МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра „Электроника и робототехника"

«ДОПУЩЕН К ЗАЩИТЕ»
Зав.кафедрой _____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.
(подпись)

ДИПЛОМНАЯ РАБОТА

На тему: Проектирование автоматизированной системы управления складом

Специальность 5В071600 Приборостроение

Выполнил Байметов Тимур Владимирович Группа ПС-14-3
(Ф.И.О.)

Научный руководитель ст. преп. Голубева Т.В
(ученая степень, звание, Ф.И.О.)
_____ «11» Мая 2018 г.
(подпись)

Консультанты:
по экономической части: доцент Самилбаева Р.О. к.э.н.
(ученая степень, звание, Ф.И.О.)
_____ «14» мая 2018 г.
(подпись)

по безопасности жизнедеятельности: д.т.н. Бекбасаров Ш.М.
(ученая степень, звание, Ф.И.О.)
_____ «26» апреля 2018 г.
(подпись)

Нормоконтролер: ассистент, магистр Орынбай А.А.
(ученая степень, звание, Ф.И.О.)
_____ «14» мая 2018 г.
(подпись)

Рецензент: к.т.н., доцент Пя Д.Р.
(ученая степень, звание, Ф.И.О.)
_____ «16» мая 2018 г.
(подпись)

Алматы 2018

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Факультет ИКИТК

Кафедра Электроники и робототехники

Специальность 5В071600 Приборостроение

ЗАДАНИЕ
на выполнение дипломной работы (проекта)

Студенту Байметову Тимуру Владимировичу
(фамилия, имя, отчество)

Тема работы
(проекта) Проектирование автоматизированной системы управления складом

Утверждена приказом по университету №_____ от «__» _____ 201__ г.

Срок сдачи законченной работы (проекта) «_____»_____201___г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Одновременный доступ к панели управления автоматизированным складом; необходима высокая точность позиционирования платформы; получение данных от датчиков в режиме реального времени. Цена на оборудование, срок окупаемости проекта; сборка конструкции; организация стабильного питания, организация допустимого уровня освещенности при монтаже.

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломной работы: Аппаратное и программное модули автоматизации: WES; Автоматическая система хранения и поиска ASRS. Подбор микроконтроллеров: esp 8266 и Mega2560 для управления нагрузками, организация питания и сборки конструкции платформы ASRS. Создание блок-схем программных модулей и описание программ для каждого микроконтроллера. Расчёт искусственного освещения при монтаже системы. Расчёт капиталовложений на закупку оборудования и разработку программного обеспечения. Оценка экономической эффективности реализации проекта и обоснование для ввода на рынок автоматизации.

Перечень графического материала (с точным указанием обязательных чертежей): Сравнение микроконтроллеров esp 8266, сравнительная таблица микроконтроллеров управления нагрузкой, распиновка микроконтроллера NodeMCU, распиновка микроконтроллера Arduino Mega 2560, схема подключения драйвера ULN 2003 и шагового двигателя, схема пар Дарлингтона, схема контактов реле, схема платформы ASRS, схема с изображением работы двигателей для перемещения платформы по горизонтальной оси, схема работы двигателей для перемещения платформы по вертикальной оси, схема работы двигателей для перемещения погрузчиков, принципиальная электрическая схема системы в программе Fritzing. Принципиальная электрическая схема системы в программе Proteus. Структурная схема системы. Схема сетевой архитектуры. Блок-схема для esp 8266. Блок-схема для Arduino Mega 2560. Схема расположения светильников на складе. График расходов и сроки окупаемости.

Основная рекомендуемая литература: Шпак Ю.А. Программирование на языке С для AVR и PIC – М. 2011., Олифер В. Компьютерные сети. Принципы, технологии, протоколы. – 5-е изд., – М. 2016. З.Д. Еркешева. Методическое указание к выполнению экономической части дипломных работ для студентов специальности 5В071600 – Приборостроение. – Алматы: АУЭС, 2012 – 29с; Документация к микроконтроллеру esp 8266; Документация к микроконтроллеру Mega 2560. Статья: „Описание и технические характеристики автоматизированной системы хранения и поиска ASRS. Статья: „Совершенствование сортировочного центра DG". Статья: „Новые методы хранения и поиска на складе".

Консультации по работе (проекту) с указанием относящихся к ним разделов работы (проекта)

Раздел	Консультант	Сроки	Подпись
БНА Расчет освещения и безопас. условия электро-монт. работ	Бейбасова Ш.С.	26.02.18-	
		26.04.18	
ТЭО Расчёт капитальных вложений, з.п., налог	Самелебаева Р.О.	09.04.18 -	
		14.05.18.	
Технолог часть	Голубева Т.В.	24.10.17-30.10.17	
Конструктор часть	Голубева Т.В	1.02-7.03.18	
Прогр обеспечение	Голубева Т.В.	2.03-2.04.18	

График
подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Технологическая часть:		
Анализ мирового опыта по производству складской автоматики, Аппаратное и		
программное модули автоматизации:		
обзор технологий WES и ASRS, постановка		
задачи дипломного проекта.	24.10.2017.	30%
2. Конструкторская часть:	29.01.2018	
Выбор микроконтроллера esp 8266 и		
контроллера для управления нагрузками.		
Описание и подключение двигателей, реле,		
датчика температуры, организация питания,		
конструкция платформы ASRS, принци-		
пиальная и структурная схема	1.03.2018	30%
3. Программное обеспечение:		
Используемые программные технологии,		
Блок-схемы программных модулей,		
описание программы для микроконтроллера		
esp 8266, описание программы для		
микроконтроллера Atmega2560.	2.04.2018	30%
4. Безопасность жизнедеятельности:		
Анализ условий труда и расчёт искусственного		
освещения. Выводы по разделу безопасности	26.04.2018	5%
5. Ресурсно-экономическое обоснование:		
Расчёт рыночной цены, оценка экономич. эффективности	14.05.2018	5%

Дата выдачи задания «23» 10 2017 г.

Заведующий кафедрой _____ (Аузова А.М.)
(подпись) (Ф.И.О.)

Научный руководитель работы (проекта) _____ (Голубева Т.В.)
(подпись) (Ф.И.О.)

Задание принял к исполнению студент _____ (Байметов Т.В)

Аннотация

В этом дипломном проекте разработана автоматизированная система управления складом, позволяющая хранить и моментально делать выборку из склада и передавать товар принимающему логистическому оператору без участия людей, что значительно увеличивает скорость работы и уменьшает вероятность возникновения ошибок, что даёт возможность внедрения этой технологии на рынок небольших складов, магазинов и сортировочных центров. С технической стороны проект подтверждён опытной моделью и может служить прототипом для последующей реализации на стадии производства.

Annotation

In this graduation project developed an automated warehouse control system that allows you to take, store and instantly make a selection from a warehouse and transfer the goods to the receiving logistics operator without the participation of people, which significantly increases the speed of work and reduces the likelihood of errors that enable the introduction of this technology in the warehouse market and sorting centers. On the technical side, the project is confirmed by an experimental model and can serve as a prototype for subsequent implementation at the production stage.

Аңдатпа

Бұл дипломдік жобада қойманың автоматтандырылған түрде басқару жүйесі көрсетілген , адамның қатысуынсыз логистикалық операторға тауарды дереу таңдау жасауға және сақтауға мүмкіндік береді . Бұл жумыстың жылдамдығын айтарлықтай арттырады және қателіктердің ықтималдығын азайтады. Сондықтан, бұл технологияны кішігірім қоймалардың , дүкендердің және сұрыптайтын орталықтардың нарығына енгізуге мүмкіндік береді. Техниқалық жақтан карасақ, бұл жоба экспериментік үлгісімен рассталды және өндірісте кейіннен іске асыру үшін прототип бола алады.

Содержание

Введение

На данный момент человечество производит огромное множество товаров, которое чтобы доставить от производителя к покупателю необходимо где-то распределять и хранить. Для этого строят различные сооружения, такие как складские здания и распределительные центры. Кроме необходимых построек также требуются физическая сила для транспортировки, интеллектуальная – для определения типа товара, способов транспортировки и других различных расчетов. Поэтому многими компаниями были разработаны крупные системы, включающие в себя все возможные функции для удовлетворения требований по логистике, менеджменту и хранения самых разных товаров по требованию заказчика. Но в мире такие компании работают с крупными заказчиками и почти все сосредоточены в странах Европы и США, поэтому доступность таких технологий в странах СНГ существенно снижена.

В данном дипломном проекте я предлагаю создать систему, работающую в полном цикле движения и хранения товара в пределах склада или сортировочного центра. Также будут рассмотрены возможные варианты реализации и будут созданы демонстрационные модули складской автоматики, разработано программное обеспечение для выборки товаров, непосредственного управления системой хранения и наблюдения за процессами в реальном времени.

1 Технологическая часть

1.1 Автоматизация склада: следующее поколение

Автоматизация является одним из последних рубежей бизнеса на пути к максимальной эффективности. Например, вилочный погрузчик, загружающий товары в паллеты, которые устанавливаются в несколько этажей на высоте 6 метров на складских стеллажах имеет низкую пропускную способность, в то время как мини-робот с автоматическим управлением в виде сенсорных датчиков или камерами реального времени с алгоритмами компьютерного зрения или автоматическая грузовая платформа, транспортирующая небольшие грузы позволяют обрабатывать до 2400 заказов в час. По мере того как заказ совершен, скоростной конвейер перемещает паллеты в сторону стационарного робота или платформы, которые загружают содержимое паллетов в беспилотный грузовик, доставляющий товары в любое место страны. Так может выглядеть склад 21-го века. Увы, на текущий момент большинством компаний используются устаревшие технологии. Например, в США автоматизация уже меняет парадигмы операций в традиционных складских операциях, начиная с глобальной цепочки поставок. С появлением новых центров электронной обработки в Северной Америке компании ищут пути максимизации пропускной способности, повышая точность заказов за счет использования оборудования для автоматической обработки заказов, используя высокоскоростные конвейерные системы.

Автоматизация не ограничивается только модернизацией высокоскоростных конвейерных машин. Инновационные решения для цепочки поставок на современном складе следующего поколения включают в себя модули с автоматическим управлением (AMV или ASRS) для хранения и извлечения, технологию голосовой передачи, носимые гаджеты, сканирование радиочастотной идентификации (RFID) и роботизированные приложения.

После внедрения автоматизированных систем управления складом в крупных компаниях была улучшена точность заказа, а человеко-часовая эффективность увеличилась на 240 процентов, точность распределения ресурсов увеличилась почти на 170 процентов в результате перехода от традиционной системы к автоматизированной [1].

В этой области Amazon была лидером в области роботизированных приложений и автоматики за последние пять лет. Компания планирует добавить в свою дистрибуционную экосистему к 2020 году 16 000 новых роботов. Компания Walmart с 2017 году инвестирует 2 миллиарда долларов в платформы электронной коммерции в течение следующих двух лет. Интеграция будет включать в себя высокопроизводительные решения WMS.

Однако автоматизация нового поколения в сфере складирования не является эксклюзивным решением для поставщиков мобильной и розничной торговли.

Многие дистрибьюторские организации среднего размера взяли на себя обязательства по роботизированным WMS и высокоскоростным конвейерным

системам.

Оптимальное время для перехода на умный и автоматический склад – в случаях консолидации, перемещения или расширения центров распространения. Центры электронной коммерции возглавляют тенденцию к переходу на передовое оборудование ASRS, но остальные рынки сбыта могут анализировать какие интеграции наиболее эффективны и могут адаптироваться к стандартам складских операций. Новые достижения в области управления складами связаны с интеграцией программных решений, которые используют оборудование автоматизации, инструменты оптимизации ресурсов и робототехники. Лучшие в своем классе комплекты программного обеспечения нового поколения объединяют высокоавтоматизированное оборудование ASRS с корпоративными программными решениями. Примером современного склада является система управления складом WMS, которая обменивается данными в режиме реального времени с помощью приложения, направленного на передачу и обработку данных.

Новой тенденцией в программных приложениях является концепция систем управления складом (WCS). WCS позволяет программному обеспечению WMS взаимодействовать с оборудованием для обработки материалов и товаров, системами поиска заказов и технологий распределения ресурсов.

Роджер Уордихан, исполнительный директор компании Fortna, профессиональной службы и инженерной фирмы, базирующейся в Вест-Рид, США представил концепцию своей компании следующим образом: «Когда речь заходит о высокоавтоматизированных средствах, системы управления складом могут предоставлять доступ к данным в режиме реального времени, когда работа измеряется в миллисекундах». На умном складе все аппаратные и программные технологии взаимодействуют друг с другом. WCS оптимизирует уровни запасов и максимизирует пропускную способность на складе с использованием технологии робототехники и высокопроизводительных конвейерных систем.

Центры сортировки прошлого напоминают «золотую лихорадку», поскольку рабочие рассеиваются по складскому помещению в поисках следующего предмета для выполнения заказа, пытаясь минимизировать время на поиск пути. Согласно данным исследования, опубликованного в книге Роберта Палевича «Устойчивая цепочка поставок», сотрудники среднего рабочего склада для выполнения заказов обходят примерно 10 километров за рабочий день, а это составляет 75% рабочего дня.

По информации от Honeywell, неправильный выбор заказов может стоить классической организации в среднем 67 долларов за ошибку. В соответствии с старой моделью сбора заказов, ориентированной на работу людей, количество заказов составляло от 2000 до 3000 в день.

Благодаря передовым системам конвейера, работающим бок о бок с системами «выбор-отгрузка», эффективность выполнения заказов была оптимизирована, а пропускная способность достигла 5000 заказов в день, в

зависимости от типа отрасли. Kiva Systems, основанная в 2003 году в Бостоне была одной из первых компаний в сфере складской робототехники. Основываясь на раннем успехе компании, гигант Amazon приобрел Kiva в 2012 году, для развития рынка поисковой робототехники на собственных крупнейших складах в США. Другие высокотехнологичные лидеры следовали за Kiva и недавно представили аналогичные технологии поисковых роботов. Исходя из предпосылки, что большая часть грузов погружается на паллеты, аппаратная платформа логистической компании Clearpath является моделью, ориентированной на паллеты. Саймон Дрекслер, директор по материальному транспорту Clearpath описывает данную модель следующим образом: «Согласно нашим исследованиям, в Северной Америке есть только 2 миллиарда паллетов, а 80 процентов всех грузовых перевозок связаны с погрузкой товара только на одни паллеты». Используемые роботы обучаемы, чем больше робот перемещается по определенному складу, тем более знакомым становится обстановка, а также любые потенциальные опасности [2].

Не менее увлекательным является появление робототехники, предназначенной для разгрузки, поиска штрихкодов и укладки паллетов. Стационарные роботы с рычажными механизмами способны выполнять множество задач, в том числе разгружать трейлеры с картонными коробками, складывать поддоны и производить упаковку. Эти роботизированные функции призваны заменить складские задачи, на которых рабочие терпят неудачу, которые относятся к категориям грязных, опасных и сложных.

У роботов, используемых для укладки картона или поиска заказов, есть встроенный модуль компьютерного зрения, позволяющий им определять размер коробки для извлечения поддона с товаром. «Робот запрограммирован на размещение большего ящика, погруженного на паллет и меньших коробок на вершину всех больших ящиков для оптимального использования пространства. Роботизированные устройства адаптируются к современному темпу работы благодаря возможности изменять свои функции, чтобы согласовать свою работу с требованиями владельца склада. Например, один и тот же адаптируемый робот может выгрузить коробки из входящего трейлера утром и перейти к укладке ящиков на поддон во второй половине дня используя только базовую настройку манипулятора робота.

Цель внедрения роботов - создавать интеллектуальный склад с повышенной производительностью и повышенной пропускной способностью. Фактически, роботы предназначены для работы в сочетании с рабочей силой. Периодические складские задачи, такие как разгрузка напольного прицепа или контейнера, назначаются роботам автоматическими корпоративными процессами и задачами.

Согласно слов генерального директора Blue Ocean Тима Дероссета: «По мере снижения затрат и совершенствования функций робота мы увидим, что роботы, работающие вместе с сотрудниками склада, повышают эффективность работы. Роботы на самом деле предназначены для сотрудничества с людьми».

Вопрос об использовании срока службы оборудования остаётся отрытым

даже для автоматического передового оборудования при обработке материалов. По словам Клемонса, большинство машин, оборудования ASRS в распределительном секторе имеют прогнозируемый срок службы 20 лет.

Тем не менее, технологические приложения обойдут аппаратные системы в гонке по устареванию. Срок службы программных модулей в системах распределения будет поддерживать только 25 процентов прогнозируемого срока службы автоматизированной аппаратной реализации, а это означает, что большинство инноваций в области программного обеспечения продлится примерно пять лет, прежде чем новая технология WMS/WCS будет представлена на рынке.

Хотя все эти реализации следующего поколения звучат соблазнительно, стоимость для интеграции новейшей технологии RFID, конвейеров, обновленной WMS или робототехники остаётся высокой. На текущий момент сложно рассчитать эффективность окупаемости вложений в складские аппаратные и программные приложения, доступные в сегодняшнем арсенале. При отсутствии вложений в современные технологии не удовлетворяет текущим требованиям заказчика.

Капитальные затраты на конвейерные системы до 2005 года составляли десятки миллионов долларов на сложные и трудоёмкие установки конвейеров и в последующие годы значительно снизились[3].

При рассмотрении инвестиций в высокоавтоматизированное складское оборудование, соображения стоимости должны быть основным фактором в развертывании и реализации.

Например, роботизированные системы, используемые в пакетировании и коробочной упаковке, начинаются от 150 000 долларов и могут легко превзойти инвестиции в миллионы долларов в зависимости от количества машин и масштаба оборудования. Несмотря на кажущуюся высокую стоимость, дистрибьюторские и производственные компании могут рассчитывать на окупаемость в течение двух-трех лет. Клемонс предполагает: «На высокоуровневой автоматизации некоторые компании окупятся за четыре года».

Поскольку пользователи AS\WMS\WCS систем электронной коммерции постоянно увеличивают спрос на заказы, автоматизация и технология должны идти в ногу с высокими требованиями к пропускной способности. Спрос на технологии для e-commerce продолжает расти на 20-25% по сравнению с предыдущим годом [4].

В то время как даже самые прогрессивные центры логистики все еще оценивают различные тенденции автоматизации, такие как робототехника для погрузки / разгрузки и укладки паллет - AGV и технология pick-to-light, более узнаваемые системы легче поддаются оцениванию. Учитывая тенденции в автоматизации складов, ожидаются в разработке оборудования и программного обеспечения следующего поколения не только автоматизированные, но и интеллектуальные модули.

В последнее время была введена концепция использования беспилотных летательных аппаратов внутри центров распространения для приложений «от

человека к человеку». «Это еще одна ранняя технология, - комментирует Питер Уордихан, председатель Fortna. Другие инновации в складах приобретают все большую силу. Индивидуальная трехмерная печать - это тенденция, которая окажет значительное влияние на эволюцию материалов и интернета вещей.

Автоматизация наиболее быстро развивается среди определенных вертикалей отрасли, но только на ограниченном числе мировых рынков. Фактически, согласно Международной федерации роботов, 70 процентов общего объема продаж для роботизированных приложений происходит в следующих пяти странах: Китай, Япония, Соединенные Штаты, Южная Корея, Германия.

Основываясь на недавних экспоненциальных темпах роста в эмиссионном электронном виде, центры по распределению готовятся к удовлетворению рыночного спроса на поставки в тот же день и на следующий день, особенно в праздничные дни[5].

Гибкость имеет решающее значение, когда дело доходит до автоматизации. С таким большим изменением на рынке, и тем более, что электронная коммерция продолжает формировать новые траектории для интеграции технологий и инфраструктуры в компаниях-центрах распределения, которые должны быть в состоянии адаптироваться.

Компании и поставщики постоянно совершенствуют процессы автоматизации. Задача заключается в совместной работе в более высокой пропускной способности, более точно, при меньших затратах. Одним из ключевых моментов является гибкая автоматизация, например, использование устройства-сортировщика для обработки как электронной коммерции, так и розничного пополнения.

Учитывая бесчисленные направления, которые могут предпринять компании, каждый проект автоматизации уникален. Некоторые из них могут принять инкрементный подход: самостоятельно инвестировать в систему WMS/ASRS/WCS или автоматическую конвейерную систему, а затем модифицировать стеллаж, чтобы выстроить инвентарь вертикально.

Другие начинают с нуля с нового объекта и конвертируют ручные операции на основе заказов в полностью автоматизированные. Независимо от того, под ключ или используя готовые решения, нет стандартного руководства для того, как компании подходят к автоматизации - с одним только предостережением.

В зависимости от ассортимента складских товаров с высоким, промежуточным и медленным перемещением, поставщики AS/RS могут свести к минимуму пространство, необходимое для хранения и обработки. Приспособления для транспортировки быстро и плавно транспортируют поддоны и контейнеры в стойку хранения и из нее, обеспечивая более высокую пропускную способность[6]. Автоматизированные системы также обеспечивают следующее очень важное экологическое преимущество: меньшее использование земли. Склад с AS/RS использует на 40 процентов меньше места, чем обычный склад, чтобы хранить такое же количество продуктов.

Автоматизированные склады имеют меньшие размеры, поэтому при строительстве новых объектов требуется меньше земли. Для фирм, расширяющих хранилище благодаря этому, даже в существующем объекте AS / RS может сэкономить затраты на строительство, связанные с созданием более крупного склада. Использование меньшего количества земель приводит к снижению воздействия на окружающую среду. Автоматизированные склады экономят энергию несколькими способами. Например, меньшие площади требуют меньшего освещения и охлаждения. Кроме того, регенеративное торможение на устройствах хранения / поиска дополнительно снижает потребление энергии, поскольку оно подает избыточную энергию торможения в систему электропитания. Наконец, WMC объекта контролирует все потоки продуктов и оптимизирует движения контейнера, что экономит энергию. Автоматический склад уменьшает повреждение продукта, потому что поддоны обрабатываются плавно. Меньше растягивающейся упаковки требуется для закрепления поддонов при движении, что приводит к меньшему количеству отходов расходных материалов. Кроме того, поскольку автоматический сбор заказов более точный, появляется экономия на энергии, рабочей силе и эксплуатационных расходах приносит пользу всему бизнесу. Снижение технического обслуживания. Эксплуатационные расходы для AS / RS ниже, чем для грузовых автомобилей. Автоматизированные средства экономят деньги на лизинговые расходы, станции зарядки и замены, расходы на чистку и ремонт повреждений грузовиков.

1.2 Анализ мирового опыта по производству складской автоматики

Одна из самых популярных компаний-производителей складского оборудования Kardex Remstar, основанная в 1980-х годах, произвела множество оборудования, включая системы вертикального складирования, вертикальная лифтовая система Shuttle XP. Shuttle XP 250/500 показан на рисунке 1.1.

Благодаря своей компактной конструкции Shuttle XP 250/500 может быть оптимально адаптирована к доступному пространству, обеспечивая максимальный объем хранения на минимальной занимаемой площади. Недостатком такой модели является ручное управление и её дороговизна[7].

Рисунок 1.1 – Shuttle XP 250/500 от Kardex Remstar

Mini-Picker от Blue Ocean совместим с ведущими производителями роботов на рынке. Робот-манипулятор Mini-Picker может быть прикреплен к универсальному, KUKA или ABB-роботу в качестве аксессуара. Как следует из названия, Mini-Picker предназначен для выбора небольших предметов, таких

как одиночный винт или болт, и поместить элемент в корзину, коробку или другое устройство доставки. Mini-Picker показан на рисунке 1.2.

Рисунок 1.2 – Mini-Picker от Blue Ocean

Country Vintner и W&H приняли решение на макроуровне, когда они спроектировали центр сортировки Ashland. Начиная с 2008 года, они внесли небольшие изменения в область автоматизации, помимо некоторых настроек для конвейерных систем и носителей информации, что говорит о продуманном планировании, инвестированном в начале проекта.

Vintner создала прямую погрузку в грузовики, протянув конвейеры от док-станций к парковочным местам. В новом центре также была установлена система голосового управления.

В Денвере в 2014 году был открыт объект, который обслуживает около 160 магазинов в двух штатах. Объект сосредоточен вокруг полностью автоматизированной системы хранения и обработки заказов. Данная система может хранить до 36 000 ящиков и может выбирать 32 000 ящиков в день со 100% точностью.

Решение включает в себя систему управления складом (WMS), роботизированные платформы, программные модули и межплатформенную систему связи. Вместо того, чтобы устанавливать традиционный конвейер на уровне пола, система обрабатывает стопки или отдельные пластиковые молочные коробки на ленточных конвейерах с коленом. Шкафы и\или коробки собираются в соответствии с указанной последовательностью Крогера на одном конце объекта, а затем пакетируются для загрузки в грузовик на другом конце объекта, что обеспечивает значительную буферизацию хранилища.

WMS точно контролирует весь поток материала объекта и предлагает гибкое использование пространства, легкую очистку и экономию рабочей силы. Комплексное решение также позволяет автоматически собирать данные для детального анализа диспетчерских операций и планирования производства. Компания получила признание за свой инновационный дизайн объекта и была названа компанией «Dairy Foods 2015 Plant of the Year»[8].

Консолидация двух складов в один сокращает пространство для хранения

на 58%, увеличивает мощность и повышает производительность на 33%. Vac-Con является ведущим производителем промышленных роботов для очистки канализационных и дренажных систем. После объединения двух складов в единый новый объект, предназначенный для вертикального хранения сотен тысяч деталей для строительства грузовиков, компания смогла подвести свой инвентарь под одну крышу.

Автоматическое хранилище сокращает пространство для хранения на 85% при одновременном повышении производительности и точности. Промышленный автоматизированный центр снабжения (IAS), расположенный в Портленде, является дистрибьютором компонентов электрических средств управления и автоматизации с упором на обслуживание рынка производителей оригинального оборудования в Новой Англии. Они хранят широкий спектр электрических продуктов, от источников питания до клемных колодок.

Поскольку бизнес Industrial Automation Supply продолжает расти, стало очевидно, что у них заканчиваются складские помещения. Новый модуль вертикального подъема (VLM) помог компании отложить перенос и повысить производительность.

До покупки блока VLM (Modula) работникам приходилось ходить по складской площади, собирая по одному заказу за раз. Это создавало проблемы, потому что большая часть скоропортящегося продукта, который хранится и обрабатывается, поступает утром и его необходимо отправлять клиентам в тот же день. С самого начала в IAS оптимизировали процессы доставки чтобы сделать их максимально эффективными.

Новое решение VLM обеспечило значительную экономию пространства, вместимостью 98 квадратных футов - более 85% от его 1500 продуктов, которые ранее хранились на стеллажах. Инженеры компании утверждают, что устройство имеет очень короткую кривую обучения и является удобным для пользователя. Другим важным преимуществом является повышение производительности и точности сбора. Время сбора сократилось наполовину, что позволило сократить время для работы над другими задачами. Каждый раз, когда VLM представляет лоток, все части, требуемые для нескольких заказов, могут быть выбраны за один раз. Они также могут выполнять заказы на выбор из одного места для сбора, увеличивая пропускную способность и эффективность работы, поскольку им больше не нужно ходить и искать нужные предметы.

Ifm electronic, основанная в 1969 году в городе Эссен в Германии, разрабатывает, производит и продает датчики, контроллеры и системы промышленной автоматизации по всему миру. На сегодняшний день она выпустила около 17 миллионов продуктов с более чем 650 патентами, 70 из которых были предоставлены только в 2015 году. Столкнувшись с быстрым ростом, компания перешла в новый глобальный распределительный центр. Построенный вокруг автоматизированной системы хранения и поиска на шасси (AS / RS), новый объект увеличил пропускную способность и не позволил компании добавить вторую смену.

На старом складе обработка заказов носила преимущественно ручной характер, а персонал склада работал со списками и RF-терминалами между полками и стеллажами. Ошибки выбора и низкая производительность оператора были обычным делом, которые в сочетании с ограниченными местами хранения и установки останавливали рост компании.

Торбен Петерсен, управляющий директор IFM считает, что средний заказ за последний год составлял две позиции от одной до десяти штук. Если компания хочет увеличить оборот, то возникает необходимость увеличения количества отправляемых пакетов: «Мы тратим около 8% нашего оборота на НИОКР, что означает, что новые продукты внедряются все время. Мы хотим гарантировать, что клиенты могут заказать их, как только они будут доступны, но это означает наличие места для хранения».

Новый дистрибьюторский центр площадью 67 000 квадратных футов компании поставляет дистрибьюторскую сеть компании более чем в 70 странах. Новая система (Vanderlande) включает в себя 10-проходную стеллажную конструкцию, интегрированную систему челночных колес, 25 многонаправленных челноков и семь подъемников, которые позволяют челнокам перемещаться между уровнями стойки и выходами / входами в систему.

Система обеспечивает легкий доступ к 8000 продуктам в 23320 местоположениях. Поскольку каждый челнок может добраться до каждого места, уровень обслуживания поддерживается на максимуме.

Платформы обслуживания с дополнительными лестницами также позволяют оперативно получать доступ к оператору для устранения неполадок. Система также поддерживается планом запасных частей поставщика и горячей линией обслуживания. Perfect Pick показан на рисунке 1.3. Perfect Pick от компании OPEXCorp. - роботизированная технология выбора товаров для человека, которая была разработана для упрощения выполнения заказов. В отличие от более традиционных решений для личного потребления на современном рынке, Perfect Pick является автономным решением. Perfect Pick состоит из модульной стойки с высокой плотностью, а также набора автономных роботизированных транспортных средств, iBOT, которые могут получить доступ к каждому месту хранения в пределах их прохода - как по горизонтали, так и по вертикали. iBOT поставляют инвентарь непосредственно на рабочую станцию, расположенную на одном или обоих концах прохода. Этот прямой интерфейс устраняет необходимость в сложных конвейерных системах или передающем оборудовании, таком как лифты, что усложняет систему. Программное обеспечение управления движением Perfect Pick контролирует положение всех iBOT в проходе и направляет их движения для обеспечения полной оптимизации ресурсов и эффективности работы[9].

Рисунок 1.3 – Perfect Pick от OPEXCorp

Sure Sort – сортировочная станция, обработанные контейнеры которой отправляются оператору Sure Sort, расположенному на эргономичной рабочей станции. Элементы помещаются на конвейер и сканируются. Ожидающий iBOT поставляет элемент в обозначенный контейнер заказа, расположенный по обеим сторонам хранилища.

iBOT управляются программным обеспечением Sure Sort в сочетании с практически любой системой управления автоматизацией склада. Уверенная сортировка обеспечивает доставку всех предметов в назначенные им места заказа с точностью до 2400 единиц в час. Sure Sort обеспечивает наибольшее количество мест сортировки на минимальном расстоянии.

После того, как заказ будет завершен, контейнер для заказа или транспортный контейнер будет отправлен на упаковочную станцию на подготовку к отправке. Sure Sort показан на рисунке 1.4

Рисунок 1.4 - Sure Sort от OPEXCorp

Полностью функциональная базовая версия системы управления складскими запасами непрерывного заказа Vargo была выпущена в виде стартовых комплектов для обработки заказов всего 2000 заказов в день (44 000 единиц в неделю).

Система обрабатывает заказы в распределительном центре, используя подход, основанный на обработке, без движения. Модульная и настраиваемая программа направляет машины для обработки материалов, а также устройства, людей и процессы. Он объединяет и синхронизирует работников склада с прямым доступом к контейнерам с использованием оборудования для выборки заказов через распределительные центры.

В комплект входят информационные панели и визуализация, восемь возможностей функциональности (выбор, сборка, упаковка и отправка) и оптимизированная реализация. Чтобы учесть рост, стартовый комплект может быть дополнен дополнительными модулями для интеграции устройств pick / put-to-lights, сортировочных систем, AS / RS, конвейеров, дополнительных пользователей RF систем.

Традиционно товары перемещаются по распределительному центру с использованием конвейерной системы или с помощью управляемых человеком машин (таких как вилочные погрузчики). В подходе Kiva элементы хранятся на портативных запоминающих устройствах. Когда в систему базы данных Kiva вводится заказ, программное обеспечение размещает ближайший автоматически управляемый автомобиль (бот) к элементу и направляет его для его получения. Мобильные роботы перемещаются по складу, следуя серии компьютеризированных стикеров штрих-кода на полу.

Каждый бот имеет датчик, который предотвращает его столкновение с другими. Когда бот достигает целевого местоположения, он поднимает модульный шкафчик с земли через штопор. Затем робот переносит его к указанному человеку-оператору, чтобы выбрать предметы.

У Kiva две модели роботов. Меньшая модель составляет приблизительно 2 фута на 2,5 фута, способна поднимать 1000 фунтов. Большая модель может нести поддоны и грузы весом до 3000 фунтов. Оба окрашены отличительным оранжевым цветом. Максимальная скорость робота составляет 1,3 метра в секунду. Мобильные боты питаются от батареи и должны заряжаться каждый час в течение пяти минут. Робот Kiva показан на рисунке 1.5.

Относительно новый подход Kiva к автоматизированным системам обработки материалов для выполнения заказов набирает обороты при использовании электронной коммерции, розничным пополнением запасов, распределении запчастей и распространении медицинских устройств. Система намного эффективнее и точнее, чем традиционный метод, заключающийся в том, что работники-люди путешествуют по складу, располагая и собирая предметы.

В настоящее время Amazon имеет 30 000 роботов, работающих на складах[10].

Рисунок 1.5 – Kiva bot от Kiva Systems

1.3 Аппаратные и программные модули автоматизации

1.3.1 Обзор технологий WMS, WCS, WES

Рассмотрим 3 вида программного обеспечения, выполняющие похожие функции, но условия применения различны.

WMS – Warehouse Management System – система менеджмента склада. Это программное обеспечение, предназначенное для поддержки и оптимизации управления складом или распределительным центром.

Они предназначены для ежедневного планирования, организации, укомплектования, управления и контроля за использованием имеющихся ресурсов, перемещений и хранения материалов на складе, внутри и вне склада, одновременно поддерживая персонал в процессе перемещения материала и хранения внутри, а также вокруг склада.

Решение покупки этого ПО для управления складами является прежде всего тактическим шагом, приобретается и используется компаниями для удовлетворения уникальных требований к потребностям клиентов в цепочке поставок и каналах сбыта, когда запасы и рабочая нагрузка больше, чем то, что можно обрабатывать вручную, с электронными таблицами. Мотивация к покупке обычно связана с необходимостью поддерживать рост продаж или улучшать производительность складских работников, а иногда и то и другое. WMS использует базу данных, настроенную для поддержки операций хранилища, содержащую детальные описания различных элементов хранилища, включая отдельные складские единицы хранения (SKU), с помощью которых заказы обрабатываются и хранятся, например: вес, габариты, упаковочный материал, автоматические идентификационные метки (штрих-коды и т.д.) Место хранения склада, например, индивидуальный номер места, последовательность комплектования, тип использования (сбор, резервное хранение и т.д.), Тип хранилища (картонная упаковка, кейс, поддон), размер или емкость места, ограничение хранения (легковоспламеняющееся, опасное, ценные материалы, не подходящие по размеру хранения на паллетах)и т.д., а также ожидаемые темпы производительности труда по размещению или сбыту,

такие как затрачиваемые человеко-часы. Ежедневные функции управления включают:

- планирование ежедневного приема документов активности товара, выбор рабочей нагрузки / заказов, подлежащих обработке в день или смену (это также может быть сделано бизнес-системой на уровне MES или ERP), и вычисление оценки рабочей силы и транспортных средств, необходимых для выбора и отгрузки заказов, обеспечение укомплектованности грузов и своевременное уведомление перевозчиков, чтобы соответствовать ежедневным требованиям.

- организация и упорядочивание выбранных заказов. Организация заказов на выбор может быть реализована многими способами, удовлетворяя потребности клиента. Основная цель состоит в том, чтобы делать преднамеренную выборку, а не выбирать заказы в последовательности, в которой они были получены, если компания не хочет платить перевозчику за транспортировку и доставку. Данный способ организации был назван “Wave Planning” или “Wave Picking” с двумя целями: чтобы свести к минимуму потребность в промежуточном пространстве док-станции, загрузив заказ в место отгрузки до прибытия в док-станцию, минуя в последовательности загрузки на склад.

Также создает порядок в виде бесконечного потока, который будет поддерживать постоянный темп работы складского персонала в течение дня и исключить/сократить запросы клиентов в последнюю минуту создавая сверхурочные виды работ или задержки жёстко регламентированной отправки и т. д.

- укомплектование персонала. Назначение персонала для работы в областях с минимальными перемещениями, используя выше описанный метод «Wave», чтобы свести к минимуму расходы.

- управление данными. Это обеспечение документированных процессов и процедур, встроенных в WMS, и их последовательное применение, использование и соответствие характеру намерений работы и уровня обслуживания компании. Эта функция также может использоваться для разделения отдельных заказов на логические рабочие единицы и возможность назначать их отдельным работникам для выполнения, согласование требований к пропускной способности и физической компоновки, например, отделяя индивидуальный сбор данных от каждого набора единиц и индивидуальную сборку паллетов, для повышения производительности и поддержки управления с обратной связью.

- контрольные метки. Это предоставление контрольных точек для управления и мониторинга прогресса в течение дня, предоставления возможности своевременно реагировать на проблемы и представлять данные для анализа эффективности[11].

Существует множество видов систем менеджмента складом. Они могут быть автономными системами или модулями системы ERP или набором скриптов исполнения цепочки поставок. В зависимости от размера и сложности

организации система может быть такой же простой, как серия рукописных списков, которые обновляются по мере необходимости, электронные таблицы с использованием программного обеспечения, такого как Microsoft Excel или Access, или более ориентированная на рост созданная разработчиками на заказ.

Система управления складом (WCS) - это программное приложение, которое управляет действиями в режиме реального времени на складах и в распределительных центрах (Distribution Centers). В качестве “управляющего узла” для склада / распределительного центра, WCS несет ответственность за то, чтобы все работало бесперебойно, максимизируя эффективность подсистем обработки материалов, и в частности деятельность склада в целом.

Он обеспечивает единый интерфейс для широкого спектра оборудования для обработки материалов, такого как AS/RS, вертикальные и горизонтальные карусели, конвейерные системы-сортировщики, упаковщики и укладчики паллетов и т.д. WCS включает в себя такие функции:

- взаимодействие с системой управления складом верхнего уровня (WMS) и обмен информацией, необходимой для управления ежедневными операциями центра;
- выделение работы на различные подсистемы обработки материалов для балансировки активности системы по создания требуемой рабочей нагрузки.
- предоставление в режиме реального времени директив действий операторам и контроллерам оборудования для погрузочно-разгрузочных работ и для выполнения заказов и требований к маршрутизации продукции;
- динамически назначать коробки для переадресации местоположений на основе определенных алгоритмов сортировки или на основе информации о маршруте или порядке, полученной от WMS (при необходимости);
- создавать файлы данных результатов для создания отчетов и/или загрузки с помощью хост-системы WMS;
- управление операционными экранами (графический пользовательский интерфейс) и функциями для эффективного управления хранилищем;
- сбор статистических данных об эксплуатационных характеристиках системы, чтобы персонал мог оперативно поддерживать оборудование в максимальной производительности и чинить в случае поломки;

Каждая основная функция предназначена для работы как часть интегрированного решения для эффективного связывания хост-систем WMS с системой управления нижнего уровня WCS, одновременно освобождая WMS от требований реального времени, таких как экраны оператора и интерфейсы управления оборудованием нижнего уровня.

Типичный склад / распределительный центр состоит из многоуровневой архитектуры управления, в которой каждый уровень иерархии управления имеет определенную роль.

Верхний уровень иерархии управления - это система управления складом (WMS) или хост. Эта система обрабатывает бизнес-задачи системы, такие как получение заказов клиентов, распределение ресурсов и выдача деклараций доставки и счетов-фактур на основе информации о выполнении заказа и

информации о доставке, полученной от системы управления обработкой материалов (WCS). Она обычно взаимодействует с системой обработки материалов в асинхронном режиме т.е. вне реального времени.

Координация деятельности различных подсистем обработки материалов. Эта роль отведена системе управления складом (WCS). WCS создана для управления в реальном времени (Real Time Control - RTC), а также предоставляет экраны пользовательского интерфейса для мониторинга, управления и диагностики для работников склада.

В качестве координационного центра для управления операционными аспектами системы управления WCS обеспечивает критическую связь между хостом, не являющимся агентом в реальном времени, и системой управления оборудованием в реальном времени. Он получает информацию от хоста верхнего уровня и координирует различные устройства в реальном времени (конвейеры, печатные устройства, индикаторы состояний и т.д.) Для выполнения ежедневной рабочей нагрузки. В каждой точке принятия решения WCS определяет наиболее эффективный маршрут для продукта и передает команды контроллерам оборудования для достижения желаемого результата.

На самом низком уровне, наиболее близком к физическому оборудованию, есть контроллеры оборудования.

Эти контроллеры обычно представляют собой некоторую форму программируемого управляющего узла в виде специализированной системы управления на основе микроконтроллеров. Они взаимодействуют с периферийными устройствами ввода/вывода (I/O), такими как сканеры, принтеры, камеры компьютерного зрения, двигатели, манипуляторы и т. д. ,а также устройствами сбора данных, такими как сканеры штрих-кодов (считыватели штрих-кодов) и весы, и отвечают за физическую работу погрузочно-разгрузочного оборудования. Контроллеры оборудования также отвечают за физическую обработку продукта и отслеживание его от точки к точке в зависимости от направления от систем управления верхнего уровня. Обычно один контроллер связан только с действиями определенной области или подсистемы всей системы обработки материалов.

В конечном счете, иерархия управления в распределительном центре или складе отражает организационную структуру из людей. Персонал управления (или WMS) определяет рабочую нагрузку, которая должна быть выполнена в течение дня, в то время как рабочий персонал (WCS) осуществляет деятельность склада в реальном времени (контролеры оборудования) для завершения поставленных задач. Каждому специалисту по складу назначается конкретная задача, основанная на их области знаний (выборка заказа - транспорт - конвейеры и т. д.). Поскольку каждый оператор (контроллер) выполняет свое индивидуальное назначение, супервизор (WCS) назначает следующую задачу на основе текущей рабочей нагрузки. По мере того, как заказы завершены, WCS возвращает WMS управление статусом заказов вместе с любой соответствующей информацией о заказе[12].

WES возникла как гибридная система, объединяющая специфические

функциональные возможности WMS с функциональностью системы управления складами (WCS) для автоматизированных складов. WCS - это программное обеспечение, которое контролирует перемещение корпусов, картонных коробок, тотализаторов или поддонов на конвейерных и сортировочных системах. В автоматизированных складах, которые развертывают эти типы оборудования для погрузочно-разгрузочных работ, WES добавляет логику бизнес-процессов к почти данным в режиме реального времени WCS.

Поскольку WES тесно интегрирована с автоматизированными системами, такими как конвейеры, сортировка, pick-to-light и т. Д., Она имеет почти реальную видимость проблем с узким местом.

WES (Warehouse execution system) являются компьютеризованной системой, используемой в операциях распределения (логистика) и функционально эквивалентна системе исполнения производства или MES. Также имеет тенденцию быть более «продуманной», чем WCS. Возможности WMS полностью разработаны и часто становятся стандартными.

Поставщики обычно предлагают модульные программные решения, которые вместе образуют всеобъемлющую WES.

Модули охватывают WMS, управление конвейерами, расширенную аналитику/отчетность и функциональные возможности интеграции. Типичные пакеты WES содержат набор лучших методов практики для складских операций - например, сбор и упаковку работники могут настраиваться на основе рабочего процесса. Это позволяет управлять процессом без настройки продукта перед внедрением на склад.

Типичные функциональные возможности склада, такие как пополнение и сбор заказов, используют условия реального времени для оптимизации производительности. В то время как типичная функциональность WMS требует планирования и выпуска заказов, WES ориентирована на выполнение этих задач на основе существующих условий объекта. В результате склады становятся более гибкими.

WES имеет возможность охватывать несколько областей функциональных возможностей хранилища, которые традиционно управляются различными специализированными программными системами. WES может быть развернута для обеспечения функциональности управления складом, функциональности системы управления базой данных, управления оборудованием для обработки материалов (WCS), бизнес-аналитики и интеграции с системами ERP. Охват этого широкого спектра функциональных возможностей является явным преимуществом для WES. В результате WES может использовать свою видимость данных низкого уровня для быстрой адаптации функциональности к текущим условиям. Это особенно актуально для предприятий с автоматизированными системами. WES может использовать свои корни WCS для доступа к передовым системам комплектования и сортировки, предлагая гибкий подход к оптимизации операций в режиме реального времени.

Термин WES используется в отрасли обработки материалов и распределения. Термин возник в следствие «пробела» в предыдущих возможностях функциональности, необходимых для работы распределительного центра и складов.

Поскольку требования широковещательного распределения продолжают оказывать давление на объекты и цепочки поставок, чтобы работать более эффективно, многие дистрибьюторы понимают, что существующие функции WMS и WCS недостаточны для согласования, автоматизации и синхронизации дискретных процессов, необходимых для оптимального управления их бизнесом[13].

1.3.2. Автоматическая система хранения и поиска ASRS

Автоматизированные системы хранения и поиска (Automated storage and retrieval system) представляют собой системы управления запасами, которые широко используются в производственных объектах, распределительных центрах и складах. ASRS от Dematic Systems показан на рисунке 1.6.

Системы AS/RS обычно состоят из машин, которые перемещаются вверх и вниз по одному или нескольким параллельным стеллажам и извлекают продукты и материалы.

Преимущества этих систем многочисленны. Они предоставляют владельцам складов повышенный контроль и отслеживание запасов, в том числе большую гибкость для удовлетворения изменяющихся условий ведения бизнеса. Эти системы AS/RS состоят из модульных подсистем, которые можно легко заменить, чтобы минимизировать время простоя и продлить срок службы всей системы. Они также снижают затраты на рабочую силу, снижают потребности в рабочей силе, повышают безопасность на рабочем месте и удаляют персонал из трудных условий работы (например, среды хранения холодных продуктов). Возможно, что наиболее важно, системы AS/RS могут обеспечить значительную экономию средств при хранении запасов, поскольку значительно улучшенное использование складских площадей - как по вертикали, так и по горизонтали - создает большую плотность хранения. Максимизация доступного пространства для хранения в существующих структурах, исключая хранение и расширение вне площадки. Минимизация общей площади здания до 150% по сравнению с обычными складами. Снижение энергозатрат на 140% в более прохладных средах. Снижение затрат на оплату труда и расходных материалов. Повышение точности инвентаризации и обслуживания клиентов[14].

В качестве силовых механизмов могут использовать комбинированные двигатели с установленным редуктором или высокоскоростные асинхронные двигатели.

Существует несколько типов автоматизированного хранилища и поискового оборудования. К ним относятся:

- системы хранения и поиска с фиксированной загрузкой или

фиксированного доступа (FA) - машины, которые хранят большие грузы (обычно 400+ килограмм), как правило, на поддонах. Эти системы включают в себя один или несколько длинных и узких проходов со стойками хранения, достигающие 30 метров в длину. Большинство систем имеют высоту менее 12 метров.

Такая система имеет систему «барж», которые принимают и хранят грузы и вызываются по требованию персонала.

- mini-load ASRS - работает так же, как и ASRS с единичной нагрузкой, мини-погрузчик ASRS с более легкими нагрузками, обычно весом менее 100 килограмм. Вместо хранения поддонов, данные машины хранят лотки или коробки;

- micro-load ASRS. Работа с нагрузками, даже более легкими, чем ASRS mini, ASRS с микронагрузкой обычно обрабатывает грузы весом менее 10 килограмм в очень маленьких контейнерах или лотках;

- вертикальные подъемные модули (VLM - Vertical Lift Module) - VLM состоят из столбца лотков спереди и сзади модуля с автоматическим «экстрактором» в центре, который хранит и извлекает необходимые лотки.

Лотки могут храниться в фиксированных положениях для повышения пропускной способности, динамически для оптимизации емкости хранилища или в сочетании с ними. Интегрированные системы управления, программное обеспечение и системы «pick-to-light» помогают повысить точность и плотность хранения;

- горизонтальные карусели - идеально подходят для хранения мелких деталей, горизонтальные карусели состоят из серии лотков, которые вращаются горизонтально вокруг центра. Эта система обычно использует от двух до пяти горизонтальных каруселей на интегрированной рабочей станции под названием «башня». Когда оператор вводит информацию инвентаря в программное обеспечение управления карусели, карусели поворачиваются в соответствующие позиции для выбора. Устройство подсвечивает позиции для каждой карусели, чтобы выбрать необходимую позицию и количество. Оператор выбирает нужную карусель и завершает заказ. Инвентаризация может быть создана вручную или роботизирована с использованием автоматизированного экстрактора;

- вертикальные карусели - вращающиеся вертикально, как колесо обозрения, вертикальные карусели располагают серией полок для обеспечения хранения с высокой плотностью, что идеально подходит для быстроходных мелких предметов. Часто используемые для оснастки, компонентов, документов или хранилищ сырья, вертикальные карусели прилагаются со всех сторон для обеспечения хранения предметов с высокой стоимостью, таких как ювелирные изделия и электроника. Когда оператор вводит информацию инвентаризации в программное обеспечение управления карусели, машина поворачивается по кратчайшему пути, чтобы расположить конкретную полку хранения товара в окне выбора. Интегрированные технологии «pick-to-light» помогают в определении правильного товара[15];

Рисунок 1.6 – ASRS от Dematic Systems

1.4 Постановка задачи дипломного проекта

На основе рассмотренных вариантов реализации автоматизации склада, было выяснено что в наше время необходимо максимально абстрагироваться от деятельности человека в условия огромного количества данных и необходимости безошибочно и моментально принимать решения по размещению и выгрузке товаров со склада. Помимо расхода времени людьми на нахождение необходимого товара и коммуникацию между собой, в классических складах существуют огромные статьи расходов на методы доставки информации между людьми и сопутствующие дополнительные траты в виде покупки расходных материалов на производство и оформление сопроводительной документации. Наиболее возможным и перспективным решением проектирования автоматизированной системы собственными силами без привлечения сторонних инвестиций и трудовых затрат будет создание стационарной автоматизированной системы хранения и поиска в комбинации с программной реализацией WES (программного синтеза WMS и WCS). Так как автоматике необходим приличный крутящий момент в зависимости от небольшой массы конструкции, грузов и высокая точность перемещений без затрат на дополнительное управляющее оборудование. На основании этого были выбраны низковольтные шаговые двигатели 28byj-48. Также в связи с требованием работы с массивами данных, необходимо сделать склад полностью цифровым и с возможностью управления удалённо складом и получать данные от него в режиме реального времени в локальной сети.

Для этого будет использоваться набирающий популярность контроллер esp8266 в купе с реле для аварийного включения освещения и датчиком температуры и влажности.

Так как проект несет практический характер, необходимо подумать над внедрением его на рынок. Для этого будет проведён экономический расчёт и

анализ безопасности труда.

Поэтому основными задачами данного дипломного проекта будут являться разработка конструкции, механических исполнительных механизмов, создание схемы управления, организация удалённого управления и разработка программного обеспечения.

2 Конструкторская часть

Система базируется на двух микроконтроллерах, выполняющих работу как независимо друг от друга, так и принимая решения на основе переданных между собой данных. Монтируются эти контроллеры на платформу, которая может изменять своё положение в трёх независимых друг от друга осях и выполняет механическую работу по перемещению грузов с помощью нескольких шаговых двигателей. Центром управления всей системы служит запущенный на контроллере esp8266 HTTP сервер, к которому можно получить доступ через браузер и отдать команды на выборку нужной ячейки склада, используя описанные ниже приведённые компоненты. Схема сборки данной системы будет продемонстрирована в программной части данного дипломного проекта. Для реализации данной системы необходимо обосновать выбор используемых микроконтроллеров.

2.1 Выбор микроконтроллера esp8266

Контроллеры esp8266 поставляются на рынок в виде devboard'ов, то есть помимо контроллера esp8266 на плате присутствует USB-UART мост на базе микросхем CH340,341 и CP2102, стабилизатор напряжения ams1117 на 3.3 вольта, удобными кнопками reset и boot и гребёнкой на 2.54мм, то есть 1 дюйм, что позволяет удобно установить контроллер на макетную плату. Помимо плат разработчика есть чистые контроллеры без дополнительной периферии. Они немного дешевле, но требуют аккуратного пользования в питании и прошивке, при их использовании также необходим отдельный контроллер для связи с компьютером что повышает вероятность возможности вывести контроллер из строя. Сравнение контроллеров представлено в таблице 2.1.

Таблица 2.1 – Сравнение микроконтроллеров esp8266

Название	Esp-01	Esp-12E NodeMCU	Esp-07
Внешний вид			
Кол-во GPIO пинов	2	11	9
Объём SPI-Flash, Мб	1	4	1
Работа на макетной плате	Легко	Легко	Необходима 2.54мм гребенка
USB-UART	Нет	Есть	Нет
Размер, мм	15x30	25x45	16x24
Антенна	Разведенная	Разведенная	Керамическая

Так как данный проект является прототипом, следовательно необходима быстрая установка, соединение с периферией и программирование. Требований к размерам контроллера нет, более того необходимо иметь достаточный запас портов GPIO для дальнейшего расширения функционала автоматизированного склада, на основании приведённых выше требований наиболее подходящей является плата Amica NodeMCU esp-12e.

Wi-Fi модуль Amica NodeMCU v2 Devkit представляет собой вариацию контроллера esp8266, а точнее esp 12-e разработанный китайской компанией “espressif systems” на основе чипсета esp8266ex с дополнительной обвязкой в виде USB-UART моста на базе микросхемы CP2102, стабилизатора напряжения на 3.3 вольта AMS-1117, SPI-flash накопителя на 4 Мб и 2.54 мм. гребёнкой для удобной работы на беспаечной макетной плате. Также снабжена одноимённой прошивкой NodeMCU, позволяющий писать скрипты на языке Lua в событийно-ориентированном стиле, размером в несколько строк, позволяющие быстро создать прототип программы необходимого IoT устройства. Также есть возможность компилировать скетчи в Arduino IDE и управлять через AT команды. Всё это позволяет сразу приступить к разработке в стиле “Plug & Play”[16]. Модуль показан на рисунке 2.1. Распиновка модуля на рисунке 2.2.

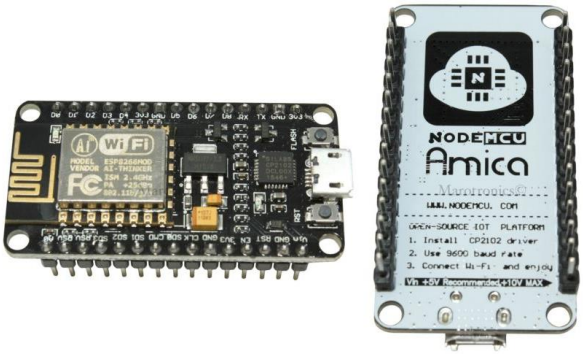


Рисунок 2.1 - Wi-Fi модуль Amica NodeMCU v2. Devkit

Основные характеристики NodeMCU (ESP-12E):
- поддержка WiFi протоколов 802.11 b/g/n;
- режимы STA, AP, STA-AP;
- встроенный стек TCP/IP;
- усилитель мощности +25 dBm в режиме 802.11b;
- I2C, SPI, UART, PWM, ADC;
- пробуждение и отправка пакетов за время до 22 мс;
- номинальное напряжение логики: 3,3 В;
- входное напряжение питания: 3,7–5 В;
- максимальный потребляемый ток: 300 мА.

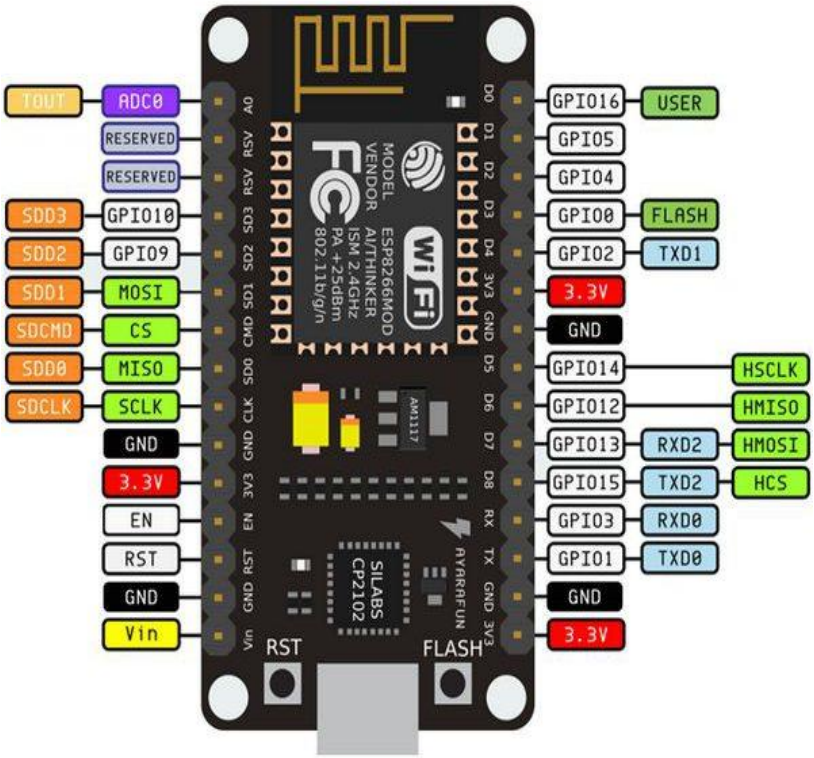


Рисунок 2.2 – Распиновка NodeMCU

2.2 Выбор контроллера для управления нагрузками

Основным назначением будущего контроллера будет одновременное управление большим количеством шаговых двигателей и одним реле, соответственно контроллер должен ссоответствовать лишь 3 условиям:

- много цифровых портов ввода\вывода.
- большой объём flash-памяти для хранения программы
- наличие I2C интерфейса для связи с esp8266

Таблица сравнения контроллеров для управления исполнительными механизмами представлена в таблице 2.2.

Исходя из сравнения характеристик, наиболее походящей является Arduino Mega2560, так как для данного проекта размер платы не имеет значения, поскольку она имеет необходимое количество цифровых входов\выходов.

Также на контроллере flash-памяти больше, чем у Arduino Uno в 8 раз, что даёт возможность на стадии прототипирования проекта не заботиться об оптимизации кода.

Основную работу платы выполняет 8-битный микроконтроллер семейства AVR — ATmega2560. На борту у него 256 КБ флэш-памяти для хранения, кода программы, 8 КБ быстродействующей памяти SRAM и 4 КБ энергонезависимой памяти EEPROM для хранения локальных данных.

Микроконтроллер ATmega16U2 создает USB-UART мост к микроконтроллеру ATmega2560 с USB-порта компьютера. При подключении

создаётся виртуальный COM-порт. Прошивка микросхемы 16U2 использует стандартный протокол USB-UART(COM)[17]. Плата Arduino MEGA 2560 показана на рисунке 2.3.

Таблица 2.2 – Сравнительная таблица микроконтроллеров управления нагрузкой

Характеристики	ArduinoMega2560	Arduino Uno
Микроконтроллер	Atmega 2560	Atmega 328
Логическое напряжение, В	5	5
Flash память, кБ	256	32
Кол-во цифровых входов\выходов	54	14
I2C	есть	есть
Размер, мм	102x53	69x53

Рисунок 2.3 - Arduino MEGA 2560

Порты питания:

VIN: Напряжение от внешнего источника питания (не связано с 5 В от USB или другим стабилизированным напряжением). Через этот порт можно как давать внешнее питание, так и подключить небольшую нагрузку, когда устройство работает от внешнего источника.

5V: На вывод поступает напряжение 5 В от стабилизатора напряжения на плате, независимости от того, как подключено устройство: от импульсного блока питания (7–12 В), от USB (5 В) или через порт V IN (7–12 В). Стабилизатор обеспечивает питание контроллера ATmega2560. Питать устройство через вывод 5V не рекомендуется — в этом случае не используется стабилизатор напряжения, что может привести к выходу платы из строя. Но необходимо следить за просадкой. Если напряжение упадёт до критического уровня, контроллер перезагрузится.

3.3V: 3,3 В от стабилизатора напряжения платы. Максимальный ток — 50 мА.

GND: Земля.

IOREF: Этот вывод предоставляет платам расширения информацию о рабочем напряжении микроконтроллера.

В зависимости от напряжения на нём, плата расширения может переключиться на соответствующий источник питания либо задействовать преобразователи уровней, что позволит ей работать как с 5 В, так и с 3,3 В устройствами.

Порты ввода/вывода.

Цифровые входы/выходы: пины 0–53 Логический уровень единицы — 5 В, нуля — 0 В. Максимальный ток выхода — 40 мА. К контактам подключены подтягивающие резисторы, которые по умолчанию выключены, но могут быть включены программно.

ШИМ: пины 2–13 и 44–46 позволяют выводить 8-битные аналоговые значения в виде ШИМ-сигнала.

АЦП: пины A0–A16. 16 аналоговых входов, каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 значений). Разрядность АЦП — 10 бит.

TWI/I²C: пины 20(SDA) и 21(SCL). Для общения с периферией по синхронному протоколу, через 2 провода. Для работы используйте библиотеку Wire.

SPI: пины 50(MISO), 51(MOSI), 52(SCK) и 53(SS).

Пины коммутации по интерфейсу SPI (используйте библиотеку SPI).

UART: Serial: пины 0(RX) и 1(TX); Serial1: пины 19(RX) и 18(TX);Serial 2: пины 17(RX) и 16(TX); Serial3: пины 15(RX) и 14(TX).

Эти выводы используются для получения (RX) и передачи (TX) данных по последовательному интерфейсу. Выводы 0(RX) и 1(TX) соединены с соответствующими выводами микросхемы ATmega16U2, выполняющей роль преобразователя USB-UART.

Разъём USB Type-B. Разъём USB Type-B предназначен для прошивки платформы Arduino Mega 2560 с помощью компьютера. Разъём для внешнего питания. Разъём для подключения внешнего питания от 7 В до 12 В.

ICSP-разъём предназначен для внутрисхемного программирования микроконтроллера ATmega2560. Также с применением библиотеки SPI данные выводы могут осуществлять связь с платами расширения по интерфейсу SPI.

Линии SPI выведены на 6-контактный разъём, а также продублированы на цифровых пинах 50(MISO), 51(MOSI), 52(SCK) и 53(SS).

ICSP-разъём для ATmega16U2 для внутрисхемного программирования микроконтроллера ATmega16U2.

Распиновка показана на рисунке 2.4.

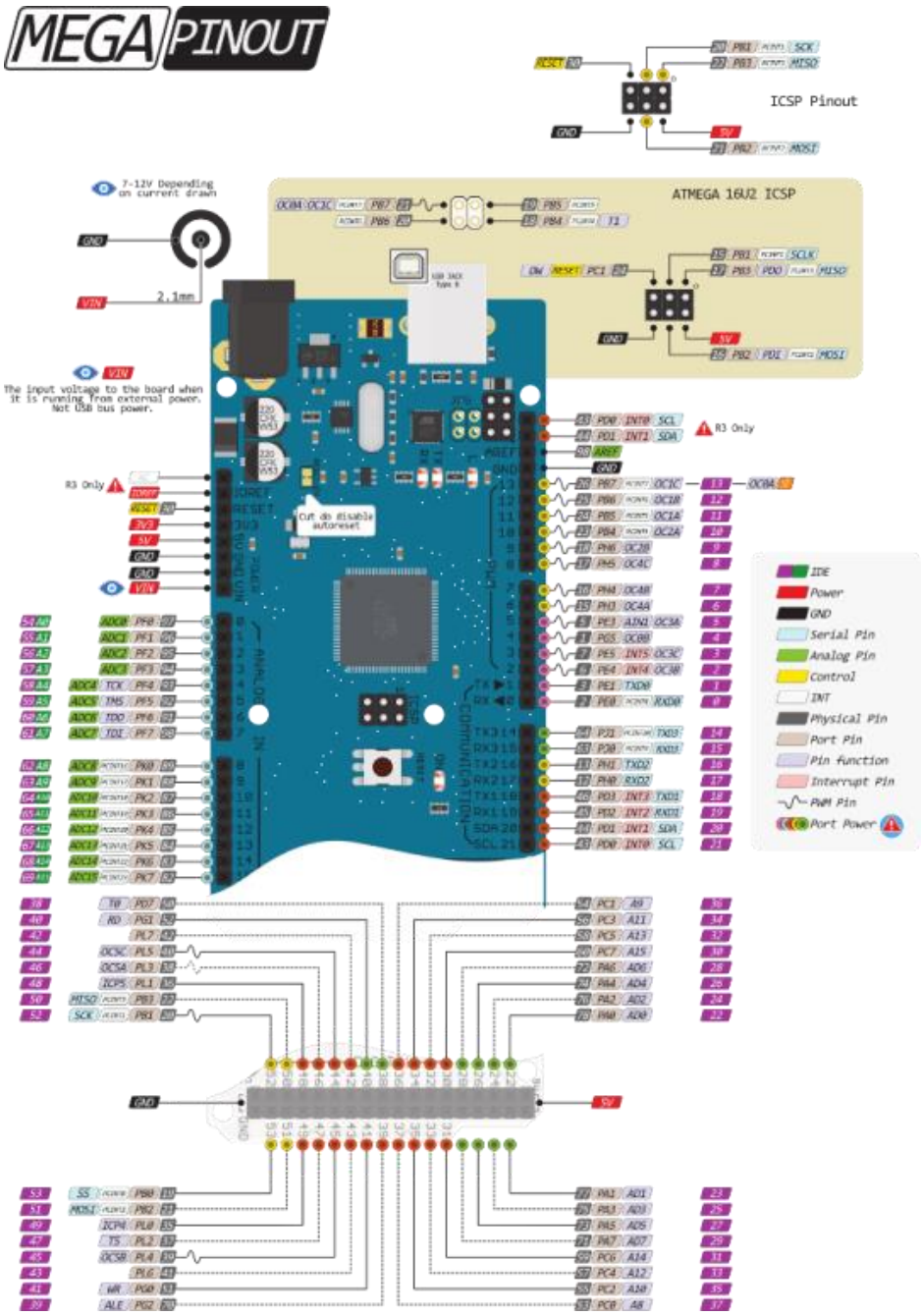


Рисунок 2.4 – Распиновка Arduino Mega 2560

2.3 Выбор шаговых двигателей

Для подобного проекта точность и скорость перемещения платформы является критической. Но так как проект является прототипом и стоимость должна быть минимальной, необходимы самые доступные на рынке шаговые двигатели. Соответственно возможно придётся пожертвовать скоростью вращения вала. На рынке представлено множество двигателей. Основной критерий выбора это крутящий момент и скоростью. Точность работы приблизительно равна. Рассмотрим в таблице 2.3 некоторые двигатели. Это 28BYJ-48, Nema23 FL57STH и 20BYT-01.

Таблица 2.3 – Сравнительная таблица шаговых двигателей

Название	28BYJ-48	Nema23 FL57STH76-2804	20BYT-01
Внешний вид			
Напряжение питания, В	5 и 12	12	5
Крутящий момент, кг/см	0,3 или 0,6	18,9	0,06
Угловой шаг, градусов	5,625	1,8	18
Цена, тг	500	10 000	300
Потребляемый ток, А	0,5	4	0,2

Исходя из таблицы видно, что для проекта больше подойдёт шаговый двигатель 28BYJ-48, так как стоит недорого, мощности должно хватить для небольших грузов, и точность углового шага на хорошем уровне. В то время как у 20BYT-01 крутящий момент слишком мал, а Nema23 FL57STH76-2804 гораздо дороже остальных и его потенциал не будет использован на максимум.

В данном проекте будет использоваться 8 двигателей на 12 вольт и 4 двигателя на 5 вольт. И каждый двигатель управляется через собственный драйвер, потому, что напрямую соединять двигатель с микроконтроллером нельзя, т.к. большая нагрузка, превышающая 20-40мА на пине контроллера, может вывести из его из строя или даже весь контроллер целиком. Двигатель показан на рисунке 2.5.

Рисунок 2.5 – Внешний вид шагового двигателя и драйвера ULN2003

Двигатель имеет 4 катушки, которые соединены последовательно, чтобы создать вращение за счёт магнитного поля. При питании двух соседних обмоток одновременно будет создан полношаговый режим. Этот метод используется в стандартной тестовой библиотеке по умолчанию, которая поставляется предустановленной Arduino IDE называемой “step motor”. В техническом описании 28BYJ-48 указывается, что предпочтительный способ управления этим двигателем - использовать метод полушага, где мы сначала используем только одну катушку, затем катушку 1 и 2 вместе, затем только катушку 2 и т.д. Данный метод использования попеременного использования одной и двух катушек позволяет получить 8 различных сигналов. Порядок подачи сигналов показан в таблице 2.4.

Таблица 2.4 – Порядок подачи сигналов на двигатель для вращения по часовой стрелке

Цвет провода	Порядок шага							
	1	2	3	4	5	6	7	8
4 Оранжевый	+	+						+
3 Желтый		+	+	+				
2 Розовый				+	+	+		
1 Синий						+	+	+

На плате драйвера ULN2003 имеются пины управления IN1, IN2, IN3, IN4, пины питания 5-12V (- +), светодиоды работы обмоток и переключатель, позволяющий контролировать питание, поступающее на двигатель.

К драйверу необходимо подключать внешнее питание, например, блок питания, так как двигатель потребляет на пике до 500 мА, поэтому плата Ардуино не подходит для работы под нагрузкой. Драйвер ULN2003 показан на рисунке 2.6.

Схема подключения такова. Пины IN1, IN2, IN3, IN4 драйвера ULN2003, подключаем к любым цифровым выводам платы Ардуино.

Микросхема ULN2003a - это транзисторная сборка Дарлингтона с выходными ключами повышенной мощности, имеющая на выходах защитные диоды, которые предназначены для защиты управляющих электрических цепей от обратного выброса напряжения от индуктивной нагрузки.

Каждый канал (пара Дарлингтона) в ULN2003 рассчитан на нагрузку 500 мА и выдерживает максимальный ток до 600 мА. Входы и выходы расположены в корпусе микросхемы друг напротив друга, что значительно облегчает разводку печатной платы. Пары Дарлингтона показаны на рисунке 2.8 и 2.9.

ULN2003 относится к семейству микросхем ULN200X. Различные версии этой микросхемы предназначены для определенной логики. В частности, микросхема ULN2003 предназначена для работы с TTL логикой (5В) и логических устройств CMOS. Широкое применение ULN2003 нашло в схемах

управления широким спектром нагрузок, в качестве релейных драйверов, драйверов дисплея, линейных драйверов и т. д.

ULN2003 также используется в драйверах шаговых двигателей. Схема подключения показана в приведенном ниже рисунке 2.7.

Рисунок 2.6 – Внешний вид микросхемы ULN2003

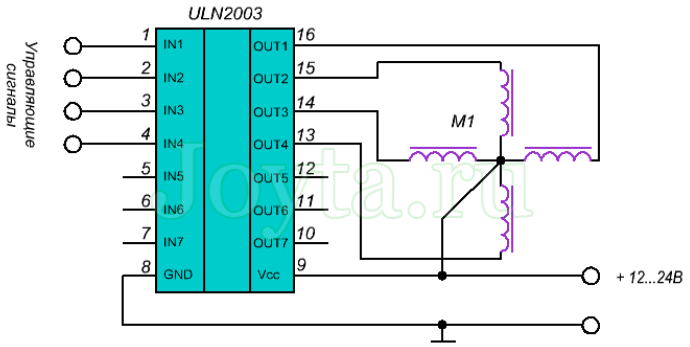


Рисунок 2.7 - Схема подключения драйвера и двигателя

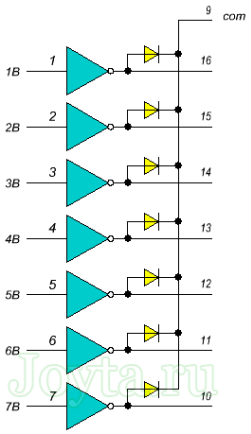


Рисунок 2.8 – Пары Дарлингтона в общем виде

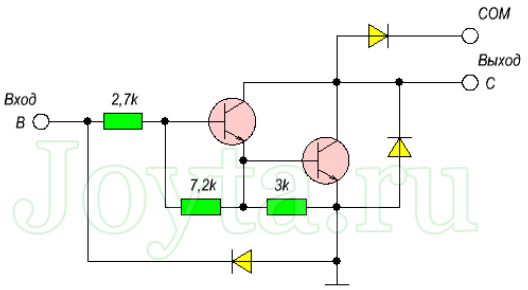


Рисунок 2.9 – Пары Дарлингтона в подробном виде

2.4 Выбор реле для управления аварийным освещением

Модули реле в основном делятся на 2 вида. Это электромеханические и твердотельные. Электромеханическое реле имеет две раздельных цепи: цепь управления, представленная контактами A1, A2 и управляемая цепь, контакты 1, 2, 3. Цепи никак не связаны между собой. Схема контактов приведена на рисунке 2.10.

Между контактами A1 и A2 установлен металлический сердечник, при протекании тока по которому к нему притягивается подвижный якорь(2).

Контакты же 1 и 3 неподвижны. Стоит отметить что якорь подпружинен и пока мы не пропустим ток через сердечник, якорь будет удерживается прижатым к контакту 3. При подаче тока, сердечник превращается в электромагнит и притягивается к контакту 1. При обесточивании пружина снова возвращает якорь к контакту 3.

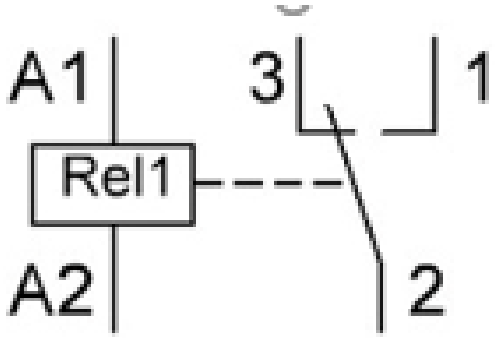


Рисунок 2.10 – Схема контактов

Твердотельное реле (ТТР) – электронное устройство, включающее и выключающее при помощи низких напряжений высокомощностную цепь. В этом виде реле отсутствуют механические движущиеся элементы. Прибор состоит из датчика, реагирующего на вход (управляющий сигнал), и твердотельной электроники, компонентом которой является цепь высокой мощности. Оно может применяться в сетях как постоянного, так и переменного тока. В твердотельных реле, выпускающихся серийно, используются тиристоры и транзисторы, способствующие переключению токов до сотен ампер. По сравнению с электромеханическими, твердотельные реле обладают более высокой скоростью переключения. Однако они менее пригодны к работе в условиях кратковременных перегрузок. В твердотельных реле взаимодействие управляющего сигнала с управляемым происходит путем формирования гальванической развязки – как правило, с помощью оптрона. Управляющее напряжение подает питание на светодиод, а он, в свою очередь, освещает фотодиод, и с помощью тока последнего включается МОП или тиристор, управляющий нагрузкой. Тиристоры и симисторы используются в устройствах, применяемых при переменном токе, а транзисторы – в приборах с постоянным током. Также применяются и специализированные

оптоэлектронные приборы – оптотиристоры и фототиристоры. Схема твердотельного MOSFET реле показана на рисунке 2.11.

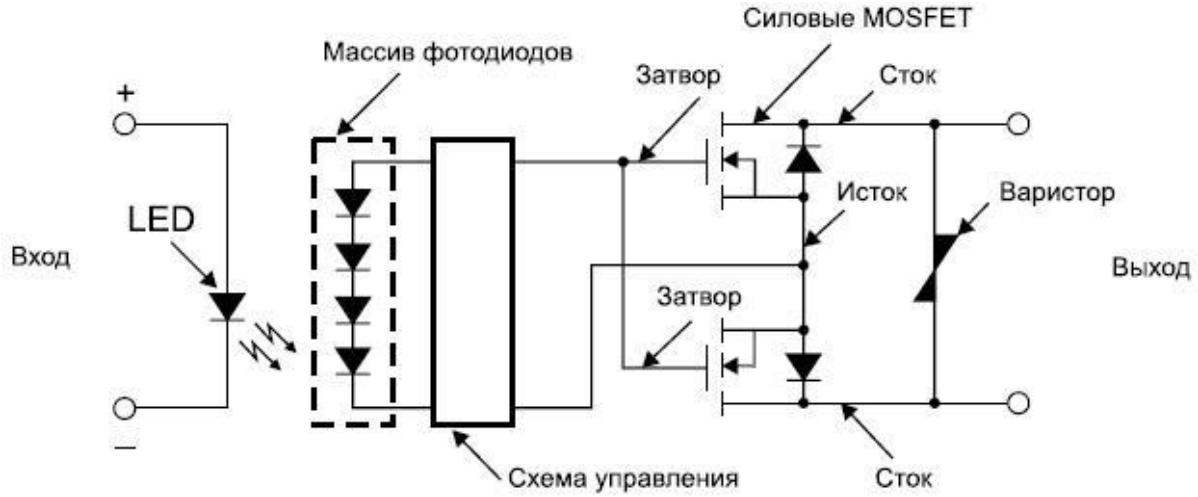


Рисунок 2.11 – Схема твердотельного MOSFET реле

Рассмотрим представителей обоих видов в таблице 2.5.

Таблица 2.5 – Сравнение реле для управления аварийным освещением

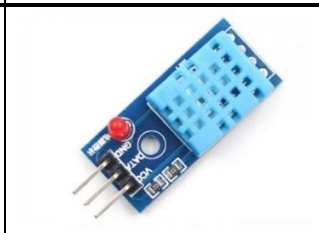
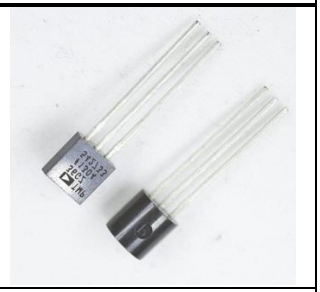
Название	SONGLE SRD-05VDC	SRD-12VDC-SL-C	Omron G3MB-202P
Внешний вид			
Тип	Электромеханический	Электромеханический	Твердотельный
Коммутируемый ток, А	10	10	2
Напряжение	5	12	12
Цена, тг	500	200	300
Клеммы, Индикация	есть	нет	Нет

Как видно по таблице для проекта будет выбрано реле SRD-05VDC. Так как напряжение логики контроллера в системе 5 вольт, а также цена не различается очень сильно и модуль полностью готов к работе. Нет необходимости в дополнительной работе с пайкой.

2.5 Выбор датчика температуры

Для наблюдения за климатическим состоянием склада необходимы как минимум датчики температуры. На рынке представлены большое множество. Но для проекта необходимы самые доступные и простые в использовании. Рассмотрим таковые в таблице 2.6.

Таблица 2.6 – Сравнение датчиков температуры

Название	DHT11	LM35	TMP36
Внешний вид			
Диапазон измерения	0-50	10-125	-40 - 150
Напряжение, В	3.3 - 5	2.7-5.5	2.7 – 5.5
Цена, тг	400	300	300
Дополнительные функции	Измерение влажности	нет	Нет

Как видно по таблице для проекта подходят все датчики. Но будет выбран датчик DHT11, так как имеет дополнительную измеряемую величину – относительную влажность воздуха. А также потому, что данный датчик уже имеется в наличии, и нет необходимости покупать новый.

Датчик влажности и температуры DHT11 состоит из 3 основных компонентов. Датчик влажности резистивного типа, термистор NTC (отрицательный температурный коэффициент) (для измерения температуры) и 8-разрядный микроконтроллер, который преобразует аналоговые сигналы от обоих датчиков и отправляет цифровой сигнал по одному проводу «One Wire». DHT11 измеряет температуру с помощью поверхностного термодатчика NTC (термистора), встроенного в устройство.

Характеристики:

- напряжение питания : 3-5 V;
- диапазон измерениия температуры : 0-50 °C;
- частота измерения: 1Гц;
- определение влажности в диапазоне 20-80%.

2.6 Организация питания

Так как система предназначена для работы в помещении, используется возможность питания от сети 220В. Для этого используются импульсные блоки

питания. AC-DC преобразователи (Импульсные блоки питания). В связи с тем, что на платформу установлены шаговые двигатели на 12 и 5 вольт и каждый потребляющий на пике до 500мА тока, и к контроллеру подключено около 50 выводов на 5 вольт и потреблением тока в 20 мА, необходимы мощные преобразователи. Были выбраны 2 блока питания. NES-50-5 и LEDW-200-12. NES-50-5 рассчитан на 50 ватт при выдаваемом напряжении на выходе в 5 вольт. Это блок питания изображен на рисунке 2.12. NES-50-5 рассчитан на 200 ватт мощности при выдаваемом напряжении на выходе в 12 вольт. Это блок питания изображен на рисунке 2.13.

Рисунок 2.12 – Импульсный блок питания NES-50-5

Рисунок 2.13 – Импульсный блок питания LEDW-200-12

2.7 Конструкция платформы ASRS и принцип работы

ASRS - Automated Storage and Retrieval System. Или другими словами автоматизированная система хранения и поиска. Представляет собой металлическую платформу, позволяющую перемещать грузы в определённые ячейки склада представляющий из себя двумерный массив ячеек. Принцип работы заключается в независимой работе шаговых двигателей, расположенных на ней. Каждая часть шаговых двигателей отвечает за свою область перемещения платформы по 3 разным осям: горизонтальную, вертикальную и глубинную. Всего на платформе установлено 12 двигателей. По 4 на каждую ось. Управляющие сигналы поступают от микроконтроллера

Atmega 2560 на драйверы ULN2003, а уже от них напрямую к двигателям которые вращаются по часовой или против часовой стрелки в зависимости от требования направления.

Вся конструкция выполнена из металла для повышенной прочности, устойчивости и невозможности изгиба под влиянием нагрузки. Перемещение осуществляется с помощью ременной передачи. На вал каждого шагового двигателя установлен зубчатый шкив, на который установлен зубчатый ремень. Такой метод работы лежит в основе работы 3D принтеров. В итоге получается, что для стабильной работы необходимо создавать равномерные тяговые усилия, что достигаются путём соединения двух шаговых двигателей одним ремнём. Но в силу принятой конструкции, необходимо устанавливать такие механизмы параллельно. По 2 на каждую ось перемещения. Также для надёжной работы сцепления шкива с ремнём необходимо усиление в натяжении ремня, иначе при потере трения, шкив может пропускать звенья ремня, что создаст ассиметричное положения платформы, что в итоге приведёт к полной неработоспособности.

Для натяжения ремня будет использовать специальный пружинный механизм, устанавливаемый прямо на ремень между двумя двигателями. Для эффективного перемещения, платформы максимально исключающего трение, необходимо было установить колёсную базу в основании всей платформы. ASRS изображена на рисунках 2.14 и 2.15.

Также для демонстрационной модели ASRS был собран склад из металла на 12 ячеек, 3 по вертикали и 4 по горизонтали. Склад изображен на рисунке 2.16. Схема платформы показана на рисунке 2.17. Схемы вращения двигателей для перемещения платформы показаны на рисунках 2.18, 2.19, 2.20, 2.21, 2.22, 2.23.

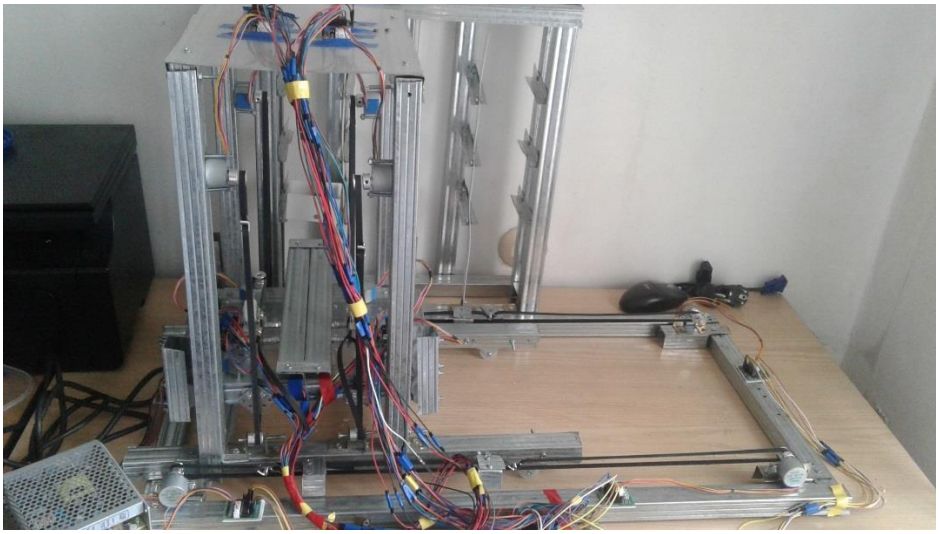


Рисунок 2.14 – ASRS. Фронтальный вид

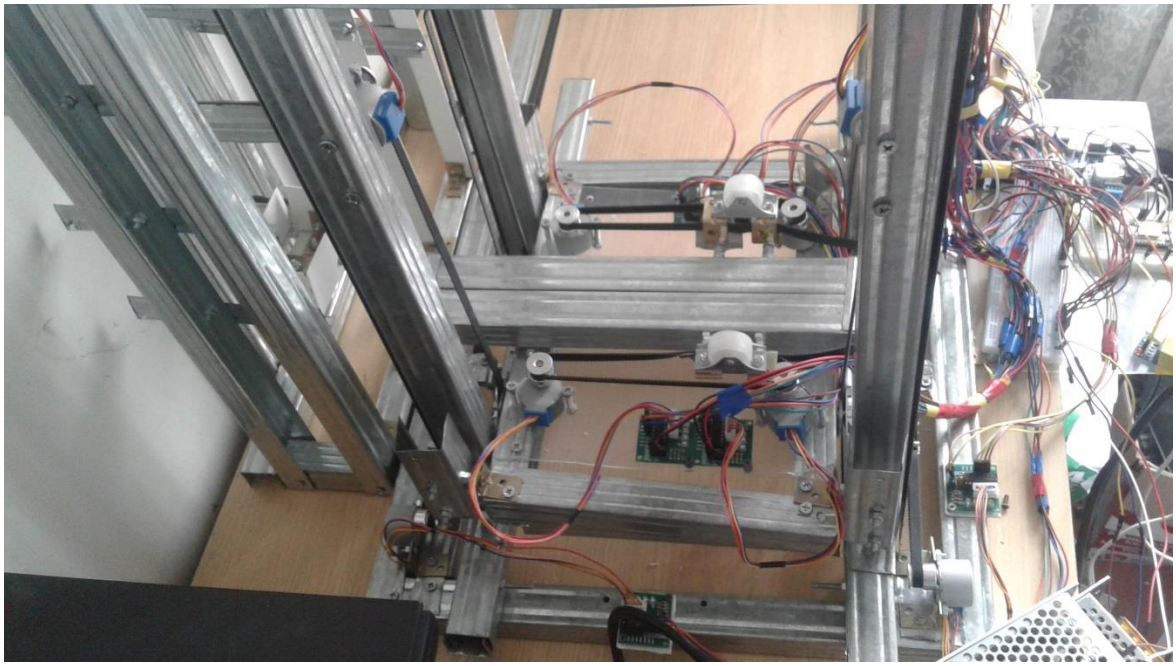


Рисунок 2.15 – Рабочая область платформы ASRS

Рисунок 2.16 – Склад для ASRS

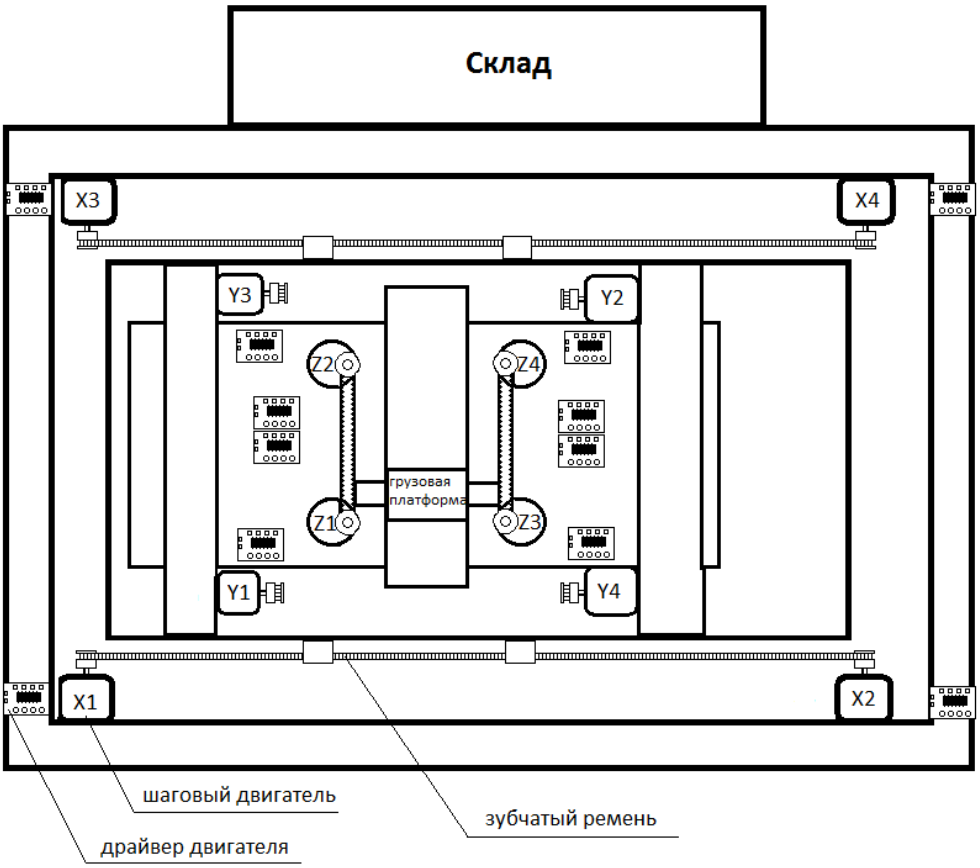


Рисунок 2.17 – Схема платформы ASRS

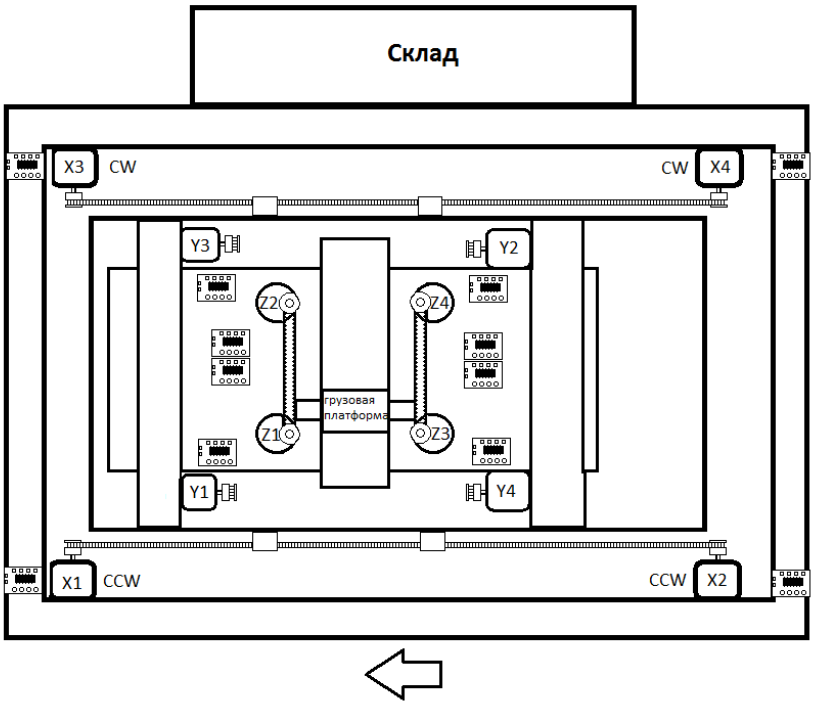


Рисунок 2.18 – Работа двигателей для перемещения платформы налево по горизонтали (оси X)

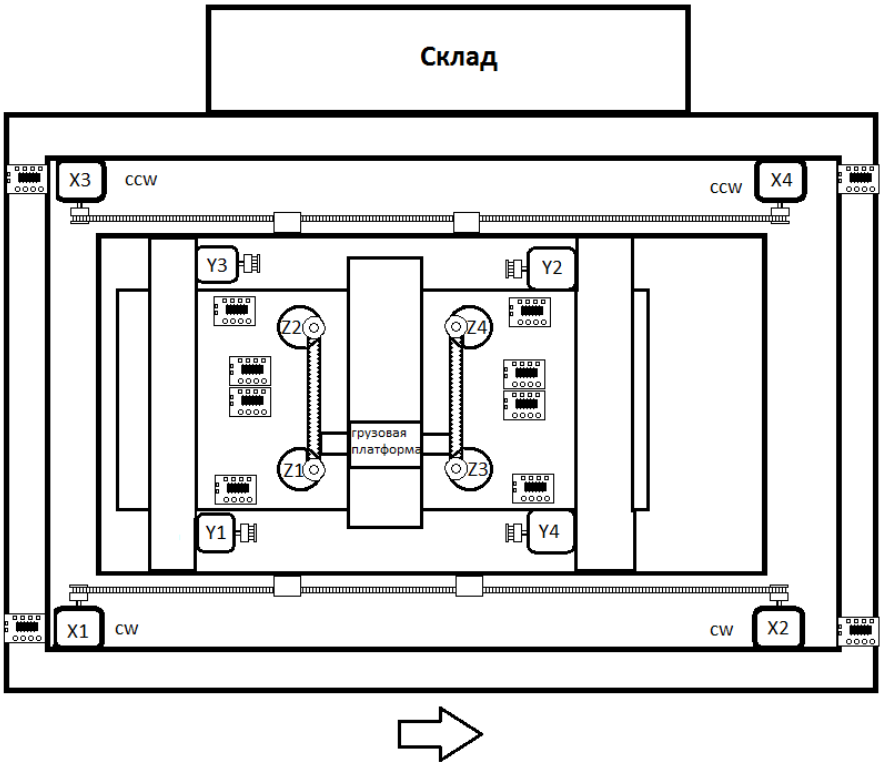


Рисунок 2.19 – Работа двигателей для перемещения платформы направо по горизонтали (оси X)

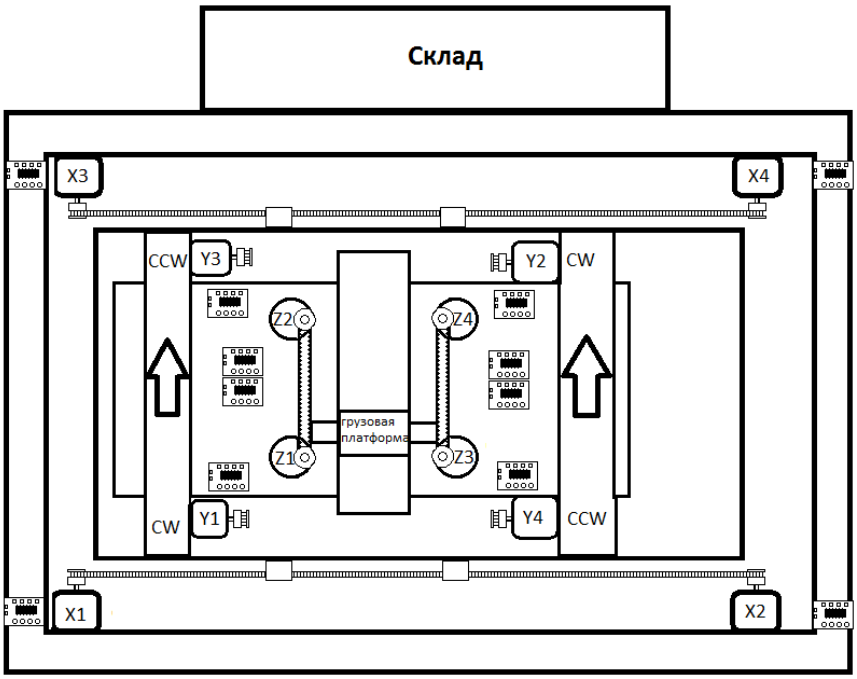


Рисунок 2.20 – Работа двигателей для перемещения платформы вверх по вертикали (оси Y)

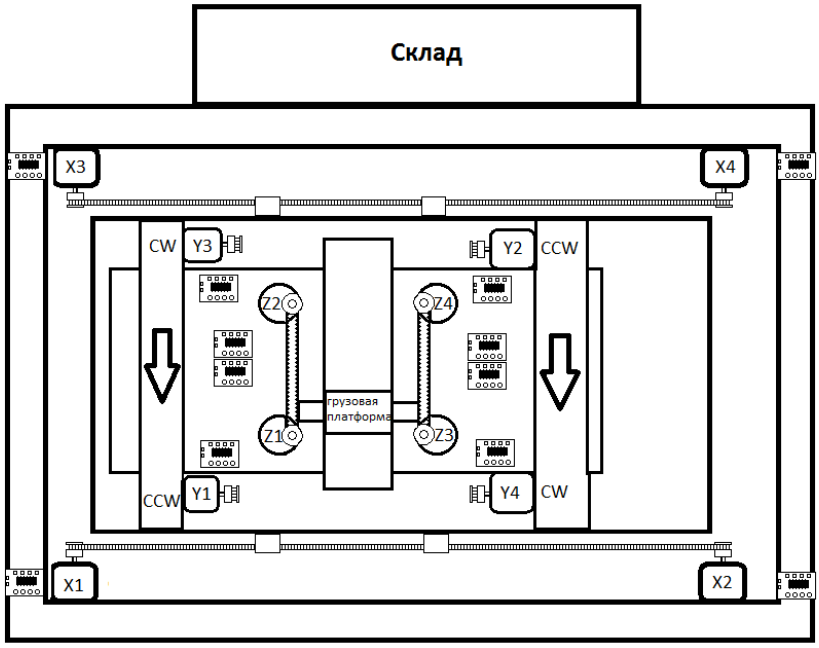


Рисунок 2.21 – Работа двигателей для перемещения платформы вниз по вертикали (оси Y)

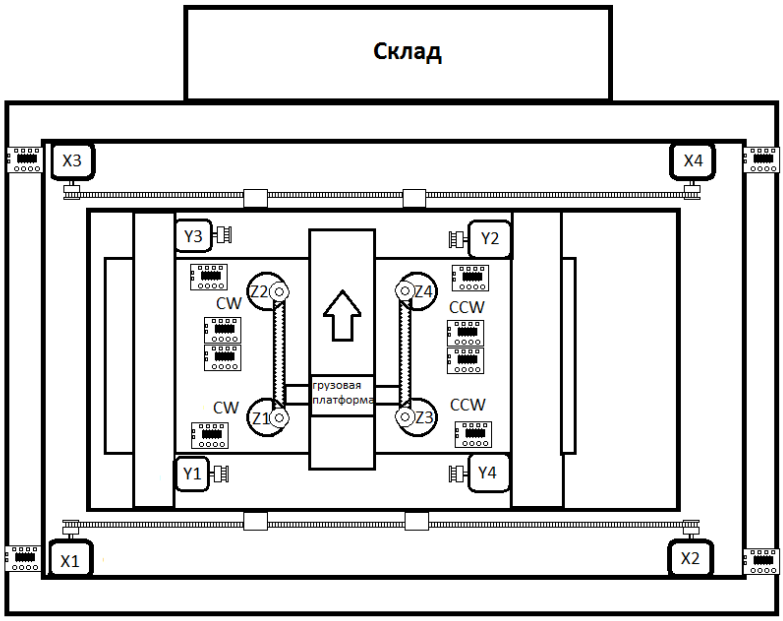


Рисунок 2.22 – Работа двигателей для перемещения погрузчика в ячейку (оси Z)

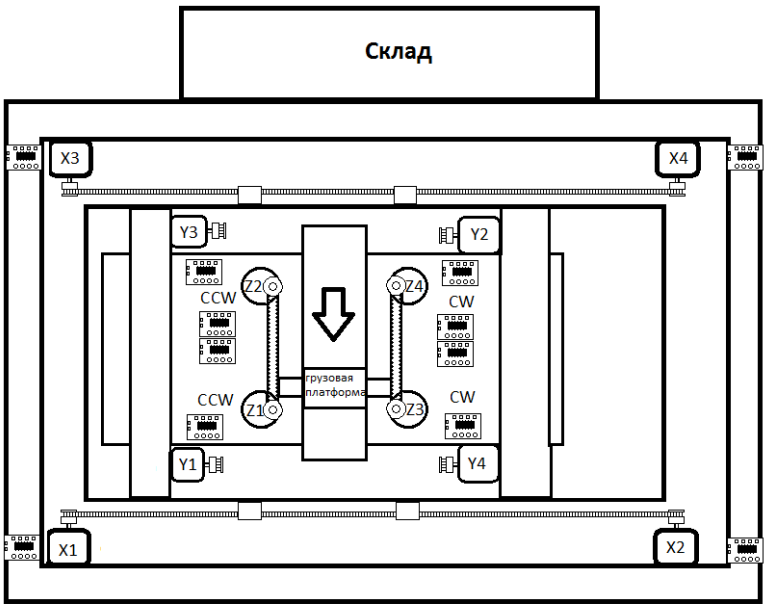


Рисунок 2.23 – Работа двигателей для перемещения погрузчика из ячейки (оси Z)

2.8 Принципиальная электрическая схема системы

Поскольку, это прототип системы, модули необходимо соединять между собой джамперами, а основой для макетирования будут выступать беспаечные макетные платы. Электрические схема представлены на рисунках 2.24 и 2.25.

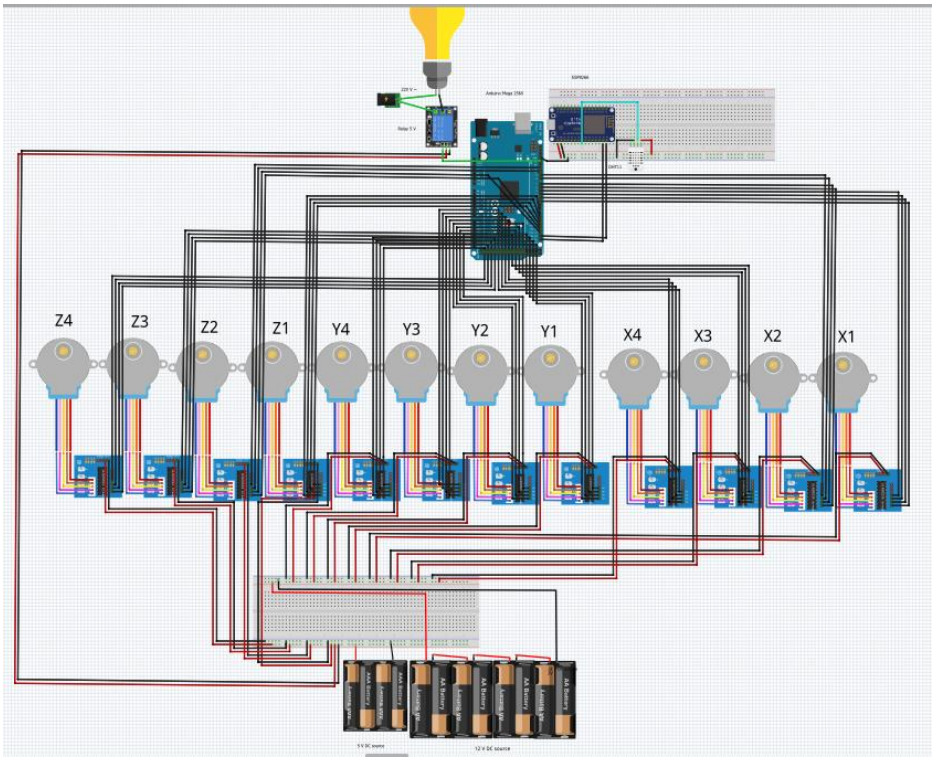


Рисунок 2.24 – Электрическая схема системы

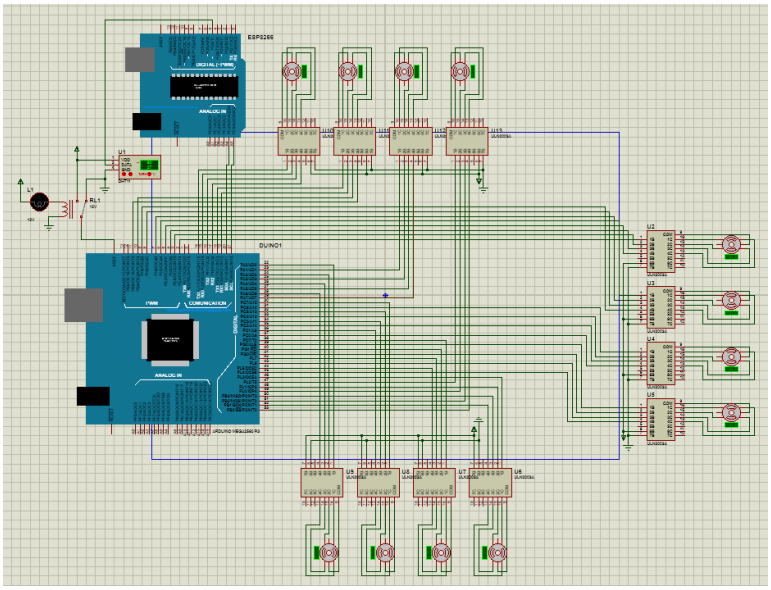


Рисунок 2.25 – Электрическая схема системы в Proteus

2.9 Структурная схема системы

Структурная схема системы показывает совокупность атомарных частей системы и связи между ними.

Под атомарными частями подразумевается компонент, выполняющий примитивные действия, и все действия происходят на одном уровне абстракции системы. Структурная схема показана на рисунке 2.26.

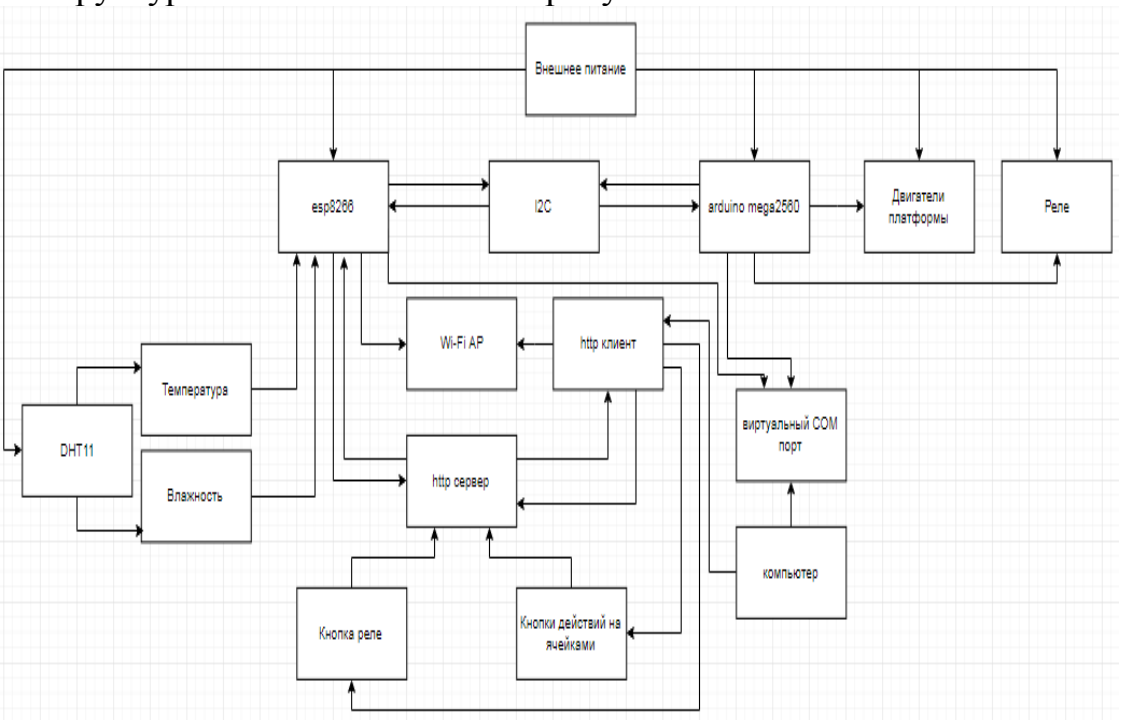


Рисунок 2.26 – Структурная схема системы

3 Программное обеспечение

3.1 Среда разработки

В данном проекте используется Arduino IDE. Эта среда разработки (интегрированная среда разработки) является частью онлайн-платформы Arduino Create, которая позволяет производителям писать код, получать доступ к учебным пособиям, настраивать платы и совместно использовать проекты. Разработанный для обеспечения непрерывного рабочего процесса, Arduino Create связывает точки между каждой частью пути производителя от вдохновения до реализации. Значит есть возможность управлять всеми аспектами вашего проекта прямо с единой панели. Основой язык программирования C++, но также есть возможность использовать JavaScript, Python, Lua, но в других средах. Основное окно показано на рисунке 3.1.

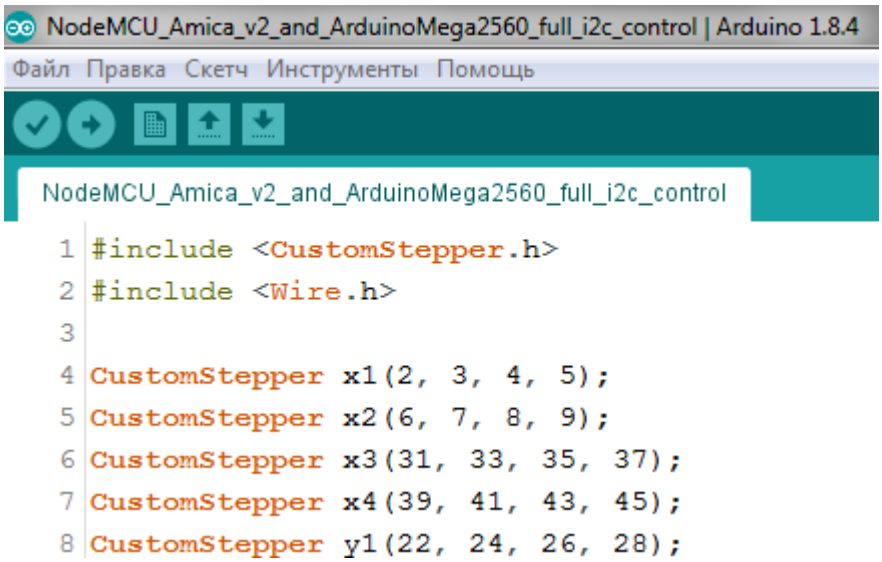


Рисунок 3.1 – Основное окно Arduino IDE

3.2 Используемые программные технологии

Так как работа проектируемой системы подразумевает работу в помещении и с множественным доступом, а также необходимо создать удобный графический интерфейс для пользователя, то было принято решение о создании веб-сервера для множественного доступа и графического интерфейса и использовании беспроводной технологии передачи данных по причине возможных изменений в конструкции (расширении склада) при которых необходима доступность функционирования автоматики во время реконструкции.

Ключевые методы управления всей системы автоматики работают благодаря стеку протоколов TCP/IP, стандарту IEEE 802.11 и протоколу прикладного уровня HTTP.

Протокол TCP / IP относится к четырехслойной концептуальной модели, известной как модель DARPA, названной в честь правительственного агентства США, которое первоначально разработало TCP / IP. Четыре уровня модели DARPA: Приложение, Транспорт, Интернет и Сетевой интерфейс. Каждый уровень в модели DARPA соответствует одному или нескольким слоям семиуровневой модели межсетевого взаимодействия Open Systems Interconnection (OSI).

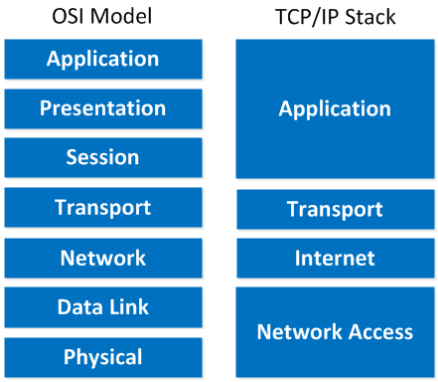


Рисунок 3.2 – TCP/IP стек и модель OSI

IEEE 802.11 представляет собой набор спецификаций управления доступом к среде передачи (MAC) и физического уровня (PHY) для осуществления компьютерной связи беспроводной локальной сети (WLAN) в частотных диапазонах 900 МГц и 2,4, 3,6, 5 и 60 ГГц. Они являются наиболее широко используемыми в мире стандартами беспроводной компьютерной сети, которые используются в большинстве домашних и офисных сетей, чтобы позволить ноутбукам, принтерам и смартфонам обмениваться данными друг с другом и получать доступ к Интернету без кабелей. Они создаются и поддерживаются Институтом инженеров по стандартизации LAN / MAN (IEEE 802) Института инженеров по электротехнике и электронике (IEEE 802).

HTTP функционирует как протокол запроса-ответа в вычислительной модели клиент-сервер. Например, веб-браузер может быть клиентом, а приложение, работающее на компьютере с веб-сайтом, может быть сервером. Клиент отправляет на сервер сообщение с HTTP запросом. Сервер, который предоставляет такие ресурсы, как HTML-файлы и другой контент или выполняет другие функции, возвращает ответное сообщение клиенту. Ответ содержит информацию о статусе завершения запроса и может также содержать запрошенный контент в теле сообщения.

Веб-браузер является примером пользовательского агента (UA). Другие типы пользовательского агента включают программное обеспечение индексирования, используемое поисковыми провайдерами (веб-сканерами), голосовые браузеры, мобильные приложения и другое программное обеспечение, которое потребляет или отображает веб-контент.

HTTP предназначен для того, чтобы позволить промежуточным сетевым элементам создавать связь между клиентами и серверами. Веб-сайты с высоким трафиком часто используют серверы веб-кешей, которые доставляют контент от имени серверов, работающих на исходящем канале, для улучшения времени отклика. Кэш веб-браузеров представляет собой данные полученные ранними обращениями к веб-ресурсам и повторно использует их, когда это возможно, для снижения сетевого трафика к серверу назначения. Прокси-серверы HTTP на границах частной сети могут облегчать обмен информацией для клиентов без глобального маршрутизируемого адреса путем ретрансляции сообщений с внешних серверов.

HTTP - это протокол уровня приложения, разработанный в рамках пакета интернет-протокола. Его определение предполагает базовый и надежный протокол транспортного уровня, и протокол управления передачей (TCP). Однако HTTP может быть адаптирован для использования ненадежных протоколов, таких как протокол пользовательских дейтаграмм (UDP), например, в HTTPU и Simple Service Discovery Protocol (SSDP).

Ресурсы HTTP идентифицируются и размещаются в сети посредством Uniform Resource Locators (URL), используя схемы Uniform Resource Identifiers (URI) http и https. URI и гиперссылки в документах HTML образуют взаимосвязанные гипертекстовые документы[18].

HTTP / 1.1 - это ревизия исходного HTTP (HTTP / 1.0). В HTTP / 1.0 для каждого запроса ресурса создается отдельное соединение с тем же сервером.

HTTP / 1.1 может повторно использовать соединение несколько раз для загрузки изображений, скриптов, таблиц стилей и т. Д. После того, как страница была доставлена. Таким образом, сообщения HTTP / 1.1 отличаются меньшей задержкой, так как установление соединений TCP создает значительные накладные расходы. Схема сетевой архитектуры показана на рисунке 3.3.

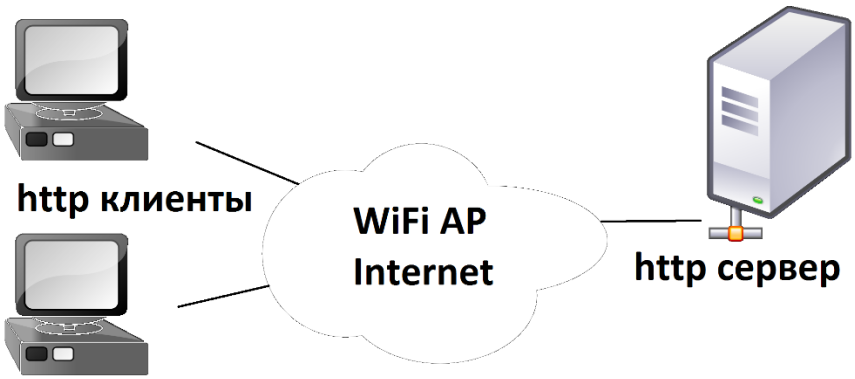


Рисунок 3.3 - Схема сетевой архитектуры системы автоматики

3.3 Блок-схемы программных модулей

На ниже представленных блок-схемах отображен алгоритм работы всей системы. На рисунках 3.4 и 3.5 показан алгоритм работы контроллера esp8266, а на рисунке 3.6 алгоритм работы контроллера Mega2560.

Рисунок 3.4 – Блок-схема для контроллера esp8266

Рисунок 3.5 – Блок-схема для контроллера esp8266 в случае события

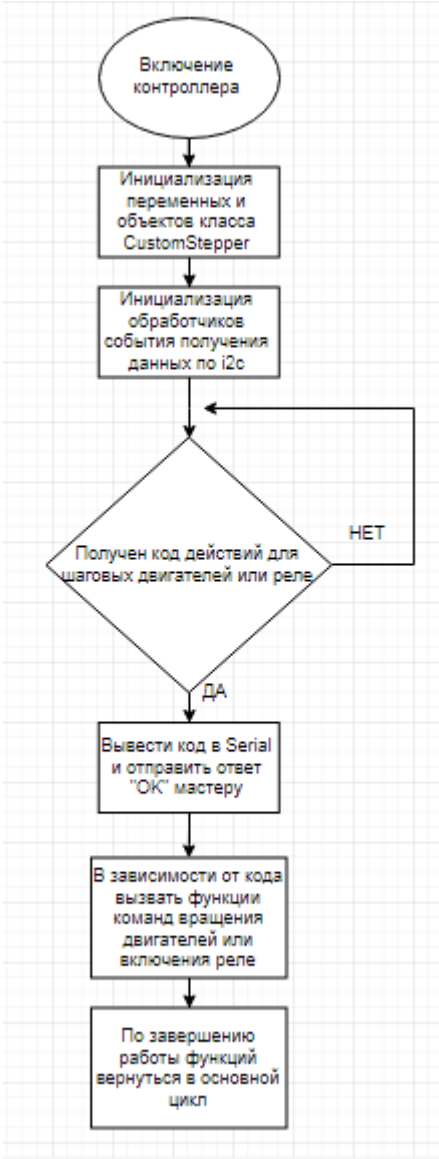


Рисунок 3.6 – Блок-схема для контроллера Mega 2560

3.4 Описание программы для микроконтроллера esp8266

Подключение статических библиотек для датчика температуры DHT11, интерфейса I2C, и реализации возможности работы по сети Wi-Fi, а также создания веб-сервера.

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Wire.h>
```

Инициализация констант, переменных и объектов класса ESP8266WebServer.

```
const char* ssid = "tihotspot";
```

```
const char* password = "nn810811";
DHT dht;
ESP8266WebServer server(80);
float humidity;
float temperature;
char operation;
int cell = 0;
```

В этой части будет инициализирован протокол I2C, и начнутся попытки подключения контроллера к точке доступа Wi-Fi.

```
void setup() {
  Serial.begin(9600);
  Wire.begin(D1, D2);
  dht.setup(D5);
  delay(100);
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
```

Инициализация методов класса server, с передачей им в качестве аргументов функций обратного вызова, к которым перейдёт исполнение после изменения корневого URL.

```
  server.on("/", handle_OnConnect);
  server.on("/put1", error);
  server.on("/take1", error);
  server.on("/put2", handle_put2);
  server.on("/take2", handle_take2);
  server.on("/put3", handle_put3);
  server.on("/take3", handle_take3);
  server.on("/put4", handle_put4);
  server.on("/take4", handle_take4);
  server.on("/put5", handle_put5);
  server.on("/take5", handle_take5);
  server.on("/put6", error);
  server.on("/take6", error);
```

```
  server.on("/put7", handle_put7);
  server.on("/take7", handle_take7);
  server.on("/put8", handle_put8);
  server.on("/take8", handle_take8);
  server.on("/put9", handle_put9);
  server.on("/take9", handle_take9);
  server.on("/put10", error);
  server.on("/take10", error);
  server.on("/put11", handle_put11);
  server.on("/take11", handle_take11);
  server.on("/put12", handle_put12);
  server.on("/take12", handle_take12);
  server.on("/relay", handle_relay);
  //  server.on("/ledon", handle_ledon);
  //  server.on("/ledoff", handle_ledoff);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("HTTP server started");
}
```

В этом цикле идёт опрос датчика температуры и влажности и выводятся в COM порт.

```
void loop() {
  server.handleClient();
  delay(dht.getMinimumSamplingPeriod());/* Delay of amount equal to sampling period */
  humidity = dht.getHumidity();/* Get humidity value */
  temperature = dht.getTemperature();/* Get temperature value */
  Serial.print(dht.getStatusString());/* Print status of communication */
  Serial.print("\t");
  Serial.print(humidity, 1);
  Serial.print("\t\t");
  Serial.print(temperature, 1);
  Serial.print("\t\t");
  Serial.println(dht.toFahrenheit(temperature), 1);/* Convert temperature to Fahrenheit units */
}
```

Вызов этой функции произойдёт в случае перехода на некорректный URL.

```
void error()
{
```

```
  Serial.println("Access denied");
}
```

При появления события нажатия кнопки “Relay”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_relay()
{
  operation = 'r';
  Serial.print("operation: ");
  Serial.print(operation);
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(222);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появления события нажатия кнопки “Put2”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put2() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-2");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(2);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
```

```
}
```

При появления события нажатия кнопки “take2”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take2() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print(22);
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(22);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put3”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put3() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print(3);
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(3);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
```

```
}
```

При появлении события нажатия кнопки “take3”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take3() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-3");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(33);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put4”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put4() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-4");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(4);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
```

```
}
```

При появлении события нажатия кнопки “take4”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take4() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-4");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(44);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put5”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put5() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-5");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(5);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
   Serial.print(c);
  }
```

```
}
```

При появлении события нажатия кнопки “take5”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take5() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-5");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(55);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put6”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put6() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-6");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(6);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
```

```
}
```

При появлении события нажатия кнопки “take6”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take6() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-6");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(66);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put7”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put7() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-7");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(7);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
```

```
}
```

При появлении события нажатия кнопки “take7”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take7() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-7");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(77);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “pu8”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put8() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-8");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(8);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);

while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
```

```
  }
}
```

При появлении события нажатия кнопки “take8”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take8() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-8");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(88);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put9”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put9() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-9");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(9);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
 while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
```

```
  }
}
```

При появлении события нажатия кнопки “take9”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take9() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-9");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(99);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put10”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put10() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-10");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(10);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
 while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
```

```
  }
}
```

При появлении события нажатия кнопки “take10”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take10() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-10");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(110);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put11”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put11() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-11");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(11);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
 while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
```

```
  }
}
```

При появлении события нажатия кнопки “take11”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take11() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-11");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(111);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

При появлении события нажатия кнопки “put12”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_put12() {
  operation = 'p';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-12");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(12);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
```

```
  }
}
```

При появлении события нажатия кнопки “take12”, вывести в COM порт код команды, и отправить по I2C второму контролеру числовое значение кода, отправить html строку и после принятия обратного ответа по I2C от контроллера вывести этот ответ в COM порт.

```
void handle_take12() {
  operation = 't';
  Serial.print("operation: ");
  Serial.print(operation);
  Serial.print("-12");
  Serial.println();
  server.send(200, "text/html", SendHTML());
  Wire.beginTransmission(8);
  Wire.write(112);
  Wire.endTransmission();
  Wire.requestFrom(8, 2);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}
```

Вызвать эту функцию когда контроллер подключится к точке доступа.

```
void handle_OnConnect() {
  server.send(200, "text/html", SendHTML());
}
```

Вызвать эту функцию, если указан неверный URL.

```
void handle_NotFound() {
  server.send(404, "text/plain", "Not found");
}
```

Эта функция вызовется в качестве функции обратного вызова после работы метода server.send(). Здесь с помощью тэгов html и стилей css написан пользовательский интерфейс, который видно на стороне клиента.

```
String SendHTML() {
  String ptr = "<!DOCTYPE html>\n";
  ptr += "<html>\n";
```

```
  ptr += "<head>\n";
  ptr += "<title>ASRS Control panel - WMS</title>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<h1>ASRS Control panel - WMS</h1>\n";
  ptr += "<p>Temperature: ";
  ptr += temperature;
  ptr += "</p>\n";
  ptr += "<p>Humidity: ";
  ptr += humidity;
  ptr += "</p>\n";
  ptr += "<p>Click to start operation</p>\n";
  ptr += "<form method=\"get\">\n";
  ptr += "<input type=\"button\" value=\"relay\"
onclick=\"window.location.href='/relay'\">";
  ptr += "</form>\n";
  ptr += "<form method=\"get\">\n";
  ptr += "<table style=\"width:50%\">";
  ptr += "<tr>";
    ptr += "<th><input type=\"button\" value=\"put9\"
onclick=\"window.location.href='/put9'\">";
    ptr += "<input type=\"button\" value=\"take9\"
onclick=\"window.location.href='/take9'\"></th>";
    ptr += "<th><input type=\"button\" value=\"put10\"
onclick=\"window.location.href='/error'\">";
    ptr += "<input type=\"button\" value=\"take10\"
onclick=\"window.location.href='/error'\"></th>";
    ptr += "<th><input type=\"button\" value=\"put11\"
onclick=\"window.location.href='/put11'\">";
    ptr += "<input type=\"button\" value=\"take11\"
onclick=\"window.location.href='/take11'\"></th>";
    ptr += "<th><input type=\"button\" value=\"put12\"
onclick=\"window.location.href='/put12'\">";
    ptr += "<input type=\"button\" value=\"take12\"
onclick=\"window.location.href='/take12'\"></th>";
  ptr += "</tr>";
    ptr += "<th><input type=\"button\" value=\"put5\"
onclick=\"window.location.href='/put5'\">";
    ptr += "<input type=\"button\" value=\"take5\"
onclick=\"window.location.href='/take5'\"></th>";
    ptr += "<th><input type=\"button\" value=\"put6\"
onclick=\"window.location.href='/error'\">";
    ptr += "<input type=\"button\" value=\"take6\"
onclick=\"window.location.href='/error'\"></th>";
```

```
  ptr += "<th><input type=\"button\" value=\"put7\" onclick=\"window.location.href='/put7'\">";
  ptr += "<input type=\"button\" value=\"take7\" onclick=\"window.location.href='/take7'\"></th>";
  ptr += "<th><input type=\"button\" value=\"put8\" onclick=\"window.location.href='/put8'\">";
  ptr += "<input type=\"button\" value=\"take8\" onclick=\"window.location.href='/take8'\"></th>";
 ptr += "</tr>";
  ptr += "<th><input type=\"button\" value=\"put1\" onclick=\"window.location.href='/error'\">";
  ptr += "<input type=\"button\" value=\"take1\" onclick=\"window.location.href='/error'\"></th>";
  ptr += "<th><input type=\"button\" value=\"put2\" onclick=\"window.location.href='/put2'\">";
  ptr += "<input type=\"button\" value=\"take2\" onclick=\"window.location.href='/take2'\"></th>";
  ptr += "<th><input type=\"button\" value=\"put3\" onclick=\"window.location.href='/put3'\">";
  ptr += "<input type=\"button\" value=\"take3\" onclick=\"window.location.href='/take3'\"></th>";
  ptr += "<th><input type=\"button\" value=\"put4\" onclick=\"window.location.href='/put4'\">";
  ptr += "<input type=\"button\" value=\"take4\" onclick=\"window.location.href='/take4'\"></th>";
 ptr += "</tr>";
 ptr += "</table>";
 ptr += "</form>\n";
 ptr += "</body>\n";
 ptr += "</html>\n";
 return ptr;
}
```

3.5 Описание программы для микроконтроллера Mega2560

Рассмотрим программу для микроконтроллера Mega2560.

Подключение библиотеки для управления шаговыми двигателями и использования однопроводного протокола I2C.

```
#include <CustomStepper.h>
#include <Wire.h>
```

Инициализация объектов класса CustomStepper, представляющих шаговые двигатели, переменных для отслеживания состояния работы двигателей и переменных для хранения значений, полученных по I2C.

```
CustomStepper x1(2, 3, 4, 5);
CustomStepper x2(6, 7, 8, 9);
CustomStepper x3(31, 33, 35, 37);
CustomStepper x4(39, 41, 43, 45);
CustomStepper y1(22, 24, 26, 28);
CustomStepper y2(30, 32, 34, 36);
CustomStepper y3(38, 40, 42, 44);
CustomStepper y4(46, 48, 50, 52);
CustomStepper z1(14, 15, 16, 17);
CustomStepper z2(18, 19, 10, 11);
CustomStepper z3(23, 25, 27, 29);
CustomStepper z4(47, 49, 51, 53);
int i2c_value = 0;
int State = 0;
int Degree = 0;
char mychar;
int mynumber;
int relay_state = 0;
const long interval = 100;
unsigned long previousMillis = 0;
```

Инициализация протокола I2C, и настройка скорости вращения двигателей, и количество шагов за оборот.

```
void setup()
{
  pinMode(12,OUTPUT);
  Serial.begin(9600);
  Wire.begin(8);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
  x1.setRPM(15);
  x1.setSPR(4000);
  x2.setRPM(15);
  x2.setSPR(4000);
  x3.setRPM(15);
  x3.setSPR(4000);
  x4.setRPM(15);
  x4.setSPR(4000);
  y1.setRPM(15);
```

```
  y1.setSPR(4000);
  y2.setRPM(15);
  y2.setSPR(4000);
  y3.setRPM(15);
  y3.setSPR(4000);
  y4.setRPM(15);
  y4.setSPR(4000);
  z1.setRPM(15);
  z1.setSPR(4000);
  z2.setRPM(15);
  z2.setSPR(4000);
  z3.setRPM(15);
  z3.setSPR(4000);
  z4.setRPM(15);
  z4.setSPR(4000);
}
```

В данном цикле идёт опрос состояния двигателей и в зависимости от того, остановлены ли они, вызвать необходимую функцию, в которой настраиваться угол поворота двигателя.

```
void loop()
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 850 и degree y равным 0, если полученное по I2C значение равно 2.

```
    if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
2)
    {
      Put_2_3_4(850, 0);
    }
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 850 и degree y равным 0, если полученное по I2C значение равно 22.

```
    else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
```

```
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 22)
  {
   Take_2_3_4(850, 0);
  }
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 1650 и degree y равным 0, если полученное по I2C значение равно 3.

```
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
     y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
     z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
3)
  {
   Put_2_3_4(1650, 0);
  }
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 1650 и degree y равным 0, если полученное по I2C значение равно 33.

```
  else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 33)
  {
   Take_2_3_4(1650, 0);
  }
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 2500 и degree y равным 0, если полученное по I2C значение равно 4.

```
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
     y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
     z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
4)
  {
   Put_2_3_4(2500, 0);
  }
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 2500 и degree y равным 0, если полученное по I2C значение равно 44.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 44)
{
  Take_2_3_4(2500, 0);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 0 и degree y равным 850, если полученное по I2C значение равно 5.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
    y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
    z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
5)
{
  Put_5_9(0, 850);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 0 и degree y равным 850, если полученное по I2C значение равно 55.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 55)
{
  Take_5_9(0, 850);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 850 и degree y равным 900, если полученное по I2C значение равно 6.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
    y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
    z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
6)
```

```
{
  Put_6_7_8_10_11_12(850, 900);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 850 и degree y равным 900, если полученное по I2C значение равно 66.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 66)
{
  Take_6_7_8_10_11_12(850, 900);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 1650 и degree y равным 900, если полученное по I2C значение равно 7.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
    y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
    z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
7)
{
  Put_6_7_8_10_11_12(1650, 900 );
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 1650 и degree y равным 900, если полученное по I2C значение равно 2277

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 77)
{
  Take_6_7_8_10_11_12(1650, 900);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 2500 и degree y равным 900, если полученное по I2C значение равно 8.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
    y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
    z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
8)
{
  Put_6_7_8_10_11_12(2500, 900);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 2500 и degree y равным 900, если полученное по I2C значение равно 88

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 88)
{
  Take_6_7_8_10_11_12(2500, 900);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 0 и degree y равным 1700, если полученное по I2C значение равно 9.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
    y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
    z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
9)
{
  Put_5_9(0, 1700);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 0 и degree y равным 1700, если полученное по I2C значение равно 99.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
        y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
        z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 99)
{
  Take_5_9(0, 1700);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 820 и degree y равным 1700, если полученное по I2C значение равно 10.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
   y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
   z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
10)
{
  Put_6_7_8_10_11_12(820, 1700);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 820 и degree y равным 1700, если полученное по I2C значение равно 110.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
      y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
      z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 110)
{
  Take_6_7_8_10_11_12(820, 1700);
}
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 1670 и degree y равным 1650, если полученное по I2C значение равно 11.

```
if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
   y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
   z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
11)
{
  Put_6_7_8_10_11_12(1670, 1650);
}
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 1670 и degree y равным 1650, если полученное по I2C значение равно 111.

```
else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
      y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
      z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 111)
```

```
  {
    Take_6_7_8_10_11_12(1670, 1650);
  }
```

Вызвать функцию перемещения по оси X и Y и установки груза в ячейку хранения, с аргументом degree x равным 2480 и degree y равным 1650, если полученное по I2C значение равно 12.

```
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
      y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
      z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value ==
12)
  {
    Put_6_7_8_10_11_12(2480, 1650);
  }
```

Вызвать функцию перемещения по оси X и Y и взятия груза из ячейки хранения, с аргументом degree x равным 2480 и degree y равным 1650, если полученное по I2C значение равно 112.

```
  else if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() &&
          y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() &&
          z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && i2c_value
== 112)
  {
    Take_6_7_8_10_11_12(2480, 1650);
  }
 }
```

Запустить двигатели, когда вызван метод rotateDegrees() и isDone() возвращает истину.

```
 x1.run();
 x2.run();
 x3.run();
 x4.run();
 y1.run();
 y2.run();
 y3.run();
 y4.run();
 z1.run();
 z2.run();
 z3.run();
 z4.run();
```

```
}
```

Когда получены данные по I2C, вывести их в COM порт.

```
void receiveEvent() {
  while (0 < Wire.available()) {
    i2c_value = Wire.read();
    Serial.print("i2c_value: ");
    Serial.print(i2c_value);
  }
  Serial.println();
```

Если данные равны 222, переключить реле и вывести в COM порт relay switch (состояние).

```
  if (i2c_value == 222)
  {
    switch(relay_state)
    {
      case 0: {digitalWrite(12,HIGH);
      relay_state = 1;
      Serial.println("relay switch 1");
      break;}
      case 1: {digitalWrite(12,LOW);
      relay_state = 0;
      Serial.println("relay switch 0");
      break;}
      }
  }
}
void requestEvent() {
  //Wire.write("ok");
}
```

Функция описывает последовательность действий для установки груза в 2, 3 и 4 ячейки, так как шаблон движения платформы один, различно только расстояния перемещения.

```
void Put_2_3_4(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 0){
    Load_in();
    State = 1;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 1){
    Lift_up(200);
```

```
    State = 2;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 2){
    Load_out();
    State = 3;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 3) {
    Move_right(Degree_x);
    State = 4; }
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && State == 4){
    Load_in();
    State = 5;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 5){
    Lift_down(200);
    State = 6;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 6){
    Load_out();
    State = 7;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 7){
    Move_left(Degree_x);
    i2c_value = 0;
    State = 0;}}
```

Функция описывает последовательность действий для взятия груза из 2, 3 и 4 ячейки, так как шаблон движения платформы один, различно только расстояния перемещения.

```
void Take_2_3_4(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && State == 0){
    Move_right(Degree_x);
    State = 1;}
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 1){
    Load_in();
    State = 2;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 2){
    Lift_up(200);
    State = 3;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 3){
    Load_out();
    State = 4;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 4){
    Move_left(Degree_x);
    State = 5;}
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && State == 5){
    Load_in();
```

```
    State = 6;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 6){
    Lift_down(200);
    State = 7;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 7){
    Load_out();
    i2c_value = 0;
    State = 0;}}
```

Функция описывает последовательность действий для установки груза в 5 и 9 ячейки, так как шаблон движения платформы один, различно только расстояния перемещения.

```
void Put_5_9(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 0){
    Load_in();
    State = 1;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 1) {
    Lift_up(200);
    State = 2;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 2){
    Load_out();
    State = 3;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 3){
    Lift_up(Degree_y);
    State = 4;}
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 4){
    Load_in();
    State = 5;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 5){
    Lift_down(200);
    State = 6;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 6){
    Load_out();
    State = 7;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 7){
    Lift_down(Degree_y);
    i2c_value = 0;
    State = 0;}}
```

Функция описывает последовательность действий для взятия груза из 2, 5 и 9 ячейки, так как шаблон движения платформы один, различно только расстояние перемещения.

```
void Take_5_9(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && State == 0){
    Lift_up(Degree_y);
    State = 1; }
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 1){
    Load_in();
    State = 2;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 2){
    Lift_up(200);
    State = 3;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 3){
    Load_out();
    State = 4;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 4){
    Lift_down(Degree_y);
    State = 5;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 5){
    Load_in();
    State = 6;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 6){
    Lift_down(200);
    State = 7;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 7){
    Load_out();
    i2c_value = 0;
    State = 0;}}
```

Функция описывает последовательность действий для установки груза в 6,7,8,10,11 и 12 ячейки, так как шаблон движения платформы один, различно только расстояние перемещения.

```
void Put_6_7_8_10_11_12(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 0){
    Load_in();
    State = 1;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 1){
    Lift_up(200);
    State = 2;}
```

```
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 2){
    Load_out();
    State = 3;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 3){
    Move_right(Degree_x);
    Lift_up(Degree_y);
    State = 4; }
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 4){
    Load_in();
    State = 5;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 5){
    Lift_down(200);
    State = 6;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 6){
    Load_out();
    State = 7;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 7){
    Move_left(Degree_x);
    Lift_down(Degree_y);
    i2c_value = 0;
    State = 0;}}
```

Функция описывает последовательность действий для взятия груза из 6, 7,8,9,10,11 и 12 ячейки, так как шаблон движения платформы один, различно только расстояние перемещения.

```
void Take_6_7_8_10_11_12(int Degree_x, int Degree_y){
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && State == 0){
    Move_right(Degree_x);
    Lift_up(Degree_y);
    State = 1;}
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 1){
    Load_in();
    State = 2;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 2){
    Lift_up(200);
    State = 3;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 3){
    Load_out();
    State = 4;}
  if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 4){
    Move_left(Degree_x);
```

```
    Lift_down(Degree_y);
    State = 5;}
  if (x1.isDone() && x2.isDone() && x3.isDone() && x4.isDone() && y1.isDone()
&& y2.isDone() && y3.isDone() && y4.isDone() && State == 5){
    Load_in();
    State = 6;}
 if (z1.isDone() && z2.isDone() && z3.isDone() && z4.isDone() && State == 6){
    Lift_down(200);
    State = 7;}
  if (y1.isDone() && y2.isDone() && y3.isDone() && y4.isDone() && State == 7){
    Load_out();
    i2c_value = 0 ;
    State = 0;}}
```

Функция выполняет действия по перемещению платформы по горизонтальной оси x в правом направлении. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Move_right(int Degree)
{
  x1.setDirection(CW);
  x1.rotateDegrees(Degree);
  x2.setDirection(CW);
  x2.rotateDegrees(Degree);
  x3.setDirection(CCW);
  x3.rotateDegrees(Degree);
  x4.setDirection(CCW);
  x4.rotateDegrees(Degree);
}
```

Функция выполняет действия по перемещению платформы по горизонтальной оси x в левом направлении. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Move_left(int Degree)
{
  x1.setDirection(CCW);
  x1.rotateDegrees(Degree);
  x2.setDirection(CCW);
  x2.rotateDegrees(Degree);
  x3.setDirection(CW);
  x3.rotateDegrees(Degree);
  x4.setDirection(CW);
  x4.rotateDegrees(Degree);
}
```

Функция выполняет действия по взятию груза из ячейки. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Load_in()
{
  z1.setDirection(CW);
  z1.rotateDegrees(650);
  z2.setDirection(CW);
  z2.rotateDegrees(650);
  z3.setDirection(CCW);
  z3.rotateDegrees(650);
  z4.setDirection(CCW);
  z4.rotateDegrees(650);

}
```

Функция выполняет действия по установке груза в ячейку. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Load_out()
{
  z1.setDirection(CCW);
  z1.rotateDegrees(650);
  z2.setDirection(CCW);
  z2.rotateDegrees(650);
  z3.setDirection(CW);
  z3.rotateDegrees(650);
  z4.setDirection(CW);
  z4.rotateDegrees(650);

}
```

Функция выполняет действия по перемещению платформы по вертикальной оси y наверх. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Lift_up(int Degree)
{
  y1.setDirection(CW);
  y1.rotateDegrees(Degree);
  y2.setDirection(CW);
  y2.rotateDegrees(Degree);
  y3.setDirection(CCW);
  y3.rotateDegrees(Degree);
  y4.setDirection(CCW);
  y4.rotateDegrees(Degree);
```

```
}
```

Функция выполняет действия по перемещению платформы по вертикальной оси y вниз. Здесь устанавливается направление и угол поворота для каждого двигателя.

```
void Lift_down(int Degree)
{
  y1.setDirection(CCW);
  y1.rotateDegrees(Degree);
  y2.setDirection(CCW);
  y2.rotateDegrees(Degree);
  y3.setDirection(CW);
  y3.rotateDegrees(Degree);
  y4.setDirection(CW);
  y4.rotateDegrees(Degree);
}
```

4 Безопасность жизнедеятельности

В процессе монтажа системы в помещении склада или любого другого помещения, включая жилые, необходимо создать безопасные и комфортные условия для монтажа и пуско-наладочных работ. Далее будут рассмотрены и рассчитаны различные методы безопасности и эксплуатации электротехнического оборудования.

А также необходимо будет рассмотреть различные стандарты по безопасности жизнедеятельности и охране труда, включающие в себя справочные материалы, методы расчёта и перечни используемых средств и инструментов для безопасной работы как во время монтажа, пуско-наладочных работ, так и во время эксплуатации и предотвращения появления критических сбоев системы.

4.1 Анализ условий труда персонала при монтаже технического оборудования

Главной целью этого дипломного проекта является разработка автоматизированной системы. Но так как эту разработку необходимо где-то применять, её соответственно необходимо для этого установить. Есть 2 варианта места использования такой системы. Это может быть промышленный склад, достигающий размеров в несколько сотен, а может и тысяч квадратных метров и небольшой склад частной компании, с максимально возможной площадью в 200 квадратных метров.

Анализ труда монтажников будет основываться на работе в условиях небольшого склада, который может являться частью магазина, центра дистрибуции или складов с долгим промежутком хранения, где хранится товары не пищевого назначения. Как правило, в подобных помещения работает немного людей, что значит условия труда необходимо заранее просчитать по сравнению с работой под открытым небом, где требования к безопасности могут кардинально различаться. Отсюда вытекает следствие, что условия будут следующими: наличие противопожарной безопасности, искусственного освещения и вентиляции воздуха.

Рабочие связаны с работой с микроэлектроникой и механическим электрооборудованием. Отсюда выбраны микроклиматические условия:

- оптимальная температура 22-24 С, диапазон температур от 18 до 26;
- относительная влажность воздуха от 40 до 60 %;
- скорость движения воздуха 0,1 м/с.

Схема складского помещения при монтаже системы показана на рисунке 4.1. На рисунке 4.1 обозначены: 1 – Рабочая область монтажа; 2 – входная дверь; 3 – Электрический щиток; 4 – Вентиляционная шахта; 5 – Пожарный щиток; 6 – Первое окно; 7 – Второе окно;

4.2 Условия безопасности при пайке электронных компонентов системы

При монтаже системы необходимо провести паяльные работы с проводами питания и проводами коммуникации контроллеров и исполнительных механизмов.

При работе пайки и лужении необходимо точно выполнять меры техники безопасности. Несоблюдение данных нижеописанных правил может привести к последствиям, влияющим на здоровье работника.

Необходимо заметить, что к данному процессу допускаются только совершеннолетние лица и прошедшие соответствующую подготовку. Ниже представлен краткие правила при таких работах.

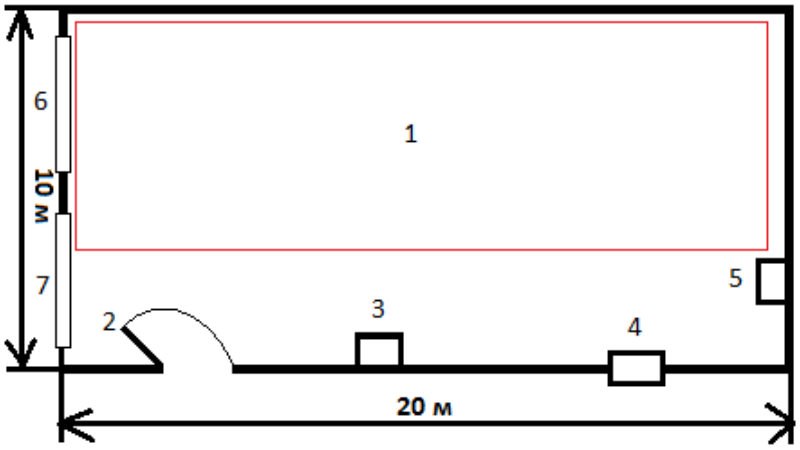


Рисунок 4.1 – Схема складского помещения при монтаже системы

Паяльник - никогда не прикасаться к нагревательному элементу паяльника, температура доходит до 400 ° С. Держать провода нагретыми с помощью пинцета или зажимов. Во время использования держать чистящую губку влажной. Всегда возвращать паяльник на подставку, когда он не используется. Никогда не класть его на рабочий стол. Выключать устройство и отсоединяйте его от сети, когда он не используется.

Припой, флюс и очистители - надевать защиту для глаз. Припой может брызнуть от моментального нагрева и попасть в глаза Использовать, по возможности, без канифоли и бессвинцовые припои. Продолжать чистить растворители в раздаточных бутылках. Всегда мыть руки с мылом и водой после пайки.

Работа со свинцом - свинец может вызвать серьезные хронические последствия для здоровья. Воздействие будет происходить, прежде всего, при случайном проглатывании. Носить перчатки, если будет работа с припоем. При пайке могут образовываться пары. Флюс генерирует видимые испарения, возникающие при пайке. Воздействие канифоли может вызвать раздражение глаз, гортани и легких, кровотечения из носа и головные боли.

Повторное воздействие может вызвать респираторную и кожную сенсибилизацию, вызывая и усугубляющую астму. Припой со свинцом является серьезной опасностью для здоровья.

Контроль испарений - пайка с использованием канифоли разрешается только в строго контролируемых условиях, где нет эффективной альтернативы. Извлечение дыма должно осуществляться через вытяжку (предпочтительно). В идеале дым должен выходить наружу. Наконечники для отбора, которые используют фильтры, должны включать фильтры с активированным углем.

Системы вытяжных фильтров для настольных фильтров могут использоваться для пайки без канифоли в хорошо проветриваемых помещениях (т.е. в больших объемах рабочего пространства). Размещение их важно для выполнения безопасных работ.

Все системы вытяжки должны тестироваться не реже одного раза в год и поддерживать т.е. регулярно менять фильтры. Содержать журнал изменений фильтра или отмечать дату на самом фильтре / системе.

Обучение и контроль - руководители / линейные менеджеры должны информировать пользователей о рисках при пайке. Руководители / линейные менеджеры должны следить за тем, чтобы элементы управления были установлены и работали, и что они используются правильно. Все эти процессы должны быть изучены и контролироваться надлежащим образом.

Здравоохранение - все эти паяльные работы должны часто (т.е. более одного раза в неделю) проверяться на использование канифоли, содержащей припои и следует направлять в управление безопасности для наблюдения за состоянием здоровья рабочих.

Электрическая безопасность - использовать паяльники, которые имеют очевидный ущерб для тела, кабелей питания или электроники. В течение двенадцати месяцев все паяльники должны быть испытаны на электробезопасность.

Хранить паяльную станцию без подключения электрических кабелей к сети, чтобы предотвратить повреждение нагреваемого наконечника. При коротком замыкании использовать заземленную розетку и заземляющий контакт. Работайте на огнестойкой поверхности. Носить огнестойкую одежду, которая закрывает руки и ноги, чтобы предотвратить случайные ожоги. Необходимо знать, где находится ближайший огнетушитель и как его использовать (служба безопасности может организовать обучение).

Первая помощь - немедленно помещать любые ожоги под холодную воду в течение 15 минут.

Отходы - соберите припой в контейнере с крышкой. Замените крышку, когда она не используется. Соблюдайте этикетку и утилизируйте ее как опасные отходы. Использованные паяные губки и загрязненные тряпки должны быть помещены в герметичный мешок для удаления в качестве опасных отходов.

В данном проекте будет использоваться паяльная станция Lukey 853D, показанная на рисунке 4.2.

Технические параметры:
- тип нагревательного элемента: керамический;
- мощность паяльника: 50 Вт;
- минимальная рабочая температура паяльника: 200°С;
- максимальная рабочая температура паяльника: 480°С;
- тип подключения паяльника: отсоединяемый;
- мощность фена: 350 Вт;
- автоматическое отключение фен: есть;
- тип нагнетателя воздуха: турбина;
- максимальная рабочая температура фена: 480°С;
- датчик температуры: есть;
- управления станцией: цифровое;
- датчик воздушного потока фена: есть.

Рисунок 4.2 - Паяльная станция Lukey 853D

4.3 Расчёт искусственного освещения в помещении

Так как складское помещение возможно не имеет либо имеет в недостаточном количестве естественного освещения, то необходимо заранее просчитать возможность реализации искусственного освещения. В качестве примера будет рассмотрен вариант помещения с недостаточным естественным освещением, что потребуется создать дополнительные источники света для максимально комфортной работы работников. Во время монтажа системы, монтажникам необходимо полноценное освещение, но так как система автоматизации состоит из микроэлектронных компонентов, необходимо дополнительное освещение, чтобы видеть все соединения плат и контроллеров.

Складское помещение имеет естественное освещение через боковые окна, и искусственное освещение, что даёт возможность проводить работы

ночью и днем в помещении, при котором уровень КЕО не соответствует заявленным требованиям.

Таким образом, для расчета общего освещения помещения, длиной равной 20 метров, шириной равной 10 метров и высотой равной 3 метрам с белым потолком, стены в светло-сером цвете и прикрытыми жалюзи окнами. По стандарту монтажная работа является третьим уровнем высокой точности. Нормируемая освещенность 250 лк. Для складского помещения будет использоваться светодиодные лампы LED TQ 15W, мощностью 15 Вт. Световым потоком 2500 люмен, диаметром 500 мм и длиной с контактами 500 мм. Угол свечения 180 градусов. Эквивалентом является люминесцентная лампа на 120 Вт. Светодиодная лампа LED TQ 15W показана на рисунке 4.3.

Рисунок 4.3 - Светодиодная лампа LED TQ 15W

Для расчёта коэффициента использования необходимо определить индекс помещения:

$$i = \frac{A*B}{h*(A+B)}, \quad (4.1)$$

где A и B – длина и глубина (ширина) помещения, м;

h – высота подвеса светильников, м.

$$h = H - h_{рп} - h_{сл}, \quad (4.2)$$

где h - высота подвеса, м;

$h_{рп}$ - высота рабочих поверхностей, м;

$h_{сл}$ - высота свеса ламп, м;

H - высота помещения, м.

Высота рабочей поверхности равняется 0 м, высота свеса лампы 0,5м а высота помещения 3м.

$$h = 3 – 0 – 0,5 = 2,5 \text{ (м)}.$$

Индекс помещения:

$$i = \frac{10*20}{3*(10+20)} = 2.23.$$

Определение необходимого расстояния между светильниками:

$$L = \lambda * h, \quad (4.3)$$

где λ = 1.4;

h - высота подвеса светильника над рабочей областью.

В результате получим расчёт:

$$L = 1.4 * 2{,}5 = 3{,}5(\text{м}).$$

Определим количество светильников

$$N = \frac{E*K_З*S*Z}{n*\Phi_л*\eta}, \quad (4.4)$$

где S – площадь склада, S = 200 м^2;

$K_З$ –коэффициент запаса, показаны в таблице 4.1, $K_З = 1$;

E – заданная минимальная освещённость, E = 250 лк;

Z – коэффициент неравномерности освещения, Z = 1;

n – количество ламп в светильнике, n = 2;

$\Phi_л$ – световой поток лампы, $\Phi_л$ = 2500 лм;

η – коэффициент использования, значения взято из таблицы 4.2, η = 0.65.

Для определения коэффициента использования светового потока h находят индекс помещения i и предполагаемые коэффициенты отражения поверхностей помещения: потолка rп, стен rс, пола rр.

Обычно для светлых административно-конторских помещений: rп=70%, rс=50%, rр = 30%. Расположение показано на рисунке 4.4.

Таблица 4.1 – Коэффициенты запаса

Помещения	Примеры	Коэффициент запаса		
		Газоразрядные лампы	Лампы накаливания	Светодиодные лампы
Запыленность свыше 5 мг/м3	Цементные заводы, литейные цеха и т. п.	2	1.7	1.5
Дым, копоть 1-5 мг/м3	Кузнечные, сварочные цеха и т. п	1.8	1.5	1.3

Продолжение таблицы 4.1

Менее 1 мг/м3 Значительная концентрация паров кислот и щелочей	Инструментальные, сборочные цеха Цеха химических заводов, гальванические цеха	1.5-1.8	1.3	1.1
Запыленность значительно менее 1 мг/м3, отсутствие паров кислот и щелочей	Жилые, административные и офисные и т.п. помещения	1.4	1.5	1

Определим количество светильников по формуле 4.4:

$$N = \frac{250 * 1 * 200 * 1}{2 * 2500 * 0.65} = \frac{50000}{3250} = 15.$$

Таблица 4.2 – Определение коэффициента использования светового потока

Индекс помещения, I	rп , % 70 rс , % 50 rр , % 30	50 30 10	30 10 10
10,5	28	21	18
1,0	49	40	36
3,0	73	61	58
5,0	80	67	65

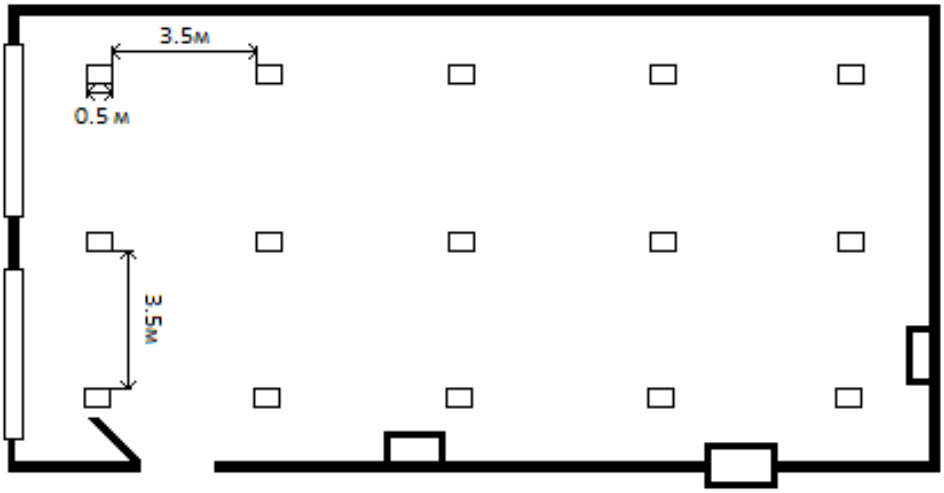


Рисунок 4.4 – Расположение светильников в складском помещении

4.4 Выводы по разделу безопасности жизнедеятельности

Проанализировав решения по безопасности жизнедеятельности, я пришёл к выводу, что проведённые мероприятия по усовершенствованию места работы рабочих при монтаже автоматизированной системы способствуют улучшению монтажных работ, при которых требуется высокая точность работы и реализация возможности пайки электронных компонентов. В связи с требованием к работе с элементами малых размеров, появилась необходимость дополнительного искусственного освещения, путём изменения ламп накаливания на светодиодные лампы, увеличении их количества, уменьшения потребляемой мощности, так как подразумевается работа автоматизированного склада круглосуточно и без перерывов. В ходе анализа требований к безопасности, было выяснено, что создание безопасности на рабочем месте необходимо создать только во время монтажа автоматизированной системы. При дальнейшей эксплуатации автоматики, вся ответственность по обеспечению безопасности жизнедеятельности будет переложена на компанию-покупателя автоматизированного склада.

5 Расчётно-экономическое обоснование

5.1 Расчёт капитальных вложений на реализацию проекта

Общая стоимость капитальных инвестиций на реализацию этого дипломного проекта в единичном виде включает в себя расходы на покупку оборудования, инструментов, транспортировку и монтаж, расходные материалы и разработку ПО.

$$К_{рп} = К_{зо} + К_{по} + К_{тм}, \quad (5.1)$$

где $К_{рп}$ – капиталовложения на реализацию проекта;

$К_{зо}$ - капиталовложения на закупку оборудования;

$К_{тм}$- капиталовложения на транспортировку и монтаж;

$К_{по}$- капиталовложения на разработку программного обеспечения.

5.2 Расчёт капиталовложений на закупку оборудования

Под расходами на оборудование понимается покупка исполнительных механизмов, управляющих контроллеров и расходных материалов, показанных в следующей таблице 5.1.

Таблица 5.1 – Перечень закупаемого оборудования

Наименование	Количество, штук	Цена за единицу (тг.)	Итого (тг.)
NodeMCU Devkit v2	1	1200	1200
Arduino Mega2560 R3	1	5000	5000
Модуль реле 5v	1	400	400
Датчик DHT11	1	300	300
Макетная плата	2	300	600
Люминесцентная лампа	1	1500	1500
Шаговый двигатель 28byj-48 12v	8	400	3200
Шаговый двигатель 28byj-48 5v	4	400	1600
Шкив 5мм, 20х	16	400	6400
Ремень 5мм, 1м	5	500	2500
Металлические направляющие (конструкция), 1м	26	300	7800
Шлейф из проводов	6	400	2400
Роутер TL-WR720N	1	3000	3000
Патч-корд	1	1000	1000

Продолжение таблицы 5.1

Блок питания 12v 16A	1	3000	3000
Блок питания 5v 10A	1	1500	1500
Паяльная станция	1	5000	5000
Набор инструментов	1	20000	20000
Набор винтов, гаек, свёрл, припой, флюс	1	3000	3000
Итого			69400

Расходы на транспортировку и монтаж займут до 30% от расходов на оборудование, т.е. 69400*0,3=20820тг.

$$К_{тм} = К_{зо} * 0{,}3. \quad (5.2)$$

5.3 Расчёт капиталовложений на разработку программного обеспечения

Расчёт капиталовложений на разработку программного обеспечения рассчитывается по следующей формуле:

$$К_{по} = З_{фот} + О_{сн} + П_{спо} + Р_{ам} + Р_{ээ} + Р_{нак}, \quad (5.3)$$

где $З_{фот}$ – Фонд оплаты труда, предназначенный для разработчика программного обеспечения, тг;

$О_{сн}$ – Отчисления по социальному налогу, тг;

$П_{спо}$ –Покупка стороннего программного обеспечения, тг;

$Р_{ам}$ – Расходы на амортизацию техники разработчиков ПО, тг;

$Р_{ээ}$ –Расходы на электроэнергию, тг;

$Р_{нак}$ –Накладные расходы, тг.

Объём фонда оплаты труда, предназначенный для оплаты услуг разработчиков $З_{фот}$, определяется по формуле:

$$З_{фот} = З_{осн} + З_{доп}, \quad (5.4)$$

где $З_{осн}$- основная заработная плата, тг;

$З_{доп}$- дополнительная заработная плата, тг.

Основной оклад разработчиков ПО рассчитывается по формуле:

$$З_{осн} = t_{тр} * З_{дн}, \quad (5.5)$$

где $t_{тр}$ – трудоемкость разработки программного продукта (чел/дни);

$З_{дн}$ – дневная заработная плата разработчика (тенге).

Затраты на заработную плату зависят от трудоемкости работ[19].

Расчет трудоемкости ($t_{тр}$) будут проводиться путем сложения затрат труда по отдельным периодам разработки:

$$t_{тр} = t_{оп} + t_{алг} + t_{бл} + t_{пр} + t_{отл} + t_{док}, \quad (5.6)$$

где $t_{оп}$ - предварительная подготовка описания задачи, чел/час;
$t_{алг}$ - разработка алгоритма решения задачи, чел/час;
$t_{бл}$ - составление блок-схемы алгоритма; чел/час;
$t_{пр}$ - программирование, чел/час;
$t_{отл}$ - отладка программы на ЭВМ, чел/час;
$t_{док}$ - подготовка документации, чел/час.

При определении трудоемкости прибегают к использованию ключевого показателя, такого как примерное или условное количество команд операторов используемых в разрабатываемом ПО, обозначается символом «Q» и находится по формуле:

$$Q = q * c, \quad (5.7)$$

где Q – условное число операторов;

q – предполагаемое число операторов зависит от типа, значения данного коэффициента приведены в таблице 3;

c – это коэффициент, учитывающий сложность программы и ее новизну (таблица 5.2).

Таблица 5.2 – Предполагаемое число операторов «q»

Тип задачи	Пределы изменений коффициента
Задачи учета	от 1400 до 1500
Задачи оперативного	от 1500 до 1700
Задачи планирования	от 3000 до 3500
Многовариантные	от 4500 до 5000
Комплексные задачи	от 5000 до 5500

По степени новизны ПП делятся на четыре группы:

а) разработка принципиально новых задач (группа А);

б) разработка оригинальных программ (группа Б);

в) разработка программ с использованием типовых решений (группа г) разовая типовая задача (группа Г).

Таблица 5.3 – Коэффициент сложности и новизны «с»

Язык программирования	Группа сложности	Степень новизны			
		А	Б	В	Г
Высокого уровня	1	1,38	26	5	9
	2	1,30	19	8	5
	3	1,20	10	0	0
Низкого уровня	1	1,58	45	2	9
	2	1,49	37	4	4
	3	1,38	26	5	9

Программное обеспечение, рассматриваемое, в данном проекте написано на языке высокого уровня с использованием типовых решений.

Расчет условного количества команд операторов согласно формуле(5.8):

$$Q = 1200 * 1,2= 1440. \quad (5.8)$$

Далее определяем время, которое потребуется на создание ПО на каждом этапе.

Время, необходимое на предварительную подготовку описания задачи $t_{оп}$ (фактическое время) в среднем от 3-х до 5-ти дней по 8 часов:

$$t_{оп} = 25 \text{ чел / час}. \quad (5.9)$$

Время необходимое на разработку алгоритма решения задачи $t_{алг}$ рассчитываем по формуле:

$$t_{алг} = Q / (50 \times k), \quad (5.10)$$

где k – коэффициент, характеризующий квалификацию программиста по опыту его работы, выбирается из таблицы 5.4.

Таблица 5.4 – Коэффициент характеризующий квалификацию программиста

Опыт работы	Коэффициент квалификации
До двух лет	0,8
2-3 года	1
3-5 лет	1,1 – 1,2
5-7 лет	1,3 – 1,4
более 7 лет	1,5 – 1,6

Время необходимое для разработки алгоритма рассчитанное по

формуле (5.11):

$$t_{алг} = 1440 / (50 *0,8) = 36 \text{ чел/час.}$$

Время необходимое для разработки блок-схемы tбл определяется таким же образом как tалг в соответствии с формулой (5.10):

$$T_{бл} = 1440 / (50 *0,8) = 36 \text{ чел/час.}$$

Время потраченное непосредственно на написание программы на языке высокого уровня tпр:

$$t_{пр} = Q *1,5 / (50 * k). \quad (5.11)$$

Соответственно время, потраченное на написание программы по формуле (5.11) равно:

$$tпр = 1440 *1,5 / (50 *0,8) = 54 \text{ чел/час.}$$

Время для отладки и тестирования программы :

$$t_{отл} = Q *4,2 / 50 * k. \quad (5.12)$$

Произведя расчет по формуле (5.13) был получен следующий результат:

$$t_{отл} = 1440 *4,2 / 50 *0,8=151 \text{ чел/час.}$$

Необходимое время на подготовку документации $t_{док}$, берется по факту и примерно составляет от 3-х до 5-ти рабочих дней по 8 часов:

$$t_{док} = 32 \text{ чел / час.}$$

Таким образом, по формуле (5.6) трудоемкость разработки программного продукта составляет:

$$t_{тр} = 25 + 36 + 36 + 54 + 151 + 32 = 334 \text{ чел/час или } 42 \text{ чел/дней.}$$

Дневная заработная плата рассчитывается в соответствии с месячным окладом и количеством рабочих дней (в среднем 22 рабочих дня). В таблице 5.5 показаны оклады специалистов. Таким образом, дневная заработная оплата программиста составляет: $З_{дн}$ = 150000/22=6818 тенге.

Таблица 5.5 – Оклады специалистов

Специалист	Количество, человек	Зарплата, тг.
Программист	1	150 000
Итого		150 000

Основная заработная плата по формуле 5.5 составит:

$$З_{осн} = 42*6818 = 286356 \text{ тенге.}$$

Дополнительная заработная плата составляет 10 % от основной и рассчитывается по формуле:

$$З_{доп} = З_{осн} * 0.1. \quad (5.13)$$

И составляет:

$$З_{доп} = 286356 * 0.1 = 28635 \text{ тенге.}$$

Таким образом, фонд оплаты труда составит:

$$З_{фот} = 286356 + 28635 = 314991 \text{ тенге.}$$

Согласно ст. 358 п. 1 НК РК социальный налог равен 11% от дохода работника, и рассчитывается по формуле:

$$О_{сн} = (З_{фот} – З_{по}) * 0.11\%, \quad (5.14)$$

где $З_{по}$ - пенсионные отчисления, которые социальным налогом не облагаются и их доля от фонда оплаты труда составляет 10%:

$$З_{по} = З_{фот} * 0.1. \quad (5.15)$$

Пенсионные отчисления рассчитаны по формуле (*5.16*) равны:

$$З_{по} = 314991 * 0.1\% = 31499 \text{ тенге.}$$

Основываясь на вышеперечисленных расчетах и используя формулу (5.14) социальный налог равен:

$$О_{сн} = (314991 - 31499) * 11\% = 31184 \text{ тенге.}$$

Базируясь информацией, полученной из исходных данных, определяется объем затрат на материалы:

$$М = (З_{осн} * Н_{рмз})/ 100\%, \quad (5.16)$$

где $Н_{рмз}$ – норма расходов материалов от основной заработной платы в среднем составляет от 3 до 5% .

Согласно данной формуле (5.16) затраты на материалы равны:

$$М = (286356* 5\%)/ 100\% =14317 \text{ тенге.}$$

В данном проекте используется среда программирования «Arduino IDE» которая находится в свободном распространении, следовательно расходы на специальные программные средства ($Р_с$) равны 0.

В расходы на амортизацию входят непосредственно отчисления на амортизацию от стоимости техники, применяемой при создании ПП, для этого используется формула

$$Р_{ам} = \frac{С_{перв} * Н_а * N}{100 * 12 * t}, \quad (5.17)$$

где На – норма амортизации, составляющая 25%;

Сперв – первоначальная стоимость техники, тенге;

N – время использования персонального техники, (42 дня); t – количество рабочих дней в месяце, дни.

Согласно формуле (5.17) амортизационные расходы составляют:

$$Р_{ам} = \frac{250000 * 0{,}25 * 42}{100 * 12 * 22} = \frac{2625000}{26400} = 99{,}43.$$

Затраты на электроэнергию вычисляется по формуле [19]:

$$Р_Э = М*k_З*Т*С_{кВт-ч}, \quad (5.18)$$

где М – мощность ЭВМ, КВт;

$k_з$ – коэффициент загрузки (0.8);

$С_{кВт-ч}$ – стоимость 1 кВт·ч электроэнергии, тенге/ кВт·ч;

Т – время работы, час (328 ч.).

Используя формулу (5.18) были определены затраты на электроэнергию и приведены в таблице 5.7:

$$Р_{эл} = 0.12*0.8*328*16.65 = 524 \text{ тенге.}$$

Таблица 5.7 - Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент загрузки	Время работы оборудования для разработки ПП, час	Цена эл/э, тенге/кВт-час;	Сумма, тенге
Ноутбук	0.12	0.8	328	16.65	524
ИТОГО затраты на электроэнергию					524

Накладные расходы, (Рнак) и составляют от 40 до 60% от основной заработной платы :

$$Рнак = Зосн * Нрн/100\%, \quad (5.19)$$

где Рнак – накладные расходы на ПП (тенге);

Нрн – норматив накладных расходов -50%.

Рнак = 286356 * 0.5=143178 тенге.

Таким образом, капитальные вложения на разработку программного продукта составили:

Кпп = 314991 + 31184 + 14317 + 0 + 99,43 +524+143178 = 504293 тенге.

Сводные результаты расчета затрат на разработку ПП представлены в таблице 5.8.

Таблица 5.8 – Сводная таблица расходов на разработку программного продукта

Статьи затрат	Сумма, тенге
Фонд оплаты труда	314991
Социальный налог	31184
Материалы	14317
Амортизация	99,43
Электроэнергия	524
Накладные расходы	143178
Итого:	504293

5.4 Расчёт рыночной цены

Итоговая рыночная стоимость продукта будет рассчитываться по следующей формуле:

$$С_{рын} = К_{рп} + П + НДС, \quad (5.20)$$

где $С_{рын}$- Рыночная стоимость;

$К_{рп}$ – Капиталовложения на реализацию проекта;

П – прибыль;

НДС – налог на добавочную стоимость, равна 12% с 1 января 2018 года.

Рассчитаем капиталовложение по формуле (5.1):

$$К_{рп} = 69400 + 20820 + 504293 = 594513 \text{ тг.}$$

Прибыль рассчитывается как 20% от капиталовложения на реализацию проекта.

В итоге получим:

$С_{рын}$ = 594513 + 594513*0,2 + (594513 + 594513*0,2)*0,12 = 713415 + 85609 = 799024 тг.

5.5 Оценка экономической эффективности

Исходя из того, что данный проект является заменой механической работы сотрудников поиска необходимого товара среди огромного множества ассортимента, а также, следовательно, не будет необходимости в создании дополнительной сопроводительной документации, где будут указаны координаты местоположения товара, будет рассмотрена экономическая эффективность проекта.

Так как автоматизированная система должна быть максимально эффективна по сравнению с предыдущей используемой на складе технологией, она должна иметь преимущества и использовать свои ресурсы на максимум.

Система имеет следующие ресурсы:

- пропускная способность
- вероятность ошибки
- время простоя

В классическом неавтоматизированном складе необходим минимум один работник, выполняющий все виды работ, включающие как организационные, так и физические.

Так как предлагаемая автоматизированная система легко масштабируема, её эффективность будет увеличиваться с увеличением размера складского помещения. Так как при ручном труде пропускная способность уменьшится по причине увеличения времени на нахождение необходимого товара, вероятность ошибки также увеличится, а время простоя у автоматики не увеличится или даже может быть равно нулю.

Теперь рассчитаем ежемесячные расходы компании при неавтоматизированном складе. Расходы показаны в таблице 5.1.

$$Р_{нас} = ЗП + РМ, \qquad (5.21)$$

где $Р_{нас}$ – Расходы неавтоматизированного склада;

ЗП – Заработная плата сотрудника;
РМ – Расходные материалы.

Таблица 5.1 - Ежемесячные расходы при неавтоматизированном складе

Наименование	Расход, тг/мес
Заработная плата сотрудника	80 000
Расходные материалы	5000

В итоге получаем:

$$Р_{на} = 80000 + 5000 = 85\ 000\ тг/мес.$$

Рассчитаем возможные расходы при автоматизированном складе. Расходы будут включать в себя только расходы на амортизацию и электроэнергию.

Теперь рассчитаем ежемесячные расходы компании при автоматизированном складе. Расходы показаны в таблице 5.2.

$$Р_{ас} = Р_{э} + Р_{а}\ , \qquad (5.22)$$

где $Р_{ас}$ – Расходы неавтоматизированного склада;
$Р_{э}$ – Расходы на электроэнергию;
$Р_{а}$ –Расходы на амортизацию.

Расход на электроэнергию рассчитывается по следующей формуле:

$$Р_{э}=М_{п} * С_{к} * Д_{м} * Ч_{д}\ , \qquad (5.23)$$

где $Р_{э}$ – Расходы на электроэнергию, тенге;
$М_{п}$ – Потребляемая мощность, кВт;
$С_{к}$ – Стоимость одного кВт*ч, тенге/кВт*ч.
$Д_{м}$ – Дней в месяце;
$Ч_{д}$ – Часов в день.

В итоге получим:

$$Р_{э} = 0{,}07 * 16.65 * 30 * 12 = 420\ тг/мес.$$

Расход на амортизацию в месяц рассчитывается по следующей формуле:

$$Р_{ам}= \frac{Н_{а}*К_{зо}}{100*12}, \qquad (5.24)$$

где $Н_{а}$ – Норма амортизации(20%);
$К_{зо}$ – капиталовложения на закупку оборудования.

В итоге получим:

$$P_{ам} = \frac{20*69400}{100*12} = \frac{1157тг}{мес}.$$

Таблица 5.2 - Ежемесячные расходы при автоматизированном складе

Наименование	Расход, тг/мес
Расходы на электроэнергию	420
Расходы на амортизацию	1157

В сумме получим общие расходы по формуле (5.22):

$$P_{ас} = 420 + 1157 = 1577тг/мес.$$

Срок окупаемости будет равен сроку, за который ежемесячные расходы неавтоматизированного склада превысят суммарные ежемесячные расходы автоматизированного склада вместе с первоначальными капиталовложениями. В таблице 5.3 показано сравнение ежемесячных расходов. На рисунке 5.1 показан график расходов, а точка пересечения прямых является сроком окупаемости автоматизированного склада.

Таблица 5.3 – Сравнение ежемесячных расходов

Капиталовложения, тг	Виды расходов	Месяцы											
		1	2	3	4	5	6	7	8	9	10	11	12
0	Расходы при неавтоматизированном складе, тг	85000	85000	85000	85000	85000	85000	85000	85000	85000	85000	85000	85000
799024	Расходы при автоматизированном складе, тг	1577	1577	1577	1577	1577	1577	1577	1577	1577	1577	1577	1577
	Суммарный расход при неавтоматизированном складе, тг	85000	170000	255000	340000	425000	510000	595000	680000	765000	850000	935000	1020000
	Суммарный расход при автоматизированном складе, тг	800601	802178	803755	805332	806909	808486	810063	811640	813217	814794	816371	817948

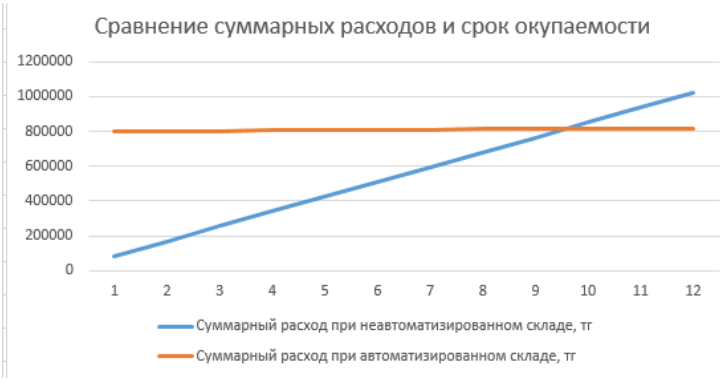


Рисунок 5.1 – Сравнение суммарных расходов и срок окупаемости

Как видно из графика, срок окупаемости составляет 10 месяцев, что подтверждает возможность внедрения данного проекта на рынок автоматизации складов.

Заключение

В данном дипломном проекте был произведён анализ существующих систем автоматизации склада, на основе этого анализа была продумана наиболее подходящая реализация автоматики, продумана электрическая, структурная и функциональная схемы, выбраны наиболее подходящие для этого микроконтроллеры, построена демонстрационная модель, разработано программное обеспечение функционирования автоматизированной системы, разработан графический интерфейс, позволяющий управлять автоматикой на любой платформе (операционной системе), рассмотрены вопросы безопасности жизнедеятельности при монтаже системы, также произведены экономические расчеты для обоснования создания подобной системы, что позволит внедрить её на рынок автоматизации.

В итоге, поставленное задание было выполнено в полном объёме, инженерные расчёты способствовали точному созданию прототипа системы, что даёт возможность для дальнейшего успешного изучения данной темы и возможности реализации с большими инвестициями и соответственно более способному удовлетворить даже самые крупные потребности бизнеса.

Список литературы

1. Статья: «Чего ожидать от автоматизации в будущем» http://www.inboundlogistics.com/cms/article/warehouse-automation-the-next-generation/ (дата обращения 26.01.2018)

2. Статья: «История и тренды автоматизации складов и их технологии» https://www.linkedin.com/pulse/warehousing-history-present-trends-prashant-dedhia (дата обращения 26.01.2018)

3. Статья: «Драматическое изменение индустрии электронной коммерции» http://www.globaltrademag.com/global-logistics/industr-ecommerce-dramatic-changes-warehousing-distribution (дата обращения 26.01.2018)

4. Статья: «Автоматизированные хранилища приближаются» https://www.mmh.com/article/automated_storage_on_the_move (дата обращения 28.01.2018)

5. Статья: «автоматизированное хранилище очень быстро увеличивает ваш рост» https://www.mmh.com/article/automated_storage_scales_with_rapid-_growth (дата обращения 28.01.2018)

6. Статья: «Новые методы хранения и поиска на вашем складе» https://www.mmh.com/article/8_new_ways_to_think_about_puta-way_and_storage_in_your_warehouse (дата обращения 28.01.2018)

7. Статья: «Решение по автоматизации от Kardex remstar» http://www.kardexremstar.com/us/materials-handling-storage-solutions/vertical-lift-systems-en/shuttle-xp-250500.html (дата обращения 28.01.2018)

8. Статья: «Совершенствование сортировочного центра DC» http://www.dmwandh.com/dc-automation-sorting-it-out/ (дата обращения 28.01.2018)

9. Статья : «Описание и технические характеристики системы Perfect Pick» https://www.opex.com/material-handling/perfect-pick-hd (дата обращения 28.01.2018)

10. Статья: «Познакомьтесь с роботом доставщиком Kiva» http://robohub.org/meet-the-drone-that-already-delivers-your-packages-kiva-robot-teardown/ (дата обращения 28.01.2018)

11. Статья: «Что такое автоматизированная система управления складом» https://www.skuvault.com/blog/what-is-a-warehouse-management-system-wms (дата обращения 15.02.2018)

12. Статья: «Сравнение систем WMS и WCS и их роль» http://www.cisco-eagle.com/blog/2014/03/16/wcs-vs-wms-complimentary-roles/ (дата обращения 17.02.2018)

13. Статья: «Какие типы оборудования автоматизированного склада существуют» https://www.quora.com/What-types-of-automated-equipment-does-a-Warehouse-Execution-System-WES-interface (дата обращения 23.02.2018)

14. Статья: «автоматизированная система хранения и поиска ASRS от westfaliausa» https://www.westfaliausa.com/products/automated-storage-retrieval-systems (дата обращения 27.02.2018)

15. Статья: «автоматизированная система хранения и поиска ASRS» https://www.conveyco.com/automated-storage-and-retrieval-types/
(дата обращения 27.02.2018)

16. Документация к микроконтроллеру esp8266 https://www.espressif.com/sites/default/files/documentation/-esp8266-technical_reference_en.pdf (дата обращения 01.03.2018)

17. Шпак Ю.А. Программирование на языке С для AVR и PIC – М. 2011.

18. Олифер В. Компьютерные сети. Принципы, технологии, протоколы. - 5-е изд., -М. 2016

19. З.Д. Еркешева. Методическое указание к выполнению экономической части дипломных работ для студентов специальности 5В071600 – Приборостроение. – Алматы: АУЭС, 2017 – 29с.