

СОДЕРЖАНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ	7
ВВЕДЕНИЕ	8
1 ОПРЕДЕЛЕНИЕ ФУНКЦИЙ СИСТЕМЫ И ТЕХНИЧЕСКИХ ТРЕБОВАНИЙ	10
1.1. Мини роботы	10
1.2. Особенности коллаборативных роботов манипуляторов	11
1.3. Проблема существующих мини роботов манипуляторов	14
1.4. Требования к управлению электродвигателя	16
1.5. Определение функций и технических характеристик системы управления для мини роботом	17
2 СТРУКТУРНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА	20
2.1. Особенность иерархической структуры	20
2.2. Структурная схема системы управления мини робота	21
2.3. Структурная схема системы исполнительного управления	23
2.4. Алгоритм работы устройства тактического управления	25
2.5. Алгоритм работы устройства исполнительного управления	28
3 ФУНКЦИОНАЛЬНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА	31
3.1. Gimbal двигатель	31
3.2. Энкодер	32
3.3. Устройство стратегического управления	34
3.4. Контроллер тактического и исполнительного управления	36
3.5. Тестирование микроконтроллеров	36
3.6. Силовые ключи	42
3.7. Драйвер силовых ключей	44
3.8. Передатчик шины данных	45
3.9. Преобразователь питания	46
3.10. Функциональная схема системы управления мини роботом	47
3.11. Разработка функциональной схемы системы исполнительного устройства	51
4 ПРИНЦИПИАЛЬНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА	54
4.1. Принципиальная схема устройства исполнительного управления	54

4.2. Принципиальная схема устройства тактического управления	58
5 АЛГОРИТМ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ КОЛЛАБОРАТИВНЫМ МИНИРОБОТОМ МАНИПУЛЯТОРОМ	62
5.1. Программная реализация устройства стратегического управления	62
5.2. Алгоритмы устройства исполнительного управления	64
5.3. Программная реализация устройства исполнительного управления	71
5.4. Программная реализация устройства тактического управления	72
ЗАКЛЮЧЕНИЕ	77
СПИСОК ЛИТЕРАТУРНЫХ И ДРУГИХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ	80
ПРИЛОЖЕНИЯ	82

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

ACD – Устройство исполнительного управления

ADC – Аналого-цифровой преобразователь

BLDC – Brushless Direct Current Motor

BLDC – Бесколлекторный двигатель постоянного тока

CAN – Сеть контроллеров

DC – Постоянный ток

DMA – Прямой доступ к памяти

GPIO – Универсальный ввод/вывод

HAL – Аппаратный абстрактный уровень

I2C – Inter-Integrated Circuit

LED – Светодиод

MCU – Микроконтроллер

MOSFET – Полевой транзистор с металл-оксидным затвором

PID – Пропорционально-интегрально-дифференциальный регулятор

PLL – Фазовая автоподстройка частоты

PWM – Pulse-Width Modulation

PWM – Широтно-импульсная модуляция

SCD – Устройство стратегического управления

SPI – Последовательный периферийный интерфейс

SSH – Защищенная оболочка

SVPWM – Пространственно-векторная широтно-импульсная модуляция

TCD – Устройство тактического управления

UART – Универсальный асинхронный приемник/передатчик

UVLO – Защита от пониженного напряжения

kOPS – Тысяча операций в секунду

ВВП – Валовой внутренний продукт

ВВЕДЕНИЕ

Активное распространение роботов после середины 20 века привело к автоматизации почти всех аспектов производственных процессов, включая сборку, сварку, покраску, контроль качества и упаковку. Это также оказало значительное влияние на логистику, складское хозяйство и дистрибуцию, где роботизированные системы используются для сортировки, перемещения и хранения товаров. Прогресс в области механики, электроники и цифровых устройств, привел к созданию инновационных устройств. Использование существующих универсальных мини роботов манипуляторов является эффективным и рациональным решением, ввиду их небольшого размера и параметров соответствующих для работы с мелкими деталями и конструкциями. Однако несмотря на преимущества мини роботов манипуляторов, они обладают недостатками. Главным недостатком является опасность работы роботов совместно с людьми. Для организации работы данных роботов необходимо создавать специальные закрытые рабочие зоны, для минимизации возможного риска, а работа совместно с человеком представляется невозможной. Мини робот не может стать коллаборативным если ограничить мощность или силу для каждого звена. Для того что бы робот являлся коллаборативным необходимо производить изменения в конструкции самого робота. Размер мини роботов манипуляторов создает ограничения для всех элементов системы, установка датчиков момента силы не является лучшим решением, так как занимают дополнительное пространство в каждом звене робота и усложняют конструкцию мини робота.

Наиболее подходящий тип электродвигателя является бесщеточный двигатель постоянного тока, для обеспечения всеми особенностями коллаборативного робота, тип BLDC двигателя Gimbal motors наиболее подходящим.

Целью работы является: Демонстрация возможности использования двигателя типа от карданных камер (gimbal motors) для создания системы управления коллаборативного мини робота манипулятора с учетом упрощения их механической конструкции.

Работа содержит 5 разделов. В первом разделе производится анализ мини роботов манипуляторов, которые доступны на рынке и их характеристики. Исследуются различные варианты двигателей и производится выбор, разработка функций и технических параметров для системы управления мини роботом. Во втором разделе производится разработка структурной схемы на основе иерархической парадигмы управления роботом, а также алгоритма её описания для системы управления мини роботом. В третьем разделе производится разработка функциональной схемы и описание и назначения ее элементов. В четвертом разделе производится разработка принципиальной схемы с ее описанием, расчет элементов схемы. В пятой части производится

разработка алгоритмов и программная реализация на микроконтроллере.

1 ОПРЕДЕЛЕНИЕ ФУНКЦИЙ СИСТЕМЫ И ТЕХНИЧЕСКИХ ТРЕБОВАНИЙ

1.1. Мини роботы

Активное распространение роботов после середины 20 века привело к автоматизации почти всех аспектов производственных процессов, включая сборку, сварку, покраску, контроль качества и упаковку. Это также оказало значительное влияние на логистику, складское хозяйство и дистрибуцию, где роботизированные системы используются для сортировки, перемещения и хранения товаров. Прогресс в области механики, электроники и цифровых устройств, привел к созданию инновационных устройств. Это в свою очередь способствовало прогрессу, повышению качества и возможностей ранее разработанных решений. Одним из улучшений недавних разработок можно считать появление мини роботов манипуляторов. Хотя стоит отметить, что большинство работ связанных со сборкой деталей небольших размеров до сих пор выполняются людьми. Использование существующих универсальных мини роботов манипуляторов является эффективным и рациональным решением, ввиду их небольшого размера и параметров соответствующих для работы с мелкими деталями и конструкциями.

Мини роботы манипуляторы представлены на рынке и разработки ведутся в данном направлении (Li et al., 2022), примером таких роботов могут считаться роботы моделей «MotoMini» от компании «Yaskawa», «Meca500» (Рисунок 1.1) от компании «Mecademic» и другие. Тема мини роботов является актуальной, однако стоит отметить, количество информации, на данный момент, о роботах этого класса крайне небольшое.



Рис. 1.1. Meca500 Robot Arm

Благодаря маленькому размеру всей конструкции, возможно создавать целые производ-

ственные линии прямо на столе. Это открывает новые возможности для мелкосерийного производства и разработки прототипов, где требуется высокая точность и гибкость процессов. Однако несмотря на преимущества мини-роботов манипуляторов, они обладают недостатками. Главным недостатком является опасность работы роботов совместно с людьми. Для организации работы данных роботов необходимо создавать специальные закрытые рабочие зоны, для минимизации возможного риска, а работа совместно с человеком представляется невозможной. Эта проблема ограничивает область применения мини-роботов, делая их непригодными для задач, где требуется тесное взаимодействие между человеком и машиной особенно при работе с мелкими предметами, требующие участия оператора. Также это снижает гибкость производственных систем, поскольку для изменения конфигурации линии может потребоваться значительное время и ресурсы на переустройство рабочей зоны. Отсутствие коллаборативности является основной проблемой данных типов роботов.

1.2. Особенности коллаборативных роботов манипуляторов

Коллаборативные роботы или коботы - это индустриальный тип робота, оборудованный набором датчиков и фиксирующих устройств, которые помогают с высокой точностью избегать столкновений с людьми и различными препятствиями, даже в случаях неисправности программного обеспечения. Эти устройства разработаны для работы в непосредственной близости с человеком, не требуя размещения в изолированных рабочих зонах. Они так же отличаются простотой программирования, что делает их доступными для использования персоналом без специализированных навыков.

Согласно международному стандарту ISO 10218 (части 1,2) предусматривает четыре основных типа коллаборативных роботов (Bélanger-Barrette, 2015), которые могут использоваться для совместной работы с человеком.

- С остановочным защитным механизмом. Такие коллаборативные роботы способны работать автономно по введенной управляющей программы, но при появлении человека датчики останавливают его работу. Работа робота продолжается, когда человек покидает рабочую зону;
- С ручным управлением. В присутствии человека такой коллаборативный робот выполняет его команды, подаваемые рукой. Например, команда может подаваться путем касания рукой до определенной части корпуса или распознает движение рук на расстоянии. При отсутствии человека коллаборативный робот может работать автономно, если введена необходимая управляющая программа;
- С «компьютерным зрением». Специальные датчики отслеживают движение людей. При

входе человека в рабочую зону устанавливается пониженная, безопасная скорость. Если он приближается слишком близко, то коллаборативный робот останавливает работу;

- С ограниченной силой. Эти короботы при перемещении не сметают все по пути. Если они наталкиваются на препятствие, то останавливаются, в т. ч. при столкновении с человеком. Такие коллаборативные роботы могут использоваться в непосредственной близости от людей.

ISO/TS 15066:2016 представляет собой первый в своем роде документ, устанавливающий критерии безопасности для использования коллаборативных роботов. Разработка этого технического стандарта была осуществлена международным комитетом ISO, в который входили специалисты из 24 стран и представители ведущих производителей робототехники, начиная с 2010 года. Стандарт служит дополнением к уже существующим нормам безопасности промышленных роботов, зафиксированным в ISO 10218-1 и ISO 10218-2, и устанавливает конкретные требования к безопасному взаимодействию между промышленными роботами и людьми в рабочей среде. ISO/TS 15066 предлагает методические рекомендации для проведения оценки рисков, связанных с коллаборативным использованием роботов и человека.

Контролируемая остановка (Safety-rated monitored stop), этот подход применяется, когда робот в основном работает автономно, однако иногда требуется вмешательство человека в рабочую зону. К примеру, робот может выполнять обработку детали, но в определенный момент процесса необходимо человеческое вмешательство для выполнения задачи, недоступной роботу. В случае входа человека в установленную зону безопасности, робот останавливается, при этом моторы остаются под напряжением и готовы к возобновлению работы. Это критично, так как после выхода человека из зоны безопасности, робот мгновенно продолжает свою деятельность, исключая потребность в полной перезагрузке программы, что могло бы произойти при полной остановке. Такой метод исключает простои в работе робота, которые бы возникали, если бы люди регулярно перемещались через его рабочую зону;

- Остановка обеспечивается без потери мощности двигателей (пауза);
- Оператор может взаимодействовать с роботом;
- Автоматическая работа может возобновиться, когда человек покидает рабочее пространство;
- В один момент времени может двигаться либо человек, либо робот;
- Может использоваться с обычными промышленными роботами, но нужно добавить световые барьеры безопасности (лазерные дальномеры, фотодетекторы).

Ручное ведение (Hand guiding); Этот метод коллaborации применяется при выполнении сложных манипуляций с тяжелыми предметами и может быть адаптирован для использования

с традиционными промышленными роботами при добавлении специального устройства. Это устройство, обычно датчик момента, установленный на фланце робота, позволяет ощущать силу, прикладываемую оператором к манипулятору, обеспечивая таким образом возможность более точного управления действиями робота.

- Оператор находится в непосредственном контакте с роботом.
- Робот находится под ручным управлением.
- И человек, и робот могут двигаться одновременно (движения контролируются человеком).
- Могут использоваться обычные промышленные роботы.
- Требуется дополнительное оборудование (датчик момента)

Контроль скорости и разделения (Speed and separation monitoring); В этом подходе к совместной работе рабочее пространство робота ограничено световыми барьерами безопасности, аналогично первому типу коллaborации, которые отслеживают расположение людей вокруг. Основное отличие состоит в цели: в то время как в первом случае основной задачей робота является остановка при приближении человека, в данном сценарии целью является возможность параллельной работы человека и робота. Робот адаптирует свое поведение в соответствии с предварительно заданными зонами в его программе: с уменьшением дистанции до человека робот постепенно снижает скорость своих движений и останавливается при критическом сближении для предотвращения столкновения. Когда человек удаляется, робот возобновляет работу и увеличивает скорость.

- Скорость робота уменьшается, по мере приближения человека.
- Робот останавливается, когда может произойти столкновение с человеком.
- Человек и робот могут перемещаться одновременно.
- Может использоваться с обычными промышленными роботами, но нужно добавить световые барьеры безопасности (лазерные дальномеры, фотодетекторы).
- Используется для операций, требующих частого присутствия персонала

Все описанные выше формы совместной работы могут использоваться с обычными промышленными роботами, при условии наличия дополнительных устройств. Такие решения называются: коллаборативные робототехнические системы. Отличие коллаборативного робота в том, что он может не использовать дополнительные (внешние) устройства обеспечения безопасности – они уже встроены в него. Еще одно важное отличие коллаборативных робототехнических систем от коллаборативных роботов состоит в том, что при совместной работе исключается контакт с человеком. А вот коллаборативный робот может контактировать с телом человека без вреда для него. Это достигается за счет ограничения мощности и усилия.

Ограничение мощности и усилия (Power and force limiting). В «системах» робота находятся датчики момента, которые могут определить факт столкновения с человеком. Если датчики обнаруживают превышение допустимых усилий, робот останавливается. Эти роботы также предназначены для рассеивания сил на широкой поверхности, в случае удара – именно поэтому детали их корпуса чаще всего сделаны с округлыми формами. Функционал по ограничению мощности и силы, как правило, входит в штатное ПО.

1.3. Проблема существующих мини роботов манипуляторов

Мини робот не может стать коллаборативным если ограничить мощность или силу для каждого звена. Для того что бы робот являлся коллаборативным необходимо производить изменения в конструкции самого робота. Размер мини роботов манипуляторов создает ограничения для всех элементов системы, установка датчиков момента силы не является лучшим решением, так как занимают дополнительное пространство в каждом звене робота и усложняют конструкцию мини робота. Требуется рассмотреть другие части робота, которые позволяют обеспечить особенности коллаборативного робота. Основная часть робота которая обеспечивает передвижения звеньев, является приводы. Приводные механизмы связаны с кинематическими элементами манипулятора и выполняют перемещения. Привод любого робота манипулятора состоит из:

- Преобразователя энергии
- Электродвигателя
- Передаточного механизма

Электродвигатель — электротехническое изделие, основной функцией которого является преобразование энергии электрической в механическую. Для обеспечения функций коллаборативного робота, необходимо выбрать подходящий тип электродвигателя.

Наиболее часто используемые двигатели, которые используются в роботах манипуляторах двигатели: постоянного тока, шаговые и сервоприводы.

- Двигатели постоянного тока бывают коллекторные и бесщеточные. Принято считать, что бесщеточные двигатели постоянного тока вытеснили коллекторные. Бесщеточный двигатель постоянного тока тише, эффективен на низких оборотах и может поддерживать постоянный максимальный крутящий момент, но данный тип двигателя сложен в управлении.
- Шаговые двигатели могут контролировать точное движение, имеют максимальный крутящий момент на низких скоростях и просты в управлении. Однако у них есть несколько недостатков: они шумны и относительно неэффективны, а также перегреваются, посколь-

ку постоянно потребляют максимальный ток, пропускают ступени при высоких нагрузках и за "шаговых" перемещений создают дополнительную вибрацию на низких скоростях вращения.

- Серводвигатели обеспечивают чрезвычайно точное движение. К недостаткам использования серводвигателей относятся возможность их джиттера при реакции на обратную связь и необходимость сложной логики управления, таким образом занимают много места и добавляют сложности в конструкцию робота. Так же для серводвигателей предполагается установка редуктора с большой степенью передаточного отношения, что так же влияет на габариты и сложности самого робота.

Наиболее подходящим типом электродвигателя для робота манипулятора является бесщеточный двигатель постоянного тока как видно из таблицы 1.1, за счет большого момента на низких оборотах, двигатели Gimbal, которые являются бесщеточные двигатели постоянного тока, ориентированы под использование корданных подвесов камер, у данных подтипа двигателей постоянного тока очень высокий момент при низких оборотах. Данная особенность облегчает выбор редуктора, не требуется высокая степень редукции, это упрощает конструкцию самого редуктора и делает его более компактным. Небольшая степень передаточного отношения создаёт возможность обратного управления приводом (Matsuki et al., 2019).

Таблица 1.1

Таблица характеристик двигателей различных типов

Name	Type	Power (W)	Torque (Nm)	Weight (g)	RPM	Reduction Ratio to 60 RPM	Torque at 60 RPM (Nm)
GM8112	Gimbal	80	0,89	428	420	7	2,45
GM110-8T	Gimbal	50	0.35	249	209	4	1,4
EC 45 Flat	BLDC	50	0.09	110	2360	39	3,53
EC 45 Flat	BLDC	70	0.134	140	2540	42	5,67
Tarot 4008 330kv	BLDC	460	0.63	80	7920	132	83,16
Nema 8 x 38mm	Step	-	0.03	80	1200	20	0,6
Nema 17 x 34mm	Step	-	0.26	230	1200	20	5,2
Nema 23 x 76	Step	-	1.9	1200	1200	20	38
TSM4102	Servo	50	0.159	400	3000	50	7,92

Так же исходя из тенденций последних исследований (Sakama et al., 2022) рис 1.2, удельная мощность бесщеточных двигателей постоянного тока, с 1990-х годов, увеличилась более чем в десять раз за 20 лет, и теперь бесщеточные двигатели постоянного тока являются электродвигателями с самой высокой удельной мощностью. Удельная мощность данных двигателей увеличилась после появления постоянных магнитов с высокой максимальной энергией. В частности, появление неодимового магнита.

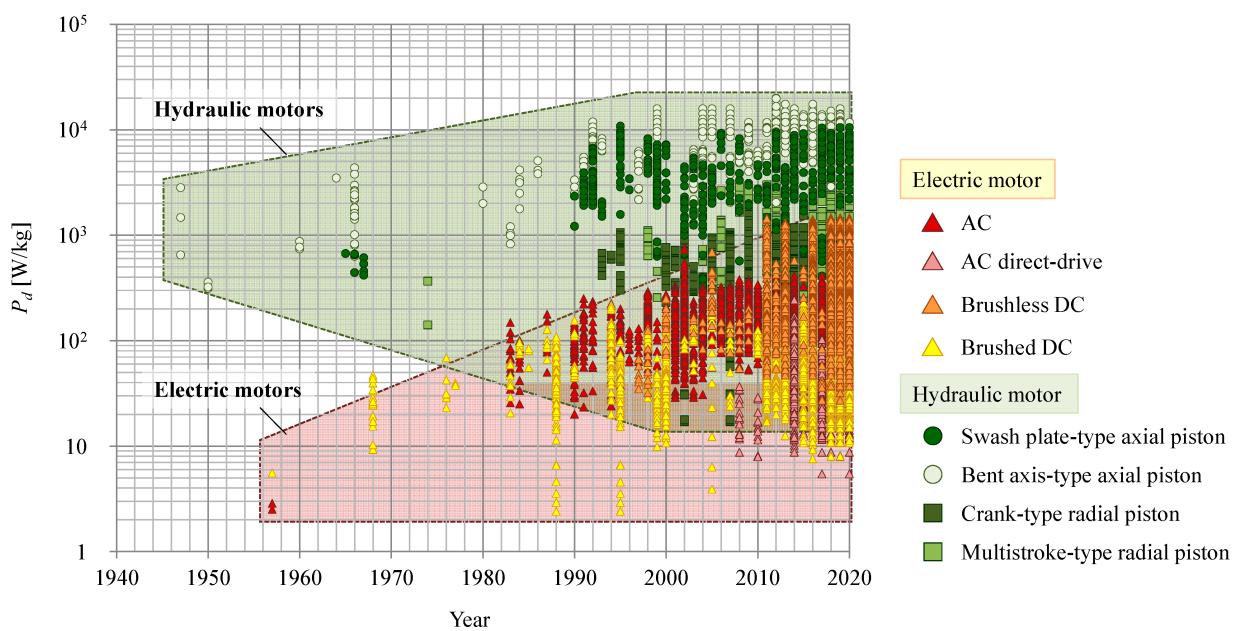


Рис. 1.2. Измерение удельной мощности в электрических двигателях во времени (Sakama et al., 2022)

1.4. Требования к управлению электродвигателя

Наиболее подходящий тип электродвигателя исходя из раздела 1.3 является бесщеточный двигатель постоянного тока, для обеспечения всеми особенностями коллаборативного робота. Касательно мини робота, тип BLDC двигателя Gimbal motors наиболее подходящим. Но Gimbal motors обладает недостатками, которые присущи всем BLDC двигателям - сложность в управлении. Варианты управления бесщёточными двигателями постоянного тока:

- Управление с шестистепенной коммутацией;
- Широтно-импульсная модуляция (PWM);
- Управление с векторной модуляцией (FOC);
- Датчиковое и бездатчиковое управление.

При выборе метода управления требовалось учитывать необходимость управления скоростью, ускорением и моментом двигателя. Векторное управление является наиболее подходящим методом для управления электродвигателем осью мини робота манипулятора. FOC обеспечивает высокую эффективность и плавность работы за счет независимого управления магнитным потоком и крутящим моментом двигателя. Достигается за счет преобразования трехфазного сигнала в двухфазный во вращающейся системе координат, который используется для дальнейшего управления. Для векторного регулирования необходимо определять текущее положение ротора двигателя с использованием датчиков угла поворота.

При разработке системы управления для коллаборативного мини робота, необходимо учи-

тывать не только различия управлении приводов, но саму систему управления в целом. Поэтому необходимо производить изменение всего управления робота.

1.5. Определение функций и технических характеристик системы управления для мини роботом

Как уже упоминалось ранее любой робот манипулятор состоит из различных систем: электронной, механической, вычислительной, сенсорной и других. Система управления робота должна организовывать работу между ними и обеспечить функционирование робота манипулятора по следующим требованиям:

- Точное управление движением, плавное управление движениями робота с большой точностью, включая синхронизацию всех суставов и механизмов;
- Реализация заданных алгоритмов и задач, включает в себя задачи последовательности движений и операций действий робота;
- Обработка сенсорных данных, точно обрабатывать данные с датчиков для адаптации к изменениям окружающей среды;
- Взаимодействие с оператором, возможности для ручного управления и программирования;
- Интеграция с другими системами, робот являясь не только одним элементом на производстве, использование совместно с системами конвейерных линий.

Система управления коллаборативного мини робота не сильно отличается от требований обычного промышленного манипулятора, но должна обеспечивать функциям:

- Расчёта траектории движения, решение кинематических задач;
- Точное и плавное управление движением двигателем используя алгоритм векторного регулирования с обратной связью;
- Управление скоростью, ускорением, моментом электродвигателя;
- Ограничение минимальных и максимальных параметров скорости, ускорения и позиции, тока напряжения;
- Управление моментом;
- Обработка данных с различных датчиков в реальном времени;
- Визуальное отображение состояния робота;
- Возможность интеграции с другими системами;
- Мониторинг температуры и отключение устройства при высоких температурах.

Разрабатывать систему управления для мини робота манипулятора невозможно в отрыве от конструкции самого робота. На основе данных, выполненного в разделе 1.1. Было опреде-

лённо, что система управления коллаборативного мини роботом манипулятора с использованием Gimbal motors, должна иметь технические параметры, соответствующие характеристикам существующих мини роботов. Конструкция робота - 6 осевой робот манипулятор: шесть степеней свободы необходимо для приближенной имитации возможных движений человеческой руки. Для разработки системы управления мини роботом манипулятором взята наиболее часто встречающаяся кинематическая структура, приведена на рисунке 1.3.

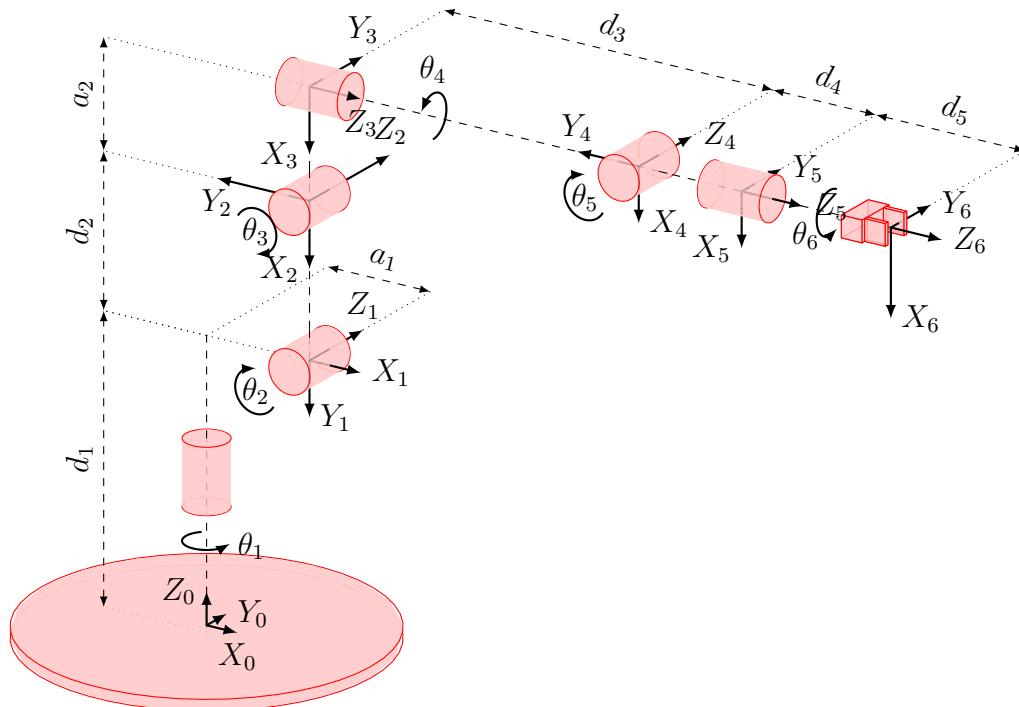


Рис. 1.3. Кинематическая структура мини робота

Параметры кинематической структуры:

- d_1, d_2, d_3, d_4, d_5 - расстояния между звеньями;
- $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ - углы звеньев;
- a_1, a_2 - длины звеньев;
- X_0, Y_0, Z_0 - базовые координаты;
- X_6, Y_6, Z_6 - координаты инструмента.

За основу реализации системы управления были взяты существующие параметры от мини-робота манипулятора MotoMini от компании "Yaskawa":

$$d1 = 171mm; d2 = 165mm, a1 = 20mm, a3 = 0mm; d3 = 165mm, d4 = 40mm, d5 = 0mm$$

Технические характеристики:

- Количество осей – 6 осей;

- Повторяймость 0.02 мм;
- Ограничение максимальной рабочей температуры: 80 °C;
- Ограничение максимального тока двигателя до 25 A;
- Размер системы управления не должна превышать размера 300x300mm;
- Масса системы управления, не должна превышать 5 кг;
- Минимальная скорость поворота оси не менее 150 °/s;
- Средняя мощность потребления <200W;
- Рабочее напряжение 24V.
- Ограничения углов:
 - Ось 1 от -180° до +180°;
 - Ось 2 от -90° до +90°;
 - Ось 3 от -90° до +90°;
 - Ось 4 от -180° до +180°;
 - Ось 5 от -180° до +180°;
 - Ось 6 от -360° до +360°;

2 СТРУКТУРНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА

2.1. Особенность иерархической структуры

Система управления для коллаборативного мини робота манипулятора включают другие системы, как управления двигателями, внутренних и внешних периферий и другие, которые зависят между собой. Поэтому для разработки структурной схемы системы управления коллаборативного мини робота манипулятора была использована иерархическая парадигма архитектуры систем управления роботами. Согласно этой парадигме, иерархическая архитектура системы управления роботом (рис. 2.1), в общем случае, содержит четыре уровня управления: интеллектуальный уровень, стратегический уровень, тактический уровень и исполнительный уровень(Khatib et al., 1997).

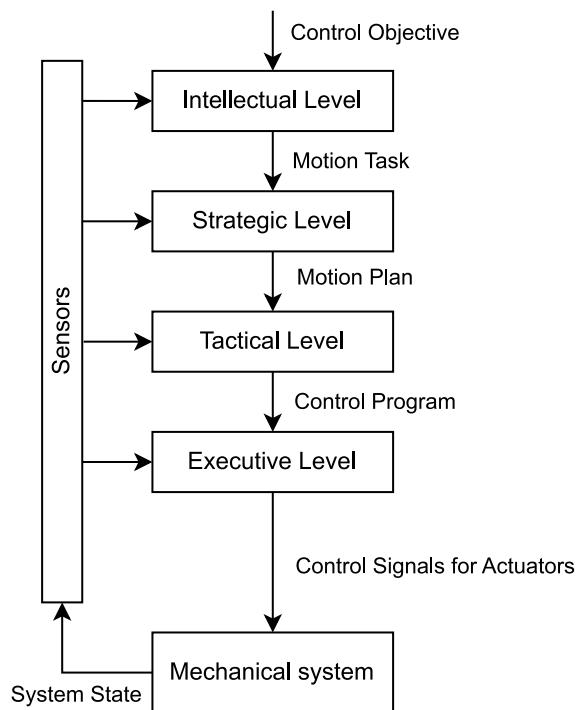


Рис. 2.1. Иерархическая архитектура системы управления роботом

В иерархической архитектуре системы управления роботом манипулятором уровни управления обеспечивают выполнение специфических задач управления роботом:

- На интеллектуальном уровнерабатываются (формируются) решения о выполнении действий роботом манипулятором в условиях неполной информации о внешней среде и объектах, на которые робот воздействует (например, перемещение объекта из одной точки пространства в другую, обработка детали и т.д.).
- Стратегический уровень управления предназначен для планирования движений робота

манипулятора: разделение задачи действия, выработанной на интеллектуальном уровне, на последовательность согласованных во времени элементарных действий (движений) и формализацию целей управления для каждого из этих действий. Стратегический уровень выдает тактическому уровню план движения в форме команд управления движением. Например, вывод рабочего органа (захватывающего устройства) робота в заданную позицию, захват объекта, тестовое движение для определения сил реакции со стороны объекта, перемещение объекта и возвращение робота в исходную позицию.

- На тактическом уровне выполняется преобразование команд управления движением, поступающих со стратегического уровня управления, в программу управления, которая определяет законы согласованного движения во времени всех звеньев робота с учетом технических характеристик блока приводов (в первую очередь ограничений на обобщенные скорости, ускорения и силы). Для выполнения команды позиционного управления движением робота манипулятора на тактическом уровне необходимо определить обобщенные координаты манипулятора, которые соответствуют желаемым декартовым координатам характеристической точки захватывающего устройства робота манипулятора. Для этого решается обратная кинематическая задача о положении манипулятора в заданной точке траектории движения.
- Исполнительный уровень управления предназначен для расчета и выдачи управляющих сигналов на блок приводов робота манипулятора в соответствии с программой управления, созданной на тактическом уровне, и с учетом технических характеристик исполнительных устройств.

2.2. Структурная схема системы управления мини робота

Рисунок 2.2 иллюстрирует структурную схему системы управления мини роботом. Система управления для мини робота манипулятора имеет иерархическую структуру, которая соответствует иерархической архитектуре системы управления роботом (рис. 2.1), так как в разработанной структурной схеме есть устройства, соответствующие функциям уровней управления на (рис.2.1):

1. Устройство стратегического управления (SCD)
2. Устройство тактического управления (TCD)
3. Устройства исполнительного управления (ACD)

Устройство тактического управления (TCD) мини роботом, координирует работу всех устройств системы управления мини роботом, обеспечивая эффективное функционирование всего механизма робота. Оно выполняет обработку управляющих команд с устройства стра-

тического управления (SCD) и отвечает за формирование выходных сигналов на устройства исполнительного управления (ACD), обеспечивая соответствие поведения робота с заданными командами.

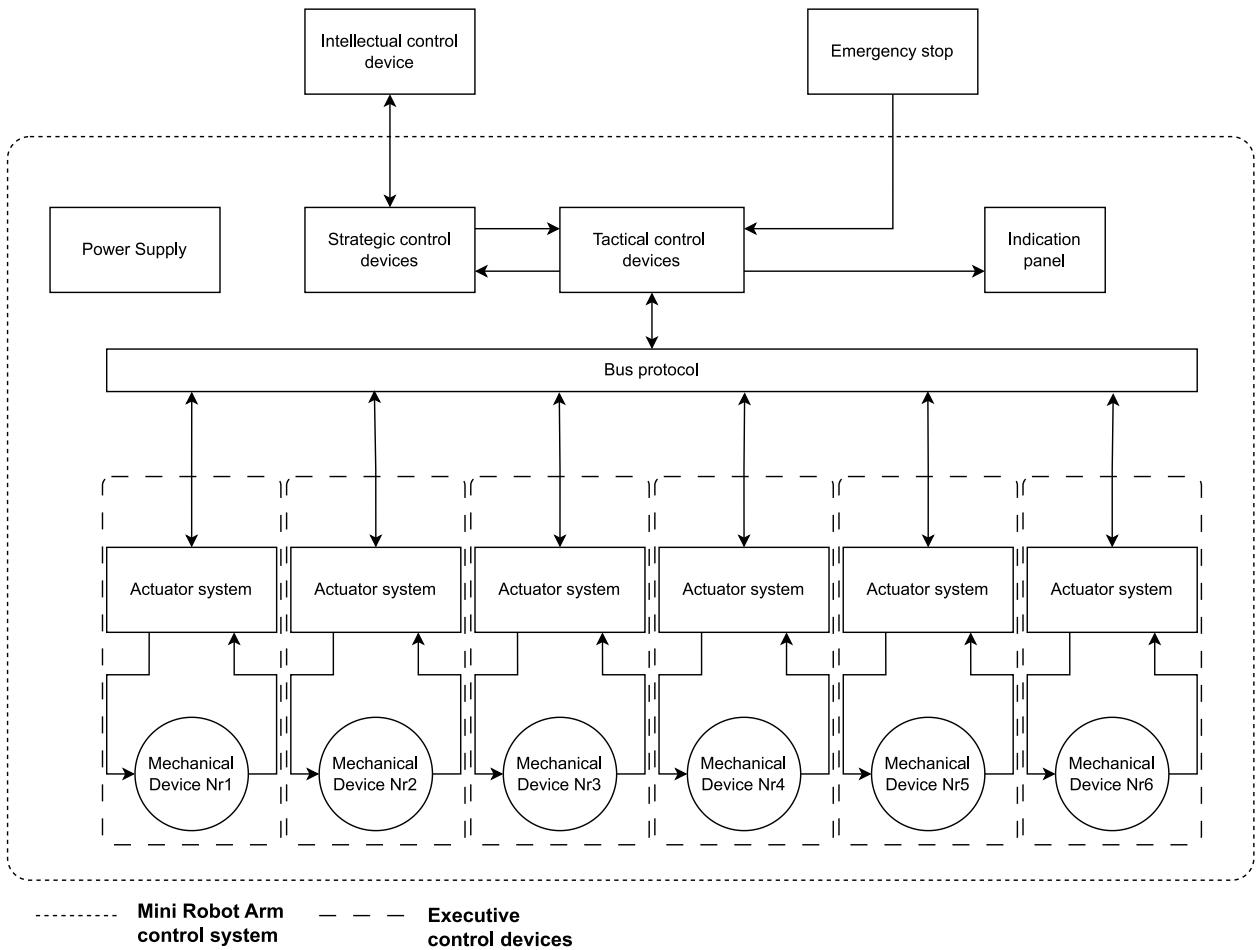


Рис. 2.2. Структурная схема системы управления мини робота

Устройство исполнительного управления (ACD) — это устройство, главная цель которого формирование сигналов для точного управления отдельного механического устройства (электродвигателем), обеспечивая поворот плеча робота манипулятора. Данное устройство позволяет выполнять управление двигателем без использования устройства тактического управления (TCD). Благодаря этому можно производить сложные манипуляции с высокой степенью точности и осуществлять поворот осью двигателя на требуемый угол.

После получения данных от устройства стратегического управления (SCD) устройство тактического управления (TCD) отправляет данные на систему управления движением (ACD) каждой из осей (ACD).

После получения команды на движение происходит расчёт позиций, ускорений, а также отправка данных на систему управления движением (ACD) каждой оси. В случае выхода контролируемых параметров за диапазон разрешённых значений улавливается режим аварии,

и происходит вывод информации на информационную панель.

Панель индикации предоставляет информацию о текущих параметрах и состояний систем, а также о состоянии системы робота. Устройство облегчает мониторинг состояния робота манипулятора и обеспечивает быстрое выявление проблем, путём уведомления оператора. Когда система определяет невозможное дальнейшее движение робота, выполняется индикация красным цветом, цвет ожидания жёлтым, а работу зелёным.

Источник питания является неотъемлемым элементом системы. Так как он, преобразует сетевое электричество, путём формирования значений величины тока и напряжения (24В) до требуемых, для снабжения питанием всех электрических устройств робота.

Преобразователь напряжения необходим для изменения уровней питающего напряжения до требуемых параметров, так как различные устройства имеют различные значения допустимых параметров питающих напряжений и тока.

Бесщеточные электродвигатели постоянного тока, преобразуют электрическую энергию в механическую. Они обеспечивают движение различных осей робота манипулятора.

Устройство экстренного выключения, используется для мгновенной остановки робота в экстренных ситуациях, предотвращая аварии и повреждения конструкций.

Шина передачи данных обеспечивает коммуникацию между устройством тактического управления (TCD) и устройствами исполнительного управления (ACD). Данная коммуникация позволяет эффективно координировать действия всех устройств управления движением (ACD) между собой, а также производить одновременную синхронизацию во время движения.

2.3. Структурная схема системы исполнительного управления

Структурная схема систем управления движением осью изображена на рисунке 2.3. Назначение данной системы непосредственное управление синхронным двигателем и контроль его параметров. Главными элементами управления системой является устройства исполнительного управления (ACD). Поскольку для управления электродвигателем, необходимы сигналы с регулированием частоты и фазы, и именно ACD формирует управляющие PWM сигналы.

Драйвера увеличивают мощность управляющих PWM сигналов, тем самым обеспечивая согласование сигнала ACD и выходных силовых транзисторов. Драйверы позволяют производить своевременное и безопасное переключение транзисторов инвертора.

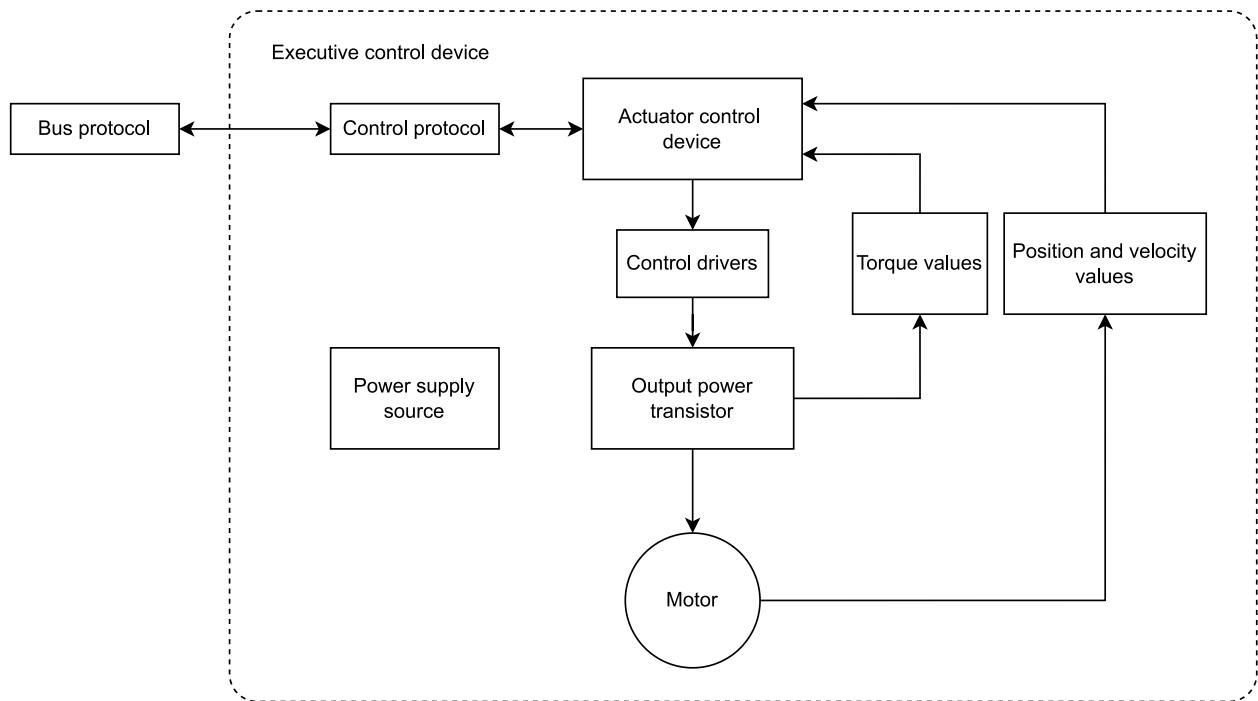


Рис. 2.3. Структурная схема устройства исполнительного управления

Сигналы, подаваемые на электродвигатель, контролируются силовыми ключевыми транзисторами. Электродвигатель необходимо регулировать путем изменения частоты и фазы сигнала

Силовые транзисторы образуют трехфазный мостовой инвертор напряжения и режим работы транзисторов — ключевой. В этом режиме транзисторы переключаются между полностью открытым и полностью закрытым состоянием, это обеспечивает высокий КПД и минимизирует нагрев транзисторов, что особенно важно при создании устройства небольшого размера.

Формирование PWM сигналов выполняется с использованием значений текущего угла поворота оси, ускорения и момента на двигателе. Путем изменения ширины импульсов PWM, система управления АСД может регулировать количество энергии, подаваемой на двигатель, тем самым изменяя параметры скорости и момента.

Преобразователь напряжения, обеспечивает формирование питающих напряжений для различных устройств. Тип преобразователя – DC/DC преобразователь, а преобразуемое напряжение – это напряжение 24В общего источника электропитания. В системе присутствуют различные компоненты с различными уровнями питающего напряжения, поэтому подавать питающее напряжение извне является не эффективным способом.

Шина данных позволяет производить передачу данных между различными устройствами, за счёт соединения компонентов в единую сеть. Благодаря использованию сети появляется возможность производить обмен данными как с одним, так и с несколькими устройствами одновременно.

2.4. Алгоритм работы устройства тактического управления

На рисунке 2.4 изображён главный алгоритм работы устройства тактического управления, главной задачей которого является управление двигателем одной оси робота. По умолчанию, при включении системы, происходит инициализация системы, подразумевает проверку работоспособности коммуникаций между устройствами и проверку некритического состояния каждого устройства ACD и двигателя. В качестве последнего шага, выполняется инициализация и установка параметров панели индикации.

Если связь с двигателями и интерфейсом управления ACD не нормализовалась, то система управления будет находиться в режиме ожидания до момента восстановления связи. После успешного соединения со всеми устройствами робота происходит установка и отправка команд на исполнительные устройства ACD, минимального значения скорости, угла и ускорения поворота для каждой оси робота. Так как положение осей робота манипулятора после предыдущего выключения неизвестно, необходимо выполнить калибровку всех осей. Калибровка происходит после получения команды разрешения со стороны устройства стратегического управления (SCD).

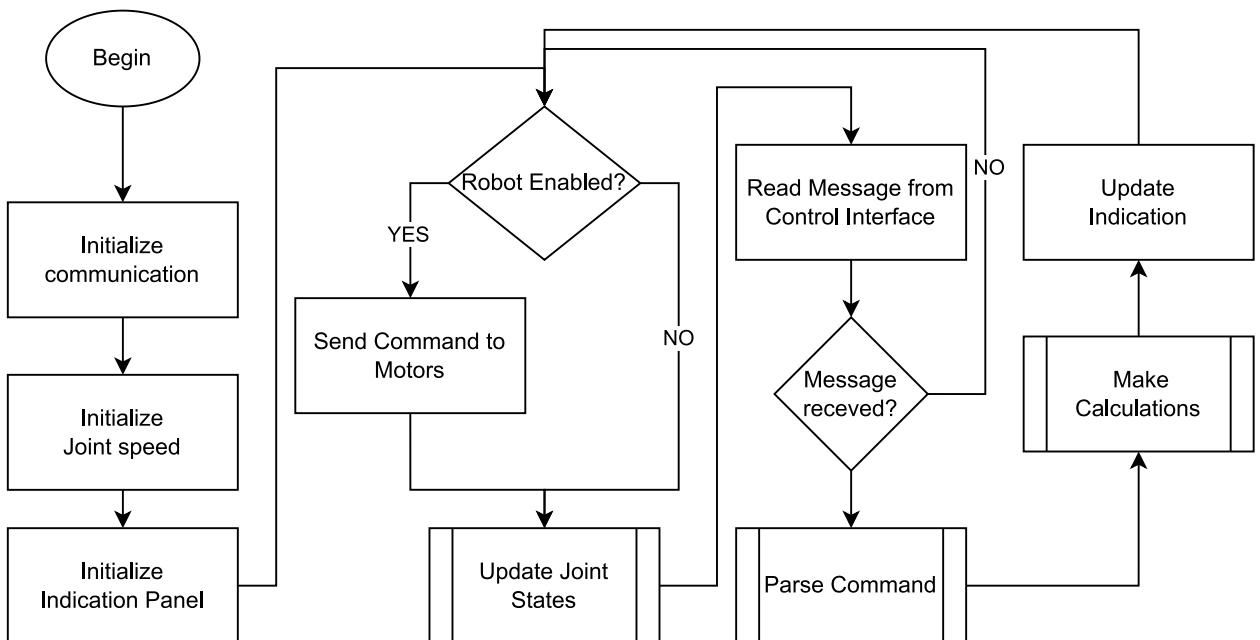


Рис. 2.4. Алгоритм работы устройства тактического управления

В варианте работы системы с поднятым флагом включённого состояния робота, после получения соответствующей команды, происходит считывание углов каждой из осей и отправка команд на ACD в зависимости от произведённых вычислений. Таким образом, устройство стратегического управления (SCD) не участвует в расчётах построения пути, а только отправляет данные на устройство тактического управления (TCD).

На изображении 2.5 иллюстрируется алгоритм «Parse command». После получения команды со стороны (SCD) происходит выполнение расчёта вида движения. Расчёт включает в себя набор алгоритмов вида прямолинейного (Move L) и осевого (Move J) движения. В случае получения команд, связанных с действиями робота, например данных об углах поворота или момента каждой из осей, система посыпает команду на опрос устройств ACD. Устройство TCD ожидает подтверждения на отосланную команду (ACK command). После этого заканчивается применение алгоритма «Parse command» и происходит переход в «Make calcualtion».

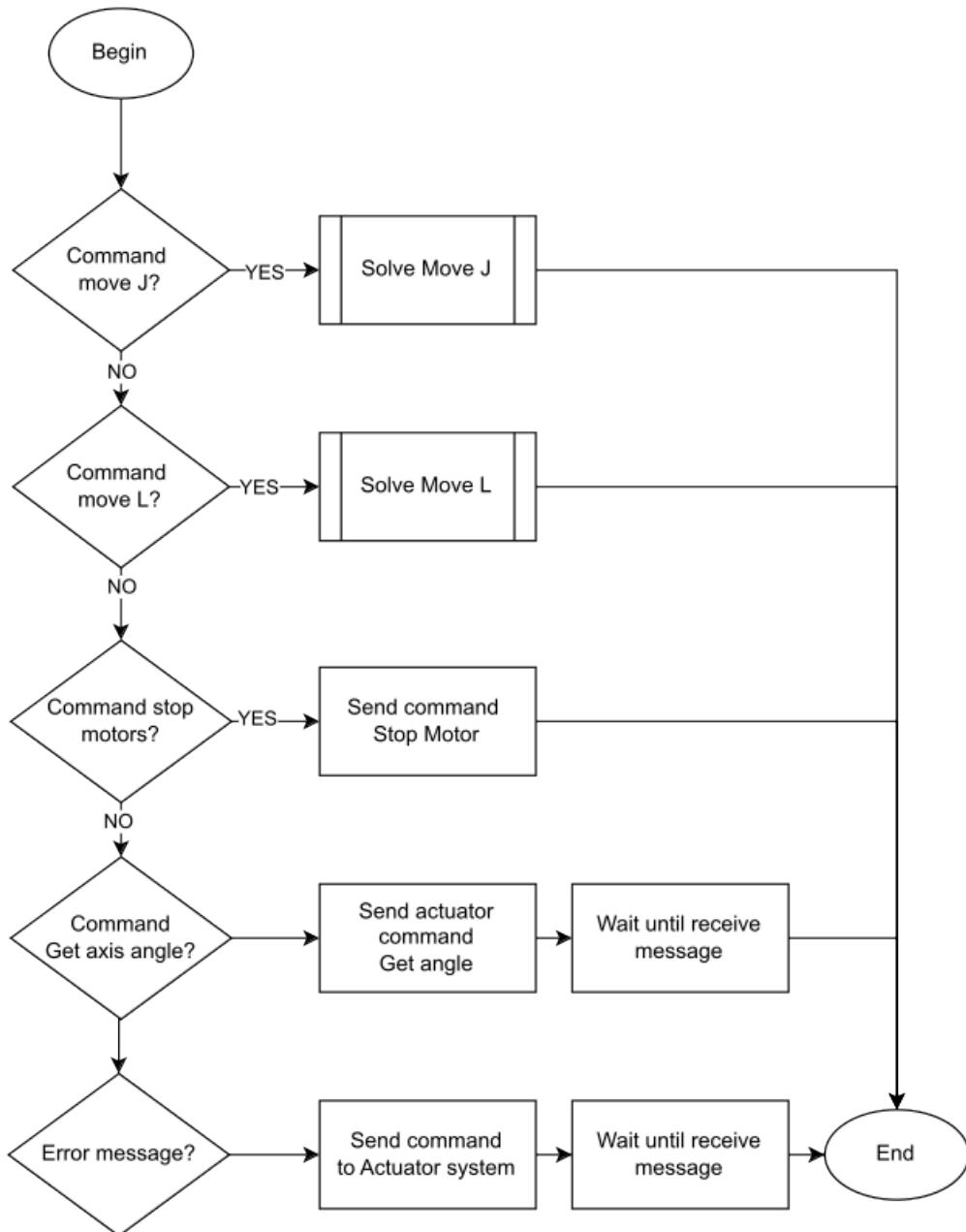


Рис. 2.5. Алгоритм «Parse command»

Основная подпрограмма расчёта траектории движения является, команда "Move J" рисунок 2.6. Основная суть - выполнение движения в определённую позицию более быстрым спосо-

бом с использованием угловых перемещений. В реализации данной функции происходит расчёт ускорений для каждой оси робота, но только в случае допустимых значений углов планируемой точки.

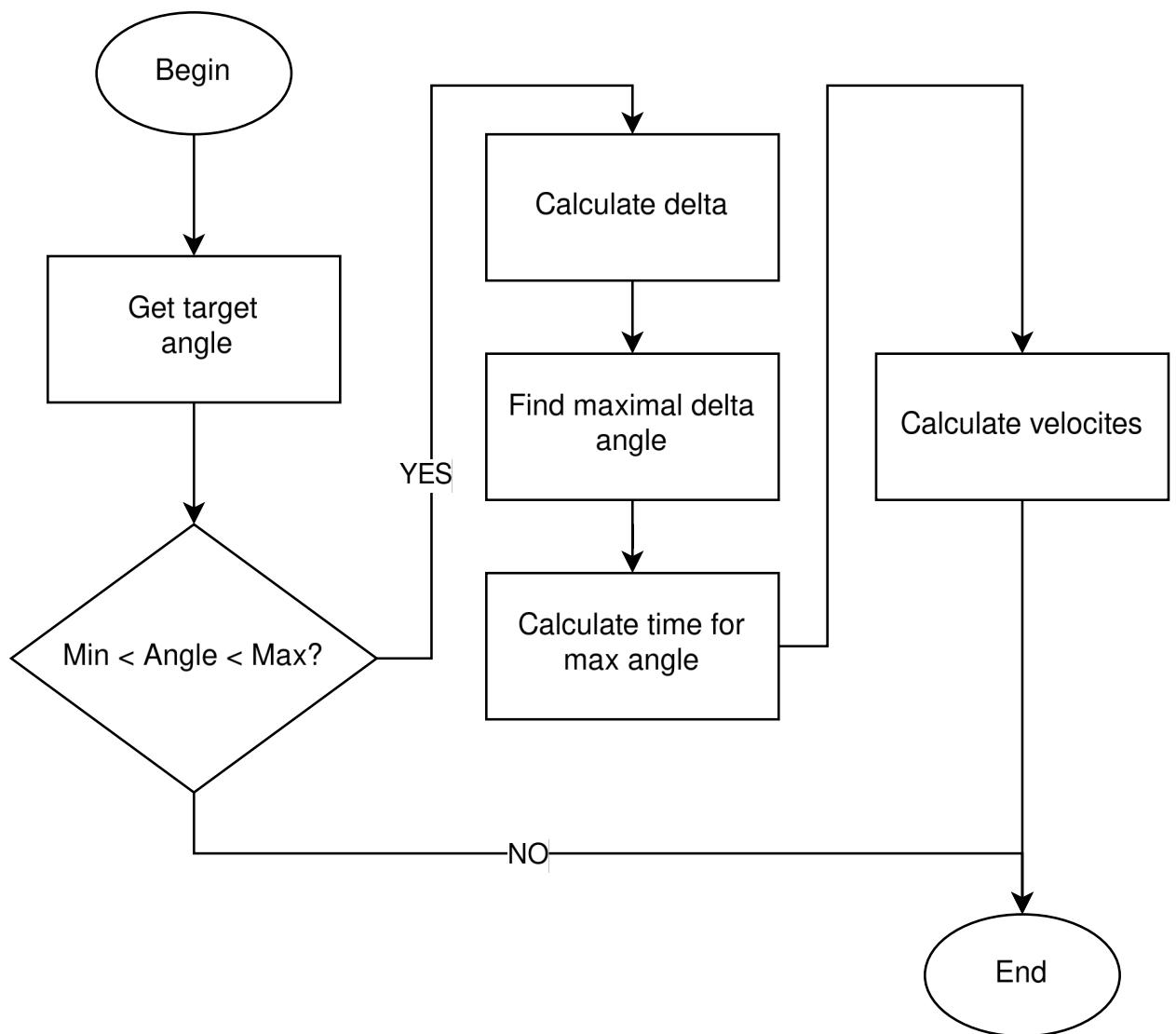


Рис. 2.6. Алгоритм подпрограммы «Move J»

Подпрограмма расчёта траектории движения, является командой вычисления линейного движения («Move L», рисунок 2.7), руки робота. В наборе алгоритмов выполняется решение обратной задачи, результатом данной задачи является несколько вариантов позиций, углов поворота звеньев робота. Далее требуется формирование списка вариантов позиций робота, которые не будут выходить за границы физических ограничений каждой оси робота. Так же требуется расчёт наименьшего угла от начальной позиции робота.

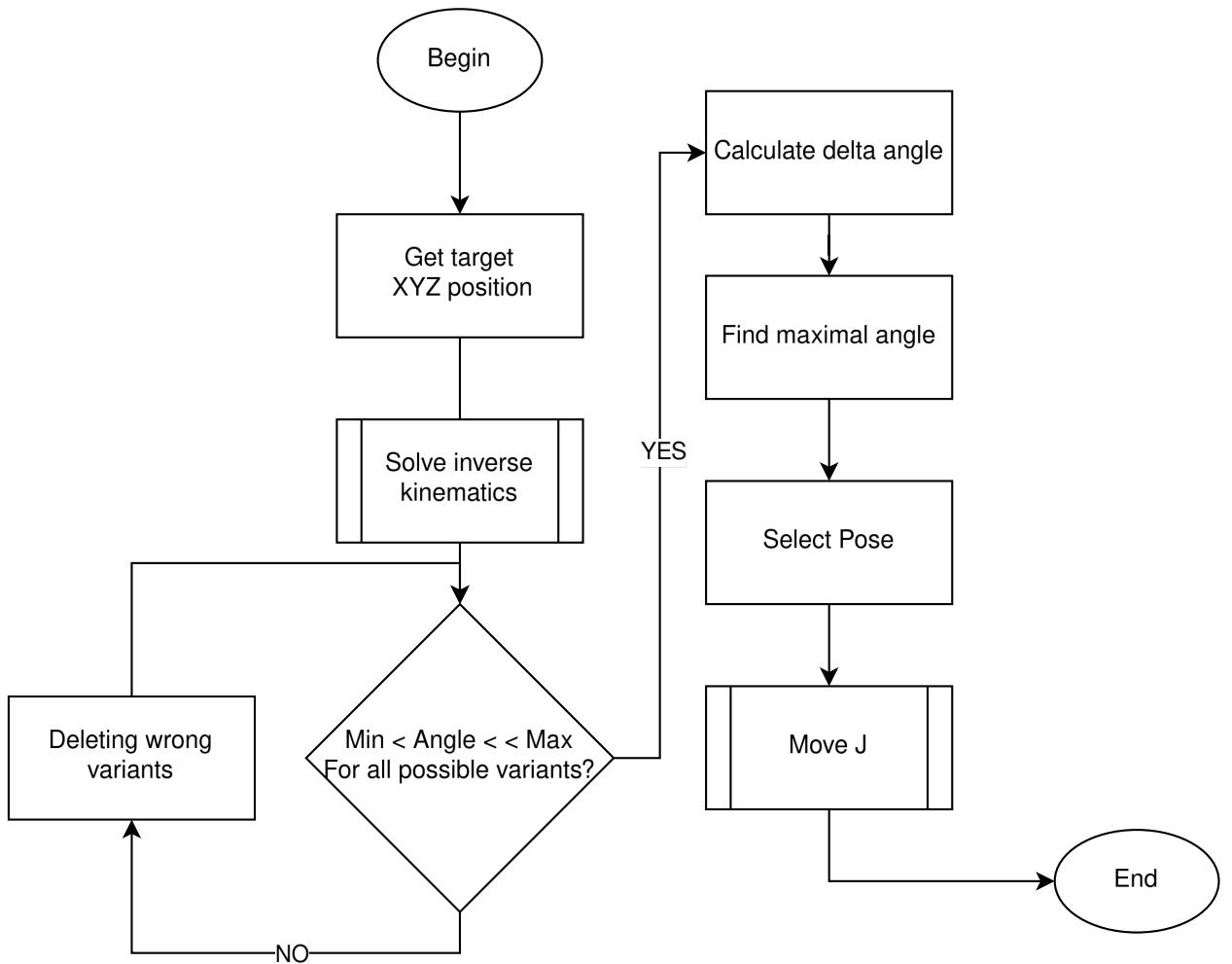


Рис. 2.7. Алгоритм подпрограммы «Move L»

2.5. Алгоритм работы устройства исполнительного управления

Алгоритм устройства исполнительного управления демонстрируется на рисунке 2.8. Данный алгоритм предназначен для управления электродвигателем постоянного тока синхронного типа, который отвечает за поворот управляемой оси робота на определённый угол, и контроля его параметров.

После включения системы происходит инициализация движения внутренней и внешней периферии. Если инициализация прошла успешно и связь установлена, то происходит проверка необходимости калибровки электродвигателя. Выполнение функции калибровки зависит от набора и значений параметров исполняемой программы. Калибровка необходима, если ось робота подвергалась разборке и было изменено положение устройства измерения угла поворота относительно исходного положения оси.

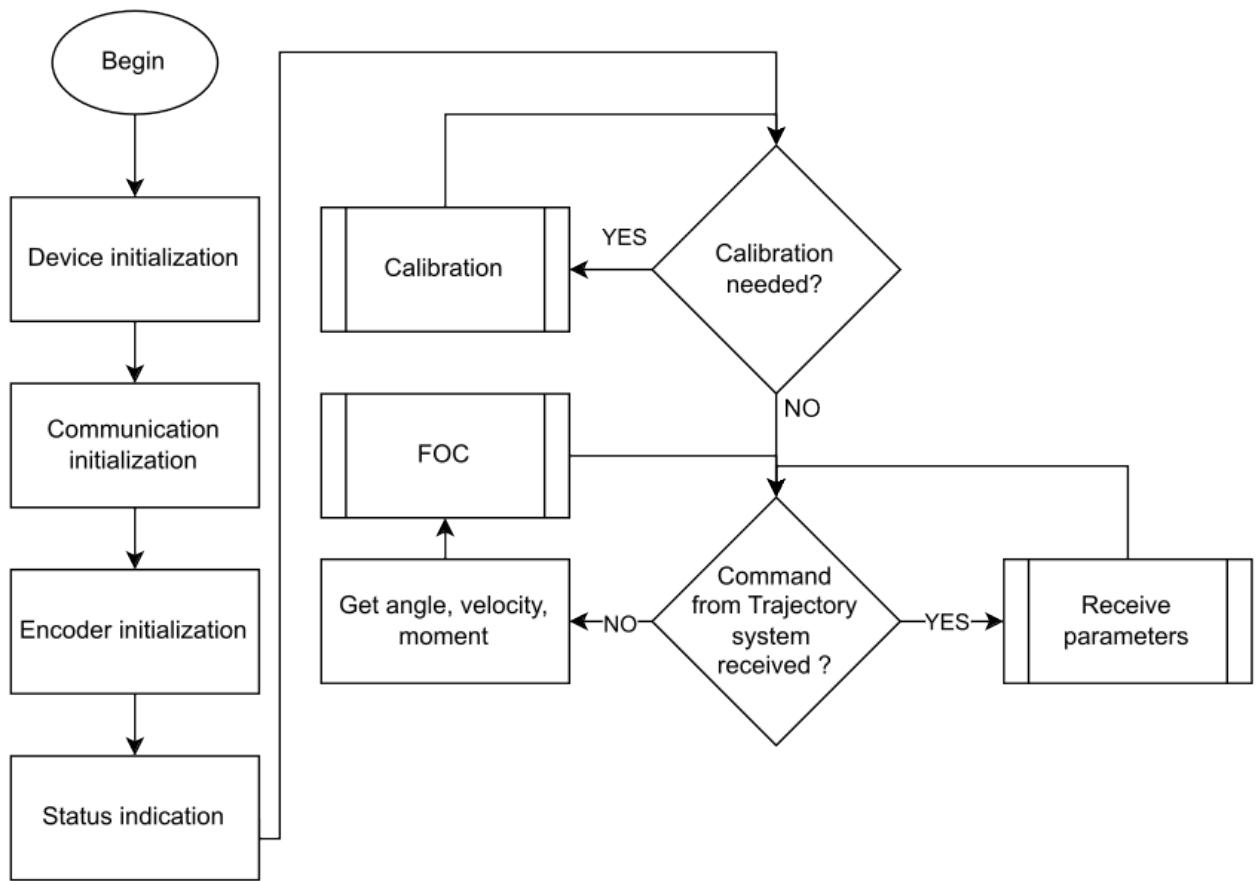


Рис. 2.8. Алгоритм работы «Actuator control device»

Далее с определённой периодичностью происходит опрос значений энкодера, которые преобразуются в углы поворота оси двигателя. Данные параметры необходимы в алгоритме расчёта векторного управления. При успешном вычислении параметров векторного управления происходит генерация выходных сигналов на драйвер.

На рисунке 2.9 изображён алгоритм обработки сообщений. Если сообщение от устройства тактического управления (TCD) было получено, то выполняется процесс извлечения данных из сообщения, которые содержат информацию о необходимой позиции, скорости и коэффициентах регулятора. Данные используются для вычисления параметров векторного регулирования. В случае получения сообщения на запрос данных, производится отправка параметров на устройство отправителя сообщения.

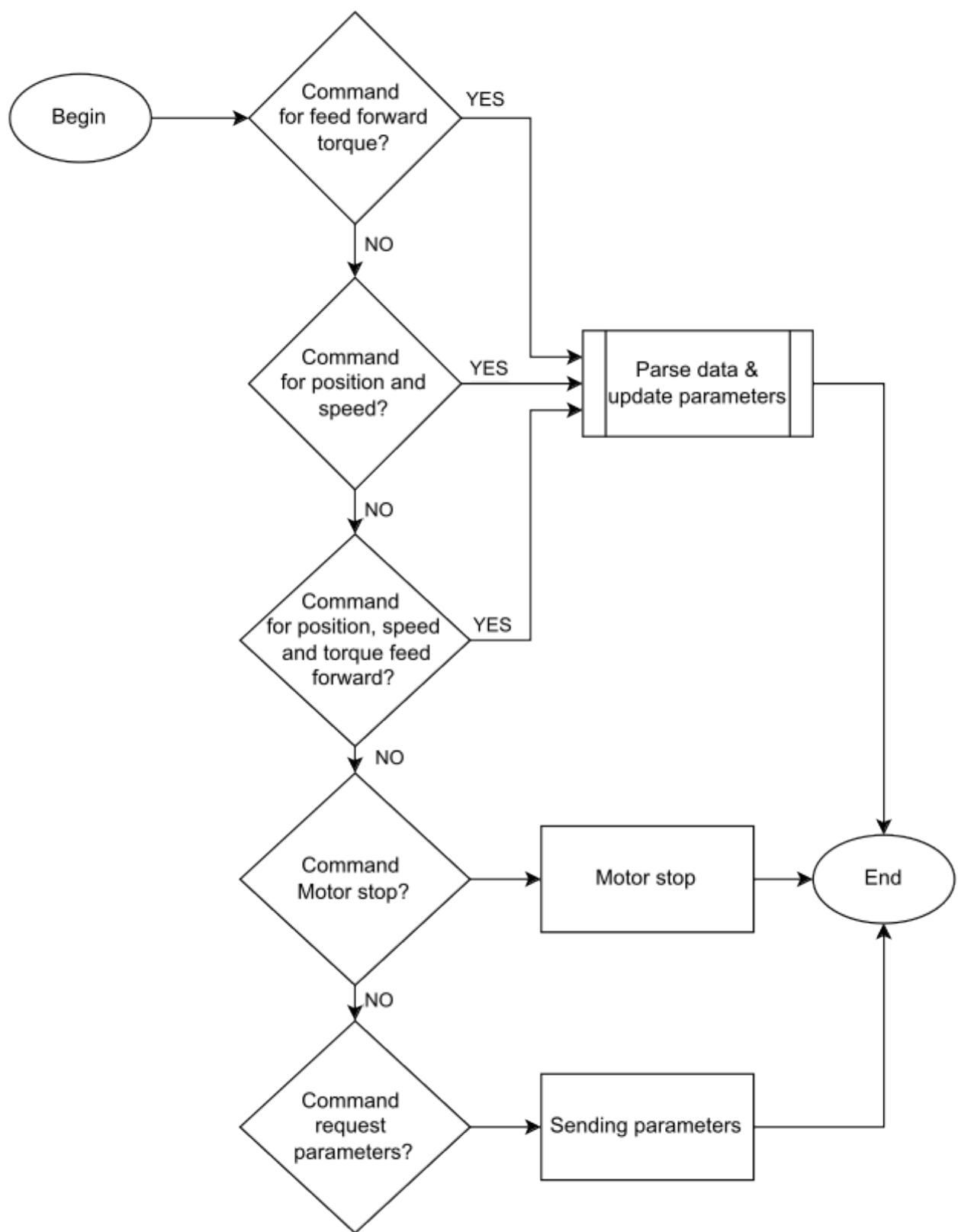


Рис. 2.9. Алгоритм работы подпрограммы «Receive parameters»

3 ФУНКЦИОНАЛЬНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА

Проектируемая система предназначена для выполнения задач управления робота манипулятором. Нецелесообразно размещать устройства исполнительной управления (ACK) в одном месте ввиду их количества. Поэтому целесообразно решение располагать устройства исполнительной части (ACK) вблизи электродвигателя для обеспечения своевременного и точного отклика на управляющие команды, а также для уменьшения электромагнитных помех и потерь мощности на проводах. Так же близкое расположение способствует упрощению обслуживания и повышению надежности системы управления в целом.

3.1. Gimbal двигатель

Gimbal электродвигатель в роботе является одним из компонентов механической части оси манипулятора. Двигатель выполняет передачу крутящего момента на ось манипулятора. Это усилие используется для перемещения различных частей робота, таких как звенья и суставы. В большинстве случаев для увеличения крутящего момента между двигателем и осью робота применяется редуктор (или передаточный механизм). Редуктор снижает скорость вращения выходного вала двигателя, увеличивая при этом крутящий момент. Так как прямое соединение двигателя с осью может не обеспечить достаточного усилия или может быть слишком быстрым для точных манипуляций.

Нагрузка, действующая на каждую ось, определяет требуемый крутящий момент, следовательно, был проведен выбор подходящего двигателя и редуктора для каждой оси. Учитывались габариты двигателя, ведь каждое последующее звено, начиная от первой оси и заканчивая последней должно быть легче и компактнее, обусловлено это необходимостью снижения нагрузки на двигатели и распределением массы самой конструкции. Исходя из технических требований, представленных в главе 1, необходимо использовать двигатели нескольких моделей.

Синхронные электродвигатели должны следующим требованиям:

- Внутреннее сопротивление примерно 10 Ом, для уменьшения нагрева электродвигателя с учётом габаритов;
- Высокий крутящий момент на низких скоростях, характеристика Kv менее 200 единиц;
- Рабочее напряжение от 10 до 20 В;
- Малый вес при моменте силы более 1.0 [кгс·см].

По результатам анализа моделей с различными параметрами в конструкции робота использовать электродвигатели от карданный шарниров камер (Gimbal). Модели данной катего-

рии двигателей обеспечивают оптимальный баланс между крутящим моментом и низкой скоростью вращения. Были выбраны модели GBM4008H-150T, GM5208-120T и GM3506 которые изображены на Рисунке 3.1. В данных моделях электродвигателей характерной особенностью является количество пар полюсов магнита и катушек ротора - более 20 (Таблица 3.1), так как скорость двигателя обратно пропорциональна числу полюсов ротора можно получить больший крутящий момент при низкой скорости вращения двигателя. Данные двигатели имеют большой крутящий момент при сравнительном небольшом весе до 200 граммов. Достоинством данных электродвигателей является широкий диапазон рабочих температур от -20°до +60°(simplefoc, 2023).



Рис. 3.1. «BLDC Gimbal motor GBM4008H-150T»

Таблица 3.1
Технические параметры моторов GBM4008H-150T, GM5208-120T и GM3506

Model	Diameter [mm]	Height [mm]	Weight [g]	Kv	Poles & Magnets	Resist. [ohms]	Torque [kgf-cm]	Max power [W]
GBM4008H-150T	46±0.05	21±0.2	107±0.5	68	24N22P	6.7	1.2	40
GM5208-120T	63±0.05	22.7±0.2	195±0.5	68	24N22P	6.7	1.9	40
GM3506	40±0.05	17.8±0.2	64±0.5	40	24N22P	5.6	1	25

3.2. Энкодер

В разработке системы управления робота манипулятора важную роль играет точное определение углов поворота его частей. Так же необходимо определять положение ротора электродвигателя для более плавного управления. Для определения угла поворота обычно используются энкодеры (sciencedirect, 2023).

Одной из важнейшей части является определение позиции каждого звена оси. Критерию точности предъявляется особые требования, исходя из технических требований, необходимо обеспечить повторяемость 0.02 mm на расстоянии 350мм, для обеспечения данной точности необходимо, обеспечить минимальной угол поворота оси на 0.008185°. Разрешение абсолютного энкодера должно соответствовать не менее 16 бит, но энкодеры такого разрешения только начинают появляться и являются дорогим решением в данном применении. Так же стоит учитывать, для векторного регулирования необходимо определять положение ротора двигателя, для обеспечения плавного пуска двигателя. Наилучшим решением двух проблем является установка энкодера на ротор двигателя, а за счет редуктора вырастет не только момент, но и разрешающее способность угла поворота. Для обеспечения приемлемой точности необходимо использовать редуктор со степенью редукции не менее 11 при разрешении самого энкодера 12 бит. Только необходимо учитывать люфт во всех механических механизмах и устанавливать гармонические или циклоидальные безлюфтовые редукторы (Sensinger and Lipsey, 2012).

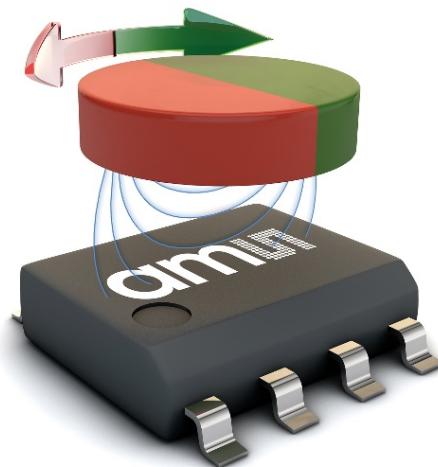


Рис. 3.2. Микросхема определения осевого положения AS5600

В качестве энкодера была выбрана микросхема AS5600 (Рис. 3.2) с цифровым выходом от фирмы «AMS». Датчик магнитного осевого положения AS5600 является бесконтактным датчиком определения абсолютного угла поворота, которая обеспечивает 12 битную точность в измерении углов, в диапазоне от 0 до 360 градусов. AS5600 сочетает в себе надежность и простоту интеграции за счет использования цифрового интерфейса передачи данных I²C (Таблица 3.2), что делает его отличным выбором непрерывного и точного контроля за угловым положением оси мотора робота (ams, 2023).

Таблица 3.2
Таблица параметров микросхемы осевого положения AS5600

Parameters	Value
Resolution [bit]	12
Output	Analog out / PWM / I ² C
Supply Voltage [V]	3-3,6
Temperature Range [°C]	-40 to +125
Package	SOIC-8
Max current [mA]	100
Sampling rate [μs]	150
Max RPM	20000
Position precision [°]	0.087

3.3. Устройство стратегического управления

Устройство стратегического управления необходимо для решения задач движения, а также является интерфейсом между устройством тактического управления и устройством интеллектуального уровня. Тем самым предоставляя контроль параметрами и управления самим роботом. Устройство стратегического управления должно соответствовать следующим требованиям:

- Выполнение нескольких задач одновременно (Многозадачность);
- Небольшой размер;
- Наличие GPIO;
- Высокая вычислительная мощность;
- Возможность использования библиотек и функций операционной системы;
- Малое напряжение питания;
- Поддержание технологий беспроводной связи, спецификации IEEE 802.11 или 802.15.

Эти требованиям отвечает одноплатный мини- компьютер «Raspberry Pi Zero W 2» (Рисунок 3.3) от компании «Raspberry Pi Foundation». Высокая вычислительная мощность платы (Таблица 3.3) и преимущества операционной системы «Linux» способствуют разработке и выполнению сложных алгоритмов и задач управления на самом устройстве. К тому же поддержка широкого спектра библиотек и доступу к 40-контактному разъему GPIO из операционной системе.

мы, значительно упрощает разработку сложных программ управления и интеграцию с другими системами. В отличии от предыдущих разработок «Raspberry Pi Zero», модификация «Pi Zero 2» имеет более производительный четырехядерный процессор «ARM Cortex-A53» и исправленный дизайн антенны для обеспечения лучшей беспроводной передачи информации.

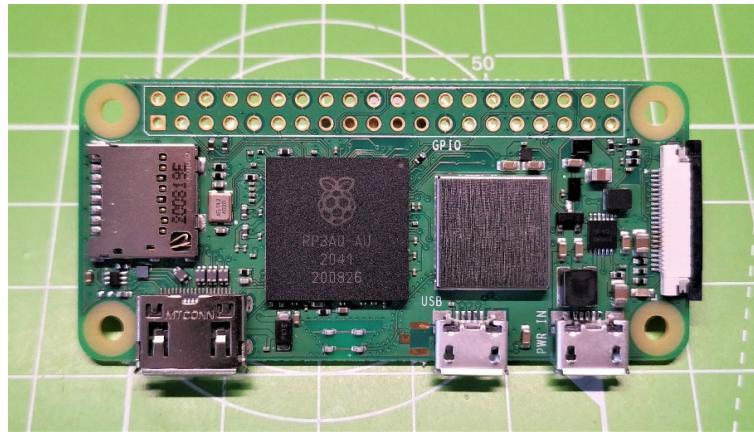


Рис. 3.3. Одноплатного миникомпьютера «Raspberry Pi Zero W 2»

Таблица 3.3

Таблица параметров одноплатного мини-компьютера Raspberry Pi Zero W 2

Parameters	Value
Processor	Broadcom BCM2710A1, Quad-core Cortex-A53 (ARMv8) 64-bit SoC @ 1GHz
RAM	512MB LPDDR2
Wireless Connectivity	802.11 b/g/n wireless LAN, Bluetooth 4.2, BLE
Ports	Mini HDMI, USB On-The-Go port, Micro USB power
GPIO Pins	40-pin GPIO header
Power Requirement	5V/2.5A DC via micro USB connector or GPIO
Size	65mm x 30mm x 5mm

Основываясь на требованиях к разрабатываемому устройству стратегического управления, с учетом его специфики работы, принято решение использовать операционную систему «Raspberry Pi OS Lite», которая основанная на дистрибутиве «Debian 12» 32-битной версии. Основной особенностью данного дистрибутива является стабильность, безопасность и широкая поддержка пакетов.

3.4. Контроллер тактического и исполнительного управления

Согласно требованиям разрабатываемой системе управления, устройством управления для тактического и исполнительного устройства должно иметь малые габариты поэтому для реализации необходимо использовать микроконтроллер, за счет малых его размера. В задачу микроконтроллера входит: считывание и обработка данных от датчиков, осуществление обмена данными по шине данных, производство численных вычислений для векторного регулирования, прямой и обратной задачи кинематики, а также осуществление генерации сигналов PWM для инвертора. Минимальные требования к микроконтроллеру:

- Рабочее питание от 3В;
- Наличие более 20 линий ввода/вывода;
- Порты для работы с CAN-шиной;
- Порты для работы с PWM;
- Аппаратная поддержка I2C протокола на выводах МК;
- Наличие внутреннего АЦП;
- Наличие механизма прямого доступа к памяти DMA.

Основной проблемой при выборе микроконтроллера, был недостаток информации применительно к решению задач системы управления тактического и исполнительного управления. Проблема заключалась в выборе лучшей модели МК при решениях кинематических задач. Невозможно было определить какой МК, среди самых распространённых на рынке, имеет лидирующее место. Микроконтроллеры сильно отличаются внутри одного семейства, а при рассмотрении моделей от разных производителей, архитектура каждого МК сильно отличается, надо было учесть и (toolchain) набор инструментов программирования, который тоже мог влиять на производительность. Следовательно, возникла необходимость провести тестирование различных видов микроконтроллеров общего применения, доступных на рынке.

3.5. Тестирование микроконтроллеров

Проведённое тестирование было необходимо для нахождения подходящей модели микроконтроллера. Написание исполняемой программы для каждого конкретного микроконтроллера является трудным и долгим процессом, необходимо учитывать архитектуру микроконтроллеров и оптимизировать алгоритм под каждое семейство. Поэтому для выбора наиболее подходящего микроконтроллера было проведено синтетическое тестирование. Основным критерием выбора была скорость выполнения конкретных математических операций. Высокая скорость выполнения необходима для обеспечения эффективной работы системы в реальном времени. Это

особенно важно в задаче векторного регулирования, где задержки в обработке данных приводят к существенным потерям точности, а также отзывчивости всей системы в целом. При более подробном рассмотрение задач связанных с вычислением задач кинематики и векторного регулирования (в частности прямое и обратное преобразование кларка формулы(3, 4, 5), обратного преобразования парка формулы(1, 2), было обнаружено большое количество вычислений(Huang et al., 2011):

- Умножение и деление чисел с плавающей запятой;
- Вычисление тригонометрических выражений, функции синуса и косинуса;

$$U_\alpha = U_q \sin(\theta) \quad (1)$$

$$U_\beta = U_q \cos(\theta) \quad (2)$$

- Вычисление квадратного корня.

$$u_a = U_\alpha \quad (3)$$

$$u_b = \frac{-U_\alpha + \sqrt{3}U_\beta}{2} \quad (4)$$

$$u_c = \frac{-U_\alpha - \sqrt{3}U_\beta}{2} \quad (5)$$

Приведённые выше пункты являются сложными для вычислений на ядре любого микроконтроллера и нуждаются в большом количестве операций процессора для их вычисления. Вычисление подобных выражений будет занимать весомое количество времени, чем негативно скажется на быстродействии работы всего управления, так и на работе отдельных её частей(Pack and Barrett, 2008).

Для проведения теста были выбраны микроконтроллеры от производителей: «STMicro electronics», «Raspberry PI Foundation», «Espressif Systems». Главным критерием сравнения было измерение затраченного времени на каждый тип вычисления. Во время тестирования не использовались действия, связанные с приостановкой главного цикла исполняемой программы, например прерывания.

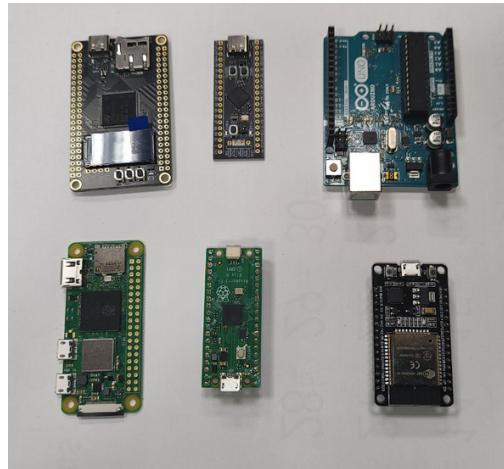


Рис. 3.4. Платы, которые участвовали в тестировании

В таблице 3.4 приведен перечень использовавшихся микроконтроллеров, их тактовая частота и процессор. Платы, которые участвовали в испытании изображены на рисунке 3.4.

Таблица 3.4
Список микроконтроллеров и их характеристики

Board	MCU	Fcpu (Mhz)	CPU
Arduino Uno (original)	ATmega328p	16	AVR
NodeMCU-ESP32	ESP32	240	Dual Xtensa LX6
NodeMcu V3	ESP8266	160	Xtensa L106
Raspberry Pi Pico	RP2040 [Arduino IDE]	133	Dual Cortex M0+
Raspberry Pi Pico	RP2040 [C++ SDK]	133	Dual Cortex M0+
Raspberry Pi Pico	RP2040 [MicroPython]	133	Dual Cortex M0+
STM32 Nucleo-32	STM32G431KB	170	Cortex M4
STM32 Nucleo-32	STM32G431 [FPU-OFF]	133	Cortex M4, FPU-OFF

Суть программы в микроконтроллере состояла в вычислении затрачиваемого времени с использованием порта GPIO. Перед вычислением операции выключаются все прерывания, микроконтроллер поднимает сигнал на выводе в высокое состояние, производится вычисление математической операции, на выводе ножки МК опускает в нижнее состояние, прерывания восстанавливаются. Далее при помощи осциллографа НМО2024 от производителя «ROHDE & SCHWARZ», измерялась длительность импульса и данные записывались в таблицу 3.5 (меньше значение - лучше), стоит отметить, что при расчётах учитывалось время переключения выходов GPIO, записанные во 2 колоне таблицы 3.5, результаты показаны на графике 3.5.

Таблица 3.5
Таблица измеренного времени на вычисление математических операций

MCU	Time IO [μs]	a + b [μs]	a - b [μs]	a * b [μs]	a / b [μs]	sin(a) [μs]	log(a) [μs]	sqrt(b) [μs]	pow(b, a) [μs]
ATmega328p	0,144	9,134	8,818	10,141	31,154	104,655	155,218	31,133	328,810
ESP32	0,312	0,402	0,389	0,399	0,567	0,798	1,357	0,691	3,120
ESP8266	0,508	0,942	0,959	1,214	2,296	13,393	26,611	8,454	76,808
RP2040 [Arduino IDE]	0,514	1,477	1,555	1,884	4,233	17,052	28,012	4,090	62,245
RP2040[C++ SDK]	0,008	0,750	0,783	0,650	0,819	4,634	6,610	0,709	17,288
RP2040 [MicroPython]	6,456	18,412	17,086	16,858	17,499	23,491	25,253	19,688	41,450
STM32G431	0,005	0,063	0,065	0,056	0,140	0,449	0,957	0,136	5,266
STM32G431 [FPU-OFF]	0,007	0,508	0,529	0,350	1,217	5,414	12,395	2,566	42,773

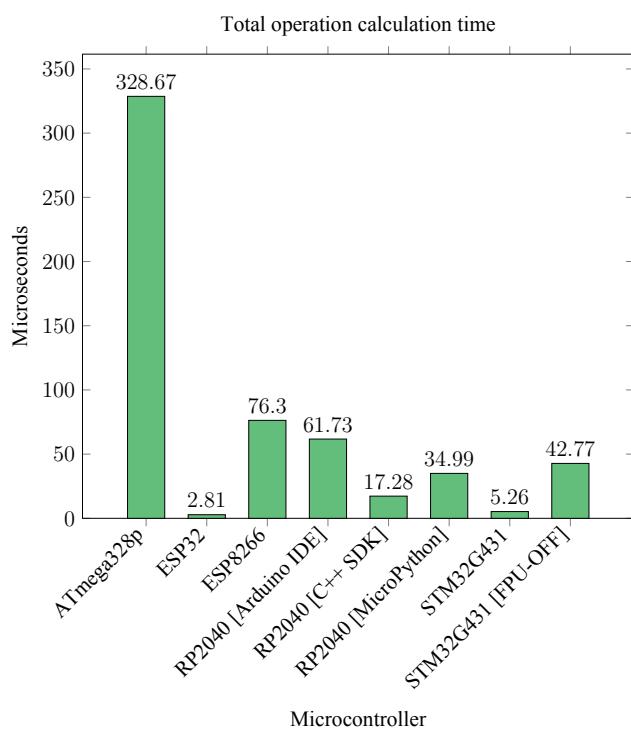


Рис. 3.5. График сравнения времени вычисления математических операций у различных МК

Приведённые данные не дали чёткую информацию о преимуществах конкретного контроллера в конкретных вычислениях, для давнейшего и более удобного анализа таблица пересчитана в значения тысяч операций в секунду kOPS таблица 3.6, формула(6).

$$kOPS = \frac{1000}{T_{cal} - T_{io}} \quad (6)$$

Таблица 3.6

Таблица сравнения тысяч операций в секунду на вычисление математических операций у каждого МК

MCU	a + b	a - b	a * b	a / b	sin(a)	log(a)	sqrt(b)	pow(b, a)
ATmega328p	111,23	115,29	100,02	32,25	9,57	6,45	32,27	3,04
ESP32	11 090,82	12 961,74	11 549,13	3 918,10	2 056,26	956,62	2 641,39	356,12
ESP8266	2 305,87	2 217,75	1 416,38	559,44	77,61	38,31	125,85	13,11
RP2040 [Arduino IDE]	1 039,35	961,04	730,02	268,92	60,47	36,37	279,68	16,20
RP2040 [C++ SDK]	1 347,59	1 289,95	1 558,30	1 232,40	216,19	151,47	1 425,57	57,87
RP2040 [MicroPython]	83,64	94,07	96,13	90,56	58,70	53,20	75,57	28,58
STM32G431	17 179,17	16 702,61	19 476,01	7 383,89	2 253,15	1 050,29	7 624,36	190,09
STM32G431 [FPU-OFF]	1 996,11	1 918,50	2 917,02	826,69	184,94	80,73	390,85	23,38

Суммарные результаты пересчёта времени в значения тысячи операций в секунду (kOPS) представлены в таблице 3.6 (значения больше — лучше), на рисунке 3.6 изображён график максимальных значений тысяч операций в секунду для каждого микроконтроллера. В приведённых расчетах микроконтроллеры выполняли операции на максимальной возможной тактовой частоте Fcpu, но у каждого МК максимальная тактовая частота разная (таблица 3.5), для оценки эффективности микроконтроллеров был проведён расчёт количества тысяч операций в секунду на мегагерц(7). График приведён на рисунке 3.5(значения больше — лучше), таблицу с расчётомами можно найти в приложении 1.

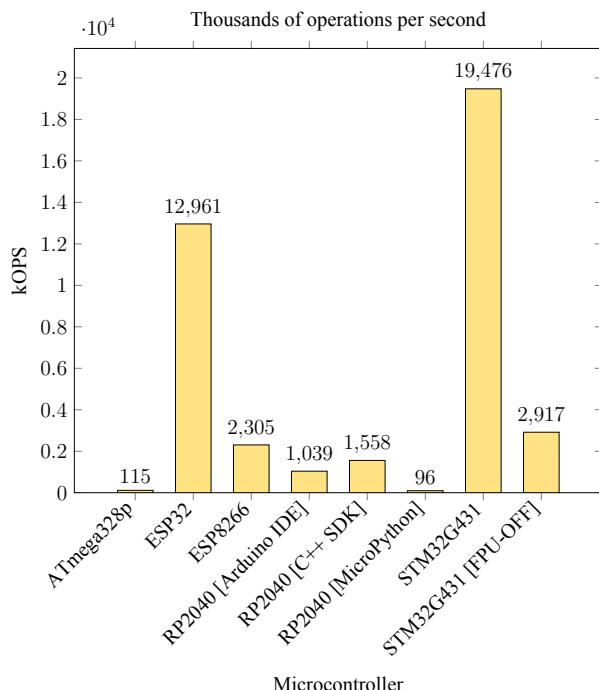


Рис. 3.6. График максимальных значений тысяч операций в секунду у различных МК

$$kOPS = \frac{K_{OPS}}{Mhz} \quad (7)$$

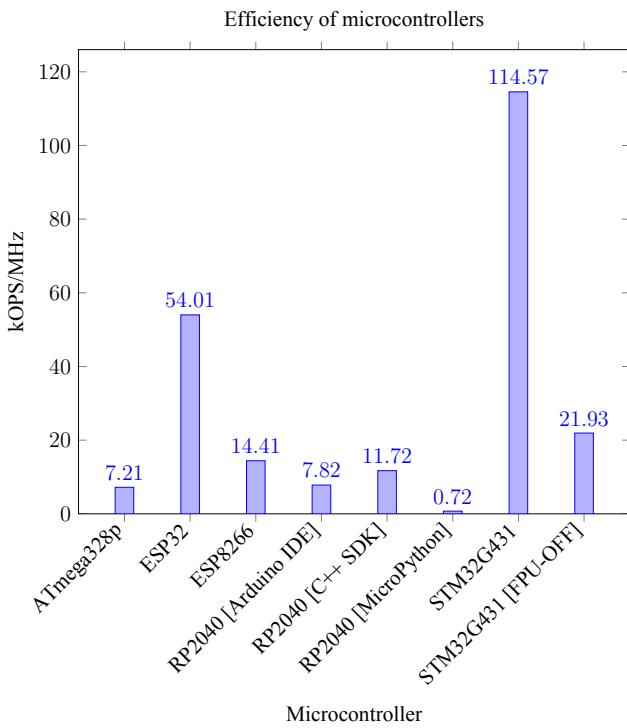


Рис. 3.7. График значений тысяч операций в секунду на 1 мегагерц каждого МК

В результате проведённых тестов было выяснено несколько интересных моментов связанных с микроконтроллерами. Как можно увидеть на рисунке 3.4 один процессор RP2040 с разными наборами средств разработки выдают различные результаты. Фаворитами теста являются МК ESP32 с процессором Dual Xtensa LX6 от китайской компании «Espressif Systems» и микроконтроллер STM32G431 от компании «STMicroelectronics» с процессором Cortex M4. Чип STM был наиболее производительнее чем ESP32, хотя тактовая частота у ESP32 почти вдвое выше STM, но нет значительного прироста в математических операциях. Стоит учесть STM вышел в лидеры по времени, из-за наличия аппаратных блоков операций с плавающей запятой (FPU) в процессоре.

А также сопроцессора аппаратного ускорения математических операций «CORDIC». В результате контроллером системы управления был выбран чип STM32G431, в таблице 3.7 представлены основные характеристики данного чипа (STM, 2023).

Таблица 3.7

Таблица основных характеристик микроконтроллера STM32G431

Parameter	Specification
Core	ARM Cortex-M4
Operating Frequency	Up to 170 MHz
Flash Memory	Up to 512 KB
RAM	Up to 128 KB
Digital I/Os	Up to 82
Timers	Multiple, including general purpose and advanced
DAC	Yes, up to 2 channels
ADC	Yes, up to 16-bit
Interfaces	I2C, SPI, UART, USB, and others
Operating Voltage	2.0 V to 3.6 V
Temperature Range	-40°C to +85°C
CAN Bus	Yes, CAN 2.0 (A, B)
Operational Amplifier (Op-amp)	Yes

3.6. Силовые ключи

Силовые транзисторные ключи формируют трехфазный инвертер, для управления электродвигателем. В данном случае транзисторные ключи задействованы для усиления PWM сигналов, поступающих от МК. Тип транзистора - N-канальный MOSFET транзисторы с индуцированным каналом. Такие транзисторы имеют очень малое сопротивление канала ($R_{ds(on)}$). Это означает меньшие потери мощности при управлении нагрузкой, нагревание элемента будет меньше, что при одинаковых значениях параметров позволяет выбрать модели с меньшими габаритами. Силовые ключи должны соответствовать следующим требованиям:

- Рабочее выходное напряжение более 24В;
- Рабочий температурный диапазон -30 до +80;
- Тип транзистора – MOSFET;
- Тип канала – n;
- Маленький размер элемента;
- Тип установки – поверхностный монтаж.

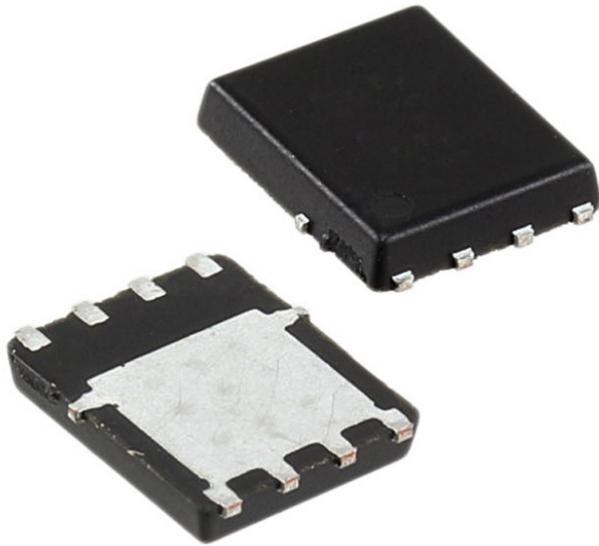


Рис. 3.8. Транзистор «SIR680DP»

Таблица 3.8

Таблица основных характеристик транзистор SIR680DP

Parameter	Specification
Type	N-Channel
Drain-Source Voltage (VDS)	80 V
Gate-Source Voltage (VGS)	± 20 V
Continuous Drain Current (ID)	100 A (at TC = 25 °C)
Pulsed Drain Current (IDM)	200 A (for 100 µs)
Power Dissipation (PD)	104 W (at TC = 25 °C)
Operating Temperature Range	-55°C to +150°C
RDS(on) max. at VGS = 10 V	0.0024 Ω
Configuration	Single
Package	PowerPAK SO-8

Были выбраны транзисторы SIR680DP (Картина 3.8) от «Vishay Siliconix», SIR680DP - n-канальный MOSFET мощностью 80В / 100А. Они обладают высокой плотностью тока и низким сопротивлением открытого канала (RDS(on)), порядка 2.4 mΩ, остальные параметры находятся в таблице 3.8, что стало идеальной характеристикой подходящего элемента для применения с использованием ширина-импульсной модуляции (PWM). Также важно отметить, что выбор

компонентов был продиктован маленьким размером корпуса «PowerPAK-SO-8»,

3.7. Драйвер силовых ключей

Для облегчения управления мостовой транзисторной схемой инвертора используются драйвера. Их задача - преобразование маломощного сигнала, снимаемого с цифрового выхода МК, в сигнал с повышенным уровнем напряжения и мощности.

В данной реализации исполнительной системы использовались драйвера общего назначения L6385ED (Рисунок 3.9) производимые фирмой «STMicroelectronics». Данная микросхема является полумостовым драйвером. Один драйвер может управлять 2 силовыми транзисторами верхнего и нижнего уровня. Технические параметры L6385ED представлены в таблице 3.9.



Рис. 3.9. Драйвер L6385ED

Таблица 3.9

Таблица основных характеристик драйвера L6385ED

Parameter	Specification
Maximum Operating Voltage	580V
Supply Voltage	±50V/nsec (across full temperature range)
Driver Current Capability (Sourcing)	400mA
Switching Time (rise/fall)	50/30 nsec
Level of Input Control Signals	CMOS/TTL Schmitt trigger with hysteresis and pull down
Under-voltage Lockout	On both lower and upper sections
Bootstrap Diode	Internal

3.8. Передатчик шины данных

Для обеспечения передачи данных между устройствами исполнительного управления, стратегического управления и других устройств целесообразно использовать шину данных, для уменьшения количества соединительных проводников. Наиболее подходящим протоколом передачи данных является CAN шина, который является промышленным стандартом промышленной сети и не однократно использовалась для внедрения в роботы манипуляторы (Megalingam et al., 2021). Главная причина использования шины CAN, это особенность в передачи данных, данные доступны всем устройствам, подключенными к сети. Возможна передача данных как одному, так и нескольким устройствам одновременно. На основе этого реализовывался вариант синхронизации всех исполнительных устройств, необходимых для своевременного начала движения (stm, 2020b). На основе выбранной модели микроконтроллера в главе 3.5 и технических данных из таблицы 3.10, была выявлена аппаратная поддержка микроконтроллером CAN шины второго поколения - CAN FD.

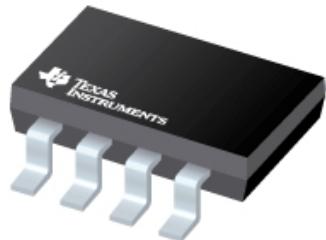


Рис. 3.10. Передатчик TCan1462DRQ1

Таблица 3.10
Таблица основных характеристик передатчика TCan1462DRQ1

Parameter	Specification
Package	SON-8 (3x3mm)
Operating Temperature Range	-40°C to +125°C
Supply Voltage Range	4.5V to 5.5V
Standby Current	Typically 5µA
Data Rate	Up to 5 Mbps
ISO 11898-2 Compliance	Yes

CAN-FD – это следующий этап развития классической шины CAN, которая обеспечивает более высокую скорость передачи данных и больший объем передаваемых данных в одном

кадре, такое возможно из-за особенности передачи данных поля (CAN data) на скорости кратно превышающую скорость передачи заголовка, что может иметь значение вплоть до 8 Мбит/с.

Чтобы не ограничивать в возможной скорости передачу данных по шине CAN, был выбран передатчик TCAN1462DRQ1 (изображен на рисунке 3.10), передатчик выполняет усиление сигналов, защиту линии в случае повреждения CAN-шины, и регулировку скорости их передачи.

3.9. Преобразователь питания

Из-за высокого потребления мини-компьютера, потребления достигает до 2.5А. По рекомендации производителя необходимо использовать отдельный DC – DC преобразователь. Источник питания должен соответствовать следующим требованиям:

- Входное напряжение 24В;
- Преобразование DC-DC;
- Выходное напряжение 5В;
- Высокий выходной ток 3А и более.

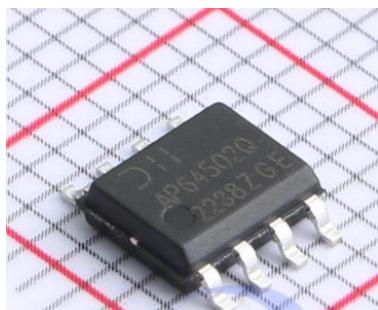


Рис. 3.11. DC – DC микросхема AP64502QSP

Была выбрана микросхема DC-DC Step-down преобразователя AP64502QSP, показанная на рис. 3.11. Преимущество импульсного преобразователя в высокой эффективности, не требующей теплоотвода, позволяет создавать компактные решения преобразования напряжения и тока. Основным преимуществом выбора является способность преобразования большого тока 5А (таблица 3.11), без использования внешних ключей и меньшего количества необходимых для работы данной микросхемы элементов (DIODES, 2021).

Таблица 3.11
Таблица основных характеристик DC-DC преобразователя AP64502QSP

Parameter	Specification
Input Voltage Range (VIN)	3.8V to 40V
Continuous Output Current	5A
Programmable Switching Frequency	100kHz to 2.2MHz
Efficiency at Light Load	Up to 85%
Gate Driver Design	Proprietary, for EMI Reduction
Frequency Spread Spectrum (FSS)	Yes, for EMI Reduction
Low-Dropout (LDO) Mode	Yes
Precision Enable Threshold	For UVLO Adjustment
Protection Features	UVLO, OVP, Peak Current Limit, Thermal Shutdown

3.10. Функциональная схема системы управления мини роботом

Функциональная схема системы управления мини роботом манипулятором показана на рис. 3.12. Главным управляющим элементом системы тактического управления, является управляющий МК STM32G431. К данному микроконтроллеру подключены все необходимые элементы.

Подключение Raspberry Pi Zero W2 (в роли стратегического устройства) к главному управляющему элементу STM32G431 возможно через два интерфейса UART, передача информации в текстовом виде и через SPI, для передачи сериализованных данных больших объёмов на большой скорости. Устройство Raspberry Pi задействует 5 линий ввода/вывода:

- PA7 (SPI1_MOSI) →GPIO 12 (RPI_MOSI);
- PA6 (SPI1_MISO) →GPIO 13 (RPI_MISO);
- PA5 (SPI1_SCLK) →GPIO 14 (RPI_SCLK);
- PA3 (UART2_RX) →RPI (RPI_TX);
- PA2 (UART2_TX) →RPI (RPI_RX).

Осуществление общения между устройствами исполнительной системы, используется передатчик шины CAN (TCAN1462) и передатчик задействует 2 линии ввода/вывода порта B, микроконтроллера STN32G431:

- PB9 (FDCAN_TX) →CAN Transceiver (CAN_TX);

- PB8 (FDCAN_RX) → CAN Transceiver (CAN_RX);

Для обеспечения внешней связи используется передатчик интерфейса RS485 который подключается к 2 линии ввода/вывода порта А.

- PA10 (UART2_TX) → RS485 (UART_TX);
- PA9 (UART2_RX) → RS485 (UART_RX);

Индикация состояния робота происходит через панель индикации и состоит из трёх светодиодов разного цвета и занимает 3 линии ввода/вывода порта В.

- PB0 ← Indication Panel (LED_YEL);
- PB1 ← Indication Panel (LED_RED);
- PB2 ← Indication Panel (LED_GRN);

Остальные свободные линии ввода/вывода портов В, отведены на блоки цифровых входов и выходов:

- **Цифровые входы:**

- PB11 ← DI (IN_1);
- PB12 ← DI (IN_2);
- PB13 ← DI (IN_3);
- PB14 ← DI (IN_4);
- PB15 ← DI (IN_5);
- PA8 ← DI (IN_6);

- **Цифровые выходы STN32G431:**

- PB7 → DO (OUT_1);
- PB6 → DO (OUT_2);
- PB5 → DO (OUT_3);
- PB4 → DO (OUT_4);
- PB3 → DO (OUT_5);
- PA15 → DO (OUT_6);

Подключение всех частей системы производится на электронной плате с помощью пайки элементов. Также будут применены следующие разъёмы:

- разъёмы для рабочего напряжения – XT30;
- Разъёмы для шины CAN - JST 1.25 PH 2;
- Штыревые разъёмы для остальной периферии - PBS.

Разъёмы XT30 обеспечивают надежное соединение и выдерживают значительные электрические нагрузки без потерь и риска возгорания из-за плохого контакта или перегрева.

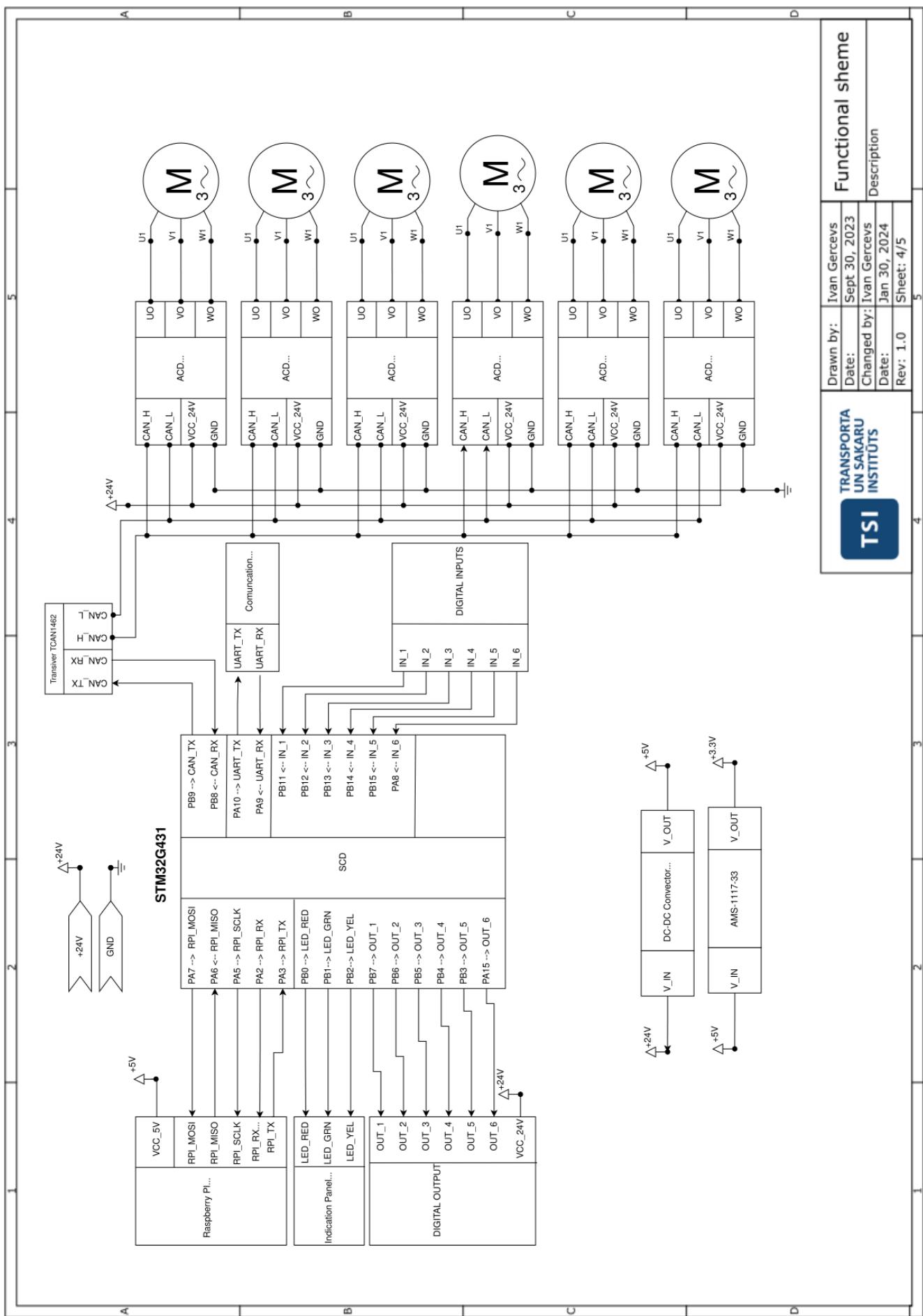


Рис. 3.12. Функциональная схема системы управления мини роботом

Разъемы JST обеспечивают простоту подключения и отключения, что удобно при обслуживании и модернизации системах. Они предотвращают случайное обратное подключение, что может быть критично для предотвращения повреждений компонентов при монтаже.

Штыревые разъемы PBS подходят для подключения компонентов не требующих высоких токов, и при этом могут многократно использованы. Штыревые разъемы обеспечивают хороший контакт и простоту монтажа на печатной плате, что важно для быстрого прототипирования и наладки системы.

3.11. Разработка функциональной схемы системы исполнительного устройства

Функциональная схема показана на рисунке 3.13/

Главным управляющим элементом обработки команд и генерации сигналов, системы исполнительной управления, является МК STM32G431. Генерация сигналов PWM происходит таймером и выводятся 6 каналами сравнения, которые выведены на линии порта А, С и В:

- PB15 → HIN1;
- PB10 → LIN1;
- PA12 → HIN2;
- PB9 → LIN2;
- PC13 → HIN3;
- PA8 → LIN3;

Драйвера усиливают сигнал от МК и подают на силовые ключевые транзисторы:

- HIN1 → HGV1;
- LIN1 → LGV1;
- HIN2 → HGV2;
- LIN2 → LGV2;
- HIN3 → HGV3;
- LIN3 → LGV3;

Для измерения момента приложенного на двигатель, происходит измерения милы тока на каждой обмотке двигателя, измеряется падение напряжения путем подачи сигналов на входы операционных усилителей, которые встроенные в микроконтроллер 6 входов подключение производится:

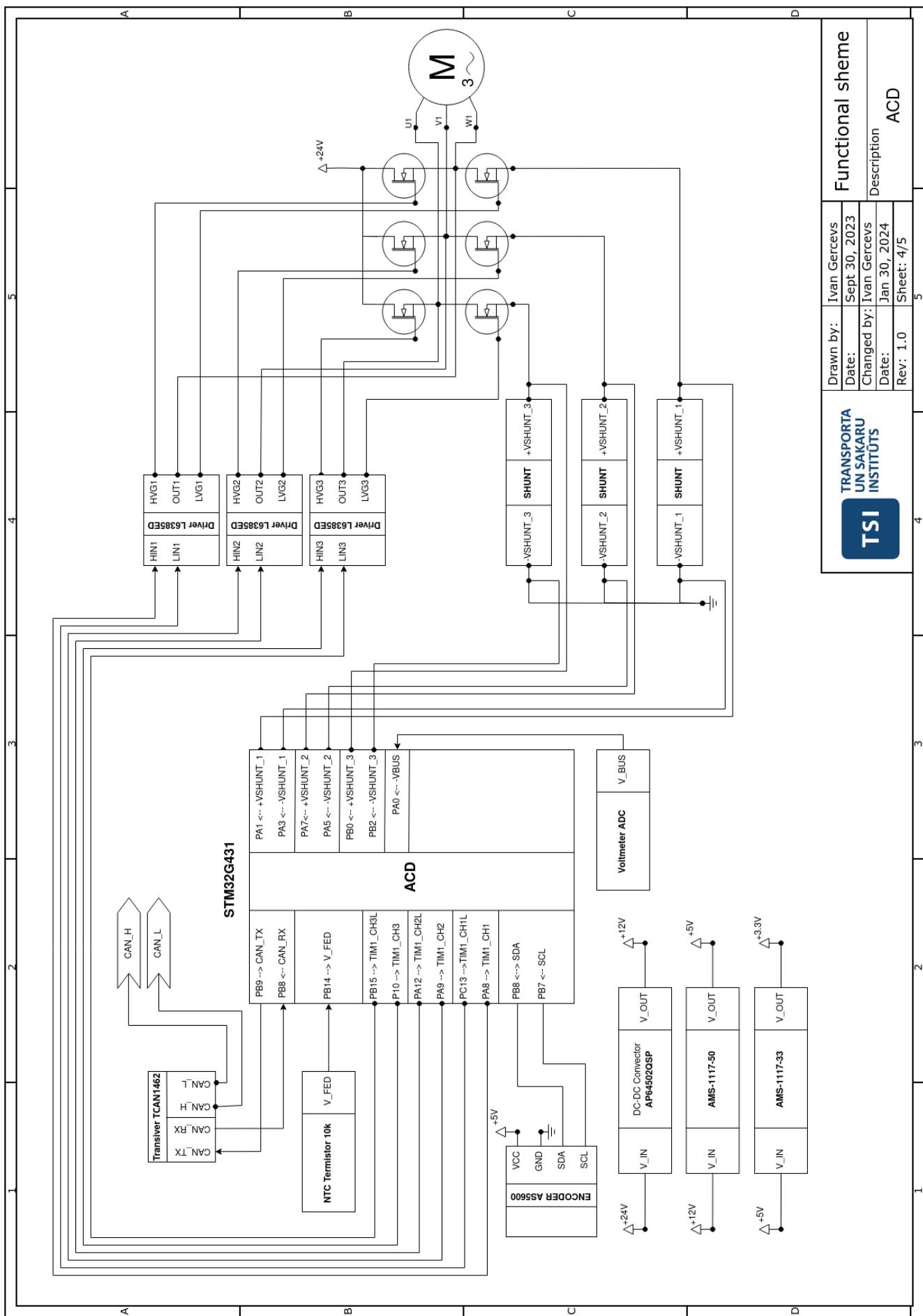


Рис. 3.13. Функциональная схема системы исполнительного управления

- PA1 $\leftarrow +VSHUNT_1;$
- PA3 $\leftarrow -VSHUNT_1;$
- PA7 $\leftarrow -VSHUNT_2;$
- PA5 $\leftarrow +VSHUNT_2;$
- PB0 $\leftarrow -VSHUNT_3;$
- PB2 $\leftarrow +VSHUNT_3;$

Во избежание поломки устройства, предусмотрены две линии порта входа для подключения к АЦП микроконтроллера, один необходим для измерения температуры, датчик будет находиться рядом с силовыми транзисторами, таким образом в случае перегрева, микроконтроллер предпринимает действия по установке в состояние ошибки. Второй АЦП используется для измерения поступающего рабочего напряжения, в случае выхода из диапазона рабочего напряжения, МК переходит в режим ошибки. АЦП расположены на линиях порта А и подключаются:

- PA5 $\rightarrow V_FED;$
- PA0 $\rightarrow V_BUS;$

4 ПРИНЦИПИАЛЬНАЯ СХЕМА СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ МИНИ РОБОТА

4.1. Принципиальная схема устройства исполнительного управления

Принципиальная схема устройства исполнительного управления мини робота манипулятора представлена на рисунке 4.1. Микроконтроллер U9 является главным вычислительным компонентом исполнительной системы управления. Чип STM32G431CBU6 подключен по стандартной схеме, из даташита (STM, 2020a)DC-DC преобразователь является главным источником питания для всего устройства исполнительного управления, а микроконтроллер работает с аналоговым сигналом. Для повышения стабильности работы аналоговых операционных усилителей и точности работы АЦП в схеме реализован фильтр подавления шумов на линии аналогового питания VDDA и линии опорного напряжения VREF+, который собран по заметкам AN5346 (STM, 2019). Микроконтроллер тактируется при помощи кварцевого резонатора X1 и X2, которые подключены по стандартной схеме с конденсаторами C23, C24 для X1 и C18,C19 для кварцевого резонатора X2. Кварцевый резонатор X1 является главным генератором тактирующего HSE сигнала 8 МГц для микроконтроллера U9. Кварцевый резонатор X2 имеет специфичную низкоскоростную частоту LSE 32.768kHz, резонатор необходим для работы часов реального времени RTC, часы используются для точного отсчета времени без накопления ошибки в случае двоичного преобразования. Для программирования микроконтроллера выводы программатора-отладчика выведены на разъем H2.

Для контроля состояния исполнительного устройства предусмотрены два светодиода LED1 и LED2. Светодиод LED1 подключен через токовый резистор к питанию и загорается при поступлении питания, LED2 загорается при необходимости от сигнала МК и является индикатором ошибки работы исполняющего устройства. К входу АЦП PA0 микроконтроллера подключен делитель входного напряжения, состоящий из резисторов R16 и R18, номиналы сопротивления выбраны более 10 kOhm для предотвращения большого падения напряжения на резисторах, максимальное возможное измеряемое напряжение 33 Вольта. Второй вход АЦП PB14 подключен к выводам схемы измерения температуры, которая состоит из NTC резистора R15 и балансного сопротивления R20, конденсатор C30 является элементом фильтра низкой чистоты, верхняя граничная частота 1.5 kHz.

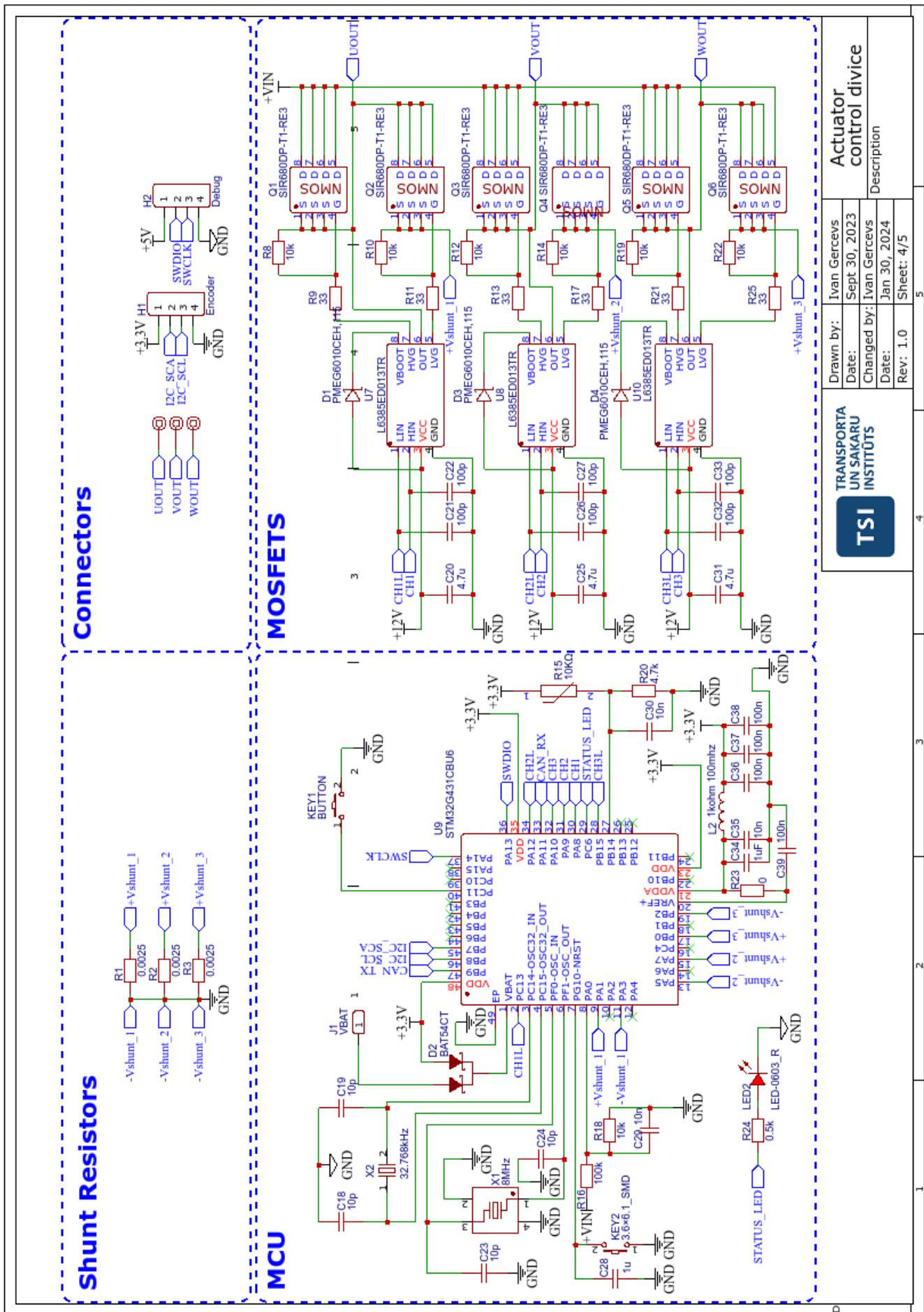


Рис. 4.1. Принципиальная схема системы исполнительного управления

Драйвера инвертора U7, U8, U10 подключены к выходу таймера TIM1, которые располагаются на линиях порта А и С, для подавления высокочастотных наводок, каждому из входов подключены емкости C21, C22, C26, C27, C32, C33 номиналом 100 пикофарад. Конденсаторы C20, C25, C31 на линии питания установлены, для предотвращения резкого падения напряжений при открытии внутренних силовых транзисторов. Диоды D1, D3, D4 образуют схему начальной нагрузки, необходимы для обеспечения питания высоковольтной секции драйвера. Драйверы управляют силовыми транзисторами от Q1 - Q6, полумостовой драйвер управляет транзисторами верхней и нижнего уровня для каждой фазы мотора. Транзисторы подключены через низкоомные резисторы R9, R11, R13, R17, R21, R25 для исключения звона (parasitic oscillation) и резисторами R8, R10, R12, R14, R19, R22, резисторы подтягивают затвор к истоку, для избежания паразитных открытий транзисторов.

Для обеспечения связи по шине данных, линии МК PB9 и PA11 подключены к передатчику CAN ШИНЫ U4, принципиальная схема 4.2. Передатчик не нуждается в дежурном режиме, поэтому вывод STB (восьмой вывод микросхемы) подключен к земле, передатчик работает с момента включения. Между линиями CANH и CANL предусмотрено включение терминального резистора R7, через перемычку Н3. Для удобства монтажа устройств исполнительного управления коннекторы JP2 и JP1 образуют параллельное включение двух разъемов позволяет производить подключений устройств без создания дополнительных соединений на проводах. По такому же принципу подключены силовые разъёмы питания CN1 и CN2.

В данном устройстве реализовано несколько вариантов преобразования напряжений и тока. Основным преобразователем является микросхема DC-DC преобразователя AP64502QSP, которая подключена по стандартной схеме из даташита, принципиальная схема 4.2.

Микросхема преобразует входное напряжение в напряжение 12В, которое необходимое для питания драйверов силовых транзисторов. Для обеспечения преобразования напряжения необходимо было рассчитать делитель обратной связи на вход линии FB (Feedback), который вычисляется (8):

$$R4 = R6 \cdot \left(\frac{V_{out}}{0.8V} - 1 \right) \quad (8)$$

где R2=10 kOhm, для данного варианты значения получились R1=140 kOhm. Конденсатор C6 на линии SS DC-DC преобразователя является задающим времени мягкого старта. Был выбран стандартный вариант 10nF, чему примерно равен временем мягкого старта - 4 ms.

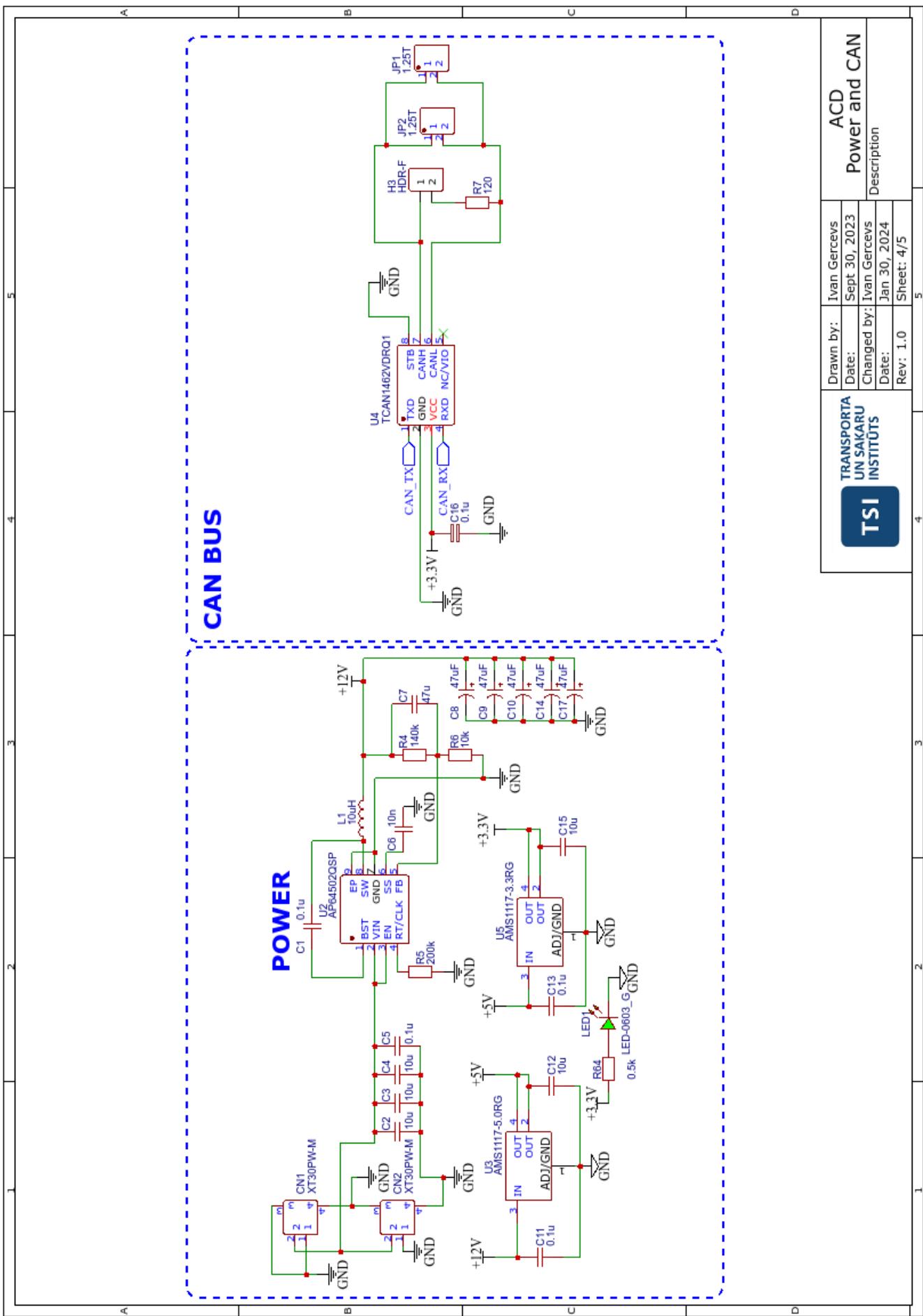


Рис. 4.2. Принципиальная схема элементов питания и шины данных системы исполнительного управления

Индуктивность катушки рассчитывается для данного преобразователя по формуле (9), где Delta I_l при данном подключении рассчитывается по формуле (10):

$$L = \frac{V_{out} \cdot (V_{in} - V_{out})}{V_{in} \cdot \Delta I_l \cdot f_{sw}} \quad (9)$$

$$\Delta I_l = 0.5 \cdot 5A \quad (10)$$

Микросхема имеет плавную регулировку частоты от 100kHz до 2.2MHz, частота задается резистором R5, частота рассчитывается по формуле (R5):

$$R5 = \frac{100000}{f_{sw}[kHz]} \quad (11)$$

Для варианта преобразования выбран самый близкий по параметрам индуктивность $L = 10 \mu H$. В стандартную схему преобразователя внесен дополнительный ряд сглаживающих tantalовых конденсаторов C8, C9, C10, C14, C17. Танталовые конденсаторы имеют маленькое значение внутреннего сопротивления (ESR), который влияет на выходные пульсации (12):

$$V_{outRipple} = \Delta I_l \cdot (ESR + \frac{1}{8 \cdot f_{sw} \cdot C_{cout}}) \quad (12)$$

Суммарная емкость конденсаторов $5 \times 47\mu F$.

Для преобразования напряжения для микроконтроллеров используются линейные преобразователи напряжения U3 и U5, линейные преобразователи подключены к линии питания 12В для уменьшения нагрева самих преобразователей, путем уменьшения падения напряжения на них.

Измерение крутящего момента производится за счет измерения тока электродвигателя. Измеряется падения напряжения на шунтах, шунты R1, R2 и R3 имеют очень малое сопротивление, во избежания влияния передаваемое напряжение и излишнего нагрева. Выводы шунтов подключены ко входам микроконтроллера. После конфигурации микроконтроллера, с линий входа шунтов подключаются ко внутренним операционным усилителям и используются в дальнейших преобразования.

4.2. Принципиальная схема устройства тактического управления

Принципиальная схема системы тактического управления устройства мини робота манипулятора представлена на рисунке 4.3.

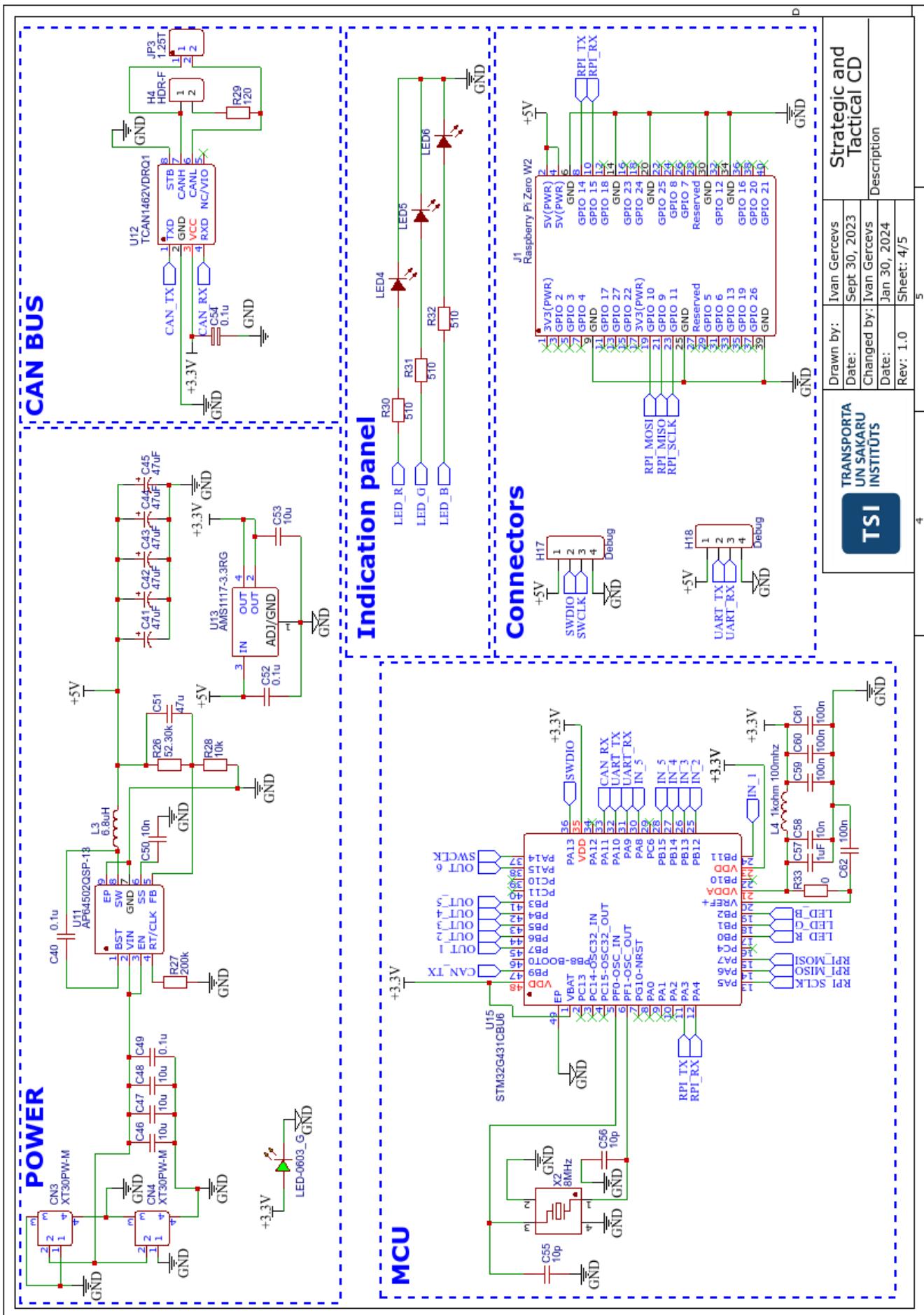


Рис. 4.3. Принципиальная схема системы тактического управления миниробота манипулятора

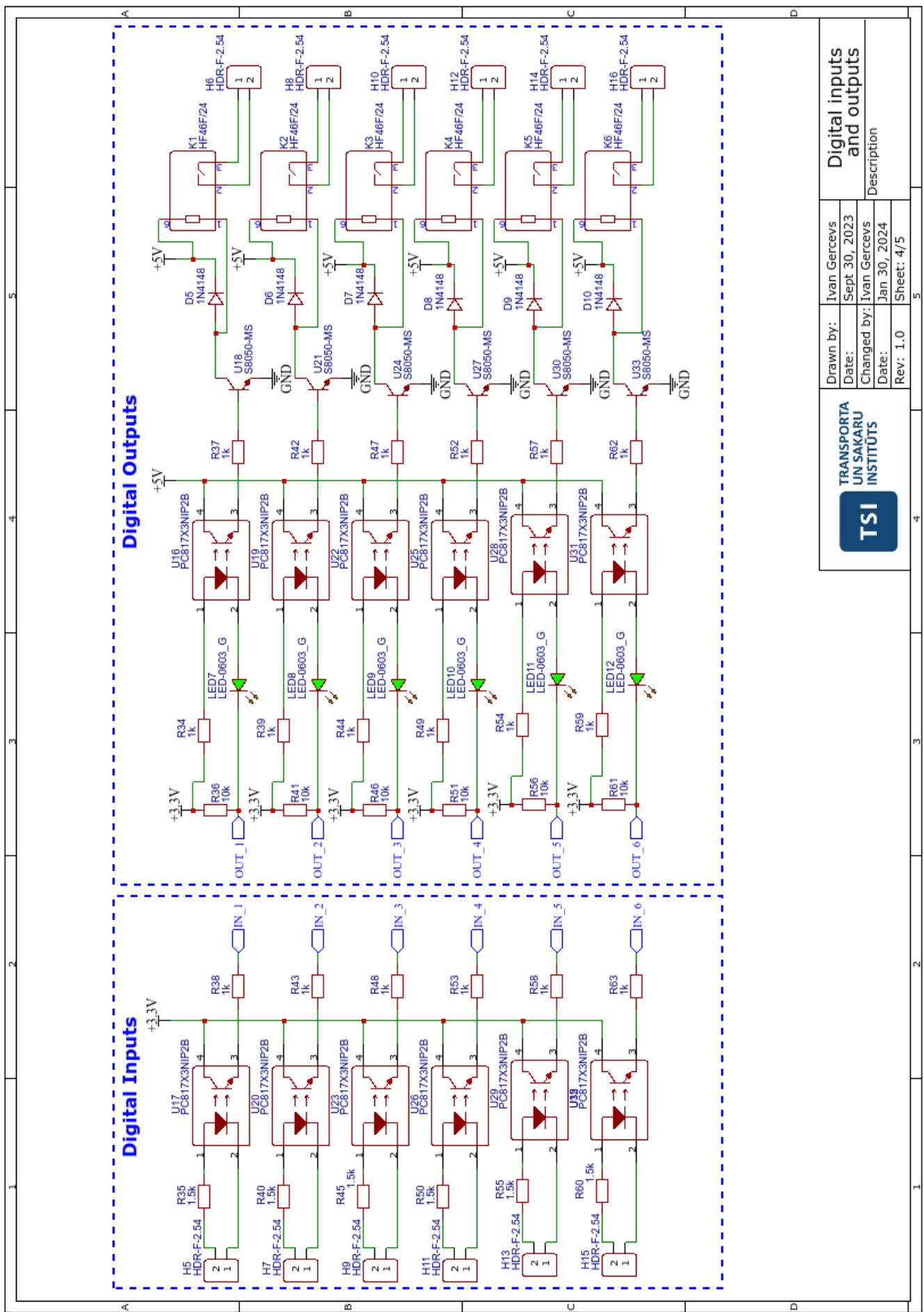


Рис. 4.4. Принципиальная схема входов выводов для системы управления миниробота манипулятора

Микроконтроллер U15 является устройством тактического управления и собран по стандартной схеме, как на рисунке 4.1. Устройство стратегического управления подключается через разъём J1 и используют два варианта интерфейса SPI и UART. Остальные выводы разъёма J1 не используются, так как к ним могут подключаться различные платы расширения для «Raspberry PI ZERO 2», который расширяют функционал одноплатного миникомпьютера.

Панель индикации состоит из трех светодиодов 3 цветов LED4, LED5, LED6 и подключены ко входу МК. Остальные свободные входы и выходы микроконтроллера разделены для управления внешних сигналы входов (6 штук) и выходов (6 штук) и подключены по стандартной схеме из даташита.

Входы и выходы изолированы при помощи оптопар, показаны на рисунке 4.4, выходные сигналы коммутируются при помощи реле K1, K2, K3, K4, K5, K6, сигналы с оптопар U16, U19, U22, U25, U28, U31 усиливаются транзисторами U18, U21, U24, U27, U30, U30. Резистор в цепи базы необходим для ограничения тока поступающего на базу транзистора. Диоды D5, D6, D7, D8, D9, D10 необходимы для шунтирования выводов катушки реле в момент отключения питания. В момент выключения, на выводах катушек образуется импульс ЭДС (электродвижущая сила самоиндукции катушки), и этот импульс может достигать десятки Вольт и может привести к выходу из строя транзисторов U18, U21, U24, U27, U30, U30, которые не рассчитан на такое напряжение.

Входные цифровые сигналы изолируются с помощью U17, U20, U23, U26, U29, U32. Подразумеваются логическая «1» соответствует 24В напряжения. Для обеспечения включения оптопары произведен расчет резисторов R35, R40, R45, R50, R60, с условием, ток на оптодиод внутри оптопары должен соответствовать 20 mA, при напряжении 1.2 Вольта, допустимое максимальное напряжение 30 Вольт, приближенное значение сопротивления резистора, выбрано из ряда номинала сопротивлений E24 и соответствует 1.5 kOhm.

5 АЛГОРИТМ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ КОЛЛАБОРАТИВНЫМ МИНИРОБОТОМ МАНИПУЛЯТОРОМ

5.1. Программная реализация устройства стратегического управления

Устройством стратегического управления является одноплатный миникомпьютер «Raspberry Pi Zero W 2» (Рисунок 3.3), система запускается сразу после подачи питания на устройство. Происходит запуск операционной системы «Linux» и ожидание подключения пользователя к устройству. Возможен два варианта использования стратегического устройства:

1. Подключение по беспроводному каналу, по протоколу SSH.
2. Через проводное подключение, путем подключения проводов монитора и устройств ввода/вывода.

Для подключения мини-компьютера по беспроводному каналу, необходимо чтобы микрокомпьютер был в локальном окружении беспроводной сети.

Для дальнейшей работы управления используется один из языков высокого уровня, например, язык программирования Python с использованием библиотек, пример представлен на рисунке 5.1.

```
import numpy as np
from pybotics.geometry import vector_2_matrix
poses = np.array([
    vector_2_matrix([600, -150, 800,
                    np.deg2rad(-90), 0,
                    np.deg2rad(-90)]),
    vector_2_matrix([700, -150, 800,
                    np.deg2rad(-90), 0,
                    np.deg2rad(-90)])
])
start_end_joints = [robot.ik(p) for p in poses]
```

Рис. 5.1. Фрагмент использования сторонних библиотек на языке питон.

При помощи библиотек возможно производить программирование робота в режиме симуляции и только потом генерировать команды для контроллера тактического управления. Использования языка высокого уровня позволяет создавать комплексные стратегии движения и

обрабатывать сложные алгоритмы. Для коммуникации между устройствами стратегического и тактического уровня предполагается использование написанного библиотеки API, пример функции представлен на Рис. 5.2 для общения между микроконтроллером и микрокомпьютера

```
def send_movej_command(self, command):
    """
    Send a moveJ command to the robot through the serial connection.

    Args:
        serial_connection (serial.Serial): The serial connection to the robot.
        command (str): The moveJ command string.

    """
    with serial.Serial(self.serial_port, self.baud_rate, timeout=1) as ser:
        log("Serial connection established.")
        serial_connection.write(command.encode())
        log(f"Sent command: {command}")
        if self.ser.in_waiting:
            response = self.ser.read(self.ser.in_waiting).decode()
            log("Received response:", response)
```

Рис. 5.2. Фрагмент функции API библиотеки для использования на языке Python.

Так же ещё одним возможным видом программирования является использование программного обеспечение RoboDK, на устройстве стратегического управления исполняется драйвер, который передает данные из сокетного TCP/IP соединения в Uart, список команд рассматривается в таблице 5.1 на контроллер тактического управления фрагмент представлен в приложение 2.

Таблица 5.1
Список базовых команд который поддерживает robotDK

Command	Description	Parameters	Example Usage
MOVJ	Moves the robot joints to specified positions.	Joint Positions (degrees or radians)	MOVJ 10 20 30 40 50 60 70
CJNT	Requests the current joint positions.	None	CJNT
MOVL	Moves the robot to a specified linear position.	Cartesian Coordinates (X, Y, Z, Rx, Ry, Rz)	MOVL 500 400 300 0 1.570
SPEED	Sets the movement speed of the robot.	Speed Value (units per minute)	SPEED 500
WAIT	Pauses the robot operation for a specified duration.	Time (seconds)	WAIT 5
STOP	Stops all robot movements immediately.	None	STOP
DISCONNECT	Terminates the connection with the robot.	None	DISCONNECT

5.2. Алгоритмы устройства исполнительного управления

Главная цель исполнительной системы управления плавное управление 3х фазным BLDC двигателем для этого необходимо считывать, обрабатывать данные с датчиков положения и тока и производить генерацию сигналов PWM. На рисунке 5.3 показан основной алгоритм работы системы исполнительного устройства.

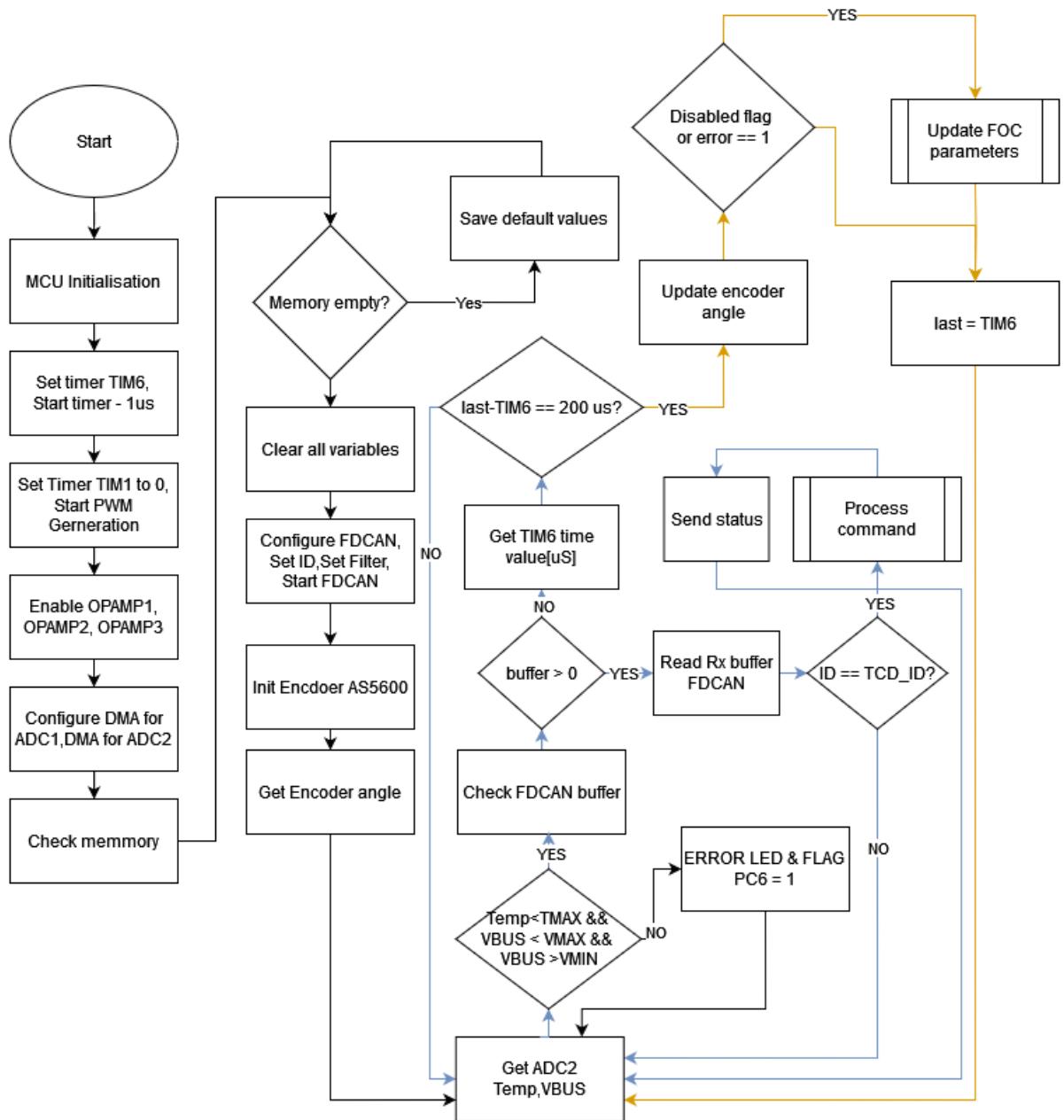


Рис. 5.3. Алгоритм работы системы исполнительной системы.

После включения микроконтроллер проводит проверку всех устройств на работоспособность путем считывания данных. Система разделена на две части, в первой части выполняются функции, которые не требуют частого выполнения (окрашено желтым цветом), вторая выпол-

няется на максимальной большой частоте, с которой возможно выполнение, такими задачами являются вычисление векторного алгоритма и парсинг сообщений Рисунок 5.4.

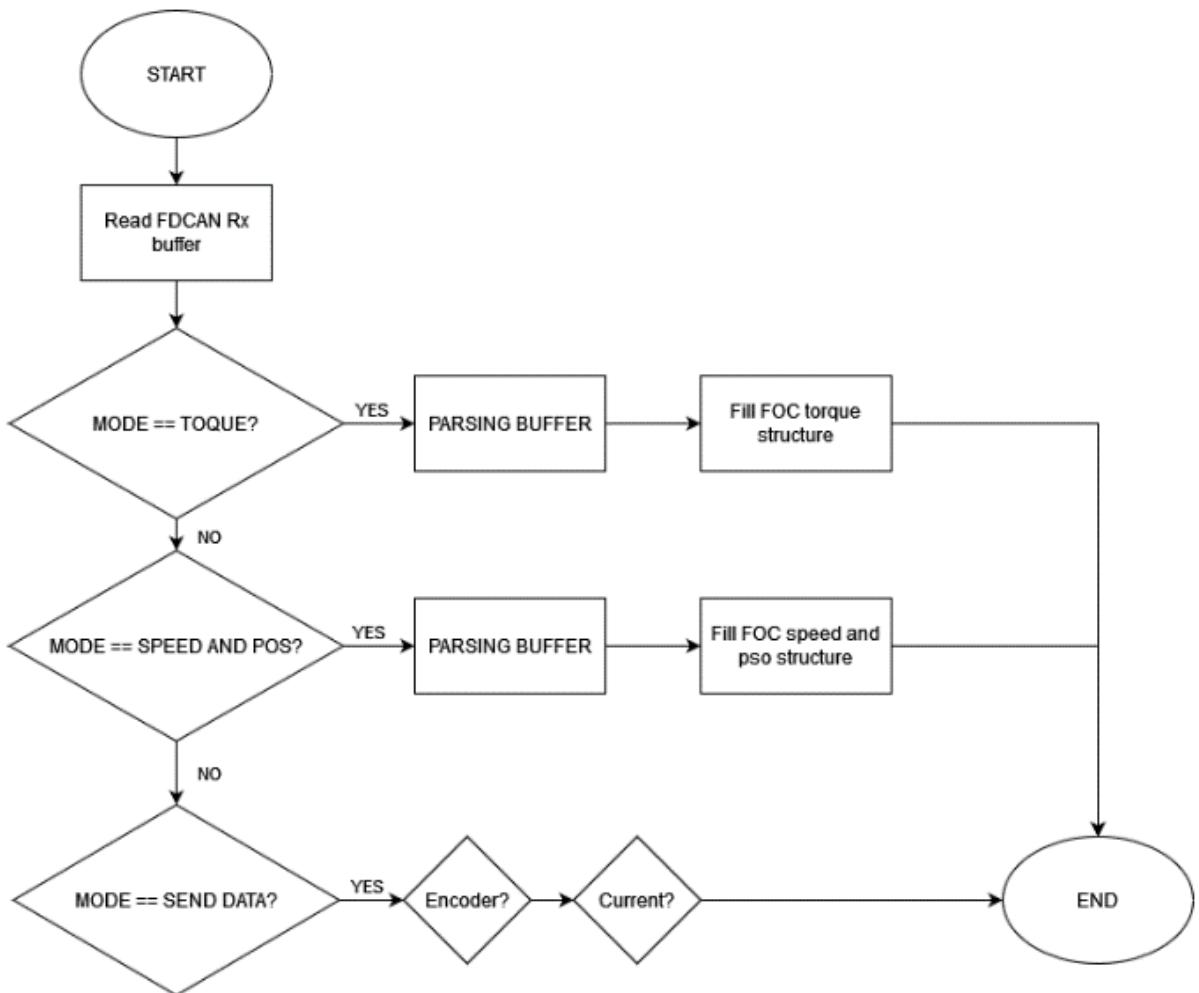


Рис. 5.4. Алгоритм обработки сообщения.

Рисунок 5.8 показывает схему работы векторного регулирования. В качестве примера произведен пример управления по току с обратной связью, что означает, что двигатель всегда создает постоянный крутящий момент (то есть постоянный ток, поскольку крутящий момент пропорционален току).

Входы i_q и i_d регулируются с помощью обратной связи через ПИД-регулятор, который также включает в себя несколько модулей преобразования «Park» и «Clark», сигналы проходят через блок «SVPWM» ?? (Широтно-импульсная модуляция с пространственным вектором) и происходит воздействие на трехфазный инвертор для управления двигателем, а величина обратной связи ПИД-регулятора представляет собой выборочное значение выходного тока двигателя.

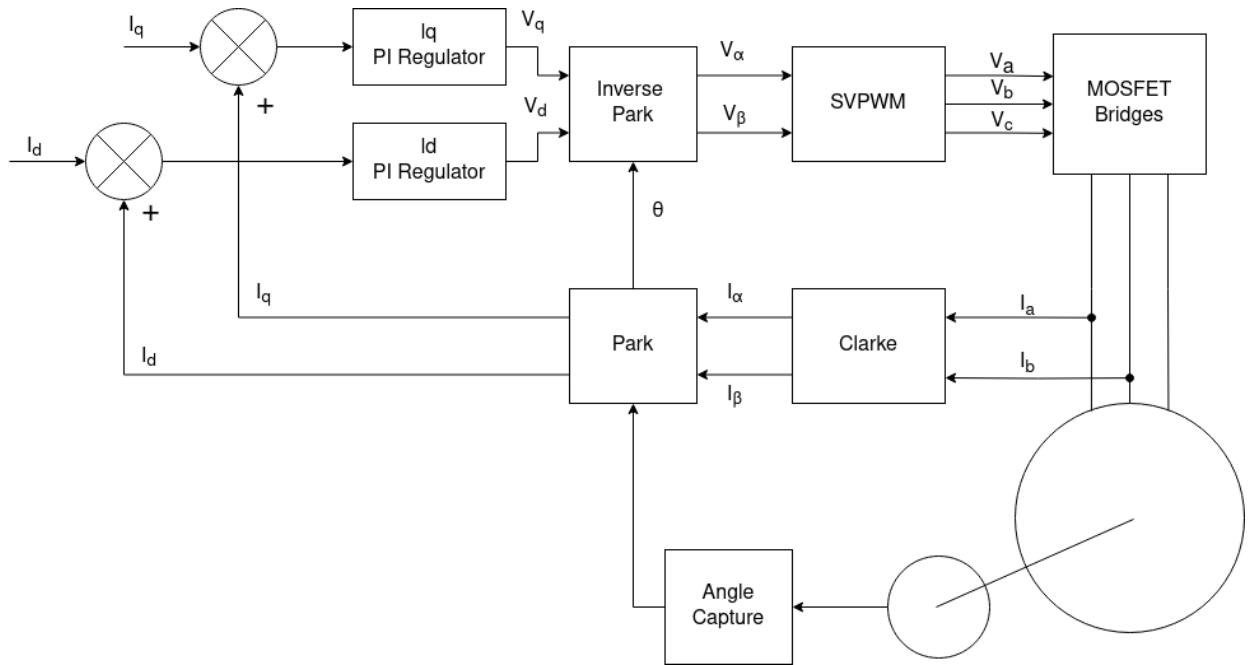


Рис. 5.5. Схема векторного управления двигателем.

Весь процесс векторного управления можно разбить на этапы:

1. Получение данных трехфазного тока двигателя для вычисления значений I_a, I_b ;
2. Преобразование I_a, I_b, I_c методом Clark, получение I_α, I_β ;
3. Преобразование I_α, I_β методом Park, получение I_q, I_d ;
4. Вычисление ошибки I_q, I_d ; из значений установки I_q, I_d поступающих от контроллера;
5. Ведение указанной ошибки в два ПИД-регулятора (используется только ПИ), для получения выходных управляющих напряжений V_q, V_d ;
6. Преобразование V_q, V_d методом обратным превращением Park, получение V_α, V_β ;
7. Использование пространственного вектора V_α, V_β для определения сигналов широтно-импульсной модуляции для трех полумостов инвертора;
8. Управление силовыми MOSFET транзисторами трехфазного инвертора в соответствии с ранее выведенным кодовым значением для привода двигателя;
9. Повторение вышеуказанных шагов;

В вычислениях достаточно использовать только ток с двух фаз мотора, так как по закону тока Кирхгофа можно рассчитать третью составляющую тока, ведь сумма токов, втекающих в узел, равна сумме токов, вытекающих из узла $I_a + I_b + I_c = 0$. Вектора I_a, I_b, I_c изначально не ортогональны, для дальнейшей работы требуется провести пересчет проекций в оси координат,

формула 13.

$$\begin{cases} I_\alpha = I_a - \cos\left(\frac{2\pi}{3}\right) I_b - \cos\left(\frac{2\pi}{3}\right) I_c \\ I_\beta = \sin\left(\frac{2\pi}{3}\right) I_b - \sin\left(\frac{2\pi}{3}\right) I_c \end{cases} \quad (13)$$

После преобразования получается синусоидальная волна, но на одну переменную меньше. Теперь можно использовать значения I_α, I_β для управления вращением двигателя, заставляя их соответствовать правилам изменения формы сигнала. Для применения к трем фазам двигателя применяется обратное преобразование Кларка. Преобразование Park, формула 14, главная задача перевести стационарную систему I_α, I_β в систему координат вращающейся вместе с ротором I_q, I_d , а так как в системе подразумевается получать данные положения угла поворота двигателя с энкодера в реальном времени. Получается, что вектор вращения становится фиксированным значением в системе координат I_α, I_β . Далее использование I_q, I_d в качестве объектов управления как обратную связь через ПИД-регулятор. В реальности используется только ПИ-регулятор, дифференциальный регулятор не вводится, потому как передаточная функция напряжения и тока является инерционным звеном первого порядка.

$$\begin{cases} I_d = I_\alpha \cos(\theta) + I_\beta \sin(\theta) \\ I_q = -I_\alpha \sin(\theta) + I_\beta \cos(\theta) \end{cases} \quad (14)$$

В обычном векторном управлении в основном используются три контура ПИД: контур тока, контур скорости и контур положения, то есть: управление током двигателя (крутящим моментом) посредством обратной связи по току \rightarrow затем управление скоростью двигателя, контролируя крутящий момент \rightarrow затем управление положением двигателя, контролируя скорость двигателя.

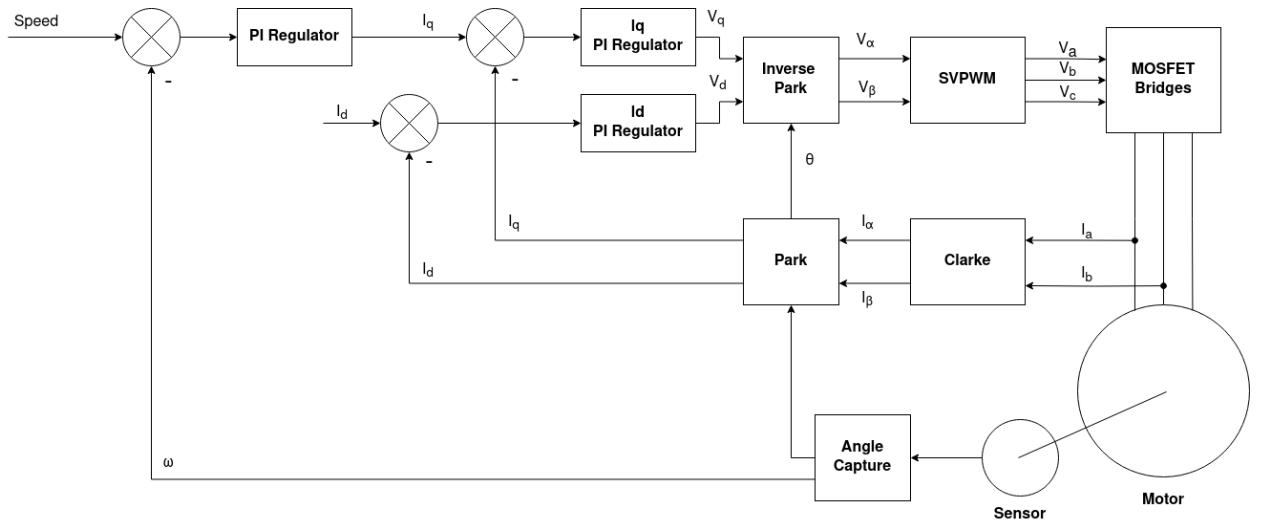


Рис. 5.6. Схема векторного управления двигателем по скорости.

На картинке выше 5.6, ω — это обратная связь по скорости двигателя, которую рассчитывается с помощью энкодера двигателя. Она контролируется PI регулятором.

Расчетная скорость двигателя и значение настройки скорости вычисляет значение ошибки и подставляет его в контур ПИ скорости. Вычисленный результат используется в качестве входного сигнала токового контура, таким образом реализуя двойное регулирование скорости-тока с обратной связью.

Самый внешний уровень — это контур положения, который управляет двигателем, поворачивая его на точный угол и поддерживая его. Блок-схема управления изображено на картинке 5.7.

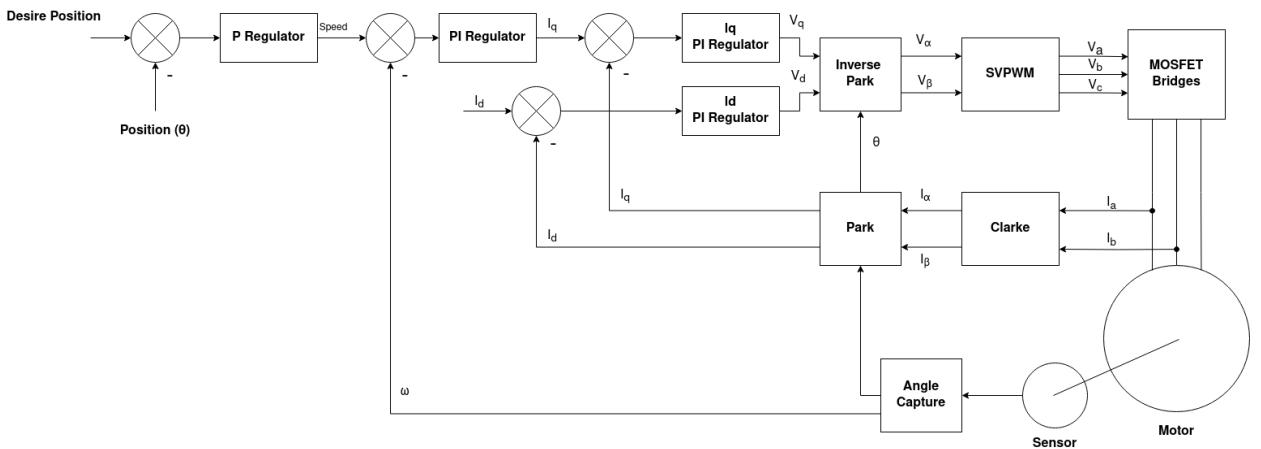


Рис. 5.7. Схема векторного управления двигателем по положению.

Но в реальном использовании алгоритма применительно к системе, энкодер не может напрямую возвращать скорость двигателя. Поэтому скорость двигателя рассчитывается, вычислив изменение значения кодирования в течение определенного периода времени (то есть используя

среднюю скорость для представления мгновенной скорости). Когда скорость двигателя относительно высока, этот метод подходит, но в режиме управления положением скорость двигателя будет очень низкой (поскольку ротор необходимо зафиксировать в определенном положении). Поэтому, чтобы избежать ошибок, вызванных связью скорости, для управления положением используется только двойной контур, состоящий из положения и тока. Однако в это время необходимо внести определенные изменения в контур положения Рисунок 5.8.

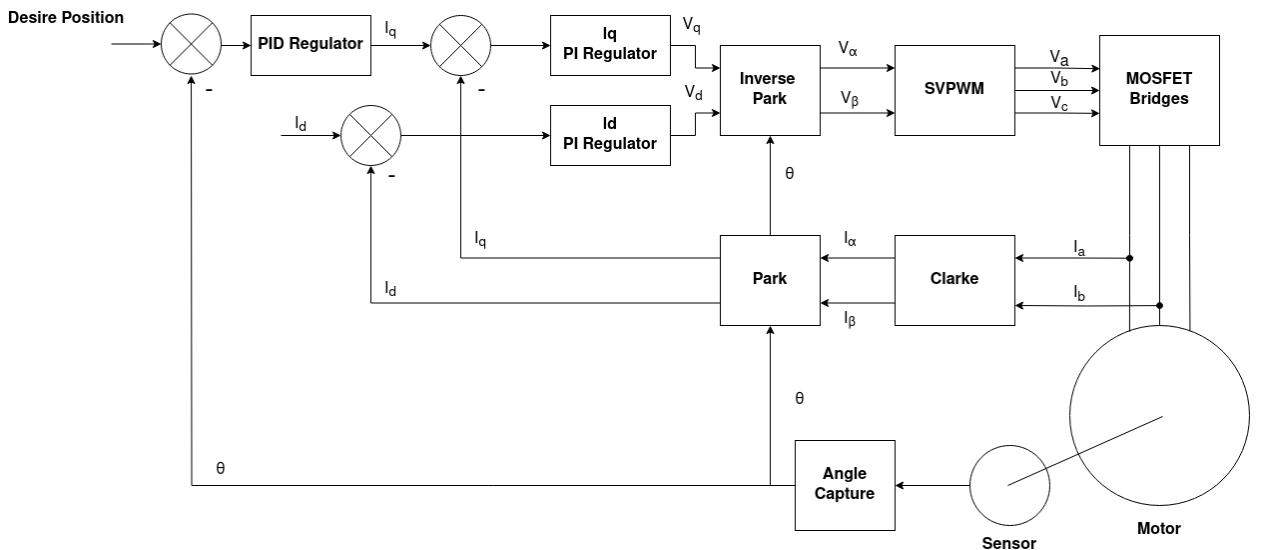


Рис. 5.8. Схема оптимального векторного управления двигателем по положению.

Но так же учитывается, что контур скорости удален, используется полное ПИД-управление для контура положения, то есть добавляем дифференциальный член (поскольку дифференциал положения — это скорость, это может уменьшить колебания регулирования положения и ускорить сходимость, функция интегрального члена заключается в устраниении статической ошибки).

Для преобразования непосредственно в сигналы подаваемые на силовые транзисторы используется метод SVPWM (Space Vector Pulse Width Modulation), так он является наиболее эффективен, чем SPWM (Mirdas et al., 2023).

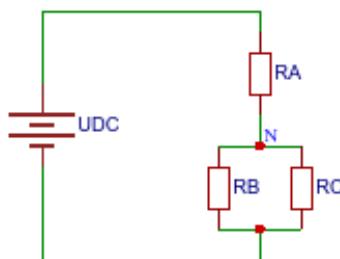


Рис. 5.9. Эквивалентная схема.

Для понимания процесса формирования сигналов необходимо понимание вектора космического напряжения. Можно представить подключение двигателя к источнику напряжения как Рис.5.9.

Трехфазные напряжения (фазное напряжение — это напряжение каждой фазы относительно средней точки подключения двигателя) выражается по формуле 15:

$$\begin{cases} U_a = U_A - U_N = \frac{2}{3}U_{dc} \\ U_b = U_B - U_N = -\frac{1}{3}U_{dc} \\ U_c = U_C - U_N = -\frac{1}{3}U_{dc} \end{cases} \quad (15)$$

Можно представить, что три напряжения формируют векторы $\vec{U}_a, \vec{U}_b, \vec{U}_c$, которые формируют результирующий вектор \vec{U} , получается возможно определить вектор магнитного поля. А так как постоянный магнит ротора будет стремиться вращаться до тех пор, пока линии внутреннего магнитного поля не будут соответствовать направлению внешнего магнитного поля, этот вектор может фактически представлять направление, в котором мы хотим, чтобы ротор вращался. Получается возможно рассчитать вектор пространственного напряжения 16 (Instruments, 2013).

$$\begin{cases} U_A(t) = U_{dc} \cos(2\pi ft) \\ U_B(t) = U_{dc} \cos(2\pi ft - \frac{2\pi}{3}) \\ U_C(t) = U_{dc} \cos(2\pi ft + \frac{2\pi}{3}) \end{cases} \quad (16)$$

Метод SVPWM позволяет синтезировать значение SPWM в каждом моменте времени 17.

$$\int_0^T U_{\text{ref}} dt = \int_0^{T_x} U_x dt + \int_{T_x}^{T_x+T_y} U_y dt + \int_{T_x+T_y}^T U_0^* dt \quad (17)$$

$$U_{\text{reference}} \cdot T = U_x \cdot T_x + U_y \cdot T_y + U_0^* \cdot T_0^* \quad (18)$$

Так программа в микроконтроллер использует дискретизацию, может упростить до 18, где $U_{\text{reference}}$ — ожидаемый нами вектор напряжения, а T — период ШИМ. Смысл приведенной выше формулы заключается в периодическом переключении между различными векторами пространственного напряжения, в котором можно синтезировать эквивалентный произвольный вектор пространственного напряжения.

5.3. Программная реализация устройства исполнительного управления

Программа для микроконтроллера разрабатывалась на языке C/C++, для облегчения и ускорения разработки использовалась библиотека HAL (Hardware Abstraction Layer). Набор библиотек позволяет пройти на следующий уровень абстракции и позволяет производить перенос программного кода на микроконтроллеры другого семейства. Для генерации и настройки проекта использовался инструмент графической конфигурации STM32CubeMX. Генерация проекта производилась под среду интегрированной разработки STM32CubeIDE, которая является измененной средой разработки Eclipse для встроенных систем. На рисунке 5.10 представлено графическое отображение настройки тактирования тактовой частоты и частоты тактирования отдельных периферийных блоков. МК контроллер работает на максимальной частоте с учетом использования внешнего кварцевого резонатора на 8 MHz. Входная частота поступает на блок фазовой автоподстройки частоты (PLL), далее происходит умножение и деление частоты до выходной частоты, которая достигает уровня 94

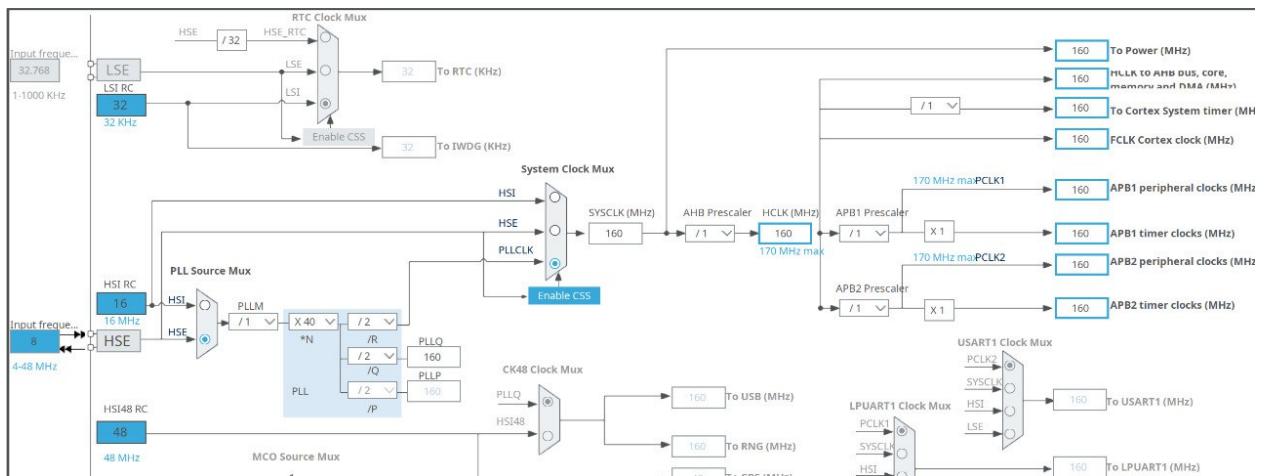


Рис. 5.10. Графическое отображение тактирования микроконтроллера STM32G431

Энкодер устанавливается на роторе двигателя совместно с редуктором, необходимо было решить проблему сохранения угла поворота. Для этого был создан алгоритм сохранения угла поворота, при варианте поворота энкодера от 0 до 360 градусов. В приложении 3 реализован тест на языке Python алгоритма угла поворота энкодера. Для обеспечения считывания данных о положении ротора микроконтроллер отправляет запрос данных у энкодера AS5600, после чего функция заполняет структуру данных, фрагмент функции показан на картинке 5.11.

```

angle_t data;
data.raw = 0;

if(HAL_I2C_Mem_Read(a->i2cHandle,a->adress,0x0E,I2C_MEMADD_SIZE_8BIT,
→ (uint8_t*)&data.raw,2,2)!= HAL_OK)

{
    status = HAL_ERROR;
}

*angle = ((data.bit.angleHIGH << 8) & 0xF00) | data.bit.angleLOW;
}

```

Рис. 5.11. Фрагмент кода функции считывания данных с энкодера.

Энкодер устанавливается непосредственно на валу электродвигателя, вращение двигателя превосходят границы измеряемых параметров абсолютного энкодера AS5600 от 0 до 360 градусов (так как используется редуктор), поэтому возникла необходимость измерения угла поворота в более широком диапазоне, функция измеряет значения текущего положения, скорости и ускорения представлен в приложении 4.

5.4. Программная реализация устройства тактического управления

В процессе различных вычислений необходимо операции, которые выходят за рамки стандартных библиотек C++. При вычислении используются часто перемножение матриц, было реализована функция перемножения двух матриц, которая показана на Рис. 5.12. Функция для умножения матриц реализована так, что она принимает на вход три указателя на массивы, представляющие две входные матрицы и одну выходную матрицу, а также три целочисленных значения, указывающих размеры этих матриц. Входные матрицы обозначены как первая и вторая, а результат их умножения записывается в выходную матрицу.

Для выполнения умножения матриц, функция задействует три вложенных цикла, где каждый цикл отвечает за свою часть процесса: внешний цикл перебирает строки первой матрицы, средний цикл - столбцы второй матрицы, а внутренний цикл занимается элементами текущих строк и столбцов, участвующих в вычислении. Это позволяет последовательно вычислить значения для каждой ячейки результирующей матрицы.

В процессе умножения для каждой пары строки и столбца входных матриц вычисляется скалярное произведение соответствующих векторов. Для этого элементы строки первой матрицы поочерёдно умножаются на соответствующие элементы столбца второй матрицы, а резуль-

таты сложения этих произведений формируют значение в ячейке результирующей матрицы.

Таким образом, каждый элемент выходной матрицы является суммой произведений элементов соответствующих строк и столбца входных матриц, что и обеспечивает результат умножения матриц.

```
void MultiplyMatrices(const float* matA, const float* matB, float* resultMat, int rowsA, int
← commonDim, int colsB) {
    float sum;
    int rowIdx, colIdx, kIdx;
    // Loop through each row of the first matrix
    for (rowIdx = 0; rowIdx < rowsA; rowIdx++) {
        // Loop through each column of the second matrix
        for (colIdx = 0; colIdx < colsB; colIdx++) {
            sum = 0.0f; // Temporary variable to store sum
            // Multiply elements across the common dimension
            for (kIdx = 0; kIdx < commonDim; kIdx++) {
                sum += matA[commonDim * rowIdx + kIdx] * matB[colsB * kIdx + colIdx];
            }
            // Assign the computed sum to the result matrix
            resultMat[colsB * rowIdx + colIdx] = sum;
        }
    }
}
```

Рис. 5.12. Функция для умножения двух матриц

Функция для преобразования матрицы поворота в углы Эйлера, рисунок 5.13, принимает два аргумента: первый аргумент — это указатель на входную матрицу поворота, второй аргумент — указатель на массив, в который будут записаны рассчитанные углы Эйлера.

В начале функции объявляются переменные для трех углов Эйлера: рыскание (yaw), тангаж (pitch) и крен (roll), а также переменная для косинуса тангажа. Эти переменные используются для хранения промежуточных вычислений и итоговых результатов преобразования.

Далее следует проверка на условие, когда матрица поворота находится в состоянии, известном как "гимбал лок". Это состояние возникает, когда одна из осей вращения становится неопределенной из-за параллельности двух других осей. Условие гимбал лока определяется

значением элемента матрицы поворота, и если это значение близко к 1 или -1, выполняется специальная обработка для расчета углов Эйлера.

```
void ConvertRotationMatrixToEulerAngles(const float* rotationMatrix, float* angles){
    float roll, pitch, yaw, cosPitch;

    // Check for gimbal lock

    if (fabs(rotationMatrix[6]) >= 1.0 - 0.0001){

        // Handle the gimbal lock case

        roll = 0.0f; // Set roll to zero as it's indeterminate

        if (rotationMatrix[6] < 0) {

            pitch = (float) M_PI_2; // 90 degrees

            yaw = atan2f(rotationMatrix[1], rotationMatrix[4]);

        }

        else{

            pitch = -(float) M_PI_2; // -90 degrees

            yaw = -atan2f(rotationMatrix[1], rotationMatrix[4]);

        }

    } else{

        // Regular case, no gimbal lock

        pitch = atan2f(-rotationMatrix[6], sqrtf(rotationMatrix[0] * rotationMatrix[0] +
            ↵ rotationMatrix[3] * rotationMatrix[3]));

        cosPitch = cosf(pitch);

        roll = atan2f(rotationMatrix[3] / cosPitch, rotationMatrix[0] / cosPitch);

        yaw = atan2f(rotationMatrix[7] / cosPitch, rotationMatrix[8] / cosPitch);

    }

    // Assign calculated angles to output

    angles[0] = yaw;

    angles[1] = pitch;

    angles[2] = roll;
}
```

Рис. 5.13. Функция преобразования матрицы в углы Эйлера

Если элемент матрицы поворота меньше нуля, угол тангажа устанавливается в 90 градусов, а угол крена — в ноль. Угол рыскания вычисляется с помощью функции арктангенса от двух других элементов матрицы. В случае, когда элемент матрицы не меньше нуля, тангаж устанавливается в -90 градусов, крен — в ноль, а рыскание вычисляется как отрицательное значение

угла, полученного с помощью функции арктангенса.

Если гимбал лок не обнаружен, тангаж вычисляется через арктангенс отрицательного значения соответствующего элемента матрицы и квадратного корня из суммы квадратов других двух элементов. Крен и рыскание затем вычисляются с использованием косинуса рассчитанного тангажа и функции арктангенса для соответствующих элементов матрицы.

В конце, вычисленные значения углов рыскания, тангажа и крена сохраняются в выходном массиве в указанном порядке, завершая преобразование матрицы поворота в углы Эйлера. Для обратного преобразования используется функция ConvertEulerAnglesToRotationMatrix.

```
void ConvertEulerAnglesToRotationMatrix(const float* angles, float* rotationMatrix)
{
    float cosRoll, cosPitch, cosYaw, sinRoll, sinPitch, sinYaw;
    // Calculate sine and cosine of angles for efficiency
    cosYaw = arm_cos_f32(angles[0]);
    cosPitch = arm_cos_f32(angles[1]);
    cosRoll = arm_cos_f32(angles[2]);
    sinYaw = arm_sin_f32(angles[0]);
    sinPitch = arm_sin_f32(angles[1]);
    sinRoll = arm_sin_f32(angles[2]);

    // Populate the rotation matrix using the sine and cosine values
    rotationMatrix[0] = cosRoll * cosPitch;
    rotationMatrix[1] = cosRoll * sinPitch * sinYaw - sinRoll * cosYaw;
    rotationMatrix[2] = cosRoll * sinPitch * cosYaw + sinRoll * sinYaw;
    rotationMatrix[3] = sinRoll * cosPitch;
    rotationMatrix[4] = sinRoll * sinPitch * sinYaw + cosRoll * cosYaw;
    rotationMatrix[5] = sinRoll * sinPitch * cosYaw - cosRoll * sinYaw;
    rotationMatrix[6] = -sinPitch;
    rotationMatrix[7] = cosPitch * sinYaw;
    rotationMatrix[8] = cosPitch * cosYaw;
}
```

Рис. 5.14. Функция преобразования углов Эйлера в матрицу

Функция для преобразования углов Эйлера в матрицу поворота принимает два указателя

на массивы, содержащий углы Эйлера (рыскание, тангаж и крен) и массив для результирующей матрицы поворота. В начале функции инициализируются переменные для косинусов и синусов каждого из углов Эйлера. Эти значения вычисляются для упрощения последующих расчетов, так как они многократно используются в формулах преобразования.

Далее, с использованием этих предварительно вычисленных значений косинусов и синусов, функция заполняет элементы матрицы поворота. Матрица поворота, которая является результатом выполнения функции, представляет собой ортогональную матрицу, описывающую вращение в трехмерном пространстве.

Главная задача тактического устройства расчет траектории робота, для того что бы это обеспечить были написаны функции для решения прямой или обратной кинематической задачи для 6 осевого мини робота, код представлен в **приложении 6**.

Функция "CalculateFK"" предназначена для определения положения и ориентации конечного исполнительного органа робота с шестью степенями свободы на основе заданных углов поворота его звеньев. Затем для каждого звена рассчитывается матрица поворота на основе его угла поворота и параметров, заданных в таблице Денавита-Хартенберга. Эти параметры описывают взаимное расположение звеньев робота. Происходит последовательное умножение матрицы поворота всех звеньев начиная от базы робота и до его конечного исполнительного органа. Это умножение дает общую матрицу поворота, которая описывает полную ориентацию и положение конечного исполнительного органа относительно базы робота. Суммируя векторы положений всех звеньев, функция находит конечное положение исполнительного органа робота. На заключительном этапе, на основе общей матрицы поворота, функция рассчитывает углы Эйлера, которые описывают ориентацию конечного исполнительного органа в пространстве. Эти углы представляют собой вращение вокруг трех осей и позволяют точно определить ориентацию исполнительного органа. Результаты расчетов — координаты положения исполнительного органа и его ориентация записываются в выходную структуру outputPose.

Самая сложная функция является решения обратной задачи кинематики, которая основываясь на матрице поворота, происходит вычисление три угла: рыскание, тангаж и крен. Для рыскания и тангажа вычисления основываются на поворотах вокруг вертикальной и горизонтальной осей соответственно, тогда как крен определяется через поворот вокруг оси взгляда. В процессе вычислений учитывается физические ограничения суставов, чтобы обеспечить реалистичность и выполнимость полученных углов. Результаты, представленные в виде углов рыскания, тангажа и крена, затем записываются в предназначенный для этого массив, готовые к использованию для управления движениями робота. Возвращаемое функцией логическое значение служит индикатором успешности операции: true свидетельствует о том, что углы были

успешно рассчитаны и находятся в пределах допустимых значений, в то время как false указывает на невозможность достижения требуемой ориентации из-за ограничений, наложенных конструкцией суставов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы были проведен анализ мини роботов, а так же рассмотрены характеристики мини роботов манипуляторов и их недостатки. Описана проблематика использования роботов совместно с людьми, необходимости особенностям создания короботов. Рассмотрены варианта двигателя, выбор и варианты управления BLDC мотором, а также вариант использования управление двигателями, был выбран метод векторного регулирования с обратной связью.

Разработаны функции и технические характеристики для разрабатываемой системы управления для колаборативного мини робота манипулятора. Была рассмотрена и принята за основу иерархическая парадигма систем управления роботами для разделения принципиальных схем систему управления для мини робота, а также, были описаны взаимодействие систем тактического, стратегического и исполнительного управления. Алгоритмы разработаны основываясь на описанных иерархическим уровням архитектуры:

- Алгоритм работы системы тактического управления;
- Алгоритм работы системы исполнительного управления.

Была разработана функциональная схема устройства в соответствии с выбранными элементами.

- Gimbal GBM4008H-150T, GM5208-120T, GM3506;
- Абсолютного Энкодера AS5600;
- Устройство стратегического контроля Raspberry Pi Zero W 2;
- Силовые ключи SIR680DP;
- Драйвера для силовых транзисторов L6385ED;
- Передатчик шины данных TCAN1462DRQ1;
- DC-DC преобразователь AP64502QSP.

Были рассмотренные какие математические операции для решения задач связи с этим было проведено тестирование и анализ доступных микроконтроллеров под задачи векторного регулирования и кинематических задач. Разработаны две принципиальные схемы устройств, с необходимыми расчетами элементов для систем тактической и стратегических управления и для систем исполнительного управления.

- Разработан алгоритм системы управления мини роботом манипулятором;
- Алгоритм управления системы тактического управления мини роботом;
- Алгоритм управления системы исполнительного управления мини роботом.

Таким образом можно сказать, что прошла успешно демонстрация возможностей использования двигателя типа от карданных камер (gimbal motors) для создания системы управления кол-

лаборативного мини робота манипулятора с учетом упрощения их механической конструкции. Данная тема остается актуальная, появляться компоненты с похожими характеристиками, но с меньшим размером.

СПИСОК ЛИТЕРАТУРЫ

1. Altan, A. and Hacıoğlu, R. (2020), ‘Model predictive control of three-axis gimbal system mounted on uav for real-time target tracking under external disturbances’, *Mechanical Systems and Signal Processing* **138**, 106548. Available at <http://dx.doi.org/10.1016/j.ymssp.2019.106548>.
2. ams (2023), ‘ams.com’. Available at https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf/649ee61c-8f9a-20df-9e10-43173a3eb323.
3. Bélanger-Barrette, M. (2015), ‘What Does Collaborative Robot Mean? — blog.robotiq.com’. Available at <https://blog.robotiq.com/what-does-collaborative-robot-mean>.
4. DIODES (2021), ‘Datasheet AP64502Q’. Available at https://lv.mouser.com/datasheet/2/115/DIOD_S_A0011818538_1-2543578.pdf.
5. Huang, G.-S., Tung, C.-K., Lin, H.-C. and Hsiao, S.-H. (2011), Inverse kinematics analysis trajectory planning for a robot arm, in ‘2011 8th Asian Control Conference (ASCC)’, pp. 965–970.
6. Instruments, T. (2013), ‘Field Oriented Control of Permanent Magnet Motors — youtu.be’. Available at <https://youtu.be/cdiZUszYLiA>.
7. Khatib, O., Quinlan, S. and Williams, D. (1997), ‘Robot planning and control’, *Robotics and Autonomous Systems* **21**(3), 249–261. Available at [http://dx.doi.org/10.1016/S0921-8890\(96\)00078-4](http://dx.doi.org/10.1016/S0921-8890(96)00078-4).
8. Li, J., Guan, Y., Chen, H., Wang, B., Zhang, T., Hong, J. and Wang, D. (2022), ‘Real-time normal contact force control for robotic surface processing of workpieces without a priori geometric model’, *The International Journal of Advanced Manufacturing Technology* **119**(3–4), 2537–2551. Available at <http://dx.doi.org/10.1007/s00170-021-07497-2>.
9. Matsuki, H., Nagano, K. and Fujimoto, Y. (2019), ‘Bilateral drive gear—a highly backdrivable reduction gearbox for robotic actuators’, *IEEE/ASME Transactions on Mechatronics* **24**(6), 2661–2673.
10. Megalingam, R. K., Sahajan, A., Rajendraprasad, A., Manoharan, S. K. and Reddy, C. P. K. (2021), Ros based six-dof robotic arm control through can bus interface, in ‘2021

- 5th International Conference on Intelligent Computing and Control Systems (ICICCS)', IEEE. Available at <http://dx.doi.org/10.1109/ICICCS51141.2021.9432341>.
11. Mirdas, Q. H., Yasin, N. M. and Alshamaa, N. K. (2023), Analytical comparison of spwm amp; svpwm techniques for three-phase induction motor v/f speed control, in 'AIP Conference Proceedings', AIP Publishing. Available at <http://dx.doi.org/10.1063/5.0154520>.
 12. Pack, D. and Barrett, S. (2008), *Microcontroller Theory and Applications: HC12 and S12*, Pearson Prentice Hall. Available at <https://books.google.lv/books?id=klZUPxiGUJ4C>.
 13. Sakama, S., Tanaka, Y. and Kamimura, A. (2022), 'Characteristics of hydraulic and electric servo motors', *Actuators* **11**(1), 11. Available at <http://dx.doi.org/10.3390/act11010011>.
 14. sciencedirect (2023), 'Robust design of independent joint control of industrial robots with secondary encoders — sciencedirect.com'. Available at <https://www.sciencedirect.com/science/article/abs/pii/S0736584521001149>.
 15. Sensinger, J. W. and Lipsey, J. H. (2012), Cycloid vs. harmonic drives for use in high ratio, single stage robotic transmissions, in '2012 IEEE International Conference on Robotics and Automation', IEEE. Available at <http://dx.doi.org/10.1109/ICRA.2012.6224739>.
 16. simplefoc (2023), 'BLDC motors'. Available at https://docs.simplefoc.com/bldc_motors.
 17. STM (2020a), 'Datasheet STM32G431'. Available at <https://www.st.com/resource/en/datasheet/stm32g431rb.pdf>.
 18. stm (2020b), 'FDCAN peripherals for STM32 product'. Available at https://www.st.com/resource/en/application_note/an5348-introduction-to-fdcan-peripherals-for-stm32-product-classes-stmicroelectronics.pdf.
 19. STM (2023), 'How to use the CORDIC to perform mathematical functions on STM32 MCUs'. Available at https://www.st.com/resource/en/application_note/an5325-how-to-use-the-cordic-to-perform-mathematical-functions-on-stm32-mcus-stmicroelectronics.pdf.

ПРИЛОЖЕНИЯ

Приложение 1.

Таблица сравнения результатов тысяч операций в секунду на 1 Мгц у каждого МК

MCU	a + b	a - b	a * b	a / b	sin(a)	log(a)	sqrt(b)	pow(b, a)
ATmega328p	6,952	7,205	6,251	2,015	0,598	0,403	2,017	0,190
ESP32	46,212	54,007	48,121	16,325	8,568	3,986	11,006	1,484
ESP8266	14,412	13,861	8,852	3,497	0,485	0,239	0,787	0,082
RP2040 [Arduino IDE]	7,815	7,226	5,489	2,022	0,455	0,273	2,103	0,122
RP2040 [C++ SDK]	10,132	9,699	11,717	9,266	1,625	1,139	10,719	0,435
RP2040 [MicroPython]	0,629	0,707	0,723	0,681	0,441	0,400	0,568	0,215
STM32G431	101,054	98,251	114,565	43,435	13,254	6,178	44,849	1,118
STM32G431 [FPU-OFF]	15,008	14,425	21,932	6,216	1,391	0,607	2,939	0,176

Фрагмент кода драйвера передачи команд TCP программы RobotDK

```
import socket
import serial
import sys

TCP_IP = '127.0.0.1' # Listen шз
TCP_PORT = 5005 # port

# UART settings
UART_PORT = '/dev/serial0' # UART1 port on Rasp
BAUD_RATE =
# ~~~~~
# Setup socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

print(f'Listening for TCP connections on {TCP_IP}:{TCP_PORT}')

conn, addr = s.accept()
try:
    while True:
        data = conn.recv(1024) # Buffer = 1024
        if not data:
            break
        ser.write(data) # Write data to UART
```

Программный код python моделирования преобразования угла поворота энкодера в суммарный угол

```

# Initialize variables

last_encoder_value = 0
full_rotation_count = 0
total_angle_degrees = 0
cpr = 4096 # Counts per revolution for the encoder

def update_angle(encoder_value):
    global last_encoder_value, full_rotation_count, total_angle_degrees, cpr
    # Calculate the change in encoder value
    d_value = encoder_value - last_encoder_value
    if d_value > cpr / 2:
        full_rotation_count -= 1
    elif d_value < -cpr / 2:
        full_rotation_count += 1

    # Update the total angle in degrees, considering multiple rotations
    total_angle_degrees = (full_rotation_count * 360) + (encoder_value / cpr) * 360
    # Update last encoder value for the next call
    last_encoder_value = encoder_value
    return total_angle_degrees

# This loop simulates the encoder values changing
for i in range(10000):
    encoder_value = i % cpr # Simulate encoder value
    angle = update_angle(encoder_value)
    print(f"Total Angle: {angle} degrees")

```

Фрагмент кода функции обработки данных с энкодера.

```

float d_time;float d_angle;
uint16_t angle_data;
float resolution = 4096;
uint16_t now = __HAL_TIM_GET_COUNTER(&htim6);

//Calculate the time between the current and last measurements
d_time = now <= encoder->lastUpdateTime ? 0xffff - encoder->lastUpdateTime + now : now -
→ encoder->lastUpdateTime;

EncoderGetAngle(encoder->handle, &angle_data);

//Track the number of revolutions
d_angle = (float)(angle_data - encoder->last_angle_data);

//If detected occurs, count it as a full revolution
if(abs(d_angle) > (0.8 * resolution)){
    if(full_rotation_offset>0{
        d_angle += 2*PI;
    }else{
        d_angle += -2*PI
    }
}

//to determine if an overflow has occurred
encoder->last_angle_data = angle_data;

// Returning a full angle
encoder->angle_rad = encoder->full_rotation_offset + ((float)angle_data / (float)resolution) * 2*PI;
encoder->angle_deg = RADIANS_TO_DEGREES(encoder->angle_rad);

// Расчет скорости
encoder->velocity_rad = ((encoder->angle_rad - encoder->angle_prev_rad) / d_time * 1000000.0f);
encoder->velocity_deg = ((encoder->angle_deg - encoder->angle_prev_deg) / d_time * 1000000.0f);

// Remembering the last angle
encoder->angle_prev_rad = encoder->angle_rad;
encoder->angle_prev_deg = encoder->angle_deg;

//Update time of last measurement
encoder->lastUpdateTime = __HAL_TIM_GET_COUNTER(&htim6);

```

Фрагмент кода обновления алгоритма векторного регулирования

```

void FOC_Update(uint16_t time_us,float setpoint_torque_current_mA,float
    → setpoint_flux_current_mA,float phase_synchro_offset_rad,uint32_t closed_loop,float
    → setpoint_velocity_dps)
{
    float phaseCurrents[3];

    // performance monitoring
    uint16_t startTime = __HAL_TIM_GET_COUNTER(&htim6);

    for(size_t i = 0; i < 3; ++i) {
        phaseCurrents[i] = -((float)motor_current_sample_adc[i] - motor_current_input_adc_offset[i]) /
            → motor_current_input_adc_mA[i];
    }

    // process cosine(theta) and sine(theta)
    float phaseOffset_rad = convert2degrees((int16_t)(MAKE_SHORT(
        → regs[REG_MOTOR_SYNCHRO_L], regs[REG_MOTOR_SYNCHRO_H])));
    float polePairs = PP, direction = REVERSE;
    float theta_rad = fmodf(absolute_position_rad * polePairs * direction, 2*PI) + phaseOffset_rad +
        → syncOffset_rad;
    static float cos_theta = 0.0f, sin_theta = 0.0f;
    CORDIC_Processor(theta_rad, &cos_theta, &sin_theta);

    //Clarke Transformation
    float Ialpha = 2.0f / 3.0f * phaseCurrents[0] - 1.0f / 3.0f * (phaseCurrents[1] + phaseCurrents[2]);
    float Ibeta = 1.0f / sqrtf(3.0f) * (phaseCurrents[1] - phaseCurrents[2]);
    // Park Transformation
    float Id = Ialpha * cos_theta + Ibeta * sin_theta;
    float Iq = -Ialpha * sin_theta + Ibeta * cos_theta;
}

```

Приложение 5.(продолжение)

```

// (Id,Iq) filtering

float Id_filtered = 0.05f * Id + (0.05f) * present_Id_filtered;
float Iq_filtered = 0.05f * Iq + (0.05f) * present_Iq_filtered;

// flux controller (PI+FF)

float setpoint_Id = fluxSet_mA;

float Flux_Kp = (float)((int16_t)(MAKE_SHORT( regs[PID_FLUX_CURRENT_KP_L],
→ regs[PID_FLUX_CURRENT_KP_H])) / 100000.0f;
float error_Id = setpoint_Id - (loopStatus == 1 ? Id_filtered : 0.0f);
float Vd = error_Id * Flux_Kp;

float setpoint_Iq = torqueSet_mA;
float Torque_Kp = (float)((int16_t)(MAKE_SHORT( regs[PID_TORQUE_CURRENT_KP_L],
→ regs[PID_TORQUE_CURRENT_KP_H])) / 100000.0f;
float error_Iq = setpoint_Iq - (loopStatus == 1 ? Iq_filtered : 0.0f);
float Vq = error_Iq * Torque_Kp;

// do inverse clarke and park transformation

float Valpha = Vd * cos_theta - Vq * sin_theta;
float Vbeta = Vq * cos_theta + Vd * sin_theta;

// Inverse Clarke Transformation

float Va = Valpha;
float Vb = (-Valpha + sqrtf(3.0f) * Vbeta) / 2.0f;
float Vc = (-Valpha - sqrtf(3.0f) * Vbeta) / 2.0f;

float Vneutral = 0.5f * (fmaxf(fmaxf(Va, Vb), Vc) + fminf(fminf(Va, Vb), Vc));
// convert (Va,Vb,Vc) to PWM duty cycles % [0.0 1.0]

float dutyA = fconstrain(((Va - Vneutral) / present_voltage_V + 1.0f) * 0.5f,
→ MIN_PWM_DUTY_CYCLE, MAX_PWM_DUTY_CYCLE);

```

Приложение 5.(продолжение)

```

float dutyB = fconstrain(((Vb - Vneutral) / present_voltage_V + 1.0f) * 0.5f,
→ MIN_PWM_DUTY_CYCLE, MAX_PWM_DUTY_CYCLE);
float dutyC = fconstrain(((Vc - Vneutral) / present_voltage_V + 1.0f) * 0.5f,
→ MIN_PWM_DUTY_CYCLE, MAX_PWM_DUTY_CYCLE);

// fPWM = 16KHz
// fTIM = 160MHz
// ARR = fTIM/(2 * fPWM) -1 => ARR = 4999

uint16_t CCRA = (uint16_t)(dutyA * (float)(__HAL_TIM_GET_AUTORELOAD(&htim1)) + 1)) - 1;
uint16_t CCRb = (uint16_t)(dutyB * (float)(__HAL_TIM_GET_AUTORELOAD(&htim1)) + 1)) - 1;
uint16_t CCRC = (uint16_t)(dutyC * (float)(__HAL_TIM_GET_AUTORELOAD(&htim1)) + 1)) - 1;

// update TIMER CCR register
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, CCRA);
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, CCRb);
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, CCRC);

// calculate loop time
int16_t endTime = __HAL_TIM_GET_COUNTER(&htim6);
uint16_t processingTime = endTime - startTime;
static const float performanceAlpha = 0.001f;
average_processing_time_us = (1.0f - performanceAlpha) * average_processing_time_us +
→ performanceAlpha * (float)processingTime;
++foc_cnt;

```

Функция вычисления прямой кинематики

```

CalculateFD(const JointAngles& inputAngles, RobotPose& outputPose)

{
    float rawInputAngles[6];
    float adjustedAngles[6];
    float cosineAngle, sineAngle;
    float cosineAlpha, sineAlpha;
    float endEffectorPosition[6];
    float endEffectorRotation[9];
    float rotationMatrices[6][9];
    float rotationMatrixPartial[4][9]; // Only need 4 as we're multiplying sequentially
    float linkVectors[4][3]; // Vectors from base to shoulder, shoulder to elbow, etc.

// Convert input angles from degrees to radians
for (int i = 0; i < 6; i++)
    rawInputAngles[i] = inputAngles.values[i] / DEGREES_TO_RADIANS;
// Compute rotation matrices based on adjusted joint angles
for (int i = 0; i < 6; i++)
{
    adjustedAngles[i] = rawInputAngles[i] + DHParams[i][0];
    cosineAngle = arm_cos_f32(adjustedAngles[i]);
    sineAngle = arm_sin_f32(adjustedAngles[i]);
    cosineAlpha = arm_cos_f32(DHParams[i][3]);
    sineAlpha = arm_sin_f32(DHParams[i][3]);
// Populate each rotation matrix for the joint
    rotationMatrices[i][0] = cosineAngle;
    rotationMatrices[i][1] = -cosineAlpha * sineAngle;
    rotationMatrices[i][2] = sineAlpha * sineAngle;
    rotationMatrices[i][3] = sineAngle;
    rotationMatrices[i][4] = cosineAlpha * cosineAngle;
    rotationMatrices[i][5] = -sineAlpha * cosineAngle;
}

```

Приложение 6.(продолжение)

```

rotationMatrices[i][6] = 0.0f;
rotationMatrices[i][7] = sineAlpha;
rotationMatrices[i][8] = cosineAlpha;
}

}

// Sequentially multiply the rotation matrices to get the cumulative rotation matrix

MultiplyMatrices(rotationMatrices[0], rotationMatrices[1], rotationMatrixPartial[0], 3, 3, 3);
for (int i = 1; i < 4; i++) // Start from 1 as first multiplication already done
{
    MultiplyMatrices(rotationMatrixPartial[i - 1], rotationMatrices[i + 1], rotationMatrixPartial[i], 3, 3,
                     3);
}

MultiplyMatrices(rotationMatrixPartial[3], rotationMatrices[5], endEffectorRotation, 3, 3, 3);

// Calculate position vectors for each segment in the arm's kinematic chain

MultiplyMatrices(rotationMatrices[0], LinkBaseToShoulder, linkVectors[0], 3, 3, 1);
MultiplyMatrices(rotationMatrixPartial[0], LinkShoulderToElbow, linkVectors[1], 3, 3, 1);
MultiplyMatrices(rotationMatrixPartial[1], LinkElbowToWrist, linkVectors[2], 3, 3, 1);
MultiplyMatrices(endEffectorRotation, LinkWristToEndEffector, linkVectors[3], 3, 3, 1);

// Sum up the position vectors to get the end effector's position

for (int i = 0; i < 3; i++)
    endEffectorPosition[i] = linkVectors[0][i] + linkVectors[1][i] + linkVectors[2][i] + linkVectors[3][i];

// Convert the end effector's rotation matrix to Euler angles

RotationMatrixToEulerAngles(endEffectorRotation, &endEffectorPosition[3]);

// Set the calculated position and orientation to the output pose

outputPose.X = endEffectorPosition[0];
outputPose.Y = endEffectorPosition[1];
outputPose.Z = endEffectorPosition[2];
outputPose.Roll = endEffectorPosition[3] * RADIANS_TO_DEGREES;
outputPose.Pitch = endEffectorPosition[4] * RADIANS_TO_DEGREES;
outputPose.Yaw = endEffectorPosition[5] * RADIANS_TO_DEGREES;
memcpy(outputPose.RotationMatrix, endEffectorRotation, 9 * sizeof(float));

return true;

```

Функция вычисления обратной кинематики

```

bool SolveIK(const Pose& targetPose, const JointStates& lastState, Solutions& solutions){

    // Initialize variables for joint angles and intermediate calculations
    float jointAngles[6], tempAngles[2], wristPos[3];
    float cosAngle, sinAngle, angle1, angle2, distSquared, dist;
    float endEffectorPos[3], endEffectorRot[9], tempRot[9], jointRot[9];
    ConvertPoseToRadians(targetPose, endEffectorPos, endEffectorRot);

    // Calculate the position of the wrist to solve for the first three joints
    CalculateWristPosition(endEffectorPos, endEffectorRot, wristPos);

    // Solve for the first three joints (base, shoulder, elbow)
    bool baseSolved = SolveBaseJoint(wristPos, jointAngles);
    bool armSolved = SolveArmJoints(wristPos, jointAngles, tempAngles);

    // If the base or arm cannot be solved, mark the solution as invalid
    MarkInvalidSolutions(baseSolved, armSolved, solutions);

    for (int i = 0; i < 2; i++) // Loop through possible arm solutions
    {
        // Calculate the rotation matrix for the current arm solution
        CalculateArmRotationMatrix(jointAngles, tempAngles[i], tempRot);

        // Determine the wrist orientation relative to the base
        MultiplyMatrices(tempRot, endEffectorRot, jointRot, 3, 3, 3);

        // Solve for the wrist joints (roll, pitch, yaw)
        bool wristSolved = SolveWristJoints(jointRot, jointAngles + 3);

        // Store the solution if valid
        StoreSolution(wristSolved, jointAngles, solutions, i);
    }

    ConvertSolutionsToDegrees(solutions);

    return true;
}

```