializedMethod
.Contents),
Type.GetType($"MarvelousAPI.{deserializedMethod.MethodType}"))
;
}
```

Метод Serialize позволяет представить любой экземпляр любого класса, который передан в качестве аргумента при вызове данного метода, в текстовом виде. Deserialize же совершает обратную процедуру. Унификация достигается благодаря динамической типизации, предоставляемой языком C# и абстрактным типом данных dynamic. В случае с процедурой сериализации, программа сама получает тип данных, записывает его в отдельное поле, превращает в формат JSON и возвращает объект типа string, содержащий готовое сообщение.

5 ТЕСТИРОВАНИЕ

5.1 Методология тестирования

Для проведения тестирования реализованного программного обеспечения были выбраны следующие виды функциональных тестов:

– системное тестирование;

– интеграционное тестирование;

– модульное тестирование.

Функциональные тесты – комплекс проверок программного продукта на соответствие требованиям, заявленным в техническом задании предприятия. Возможно как полное тестирование заявленной функциональности, так и частичная проверка только базовой функциональности. Данный комплекс тестов включает в себя следующие виды тестирований: системное, модульное и интеграционное.

Системное тестирование – метод тестирования, позволяющий выявлять ошибки в работе программы или системы в целом. Данная проверка является высокоуровневой, так как она не учитывает проверку отдельных модулей, проверяя лишь систему целиком [14]. При проведении тестирования всей системы, данная проверка проводится в первую очередь. Это позволяет обнаружить наличие проблем в программе, после чего можно приступить к локализации и устранению данных ошибок на следующих стадиях тестирования.

Интеграционное тестирование – метод тестирования, позволяющий провести анализ корректности межмодульного взаимодействия и своевременно выявить ошибки в соответствующих связях [16]. Данный метод тестирования эффективен вместе с модульным тестированием, так как позволяет точно локализовать проблему. Выполняется при наличии ошибок, выявленных на этапе

системного тестирования, а также в случаях, если системное тестирование не способно обеспечить достаточного покрытия кода тестами.

Модульное тестирование – метод проверки модулей системы по отдельности, позволяющий выявить модули, вызывающие проблемы и своевременно устранить дефекты в них. Данный вид тестирования стоит производить при возникновении проблем на этапах системного либо интеграционного тестирования [15], которые не могут быть локализованы и устранены без редактирования соответствующих модулей.

5.2 Проведение процедуры тестирования

Системное тестирование было проведено посредством имитации работы серверного приложения в реальных условиях. Была подключена аппаратная часть системы позиционирования, были запущены клиентские программы. Весь функционал, представленный в серверном приложении, был испытан, в ходе чего были выявлены и устранены небольшие ошибки, не влияющие на отказоустойчивость приложения. Ошибок, приводящих к отказу в обслуживании, найдено не было, программа функционирует штатно и стабильно. Все показатели сетевого трафика, используемой памяти и процессорной нагрузки полностью устроили заказчика. Графики, отражающие загруженность процессора и использование памяти отображены на рисунке 6.

Данные графики показывают, что после запуска программы и создания всех требуемых экземпляров классов, подключения модема и инициализации всех маяков, использование памяти приложением составляет не более 110 мегабайт. Нагрузка на процессор не превышает 5%, что является крайне незначительной нагрузкой. Пики на графике отражают обработку полученных

пакетов координат со стороны аппаратной части и последующую отправку данных на авторизованные клиентские приложения.

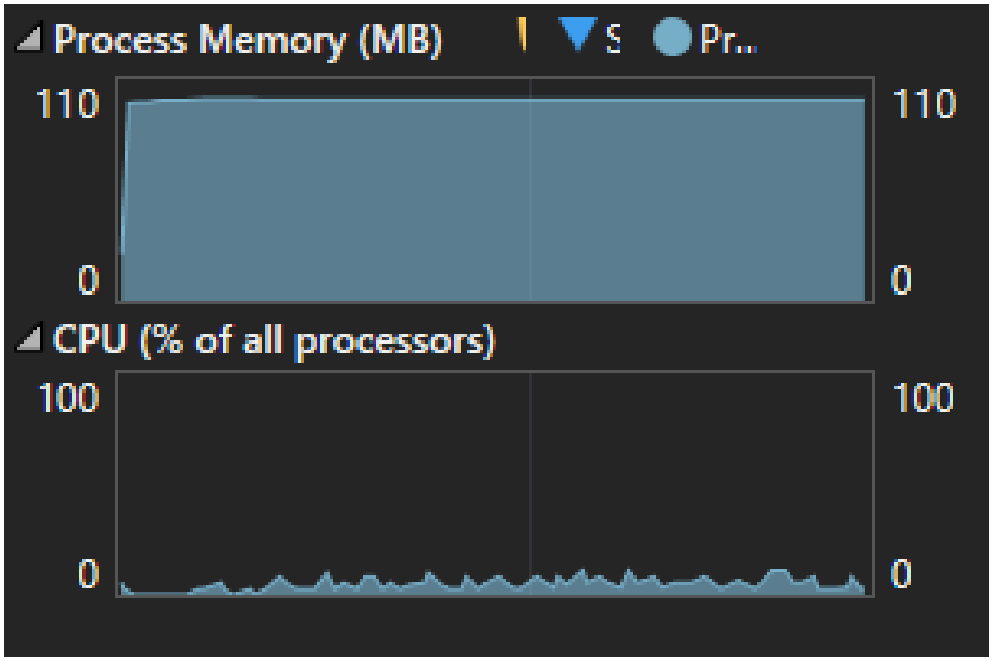


Рисунок 6 – Использование системных ресурсов программой

Интеграционное тестирование заключалось в проверке межмодульного взаимодействия внутри разработанной программы, а также взаимодействия клиентскими частями программного комплекса, имеющимися на предприятии.

В рамках данного тестирования были проверены следующие пункты:

– корректность приема данных со стороны аппаратной части системы;

– корректность формирования и отправки пакетов данных клиентским программам;

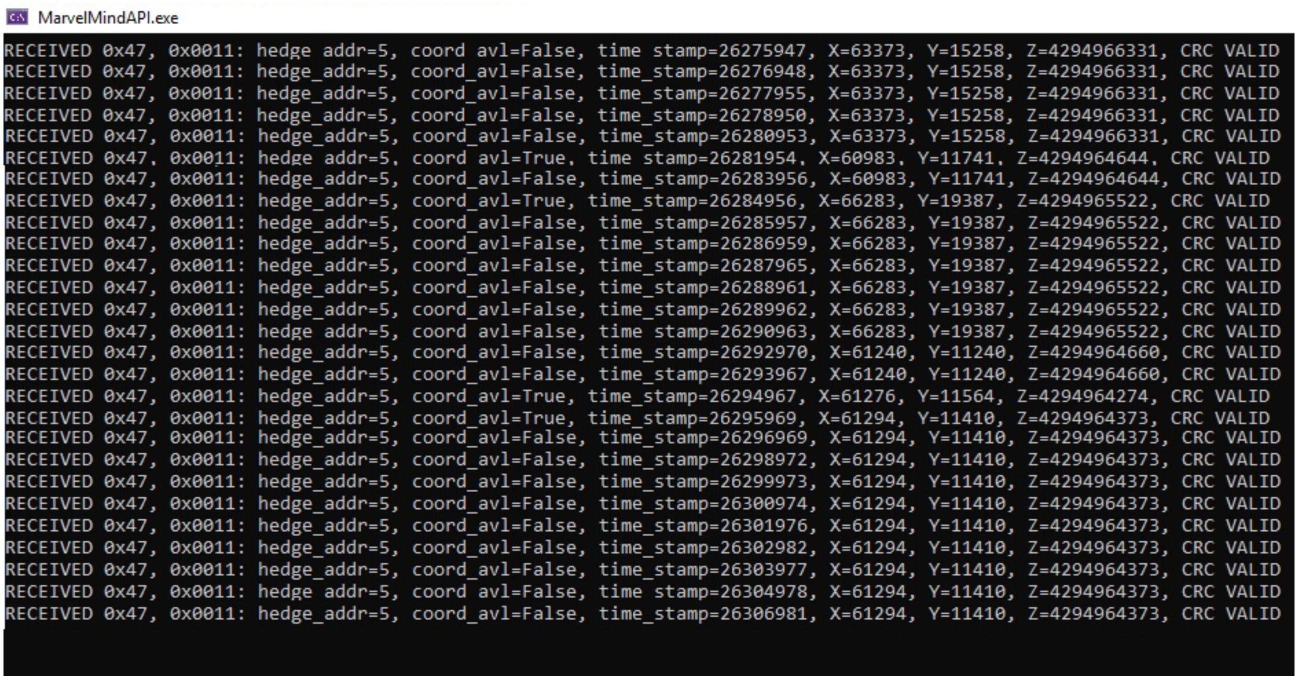
– корректность отображения полученных координат в консоли;

– полноценность функциональности серверного приложения.

Результаты данного тестирования представлены на рисунках 7-9.

На рисунке 7 с периодичностью 1 раз в секунду появляются координаты подвижного маяка. Данные сообщения несут информацию о доступности координат, временной отметке, точных координатах. После получения каждого

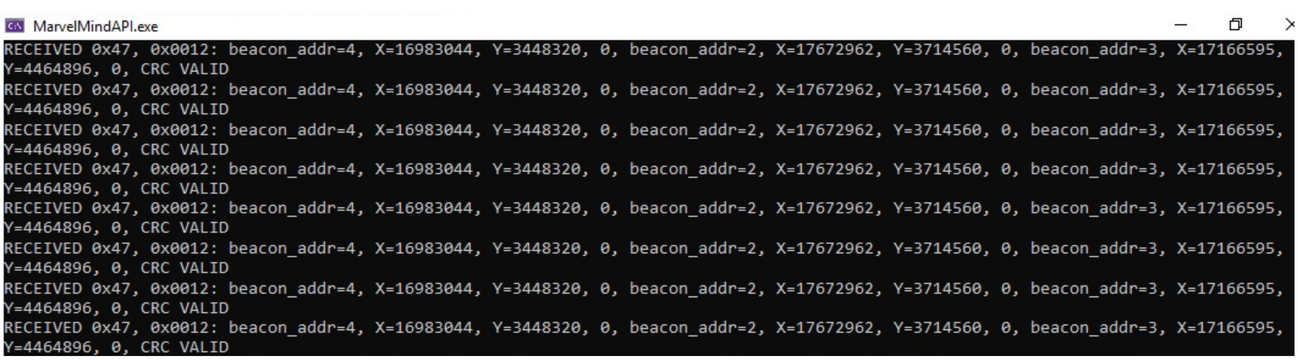
сообщения производится проверка контрольной суммы CRC, что подтверждает верность полученных данных и исключает любые потери на этапе передачи. В данном примере видно, что система не всегда считала координаты актуальными и применимыми, т.к. отладка производилось в малой комнате 15кв.м., что является недостаточным для использования этих датчиков.



```
MarvelMindAPI.exe
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26275947, X=63373, Y=15258, Z=4294966331, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26276948, X=63373, Y=15258, Z=4294966331, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26277955, X=63373, Y=15258, Z=4294966331, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26278950, X=63373, Y=15258, Z=4294966331, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26280953, X=63373, Y=15258, Z=4294966331, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=True, time_stamp=26281954, X=60983, Y=11741, Z=4294964644, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26283956, X=60983, Y=11741, Z=4294964644, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=True, time_stamp=26284956, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26285957, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26286959, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26287965, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26288961, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26289962, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26290963, X=66283, Y=19387, Z=4294965522, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26292970, X=61240, Y=11240, Z=4294964660, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26293967, X=61240, Y=11240, Z=4294964660, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=True, time_stamp=26294967, X=61276, Y=11564, Z=4294964274, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=True, time_stamp=26295969, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26296969, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26298972, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26299973, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26300974, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26301976, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26302982, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26303977, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26304978, X=61294, Y=11410, Z=4294964373, CRC VALID
RECEIVED 0x47, 0x0011: hedge_addr=5, coord_avl=False, time_stamp=26306981, X=61294, Y=11410, Z=4294964373, CRC VALID
```

Рисунок 7 – Отображение позиции отслеживаемого маяка

На рисунке 8 отображаются расстояния между маяками, закрепленными на стенах. Данные расстояния получаются с периодичностью 1 раз в 10 секунд. Можно заметить, что расстояния не меняются и всегда идентичны, так как маяки неподвижно закреплены на стенах.



```
MarvelMindAPI.exe
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
RECEIVED 0x47, 0x0012: beacon_addr=4, X=16983044, Y=3448320, 0, beacon_addr=2, X=17672962, Y=3714560, 0, beacon_addr=3, X=17166595,
Y=4464896, 0, CRC VALID
```

Рисунок 8 – Отображение позиций маяков, закрепленных на стенах

На рисунке 9 представлено главное меню консольного приложения с пунктами, отвечающими за связь с аппаратной и клиентской частями системы. Данное меню отображается после выбора метода подключения к аппаратной части системы, в данном случае был выбран метод подключения посредством проводного интерфейса UART, поэтому в пункте меню под номером 9 предоставлена возможность повторного выбора порта для подключения.

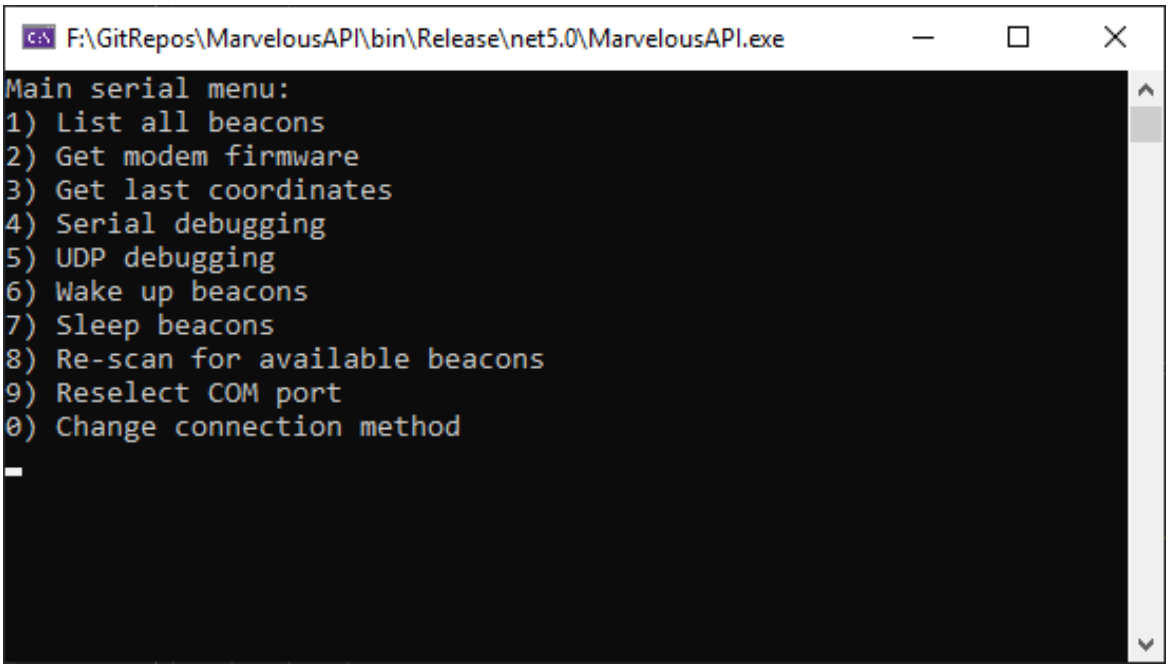


Рисунок 9 – Главное меню консольного приложения

На рисунке 10 отображен пакет с данными для обновления настроек маяка номер 5: у него выключается режим «hedge», что означает, что данный маяк становится неподвижным. Данный пакет получен с IP-адреса 192.168.0.6, который принадлежит клиентской программе.

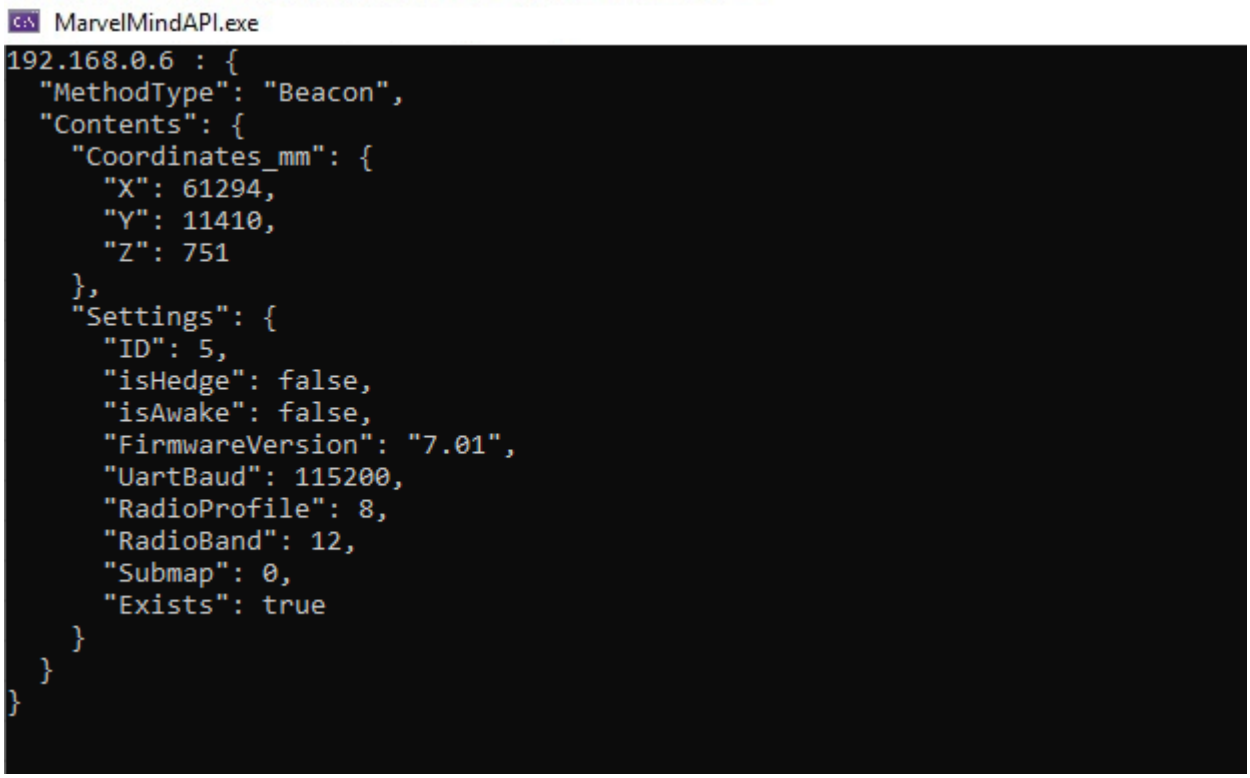


Рисунок 10 – Пример вывода полученного клиентского пакета

По итогам тестирования можно сделать вывод, что реализованное серверное программное обеспечение в полной мере удовлетворяет как функциональным, так и нефункциональным требованиям технического задания предприятия. В том числе серверное приложение выполняет свои задачи без ошибок как на уровне всей системы в целом, так и на уровне отдельных модулей.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была разработана и реализована серверная часть программного комплекса для системы позиционирования объектов внутри производственных помещений.

Были решены следующие задачи:

– проведен анализ существующих аналогов программного продукта;

– выбраны технологические решения и методы для разработки ПО;

– выполнено проектирование системы;

– выполнена реализация системы;

– проведено тестирование рабочей версии программного продукта.

Разработанное программное обеспечение соответствует всем требованиям заказчика в лице предприятия ООО «ДСТ-УРАЛ». Серверное приложение устанавливает связь с аппаратной частью системы позиционирования, после чего устанавливает связь с запущенными клиентскими приложениями. Данный программный комплекс позволяет конечным пользователям клиентских приложений отслеживать местоположения объектов, на которые установлены подвижные маяки. Помимо этого, серверное приложение позволяет в полной мере настраивать аппаратную часть системы позиционирования согласно всем документациям, предоставленным изготовителем аппаратной части.

В дальнейшем данное программного обеспечение будет интегрировано в производственные и логистические процессы внутри предприятия, что позволит оптимизировать работу и повысить эффективность и безопасность передвижения транспорта внутри помещений. Реализованная модульная структура приложения позволяет добавлять новые модули без внесения значительных изменений в исходный код, что позволит заказчику легко расширить функционал системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ДСТ-УРАЛ. О компании. – Текст. Изображения : электронные // ДСТ-УРАЛ : [официальный сайт]. – URL: https://tm10.ru/about/ (дата обращения: 07.02.2022).

2. Вахрушева, А. А. Технологии позиционирования в режиме реального времени / А. А. Вахрушева. – Текст : электронный // Вестник СГУГиТ (Сибирского государственного университета геосистем и технологий). – 2017. – №1. – С. 170–177. – URL: http://vestnik.ssga.ru/wp-content/uploads/2017/04/Вестник-Том-22-№-1_rus.pdf (дата обращения: 28.03.2022).

3. Marvelmind Indoor Navigation System Operating manual. – Текст. Изображения : электронные // Marvelmind Robotics : [официальный сайт]. – URL: https://marvelmind.com/pics/marvelmind_navigation_system_manual.pdf (дата обращения: 18.01.2022).

4. Тельо-Родригес М. Путеводитель по проектированию удобных API / М. Тельо-Родригес // Труды ИСП РАН. – 2021. – №1. – URL: https://cyberleninka.ru/article/n/putevoditel-po-proektirovaniyu-udobnyh-web-api (дата обращения: 02.04.2022).

5. Якушев И. С. Сравнительный анализ влияния параметров физического уровня на характеристики беспроводной сети стандарта IEEE 802.11 / И. С. Якушев – Текст : электронный // Вестник НГУ. Серия: Информационные технологии. – 2009. – №1. – URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-vliyaniya-parametrov-fizicheskogo-urovnya-na-harakteristiki-besprovodnoy-seti-standarta-ieee-802-11 (дата обращения: 19.03.2022).

6. Hardware interfaces and protocols of data exchange with Marvelmind devices. – Текст. Изображения : электронные // Marvelmind Robotics : [официальный сайт].

– URL: https://marvelmind.com/pics/marvelmind_interfaces.pdf (дата обращения: 19.01.2022).

7. The Python Tutorial. – Текст. Изображение : электронные // Python : [официальный сайт]. – URL: https://docs.python.org/3.10/tutorial/index.html (дата обращения: 15.02.2022).

8. C++ language documentation. – Текст. Изображение : электронные // Microsoft : [официальный сайт]. – URL: https://docs.microsoft.com/en-us/cpp/cpp/?view=msvc-170 (дата обращения: 15.02.2022).

9. Welcome to the Visual Studio IDE. – Текст. Изображение : электронные // Microsoft : [официальный сайт]. – URL: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019 (дата обращения: 15.02.2022).

10. Документация по C#. – Текст. Изображение : электронные // Microsoft : [официальный сайт]. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 15.02.2022).

11. Петров, А. А. Протокол передачи данных для UART / А. А. Петров – Текст : электронный // Достижения науки и образования. – 2016. – №1. – URL: https://cyberleninka.ru/article/n/protokol-peredachi-dannyh-dlya-uart (дата обращения: 17.03.2022).

12. User Datagram Protocol. – Текст : электронный // IETF : [сайт]. – URL: https://datatracker.ietf.org/doc/html/rfc768 (дата обращения: 05.04.2022).

13. Как сериализировать и десериализировать (маршалирование и демаршалирование) JSON в .NET. – Текст. Изображение : электронные // Microsoft : [сайт]. – URL: https://docs.microsoft.com/ru-ru/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-6-0 (дата обращения: 12.04.2022).

14. Системное тестирование (System Testing). – Текст. Изображение : электронные // ПРОтестинг.ru : [сайт]. – http://www.protesting.ru/testing/levels/system.html (дата обращения: 13.05.2022).

15. Модульное тестирование (Unit Testing). – Текст. Изображение : электронные // ПРОтестинг.ru : [сайт]. – http://www.protesting.ru/testing/levels/component.html (дата обращения: 15.05.2022).

16. Интеграционное тестирование (Integration Testing). – Текст. Изображение : электронные // ПРОтестинг.ru : [сайт]. – http://www.protesting.ru/testing/levels/integration.html (дата обращения: 15.05.2022).