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»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2022 г.

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

Разработка приложения дополненной реальности для управления роботом-манипулятором

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К МАГИСТЕРСКОЙ ДИССЕРТАЦИИ
ЮУРГУ-090401.2022.208 ПЗ ВКР

Руководитель работы,
к.т.н., зав. кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2022 г.

Автор работы,
студентка группы КЭ-222
_____ А.А. Комар
«___»_____2022 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____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 магистерскую диссертацию
студентке группы КЭ-222
Комар Анастасии Алексеевне
обучающейся по направлению
09.04.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка приложения дополненной реальности для управления роботом-манипулятором» утверждена приказом по университету от 25.04.2022 года № 697-13/12 (приложение №73)

2. **Срок сдачи студентом законченной работы:** 01 июня 2022 г.

3. Исходные данные к работе:

3.1. Учебно-лабораторный манипуляционный РТК с угловой кинематикой:

3.1.1. Сервоприводы Dynamixel MX-64T, MX-28T, AX-12A.

3.1.2. Радиус зоны сервиса не менее 30 см.

3.1.3. Номинальная грузоподъемность – 250 грамм.

3.1.4. Грузоподъемность – 450 - 800 грамм.

3.1.5. Программируемый контроллер STEM Board.

3.2. Ограничения:

3.2.1. Приложение должно запускаться в дополненной реальности.

3.2.2. Приложение должно запускаться на мобильных устройствах с ОС Android.

3.3. Функциональные требования:

3.3.1. Пользователь должен получить ответ о результате изменения реального манипуляционного робота.

4. Перечень подлежащих разработке вопросов:

1. Изучение существующих разработок в области эмуляции процессов управления роботами-манипуляторами.
2. Исследование способов взаимодействия модели робота-манипулятора и приложения.
3. Анализ существующих решений.
4. Проектирование системы дистанционного управления робота-манипулятора.
5. Реализация системы.
6. Тестирование.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы_____/Д.М. Топольский /

Студент _____/А.А. Комар /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	15.03.2022	
Разработка модели, проектирование	15.04.2022	
Реализация системы	01.05.2022	
Тестирование, отладка, эксперименты	15.05.2022	
Компоновка текста работы и сдача на нормоконтроль	27.05.2022	
Подготовка презентации и доклада	30.05.2022	

Руководитель работы _____/Д.В. Топольский /

Студент _____/А.А. Комар /

Аннотация

А.А. Комар. Разработка приложения дополненной реальности для управления роботом-манипулятором. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 64 с., 31 ил., библиогр. список – 20 наим.

В качестве магистерской диссертацией представлена разработка приложения дополненной реальности для управления роботом-манипулятором. Осуществляется анализ существующих решений удаленного управления промышленными манипуляционными роботами. Рассматривается реализация алгоритмов решения прямой и обратной задачи кинематики для возможности изменения положения робота-манипулятора. Производится реализация алгоритмов для клиент-серверного взаимодействия нескольких систем для создания удаленного управления.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Одними из самых ярких представителей промышленных роботов являются манипуляционные роботы. Роботы-манипуляторы – высокотехнологичные приборы, созданные, чтобы перемещать, вращать или иным образом воздействовать на объект путем выполнения тех или иных операций[3]. Такие устройства были созданы, чтобы заменить монотонный, опасный или сложный технический человеческий труд.

В то же время манипуляционные роботы представляют собой сложный электромеханический объект, обладающий рядом особенностей. Роботы-манипуляторы отличаются сложной кинематической структурой, содержащей множество независимых либо взаимосвязанных звеньев. Для описания структуры манипулятора вводится понятие «кинематическая пара». Кинематическая пара – это вид соединения двух звеньев, обеспечивающих определенное относительное движение[1]. Сумма степеней свободы всех кинематических пар позволяет говорить о количестве степеней свободы манипулятора, что в свою очередь является одной из главных кинематических характеристик манипулятора.

Другим важным параметром манипулятора является вид зоны обслуживания – части пространства, охватываемой всеми возможными положениями рабочего инструмента манипулятора[2]. Зона обслуживания манипулятора зависит от конфигурации манипулятора, или, другими словами, вида механики манипулятора. Также от вида механики зависит и выбор системы координат, в которой работает манипулятор. В основном, используются 5 видов механик манипулятора (рисунок 1).

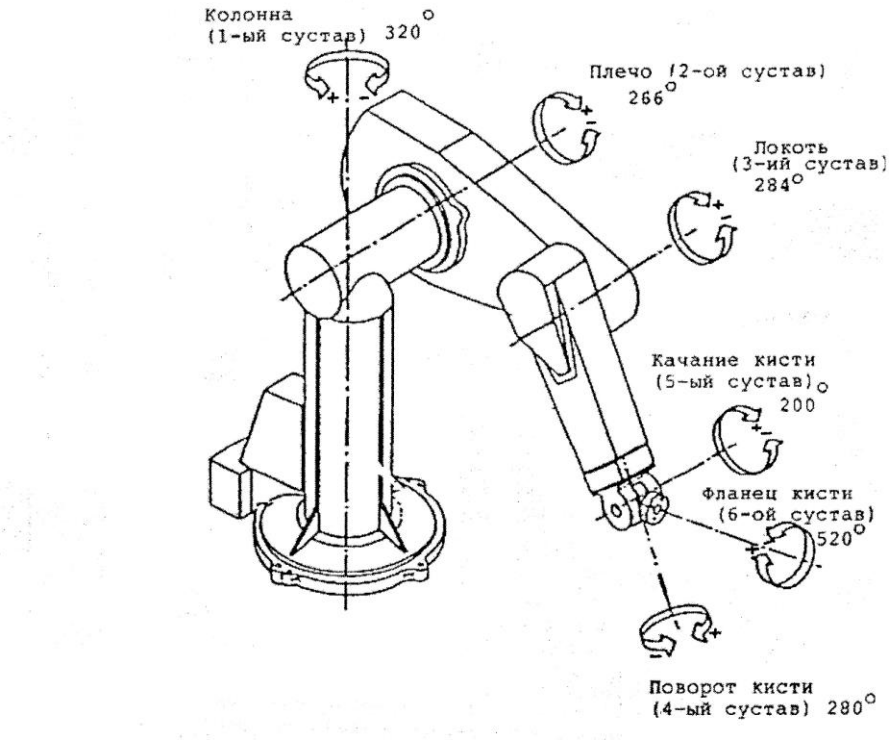

Рисунок 1 - Система координат различных конфигураций манипулятора

Управление промышленных роботов бывает нескольких типов: адаптивное, программное, ручное и основанное на методах искусственного интеллекта[5].

Адаптивное управление предполагает наличие системы обратной связи в виде различных сенсоров и датчиков. Манипулятор получает данные о положении захватываемого объекта и о положении схвата, что позволяет захватить и переместить объект без участия оператора, независимо от начального положения объекта и прочих факторов, способных нарушить работу манипулятора.

Ручное управление – это управление манипулятором, которое осуществляется непосредственно оператором посредством набора рычагов или управляющего механизма[4]. Управление, включающее в себя управляющий манипулятор, - это особый случай ручного управления, который называется копирующим манипулятором. Когда реализован копирующий

манипулятор, исполнительный промышленный робот имеет такую же структуру, как и управляющий. При изменении положения управляющего механизма исполнительный манипулятор копирует его движения.

При использовании программного управления в промышленного робота загружают жесткую последовательность действий, которая регулярно повторяется. Эти действия можно задать несколькими способами.

При цикловом программном управлении программируются последовательность выполнения движений и условия начала и окончания движений. При позиционном программном управлении команды подаются так, что перемещение рабочего органа происходит от точки к точке, причем положения точек задаются программой[6].

В связи с наличием указанных особенностей, для внедрения манипуляционного робота в производственный процесс требуются специально разрабатываемые системы управления. С развитием Интернет вещей и удаленным способом взаимодействия с различным оборудованием возникла необходимость разработки программного обеспечения для удаленного управления.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. АКТУАЛЬНОСТЬ

На протяжении многих веков люди создают механизмы и машины, умеющие упростить нашу жизнь, и современный человек едва ли сможет показать собственную жизнь без них. Каждый день появляются новые устройства и улучшаются имеющиеся. Подобных устройств уже огромное число, однако, бесспорно, наиболее значительным достижением человеческой идеи считаются роботы.

Актуальность выбора основания и использования интерактивного манипулятора определена последующим: использование информационно-коммуникативных технологий в учебной деятельности; увеличение эффективности и результативности учебного процесса, потребностью развития новейшего вида учащегося — грамотного пользователя информационных образовательных услуг.

Автоматизация на предприятиях промышленности всегда имела значение. Человеческие ресурсы в XXI веке — не самый лучший вариант на заводах с точки зрения экономики, тем более что людям свойственно уставать и допускать ошибки. К тому же, во многих регионах существует недостаток квалифицированных человеческих ресурсов. И эта ситуацию будет только усугубляться. В последние годы во всем мире широкое распространение получили промышленные роботы манипуляторы. Они позволяют улучшить качество работы, минимизировать сроки изготовления деталей и сэкономить на промышленных расходах. Промышленная робототехника используется на вредных производствах и опасных участках.

Роботы манипуляторы в производстве - надежная комбинация проверенных технологий с протестированными инновациями, которая дает безотказную работу, короткое время сервисного обслуживания, обеспечивая высокую производительность и качественный продукт.

Преимущества промышленных роботов для автоматизации производства:

1. Универсальность. Манипулятор совместим с различным навесным оборудованием, и при переоснащении может быть использован на других участках производства. В случае необходимости существует возможность использования дополнительного оборудования к роботу.

2. Высокая производительность и качество. Промышленные роботы для автоматизации позволяют производить больше продукции за меньшее время за счет автоматизации производства линий, уменьшая количество брака и улучшая качество.

3. Эффективность. Решения на базе роботизированных манипуляторов и других роботов выгодны для масштабных производств за счет автоматизации промышленности. Роботизированные заводы рентабельны и на малых объемах выпуска продукции и даже в случае ограниченных серий.

4. Гибкость. Простые в использовании программные инструменты автоматизации процессов помогают легко управлять и быстро перенастраивать оборудование при изменении параметров существующих изделий или вводе новых.

5. Результативность. В результате разработки и внедрения инженерных решений, основанных на простоте и эффективности достигается: увеличение производительности, улучшение/стабилизация качества продукции, гибкость и простота управления, снижение эксплуатационных расходов, экономия площадей, сокращение ручного и неэффективного труда, замена людей на вредных участках.

6. Быстрая окупаемость. Срок окупаемости ячейки с одним или двумя манипуляторами может начинаться от 5 месяцев.

7. Роботизация. Оптимальное и экономичное решение по внедрению роботов для автоматизации производства.

Также за последние несколько лет технология дополненной реальности совершила огромный скачок в развитии и расширении сфер применения. Дополненная реальность (AugmentedReality, AR) — это среда с прямым или косвенным дополнением физического мира цифровыми данными в режиме реального времени при помощи компьютерных устройств — планшетов, смартфонов и инновационных гаджетов, а также программного обеспечения к ним[8]. Если раньше эта технология применялась в военной промышленности, компьютерных играх, то сейчас дополненная реальность проникает практически во все сферы деятельности человека: медицину, образование, архитектуру, рекламу и так далее. Дополненная реальность вызывает такой интерес не только потому, что для этого используются новые технологии, но также дополненная реальность обещает помочь людям преодолеть современную информационную перегрузку.

Будущее мировой индустрии, как считают учёные, заключается в полной цифровизации каждого элемента производства. Интеллектуальные роботизированные решения встречаются на каждом этапе деятельности предприятия. Теперь и обучение работников промышленности практикуется с помощью технологий виртуальной и дополненной реальности. Основными задачами промышленных предприятий являются снижение издержек, оптимизация процессов и повышение уровня безопасности производства. Поэтому промышленные компании ищут новые способы и инструменты для более быстрого решения поставленных целей и находят их в области цифровизации.

Существующие платформы AR/VR предлагают проверенный, масштабируемый и оптимизированный путь от проверки концепции до развертывания корпоративных приложений AR/VR. В частности, они дают сотрудникам возможность сократить время ремонта, оптимизировать рабочие процессы, улучшить качество обучения, устранить ошибки и многое другое.

1.2. ОБЗОР АНАЛОГОВ

На рынке программного обеспечения существует множество программ обеспечивающие удаленное управление промышленными роботами. Рассмотрим некоторые из них.

1.2.1 RobotStudio

Программное обеспечение для моделирования и автономного программирования от компании ABB позволяет программировать роботов на персональном компьютере, не останавливая процессы производства, что позволяет выполнять такие задачи, как обучение, программирование и оптимизацию. Программа позволяет выполнять реалистичные симуляции с использованием программ реального робота и файлов конфигурации, идентичных тем, которые используются на производстве[11]. RobotStudio поставляется с полным пакетом функций и надстроек, позволяющих проводить автономное моделирование, снижая риски, ускоряя запуск, сокращая время переналадки и, в конечном итоге, повышая производительность. Интерфейс программы показан на рисунке 2.

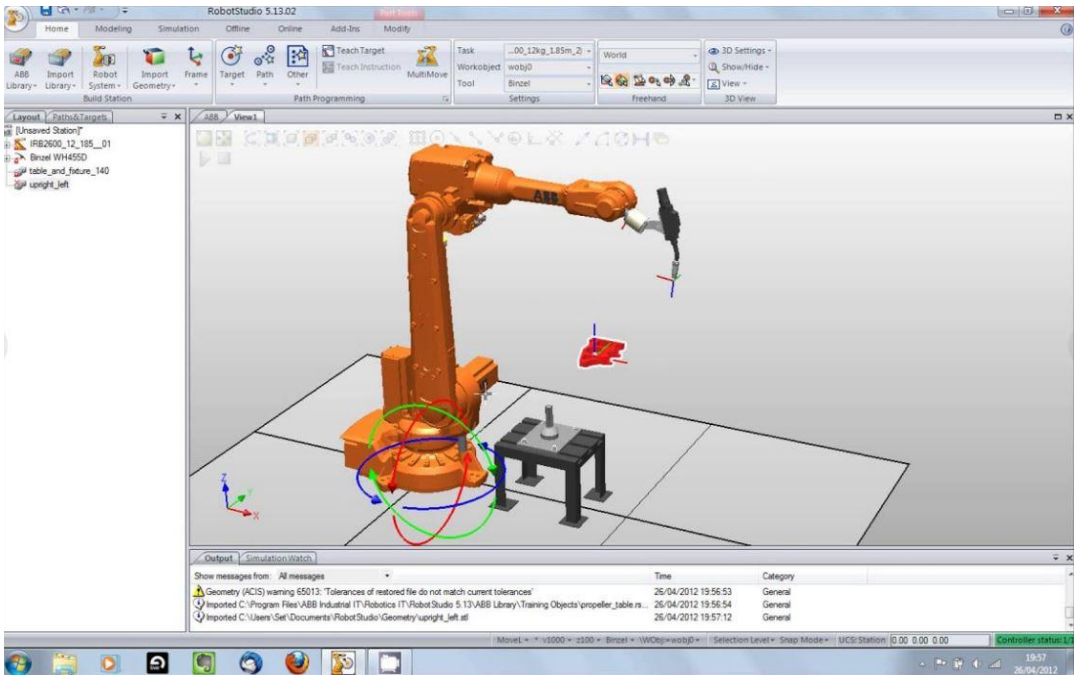

Рисунок 2 – Интерфейс программного обеспечения «RobotStudio»

Данная программа имеет множество функций, например возможность устраивать виртуальные встречи, использовать дополненную реальность и цифровых двойников.

Виртуальные встречи – это функция совместной работы, позволяющая делиться решениями для цифровых роботов на веб-конференциях. Участники погружаются в виртуальную комнату с помощью гарнитуры VR, подключенной к RobotStudio, которую можно совместно использовать для обзора дизайна и предложений по продажам.

Цифровой двойник – это концепция мониторинга и оптимизации решения по автоматизации без прерывания текущего производства, что позволяет моделировать производственную систему в реальном времени[7].

Технология дополненной реальности может визуализировать роботизированные решения, накладывая на реальную производственную среду с использованием очков дополненной реальности, смартфона или планшета.

1.2.2. MotoSim EG-VRC

Yaskawa предоставляет MotoSim EG-VRC для создания и моделирования роботизированную ячейку. MotoSim EG-VRC является улучшенным графическим контроллером виртуального робота для точного автономного программирования сложных систем[12]. С помощью функции управления виртуальным роботом программное обеспечение для моделирования можно использовать для:

1. Оптимизации размещения роботов и оборудования
2. Моделирования охвата
3. Точных расчетов циклов
4. Автоматической генерации пути
5. Обнаружения столкновений
6. Конфигурации системы

7. Конфигурации блока функциональной безопасности (FSU)
8. Удаленного доступа к реальному контроллеру робота

Интерфейс программы представлен на рисунке 3.

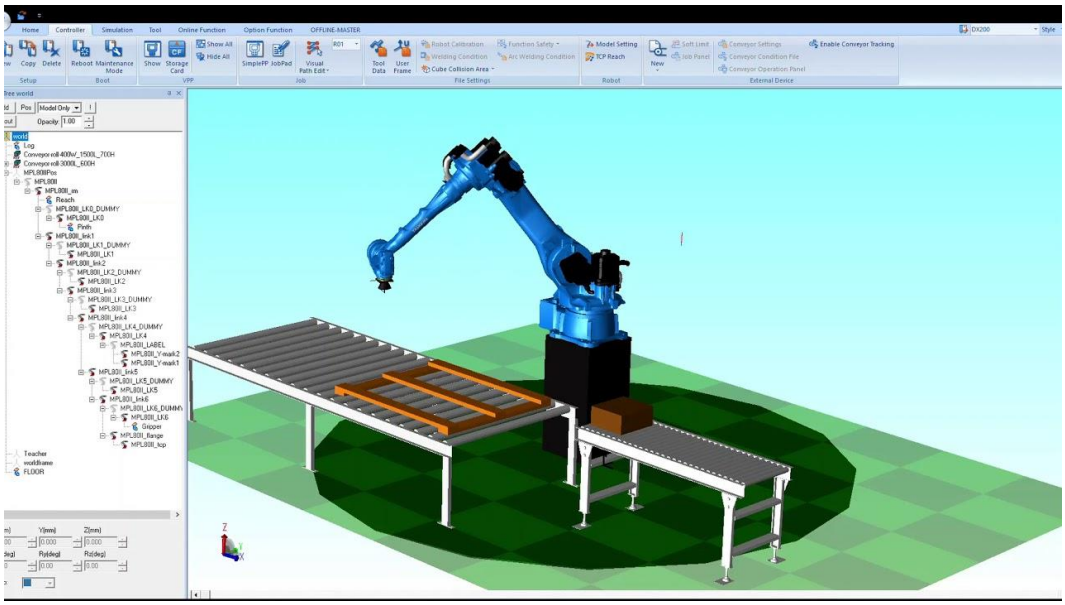

Рисунок 3 – Интерфейс программы MotoSim EG-VRC

Виртуализированный контроллер также позволяет работать и отображать программный интерфейс, идентичный реальному контроллеру. VRC полностью имитирует программное обеспечение контроллера робота Motoman.

1.2.3 Robosim Pro

RoboSim - простая система моделирования для робота-манипулятора с шестью степенями свободы[14]. Основными функциями данного программного обеспечивания являются перемещение манипулятора либо в совместных координатах, либо в декартовых координатах. На рисунке 4 представлен интерфейс данной программы.

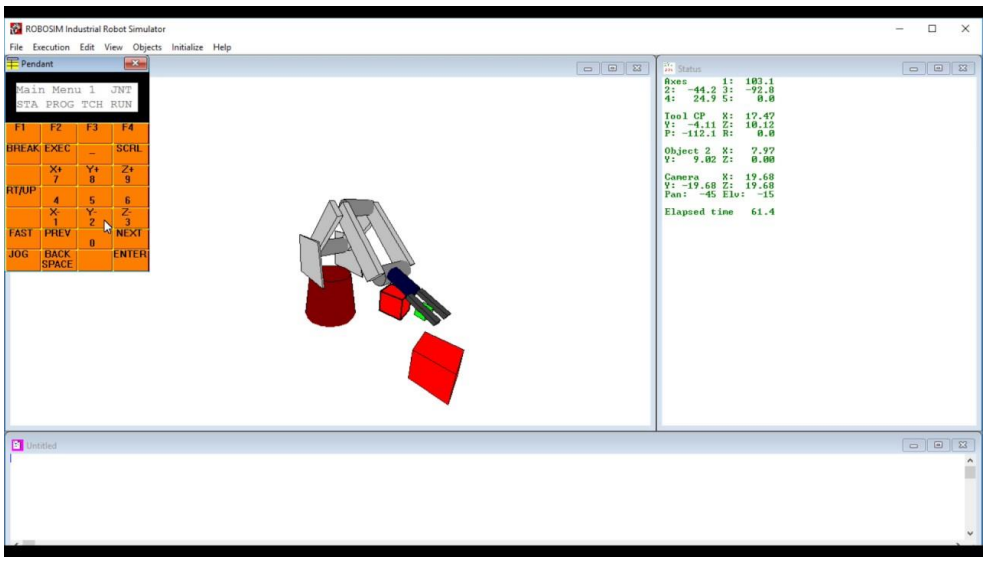

Рисунок 4 - Интерфейс программы Robosim Pro

Система моделирования включает в себя графику и кинематику манипулятора (прямая и обратная кинематика).

1.2.4. KUKA Sim Pro

KUKA Sim Pro является интеллектуальным программным обеспечением для моделирования эффективного автономного программирования роботов KUKA[13]. С данным программным обеспечением можно оптимизировать работу систем и роботов вне производственной среды. Интерфейс среды показан на рисунке 5.

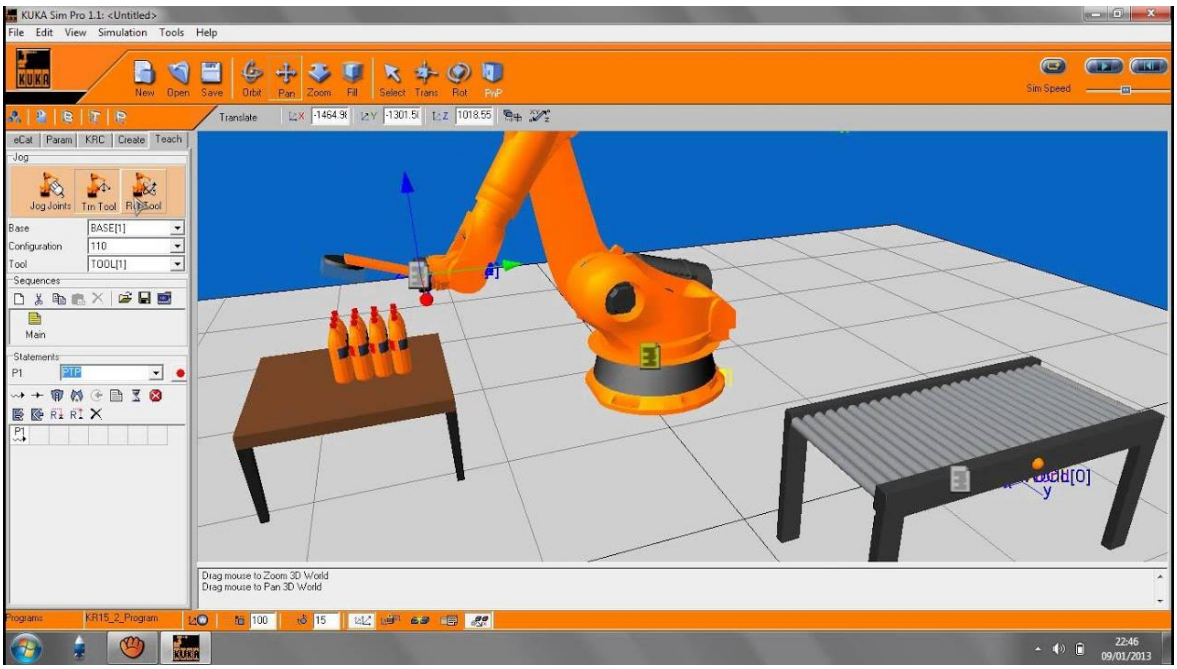

Рисунок 5 - Интерфейс программы KUKA Sim Pro

Программа представляет последовательность движения робота, запрограммированные в автономном режиме, отображает в режиме реального времени, анализируют и оптимизирует с учетом времени их цикла.

KUKA Sim создает цифрового двойника и, таким образом, визуализирует идентичное изображение последующего производственного процесса. 3D-моделирование охватывает весь процесс планирования: от проектирования процесса до визуализации материальных потоков и узких мест до кода ПЛК. Программа создает возможность виртуального ввода в эксплуатацию новых производственных линий, которые можно тестировать и оптимизировать заранее.

1.2.5. K-ROSET

K-ROSET представляет из себя виртуальный симулятор с широким функционалом и большими возможностями. Встроенные функции по обнаружению столкновений, анализу времени цикла и анализу монтажного положения делают этот инструмент проектирования высокоэффективным, позволяющим получать не только высокоточные данные, но и качественные

визуализации робототехнического комплекса, в том числе и для маркетинговых целей. Интерфейс данной программы представлен на рисунке 6.

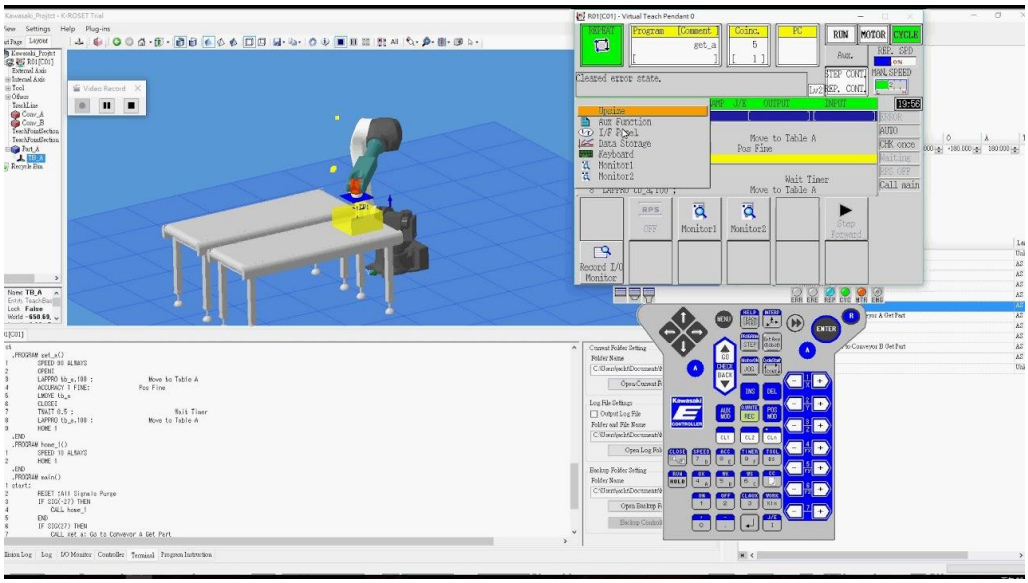

Рисунок 6 - Интерфейс программы K-ROSET

K-ROSET создает точную цифровую копию всего робототехнического комплекса, которая дает проработать в виртуальной среде траекторию движений робота с учетом установленного инструмента, синхронизировав его с дополнительным оборудованием, тем самым минимизировав проектные затраты.

Программа имеет множество функций, например проектирование систем технического зрения и создание точной цифровой копии робота.

Для разработки систем с техническим зрением, в K-ROSET встроена виртуальная камера, которая поможет найти и определить расположение заготовки при проектировании систем укладки.

1.3 ПРОМЫШЛЕННЫЙ МАНИПУЛЯЦИОННЫЙ РОБОТ

Промышленный робот — предназначенный для выполнения двигательных и управляющих функций в производственном процессе манипуляционный робот[18], т. е. автоматическое устройство, состоящее из

манипулятора и перепрограммируемого устройства управления, которое формирует управляющие воздействия, задающие требуемые движения исполнительных органов манипулятора. Многозвенный манипулятор (многосуставный манипулятор) представляет собой манипулятор с несколькими степенями подвижности.

1.3.1. Описание аппаратной части манипулятора

Манипулятор с шестью степенями подвижности, способный выполнить самые различные движения. Звенья манипулятора (рисунок 7) соединяются друг с другом с помощью поворотных шарниров («суставов») и вращаются вокруг осей систем координат, идущих через центры суставов.

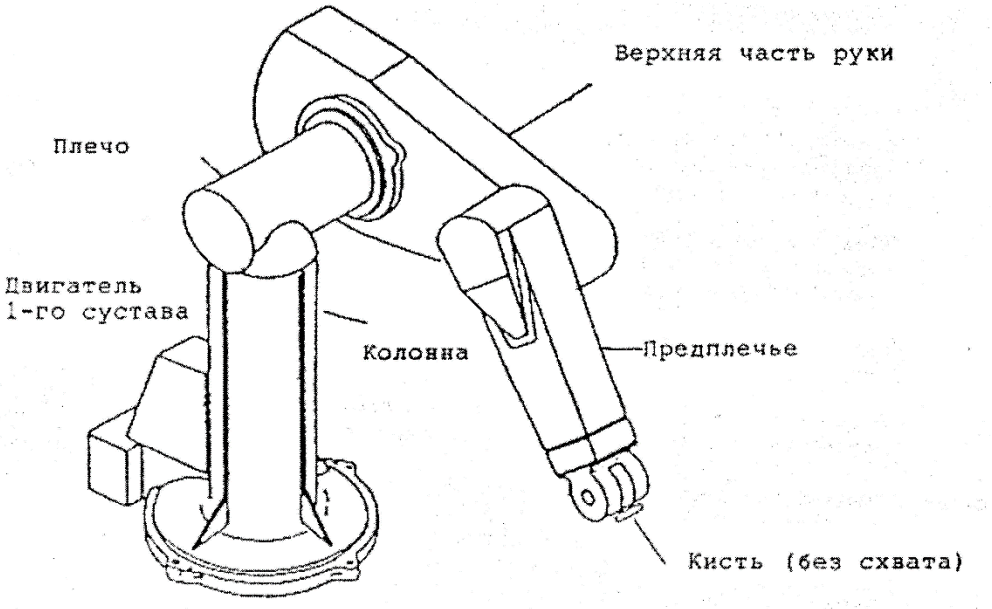

Рисунок 7 – Звенья шестизвенного робота-манипулятора

Каждое звено имеет свой следящий привод постоянного тока с постоянным магнитом. Трансмиссия осуществляется через зубчатые редукторы. Для управления движением манипулятора необходимо постоянно контролировать положение и скорость движения звеньев. Для этого на вал каждого серводвигателя установлены в одном комплекте потенциометр и импульсный фотоэлектрический датчик[10]. Вращение датчика обеспечивается от вала самого двигателя через скользящую муфту. Сигналы

от датчиков индицируют положения звеньев, а скорость вычисляется на основе этик сигналов.

Рассмотрим степени подвижности манипулятора:

1-ая степень подвижности – колонна. Двигатель 1-ой степени подвижности установлен в основании колонны, закрытый металлическим кожухом. На вале двигателя установлено цилиндрическое зубчатое колес, которое с помощью шестерни и ведущего вала передает вращательное движение на цилиндр зубчатого венца.

2-ая степень подвижности – плечо. Передача двухступенчатая. Вал серводвигателя, на котором установлено коническое зубчатое колесо с валиком вращает ведущую шестерню, установленную на промежуточный вал. В другом конце промежуточного вала находится цилиндрическое зубчатое колесо с валиком, вращающее стационарно подкрепленный к плечу зубчатый венец.

3-ая степень подвижности – локоть. Двигатель 3-ей степени подвижности размещен рядом с двигателем 2-ой степени подвижности между плечом и локтем. Передача двухступенчатая. Вал серводвигателя соединен через упругую муфту с ведущим валом, вращающим коническое зубчатое колесо с валиком, которое с помощью ведущей шестерни вращает промежуточный вал. В другом конце промежуточного вала установлено цилиндрическое зубчатое колесо, которое вращает прикрепленный к предплечью зубчатый венец и таким образом вращает все предплечье вокруг локтя.

4-ая, 5-ая и 6-ая степени подвижности – кисть. Двигатели размещены в предплечье у локтя. Передача от двигателей на зубчатые колеса кисти осуществляется через упругие муфты и промежуточные валы.

Схват снабжен пневмоцилиндром двойного действия, осуществляющим сжатие и расжатие губок схвата[20].

1.3.2. Математическое моделирование

К важнейшим задачам, возникающим в процессе разработки и проектирования универсальных промышленных роботов, относятся задачи создания и верификации алгоритмов систем управления манилуляторами[19]. Решение требует исследования структуры, геометрии и кинематики механической системы с пространственными многоподвижными механизмами.

Предметом исследования кинематики манипулятора является описание геометрии движения манипулятора относительно некоторой заданной абсолютной системы координат без учета сил и моментов, порождающих это движение. Здесь выделяют две основные задачи. Первая из них, называемая прямой задачей, или задачей анализа, заключается в поиске закона изменения абсолютных координат характеристической точки TCP (tool-center-point) на выходном звене манипулятора по заданным законам изменения относительных или абсолютных присоединенных координат сочленений звеньев. Вторая задача, называемая обратной, или задачей синтеза, состоит в определении по требуемым траекториям изменения положения характеристической точки выходного звена соответствующих им изменений углов сочленений в приводах манипулятора.

Прямая задача кинематики позволяет вычислить, где находится рабочий орган манипулятора, зная лишь длины звеньев, а также углы поворота приводов соединяющих звеньев. В качестве примера можно рассмотреть конструкцию, состоящую из двух приводов и двух звеньев, способную к движению на плоскости. На рисунке 8 показана схематичная модель двухзвенного манипуляционного робота.

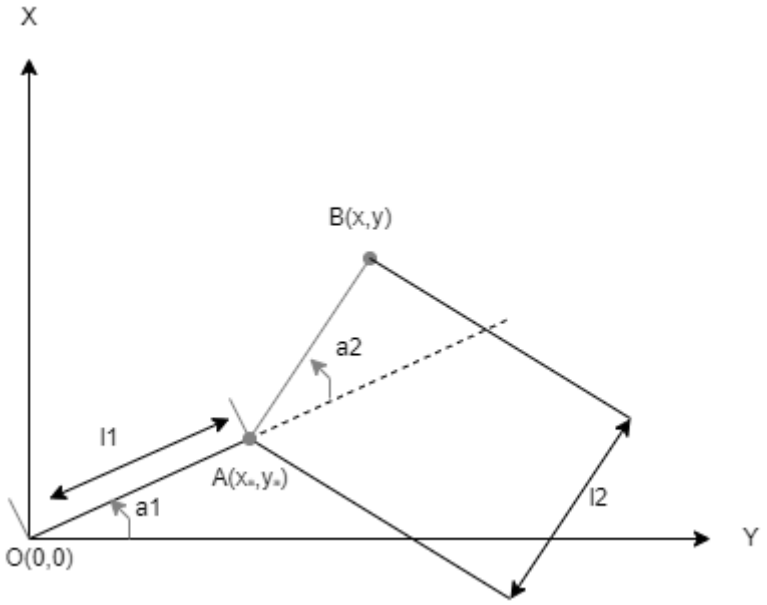

Рисунок 8 – Схема двухзвенного манипуляционного робота

На схеме изображено два звена, показанных отрезками OA и AB, два шарнира (в т. O и в т. A). Также известны длинны l1, l2 и углы a1, a2. Необходимо найти координаты точки в точки B при такой конфигурации манипулятора. Для начала необходимо отыскать координаты всех приводов в относительной СК и сложить результаты.

Для того, чтобы найти координаты точки A используется формула (1)

$$\begin{aligned} x_A &= l_1 \ * \ cos(\alpha_1) \\ y_A &= l_1 \ * \ sin(\alpha_1) \end{aligned} \tag{1}$$

В системе координат x’, y’ (рисунок 9) можно найти координаты т. В

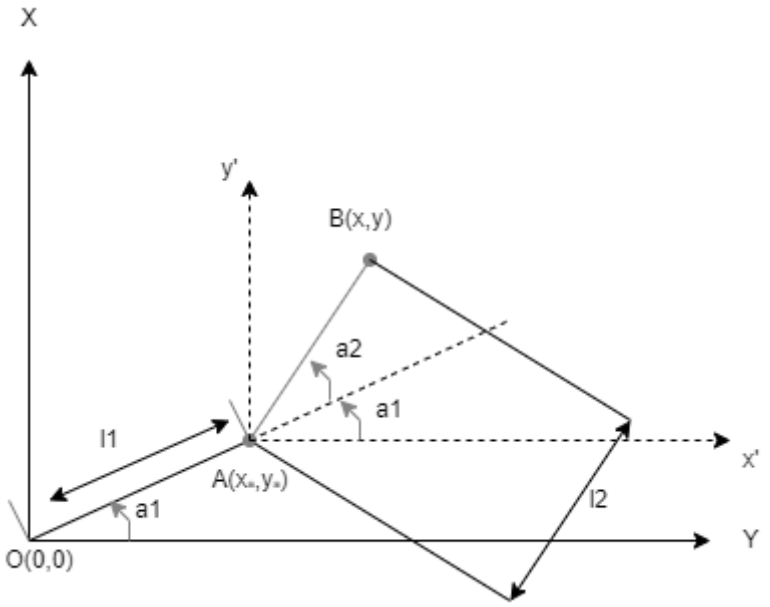

Рисунок 9 - Схема манипуляционного робота в системе координат x’, y’

Таким образом, ищутся координаты т. В относительно т. А по формуле (2)

$$x_{BA} = l_2 * \cos(\alpha_2 + \alpha_2)$$
$$y_{BA} = l_2 * \sin(\alpha_2 + \alpha_2) \quad (2)$$

Чтобы выяснить глобальные координаты т. В необходимо сложить соответствующие составляющие по x и y по формуле (3)

$$x = x_A + x_{BA} = l_1 * \cos(\alpha_1) + l_2 * \cos(\alpha_1 + \alpha_2)$$
$$y = y_A + y_{BA} = l_1 * \sin(\alpha_1) + l_2 * \sin(\alpha_1 + \alpha_2) \quad (3)$$

Обратная задача кинематики используется в случае, когда необходимо знать точные углы звеньев робота-манипулятора, чтобы он в нужное положение и поместил рабочий орган в точную позицию.

Пример решения обратной задачи рассмотрен на примере предыдущей модели манипуляционного двухзвенного робота. На рисунке 10 показано положение рабочего органа в т. В, в которое должен добраться робот-манипулятор из положения, рассмотренного в прямой задаче кинематики.

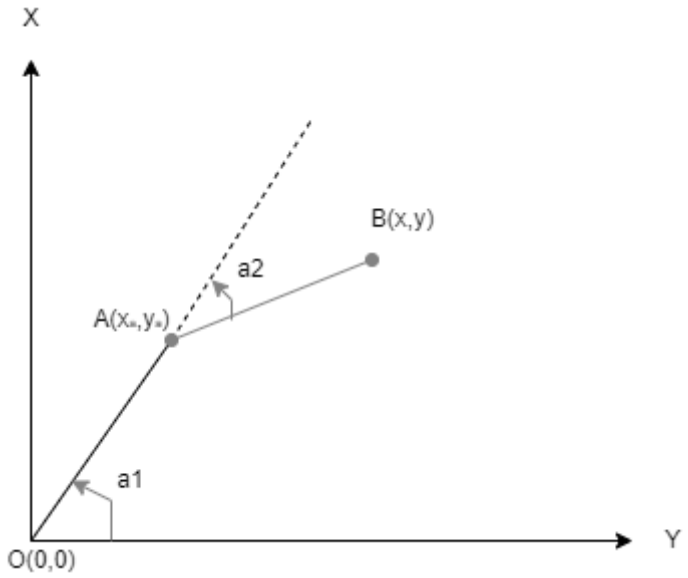

Рисунок 10 – Положение рабочего манипулятора

Для дальнейшего решения необходимо провести дополнительный отрезок d, как на рисунке 11.

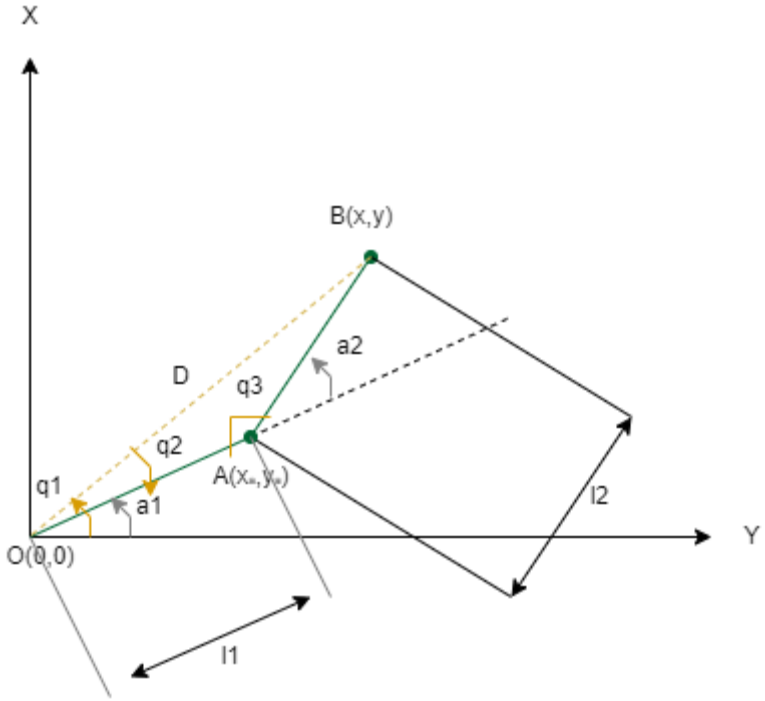

Рисунок 11 – Приведение схемы в решение обратной задачи кинематики

Используя теорему Пифагора и косинусов, можно выявить формулу (4) для вычисления углов α_1 и α_2

$$\begin{cases} \alpha_1 = q_1 + q_2 \\ \alpha_2 = \pi - q_3 \end{cases} \quad (4)$$

Таким образом, на рисунке 12 представлен конечный результат.

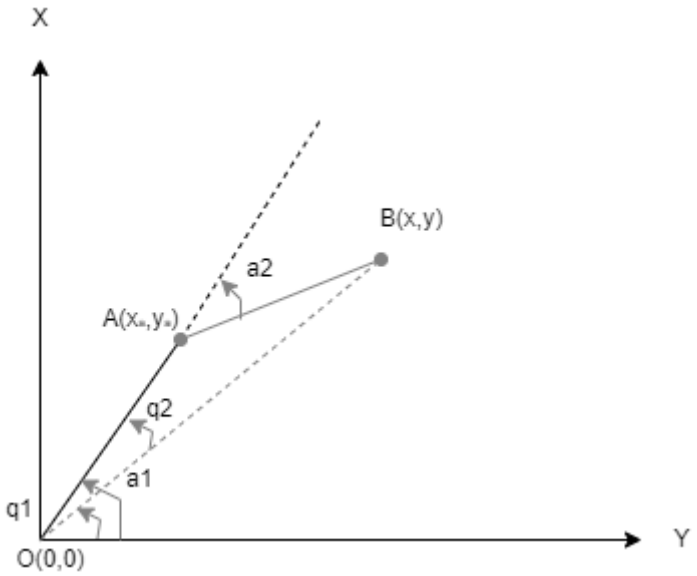

Рисунок 12 – Решение задачи обратной кинематики

1.3.3. Задачи кинематики шестизвенного робота манипулятора

Положение и ориентация твердого тела в пространстве определяется шестью координатами, тремя линейными (Декартовыми) и тремя угловыми (углами Эйлера)[16]. Использование метода, предложенного учеными Жаком Денавитом и Ричардом Хартенбергом позволяет сократить это число до четырех параметров называемыми параметрами Денавита-Хартенберга. Такое упрощение достигается с помощью стандартизированного алгоритма привязки систем координат к звеньям манипулятора.

Решение прямой задачи кинематики состоит из привязки систем координат к звеньям робота, определения параметров Денавита-Хартенберга и построения матриц однородного преобразования.

Для расстановки оси Z_i необходимо, чтобы ось Z_i совпала с осью вращения последующего сочленения (т.е. i+1), для того, чтобы относительное расположение смежных звеньев определялось переменной вокруг этой оси.

Для выбора оси X_i необходимо, чтобы ось была перпендикулярна оси Z_{i-1} , а также, чтобы она пересекала ее. При определении оси Y_i система координат, заданная единичными векторами X, Y, Z была направлена вправо.

Данный метод позволяет сократить количество координат, определяющих систему координат в пространстве, известных как параметры Денавита-Хартеберга, где:

1. a_i – расстояние вдоль оси x_i от z_{i-1} до z_i ;
2. α_i – угол вокруг оси x_i от z_{i-1} до z_i ;
3. S_i – расстояние вдоль оси z_{i-1} от x_{i-1} до x_i;
4. Q_i – угол вокруг оси z_{i-1} от x_{i-1} до x_i.

Параметры Денавита-Хартенберга для шестизвенного робота-манипулятора представлены на рисунке 13.

Параметр	Описание параметра	Звенья					
		1	2	3	4	5	6
a_i	$R(z_{i-1};\ z_i)$	0	L_2	0	0	0	0
α_i	$\angle(z_{i-1};\ z_i)$	$\pi/2$	0	$\pi/2$	$-\pi/2$	$\pi/2$	0
S_i	$R(x_{i-1};\ x_i)$	L_1	0	0	L_4	0	L_6
Q_i	$\angle(x_{i-1};\ x_i)$	q_1	q_2	$q_3+\pi/2$	q_4	q_5	q_6

Рисунок 13 – Параметры Денавита-Хартенберга

При решении прямой задачи кинематики рассматриваются 2 системы координат: базовая, связанная с основанием $O_0X_0Y_0Z_0$, и итоговая, связанная со схватом или рабочим инструментом $O_nX_nY_nZ_n$. Для решения необходимо определить матрицу (5), которая информирует о линейном смещении и пространственной ориентации одной системы относительно другой.

$$A_n^0 = \begin{bmatrix} n_n^0 & s_n^0 & a_n^0 & p_n^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_n^0 & p_n^0 \\ 0 & 1 \end{bmatrix}. \quad (5)$$

Векторы n_n^0, s_n^0 и a_n^0 отображают направление осей x_n^0, y_n^0 и z_n^0 относительно системы координат $O_0X_0Y_0Z_0$, R_n^0 - матрица вращения системы $O_0X_0Y_0Z_0$относительно $O_nX_nY_nZ_n$, p_n^0 – вектор линейного смещения начала $O_0X_0Y_0Z_0$.

Из полученных четырех параметров Денавита-Хартенберга для каждого звена манипулятора необходимо построить матрицы однородного преобразования, как представлено в формуле (6)

$$A_i = A_{z,\theta_i} * A_{z,S_i} * A_{x,a_i} * A_{z,\alpha_i} = \begin{bmatrix} R_{x,\theta_i} & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} I & P_{S_i} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} I & P_{a_i} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} R_{x,\alpha_i} & 0 \\ 0 & 1 \end{bmatrix}, \quad (6)$$

где i - номер звена, R_{x,θ_i} и R_{x,α_i} - базовые матрицы вращения, I – единичная матрица, а P_{S_i} и P_{a_i} - векторы линейных перемещений.

Подставив все параметры Денавита-Хартенберга, выявляются шесть матриц (7) – (12):

$$A_1^0 = \begin{pmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & 1 & 0 & \mathrm{L}_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (7)$$

$$A_2^1 = \begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & \mathrm{L}_1 * \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & \mathrm{L}_1 * \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (8)$$

$$A_3^2 = \begin{pmatrix} -\sin(q_3) & 0 & \cos(q_3) & 0 \\ \cos(q_3) & 0 & \sin(q_3) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (9)$$

$$A_4^3 = \begin{pmatrix} \cos(q_4) & 0 & -\sin(q_4) & 0 \\ \sin(q_4) & 0 & \cos(q_4) & 0 \\ 0 & -1 & 0 & \mathrm{L}_4 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (10)$$

$$A_5^4 = \begin{pmatrix} \cos(q_5) & 0 & \sin(q_5) & 0 \\ \sin(q_5) & 0 & -\cos(q_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (11)$$

$$A_6^5 = \begin{pmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ \sin(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & \mathrm{L}_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (12)$$

Итоговую матрицу, связывающую все системы координат, как и в случае с матрицами вращения, можно получить последовательным перемножением матриц однородного преобразования.

Кинематический анализ заключается в возможности применения кинематической декомпозиции, состоящей из двух частей: обратная задача кинематики по положению и ориентации.

Геометрический метод решения обратной задачи кинематики заключается в нахождении аналитических выражений в явном виде с использованием аппарата тригонометрических функций с учетом кинематической схемы[15]. Для того, чтобы рассчитать координаты рабочего органа манипулятора необходимо знать три линейные координаты, три угловые координаты и фиксированные параметры Денавита-Хартенберга. Для примера решения обратной задачи кинематики рассмотрим рисунок 14.

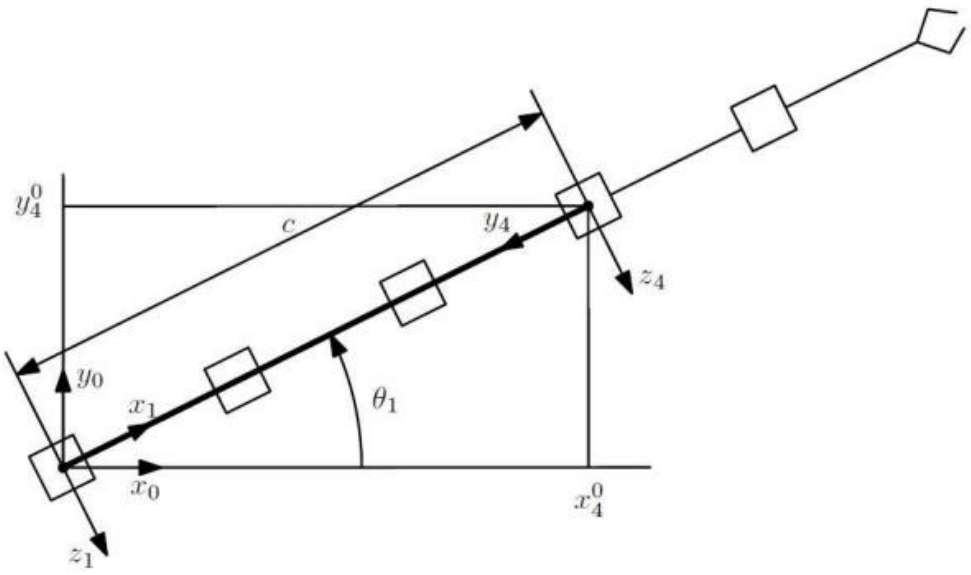

Рисунок 14 – Кинематическая схема видом сверху

Проанализировав схему, можно определить первую обобщенную координату с учетом разворота манипулятора в первом сочленении на половину оборота по формуле (13)

$$q1 = atan2(Y_4^0, X_4^0) + \pi. \quad (13)$$

Рассмотрим рисунок 14 и 15 для определения длин a, b и c.

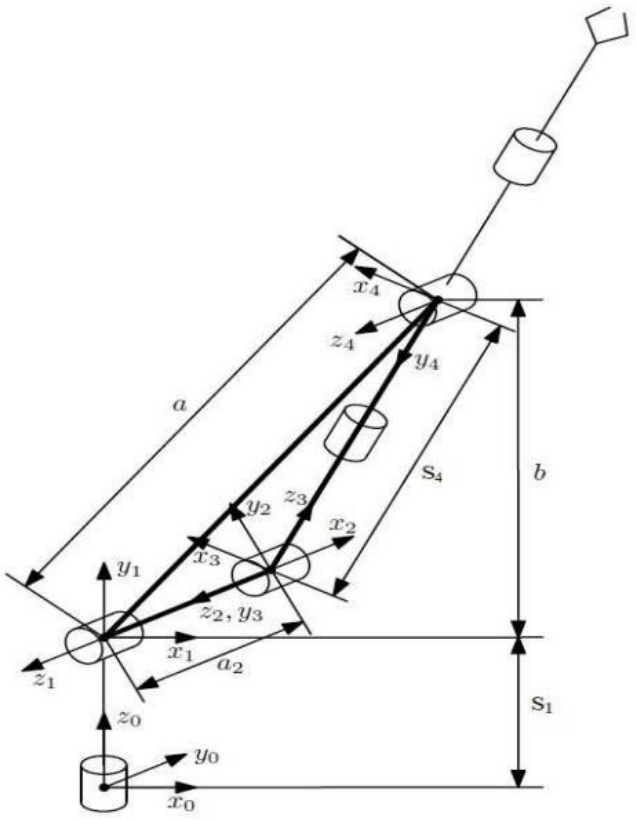

Рисунок 15 – Кинематическая схема видом сбоку

Длины данных отрезков высчитываются по формулам (15) – (17)

$$a = \sqrt{(x_4^1)^2 + (y_4^1)^2 + (z_4^1)^2}, \tag{15}$$

$$b = (z_4^0 - d), \tag{16}$$

$$c = \sqrt{(x_4^0)^2 + (y_4^0)^2}. \tag{17}$$

Используя теорему Пифагора и косинусов, выразим косинус угла θ по формуле (18)

$$cos\theta = \frac{b^2 + c^2 - a_2^2 - d_4^2}{2a_2d_2}. \tag{18}$$

Также можно выразить синус угла θ используя основное тригонометрическое тождество с помощью формулы (19)

$$sin\theta_3 = \pm\sqrt{1 - cos^2\theta_3}. \quad (19)$$

Таким образом обобщенная координата θ_3 высчитывается по формуле (20)

$$\theta_3 = atan2(\pm\sqrt{1 - cos^2\theta_3}, cos\theta_3. \quad (20)$$

Для определения угла θ_2 необходимо найти разность углов α и β. Используя расчет тригонометрических выражений, выявим формулу (21)

$$\theta_3 = atan2(b, c) - atan2(d_4 sin\theta_3, a_4 + d_4 cos\theta_3). \quad (21)$$

Чтобы найти оставшиеся углы $q4$, $q5$ и $q6$ воспользуемся решением обратной задачи кинематики по ориентации. Комбинация последовательных вращений вокруг текущих осей определяется с помощью умножения с правой стороной по формулу (22)

$$R_6^0 = R_3^0 R_6^3 . \quad (22)$$

С учетом сферической конструкции запястья, последние три звена обеспечивают ориентацию рабочего органа по матрице R_6^3, формируя матрицу, описанную в формуле (23)

$$R_6^0 = R_{zyz} = R_{z,\varphi} R_{y,\theta} R_{z,\omega} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (23)$$

Таким образом углы $q4$, $q5$ и $q6$ рассчитываются по формулам (24)-(26)

$$\theta = atan2\left(\pm\sqrt{1 - r_{33}^2}, r_{33}\right), \quad (24)$$

$$\varphi = atan2(\pm r_{23}, \pm r_{13}), \quad (25)$$

$$\omega = atan2(\pm r_{32}, \pm r_{31}). \quad (26)$$

1.4. ВЫВОД

В данном разделе был рассмотрен обзор аналогов, а также теоретическая составляющая управления роботом-манипулятором. Для поставленной задачи существует мало аналогов, применяющих технологию дополненной реальности, которые бы в полной мере отвечали нынешним потребностям. Также был рассмотрен математический анализ расчета прямой и обратной кинематической задачи шестизвенного робота-манипулятора.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Для определения функциональных требований следует выделить основных участников, взаимодействующих с системой: клиенты, взаимодействующие с системой, серверная часть и аппаратная часть (рисунок 16).

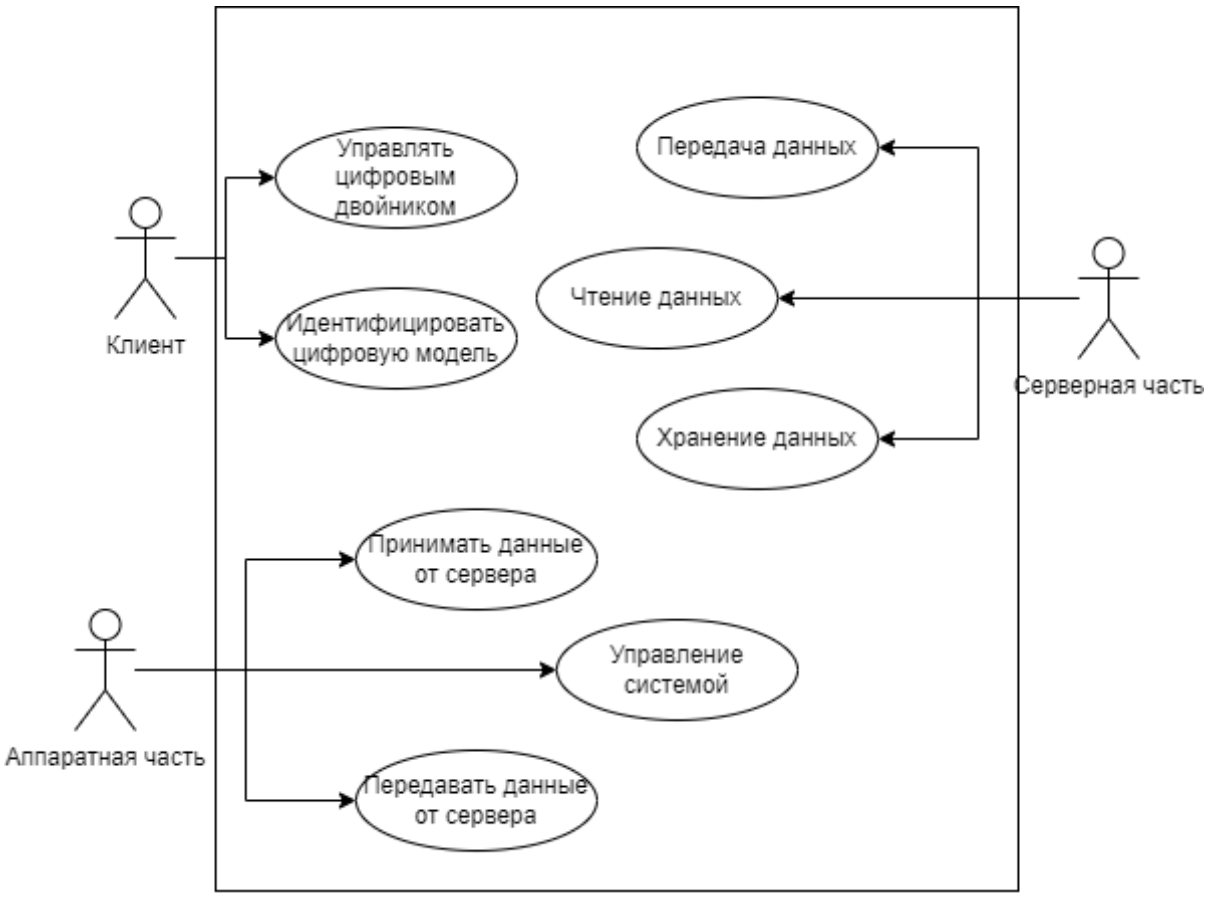

Рисунок 16 – Диаграмма вариантов использования системы управления манипуляционным роботом

Клиент – компонент, управляющий цифровым двойником манипуляционного робота

Серверная часть – компонент программно-аппаратного комплекса, выполняющий передачу, чтение и хранение данных.

Аппаратная часть – компонент, выполняющий управление роботом-манипулятором, получая и отправляя данные серверной части.

Перечень задач разрабатываемой системы:

1. Система управления всеми звеньями робота-манипулятора.
2. Отображение цифровой модели робота-манипулятора в реальном окружении.
3. Передача углов наклона каждого звена.
4. Вычисление положения рабочего органа.
5. Получение текущего положения звеньев робота-манипулятора.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Основываясь на функциональные требования системы можно выделить следующий перечень нефункциональных требований:

1. Приложение должно работать на мобильных устройствах с операционной системой Android.
2. Обеспечено корректное отображение цифровой модели робота-манипулятора.
3. Информирование пользователя о результате передачи данных манипуляционному роботу.

2.3. ВЫВОД

В разделе был рассмотрен список требований к приложению. К функциональным требованиям относятся управление всеми звеньями цифровой модели робота-манипулятора, передача необходимых параметров для установления новой позиции рабочего органа и получение текущего положения манипуляционного робота.

К нефункциональным требованиям были определены корректность отображения цифровой модели в реальной среде и удобство взаимодействия клиента с системой на мобильном устройстве.

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРОЕКТА В СРЕДЕ РАЗРАБОТКИ UNITY.

Клиентская часть приложения использует две платформы, подключаемые в среде разработки Unity: Vuforia Engine и ROS-TCP-Connector.

Vuforia Engine – это платформа дополненной реальности и инструментарий разработчика программного обеспечения дополненной реальности для мобильных устройств[9]. Для отображения виртуальных объектов используются два вида трекинга: на основе метки-изображении и на основе поиска плоскостей.

Архитектура взаимодействия среды разработки Unity с системой меток представлена на рисунке 17.

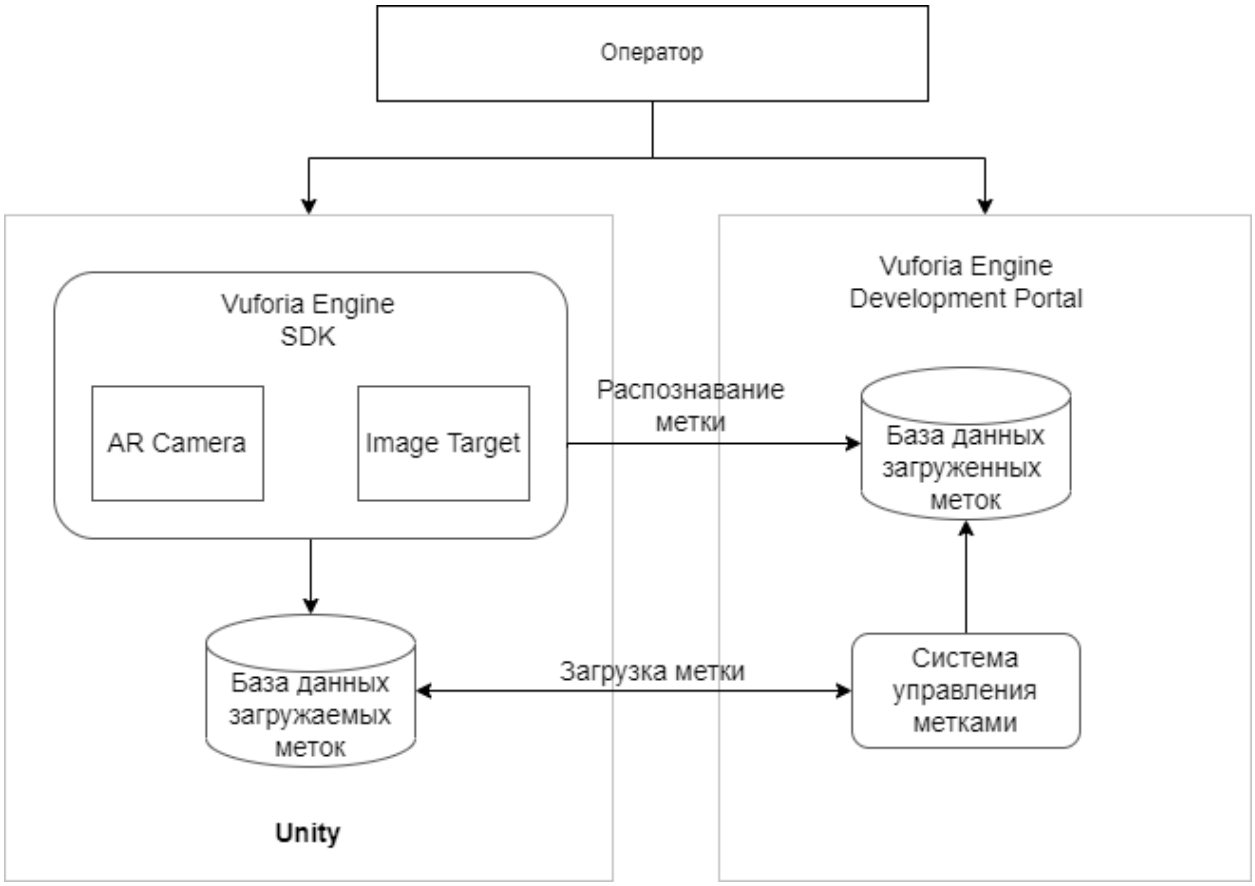

Рисунок 17 – Архитектура платформы Vuforia Engine

Данный способ отображения виртуальных объектов является наиболее стабильным и эффективным, т.к. имеется возможность загружать свои изображения в облачное хранение веб-ресурса Vuforia Engine, благодаря чему, распознавание метки осуществляется очень быстро, исключая погрешность в позиционировании виртуального объекта. Метод распознавания плоскости является менее эффективным, т.к. сервером платформы необходимо применять вычислительные действия заранее неизвестной области. В связи с чем, было принято решение использовать маркерный способ отображения элементов дополненной реальности.

Пользователь загружает собственные изображения в облачное хранилище фреймворка и устанавливает передачу данных между ним и средой разработки Unity. Для взаимодействия Unity с фреймворком подключается дополненная камера, позволяющая сканировать и распознавать изображения, хранящиеся в компоненте ImageTarget. Данный компонент необходим для того, чтобы задать определенное поведение для каждой метки и отобразить соответствующий виртуальный объект.

ROS-TCP-Connector – комплекс инструментов, предназначенный для обмена сообщениями со средой интеграции ROS. ROS (Robot Operating System) является операционная система для роботов, предоставляющая функциональность для распределённой работы[17]. Архитектура данного фреймворка изображена на рисунке 18.

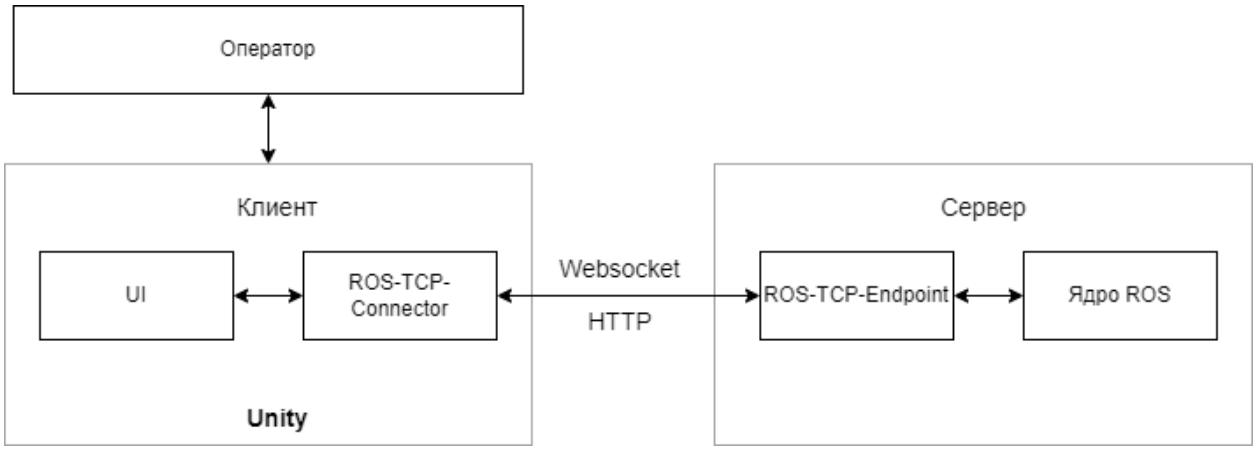

Рисунок 18 – Архитектура платформы ROS-TCP-Connector

Пользователь, взаимодействуя с приложением, осуществляет обмен данными через компонент ROS-TCP-Connector. Данный компонент устанавливает связь с сервером через протокол HTTP и формирует пакет с передаваемыми данными по веб-сокету TCP/IP.

Протокол HTTP однонаправленный и используется в момент установления связи с сервером. После цикла «запрос — ответ» соединение закрывается, а любой следующий запрос каждый раз устанавливает новое соединение с сервером. Процесс передачи данных происходит с некоторыми задержками за счет того, что есть накладные расходы на установку нового соединения при каждой передаче пакета, а также сетевая и серверная нагрузка из-за обилия периодических запросов.

Протокол WebSocket двунаправленный, полнодуплексный, позволяющий одновременно и получать, и передавать информацию. Веб-сокет делает это множество раз в одном открытом соединении. У такого соединения и скорость выше, чем у HTTP, что осуществит быстрое управление роботом-манипулятором.

Принимающей стороной пакета является компонент ROS-TCP-Endpoint. Он принимает данные о положении рабочего органа манипуляционного робота, заданного пользователем и передает их системе интеграции ROS. Операционная система выполняет команду по управлению робота-манипулятора с соответствующими параметрами, что позволяет переместиться реальному роботу в точку, заданную пользователем для цифрового двойника.

3.2. АРХИТЕКТУРА ROS

Структура системы разделяется на три основные модуля, рассмотренные на рисунке 19.

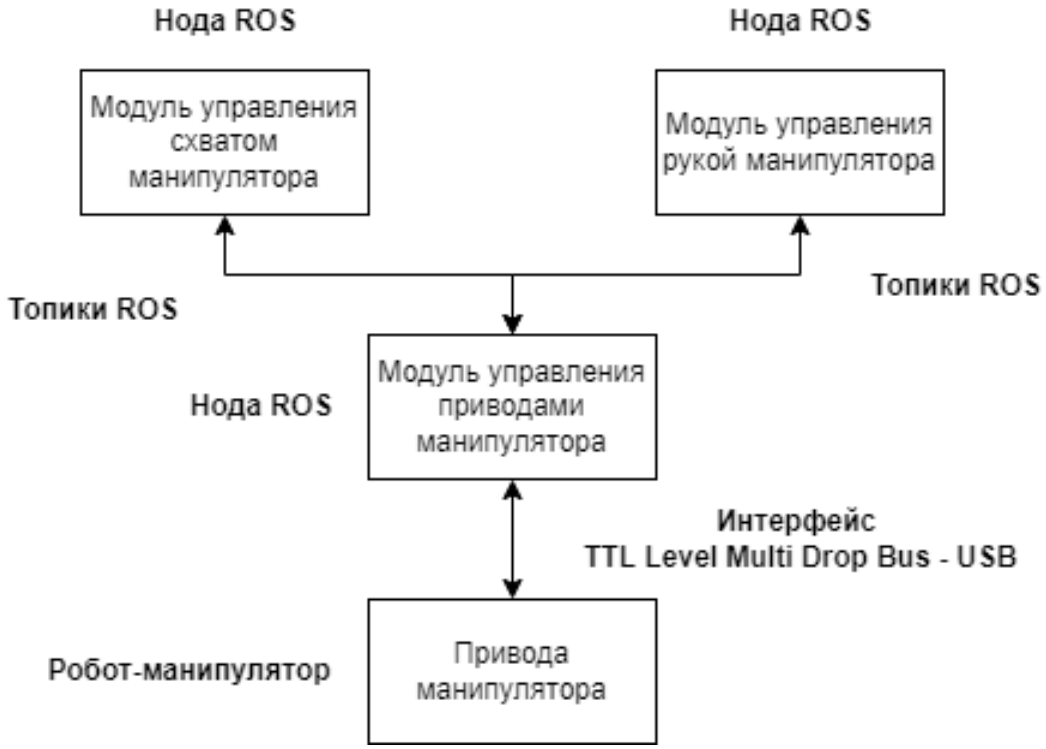

Рисунок 19 – Структура системы управления в ROS

Модуль управления рукой манипулятора предназначен для вычисления прямой и обратной позиционной задачи кинематики. В нем содержится код, формирующий и отображающий траекторию движения схвата, а также производящий преобразование координат показаний приводов в радианы и обратно. Помимо этого, в данном блоке реализуются методы, позволяющие вызывать команды по движению схвата и контролируя ее результат.

Модуль управления схватом реализует команды, позволяющие контролировать сжатие и расжатие объектов, при этом обеспечивает обратную связь для информирования пользователя о том, находится ли объект в схвате.

Модуль управления приводами манипулятора взаимодействует с готовыми библиотеками по управлению сервоприводами Dynamixel. Данный модуль получает данные, настраивает параметры устройства в управляющем контуре с обратной связью (ПИД-регулятор), получает значения с датчиков температур и тока сервоприводов с использованием порта USB.

На рисунке 20 представлена структура пакета управления роботом-манипулятором, поддерживающий режим симуляции.

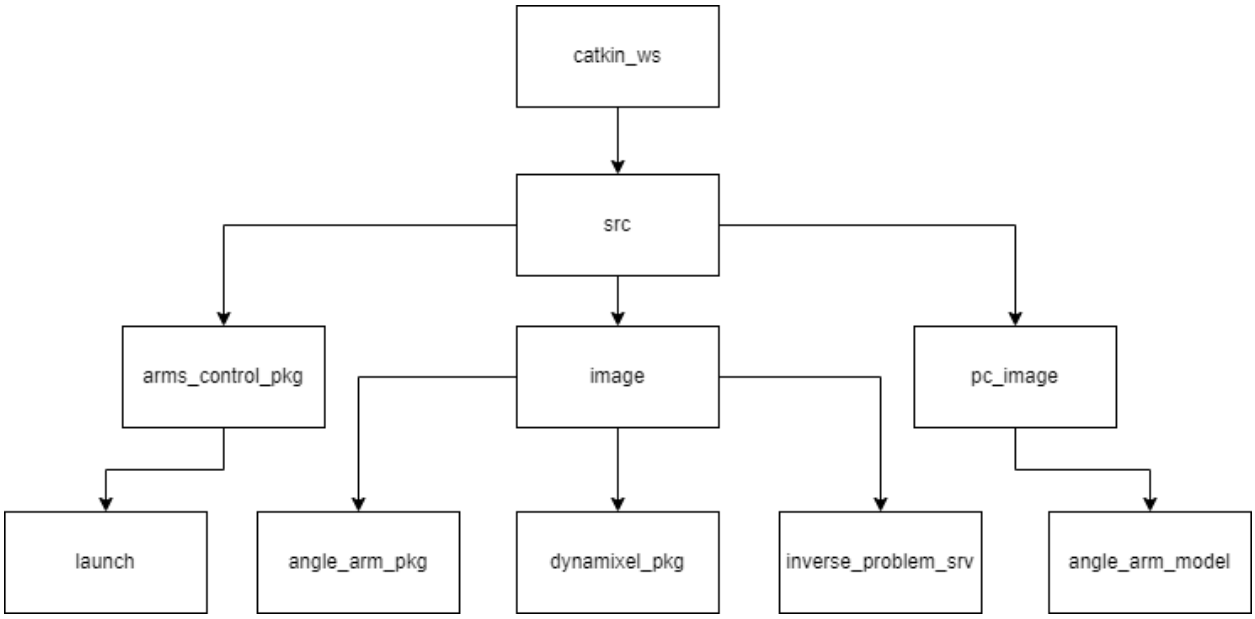

Рисунок 20 – Структура пакета ROS

Рассмотрим подробнее содержание пакетов:

1. В папке launch находится файл arms_launch_control.launch, позволящий запустить симуляцию робота в редакторе Rviz.

2. angle_arm_pkg является пакетом для управления рукой и схватом роботом-манипулятором.

3. dynamixel_pkg – пакет для управления сервоприводами Dynamixel, а также содержит пакеты Dynamixel SDK.

4. inverse_problem_srv предназначен для описания своего типа сообщений для сервисов

5. Пакет angle_arm_model позволяет взаимодействовать системе с визуализацией в Rviz.

Данная схема одинакова, как и для реального робота, так и для модели в симуляции.

3.3. АРХИТЕКТУРА УПРАВЛЕНИЯ РОБОТОМ-МАНИПУЛЯТОРОМ

Система управления манипуляционным роботом представляет собой многоуровневую структуру, состоящей из шести уровней: технический

(драйверы устройств), предметный (алгоритмы обработки сигналов датчиков и построения движений), процедурный (алгоритмы мониторов систем), системный (операционная система, программный контроллеров), уровень обучения (языки и алгоритмы обучения роботов) и интеграции с системами автоматического управления (протоколы передачи данных).

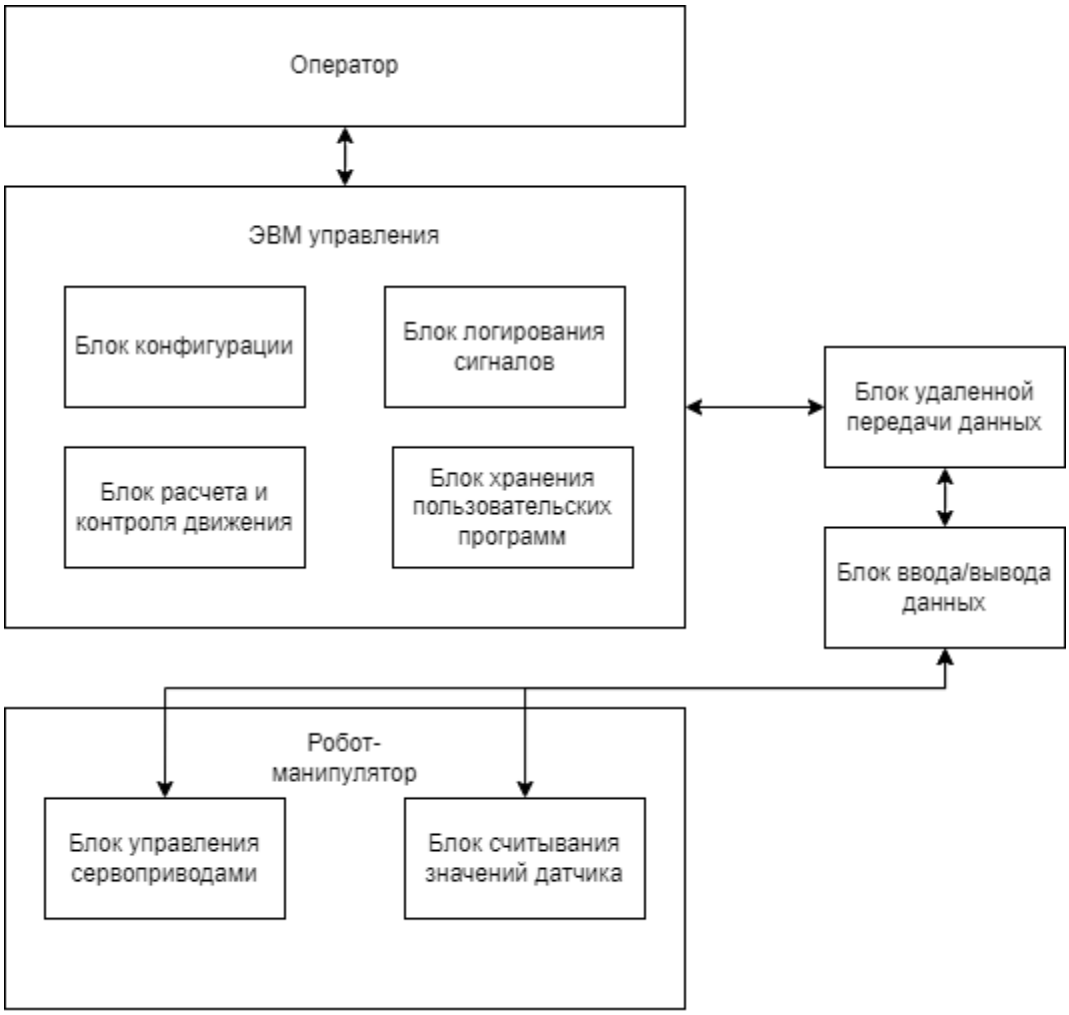

Рисунок 21 – Архитектура системы управления роботом-манипулятором

На рисунке 21 изображена обобщенная схема программного обеспечения манипуляционного робота.

Блок конфигурации отвечает за набор скалярных параметров, которые определяют положение всех точек робота относительно некоторой фиксированной системы координат.

Логирование предназначено для отслеживания событий, которые происходят при запуске программного обеспечения. Данный модуль добавляется для указания, что произошли определенные события.

За расчет движения с помощью обратной и прямой задачи кинематики, а также контролирования смены положения отвечает соответствующий блок.

Блок хранения пользовательских программ позволяет разработчикам загружать свои программы для того, чтобы выполнить их при запуске манипуляционного робота.

Совокупность всех представленных данных передаются в блок, который отвечает за обмен сообщениями.

Блоки ввода/выводы обеспечивают вызов команд и параметров, а также вывод полученного результата. Данный модуль передает и получает параметры сервоприводов и значений датчиков.

Блок удаленной передачи и ввода-вывода реализуются как на контроллере манипуляционного робота, так и в управляющем вычислительном комплексе.

3.4. ВЫВОД

В данном разделе было проведено проектирование приложения. В часть проектирования входило описание и анализ архитектуры платформы Vuforia Engine для подключения дополненной реальности. Помимо данного фреймворка, использовалась платформа ROS-TCP-Connector для взаимодействия с системой интеграции ROS через среду разработки Unity. Заданные параметры положения манипуляционного робота в дополненной реальности передаются через данный фреймворк для дальнейшего управления реального робота-манипулятора. Кроме того, была рассмотрена структура блока управления в ROS и структура воркспейса. С помощью нее производится передача данных в контроллер манипуляционного робота, а также выполняются вычислительные алгоритмы для реализации управления.

4. РЕАЛИЗАЦИЯ

4.1. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА

При запуске приложения пользователю отображается окно с возможностью подключиться к серверу ROS по IP. Изображение интерфейса представлено на рисунке 22.

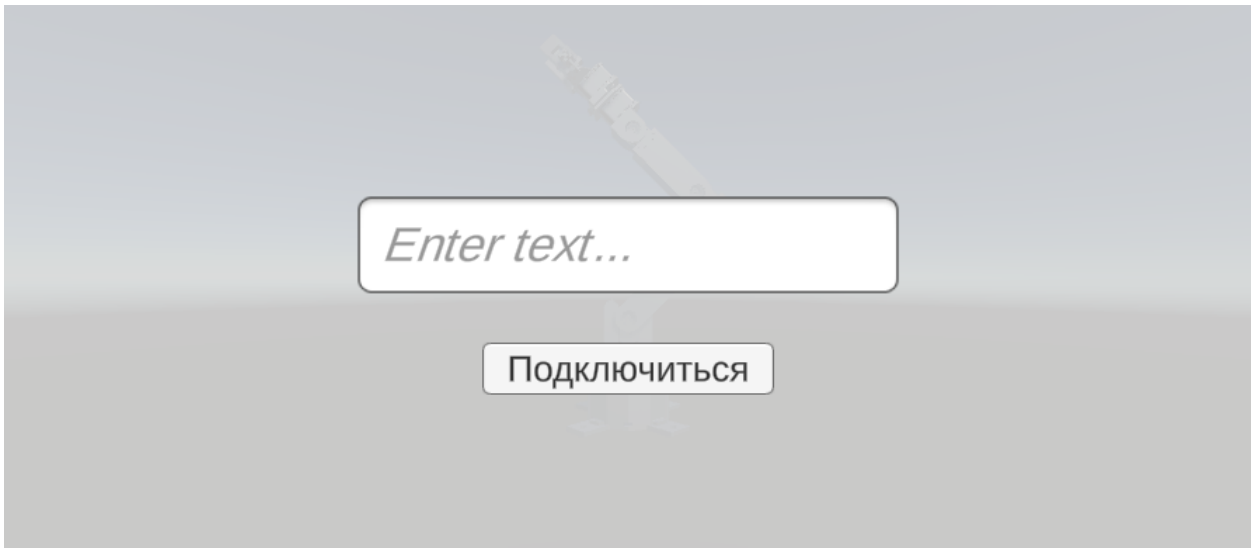

Рисунок 22 – Интерфейс подключения к серверу

Меню подключения состоит из окна ввода и кнопки. В окно ввода пользователь вводит IP сервера, к которому он хочет подключиться и, при нажатии на кнопку, он переходит в экран самого приложения.

После того, как пользователь выполнил подключение, перед ним появляется основной интерфейс приложения (рисунок 23).

Рисунок 23 – Основной интерфейс приложения

Вместо виртуального окружения с роботом-манипулятором, у пользователя включается камера дополненной реальности, а отображения манипуляционного робота производится только в том случае, если камера обнаружила соответствующую метку-изображение. В качестве метки служит обложка документации к данному роботу-манипулятору.

Слева от экрана отображено шесть слайдеров. Слайдер – это элемент графического интерфейса, представляющий собой динамический блок, который позволяет выбрать числовое значение из заданного диапазона путем перетаскивания мышки или с помощью пальца на экране мобильного устройства.

Каждый слайдер содержит сверху текст, который обозначает каким звеном робота управляет данный ползунок. Последний слайдер отвечает за раскрытие и закрытие схвата.

После того, как пользователь изменил положение манипуляционного робота с помощью слайдеров, он отправляет данные на сервер с помощью кнопки «Отправить». Перед ним появляется окно результата передачи данных (рисунок 24).

Рисунок 24 – Панель обратного сообщения

В качестве обратного сообщения от сервера пользователь получает ответ, в котором указывается дошла ли команда изменения положения для робота, а также реальное расположение рабочего органа в виде координат.

Помимо этого, в верхней части экрана отображается панель, показанная на рисунке 25, которая указывает пользователю на успешность подключения к серверу.

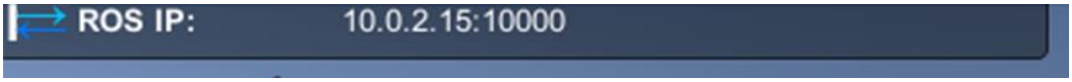

Рисунок 25 – Панель подключения к серверу

При успешном подключении, стрелки обозначаются синим цветом, в ином – красным.

4.2. РЕАЛИЗАЦИЯ АЛГОРИТМОВ

4.2.1. Управление в среде разработки Unity

Для написано программного кода в среде разработки Unity использовался язык программирования C#. На рисунке 26 показана структура сцены приложения в Unity.

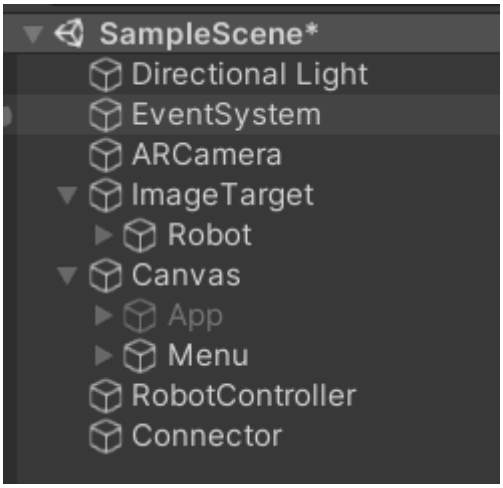

Рисунок 26 – Структура приложения в Unity

Данный список представляет объекты, находящиеся на рабочей сцене.

1. Directional Light – источник освещения, который обеспечивает отображение всех объектов, находящихся на сцене.

2. EventSystem – отвечает за обработку событий пользовательского интерфейса.

3. ARCamera – отображает на экран мобильного устройства (или иного устройства вывода) происходящий процесс работы приложения в дополненной реальности.

4. ImageTarget – метка, служащая для отображения 3D-моделей в дополненной реальности при сканировании камерой.

5. Robot – модель робота-манипулятора, которая отображается при инициализации метки камерой дополненной реальности. Модель робота должна быть привязанной к метке, поэтому она находится внутри.

6. Canvas – объект, отображающий положение элементов пользовательского интерфейса. Внутри объекта находятся два экрана. Menu – экран, появляющийся при запуске приложения. В нем находится поле ввода для подключения к серверу, и кнопка «Подключить». Объект App содержит элементы интерфейса основного экрана приложения (слайдеры, кнопки, ответы от сервера).

7. RobotController – объект, предназначенный для выполнения скриптов по управлению роботом-манипулятора. Данный объект содержит.

8. Connector – объект, выполняющий подключение к серверу ROS с помощью скрипта ROSConnection.cs.

Одной из основных задач является реализация управления каждого звена манипуляционного робота. Управление реализовывалось с помощью слайдеров, описанное в листинге 1.

Листинг 1 – Метод управления роботом-манипулятором в Unity

```
void Update()
    {
        for (int i = 0; i < links.Count; i++)
        {
            if((i ==0) || (i==3))
            {
                links[i].localRotation = Quaternion.Euler(0,
sliders[i].value, 0);
            }
```

```
            else
            {
                links[i].localRotation = Quaternion.Euler(0, 0,
sliders[i].value);
            }

        }
        grib7_1.transform.localPosition = new Vector3(0, gribSlider.value, -
0.655f);
        grib7_2.transform.localPosition = new Vector3(0, -gribSlider.value, -
0.655f);
      }
```

Метод Update – это специализированный метод, который поддерживается библиотекой UnityEngine, предназначенный для реализации динамичных действий, выполняющихся в течении запуска всего приложения. Данный метод является цикличным и выполняется с частотой в один кадр.

Код, описанный внутри метода позволяет каждому звену манипуляционного робота поворачиваться на соответствующее значение слайдера. Причем первое и третье звено имеет ось вращения по оси Z, а остальные по Y. Также в данном методе реализовано управление схвата в зависимости от значения его слайдера.

Одну из важных частей играет управление элементами графического интерфейса. В листинге 2 описан класс MenuController, который предназначен для выполнения данной задачи.

Листинг 2 – Фрагмент класса MenuController

```
public void LoadApp(GameObject canvas)
        {
            ROSConnection.instance.m_RosIPAddress = input.text;
            GameObject greating = canvas.transform.GetChild(1).gameObject;
            GameObject app = canvas.transform.GetChild(0).gameObject;
            greating.SetActive(!greating.activeSelf);
            app.SetActive(!greating.activeSelf);
        }

public void SendButton(GameObject panel)
        {
            ROS_Msg.instance.SendMsg();
            panel.SetActive(true);
           GetMsg();
        }

public void GetMsg()
        {
```

```
            answerText.text = "" + ROS_Msg.instance.isMessageRecived;
            xText.text = "X= " + ROS_Msg.instance.pos.x;
            xText.text = "Y= " + ROS_Msg.instance.pos.y;
            xText.text = "Z= " + ROS_Msg.instance.pos.z;
        }
```

В данном фрагменте представлено три метода: LoadApp, SendButton и GetMsg.

Метод LoadApp отвечает за управление первого окна (рисунок 22), который представлен пользователю при запуске приложения. По нажатию на кнопку «Подключиться» производится присоединение к серверу, который задал пользователь в окне ввода. После подключения окно закрывается и раскрывается основной экран приложения.

Метод SendButton вызывается в момент нажатия на кнопку «Отправить». После выполненного данного действия раскрывается панель с ответом на успешное изменение положения манипуляционного робота и с помощью метода GetMsg выводятся полученные координаты положения рабочего органа.

4.2.2. Обмен данными между Unity и ROS

Отправка параметров системе интеграции ROS, а также принятие от нее данных осуществляется с помощью скрипта ROS_Msg.cs (листинг 3).

Листинг 3 – Фрагмент класса ROS_Msg

```
void Start()
    {
        if(instance == null)
        {
            instance = this;
        }

        ros = ROSConnection.GetOrCreateInstance();
        ros.RegisterPublisher<PosRotMsg>("get_pos");
    }


public void SendMsg()
    {

        Inverse.instance.DirectProblem(q1, q2, q3, q4, q5);
```

```
        AnglesMsg msg = new AnglesMsg(
                q1,q2,q3,q4,q5
         );
         ros.Publish("get_pos", msg);
    }


public void ResieveMsg(RosMessageTypes.Geometry.PoseMsg poseMsg)
    {

        pos = GetPosition(poseMsg);
        isMessageRecived = true;
    }

public Vector3 GetPosition(RosMessageTypes.Geometry.PoseMsg msg)
    {
        pos.x = (float)msg.position.x;
        pos.y = (float)msg.position.y;
        pos.z = (float)msg.position.z;

        return pos;
    }
```

Метод Start выполняется один раз во время запуска приложения. Она необходима для того, чтобы инициализировать различные процессы. В данном случае создается экземпляр класса ROSConnection, с помощью которого реализуется обмен данными между Unity и ROS. Помимо этого, в данном методе создается топик для выполнения данного блока. В операционной системе ROS используется система топиков, которая осуществляет однонаправленную и непрерывную отправку или получение сообщений. Данный способ коммуникации подходит для датчиков, которым требуется периодическая передача данных.

Метод SendMsg отправляет углы поворота звеньев серверу ROS в соответствующем топике.

Метод ResieveMsg получает от сервера ROS координату положения рабочего органа реального робота-манипулятора, а также возвращает ответ в переменную isMessageRecived о том, было ли доставлено сообщение.

Метод GetPosition получает данные от сервера ROS и записывает их в локальный вектор, параметры которого выводятся текстом на панель приложения с результатом получения данных.

4.2.3. Реализация алгоритмов в ROS

В первую очередь сервер ROS должен получить команду из Unity и произвести ее выполнение. Реализация всех скриптов выполняется на языке программирования Python.

Для реализации получения данных от других источников и работы с системой ROS, подключаются библиотеки rospy, socket и re. Фрагмент кода Subscriber.py описан в листинге 4.

Листинг 4 – Фрагмент кода Subscriber.py

```
def __init__(self, topic, service_class, tcp_server, queue_size=10):
        strippedTopic = re.sub("[^A-Za-z0-9_]+", "", topic)
        self.node_name = "{}_service".format(strippedTopic)

        self.topic = topic
        self.service_class = service_class
        self.tcp_server = tcp_server
        self.queue_size = queue_size

        self.service = rospy.Service(self.topic, self.service_class,
self.send)

      self.sub = rospy.Subscriber(self.topic, self.msg, self.send)


def Send(self, request):
      return self.tcp_server.send_unity_service(self.topic,
self.service_class, request)
```

Метод __init__ принимает 3 параметра: название топика, класс сообщения, определенного в воркспейсе проекта ROS и максимальное количество хранимых записей равное 10.

В данном методе преобразуется название передаваемого топика в нужный формат производится чтение параметров, отправленных из среды разработки Unity.

Метод Send реализует подключение к конечной точки TCP и передачу данных. Данный метод принимает единственный параметр request – сообщение, которое должен получить Unity.

Для выполнения требуемый условий приложения, необходимо решить задачи прямой и обратной кинематики. В листинге 5 описан метод, вычисляющий прямую задачу кинематики.

Листинг 5 – Метод прямой задачи кинематики

```
def DirectProblem(self,q1,q2,q3,q4,q5):

        l1 = self.__l1
        l2 = self.__l2
        l3 = self.__l3
        l4 = self.__l4
        l5 = self.__l5
        q1 = q1 - self.__convertionAngleFor1
        q2 = q2 - self.__convertionAngleFor2
        q3 = q3 - self.__convertionAngleFor3
        q4 = -q4 - self.__convertionAngleFor4
        q5 = q5 - self.__convertionAngleFor5

        tT = np.array([[cos(q5)*(cos(q4)*(cos(q1)*cos(q2)*cos(q3) -
cos(q1)*sin(q2)*sin(q3)) + sin(q4)*(cos(q1)*cos(q2)*sin(q3) +
cos(q1)*cos(q3)*sin(q2))) - sin(q1)*sin(q5), -cos(q5)*sin(q1) -
sin(q5)*(cos(q4)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)) +
sin(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))), -
cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2)) +
sin(q4)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)), -
cos(q1)*cos(q2)*cos(q3)*l3 + cos(q1)*cos(q2)*l2 +
cos(q1)*l3*sin(q2)*sin(q3) + (l4 + l5)*(-
cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2)) +
sin(q4)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)))],
                        [cos(q1)*sin(q5) +
cos(q5)*(cos(q4)*(cos(q2)*cos(q3)*sin(q1) - sin(q1)*sin(q2)*sin(q3)) +
sin(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))),
cos(q1)*cos(q5) - sin(q5)*(cos(q4)*(cos(q2)*cos(q3)*sin(q1) -
sin(q1)*sin(q2)*sin(q3)) + sin(q4)*(cos(q2)*sin(q1)*sin(q3) +
cos(q3)*sin(q1)*sin(q2))), -cos(q4)*(cos(q2)*sin(q1)*sin(q3) +
cos(q3)*sin(q1)*sin(q2)) + sin(q4)*(cos(q2)*cos(q3)*sin(q1) -
sin(q1)*sin(q2)*sin(q3)), -cos(q2)*cos(q3)*l3*sin(q1) +
cos(q2)*l2*sin(q1) + l3*sin(q1)*sin(q2)*sin(q3) + (l4 + l5)*(-
cos(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2)) +
sin(q4)*(cos(q2)*cos(q3)*sin(q1) - sin(q1)*sin(q2)*sin(q3)))],
                        [cos(q5)*(cos(q4)*(cos(q2)*sin(q3) +
cos(q3)*sin(q2)) + sin(q4)*(-cos(q2)*cos(q3) + sin(q2)*sin(q3))), -
sin(q5)*(cos(q4)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)) + sin(q4)*(-
cos(q2)*cos(q3) + sin(q2)*sin(q3))), -cos(q4)*(-cos(q2)*cos(q3) +
sin(q2)*sin(q3)) + sin(q4)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)), -
cos(q2)*l3*sin(q3) - cos(q3)*l3*sin(q2) + l1 + l2*sin(q2) + (l4 +
l5)*(-cos(q4)*(-cos(q2)*cos(q3) + sin(q2)*sin(q3)) +
sin(q4)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))],
                        [0, 0, 0, 1]])

        return (np.dot(tT,np.array([[0],[0],[0],[1]])))
```

Метод DirectProblem принимает на вход 5 параметров: углы звеньев манипуляционного робота, отправленных из среды разработки Unity. В данном методе выполняется вычисление параметров Денавита-Хартенберга и находятся матрицы однородного преобразования, которые, путем перемножения, вычисляют итоговую матрицу, связывающую все системы координат и координаты положения схвата манипулятора для общего положения с учетом полученной системы координат.

Рассмотрим задачу обратной кинематики в системе ROS (листинг 6).

```
def InversProblem(self,x,y,z,pitch = 0,roll = 0):
        q5 = roll

        q1 = atan2(y,x)

        w = sqrt(x**2+y**2)

        z1 = (z - self.__l1) - (self.__l4 + self.__l5)*sin(pitch)
        w1 = w - (self.__l4 + self.__l5)*cos(pitch)

        c = (self.__l3**2-w1**2-z1**2-self.__l2**2)/(-2)
        a = z1**2 + w1**2
        b = -2*z1*c
        e = c**2 - self.__l2**2*w1**2

        D = b**2 - 4*a*e
        if (D < 0 ):
            return False,[0,0,0,0,0]
        z2_1 = (-b + sqrt(D))/(2*a)
        z2_2 = (-b - sqrt(D))/(2*a)

        w2_1 = (c-z2_1*z1)/w1
        w2_2 = (c-z2_2*z1)/w1

        q2_1 = atan2(z2_1,w2_1)
        q2_2 = atan2(z2_2,w2_2)

        joint3Angle_1 = atan2(z1 - z2_1,w1-w2_1)
        joint3Angle_2 = atan2(z1 - z2_2,w1-w2_2)

        q3_1 = pi - q2_1 + joint3Angle_1
        q3_2 = pi - q2_2 + joint3Angle_2

        q4_1 = -(joint3Angle_1 - pitch)-pi/2
        q4_2 = -(joint3Angle_2 - pitch)-pi/2

        return [q1,q2,q3,q4,q5]
```

Данный метод решает обратную задачу кинематики. На вход метода InversProblem подается 5 параметров: координаты положения схвата, pitch и roll, которые по умолчанию равны 0.

Переменная pitch определяет угол наклона рабочего органа манипуляционного робота к горизонтали, где значение -1,57 означает наклон вниз, а 0,7 – вверх. Переменная roll содержит значение поворота схвата вокруг своей оси.

Используя теорему косинусов и теорему Пифагора, описывающиеся в данном методе, в результате возвращаются углы каждого звена робота-манипулятора, которые, в дальнейшем, отправляются в контроллер реального манипуляционного робота.

Далее необходимо передать топик с нужными параметрами в контролер через скрипт arm_controll.cpp, представленный готовой библиотекой Dynamixel SDK, который выполняет изменения положения робота-манипулятора. После этого, сервер ROS отправляет данные в среду разработки Unity с текущими координатам положения рабочего органа. Методы, реализующие отправку данных обратно в Unity представлен в листинге 7.

Листинг 7 – Фрагмент класса Publisher.py

```
def __init__(self, topic, message_class, queue_size=10, latch=False):
      strippedTopic = re.sub("[^A-Za-z0-9_]+", "", topic)
      node_name = "{}_RosPublisher".format(strippedTopic)
      RosSender.__init__(self, node_name)
      self.msg = message_class()
      self.pub = rospy.Publisher(topic, message_class, queue_size=queue_size,
latch=latch)

def Send(self, data):
      self.msg.deserialize(data)
      self.pub.publish(self.msg)
      return None
```

В методе __init__ осуществляется реализация отправки сообщения, содержащего в топике, указанного в качестве параметра переменной topic. Помимо данного параметра, в метод передаются класс сообщения из воркспейса проекта в ROS, максимальное количество хранения записей. Параметр latch отвечает за возможность задержки передачи данных.

Метод Send выполняет отправку данных, заданных в методе __init__.

4.3. ВЫВОД

В разделе реализации было рассмотрено три основных блока: реализация пользовательского интерфейса, реализация алгоритмов в среде разработки Unity и алгоритмов в среде интеграции ROS.

Приложение состоит из двух экранов. Первый экран появляется при запуске приложения и дает возможность пользователю подключиться к определенному серверу ROS. В основном экране расположены элементы графического интерфейса, позволяющие реализовать управление цифровым двойником робота-манипулятора.

Также были рассмотрены основные алгоритмы приложения. В среде разработки Unity были реализованы скрипты, позволяющие взаимодействовать пользователю с приложением и производить обмен данными между Unity и ROS.

Получив данные в виде углов вращения каждого звена манипуляционного робота, алгоритмы, написанные в системе ROS, вычисляли обратную и прямую задачу кинематики и отправляли данные в контроллер реального робота-манипулятора.

5. ТЕСТИРОВАНИЕ

Для тестирования разрабатываемого приложения использовалось функциональное тестирование и альфа-тестирование.

5.1. ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения решать нужные задачи для пользователей в определенных условиях. Результаты функционального тестирования представлены в таблицу 1.

Таблица 1 – Результаты функционального тестирования

№	Название теста	Шаги	Ожидаемый результат	Результат
1	Работа кнопки «Подключитьс я»	1. Запустить приложение 2. Ввести IP-адрес в окно ввода 3. Нажать на кнопку «Подключиться» в первом экране	После нажатия на кнопку открывается основная сцена с включением камеры дополненной реальности. На панельке в верхней части экрана отображается IP-адрес, введенный пользователем	Тест пройден
2	Работа камеры дополненной реальности	1. Запустить приложение 2. Ввести IP-адрес в окно ввода 3. Нажать на кнопку «Подключиться» в первом экране.	Нажав на кнопку «Подключиться», экран подключения исчезает и включается камера смартфона с соответствующими элементами графического интерфейса.	Тест пройден
3	Отображение объектов на метке	1. Войти в основной экран приложения. 2. Навести камеру на метку	На метке должна появиться модель робота-манипулятора	Тест пройден

Продолжение таблицы 1

4	Управление слайдерами	1. Отобразить 3D-модель манипуляционного робота 2. Схватить ползунок и переносить в разные стороны	Перетаскивая ползунок слайдера соответствующего звена, производятся его вращение	Тест пройден
5	Работа кнопки «Отправить»	1. Войти в основной экран приложения. 2. Изменить положение манипуляционного-робота с помощью слайдеров 3. Нажать на кнопку «Отправить»	После нажатия на кнопку «Отправить», появляется панель в правой части экрана с отображением результата передачи параметров и координат положения рабочего органа манипуляционного робота	Тест пройден
6	Изменение положения реального робота манипулятора	1. Войти в основной экран приложения. 2. Изменить положение манипуляционного-робота с помощью слайдеров 3. Нажать на кнопку «Отправить»	Реальный робот-манипулятор должен изменить свое положение в соответствии с положением цифрового двойника в среде разработки Unity	Тест пройден
7	Отображение результата изменения положения реального робота	1. Войти в основной экран приложения. 2. Изменить положение манипуляционного-робота с помощью слайдеров 3. Нажать на кнопку «Отправить»	После успешной передачи данных и изменения положения реального робота-манипулятора в панели справа отображается ответ «True» и координаты x,y,z с значениями положения рабочего органа манипуляционного робота	Тест пройден

5.2. АЛЬФА-ТЕСТИРОНИЕ

Альфа-тестирование – это вид приемочного тестирования, которое включает имитацию реального использования продукта штатными разработчиками либо командой тестировщиков.

Первое, на что стоит обратить внимание – это возможность подключения к серверу ROS. Для проверки необходимо обратить внимание на знак в верхней части экрана (рисунок 27).

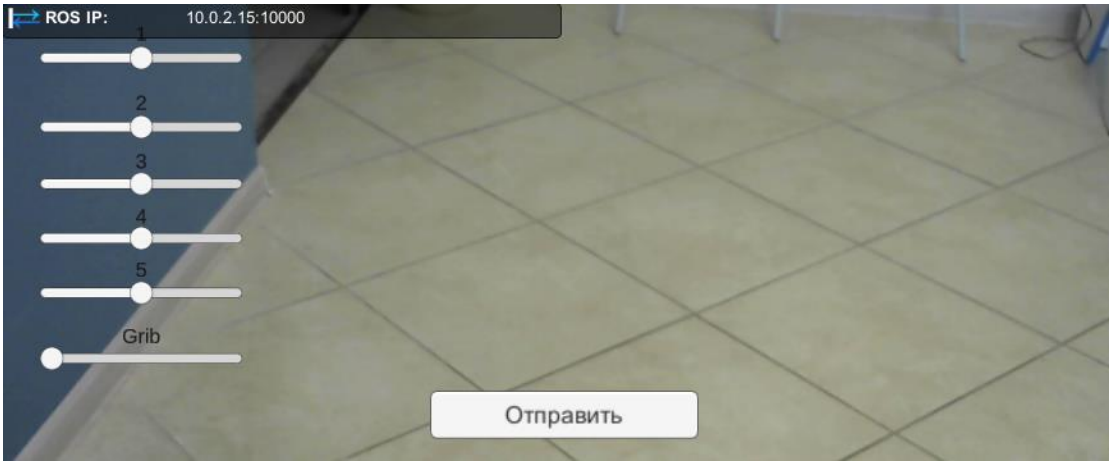

Рисунок 27 – Подключение к серверу ROS

Также стоит проверить может ли цифровая модель робота отображаться на заданной метке в виде обложки документации к данному роботу-манипулятору. Для этого достаточно навести камеру телефона на метку. Результат тестирования представлен на рисунке 28.

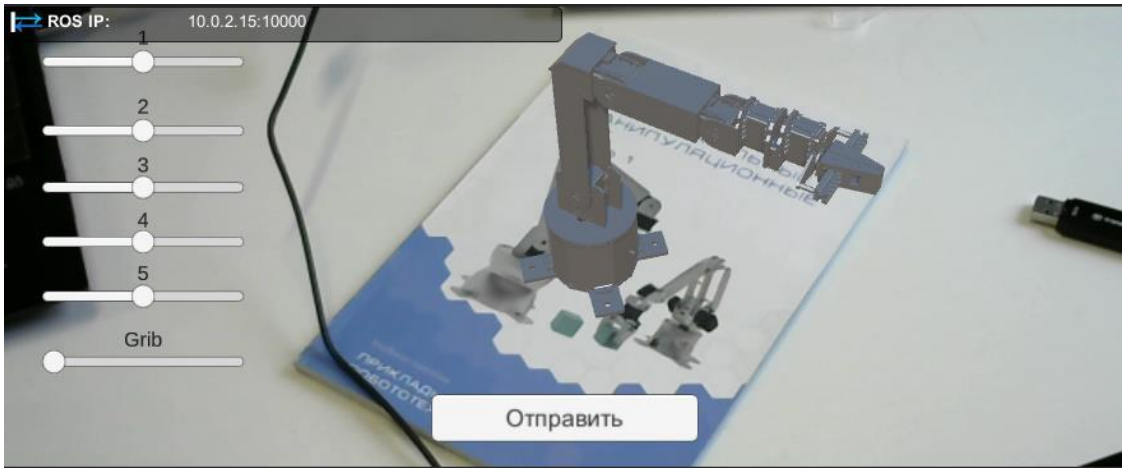

Рисунок 28 – Отображение модели манипуляционного робота

Изначально цифровая модель манипуляционного робота расположена также, как модель, представленная на рисунке 28. Чтобы изменить положение звеньев робота-манипулятора необходимо перетаскивать соответствующие ползунки на слайдерах (рисунок 29).

Рисунок 29 – Изменение положения манипуляционного робота

После того, как положение робота-манипулятора изменилось, можно отправить данную позицию реальному роботу, нажав на кнопку «Отправить». Полученный результат представлен на рисунке 30.

Рисунок 30 – Отправка положения манипуляционного робота

После того, как реальный робот-манипулятор получил необходимую позицию, его положение изменилось как показано на рисунке 31.

Рисунок 31 – Изменение положения реального манипуляционного робота

5.3. ВЫВОД

В данном разделе было проведено тестирование приложения. Проводилось два вида тестирования: функциональное тестирование и альфа-тестирование.

Функциональное тестирование позволяет провести проверку реализуемости функциональных требований. Альфа-тестирование имитирует работу с системой разработчиками приложения или потенциальными пользователями.

Все тесты были выполнены успешно.

6. ЗАКЛЮЧЕНЕ

В ходе выполнения магистерской диссертации было разработано приложение дополненной реальности для дистанционного управления робота-манипулятора. Для достижения поставленных задач следовало провести анализ аналогов и обзор литературы, чтобы выявить актуальность разработки данного приложения и определить ее концепцию. Для определения функциональной части были описаны соответствующие требования для системы. В результате работы было разработано приложение, удовлетворяющее всем функциональным и нефункциональным требованиям, а также было проведено тестирование во избежание ошибок и проверки корректности функциональной части приложения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алферов, Г.В. Информационные системы виртуальной реальности в мехатронике и робототехнике: учеб. пособие / Г.В. Алферов [и др.]. – СПб.: «СОЛО», 2006. – 146 с.
2. Михайлюк, М. В. Видеотренажерный комплекс управления роботами и манипуляторами / М.В. Михайлюк // Сб. трудов международного симпозиума «Инновационные технологии в исследовании окружающей среды». – Ларнака–М.: 2013. – С. 84–85.
3. Шаветов, С.В. Архитектура системы удаленного управления робототехническими объектами / А.А. Ведяков, А.А. Пыркин // Научно-технический вестник информационных технологий, механики и оптики. 2014. – №2(90). – С. 161–163.
4. Azuma, R. A Survey of Augmented Reality / R. Azuma // Teleoperators and Virtual Environments 6. – 1997. – No. 4. – P. 355–385.
5. Angeles J. Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms. – New York.: 2007. – 3rd ed. – 114 p.
6. Siciliano, B. Modelling, Planning and Control – Springer / Sciavicco, L., Villani, L., Oriolo G. Robotics. – London, 2009 – 101 p.
7. Заборовский, В. С. Применение киберфизического подхода в задачах сетецентрического управления роботами / В.С. Заборовский, М.Ю. Гук, В.А. Мулюха // Робототехника и техническая кибернетика. – 2014. – № 2 (3). – С. 12–18.
8. Milgram, P. Applications of Augmented Reality for HumanRobot Communication / P. Milgram // Proc. International Conference on Intelligent Robotics and Systems IROS’93. – Japan.: Yokohama, 1993. – P. 1467–1472.
9. Drascic, D. Stereoscopic Vision and Augmented Reality / D. Drascic // Scientific Computing & Automation 9. – 1993. – No. 7. – P. 31–34.
10. Кулаков, Ф. М. Информационная технология обучения роботов показом движений. Часть I. Концепция и принципы моделирования

движений / Ф.М. Кулаков, С.Э. Чернакова // Мехатроника, автоматизация, управление. – 2008. – № 7. – С. 23–28.

11. Официальный сайт компании ABB RobotStudio. – https://robotum.cz/wp-content/uploads/2019/12/3HAC032104-OM-RobotStudio-en.pdf. Дата обращения: 25.03.2022.
12. Официальный сайт компании Yaskava Motoman. – https://www.motoman.com/en-us. Дата обращения: 25.03.2022.
13. Официальный сайт компании KUKA. – https://www.kuka.com. Дата обращения: 25.03.2022.
14. Официальный сайт компании Comau. – https://robotics.ee.uwa.edu.au/robosim/robosim.pdf. Дата обращения: 03.03.2022.
15. Букреев, В.Г. Математическое обеспечение адаптивных систем управления электромеханическими объектами. Учебное пособие. – Томск: Изд-во ТПУ, 2002 – 132 с.
16. Колтович, В. Е. Программные платформы для управления роботами / Колтович, В. Е. // Информационные технологии в образовании, науке и производстве: III Международная научно-техническая интернет-конференция. – Минск: Беларусь, 2015. – 56 с.
17. Система управления ROS на примере робота Turtlebot / А. А. Белоусов, И. Н. Тихонов, А. В. Лемех, В. А. Третьяков // Перспективы развития информационных технологий. – 2014. – № 22. – С. 6–15.
18. Slabaugh G. Computing Euler angles from a rotation matrix / G. Slabaugh. // Technical Report. –1999. – P. 1-7.
19. Denavit J. A kinematic notation for lower-pair mechanisms based on matrices / J. Denavit, R. Hartenberg // ASME J. Applied Mechanics. –1955. – № 23. — P. 215–222.

20. Юревич, Е. И. Основы робототехники: учебное пособие для студентов высших учебных заведений / Е. И. Юревич. – 3-е изд. – СПб.: БХВ-Петербург, 2009. – 359 с.