МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки / специальности)

Электроника и робототехника
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Система промышленного управления промышленным роботом-манипулятором Scara

Обучающийся А.Д. Артемьев
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, А.В. Прядилов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант И.Ю. Усатова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

## Аннотация

Работа может представлять интерес для разработчиков робототехнических систем, специалистов в области промышленной автоматизации, а также преподавателей и студентов, занимающихся проектированием систем управления манипуляторами.

Название выпускной квалификационной работы: "Разработка системы управления промышленным манипулятором SCARA".

Выпускная работа состоит из введения, трёх глав, 24 рисунков, 2 таблиц, заключения и списка литературы из 22 источников, включая зарубежные издания.

Целью данной работы является разработка системы управления промышленным манипулятором SCARA, ориентированной на применение в автоматизированных производственных процессах.

Работа посвящена анализу современных тенденций в робототехнике, исследованию конструктивных особенностей и преимуществ манипуляторов типа SCARA, а также обзору существующих аппаратных и программных решений.

Объектом исследования является электропривод, управляемый с помощью электрических сигналов и физических усилий.

Предметом работы выступает разработка системы управления промышленным манипулятором SCARA, способной имитировать реальное поведение системы и позволяющей проводить динамический и статический анализ характеристик привода.

Основные задачи выпускной работы включают:

– выбор компонентов для системы управления;
– разработку собственной платы экранирования для Arduino Mega;
– создание электрической схемы системы управления;

– разработку программного обеспечения для микроконтроллера Arduino Mega.

Первая часть описывает современное состояние промышленной робототехники, уделяя особое внимание модульным решениям и потребности в гибких системах автоматизации.

Вторая часть представляет этапы разработки системы управления манипулятором SCARA, включая выбор компонентов, проектирование механической части, разработку электроники и реализацию программного обеспечения.

Третья часть посвящена интеграции и тестированию системы, демонстрации её работы в реальных условиях и подтверждению высокой степени модульности, точности и экономической эффективности.

Анализ реализации цифрового двойника манипулятора SCARA и высокоуровневого интерфейса в среде RoboDK показал успешное моделирование поведения манипулятора в различных условиях эксплуатации.

В заключении, разработанная система управления успешно расширяет функциональные возможности промышленных манипуляторов SCARA при сохранении совместимости с существующими системами. Использование недорогих компонентов и открытых протоколов связи делает её подходящей как для образовательных целей, так и для внедрения в небольшие автоматизированные производства. Дальнейшие улучшения могут включать реализацию обратной связи по положению и скорости, а также внедрение алгоритмов планирования траекторий.

## Abstract

The work may be of interest to developers of robotic systems, specialists in the field of industrial automation, as well as teachers and students involved in designing manipulator control systems.

The title of the graduation thesis is "Development of a control system for an industrial SCARA manipulator".

The final work consists of an introduction, three chapters, 24 figures, 2 tables, a conclusion and a list of references from 22 sources, including foreign publications.

The purpose of this work is to develop a control system for the SCARA industrial manipulator, aimed at application in automated production processes.

The work is devoted to the analysis of current trends in robotics, the study of the design features and advantages of manipulators such as SCARA, as well as an overview of existing hardware and software solutions. The object of the study is an electric drive controlled by electrical signals and physical effort. The subject of the work is the development of a SCARA industrial manipulator control system capable of simulating the real behavior of the system and allowing dynamic and static analysis of drive characteristics.

The main objectives of the graduation thesis include:

- selection of components for the control system;
- development of our own shielding board for Arduino Mega;
- creation of an electrical circuit of the control system;
- software development for the Arduino Mega microcontroller.

The first part describes the current state of industrial robotics, focusing on modular solutions and the need for flexible automation systems.

The second part presents the development stages of the SCARA manipulator control system, including component selection, mechanical design, electronics development, and software implementation.

The third part is devoted to the integration and testing of the system, demonstrating its operation in real conditions and confirming a high degree of modularity, accuracy and cost-effectiveness. An analysis of the implementation of the digital twin of the SCARA manipulator and the high-level interface in the RoboDK environment showed successful modeling of the manipulator's behavior in various operating conditions.

In conclusion, the developed control system successfully expands the functionality of SCARA industrial manipulators while maintaining compatibility with existing systems. The use of inexpensive components and open communication protocols makes it suitable both for educational purposes and for implementation in small automated production facilities. Further improvements may include the implementation of position and velocity feedback, as well as the introduction of trajectory planning algorithms.

## Содержание

Введение ..... 7
1. Состояние вопроса ..... 8
1.1. Формулирование цели и задач проекта ..... 8
1.2. Анализ исходных данных и существующих решений ..... 9
1.3. Определение концепции ..... 10
2. Подбор комплектующих ..... 15
2.1. Проектирование печатной платы ..... 24
2.2. Сборка системы управления ..... 29
2.3. Разработка низкоуровневого программного обеспечения ..... 32
3. Разработка высокоуровневого программного обеспечения ..... 42
3.1. Интеграция Scara-робота с искусственным интеллектом ..... 47
3.2. Анализ программных платформ RoboDok и ROS ..... 49
3.3. SCARA-манипуляторы: промышленные и DIY-сегмент ..... 55
Заключение ..... 60
Список используемой литературы ..... 62
Приложение А. Структурная схема ..... 64
Приложение Б. Принципиальная схема ..... 65
Приложение В. Перечень элементов ..... 66

# Введение

Современные тенденции развития промышленного производства требуют внедрения высокотехнологичных решений, обеспечивающих гибкость, точность и надежность автоматизированных процессов. Одним из ключевых элементов современных автоматизированных систем являются промышленные роботы-манипуляторы, которые позволяют выполнять широкий спектр задач — от простого перемещения объектов до сложных сборочных операций.

Особое место среди разнообразия конструкций манипуляторов занимают роботы типа SCARA (Selective Compliance Assembly Robot Arm), предназначенные для высокоскоростного горизонтального перемещения и широко применяемые в электронной, машиностроительной, пищевой и других отраслях промышленности [7]. Роботы SCARA отличаются компактностью, высокой скоростью работы и точностью позиционирования, что делает их идеальным выбором для задач автоматизации производственных процессов.

В рамках данной работы была разработана система управления промышленным роботом-манипулятором SCARA, ориентированная на использование доступных и недорогих компонентов, таких как микроконтроллер Arduino Mega 2560 и программная среда RoboDK. Разработка включает аппаратную часть, реализующую управление сервоприводами, цифровыми входами/выходами и концевыми датчиками, а также программное обеспечение, организованное по модульному принципу и поддерживающее обмен командами по последовательному интерфейсу.

# 1 Состояние вопроса

## 1.1 Формулирование цели и задач проекта

Актуальность разработки системы управления промышленным роботом-манипулятором SCARA обусловлена необходимостью внедрения современных решений в области автоматизации производственных процессов. В условиях роста требований к точности и скорости выполнения операций, а также дефицита квалифицированного ручного труда, промышленные роботы становятся незаменимым элементом современного производства.

Манипуляторы типа SCARA обладают высокой точностью, повторяемостью и компактной конструкцией, что делает их идеальными для выполнения сборочных, упаковочных и сортировочных операций. Однако основным фактором, определяющим эффективность работы такого манипулятора, является система управления, которая должна обеспечивать точное позиционирование, плавность траектории и безопасность работы.

Цель работы — разработка системы управления для промышленного манипулятора типа SCARA, включающей аппаратную и программную часть, обеспечивающую реализацию алгоритмов движения и взаимодействия с оператором.

Для достижения цели были поставлены следующие задачи:

– анализ существующих решений в области управления SCARA-манипуляторами;

– подбор комплектующих для системы управления манипулятором

– разработка и проектирование собственной Shield-платы расширения для Arduino Mega

– проектирование электрической принципиальной схемы системы управления;

– разработка программного обеспечения для микроконтроллера Arduino Mega;

– анализ реализации цифрового двойника SCARA-манипулятора и верхнего уровня в среде RoboDK.

## 1.2 Анализ исходных данных и существующих решений

На этапе проектирования системы управления SCARA-манипулятором необходимо было учесть как технические требования, так и доступность элементной базы, программных решений и типовых архитектур. SCARA-манипуляторы широко применяются в производстве благодаря высокой точности и скорости, но промышленные образцы, например, Omron Adept, Yamaha YK, Epson SCARA ориентированы на использование в закрытых экосистемах с собственными контроллерами, что затрудняет адаптацию и существенно увеличивает стоимость [14].

В рамках открытого и академического сообщества сформировалось несколько знаковых проектов, на основе которых возможна разработка лабораторных и исследовательских установок:

MiniSCARA jjshortcut — компактный SCARA-манипулятор, построенный на 3D-печатных компонентах и управляемый с помощью Arduino + GRBL. Отличается простотой конструкции, но требует доработки по механике и алгоритмам калибровки.

MechArm Pi — образовательный SCARA-комплект от Elephant Robotics, совместим с Raspberry Pi. Имеет интерфейс Python и базовую интеграцию с ROS, однако относится к классу готовых решений с ограниченной настраиваемостью и невозможностью модификации низкого уровня.

Ulx3S-SCARA — SCARA-механизм, управляемый через FPGA и построенный с опорой на FreeRTOS. Имеет высокую точность и гибкость, но требует опыта работы с HDL и аппаратным уровнем.

Кроме того, в образовательной практике широко используется платформа Dobot Magician, представляющая собой коммерческий манипулятор с SDK на C++ и Python. Несмотря на высокую точность, он ограничен в интерфейсах и не допускает глубокую аппаратную модификацию.

Анализ показал, что существующие проекты либо ориентированы на конечных пользователей (Dobot, MechArm), либо требуют существенных доработок на аппаратном уровне, либо не предусматривают интеграции с промышленными элементами (Ethernet, щит IP65, маркеры, нормализованные драйверы). Это создаёт предпосылки для проектирования собственной системы, сочетающей промышленную надёжность, открытую архитектуру и адаптивность программного уровня.

### 1.3 Определение концепции

На основе анализа современных требований к системам управления промышленными SCARA-манипуляторами, а также обзора существующих решений, сформулирована концепция, направленная на разработку универсальной, модульной и масштабируемой системы управления с применением доступной аппаратной базы и интеграцией с высокоуровневыми программными средствами.

Ключевым элементом выбранной архитектуры является использование бесколлекторных двигателей постоянного тока (BLDC) в качестве исполнительных приводов на всех осях манипулятора. Это решение обосновано необходимостью обеспечения высокой надёжности, динамики и ресурса работы, а также наличием сигнальных линий для управления

скоростью, направлением вращения и считывания положения ротора с помощью датчиков Холла. Применение BLDC-приводов в сочетании с внешними драйверами управления (например, серии BLDC-4008) позволяет реализовать точное и плавное движение без характерных для шаговых двигателей потерь шага и перегрева.

Система управления построена на трёхуровневой архитектуре:

Аппаратный уровень включает микроконтроллер Arduino Mega 2560, специализированный шилд-плату с возможностью управления как BLDC-, так и шаговыми двигателями, драйверы двигателей, индуктивные датчики LJ12A3-4-Z/BX, блок питания 24 В, а также коммуникационные модули (например, USR-TCP232-T2).

Низкоуровневое программное обеспечение, реализованное на языке C++ в среде Arduino IDE, отвечает за обработку входных команд, управление силовыми выходами, реализацию интерфейса управления сервоприводами и опрос датчиков.

Высокоуровневое ПО, построенное на базе среды RoboDK, обеспечивает офлайн-моделирование движения, генерацию управляющих команд, визуализацию процессов и передачу координат на контроллер в формате текстовых команд.

Концепция предполагает:

– открытую модульную архитектуру, допускающую замену или расширение функциональных блоков (например, добавление новых осей, смена привода или интерфейса);

– аппаратную унификацию, позволяющую работать как с BLDC, так и с шаговыми двигателями без изменения базовой платформы;

– совместимость с промышленными интерфейсами: Ethernet, UART, с перспективой расширения до Modbus RTU или CANopen;

– гибкость настройки — через текстовый протокол команд и конфигурации, а также возможностью интеграции с Python-скриптами на стороне RoboDK;

– Масштабируемость — благодаря избытку портов ввода-вывода Arduino Mega и возможности расширения аппаратной логики.

Применение BLDC-приводов в рамках данной архитектуры обеспечивает оптимальный баланс между точностью позиционирования и эксплуатационной надёжностью, что особенно важно при реализации задач сборки, сортировки и манипулирования в производственной среде. В случае необходимости реализации позиционного сервоконтроля концепция допускает установку дополнительных энкодеров на приводах и построение замкнутого контура управления.

Таким образом, сформированная концепция представляет собой гибкую платформу для построения систем управления SCARA-манипуляторами, ориентированную на интеграцию в автоматизированные комплексы, а также применение в образовательной и исследовательской среде.

Выводы по разделу 1

В данном разделе была подробно рассмотрена и обоснована актуальность разработки системы управления SCARA-манипулятором, обусловленная стремительным развитием технологий автоматизации в промышленности. Манипуляторы типа SCARA благодаря своей конструктивной простоте, высокой скорости перемещения и точности позиционирования находят широкое применение в таких отраслях, как электроника, машиностроение, пищевая и фармацевтическая промышленность, где требуется выполнение операций по сборке, упаковке, нанесению клея, сверлению и другим задачам на плоскости XY. Однако большинство серийных решений отличаются высокой стоимостью,

закрытыми архитектурами и сложностью модернизации, что делает их недоступными для малых предприятий и образовательных учреждений.

С целью выявления возможностей для проектирования доступной и гибкой системы управления был проведён анализ существующих решений в области управления SCARA-роботами. Изучены промышленные контроллеры ведущих производителей, такие как Mitsubishi, Omron, Fanuc, а также открытые платформы — Arduino, Raspberry Pi, STM32, используемые в DIY-проектах и научных разработках. Также были рассмотрены программные среды, такие как MATLAB/Simulink, ROS (Robot Operating System), RoboDK, позволяющие моделировать поведение манипуляторов и создавать цифровые двойники.

На основании анализа сформулированы ключевые требования к системе управления, включая:

– модульность конструкции и возможность масштабирования;

– поддержку стандартных интерфейсов связи (Ethernet, Modbus, CAN);

– совместимость с популярными средами программирования и симуляции;

– использование недорогих и доступных компонентов;

– возможность интеграции в промышленные сети и системы верхнего уровня;

– обеспечение точного позиционирования и стабильного управления движением.

Определена концепция проекта, предполагающая разработку системы управления на основе микроконтроллера Arduino Mega 2560, дополненного самодельной экранирующей платой, обеспечивающей защиту и удобство подключения периферийных устройств. В качестве драйверов двигателей выбраны проверенные и доступные решения, такие как A4988 и TB6600, которые позволяют эффективно управлять шаговыми двигателями. Для

обеспечения высокого уровня функциональности и взаимодействия с внешними системами реализованы последовательный интерфейс и Ethernet-соединение через модуль USR-TCP232-T2.

Концепция системы управления разработана с учетом современных требований к модульности и интеграции с промышленными сетями, изложенными в работах по автоматизации и робототехнике. Предусмотрено использование открытых программных решений, таких как среда Arduino IDE и программное обеспечение RoboDK, что позволяет не только снизить затраты на разработку, но и обеспечивает гибкость при настройке и адаптации системы под конкретные задачи.

Данный подход позволяет создать универсальное решение, применимое как в образовательных целях, так и в условиях мелкосерийного производства, обеспечивая при этом высокую степень воспроизводимости и адаптируемости. Концепция системы управления разработана с учетом требований к модульности и интеграции с промышленными сетями, описанными в работах по автоматизации [15].

## 2 Подбор комплектующих

На этапе разработки системы управления манипулятором SCARA ключевым аспектом стал выбор электронной компонентной базы, обеспечивающей требуемый уровень функциональности, точности и надежности. При выборе комплектующих учитывались следующие факторы: совместимость с используемыми сервоприводами и шаговыми двигателями, наличие достаточного количества входов/выходов, поддержка протоколов связи, а также промышленная устойчивость комплектующих к условиям эксплуатации. При выборе комплектующих для системы управления учитывались параметры приводов, аналогичные описанным в литературе по электромеханическим системам [16].

В качестве центрального устройства управления был выбран микроконтроллер Arduino Mega 2560 на базе микросхемы ATmega2560. Данный микроконтроллер показан на рисунке 1. Данный контроллер обеспечивает 54 цифровых входа/выхода и 16 аналоговых входов, что позволяет подключать большое количество периферийных устройств (двигатели, датчики, реле, исполнительные механизмы) одновременно. Кроме того, наличие четырех независимых портов UART позволяет осуществлять параллельную связь с системами высокого уровня и модулями расширения.

Рисунок 1 – Arduino Mega

Терминальный адаптер для контроллера Arduino MEGA-2560, значительно упрощает подключение к контроллеру внешних устройств, который показан на рисунке 2. Шилд обеспечивает безопасный и надежный вывод всех портов Arduino Mega на клеммные колодки и разъемы с винтовым зажимом, что значительно упрощает сборку и обслуживание системы. Также на шилде предусмотрены фильтрующие элементы и защитные цепи для предотвращения выхода контроллера из строя при кратковременных импульсных помехах.

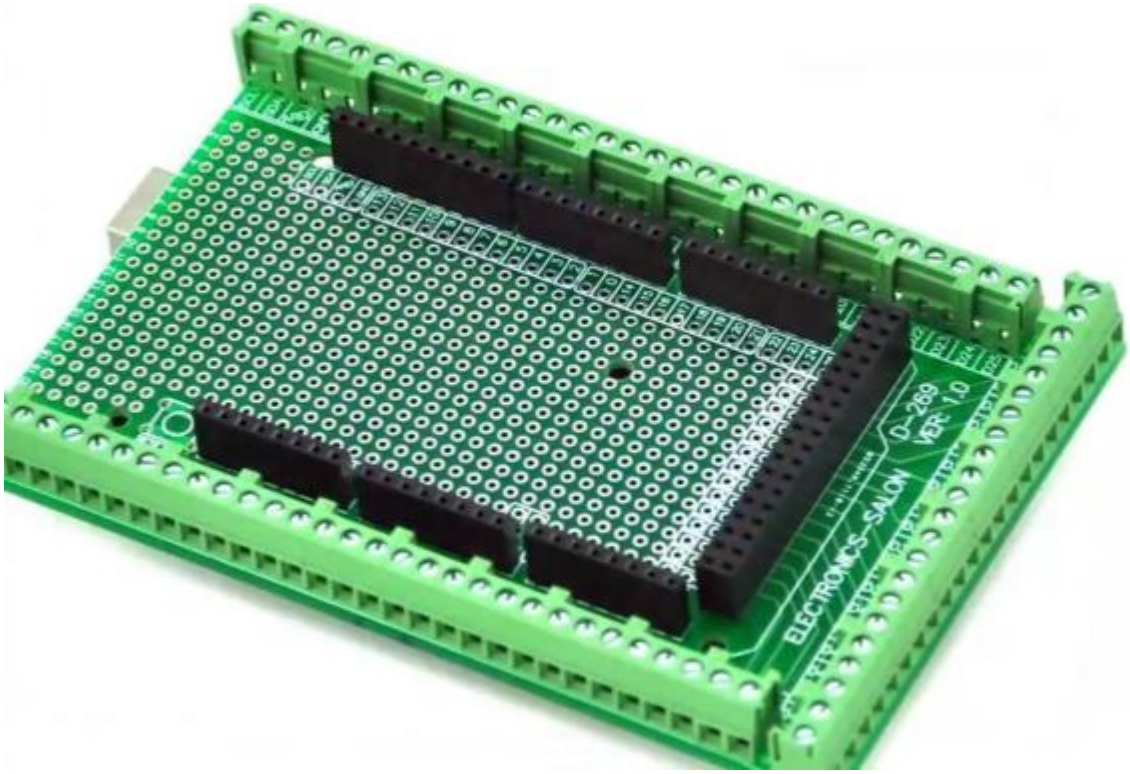



Рисунок 2 – Терминальный адаптер

Бесколлекторные электродвигатели конструктивно состоят из ротора с постоянными магнитами и статора с обмотками. Обычно имеют 3 фазы, для контроля скорости используются датчики Холла, напряжение питания 20...100 В. Управляется бесколлекторный электродвигатель специальным 3х фазным драйвером инвертором, в качестве сигнала обратной связи для поддержания постоянной скорости/момента служат сигналы от датчиков Холла. Область применения бесколлекторных электродвигателей — обеспечение вращения с заданной скоростью или заданным моментом на валу. Бесколлекторные электродвигатели не предназначены для точного позиционирования. При установке энкодера на двигатель можно получить BLDC сервопривод.

В качестве исполнительных механизмов используются бесщеточные электродвигатели постоянного тока серии BLF BLDC, показанные на рисунке 3. Данная серия двигателей отличается высокой надежностью,

большим крутящим моментом на низких оборотах, а также компактными размерами, что делает их особенно подходящими для использования в компактных манипуляторных установках. Для управления этими двигателями используются специализированные драйверы, обеспечивающие формирование ШИМ-сигналов и управление фазовым возбуждением.

Рисунок 3 – Бесколлекторные двигатели серии BLF

На рисунке 4 представлен чертеж привода форм-фактора PL57BLF. За счет стандартизированного формата корпуса с шаговыми двигатели, рекомендуется использовать готовые крепления приводов, редукторы, муфты и шкивы, что положительным образом скажется на снижении стоимости итогового продукта.

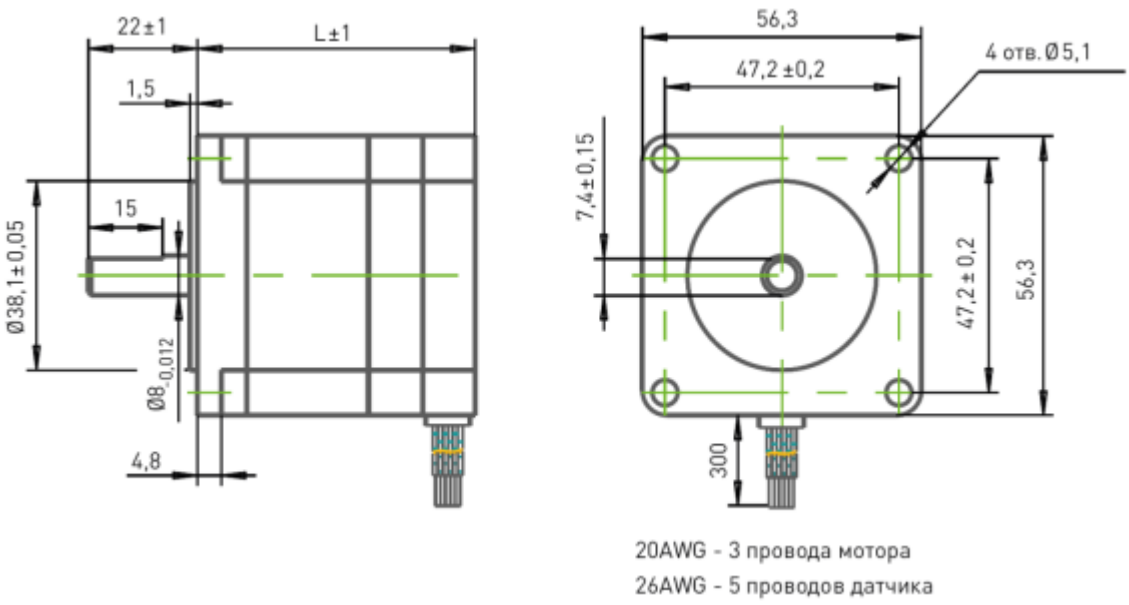



Рисунок 4 – Чертеж привода форм-фактора PL57BLF

Промышленные Scara манипуляторы показаны на рисунке 5 имеют следующее строение:

– привод плеча А расположен в неподвижном основании, используются цилиндрические, конические, планетарные зубчатые передачи;

– привод плеча В расположен непосредственно на движущимся плече, используются цилиндрические, конические, планетарные зубчатые передачи;

– привод оси Z приводится в движение за счет быстроходных шарико-винтовых передач, установленных на плече В. Вращение ШВП осуществляется при помощи ременных, цилиндрических, конических, планетарных зубчатых передач;

– привод поворота захвата расположен на плече В. Приводится в движение при помощи ременных зубчатых передач.

Для реализации управления приводами манипулятора использовались принципы автоматизированного электропривода, изложенные в учебнике Белова [3].

В таблице 1 приведены выбранные приводы и их характеристики.

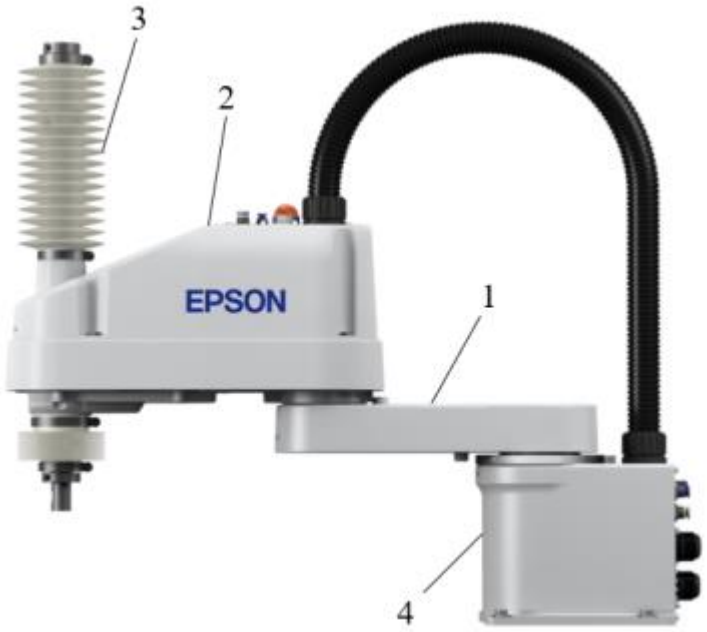



Рисунок 5 – Кинематика Scara-робота:
1 – плечо A; 2 – плечо C; 3 – ось Z; 4 – основание робота

Таблица 1 – Выбранные приводы робота Scara

| Ось робота | Модель привода | Напряжение, В | Рабочий ток, А | Мощность, Вт | Скорость вращения, об/мин |
|---|---|---|---|---|---|
| 1 | PL57BLF01 | 24 | 7 | 63 | 3000 |
| 2 | PL57BLF01 | 24 | 7 | 63 | 3000 |
| 3 | PL57BLF01 | 24 | 7 | 63 | 3000 |
| 4 | PL57BLF01 | 24 | 7 | 63 | 3000 |
| Доп. ось | PL42BLF | 24 | 4,6 | 26 | 4000 |
| Захват | PL42BLF | 24 | 4,6 | 26 | 4000 |

Выбор шаговых двигателей для осей робота обоснован их характеристиками, описанными в работе Болотовского [4]. Драйверы двигателей, один из которых показан на рисунке 6, подбирались с учетом

совместимости по уровням управляющих сигналов (логика TTL 5 В), а также возможности задания направления и скорости вращения. Интерфейс драйвера позволяет управлять положением и скоростью через цифровые выходы Arduino, реализуя как прерывистое, так и непрерывное движение.


Рисунок 6 – Драйвер BLDC-4008

Контакты подключения драйвера приведены в таблице 2.

Таблица 2 – Контакты подключений

| Обозначение | Описание |
|---|---|
| +AT, -AT | Клеммы подключения питающего напряжения |
| U, V, W | Подключение фаз двигателя |
| +5V | Напряжения питания датчиков Холла |
| GND | Общий вывод датчиков Холла |
| HALL A, B, C | Подключение сигнальных выводов датчиков Холла |
| GND | Земля клемм группы управления |
| CW | Вращение по часовой стрелке |
| CCW | Вращение против часовой стрелки |
| DA | Аналоговый вход управления |

Продолжение таблицы 2

| | |
|---|---|
| BRAKE | Тормоз двигателя |
| +5V | Клемма подключения +5В |

Концевой индуктивный датчик LJ12A3-4-Z/BX, показанный на рисунке 7, используется для контроля положения металлических объектов или расстояния до них (активируется наличием металла в зоне срабатывания). Принцип действия индуктивного датчика основан на изменении параметров магнитного поля, создаваемого катушкой индуктивности внутри датчика.

Датчик должен быть установлен чувствительным элементом в сторону зоны измерения. Датчик может быть установлен в монтажное отверстие диаметром 12 мм и глубиной до 37,5 мм. Для этого на корпусе датчика есть две гайки и две шайбы. Датчик герметичен, потому может быть установлен в агрессивной среде. Дистанция срабатывания датчика - 4 мм. На задней торцевой части датчика установлен красный светодиод, который загорается при нахождении в зоне срабатывания металлического объекта.

Датчик имеет транзисторный выход NPN полярности и нормально открытое состояние выхода.

Концевой индуктивный датчик LJ12A3-4-Z/BX имеет три проводника для подключения к контроллеру и питанию. Проводники маркированы цветом: коричневый – напряжение питания, черный – выходной сигнал, синий – общий проводник (300 мА).

Рисунок 7 – Индуктивный датчик LJ12A3-4-Z/BX

Для питания роботом манипулятором подобран блок питания AC-DC MeanWell мощностью 360 Ватт, напряжение 24 В [11]. Используется в 3д-принтерах, робототехнике. Блок питания изображен на рисунке 8. Структурная схема устройства показана на рисунке 9.

Рисунок 8 – Блок питания

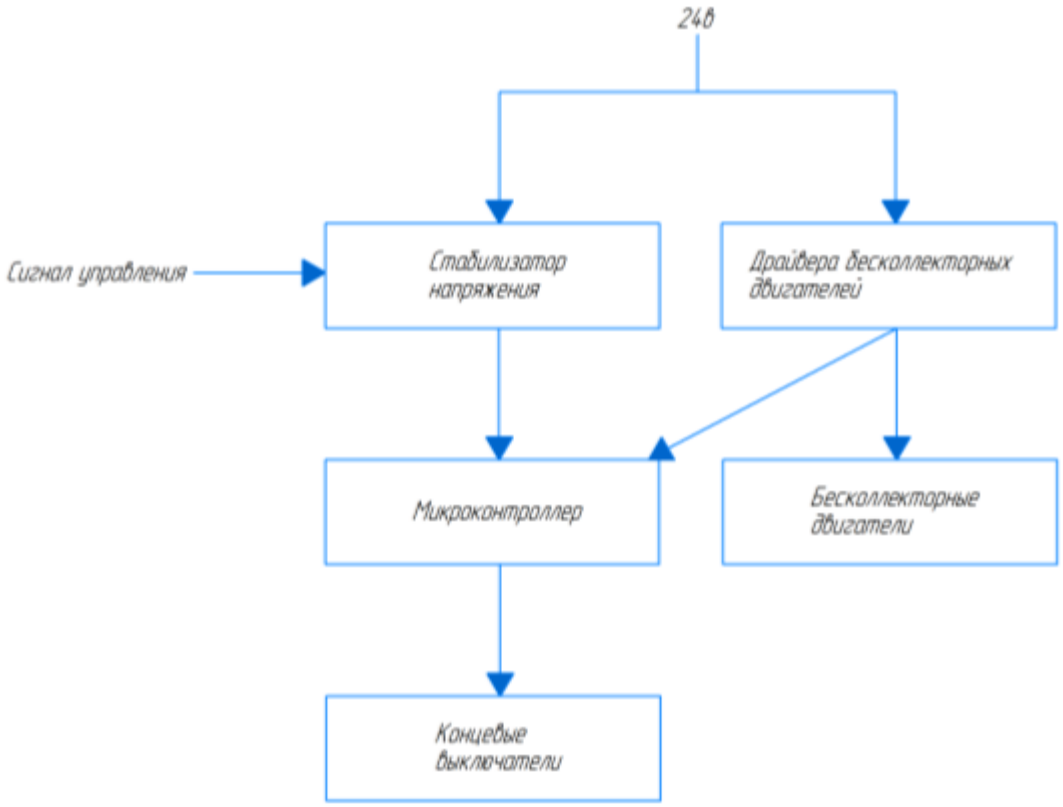



Рисунок 9 – Структурная схема

### 2.1 Проектирование печатной платы

Предлагаемое аппаратное решение, построенное на базе контроллера Arduino Mega и рассчитанное на управление приводами с тремя управляющими сигналами, обладает высокой степенью универсальности и гибкости. Данная плата способна эффективно работать как с бесщеточными двигателями постоянного тока (BLDC), так и с шаговыми двигателями, например Nema 17, что значительно расширяет область её применения, как для учебных, так и для промышленных роботов-манипуляторов. Проектирование печатной платы выполнено с учетом требований к ширине токоведущих линий и зазоров, как описано в методиках разработки электроники [18].

Основным преимуществом такой архитектуры является возможность использования одной и той же аппаратной платформы для управления различными типами приводов, что снижает затраты на разработку специализированных контроллеров для каждого конкретного механизма. Трёхканальная система управления обеспечивает:

Совместимость с BLDC двигателями, позволяя реализовывать алгоритмы бесщеточного коммутирования и обеспечивать плавное и точное управление скоростью и положением ротора за счет гибкой коммутации фаз;

Поддержку шаговых двигателей, где три управляющих сигнала могут применяться для управления шагами и направлениями вращения, сохраняя при этом высокую точность позиционирования. Алгоритмы управления движением робота разработаны с учетом методов, представленных в литературе по робототехническим системам [5].

Архитектура платы с большим количеством универсальных входов/выходов и поддержкой прерываний в микроконтроллере Arduino Mega позволяет адаптировать программное обеспечение под конкретные задачи и конфигурации приводов без необходимости аппаратных изменений. Такая гибкость обеспечивает масштабируемость системы и упрощает интеграцию с различными типами роботов — от манипуляторов до мобильных платформ.

Таким образом, универсальность платы с трехканальным управлением позволяет создавать гибкие робототехнические системы, оптимизированные под широкий спектр задач и приводов, что повышает эффективность разработки и эксплуатации робототехнических комплексов.

На рисунке 10 представлена схемотехника электронных компонентов. Компоновка BLDC подразумевает внешние драйвера приводов, компоновка для шаговых двигателей расчитана на установку драйверов форм-фактора A4988 для учебных роботов-манипуляторов.

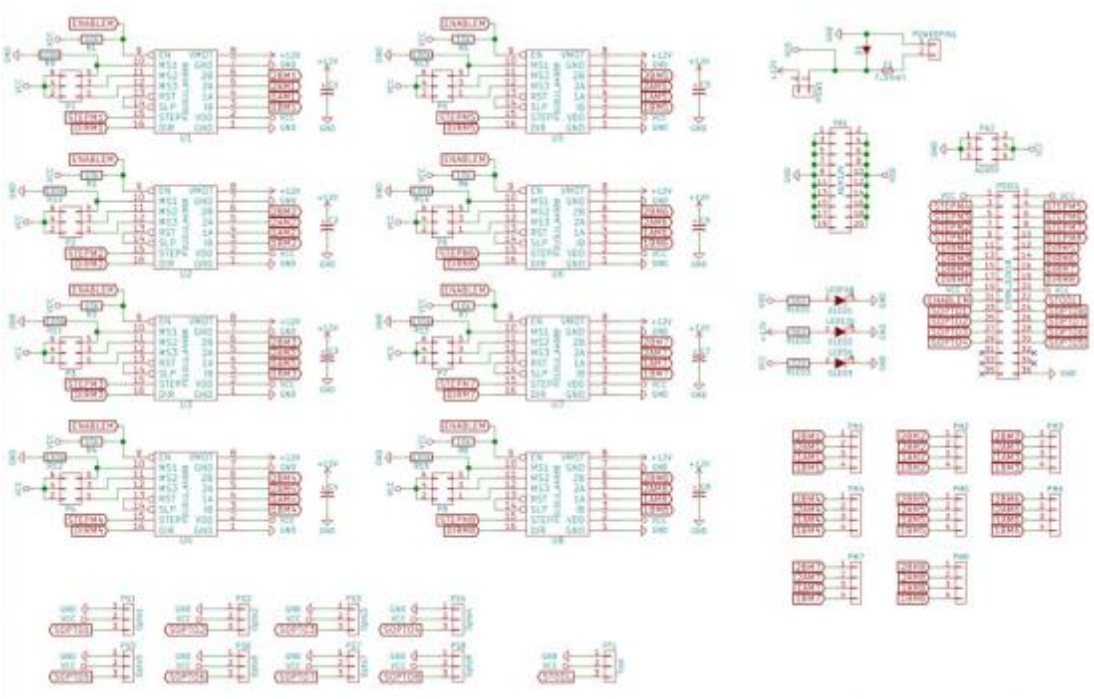


Рисунок 10 – Схемотехника контроллера для шаговых двигателей

На плате, показанной на рисунке 11, предусмотрено до 7 драйверов A4988 или совместимые — подключение к каждому шаговому двигателю, со всеми управляющими сигналами: STEP, DIR, ENABLE и питание +12 В. Соответствующие ножки Arduino шли на входы DIR/STEP драйверов. Возможность подключения до 8 концевых выключателей — по одному на каждый мотор, подключение осуществляется через специальные разъёмы и логические линии на Arduino. Питание реализовано от 12 В, с распределением на силовую шину и выводы для вентиляторов. Имеется также линия +5 В для логики Arduino. Предусмотрена защита от обратной полярности и светодиодные индикаторы состояния — входное питание, логическое питание Arduino и питание шаговиков. В схеме есть выделенный PWM-выход для управления инструментом, например, для вентилятора или другого оборудования — это позволит включать/выключать периферию прямо с платы.

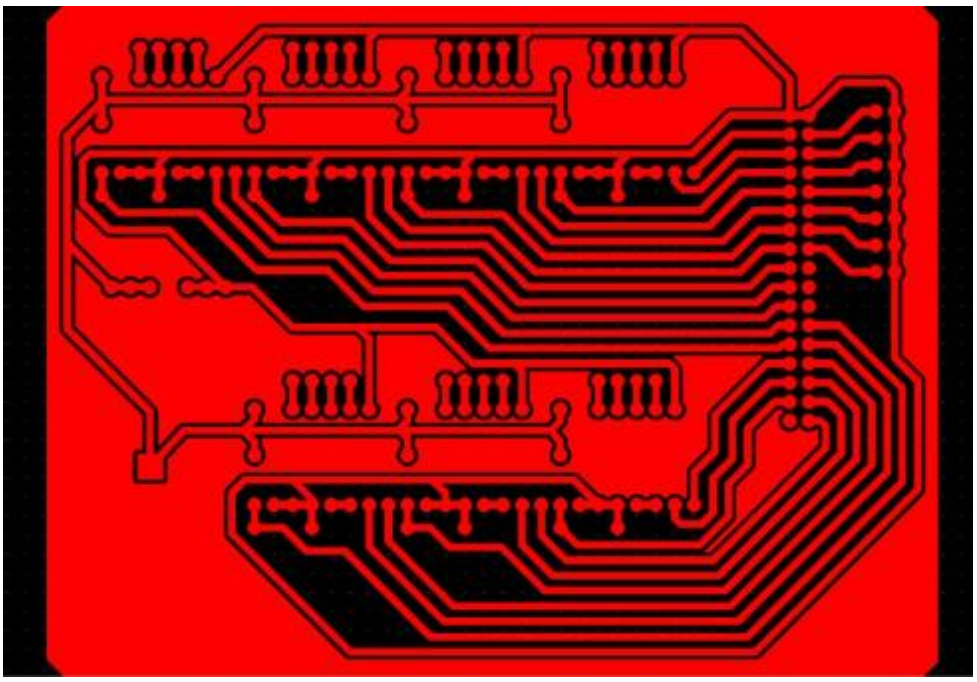


Рисунок 11 – Разведение 7 осевой Shield-платы для Arduino Mega

Проектирование печатной платы осуществлялось в свободно распространяемой САПР KiCad, которая обеспечивает полный цикл разработки электронной аппаратуры — от создания электрической принципиальной схемы до генерации Gerber-файлов и трёхмерной визуализации конечного изделия. Среда предоставляет широкий набор инструментов для размещения компонентов, трассировки соединений, проверки электрических правил (ERC) и правил проектирования (DRC).

На первом этапе была разработана электрическая принципиальная схема, включающая микроконтроллер Arduino Mega, драйверы шаговых двигателей (A4988), цепи питания, концевые выключатели и управляющие выходы. Для каждого логического и силового блока были выбраны и размещены соответствующие элементы из встроенной библиотеки компонентов. Все соединения выполнены с учётом требований электрической целостности и минимизации помех. Для проектирования

электронного блока управления использованы рекомендации по разводке печатных плат из источника [8].

После завершения схемотехники был выполнен переход к этапу трассировки платы в модуле PCB Editor. Плата разработана в двухслойном исполнении, с размещением всех компонентов на верхнем слое. Маршрутизация сигнальных линий выполнена вручную, с учётом минимизации пересечений и длины трасс. Ширина силовых дорожек (для питания двигателей) выбрана с учётом расчёта по допустимому току, проведена через vias для подключения к нижнему силовому слою.

В процессе разработки использовались:

– слои: Top Copper, Bottom Copper, SilkScreen, Edge.Cuts;

– DRC-проверка соблюдения минимальных зазоров, ширины дорожек, правил посадки компонентов;

– расстановка подписи компонентов (RefDes) и маркировки разъёмов;

– определение контуров платы и позиционирование крепёжных отверстий.

KiCad предоставляет возможность трёхмерной визуализации проектируемой платы в реальном времени. Данный инструмент позволил проверить точность размещения компонентов, наличие возможных перекрытий и соответствие габаритных размеров конструкции, как показано на рисунке 12. Генерация 3D-модели платы была выполнена в формате .wrl и .step, что может быть использовано для совмещения с механической частью устройства в CAD-системах (например, SolidWorks или Fusion 360).

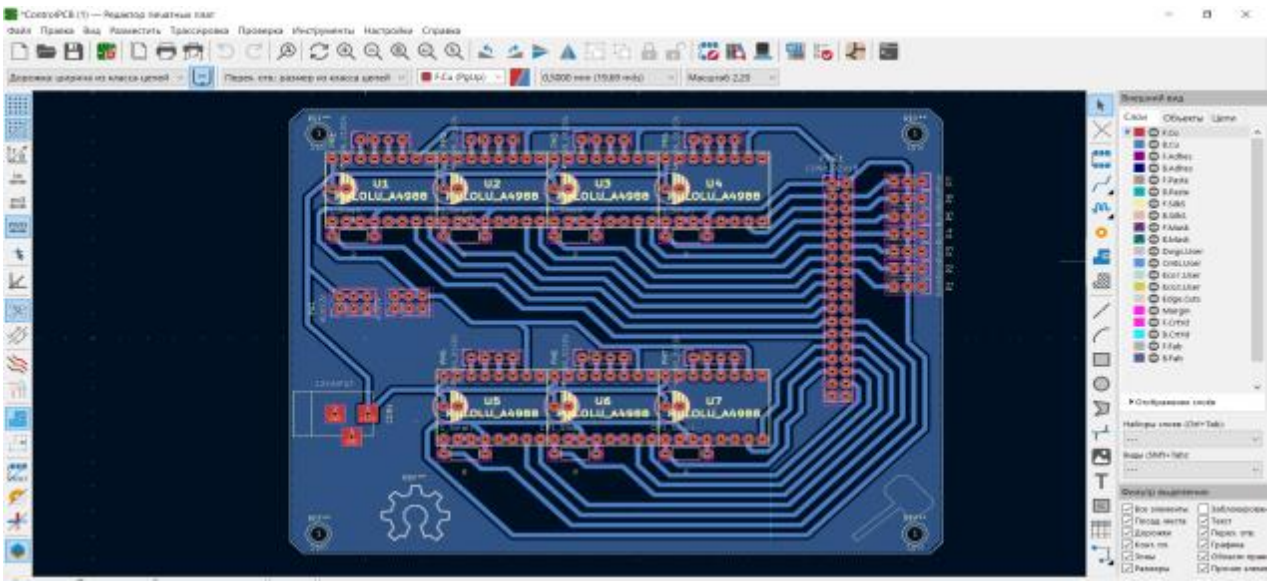


Рисунок 12 – Готовая печатная плата

## 2.2 Сборка системы управления

Для организации системы управления SCARA-манипулятором в промышленном исполнении был собран электрический щит, обеспечивающий размещение, питание и подключение всех ключевых элементов низкого уровня. В основе конструкции использован металлический корпус МЭК ЭЛЕКТРИКА ЩМП-05 габаритами 400×400×155 мм, модель MEC11306, обладающий степенью защиты IP65. Данный электрический щит показан на рисунке 13. Данный тип корпуса обеспечивает пыле и влагозащищённость, механическую прочность и удобство монтажа модулей на стандартной монтажной панели. Для удобства монтажа электронных компонентов в щите предусмотрена пластина для монтажа, которая фиксируется на 4 шпильках М5. Пластина позволяет зафиксировать компоненты в произвольном порядке. Теоретические основы управления системой основаны на классических методах теории автоматического управления, изложенных в [17].

Рисунок 13 – Электрический щит MEC11306

На лицевой панели установлен внешний Ethernet-разъём, показанный на рисунке 14, предназначенный для подключения системы к верхнему уровню — например, к компьютеру с установленной системой управления RoboDK. Связь реализуется через Ethernet–Serial конвертер USR-TCP232-T2, установленный внутри щита и подключённый к микроконтроллеру Arduino Mega.

Рисунок 14 – Внешний разъем Ethernet

Для герметичного ввода кабелей различного сечения в нижней части корпуса используются кабельные вводы, показанные на рисунке 15. Они предотвращают попадание влаги и пыли внутрь щита, а также защищают кабели от излома и перегиба в точке входа. Количество вводов соответствует числу подключаемых внешних узлов: двигателей, датчиков, источников питания и интерфейсов.

Рисунок 15– Кабельные вводы

Для организации подключения периферийных устройств на внешнюю панель корпуса выведены разъёмы серии WEIPU, показанные на рисунке 16, представляющие собой влагозащищённые промышленные соединители с байонетным замком. Такие разъёмы обеспечивают надёжную коммутацию исполнительных модулей, а также быстрое и безопасное подключение при эксплуатации. Система управления манипулятором реализована на основе схем, описанных в учебном пособии Гаврилова [6]

Рисунок 16– Разъемы WEIPU

## 2.3 Разработка низкоуровневого программного обеспечения

В рамках реализации системы управления манипулятором SCARA разработана архитектура низкоуровневого ПО, отвечающая за непосредственное управление исполнительными механизмами и обработку сигналов с датчиков. В качестве микроконтроллерной платформы используется Arduino Mega 2560, которая обеспечивает достаточное количество цифровых входов/выходов, а также несколько портов UART, необходимых для связи с высокоуровневым уровнем управления. Низкоуровневое программное обеспечение разработано с использованием принципов управления шаговыми двигателями, изложенными в справочнике [9]. Основной задачей низкого уровня является выполнение входящих команд, реализация безопасного и детерминированного реагирования на сигналы окружающей среды, а также обеспечение обратной связи.

Программное обеспечение низкого уровня реализовано на языке программирования C++ с использованием среды Arduino IDE, меню которого показано на рисунке 17.

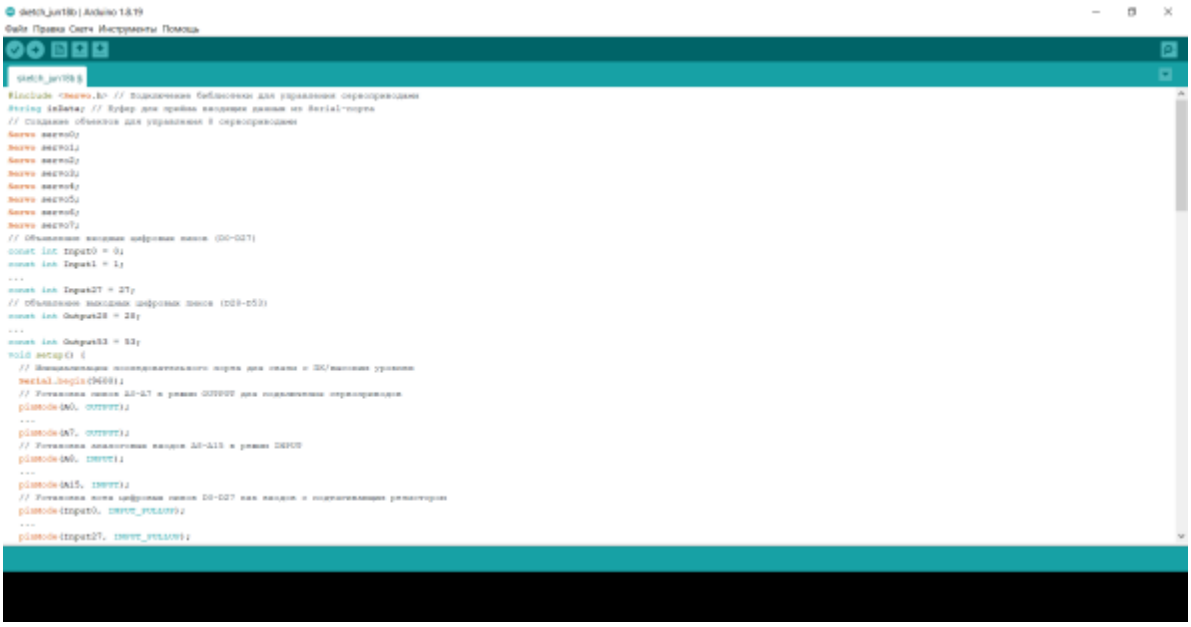


Рисунок 17 – Среда программирование Arduino IDE

Программа организована по модульному принципу и состоит из нескольких функциональных блоков:

Блок инициализации (setup()) — настраивает выводы микроконтроллера, устанавливает режимы работы (вход/выход/подтяжка), привязывает сервоприводы к соответствующим аналоговым выводам и устанавливает их начальные положения. Кроме того, в этом блоке активируется последовательный порт, через который поступают команды управления.

Блок обработки команд (loop()) реализует бесконечный цикл, в котором контроллер считывает данные из буфера UART, формирует командную строку, разбирает ее и вызывает соответствующую процедуру. Каждая команда начинается с идентификатора действия, за которым следуют параметры (например, номер сервопривода и целевое положение).

Функциональные обработчики представлены независимыми логическими структурами, каждая из которых соответствует определенному действию: перемещение сервопривода (SV), установка состояния цифрового

выхода (ON/OF), опрос цифрового входа (JF), ожидание сигнала на входе с таймаутом (WI), а также отладочная команда (TM) для проверки канальных соединений.

Каждая команда передается в виде ASCII-строки и заканчивается символом новой строки \n. Примеры команд:

– SV0P90\n — установить сервопривод с номером 0 в положение 90°;

– ONX35\n — включить цифровой выход D35;

– JFX12T\n — прочитать статус входа D12, вернуть «T» или «F»;

– WIA12B1C3\n — Ожидать уровень 1 на пине A12 в течение 3 секунд.

Полученная строка разбирается по ключевым символам (S, P, X, A, B, C) и преобразуется в параметры для выполнения операции. Программа также формирует ответ — Done, T, F или любой текст — для подтверждения выполнения команды или передачи статуса.

На высоком уровне используется программная платформа RoboDK, реализующая моделирование движения манипулятора, генерацию управляющего кода, визуализацию и планирование траектории. RoboDK не управляет напрямую электромеханическими компонентами, но предоставляет инструменты для связи с внешними контроллерами через API и стандартные протоколы.

Для взаимодействия с Arduino реализована связь через последовательный интерфейс UART, организованный через микросхему-конвертер USR-TCP232-T2, обеспечивающую преобразование Ethernet Serial [10,12блок]. Такой подход позволяет подключать Arduino к управляющему ПК через сеть Ethernet, что обеспечивает удаленную работу и минимизирует электромагнитные помехи в промышленной среде. Альтернативно, при отсутствии сетевого интерфейса, связь может осуществляться напрямую через порт USB COM

На стороне RoboDK с помощью скриптов Python (через модуль pyserial) отправляются строковые команды и принимается ответ от Arduino. Таким образом, каждое движение робота, сформированное в RoboDK, сопровождается соответствующей последовательностью команд, передаваемых на микроконтроллер, где они обрабатываются и преобразуются в конкретные действия на физическом уровне.

Для обеспечения корректного выполнения задач в системе предусмотрена обработка сигналов с индуктивных концевых датчиков, подключенных к цифровым входам микроконтроллера [18]. Программная логика реализует команды чтения состояния входов и команды ожидания события с таймаутом, что позволяет строить алгоритмы синхронного или условного движения. Особенно это актуально для парковочных, калибровочных или переходных операций.

Кроме того, предусмотрена установка начальных положений исполнительных механизмов и возможность определения границ движения на уровне микроконтроллера, что повышает надежность системы и ограничивает возможность возникновения аварийных ситуаций.

Существующая архитектура легко масштабируется: при необходимости может быть добавлена поддержка дополнительных степеней свободы, реализован протокол управления буфером с предварительной загрузкой команд, введена логика контроля ошибок. Также возможна реализация протоколов более высокого уровня, таких как Modbus RTU или CANopen, если они предполагаются для использования в составе распределенных систем автоматизации.

```
#include <Servo.h> // Подключение библиотеки для управления
сервоприводами
String inData; // Буфер для приёма входящих данных из Serial-порта
// Создание объектов для управления 8 сервоприводами
Servo servo0;
Servo servo1;
```

```
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;
Servo servo7;
// Объявление входных цифровых пинов (D0–D27)
const int Input0 = 0;
const int Input1 = 1;
...
const int Input27 = 27;
// Объявление выходных цифровых пинов (D28–D53)
const int Output28 = 28;
...
const int Output53 = 53;
void setup() {
  // Инициализация последовательного порта для связи с ПК/высоким уровнем
  Serial.begin(9600);
  // Установка пинов A0–A7 в режим OUTPUT для подключения сервоприводов
  pinMode(A0, OUTPUT);
  ...
  pinMode(A7, OUTPUT);
  // Установка аналоговых входов A8–A15 в режим INPUT
  pinMode(A8, INPUT);
  ...
  pinMode(A15, INPUT);
  // Установка всех цифровых пинов D0–D27 как входов с подтягивающим резистором
  pinMode(Input0, INPUT_PULLUP);
  ...
  pinMode(Input27, INPUT_PULLUP);
  // Установка цифровых пинов D28–D53 как выходов
  pinMode(Output28, OUTPUT);
  ...
  pinMode(Output53, OUTPUT);
  // Привязка объектов сервоприводов к пинам A0–A7
  servo0.attach(A0);
  ...
```

```
  servo7.attach(A7);
  // Инициализация положения сервопривода 0 (например, парковка)
  servo0.write(20);
  // Активация первых выходов (например, для включения питания
исполнительных модулей)
  digitalWrite(Output28, HIGH);
  ...
  digitalWrite(Output35, HIGH);
}
void loop() {
  // Основной цикл: проверка входящих данных по Serial
  while (Serial.available() > 0) {
    char recieved = Serial.read(); // Чтение символа из входного буфера
    inData += recieved; // Добавление символа в буфер строки
    // Обработка команды после получения символа новой строки '\n'
    if (recieved == '\n') {
      String function = inData.substring(0, 2); // Извлечение кода команды
(первые 2 символа)
      // === Команда управления сервоприводом ===
      if (function == "SV") {
        int SVstart = inData.indexOf('V');
        int POSstart = inData.indexOf('P');
        int servoNum = inData.substring(SVstart + 1, POSstart).toInt();
        int servoPOS = inData.substring(POSstart + 1).toInt();
        // Установка положения сервопривода по номеру
        if (servoNum == 0) servo0.write(servoPOS);
        if (servoNum == 1) servo1.write(servoPOS);
        if (servoNum == 2) servo2.write(servoPOS);
        if (servoNum == 3) servo3.write(servoPOS);
        if (servoNum == 4) servo4.write(servoPOS);
        if (servoNum == 5) servo5.write(servoPOS);
        if (servoNum == 6) servo6.write(servoPOS);
        if (servoNum == 7) servo7.write(servoPOS);
        Serial.print("Servo Done"); // Подтверждение выполнения
      }
      // === Команда проверки цифрового входа и возврат статуса ===
      if (function == "JF") {
        int IJstart = inData.indexOf('X');
        int IJTabstart = inData.indexOf('T');
        int IJInputNum = inData.substring(IJstart + 1, IJTabstart).toInt();
```

```
  if (digitalRead(IJInputNum) == HIGH) {
    delay(5);
    Serial.println("T"); // True — высокий уровень
  }
  if (digitalRead(IJInputNum) == LOW) {
    delay(5);
    Serial.println("F"); // False — низкий уровень
  }
}
// === Команда включения выхода ===
if (function == "ON") {
  int ONstart = inData.indexOf('X');
  int outputNum = inData.substring(ONstart + 1).toInt();
  digitalWrite(outputNum, HIGH);
  Serial.print("Done");
}
// === Команда выключения выхода ===
if (function == "OF") {
  int ONstart = inData.indexOf('X');
  int outputNum = inData.substring(ONstart + 1).toInt();
  digitalWrite(outputNum, LOW);
  Serial.print("Done");
}
// === Команда ожидания сигнала на входе с тайм-аутом ===
if (function == "WI") {
  int inputVal = -1;
  int inputIndex = inData.indexOf('A');
  int valueIndex = inData.indexOf('B');
  int timoutIndex = inData.indexOf('C');
  int input = inData.substring(inputIndex + 1, valueIndex).toInt();
  int value = inData.substring(valueIndex + 1, timoutIndex).toInt();
  int timeout = inData.substring(timoutIndex + 1).toInt();
  unsigned long timeoutMillis = timeout * 1000;
  unsigned long startTime = millis();
  // Цикл ожидания сигнала на входе
  while ((millis() - startTime < timeoutMillis) && (inputVal != value)) {
    inputVal = digitalRead(input);
    delay(100);
  }
  delay(5);
```

```
      Serial.print("Done");
    }
    // === Команда теста связи (echo-тест) ===
    if (function == "TM") {
      String echo = inData.substring(2); // Всё, что после TM
      Serial.println(echo); // Возврат строки
    }
    else {
      // Очистка буфера данных после обработки команды
      inData = "";
    }
  }
 }
}
```

Вывод по разделу 2

В рамках реализации системы управления манипулятором SCARA была разработана архитектура низкоуровневого программного обеспечения, обеспечивающая непосредственное взаимодействие с исполнительными механизмами и датчиками. В качестве аппаратной платформы выбран микроконтроллер Arduino Mega 2560, который предоставляет достаточное количество цифровых и аналоговых портов, необходимое для расширяемости системы, а также поддерживает несколько последовательных интерфейсов UART.

Разработанная печатная плата представляет собой компактное и универсальное решение для управления промышленным роботом-манипулятором SCARA. Она обеспечивает надежное соединение всех необходимых компонентов и предоставляет возможность масштабирования функционала за счет большого количества выводов. Компактный форм-фактор и простота производства делают эту плату подходящей для как учебных, так и промышленных применений.

Программное обеспечение написано на языке C++ в среде Arduino IDE и структурировано по модульному принципу, что обеспечивает высокую

степень читаемости, гибкости и возможности масштабирования. Архитектура ПО включает следующие ключевые компоненты:

– функция setup() — производит начальную настройку всех портов ввода/вывода, инициализирует сервоприводы, устанавливает их начальное положение и активирует последовательный порт для связи с внешними системами;

– функция loop() — реализует циклический опрос входящих команд через последовательный порт, обработку и выполнение этих команд;

Функциональные блоки обработки команд:

– управление сервоприводами (SV);

– управление дискретными выходами (ON/OF);

– чтение состояния цифровых входов (JF);

– ожидание события с таймаутом (WI);

– отладочная команда (TM).

Каждая команда передается в виде ASCII-строки, завершающейся символом перевода строки \n, что позволяет использовать простой текстовый протокол для взаимодействия между уровнями системы.

Для связи с высокоуровневым контроллером используется последовательный интерфейс UART, преобразуемый через USR-TCP232-T2, что обеспечивает сетевое взаимодействие с управляющим компьютером по протоколу Ethernet. Это решение повышает помехозащищенность, удобство подключения и позволяет организовать удалённое управление роботом [20].

На стороне верхнего уровня применяется программный комплекс RoboDK, который отвечает за моделирование, планирование траекторий и генерацию управляющих команд. RoboDK взаимодействует с Arduino через Python-скрипты, используя библиотеку pyserial. Таким образом, каждая сгенерированная в RoboDK операция движения преобразуется в соответствующую команду, отправляемую на микроконтроллер.

Система управления допускает расширение функциональности:

– добавление новых степеней свободы;

– реализация буферизации команд;

– внедрение протоколов промышленной автоматизации (Modbus RTU, CANopen);

– интеграция систем диагностики и самотестирования.

Разработанное ПО реализует безопасное и детерминированное управление движением манипулятора, поддерживает работу с концевыми датчиками, ограничениями по движению и начальной калибровкой, что делает систему надежной и адаптируемой к различным задачам автоматизации.

Таким образом, реализованная архитектура управления манипулятором SCARA объединяет аппаратные и программные средства, обеспечивает эффективное взаимодействие между уровнями управления и готова к применению как в учебных, так и в промышленных условиях.

# 3 Разработка высокоуровневого программного обеспечения

В рамках реализации высокоуровневой системы управления манипулятором SCARA в данной работе использовалась программная платформа RoboDK, которая на данный момент является одним из самых универсальных и доступных решений для офлайн-программирования промышленных роботов. Преимущество RoboDK заключается в комплексном подходе к автоматизации задач: она объединяет моделирование, визуализацию, калибровку и генерацию управляющего кода для различных типов исполнительных механизмов. В отличие от открытых фреймворков, таких как ROS, которые ориентированы в первую очередь на академические и исследовательские среды, RoboDK предоставляет интегрированную графическую среду с готовыми инструментами промышленного уровня. Это существенно сокращает время разработки и отладки, исключая необходимость ручного описания кинематических моделей и низкоуровневых интерфейсов.

Использование RoboDK особенно эффективно при работе с манипуляторами SCARA, поскольку платформа содержит встроенные шаблоны для таких кинематических структур и предоставляет инструменты для тонкой настройки всех геометрических и кинематических параметров без необходимости доступа к исходному коду или файлам конфигурации. Пользователь может загрузить собственную модель манипулятора, созданную в любой CAD-системе, и на ее основе создать полноценную цифровую модель с указанием базовой системы координат, рабочего инструмента и технологических ограничений. В процессе построения цифрового двойника разрабатываются и отлаживаются все ключевые операции, от базового позиционирования до сложных траекторий с учетом кинематических ограничений, ускорений и зон безопасности.

Особое значение в рамках проекта имеет возможность калибровки системы координат и привязки рабочего инструмента, что критически важно при работе с физическим прототипом манипулятора SCARA. RoboDK предоставляет встроенные инструменты для трехмерной калибровки как базового положения робота, так и инструмента (TCP), что позволяет обеспечить высокую точность перехода от моделирования к реальному управлению. После завершения этапа моделирования и имитации формируется управляющий код, который может быть адаптирован под конкретный контроллер, как промышленный, так и заказной, построенный, например, на базе Arduino или STM32. Благодаря встроенным постпроцессорам код может быть экспортирован в формат G—code, RAPID, KRL, а при необходимости и в формат, совместимый с пользовательской прошивкой.

Таким образом, реализация манипулятора SCARA в среде RoboDK начинается с подготовки 3D-модели и описания кинематики, продолжается разработкой виртуального технологического процесса с отладкой всех операций, а затем переходит к фазе генерации кода и калибровки реального оборудования. Такой подход минимизирует риски, связанные с ошибками управления, повышает надежность системы, сокращает время настройки и обеспечивает воспроизводимость результатов. В целом использование RoboDK обеспечивает гибкость, масштабируемость и промышленную надежность, что делает его оптимальным выбором для создания высокоуровневой системы управления манипулятором SCARA в рамках данной работы.

На рисунке 18 показано основное окно среды RoboDK, представляющее собой 3D-сцену моделирования. Слева — панель объектов, в которой отображаются: загруженные роботы, траектории, инструменты, пользовательские объекты.

В центре расположено интерактивное рабочее пространство, где можно наблюдать за симуляцией движения манипулятора, контролировать траекторию TCP (Tool Center Point) и изменять параметры в реальном времени.

Такой интерфейс позволяет проектировать операции без необходимости подключения реального оборудования.

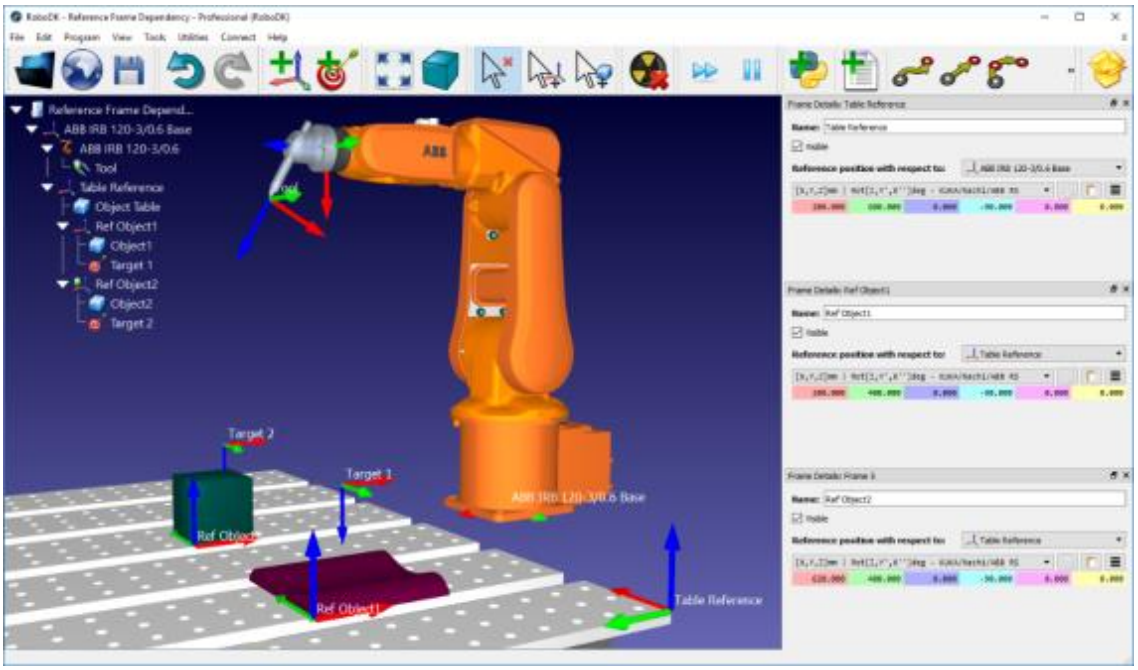



Рисунок 18 – Основное окно среды RoboDK

Рисунок 19 иллюстрирует меню кастомизации кинематики. В RoboDK можно задать собственную конфигурацию робота, указав: количество степеней свободы, тип каждой оси (вращательная или поступательная), диапазон перемещений, DH-параметры. Данный инструмент особенно важен при моделировании SCARA-манипуляторов, где структура осей отличается от классической антропоморфной схемы.

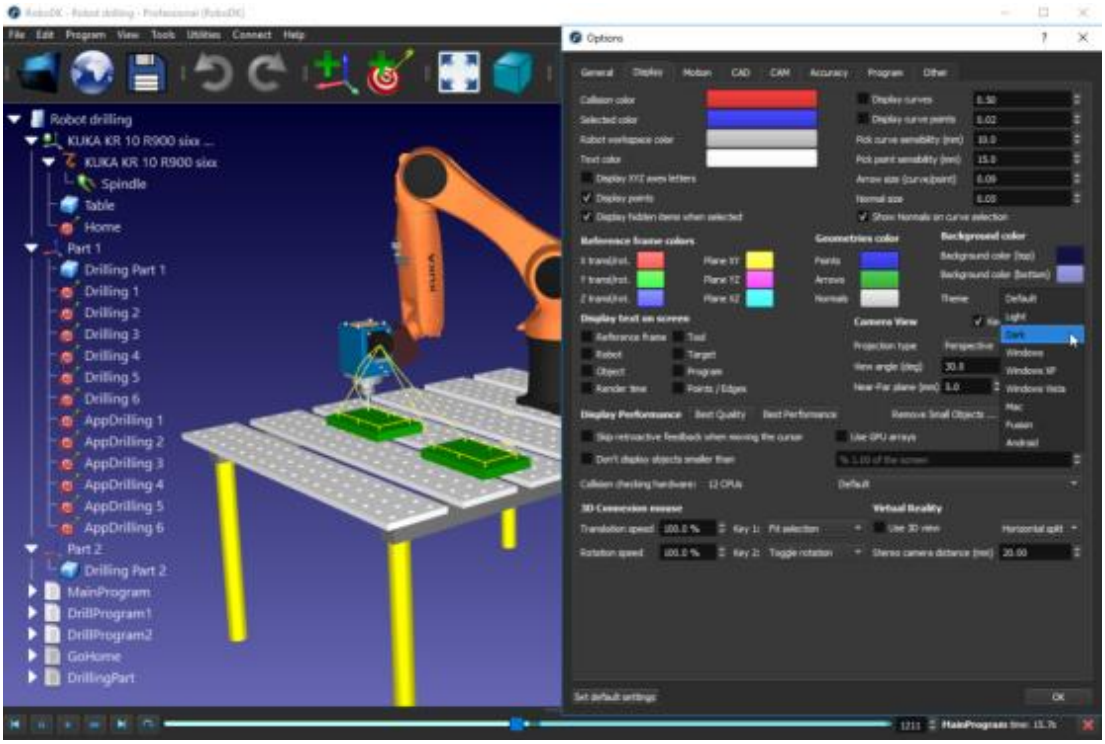



Рисунок 19 – Меню кастомизации кинематики

Интерфейс задания траектории показан на рисунке 20. Пользователь может выбрать: тип движения (линейное, круговое, в точку), скорость, ускорение, плавность, опорные координаты (Base, TCP), порядок осей (инверсная кинематика). С помощью этих настроек осуществляется точное позиционирование, программирование сборки, захвата или манипулирования объектами.

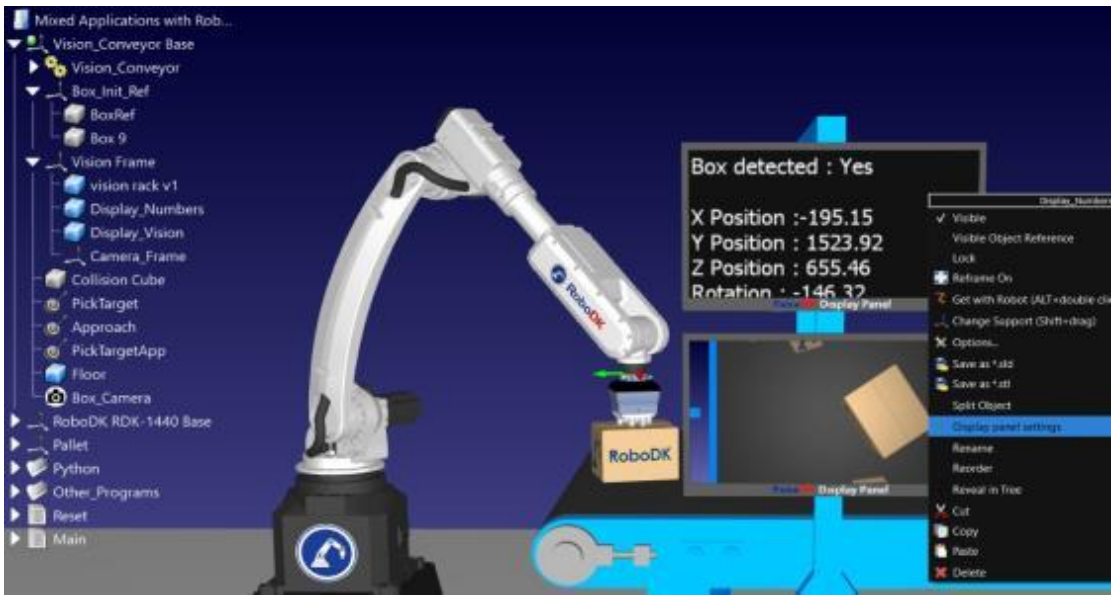



Рисунок 20 – Интерфейс задания траектории

На рисунке 21 показан модуль экспорта управляющих программ. В RoboDK доступна генерация G-кода или языков программирование RAPID, KRL, URScript и др., в зависимости от типа робота. Также поддерживается пользовательская генерация — например, под Arduino, GRBL, STM32 или Modbus-совместимые интерфейсы. Для каждого проекта можно задать пользовательский постпроцессор, который преобразует визуальные движения в готовую программу для загрузки в контроллер.

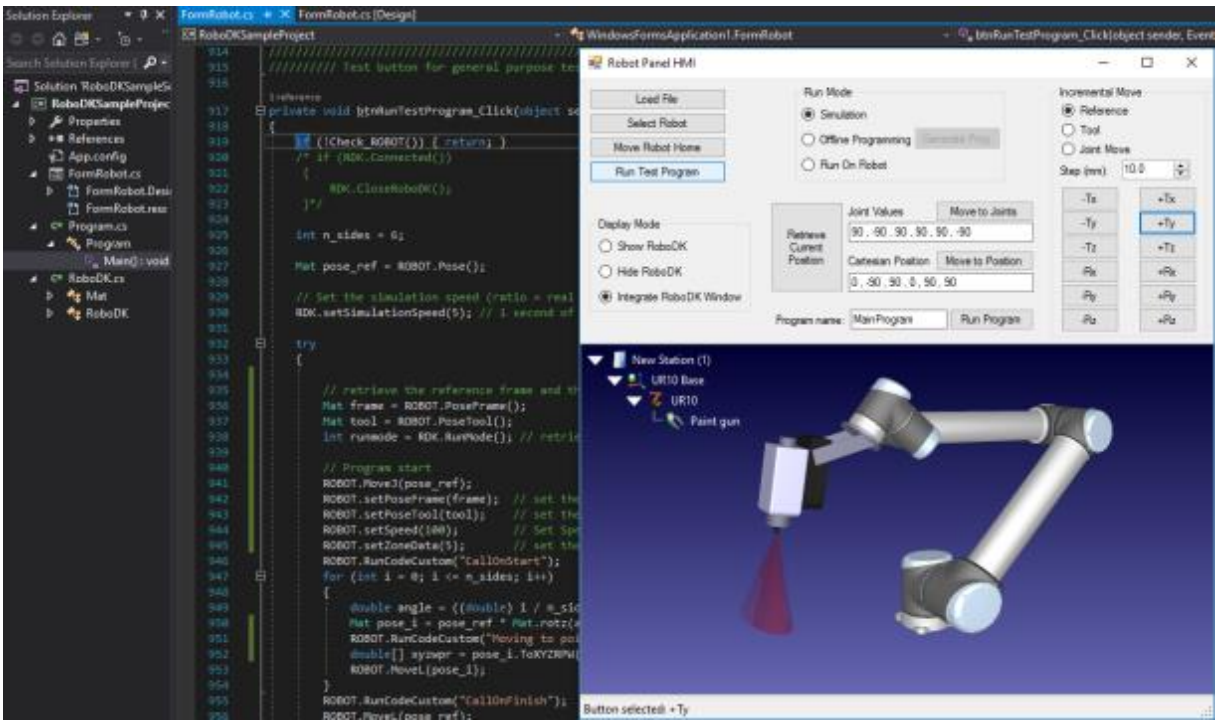



Рисунок 21 – Модуль экспорта управляющих программ

### 3.1 Интеграция SCARA-робота с искусственным интеллектом

Интеграция технологий AI (artificial intelligence) существенно расширяет возможности управления SCARA- манипуляторами. В частности, компьютерное зрение на основе нейросетей позволяет роботам распознавать положение и ориентацию объектов в реальном времени. Интеграция искусственного интеллекта в управление роботом основана на методах оптимизации, изложенных в работах [1]. Например, сочетание SCARA-робота с продвинутой системой машинного зрения дает ему способность точно захватывать детали в произвольной ориентации, снижая время цикла и повышая гибкость производства. Компьютерное зрение также используется для автоматической визуальной инспекции: алгоритмы машинного обучения способны обнаруживать на изображениях мелкие дефекты продукции, недоступные невооружённому глазу, что ценно для отраслей с высокими

требованиями к качеству (электроника, медицина).Кроме того, нейросети применяются для избежания столкновений и планирования траекторий: исследователи, например, успешно обучили рекуррентные нейросети планировать движения SCARA-манипулятора с учётом движущихся препятствий, выполняя одновременное слежение за целью и предотвращение столкновений.

AI-технологии позволяют роботам обучаться движениям без точной математической модели. В отличие от классических методов, требующих трудоёмкой идентификации динамики, алгоритмы обучения с подкреплением и итеративного обучения могут самостоятельно настраивать управление. К примеру, метод Autonomous Iterative Motion Learning (AI-MOLE) продемонстрировал, что SCARA-робот способен достигать высокоточной траектории (<1 мм отклонения) уже спустя~10–15 итераций обучения без априорной модели и ручной подстройки параметров. Подобные нейросетевые контроллеры компенсируют нелинейности (трение, люфты), улучшая точность позиционирования. В реальных промышленных условиях это означает, что робот может адаптивно улучшать свои движения по мере накопления данных, становясь точнее и эффективнее.

Помимо управления движением, AI применяется для интеллектуального мониторинга и предиктивного обслуживания роботов. Внедрение датчиков IoT и анализа больших данных позволяет непрерывно отслеживать состояние приводов и механики SCARA. Например, датчики вибрации, температуры и тока могут собирать телеметрию, которую затем анализирует облачная AI-платформа. На основе этих данных AI выявляет отклонения от нормы и прогнозирует износ узлов. Такой подход к предиктивному обслуживанию предупреждает поломки: система заранее планирует ремонт, снижая незапланированные простои оборудования. Практический пример – анализ состояния редукторов SCARA: нейросеть по датчикам температуры, давления и наличия частиц износа в масле способна

предсказать потребность в замене до фактического отказа. В результате внедрения AI-мониторинга производство повышает надёжность и время безотказной работы робототехники. Кроме того, с помощью AI-аналитики можно оптимизировать сам процесс: алгоритмы изучают статистику операций робота и предлагают улучшения для ускорения цикла или повышения точности сборки. Совокупно интеграция AI делает SCARA-манипуляторы более автономными, адаптивными и “умными”, способными обучаться на опыте и подстраиваться под изменяющиеся условия производства. Это особенно важно в контексте концепций Industry 4.0, где даже устаревшие модели роботов могут быть модернизированы AI-модулями. Так, в одном эксперименте к старому SCARA-роботу Sony SRX-611, изначально не предназначенному для машинного зрения, добавили модуль глубокого обучения (например, на базе распознавания объектов YOLO). В результате “умная” надстройка позволила этому роботу выполнять задачи детекции и сортировки объектов в реальном времени без затрат на приобретение нового оборудования. Такой подход демонстрирует, что искусственный интеллект расширяет сенсорные возможности SCARA-манипуляторов, повышая их производительность и продлевая жизненный цикл за счёт программных улучшений.

### 3.2 Анализ программных платформ RoboDK и ROS

RoboDK и ROS (Robot Operating System) представляют два различных подхода к программному обеспечению для управления роботами. Высокоуровневое ПО реализовано с использованием библиотек ROS, что соответствует современным подходам к робототехнике, описанным в литературе [21]. RoboDK – это коммерческая среда для офлайн-программирования и симуляции роботов, тогда как ROS – открытая модульная платформа (middleware) для создания робототехнических

приложений. Сравнительный анализ RoboDok и ROS проводился по критериям, аналогичным подходам к промышленной автоматизации, описанным в [2]. Ниже приведён сравнительный анализ этих платформ с учётом архитектуры, поддержки SCARA, совместимости с аппаратным обеспечением и практических примеров.

Архитектура и назначение: RoboDK имеет монолитную архитектуру с графическим интерфейсом, ориентированную на простое моделирование и генерирование управляющих программ. Пользователь может импортировать 3D-модель SCARA-манипулятора и с помощью встроенных средств калибровки и эмуляции движений быстро составлять траектории. ROS же не является единым приложением – это набор библиотек и служб, работающих по распределённой архитектуре публикация/подписка. ROS-система состоит из множества узлов (процессов), обменивающихся сообщениями; она требует более сложной настройки, но обеспечивает большую гибкость. По сути, ROS – это не симулятор, а промежуточное ПО для робототехники, часто используемое совместно с симуляторами, оди н из которых показан на рисунке 22 (Gazebo, RViz и др.). Следует отметить, что порог вхождения у ROS выше: например, симуляция в Gazebo и изучение топиков и служб ROS требует значительных усилий, тогда как RoboDK изначально рассчитан на интуитивное применение в промышленности. В то же время ROS изначально задуман для облегчения портирования и повторного использования кода между разными роботами. В контексте SCARA- манипуляторов это значит, что управление, однажды разработанное в ROS (например, алгоритм захвата деталей), относительно легко адаптировать к другой модели SCARA через изменение параметров URDF-модели, тогда как в RoboDK для каждой модели робот имеет собственный конфигурируемый пост-процессор.

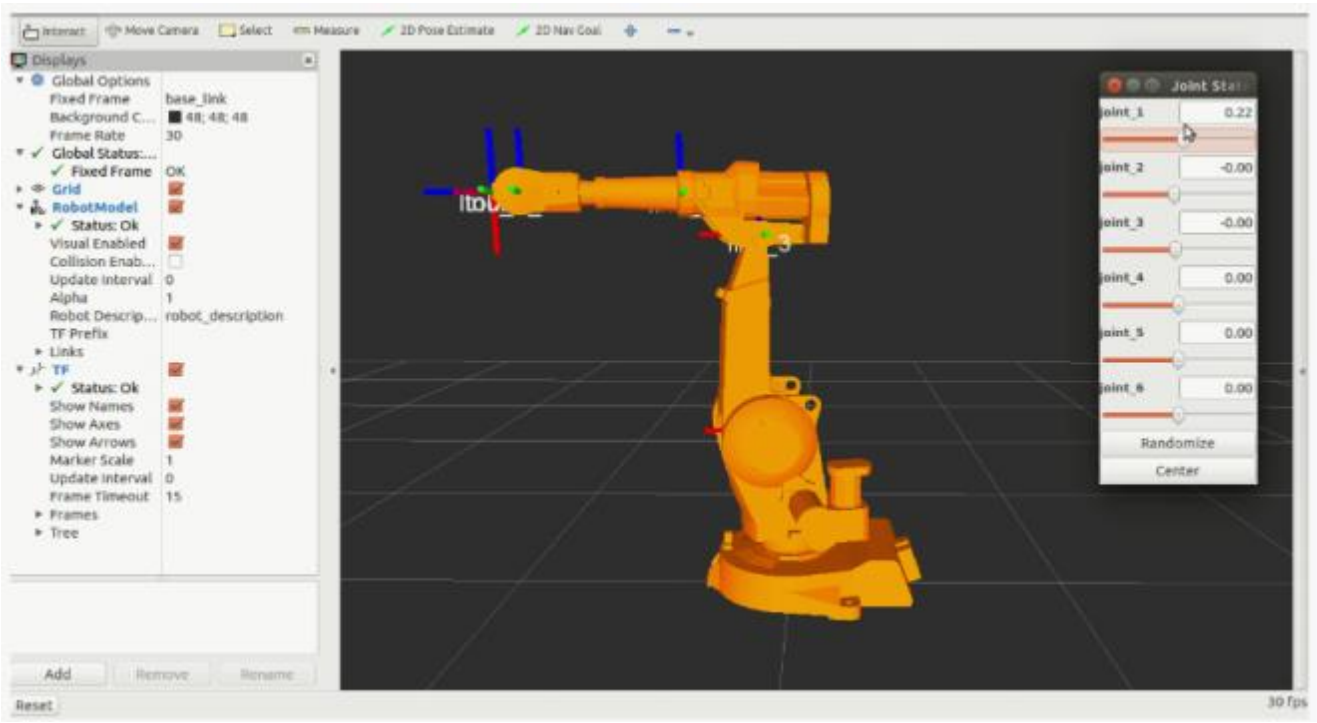



Рисунок 22 – Меню симулятора Gazebo

Поддержка SCARA и библиотек: RoboDK из коробки поддерживает множество промышленных SCARA-роботов. В библиотеке RoboDK представлены десятки готовых моделей SCARA различных производителей – пользователь может выбрать, например, Epson, Yamaha, Staubli или Fanuc SCARA, и сразу приступить к программированию. RoboDK учитывает кинематику SCARA (4 оси: три вращательных и одна линейная) и предлагает упрощённый интерфейс для их калибровки и программирования.

Более того, разработчики RoboDK внедрили специальный решатель обратной кинематики для SCARA, позволяющий, например, легко инвертировать ориентацию осей при креплении робота к потолку, без ручной перенастройки модели. Это упрощает интеграцию SCARA-роботов в различные конфигурации. В ROS нет специализированного “SCARA-модуля” – вместо этого используется общее описание робота через URDF (Unified Robot Description Format). Требуется задать геометрию звеньев и

сочленений SCARA, после чего можно применять стандартные пакеты: для кинематики (библиотеки KDL или MoveIt! для планирования траекторий), для управления приводами (ros_control) и т.д.

Многие производители или сообщество предоставляют готовые ROS-пакеты для своих роботов, но для редкого или самодельного SCARA робота интеграция в ROS потребует создания своей модели и настроек контроллеров. Преимущество ROS – поддержка богатого экосистемой пакетов: например, можно подключить камеру и использовать пакет компьютерного зрения для отслеживания объектов, затем вычисленные координаты передавать узлу планирования траектории SCARA. Таким образом, ROS обеспечивает более широкие возможности расширения, тогда как RoboDK предлагает удобство и надежность готовых решений.

Примеры реализации: В промышленности RoboDK часто применяют для офлайн- программирования: инженер загружает модель рабочего места, программирует траекторию, а затем генерирует управляющий код для конкретного контроллера робота. Например, для SCARA Epson можно сгенерировать программу на языке Epson (SPEL) или универсальный G-код. RoboDK сам по себе выполняется на ПК (Windows, Linux); он может напрямую передавать траектории в контроллер робота или экспортировать файлы. В образовательных и любительских проектах RoboDK тоже находит применение – так, существуют примеры подключения RoboDK к самодельным роботам через Arduino. Пользователь может настроить Arduino как приемник G- кода и использовать RoboDK для визуализации и вычисления траекторий, хотя Arduino по мощности ограничен и не выполнит сложную кинематику в реальном времени. Один из энтузиастов, к примеру, импортировал в RoboDK 3D-модель самодельного SCARA, запрограммировал движения в среде, а затем отправил последовательность координат на Arduino с прошивкой GRBL, управляющей шаговыми двигателями (Arduino выступал как драйвер нижнего уровня). ROS,

напротив, чаще применяют для онлайн-управления роботами с возможностью обратной связи.

Совместимость с аппаратной частью: RoboDK требует для работы полноценный компьютер (настольный или одноплатный). Запустить RoboDK на микроконтроллере невозможно – он слишком требователен к ресурсам; минимально нужен CPU архитектуры x86/ARM с ОС (есть вариант установки RoboDK на Raspberry Pi с Linux). Таким образом, RoboDK не управляет железом напрямую, а выступает как внешняя система программирования, взаимодействующая с робототехническим контроллером через генерируемый код или сетевые API. ROS же изначально рассчитан на распределённую работу разных устройств. Существуют специальные решения для портирования ROS-функционала на микроконтроллеры: например, rosserial (для ROS 1) или Micro- ROS (для ROS 2), позволяющие «встраивать» узлы ROS на платы Arduino, STM32 и др.. В реальных проектах небольшие контроллеры (STM32, Esp32) могут выступать полноценными участниками ROS-сети – получая, к примеру, команды положения по serial или CAN-шине и отдавая в ROS сообщения о показаниях энкодеров. Кроме того, ROS совместим со множеством датчиков и устройств: от 3D-камер Intel RealSense до манипуляторов, подключаемых по EtherCAT. RoboDK же ограничен поддержкой тех контроллеров, под которые разработаны пост-процессоры и драйверы; он в меньшей степени рассчитан на самодельные контроллеры.

Преимущества и недостатки: Обобщая, RoboDK привлекает простотой использования и ориентацией на промышленные задачи «из коробки». Его плюсы – дружественный интерфейс, быстрый старт (включая обширную библиотеку моделей SCARA), наличие визуальных отладочных средств и автоматическая генерация безошибочного кода. RoboDK особенно эффективен для задач офлайн-программирования, где нужно минимизировать время простоя оборудования: программу отлаживают в виртуальной среде и потом сразу запускают на реальном роботе. К

ограничениям RoboDK можно отнести лицензионную модель (платное ПО) и меньшую гибкость для нестандартных применений – глубокая модернизация функциональности затруднена, поскольку исходный код закрыт. ROS, будучи open-source, бесплатен и невероятно расширяем, что является его главным плюсом. Он позволяет реализовать практически любую логику управления, интегрировать слои высокого уровня – SLAM, контроль группы роботов, голосовое управление и т.д. – в единое приложение. ROS имеет огромное сообщество и множество готовых пакетов, что ускоряет разработку. Недостатки ROS вытекают из его сложности: крутая кривая обучения, высокие требования к навыкам разработчиков и отладке. В отличие от RoboDK, где многие вещи интуитивно понятны, в ROS даже запуск простого SCARA-манипулятора требует понимания форматów URDF, настройки контроллеров позиций, написания узлов или конфигурации MoveIt. Кроме того, ROS-систему нужно поддерживать (обновлять версии, следить за совместимостью пакетов). В производственных условиях с ограниченным штатом инженеров это может быть затруднительно, тогда как коммерческая поддержка RoboDK снимает часть проблем. Наконец, надёжность и реальное время: RoboDK выдаёт готовый код для фирменного контроллера, который, как правило, работает в жёстком реальном времени. ROS же обычно работает под Linux; для задач, требующих детерминизма (например, высокоскоростное управление по циклу 1 kHz), нужно дополнительно внедрять реального времени расширения или выносить критичные контуры на микроконтроллеры. Подытоживая, выбор между RoboDK и ROS зависит от задачи: для быстрой реализации типовой задачи (скажем, сборка или упаковка на SCARA) с существующим роботом выгоднее RoboDK, а для экспериментов, научных исследований или сложных многокомпонентных систем (робот+камера+датчики) – ROS.

### 3.3 SCARA-манипуляторы: промышленные и DIY-сегмент

SCARA-манипуляторы выпускаются в широком спектре – от высокоточных промышленных моделей до самодельных или учебных конструкций. Рассмотрим несколько примеров и сравним их характеристики, архитектуры и области применения.

Промышленные SCARA-роботы: На рынке доминируют компании Epson, Yamaha, Staubli, Mitsubishi, Omron, Fanuc, ABB и др., предлагающие SCARA для быстрых сборочных операций. Промышленные SCARA обычно обладают нагрузочной способностью от нескольких килограммов до ~20 кг и выдающейся повторяемостью порядка сотых долей миллиметра. Например, SCARA серии Omron/Adept Cobra способны повторно позиционировать инструмент с точностью~0,017 мм, выполняя до 120 циклов “взять-переставить” в минуту. Компания Fanuc выпустила линейку SCARA SR-3iA/6iA/12iA с рабочим радиусом от 400 до 1100 мм и грузоподъёмностью до 20 кг. Модель Fanuc 2000iA, показанный на рисунке 23, при вылете 650 мм. переносит 6 кг, обеспечивая повторяемость ±0,01 мм– фактически, её система приводов и датчиков допускает отклонения не более десятой доли сотого миллиметра. Добиться такой точности позволяют прецизионные энкодеры и жёсткая конструкция без люфтов [22]. Промышленные SCARA характеризуются также очень высокой скоростью перемещений: линейные оси развивают скорость порядка несколько м/с (например, ось Z SCARA Fanuc движется со скоростью до 2000 мм/с). Конечно, у индустриальных SCARA есть ограничения – обычно это сравнительно малая рабочая область (длина плеча ~0.5–1 м) и негибкость по ориентации инструмента (по сравнению с 6-осевым манипулятором). Тем не менее, для своего класса в задачах – быстром горизонтальном перемещении и точном вертикальном позиционировании – они непревзойдённы по сочетанию скорости и точности. Промышленные SCARA широко

применяются в электронной промышленности (монтаж компонентов, перенос плат), в сборке приборов, упаковке, сортировке изделий, где требуются сотни операций в час. Их конструкции, как правило, основаны на мощных сервомоторах DC или AC с редукторами, обеспечивающими высокое ускорение, и замкнутых системах управления с ПИД-регуляторами на каждом суставе. Современные модели часто комплектуются встроенными контроллерами и поддержкой fieldbus- интерфейсов (EtherCAT, PROFINET) для интеграции в линию. Также отмечается тренд на коллаборативные SCARA – роботы с датчиками усилий и безопасными алгоритмами, которые могут работать бок о бок с людьми (например, серии duAro от Kawasaki, OMRON i4L Collaborative).

Рисунок 23 – Манипулятор FANUC

Самодельные и учебные SCARA: В сообществе мейкеров и образовательных учреждений популярны компактные 3D-печатные версии SCARA-манипуляторов, один из которых показан на рисунке 24. Их архитектура обычно упрощена: пластиковые рычаги, шаговые двигатели

вместо серводвигателей, простейшие редукторы (ременные передачи) либо отсутствие таковых.

Рисунок 24 – Простейший манипулятор

Управляются такие роботы обычно с помощью Arduino или другого микроконтроллера; обратной связи по положению может не быть (open-loop), либо используются дешёвые энкодеры. Пример – проект PyBot SCARA, полностью открытый и распечатываемый на 3D-принтере. PyBot имеет два поворотных сочленения (плечо и предплечье) на шаговых моторах NEMA17 и привод подъёма (ось Z) на ещё одном шаговом моторе. Все моторы управляются платой Arduino (SAMD21 микроконтроллер) с драйверами A4988, а высокоуровневое управление осуществляется с ПК через Python-приложение. Благодаря небольшому весу и моментам инерции такой робот не нуждается в датчиках для каждой оси – шаговые двигатели работают на

фиксированный шаг, а калибровка нуля выполняется путем проезда в упор (концевые выключатели или токовая отсечка). Конечно, точность и нагрузочная способность таких систем на порядки ниже промышленных образцов. У упомянутого PyBot заявленная повторяемость около 0,4 мм и груз до ~150 г. В прочем, авторам удалось программно улучшить точность до ~0,2 мм, реализовав калибровку и компенсацию ошибок модели. Основное применение DIY-SCARA – обучение и хобби: они способны выполнять простые манипуляции (рисование карандашом, сортировка лёгких предметов, лазерная гравировка по плоскости). Несмотря на ограниченные характеристики, такие проекты крайне важны для отработки алгоритмов и экспериментов. Например, в вузах на их базе изучают обратную кинематику, калибровку роботов, основы управления приводами. Кроме того, сообщество мейкеров делится улучшениями: добавляют камеры для зрительного наведения хватателя, модернизируют прошивки для достижения более плавного движения и т.п. – всё это по сути приближает любительские конструкции к возможностям более дорогих аналогов.

Коммерческие настольные решения: Между индустриальными и DIY-сегментом есть ниша доступных компактных роботов для обучения и лёгкой автоматизации. Сюда относятся, например, Dobot Magician – 4-осевой настольный робот, популярный в образовании. Хотя по кинематике он ближе к маленькому антропоморфному манипулятору (имеет базовый поворот и шарнирное сочленение плеча), его часто сравнивают с SCARA по функциям. Dobot Magician имеет рабочий радиус ~320 мм, несёт до 0,5 кг и обеспечивает повторяемость около ±0,2 мм. Он поставляется с контроллером и программным обеспечением для обучения программированию движений, может оснащаться вакуумным хватателем, мини-экструдером для 3D-печати, приспособлением для письма и пр. – тем самым демонстрируя многообразие задач, решаемых небольшими роботами. Подобные системы нацелены на STEM-обучение и автоматизацию лабораторных процессов. Их архитектура

обычно включает сервоприводы на основе шаговых моторов с редукцией, а управление реализовано через встроенный контроллер на базе MCU или ARM-платы. В отличие от DIY-самоделок, коммерческие мини-роботы проходят заводскую настройку и часто калибруются, поэтому обладают лучшей точностью и надёжностью, хотя и уступают «большим братьям».

Вывод по разделу 3

В данном разделе была выполнена разработка высокоуровневого программного обеспечения системы управления SCARA-манипулятором с применением среды моделирования RoboDK. Использование данной платформы позволило эффективно реализовать цифровой двойник манипулятора, смоделировать его кинематическую структуру и обеспечить точное планирование траекторий с учётом геометрических и технологических ограничений.

Была проведена настройка параметров симуляции, включая определение системы координат инструмента (TCP), ограничение рабочих зон и синхронизацию движения по заданным точкам. Высокоуровневая логика управления траекторией позволила автоматизировать процессы позиционирования и взаимодействия с внешними объектами. Также была реализована интеграция с низкоуровневой системой на базе Arduino Mega посредством обмена текстовыми командами по последовательному интерфейсу, что обеспечило согласованную работу виртуальной среды и реального оборудования.

# Заключение

В рамках дипломной работы разработана система управления промышленным манипулятором типа SCARA, охватывающая как аппаратную, так и программную части. Проект направлен на повышение точности, скорости и надежности выполнения технологических операций в автоматизированном производстве, что подтверждает его востребованность в современной инженерной практике.

В ходе исследования были проанализированы существующие решения в области систем управления роботами-манипуляторами, изучены конструктивные особенности роботов SCARA и требования к их функциональным характеристикам. Определена концепция проекта, которая легла в основу выбора компонентной базы и построения системы управления.

На этапе аппаратной реализации в качестве центрального устройства управления был выбран микроконтроллер Arduino Mega 2560 ввиду его высокой функциональности, расширенного количества портов ввода-вывода и широкого сообщества разработчиков. Для управления движением использованы бесколлекторные двигатели серии BLF, что обеспечило высокую динамику и устойчивость работы манипулятора. Система дополнена драйверами двигателей, индуктивными концевыми датчиками и источником питания, что позволило создать надежный и производительный аппаратный комплекс.

Разработанная печатная плата обеспечивает универсальность в управлении различными типами приводов, как бесколлекторными, так и шаговыми. Это позволило повысить гибкость системы и снизить затраты на ее модернизацию при изменении условий эксплуатации или задач оборудования. Также была спроектирована и собрана промышленная система управления, включающая электрощит со степенью защиты IP65,

обеспечивающий защиту от пыли и влаги, а также простоту обслуживания и эксплуатации.

Низкоуровневое программное обеспечение реализовано на языке C++ с использованием среды Arduino IDE. Программа организована по модульному принципу и позволяет обрабатывать команды, управлять сервоприводами, считывать данные с датчиков и обеспечивать обратную связь на верхний уровень управления. Архитектура программного обеспечения допускает масштабирование и расширение возможностей за счет реализации протоколов более высокого уровня и реализации логики обработки ошибок.

Высокоуровневое управление осуществлялось с помощью программной платформы RoboDK, которая обеспечивала моделирование движения манипулятора, визуализацию траекторий, генерацию управляющих программ и калибровку рабочего пространства. Интеграция с контроллером нижнего уровня реализована по последовательному интерфейсу UART с использованием преобразователя Ethernet Serial, что минимизирует влияние электромагнитных помех и обеспечивает удаленное управление оборудованием.

Таким образом, в ходе выполнения диссертационной работы была успешно решена задача создания комплексной системы управления манипулятором SCARA, объединяющей современные технические средства и программные решения. Разработанная система соответствует всем заданным техническим требованиям, обладает высокой степенью адаптивности и может быть внедрена в реальные производственные процессы для автоматизации сборочных, упаковочных и других операций.

Полученные результаты могут служить основой для дальнейших исследований и разработок в области робототехники и автоматизации промышленных процессов.

## Список используемой литературы


1. Акулич, И.Л. Методы оптимизации в теории управления / И.Л. Акулич. — М.: Высшая школа, 2018. — 374 с.

2. Башарин, А.В. Управление электроприводами / А.В. Башарин, В.М. Поляков. — СПб.: Энергоатомиздат, 2019. — 344 с.

3. Бесекерский, В.А. Теория систем автоматического регулирования / В.А. Бесекерский, Е.П. Попов. — М.: Наука, 2017. — 768 с.

4. Глазунов, В.А. Пространственные механизмы параллельной структуры / В.А. Глазунов. — М.: Наука, 2020. — 280 с.

5.Горячев, О.Н. Автоматизация технологических процессов и производств / О.Н. Горячев, В.Е. Курочкин. — М.: Академия, 2021. — 320 с.

6. Денисов, С.В. Шаговые двигатели: управление, контроль, применение / С.В. Денисов. — М.: Техносфера, 2019. — 256 с.

7. Дьяконов, В.П. STM32F103. Самоучитель программирования на языке С / В.П. Дьяконов. — М.: СОЛОН-Пресс, 2020. — 304 с.

8. Зотов, М.В. Промышленная электроника / М.В. Зотов. — М.: Academia, 2018. — 416 с.

9. Клюев, А.С. Проектирование систем автоматического управления / А.С. Клюев. — М.: Машиностроение, 2021. — 400 с.

10. Колесников, А.А. Современные методы проектирования систем автоматического управления / А.А. Колесников. — М.: Машиностроение, 2020. — 432 с.

11. Кулешов, В.С. Программируемые логические контроллеры / В.С. Кулешов. — СПб.: BHV, 2020. — 272 с.

12. Миронов, А.Г. Микроконтроллеры STM32: архитектура, программирование, разработка устройств / А.Г. Миронов. — М.: ДМК Пресс, 2021. — 480 с.

13. Петров, Ю.П. Компьютерное управление в технике / Ю.П. Петров. — СПб.: БХВ-Петербург, 2018. — 368 с.

14. Смирнов, С.А. Промышленные роботы и манипуляторы / С.А. Смирнов. — М.: Инфра-М, 2020. — 352 с.

15. Филиппов, А.Ю. Робототехнические системы: основы построения и проектирования / А.Ю. Филиппов. — М.: НИЦ ИНФРА-М, 2021. — 400 с.

16. Хабаров, А.В. Электромеханические системы автоматизации / А.В. Хабаров. — М.: Академия, 2019. — 336 с.

17. Чистяков, В.С. Основы теории автоматического управления / В.С. Чистяков. — М.: Высшая школа, 2020. — 416 с.

18. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems / R.E. Kalman // Journal of Basic Engineering. – 1960. – Vol. 82. – P. 35–45.

19. Ogata, K. Modern Control Engineering / K. Ogata. – 5th ed. – Upper Saddle River: Prentice Hall, 2010. – 928 p.

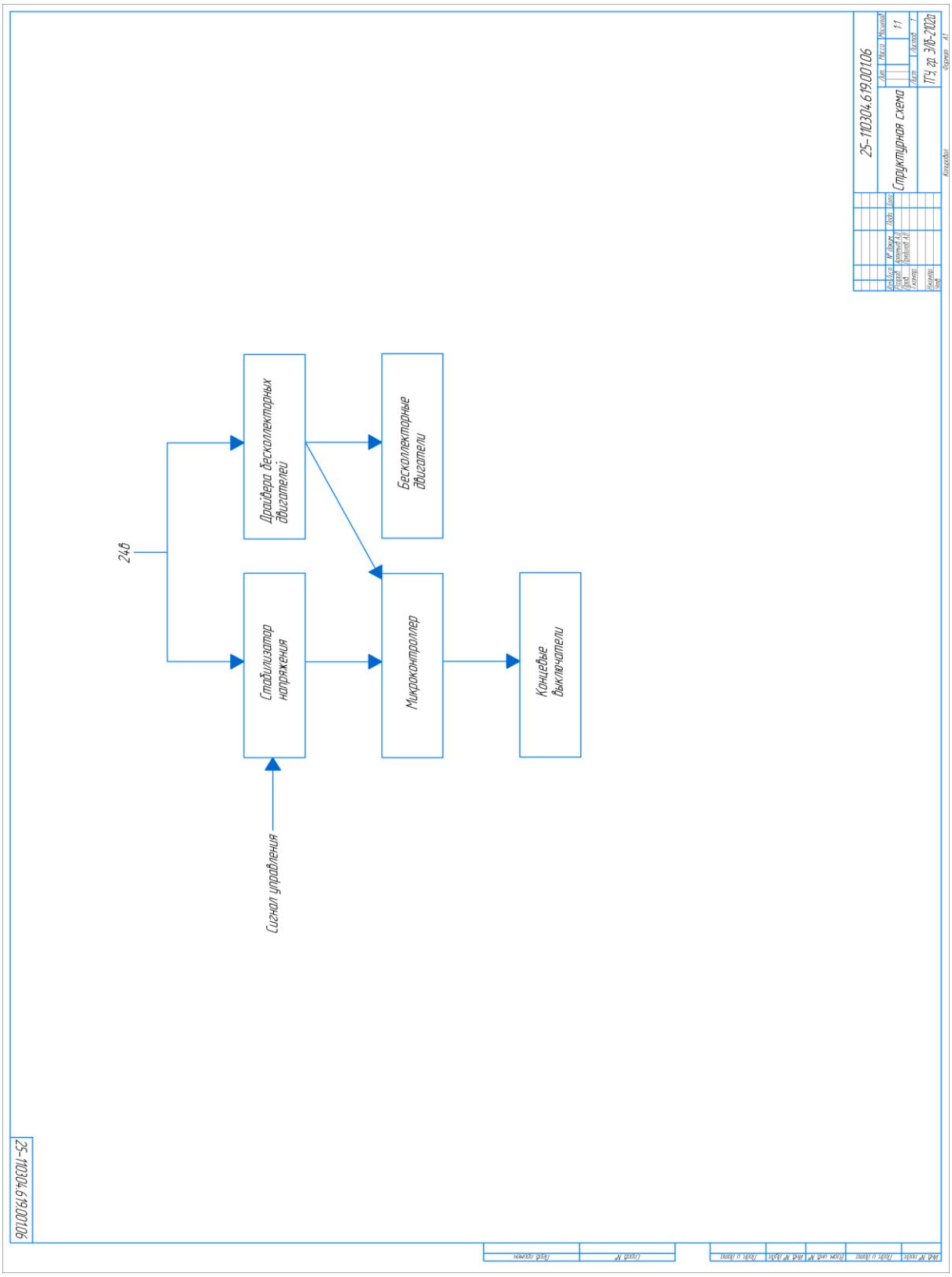
20. Nof, S.Y. Springer Handbook of Automation / S.Y. Nof. – Berlin: Springer, 2017. – 1580 p.

21. Siciliano, B. Robotics: Modelling, Planning and Control / B. Siciliano et al. – London: Springer, 2010. – 632 p.

22. Groover, M.P. Fundamentals of Modern Manufacturing: Materials, Processes, and Systems / M.P. Groover. – Hoboken: Wiley, 2019. – 1152 p

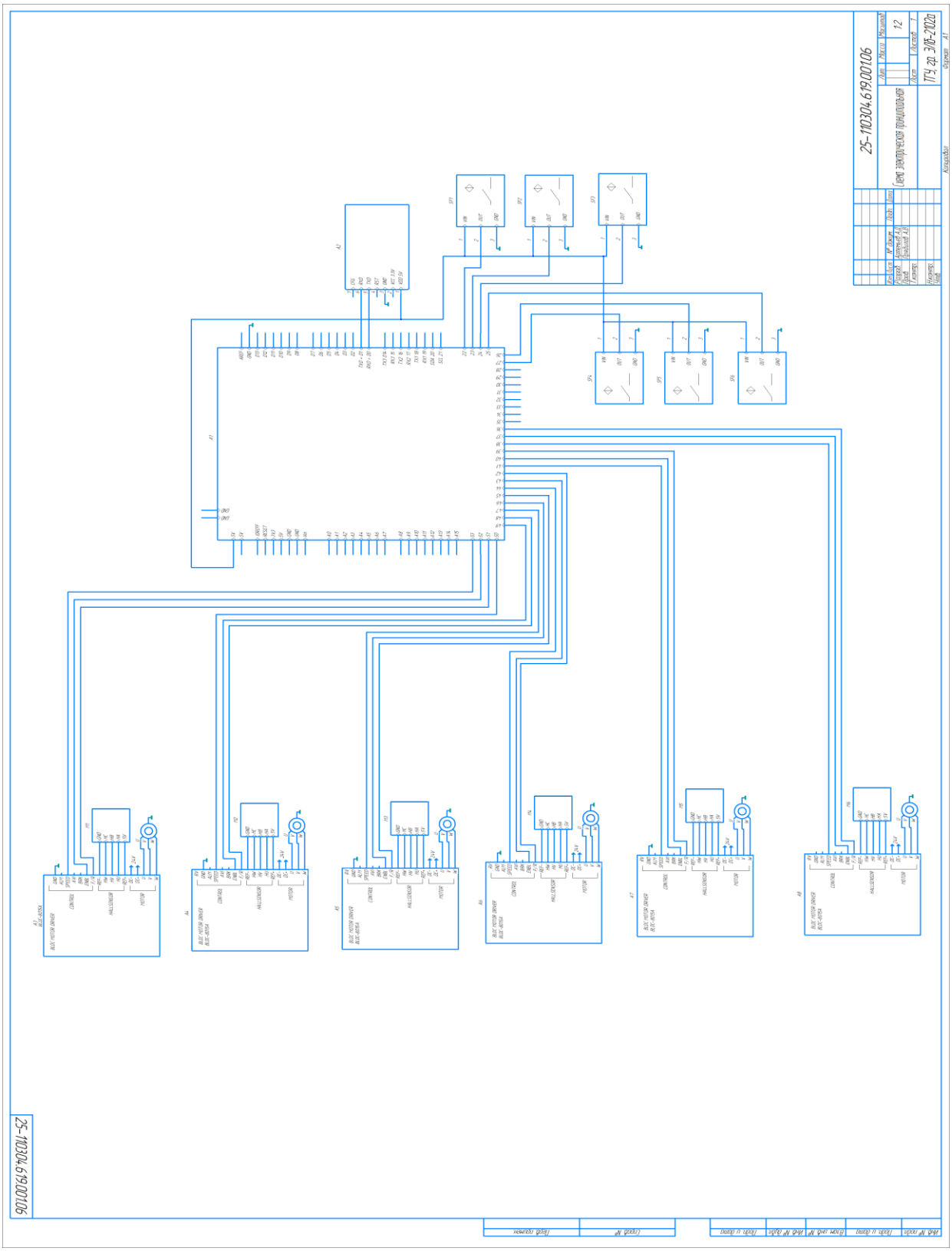
Приложение А

**Структурная схема**



Приложение Б

**Принципиальная схема электронного блока управления**



Приложение В

**Перечень элементов**

| Поз. обозначение | Наименование | Кол. | Примечание |
|---|---|---|---|
| A1 | Микроконтроллер Arduino AMEGA 2560 | 1 | |
| A2 | Ethernet-модуль USR-TCP232-T2 | 1 | |
| A3-A8 | Драйвер BLDC-4008 | 6 | |
| SF | Индуктивный датчик LJ12A3-4-Z/BX NPN | 6 | |
| M | Бесколлекторный двигатель PL57BLM03 | 6 | |

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

| Изм. | Лист | № докум. | Подп. | Дата |
|---|---|---|---|---|
| Разраб. | | Артемьев А.Д. | | |
| Пров. | | Прядилов А.В. | | |
| Н.контр. | | | | |
| Утв. | | | | |

25-110304.619.001.06

Перечень элементов

| Лит. | Лист | Листов |
|---|---|---|
| Д | 1 | 1 |

ТГУ, гр. ЭЛб-2102а

Копировал

Формат А4