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

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки)

Электроника и робототехника
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Система управления для макета технологического производства

Обучающийся Р. С. Щербаков
(Инициалы Фамилия) (личная подпись)

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

Консультант к.п.н., доцент, О.В. Лебединская
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Название бакалаврской работы: «Система управления для макета технологического производства».

Данная работа состоит из пояснительной записки представленной на сорока семи страницах и включающей в себя: восемнадцать рисунков, семь таблиц, список источников из двадцати пяти наименований, в том числе пять источников на иностранном языке и двадцать две страницы приложений, и изображений на шести листах формата А1.

Целью работы является создание системы управления на базе Arduino для макета технического производства на базе Fishertehnik, а также разработка методических рекомендаций к данному проекту.

Задачи стоящие перед этой работой:

— обзор существующих решений,

— разработка электрических схем и проведение необходимых расчётов,

— выбор элементов схем,

— написание программного обеспечения для системы управления,

— сборка готового макета, загрузка в него програмного кода,

— проведение теста системы управления,

— подготовка методических рекомендаций.

Результатом выполнения бакалаврской работы является рабочий стенд на основе Arduino и Fishertehnik, методические рекомендации и практическая реализация складского роботизированного комплекса.

Эта работа представляет интерес для широкого круга читателей. Это будет полезно, как для людей интересующихся робототехникой и электроникой, так и для людей заинтересованных в автоматизации производства. Изготовленный макет будет включён в перечень лабораторных стендов университета, используемых в процессе обучения.

Abstract

The topic of the given bachelor’s work is: «Control system for the prototype of technical production».

The work consists of an explanatory note presented on forty-seven pages and includes: eighteen figures, seven tables, a list of from twenty-five titles references including five foreign sources and twenty-two appendices, and the graphic part on six A1 sheets.

The aim of the work is to create an Arduino-based control system for a prototype of technical production based on Fishertehnik, as well as to develop methodological recommendations for this project.

Tasks facing this work:

— overview of existing solutions,

—development of electrical circuits and carrying out the necessary calculations,

—selection of circuit elements,

—writing software for the control system,

—assembling a ready-made prototype, loading program code into it,

—conducting a control system test,

—preparation of methodological recommendations,

The result of the bachelor's work is a working stand based on Arduino and Fishertehnik, methodological recommendations and practical implementation of the warehouse robotic complex.

The work is of interest for wide circle of readers. It will be useful both for people interested in robotics and electronics, and for people interested in automation of production. The manufactured model is included in the list of laboratory stands of the university used in the learning process.

Содержание

Введение

В современном мире автоматизация производства не просто важна, а является необходимостью для комфортного существования любого крупного предприятия. Без автоматизации невозможно обеспечить высокую эффективность и конкурентоспособность. Автоматизация позволяет значительно ускорить процессы, уменьшить вероятность брака в производстве и снизить затраты на оплату труда, сократить нерациональный расход ресурсов и как закономерный итог, увеличить количество и качество выпускаемой продукции.

Машины способны эффективно обрабатывать огромные объёмы данных и рассчитывать сложнейшие формулы. Очень немногие люди способны на подобное, без использования подручных средств, вроде тех же машин. Необходимо отметить ещё несколько несомненных преимуществ машин перед человеком. Машины не чувствуют усталости и они способны работать в неблагоприятных условиях, которая для человека являются смертельно опасной. Например использование роботов полезно при работе с опасными химическими веществами, или в условиях с низким содержанием кислорода.

Но по настоящему автоматизацию нельзя будет назвать полной, если автоматизированы будут только производственные станки. Необходимо также проводить автоматизацию складов и роботов манипуляторов.

Одним из наиболее важных элементов автоматизации является автоматизированный склад. Он позволяет хранить и перемещать товары с минимальным участием человека. Это снижает вероятность ошибок и повышает скорость обработки заказов. В данной работе мы рассмотрим конкретный пример работы автоматизированного склада с панелью оператора на базе Arduino. Это поможет понять, как работает автоматизированный склад и какие преимущества он может предоставить предприятию.

1 Изучение вопроса

1.1 Актуальность темы

Автоматизированные склады с панелью оператора являются одним из примеров применения технологий автоматизации на производстве. Такие склады позволяют повысить эффективность и конкурентоспособность предприятия за счёт сокращения времени на выполнение операций и уменьшения количества ошибок, что актуально для любого предприятия [13].

Одним из основных преимуществ автоматизированных складов [17] является возможность управления всеми процессами на складе с помощью единой панели оператора.

Система управления, разработанная для макета технического производства, может быть использована в качестве примера того, как сочетание электронного конструктора Arduino и инженерного конструктора Fishertehnik может привести к созданию новых более эффективных решений.

Это объединение является актуальным направлением в области образования и развития технологий за счёт совмещения сильных сторон двух конструкторов: Fishertehnik, отвечающего за механическую часть макета и Arduino, отвечающего за его электронную и программную часть. Эта особенность позволяет освоить работу механики, электроники и программирования разом [1, 4].

Данная работа также может быть использована в качестве примера по совмещению двух различных систем, что в наибольшей степени будет полезно при работе с автоматизированными механическими системами на производстве, а также в робототехнике и для создания устройств управления и многого другого.

Данная работа может заинтересовать специалистов в области электроники, механики и программирования, а также широкую аудиторию, интересующуюся современными технологиями и их применением в жизни.

1.2 Цели и задачи исследования

В цели бакалаврской работы входит следующее:

– разработка системы управления для управления макетом,

– создание удобного и понятного интерфейса для управления макетом,

– проведение тестирования в целях удостоверения работоспособности проекта.

Задачи работы:

– обзор существующих решений,

– разработка электрических схем и проведение необходимых расчётов,

– выбор элементов схем,

– сборка готового макета загрузка в него програмного кода,

– проведение теста системы управления,

– подготовка методических рекомендаций.

В работе будет проведён анализ существующих проектов на совместной базе Arduino и fishertehnik, а также представлен собственный проект, демонстрирующий применение данного сочетания технологий.

В результате работы будет получен полный обзор возможностей совмещения Ардуино и fishertehnik, а также изготовлена система управления для макета технического производства, разработаны собственные проекты на основе полученного макета, демонстрирующие применение данного сочетания технологий.

1.3 Обзор существующих технологий и решений

Мною не было обнаружено существующих решений по совмещению Arduino и заданного по техническому заданию макета вертикального склада Fishertehnik. Однако мною было обнаружено, что большинство проектов по совмещению Arduino и fishertehnik, представлены в иностранных источниках, а в Российском сегменте данный вопрос освещён не так широко. .

Кроме того, в рассмотренных проектах используется контроллер самого Fishertehnik, что не соответствует требованиям нашей работы. Однако, мы смогли получить некоторую информацию из этих проектов, которую возможно использовать при работе с нашим макетом.

Например, мы нашли проект, демонстрирующий совместную работу Fishertehnik и Micro:Bit на примере шлагбаума и роботов. Это позволило нам получить ценную информацию о том, как можно связать два устройства и использовать их вместе для решения конкретной задачи.

В целом можно сказать, что несмотря на малое количество собранной информации по данной задаче, нам всё же удалось найти ценную информацию и применить её для работы над данным проектом макета вертикального склада Fishertehnik.

1.4 Анализ данных по проекту

Для выполнения данной выпускной квалификационной работы мне потребовалось ознакомиться с наличием материально-технической базы кафедры «Промышленная электроника».

В качестве основного объекта управления был выбран макет технического производства на базе Fishertehnik - Вертикальный склад.

А для соответствия техническому заданию, системой управления был выбран микроконтроллер на базе Arduino. Данная аппаратная платформа проста в освоении и имеет относительно низкую стоимость, что позволяет её использовать для создания подобных проектов [2,3,11,12].

Так как Ардуино имеет разные типы плат, то необходимо провести исследование для подбора более подходящей под данное задание платы. В случае отсутствия подходящего контроллера будет стоять задача по его приобретению или использованию других имеющихся подходящих под требуемую задачу средств.

Для программирования контроллера используется специальная

одноимённая программная среда Arduino IDE которая работает на языке Arduino C, схожем с языками типа C/C++ [6,23].

Для создания чертежей используется программа KOMPAS и sPlan.

Для совмещения устройств было решено было решено использовать шлейф в качестве соединения. Это позволит избежать необходимости подключать каждый провод в отдельные разъёмы и оставит макет доступным для использования другими проектами.

1.5 Система управления на базе Arduino

Arduino представляет собой линейку электронных блоков-плат, которые можно подключать к компьютеру по USB, а в качестве периферии любые устройства от светодиодов до механизмов радиуоправляемых моделей и роботов. Программы для Arduino пишутся на простом и интуитивно понятном си-подобном языке Wiring, называемом также упрощённым C++, однако синтаксис, операторы и возможности представлены обычным C++.

Питание платы осуществляется двумя способами: по кабелю USB, либо по специальному разъёму как у ноутбуков. В радиомагазине можно купить такой разъём и присоединить к нему аккумулятор или 9-тивольтовую батарейку типа «Крона». Источники питания можно менять перемычкой на плате.

Необходимо отметить, что Arduino это платформа с открытым исходным кодом, основанная на микроконтроллерах. Это означает, что все ресурсы платы, включая CAD файлы и т.д., находятся в свободном доступе для всех пользователей [19,25].

Arduino состоит из двух частей:

– программная (программная оболочка платы),

– аппаратная (физическая оболочка платы).

Первое время для тестов и до подсчёта количества подключений, а также до получения более подходящего устройства нами использовалась Arduino

UNO. Однако как показало дальнейшее исследование, данный микроконтроллер нам не подходил из-за нехватки цифровых выводов на нём.

Arduino NANO, хоть и имеет некоторые преимущества в размерах и числе аналоговых пинов перед Arduino UNO, но этому микроконтроллеру, как и микроконтроллеру Arduino UNO, не хватает цифровых пинов, для работы с нашим макетом вертикального склада на базе fishertehnik.

Однако размеры платы в нашем случае являются незначительным преимуществом, в то время как обе платы либо равны между собой либо полностью уступают по характеристикам Arduino MEGA. Поэтому впоследствии выбор был сделан в пользу Arduino MEGA. Сравнение микроконтроллеров приведено в таблице 1.

Таблица 1 – Характеристики плат Arduino

Плата ардуино	NANO	UNO	MEGA	DUE
Микроконтроллер	Atmega328	Atmega328	Atmega2560	AT91SAM3X8E
Рабочее напряжения (В)	5	5	5	3,3 В
Цифровые входа/выходы	14	14	54	54
Выходы с ШИМ	6	6	14	12
Аналоговые входа/выходы	8	6	16	12
Максимальный ток с пина ввода/вывода (мА)	40	40	40	800
Flash-память (КБ)	32	32	256	512
ОЗУ (КБ)	2	2	8	96
EEPROM-Память(КБ)	1	1	4	-
Тактовая частота (мГц)	16	16	16	84
Габариты (мм)	18/45	69/53	102/53	102/53
USB-разъём	mini-USB	USB A-B	USB A-B	micro-USB

На рисунке 1 представлена Arduino Uno, которая является стандартной

платой Arduino и наиболее распространённой. Она основана на чипе ATmega328, имеющем на борту 32 КБ флэш-памяти, 2 Кб SRAM и 1 Кбайт EEPROM памяти [22]. На периферии имеет 14 дискретных каналов ввода и вывода и 6 аналоговых каналов, 2 пина с аппаратным прерыванием. Эти характеристики, позволяет данному контроллеру решать большинство базовых задач.

Рисунок 1 – Плата Arduino UNO

Также необходимо отметить, пользу данного микроконтроллера в области обучения и проверок работоспособности, а одно из главных положительных качеств этой платы, это то, что она одна из самых дешёвых и распространённых. Поэтому её легко найти и приобрести. Данная плата хорошо подходит для обучения работе с микроконтроллерами, но на ней вполне возможно собирать и более серьёзные проекты, не требующие большого количества цифровых входов.

Arduino Nano (рис. 2) - это микроконтроллерная платформа, которая представляет собой уменьшенную версию Arduino Uno. Она обладает аналогичным функционалом, но имеет более компактный размер и несколько отличается по внешнему виду. Она была разработана для использования в проектах, где требуется компактность и малый размер платы. Она имеет

габариты 19x43 мм и весит всего 8 грамм. Несмотря на свой небольшой размер, плата обладает почти всеми возможностями, которые есть у Arduino Uno.

Рисунок 2 – Плата Arduino Nano

Arduino Nano оснащена микроконтроллером ATmega328, который имеет тактовую частоту 16 МГц и в зависимости от чипа от 16 до 32 Кбайт памяти Flash, от 1 до 2 Кбайт оперативной памяти и от 512 до 1 Кбайт энергонезависимой памяти EEPROM [7].

Одним из отличий Arduino Nano от Arduino Uno является отсутствие собственного гнезда для внешнего питания. Вместо этого плата использует mini-USB кабель для питания и взаимодействия с компьютером. Кроме того, Arduino Nano имеет штырьковые контакты, что позволяет установить её на макетную плату.

Отличие также заключается в отсутствии собственного гнезда для внешнего питания, использованием чипа FTDI FT232RL для USB-Serial преобразования.

Одним из основных преимуществ Arduino Nano является её компактность. Это позволяет использовать её в проектах, где требуется минимальный размер платы. Кроме того, Arduino Nano является более экономичной альтернативой Arduino Uno, что делает её доступной для широкого круга пользователей.

В целом, Arduino Nano - это отличная платформа для создания

различных проектов, которые требуют компактности и надёжности. Она имеет все нужные функции и возможности, которые есть у Arduino Uno, но при этом занимает меньше места.

На рисунке 3 изображена Arduino MEGA, которая является аналогом Arduino UNO, но на базе более мощного микроконтроллера той же архитектуры. Имеет больше памяти: 256 КБ постоянной и 8 КБ оперативной. Также имеет больше портов: 60 из них 16 аналоговых и 15 с ШИМ. Однако имеет немного более крупные размеры 101×53 мм. Исходя из вышесказанного, можно понять, что данная модель превосходит Arduino UNO по всем параметрам и полезна при разработке крупных проектов.

Рисунок 3 – Плата Arduino MEGA

Arduino DUE представленная на рисунке 4 - это мощная плата, которая предназначена для профессиональных проектов, требующих высокой производительности и большого объема памяти. Она работает на микроконтроллере Cortex-M3, который является одним из самых мощных и энергоэффективных процессоров в своем классе. Этот процессор обеспечивает высокую скорость работы и низкий уровень энергопотребления, что позволяет использовать плату в различных приложениях.

Arduino DUE имеет аналогичный форм-фактор с Arduino Mega, что

делает ее совместимой со всеми модулями и дополнительными устройствами, которые поддерживают Arduino Mega. Она также имеет большое количество пинов ввода-вывода, что делает ее очень гибкой и универсальной для различных задач.

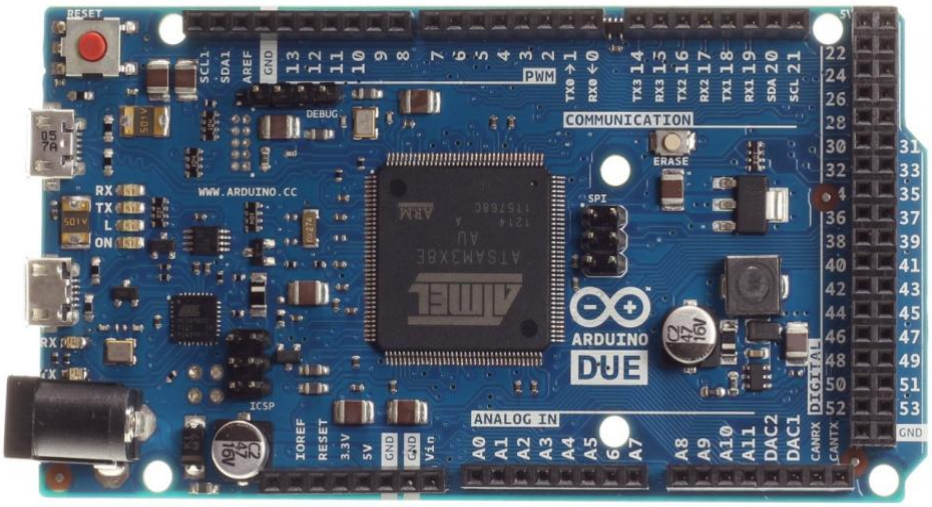

Рисунок 4 – Плата Arduino DUE

Одним из преимуществ Arduino DUE является большой объем памяти - 512 КБ. Это позволяет загружать на плату большие программы и хранить большие объемы данных. Также плата имеет 54 цифровых входа-выхода, из которых 12 могут быть использованы в качестве аналоговых входов, 12 поддерживают ШИМ и все 66 могут быть настроены как аппаратные прерывания. Это позволяет использовать плату для множества различных задач, включая управление моторами, сенсорами, светодиодами и другими устройствами. Родным напряжением для платы является 3.3 В, а не традиционные 5 В.

В целом, можно сказать, что Arduino DUE - это мощная и гибкая плата, которая может быть использована для широкого спектра проектов. Она обеспечивает высокую производительность, большой объем памяти и множество входов-выходов, что делает ее идеальной для профессиональных проектов, но в нашем случае она слишком избыточна, для данного проекта.

1.6 Устройства необходимые для реализации HMI

При выборе интерфейса для управления макетом было решено оставить задел на дальнейшее изменение и улучшение системы управления и программы. Поэтому при выборе устройств управления мы остановились на матричной клавиатуре размером четыре на четыре. Такая клавиатура хоть и немного избыточна, но оставляет открытой возможность дальнейшего изменения программы, сохранив базовые функции.

На данном этапе мы определили назначение следующих кнопок: кнопки от одного до шести отвечают за перемещение робота между ячейками склада. Кнопка ноль возвращает все двигатели в нулевое состояние. В дальнейшем данное решение сохраниться, хоть и серьёзно дополнится новым функционалом кнопок.

Помимо системы ввода данных мы используем LCD дисплей, для графического отображения выполняемой программы. Использование устройства вывода информации позволит нам убедиться, что программа выполняется правильно и сигналы доходят до устройства.

Устройства интерфейса управления мы можем запитать от самой платы, в то время, как для питания двигателей fishertehnik нам потребуется согласовать уровни напряжения.

1.7 Макет производственного участка на базе fishertehnik

Описание производственного участка:

Склад с высокими отсеками - это компактное складское помещение для хранения и выгрузки товаров. Обычно такие склады используются для хранения товаров на паллетных стеллажах. Такая стандартизация обеспечивает высокий уровень автоматизации с использованием системы ERP [10].

Склады с высокими отсеками характеризуются оптимальным

использованием площадей, но взамен на их проектирование требуются значительные капитальные затраты. Хранение и извлечение товаров осуществляется манипуляторами, которые перемещаются по различными осям между стеллажами.

Используя транспортные системы, товары поступают и передаются на стеллажные устройства подачи. Если стеллажные устройства подачи автоматизированы, вход в рабочую зону запрещён из-за возможной угрозы жизни рабочего. В случае автоматизированного склада с высокими отсеками товары могут подаваться по конвейерной ленте. Сам товар может идентифицироваться по штрих-коду, который распознаётся специальными датчиками.

Существует несколько способов хранения товаров на складе, однако наиболее распространенным является принцип динамического складирования. В отличие от статического, где каждому товару назначается фиксированное место хранения, здесь товары размещаются в любом свободном месте. Таким образом можно сказать, что принцип динамического складирования в сочетании с автоматизированной идентификацией товаров и стандартизацией складских помещений будет незаменимым элементом современной логистики и позволит эффективнее управлять автоматизированным складом.

Стандартизация складских помещений также играет важную роль в эффективной организации складской деятельности. Одинаковые внешние размеры и разрешенный вес единицы позволяют более эффективно использовать доступное пространство и ускорить процесс перемещения товаров.

Для того чтобы система управления складом могла эффективно работать, необходимо сохранять местоположение каждого хранимого товара. Для этого используются различные методы идентификации, в том числе чипы FRID или штрих-коды. Устройством идентификации можно назвать любой прибор способный считать данные со штрих-кода и определить объект, если

таковой присутствует в базе данных.

Одним из наиболее эффективных методов оптимизации маршрутов на складе является стратегия ABC, которая предполагает разделение склада на три зоны с различным расстоянием от зоны хранения и выдачи товаров. Этот подход позволяет разместить необходимые товары в зоне "A", которая расположена непосредственно рядом с зоной хранения и извлечения. В свою очередь, товары, которые редко используются, могут быть размещены в зоне "C", которая находится на значительном расстоянии от зоны хранения и извлечения.

На рисунке 5 представлен макет вертикального склада, в котором для перемещения манипулятора используется червячная передача [16], позволяющая манипулятору перемещаться по прямоугольной системе координат [9,20]. Помимо электромоторов в макете имеются концевые переключатели, позволяющие отметить определённые позиции. В таблице 2 представлены основные электрические компоненты задействованные в данной работе.

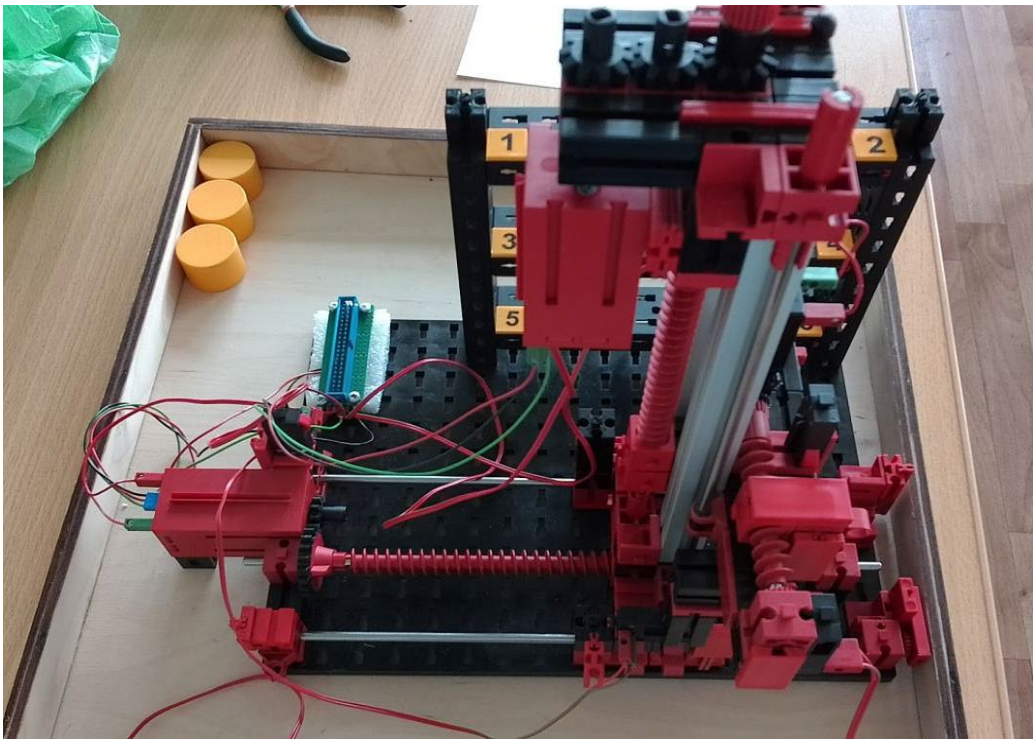

Рисунок 5 - Набор Fischertechnik вертикальный склад

Таблица 2 – Компоненты Fischertechnik

Наименование	Обозначение
Концевой выключатель	I1
Концевой выключатель	I2
Концевой выключатель	I3
Концевой выключатель	I4
Мотор с энкодером	O1
Мотор с энкодером	O2
Двигатель S	I7

Двигатель с энкодером:

Автоматизированный склад с высокими отсеками приводится в действие двумя электродвигателями с энкодером [21], которые представлены на рисунке 6. Это происходит благодаря двигателям постоянного тока с постоянными магнитами, которые позволяют также поэтапно измерять углы с помощью датчиков Холла [14].

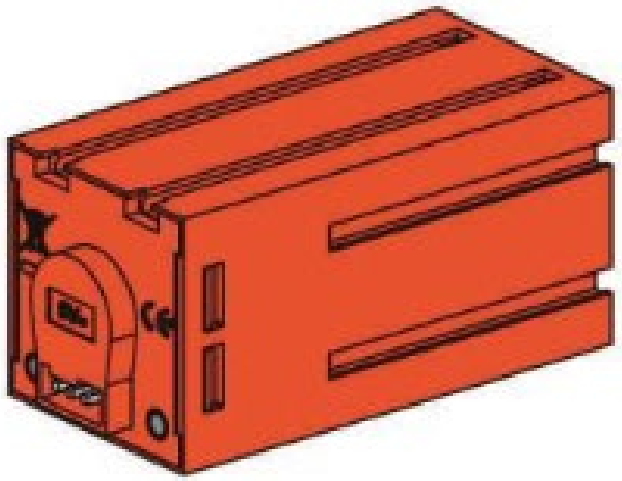

Рисунок 6 – Мотор с энкодером

Двигатели энкодера работают при номинальном напряжении 9 В постоянного тока и имеют максимальную мощность 1,2 Вт при 105 оборотах

в минуту. Потребляемый ток при максимальной мощности составляет 386 мА. Передаточное отношение встроенной коробки передач составляет 21,1:1.

Это означает, что энкодер выдаст три импульса на оборот вала двигателя, или 63,3 импульса на оборот выходного вала коробки передач. Поскольку индексируется только один импульс, используемый энкодер не может определить направление вращения двигателя. Полный состав его характеристик представлен в таблице 3.

Таблица 3 – Характеристики мотора с энкодером

Наименование	Обозначение
Тип электродвигателя:	Коллекторный двигатель постоянного тока
Тип возбуждения	От постоянных магнитов
Напряжение питания	9 В
Скорость холостого хода	326 об/мин
Ток без нагрузки	0,078 А
Максимальная мощность	0,86 Вт
Скорость при максимальной мощности	163 об/мин
Ток при максимальной мощности	0,45 А
Момент при максимальной мощности	5,05 Н*см
Ток максимальный	0,822 А
Момент максимальный	10,11 Н*см
Энкодер	Импульсный - 75 имп/об.
Тип выхода энкодера	Открытый коллектор

Энкодер подключается к контроллеру TXT с помощью трёхжильного кабеля с красным проводом для выхода 9 В и зелёным проводом для подключения к земле. Так как мы намерены использовать ардуино, нам будет необходимо согласовать напряжение.

Чёрный кабель передаёт сигнал (выход с открытым коллектором NPN, макс. 1 кГц) и должен быть подключён к входу быстрого счётчика (C1-C4).

Если контроллер fischertechnik не будет использоваться для считывания сигнала энкодера, требуется использовать подтягивающий резистор (4,7-10 Ком), однако мы намерены считывать сигнал энкодера, поэтому это не является обязательным условием для нас.

Мини-переключатель:

На рисунке 7 изображён мини-переключатель или концевой выключатель [21]. Мини-переключатели используются в качестве опорных переключателей для автоматизированного склада с высокими отсеками. При использовании инкрементных методов измерения для определения абсолютного положения или абсолютного угла используется опорный переключатель.

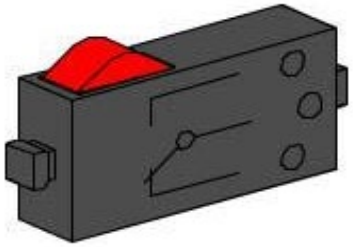

Рисунок 7 – Мини-переключатель

Когда переключатель приводится в действие, происходит эквипотенциальное соединение между контактом 1 и контактом 3, в то время как соединение между контактом 1 и контактом 2 разъединено. Принцип работы данного переключателя мы можем рассмотреть на рисунке 8.

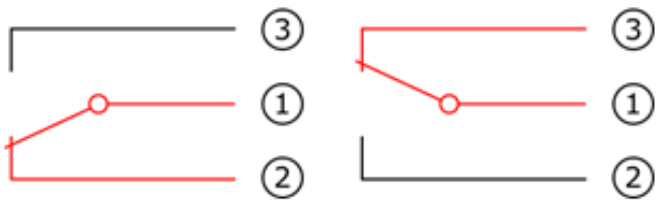

Рисунок 8 – Принцип работы переключателя а - не нажата, б - нажат

Двигатель S:

На рисунке 9 представлен малый двигатель или двигатель S. Этот

компактный двигатель представляет собой двигатель постоянного тока с постоянными магнитами, который может использоваться вместе с присоединяемым редуктором. Характеристики двигателя приведены в таблице 4.

Таблица 4 – Характеристики малого мотора без энкодера

Наименование	Обозначение
Тип электродвигателя:	Коллекторный двигатель постоянного тока
Тип возбуждения	От постоянных магнитов
Напряжение питания	9 В
Скорость холостого хода	9500 об/мин
Ток без нагрузки	0,058 А
Максимальная мощность	1,42 Вт
Скорость при максимальной мощности	4920 об/мин
Ток при максимальной мощности	0,4 А
Момент при максимальной мощности	0,27 Н*см
Ток максимальный	0,741 А
Момент максимальный	0,55 Н*см

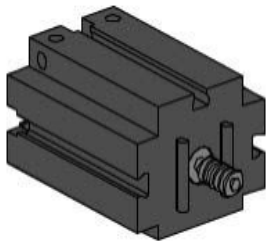

Рисунок 9 – Двигатель S

Двигатель работает при номинальном напряжении 9 В постоянного тока, а максимальный потребляемый ток составляет 650 мА.

Выводы по разделу

В этом разделе мы рассмотрели вопрос актуальности данной работы, определили её цель и доступные ресурсы необходимые для её выполнения.

Мы подробно изучили макет вертикального склада fishertehnik с которым нам предстоит работать, ознакомились с макетом который нам необходимо объединить с программируемым микроконтроллером Arduino. Устройства данного макета будут отвечать за перемещение нашего манипулятора по осям.

Помимо этого нами было рассмотрено несколько вариантов микроконтроллеров и выбран подходящий для реализации системы управления для макета технологического производства. Этим микроконтроллером стала Arduino MEGA 2560, за счёт большого количества доступных для работы портов и доступности.

2 Разработка электронных схем и подбор элементов

2.1 Разработка структурной схемы

Структурная схема представленная на рисунке 10 и в приложении А это графическое представление работы системы и взаимосвязи между различными элементами устройства.

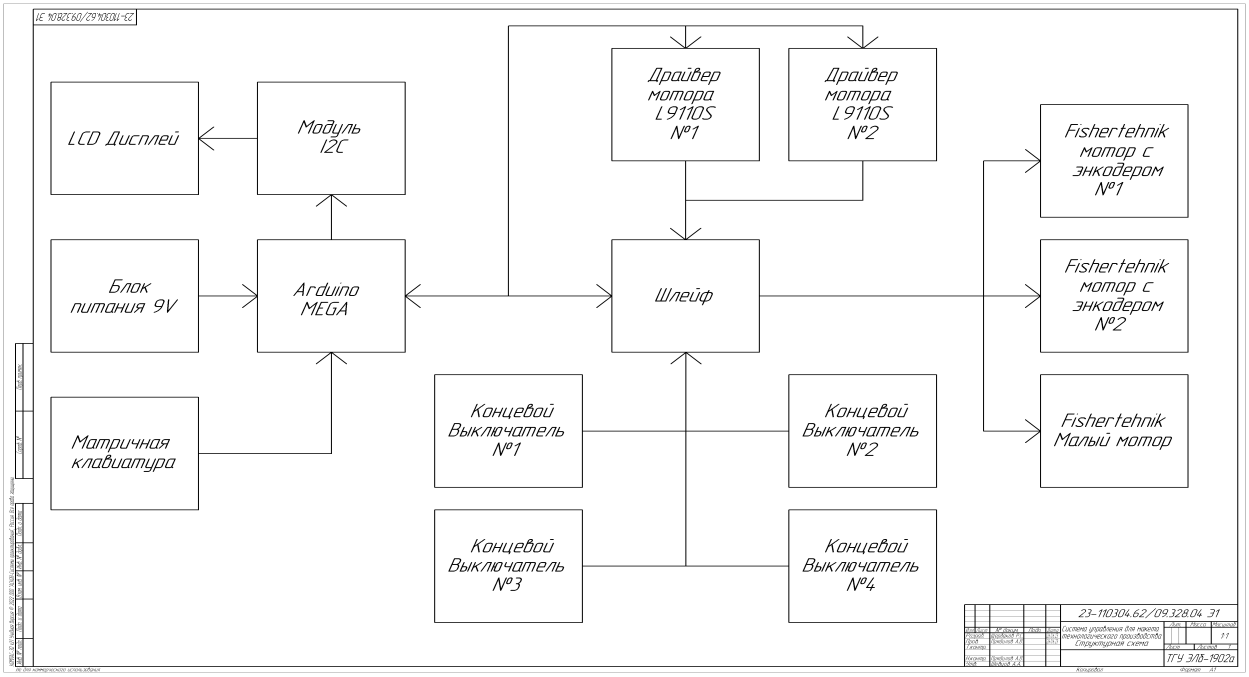

Рисунок 10 – Структурная схема

Как видно из рисунка, питание для Arduino подаётся благодаря блоку питания. Так как для устройств fishertehnik нам необходимо использовать 9 вольт мы будем использовать данное напряжение на блоке питания. Следующим устройством идёт микроконтроллер Arduino MEGA. Он напрямую связан с: модулем I2C, двумя драйверами L9110S и матричной клавиатурой откуда будут поступать команды оператора. Модуль I2C связывает микроконтроллер и LCD дисплей, который отображает указанную в программе информацию для оператора. Драйвера моторов необходимы для связи с устройствами fishertehnik требующими девятивольтового напряжения. Шлейф соединяет макет устройства и систему управления.

2.2 Разработка принципиальной схемы

Согласно ГОСТ 2.701-2008 [8] принципиальная схема определяется как «схема, определяющая полный состав элементов и связей между ними и, как правило, дающая детальное представление о принципах работы изделия». Принципиальная схема представлена на рисунке 11 и в приложении Б.

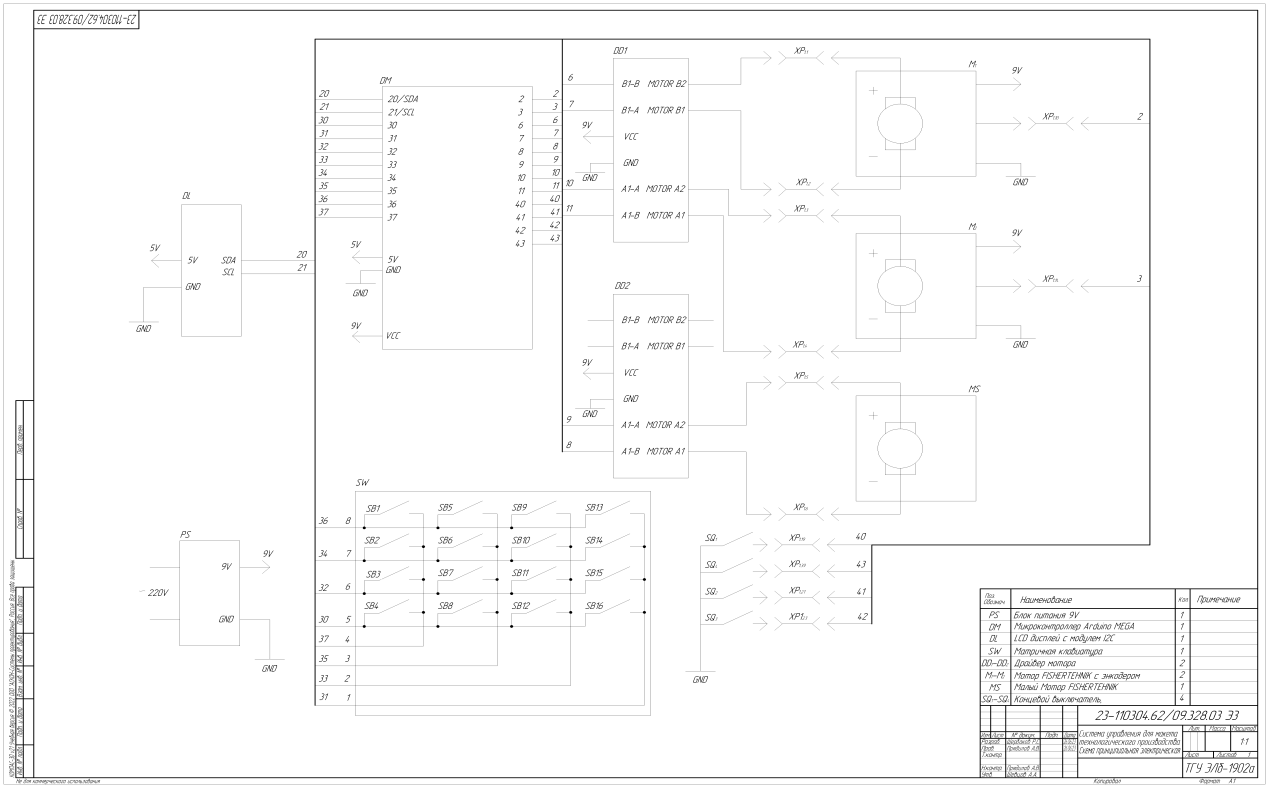

Рисунок 11 – Схема принципиальная электрическая

В данной схеме мы можем увидеть к каким выводам микроконтроллера будут подключены наши элементы и их количество. Таким образом, мы можем заметить, что матричная клавиатура в данной работе состоит из шестнадцати ключей. Блок питания с напряжением девять вольт подключается к самой плате, к которой припаивается вывод позволяющий использовать напряжение в 9 вольт.

2.3 Согласование уровня напряжения логики

Согласование уровня напряжения мы реализуем благодаря блоку питания и драйверам мотора L9110S. Блок питания, который представлен на рисунке 12 имеет максимальное напряжение в 12 вольт, что означает, что даже в случае каких-либо проблем с этим блоком, наша плата не будет повреждена. Однако двигатели fishertehnik питаются от 9 вольт, поэтому сразу выставляем необходимое значение на блоке и подключаем к нашему устройству.

Рисунок 12 – Блок питания

Однако мы не сможем запитать двигатели если просто подключим 9 вольт к ардуино, помимо этого нам необходимо использовать драйвера моторов L9110S, которые позволят нам управлять моторами fishertehnik. Данные драйвера отлично подходят в нашей работе, они совместимы с arduino и fishertehnik, они могут работать с заданным напряжением, они небольшие, что хорошо скажется на компактности системы управления.

Чтобы питание поступало на драйвера нам необходимо припаять к

Arduino дополнительный провод ведущий к необходимым 9 вольтам, а после совместить его с выводами питания с самого драйвера (рисунок 13).

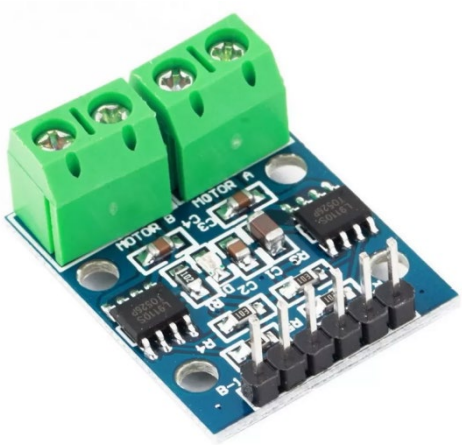

Рисунок 13 – Двухканальный H-bridge драйвер мотора L9110S

Соединение нашего микроконтроллера Arduino и fishertehnik будет реализовано через шлейф, который представлен на рисунке 14. Устройства fishertehnik: моторы и переключатели мы подключим к разъёму шлейфа и протянем шлейф от этого разъёма ко второму разъёму со стороны микроконтроллера.

Это позволит нам достаточно легко отключать нашу систему управления и заменять её на другую, избавив нас от необходимости лезть внутрь корпуса. В данной работе нам потребовалось выпаять разъём под шлейф, а также обеспечить соединение выводов с Arduino и fishertehnik с разъёмами шлейфа [14,18].

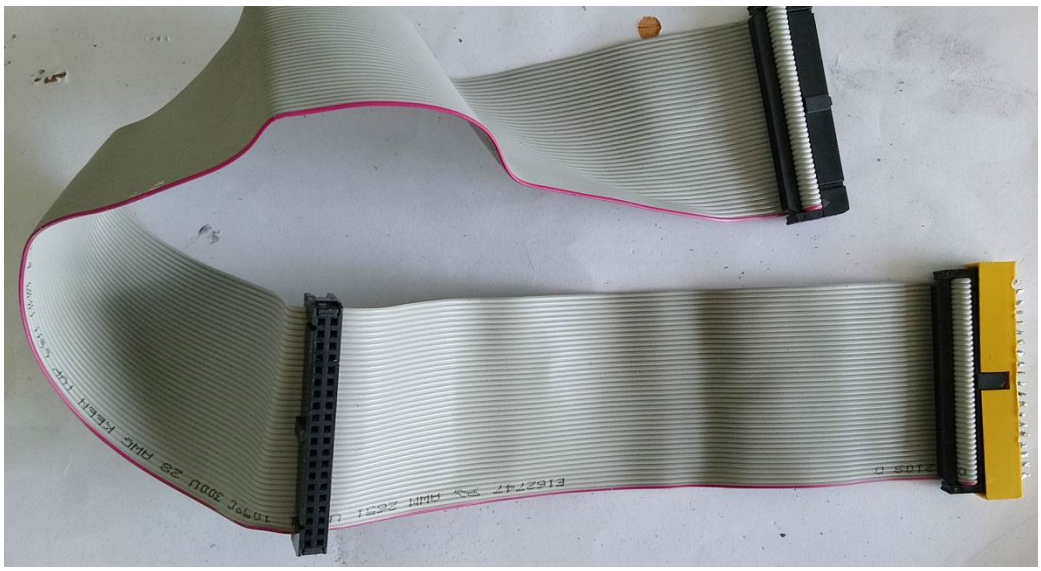

Рисунок 14 – Шлейф

На рисунке 15 представлено в каком порядке идут контакты разъёма. В данном случае наличие 40 пинов немного избыточно, но в случае изменения конструкции текущего макета, в сторону увеличения количества устройств, это позволяет оставить запасные контакты для подключения, что позволит использовать данную разработку в дальнейшем. Разъём под шлейф было решено разместить в корпусе ключом вниз, чтобы при подключении избежать сворачивания шлейфа. В таблице 5 представлены используемые контакты и куда они подключаются

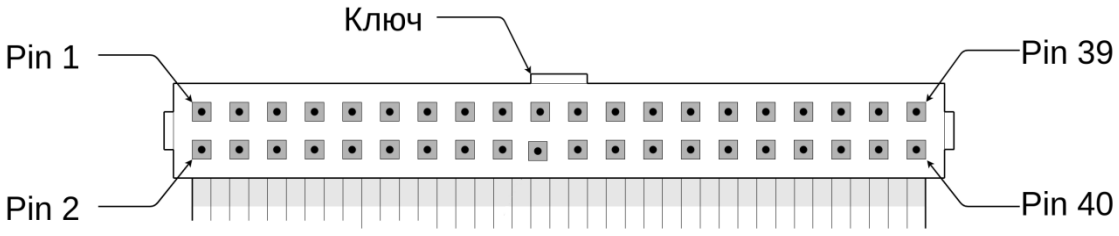

Рисунок 15 – Расположение пинов.

Таблица 5 – Функциональное назначение выводов коннектора 40 пин типа «папа»

Пин	Название в шлейфе	Подключение к Arduino	Пин	Название в шлейфе	Подключение к Arduino	
2	-M1 (out2)	6	1	+M1 (out1)	7	
4	-M2 (out4)	8	3	+M2 (out3)	9	
6	-M3 (out6)	10	5	+M3 (out5)	11	
8	GND9 ENC.M1	-	7	+9V ENC.M1	-	
10	OUT ENC.M1	2	9	OUT ENC.M1	2	
12	GND9 ENC.M2	-	11	+9V ENC.M2	-	
14	OUT ENC.M2	3	13	OUT ENC.M2	3	
16	-	-	15	-	-	
18	-	-	17	-	-	
20	GND	-	19	S9 (M1_LIMIT)	40	Ключ
22	GND	-	21	S10 (M2_LIMIT)	41	
24	GND	-	23	S11 (M3_LIMIT)	42	
26		-	25	-	-	
28	-	-	27	-	-	
30	-	-	29	-	-	
32	-	-	31	-	-	
34	-	-	33	-	-	
36	-	-	35	-	-	
38	-	-	37	-	-	
40	GND	-	39	S12 (M3_LIMIT)	43	

Расшифровка выводов шлейфа.

+ M1 / - M1 – Выводы для первого горизонтального привода.

+ M2 / - M2 – Выводы для вертикального привода.

+ M3 / - M3 – Выводы для второго горизонтального привода.

+9V ENC.M1 / +9V ENC.M2 – Питание +9В для энкодеров

GND9 – Заземление 9В

OUT ENC.M1 I0.0 – Информационный вывод привода энкодера M1.

OUT ENC.M2 I0.0 – Информационный вывод привода энкодера M2.

GND – Заземление

S9 / S10 / S11 / S12 – Информационный вывод концевых переключателей.

Данные выводы необходимы для дальнейшего подключения к контроллеру. Использование данной расшифровки и таблицы позволяет избежать ненужных ошибок при подключении и даёт возможность более точно соединить fishertehnik с проводами микроконтроллера.

2.4 Интерфейс управления

Для реализации интерфейса управления через который пользователь будет управлять макетом технического производства мы будем использовать Arduino.

В качестве устройства вывода информации был выбран жидкокристаллический дисплей LCD 1602 с синей подсветкой, который представлен на рисунке 16. В данном дисплее имеется встроенный модуль I2C, он изображён на рисунке 17 . Основное преимущество данного модуля это сокращение количества задействованных пинов до двух, что было важно на начальном этапе данной работы, во время сборки и тестирования системы управления на Arduino Uno.

Впоследствии было решено оставить задействованный вариант, так как помимо сокращения задействованных пинов и используемых проводов

использование модуля I2C даёт дополнительные преимущества: высокую степень сохранности данных из-за специального фильтра подавляющего всплески, простую процедуру диагностики сбоев, а также упрощённую работу с кодом за счёт подключения библиотек [14].

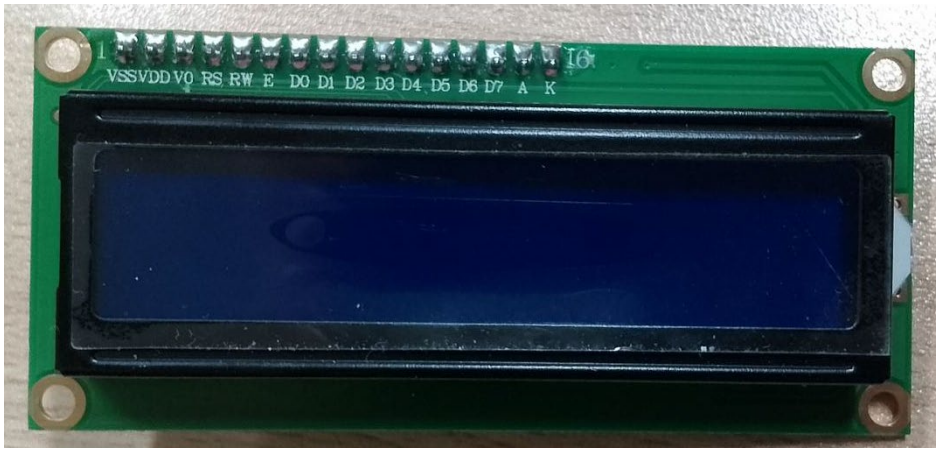

Рисунок 16 – LCD дисплей

Стоит отметить важный момент. Для работы данного модуля требуется загрузить и подключить дополнительные библиотеки, такие как Wire.h, LiquidCrystal_I2C.h или иные имеющие связь с данным модулем.

Рисунок 17 – Модуль I2C

В работе задействован LCD дисплей со встроенным модулем, но возможно приобрести LCD дисплей и модуль отдельно, после чего совместить их, но это может негативно сказать на работе кода в случае каких либо несоответствий выводов при подключений.

Матричная клавиатура. Существует много видов клавиатур, однако рассмотрев будущий функционал нашего макета, было решено отказаться от больших клавиатур и остановить свой выбор на клавиатурах из набора Arduino размером 3 на 3 и 4 на 4. При дальнейшем рассмотрении от клавиатуры 3 на 3 было решено отказаться, из-за нехватки места под задуманное количество функций. Выбранная клавиатура представлена на рисунке 18.

Далее нам требовалось выбрать между мягкой и твёрдой клавиатурой. Функционально они не обладают существенными различиями. Однако мягкая матричная клавиатура выигрывает по массе, эластичности и устойчивости к определённым воздействиям негативных факторов а также удобству расположения и эстетическому виду.

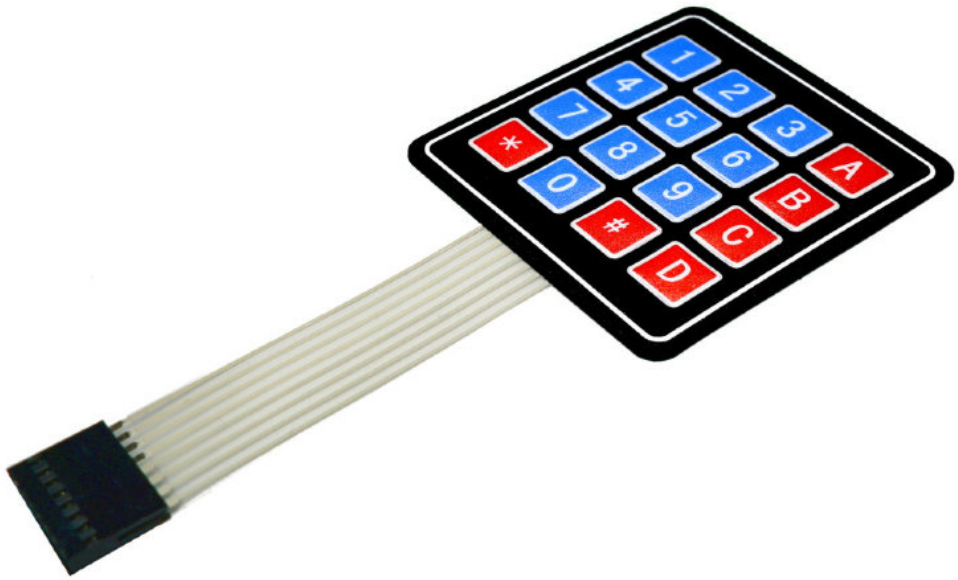

Рисунок 18 – Мягкая матричная клавиатура

Ещё одним преимуществом данной клавиатуры в конкретной работе можно отметить, что она будет также закрывать недостатки корпуса, что было бы невозможно сделать при использовании твёрдой клавиатуры из-за небольших размеров данного устройства.

Выводы по разделу

В этом разделе мы построили схемы на основании которых мы осуществим подключение всех имеющихся необходимых компонентов к плате.

Помимо этого мы определили устройство которое будет подавать питание на arduino данным устройством в данной работе служит блок питания, напряжение в котором мы установили номиналом в 9 Вольт.

При выборе типа подключения земли и питания, было решено спаять их вместе, для экономии места, так как небольшое внутреннее строение корпуса усложнит использование дополнительное платы для подключения земли и питания.

Также мы рассмотрели устройства которые будут использоваться в данной работе для реализации HMI. В качестве устройства ввода мы выбрали матричную клавиатуру размером четыре на четыре, что позволит нам реализовать больше вариантов программ. Для отображения данных мы выбрали LCD дисплей 1620 с модулем I2C.

Построив схемы и определившись с используемыми устройствами мы приступили к проектированию и конструированию корпуса устройства и размещению всех наших устройств.

3 Разработка программы

3.1 Загрузка библиотек и объявление переменных и постоянных

Так как работа кнопок и моторов связаны, то для облегчения понимания программы их надо рассматривать как одно целое. Как видно из кода, нажимая на кнопку мы активируем специальный флаг, который в свою очередь приводит в действие моторы [15].

Библиотеки представленные ниже необходимы для использования с LCD дисплеем с модулем I2C [24]. Данные библиотеки также упростят нам дальнейшую работу с дисплеем.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Библиотека Keypad.h необходима для работы с матричной клавиатурой четыре на четыре.

```
#include <Keypad.h>
```

Благодаря этой клавиатуре мы облегчаем свою работу с клавиатурой. Данная часть кода устанавливает размеры нашей клавиатуры, устанавливает пины к которым подключается клавиатура и в дальнейшем позволит нам обращаться к кнопкам через обозначения в одинарных кавычках.

```
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
```

```
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {36, 34, 32, 30};
byte colPins[COLS] = {37, 35, 33, 31};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
```

Эта команда объявляет наш дисплей и даёт нам возможность использовать его в своём коде.

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

В void setup(), мы объявляем об инициализации дисплея и использовании начальных параметров. В данном случае на экран выводиться текст для оператора "Choose Programs". Данный текст означает, что устройство ожидает выбора оператором рабочей программы.

```
void setup() {
 lcd.init();                  // Инициализация дисплея
  lcd.backlight();              // Подключение подсветки
  lcd.setCursor(0,0);            // Установка курсора в начало первой строки
  lcd.print("Choose");       // Набор текста на первой строке
  lcd.setCursor(0,1);            // Установка курсора в начало второй строки
  lcd.print("Programs");       // Набор текста на второй строке
}
```

3.2 Программирование кнопок и моторов

Так как работа кнопок и моторов, то облегчения понимания их надо рассматривать как одно целое. Как видно из кода, каждая кнопка обладает своей функцией. Краткое описание, обозначающее функции каждой кнопки можно изучить в таблице 6, а также в приложении В.

Таблица 6 – Команды матричной клавиатуры управления устройствами

Наименование кнопки	Назначение
1	Переместить манипулятор к зоне приёма.
2	Переместить манипулятор к второй позиции на складе.
3	Переместить манипулятор к третьей позиции на складе.
4	Переместить манипулятор к четвёртой позиции на складе.
5	Переместить манипулятор к пятой позиции на складе.
6	Переместить манипулятор к шестой позиции на складе.
7	Выбрать первый мотор
8	Выбрать второй мотор
9	Выбрать третий мотор
0	Вернуть устройство в состояние “По умолчанию”.
*	Блокировка управления, повторное нажатие снимает блок.
#	Переключатель ручного управления, блокировка автомата.
A	Положительная регулировка дальности перемещения мотора
B	Отрицательная регулировка дальности перемещения мотора
C	Переместить положение в положительной степени
D	Переместить положение в отрицательной степени

Числовые кнопки. В этот набор входят кнопки: 1, 2, 3, 4, 5, 6. Их задача управление манипулятором макета. Для каждой кнопки построена временная диаграмма представленная в приложении Г. Каждая кнопка отвечает за своё положение на складе, в соответствии со своим номером. Нумерация склада идёт согласно обозначению на макете. Отдельно идёт кнопка 1. Данная кнопка отвечает за приём объекта с отсека приёма.

```
if (customKey == '1' && block == false && hand == false) {
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, HIGH);
  delay(4000);
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, LOW);
   digitalWrite(MC1, LOW);  // вниз HIGH
  digitalWrite(MC2, HIGH); // Вверх HIGH
  delay(6200);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);
  digitalWrite(MC3, LOW);  // вниз HIGH
  digitalWrite(MC4, HIGH); // Вверх HIGH
  delay(2600);
  digitalWrite(MC3, LOW);
  digitalWrite(MC4, LOW);
  digitalWrite(MC1, LOW);  // вниз HIGH
  digitalWrite(MC2, HIGH); // Вверх HIGH
  delay(500);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);
  }
```

Каждое нажатие кнопки соответственно переключает выбранную программу на представленную в списке. Менять программу до завершения текущего действия не желательно, так как это может привести к ошибкам работы макета и повреждению оборудования. Данную проблему возможно решить как минимум с помощью трёх действий, предупредив оператора, о нежелательности совершения подобного действия, написать другой код или решить проблему при помощи дополнительной функции, как это сделано в данной работе. Остановить макет, в случае ошибки оператора можно с помощью активируемой кнопки - “0”. Данная кнопка активирует подпрограмму манипулятор на исходную позицию по всем осям. На каждой оси имеется свой концевой выключатель, который активирует работу следующего мотора и останавливает текущий.

```
if (customKey == '0' && block == false) {
}
```

Вторая кнопка, которая может пригодиться для решения проблем это кнопка - “*”, нажатие этой кнопки позволяет остановить все выполняемые действия и блокировать введение новых команд. Данная команда может быть полезна для проверки оборудования.

```
if (customKey == '*') {
   if (!block) {
     block = true;
     digitalWrite(MC1, LOW);
     digitalWrite(MC2, LOW);
     digitalWrite(MC3, LOW);
     digitalWrite(MC4, LOW);
     digitalWrite(MC5, LOW);
     digitalWrite(MC6, LOW);
```

```
  }
  else if (block) {
    block = false; // Разблокировано
  }
 }
```

Если есть необходимость отключить режим автоматического управления макетом, мы можем активировать режим ручного управления. Это делается через активацию кнопки - “#”, после активации этой кнопки, числовые кнопки будут заблокированы, но будут доступны для работы кнопки: A, B, C и D, а также 7, 8 и 9.

```
 if (customKey == '#' && block == false) {
   if (!hand) {
     hand = true;
     digitalWrite(MC1, LOW);
     digitalWrite(MC2, LOW);
     digitalWrite(MC3, LOW);
     digitalWrite(MC4, LOW);
     digitalWrite(MC5, LOW);
     digitalWrite(MC6, LOW);
   }
   else if (hand) {
     hand = false; // Разблокировано    }  }
```

Кнопки A и B, позволяют регулировать дальность перемещения манипулятора при ручном управлении за счёт изменения переменной timedelay. К примеру в случае использования кнопки A мы увеличиваем значение переменной на плюс 50.

```
if (customKey == 'A' && block == false && hand == true) {
    timedelay+=50;
  }
```

Кнопки 7, 8 и 9 позволяют выбирать управляемый мотор, они работают за счёт объявления переменных, как true и false. При выборе одного мотора мы объявляем переменную выбранного мотора, как истинную, а переменные других моторов, как ложные и обеспечиваем себе защиту от ошибок.

```
  if (customKey == '7' && block == false && hand == true) {
  motor1 = true;
  motor2 = false;
  motor3 = false;
 }
```

Кнопки “C” и “D”, в свою очередь, позволяют управлять моторами, а следовательно и самим манипулятором. Кнопка C соответственно удаляет манипулятор от стартовой позиции, а кнопка D наоборот приближает манипулятор к началу.

Таким образом панель оператора позволяет осуществлять полный контроль над действиями макета. Возможность активации ручного управления, позволяет оператору продолжить работу в случае, если автоматический режим, по какой-то причине не нужен, или не работает.

Полный код всей программы приведён в приложении Д, блок схема этой же программы приведена в приложении Е.

3.3 Программирование дисплея

Дисплей весьма важная часть проекта так как позволяет получать информацию о состоянии программы, поэтому необходимо сделать так, чтобы создать удобный и эффективный интерфейс для операторов, что позволит им оперативно выполнять более сложное регулирование. Такое управление называется эргатическим [13].

В данной части мы работаем с дисплеем, с целью визуализации команд и текущего состояния склада. Дисплей является важным элементом управления и отображения информации, поэтому были разработаны специальные команды, которые будут подтверждать приём сигнала с клавиатуры на плату.

При нажатии любой кнопки управления мы активируем команду, которая отображает на дисплее заданную команду и визуально подтверждает, что сигнал с кнопки поступил на плату.

Для того чтобы упростить понимание ответа дисплея и убедиться в его корректной работе, мы оформили таблицу 7, которая поможет пользователям лучше понять, как работает дисплей. Как известно, LCD дисплей имеет ограниченный набор символов, который зашит в ПЗУ дисплея, поэтому было принято решение дать пояснение программы на русском языке.

Таблица 7 – Ответная реакция дисплея на нажатие кнопки

Наименование кнопки	Выводимая на LCD дисплей текстовая информация	Пояснение выводимой на LCD дисплей текстовой информации
1	Program 1	Выполняется программа 1.
2	Program 2	Выполняется программа 2.
3	Program 3	Выполняется программа 3.
4	Program 4	Выполняется программа 4.
5	Program 5	Выполняется программа 5.
6	Program 6	Выполняется программа 6.
7	Choose M1	Мотор 1
8	Choose M2	Мотор 2
9	Choose M3	Мотор 3
0	By default	Сброс в параметры по умолчанию
*	Block control	Блокировка управления.
#	Hand control	Ручное управление.
A	timedelay+50 и текущее значение timedelay	Прибавить к переменной timedelay 50.
B	timedelay-50 и текущее значение timedelay	Вычесть из переменной timedelay 50.
C	Up value	Изменить положение путём повышения определённого значения
D	Dawn value	Изменить положение путём понижения определённого значения

Команды программы переноса. В этот набор входят кнопки: 1, 2, 3, 4, 5, 6. Их задача передать, что команда с кнопки достигла микроконтроллера и дать понять пользователю, какая команда сейчас выполняется. Соответственно надпись Program и число, которые выводится на LCD дают понять, что сейчас работает программа приводящая в действие моторы перемещающие манипулятор на указанную позицию склада.

```
lcd.clear();
  lcd.setCursor(0,0);            // Установка курсора в начало первой строки
  lcd.print("Program 1");        // Набор текста на первой строке
```

Вторая часть кода первой программы выполняется при завершении действий и сообщает о том, что объект захвачен и необходимо нажать 0 для возвращения в исходную позицию.

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Object captured");
lcd.setCursor(0, 1);
lcd.print("Press 0");
```

Надпись “By default” выводиться при нажатии кнопки “0”. Это означает, программа в данный момент возвращает манипулятор в начальное состояние. То есть все моторы будут двигаться до тех пор, пока не достигнут концевика, который остановит движение моторов.

```
lcd.clear();
lcd.setCursor(0,0);             // Установка курсора в начало первой строки
lcd.print("By default");        // Набор текста на первой строке
```

Надпись “Block control” обозначает, что выполнение программы остановлено до повторного нажатия на эту же клавишу. Поэтому если на дисплее отображается данная надпись, можно понять, почему не работают другие кнопки - они заблокированы.

```
lcd.clear();
lcd.setCursor(0,1);             // Установка курсора в начало первой строки
lcd.print("Block control");       // Набор текста на второй строке
```

Когда на дисплее выводиться надпись - “Hand control”, это означает, что активирован режим ручного управления мы можем активировать режим

ручного управления. Если эта надпись отсутствует, можно понять, что комплекс продолжает действовать в автоматическом режиме и нам недоступны кнопки отвечающие за ручное управление.

```
lcd.clear();
lcd.setCursor(0,0);             // Установка курсора в начало первой строки
lcd.print("Hand control");      // Набор текста на первой строке
```

Надпись “Up value” работает только в ручном режиме и позволяет перемещать манипулятор, увеличивая значение на моторе.

```
lcd.clear();
lcd.setCursor(0,0);             // Установка курсора в начало первой строки
lcd.print("Up value");          // Набор текста на первой строке
```

Надпись “Dawn value” работает только в ручном режиме и позволяет перемещать манипулятор, уменьшая значение на моторе.

```
lcd.clear();
lcd.setCursor(0,0);             // Установка курсора в начало первой строки
lcd.print("Dawn value");        // Набор текста на первой строке
```

Таким образом дисплей позволяет отслеживать работу программы. В случае какой-либо ошибки, это поможет определить причину проблемы или исключить неподходящие варианты. К примеру при нерабочей клавиатуре, мы будем видеть, что экран отображает стартовую надпись, что даст нам понять, что команды не доходят до микроконтроллера, а при проблеме с макетом, мы при нажатии увидим изменение на дисплее, что будет обозначать, что связь между дисплеем, микроконтроллером и клавиатурой рабочая.

Выводы по разделу

В этом разделе мы запрограммировали контроллер на выполнение необходимых действий посредством системы управления. Человеко-машинный интерфейс (HMI) позволяющий нам управлять макетом реализован через устройство ввода - матричную клавиатуру и устройство вывода LCD дисплей, которые мы подобрали ранее.

В разделе мы дали пояснение основным функциям нашей программы и описали, как они должны действовать, что позволит в случае ошибок, быстро определить источник проблемы и ликвидировать его.

Данный код можно использовать как базис, который можно улучшить или поменять функционал. Положительным моментом здесь выступает наличие клавиатуры большого размера, что позволяет добавить больше функций.

После написания кода мы загрузили его в микроконтроллер и провели тестирование и отладку работы макета. Результат работы был удовлетворительным и успешным.

Заключение

В ходе выполнения над выпускной квалификационной работой были выполнены все поставленные задачи. В рамках проекта был разработан учебный лабораторный стенд, реализованный в форме роботизированного комплекса на базе fishertehnik с системой управления на микроконтроллере Arduino с панелью оператора реализованной через устройство ввода матричную клавиатуру и устройство вывода LCD дисплей.

В качестве микроконтроллера который принимает и обрабатывает команды оператора был задействован микроконтроллер Arduino MEGA базирующийся на ATmega2560. Перед его выбором мы изучили доступные микроконтроллеры и остановились на подходящем для данной работы.

Для работы с моторами fishertehnik нами был использован блок питания на 9 вольт, а также подключены драйвера мотора с помощью которых мы управляем движением манипулятора.

Были составлены принципиальная электрическая схема и структурная схема для складского роботизированного комплекса с системой управления, а также временная диаграмма. Все схемы представлены в приложениях.

Также было написано необходимое программное обеспечение и составлено руководство по эксплуатации при работе со стендом. Собранное устройство оставляет возможности для дальнейшего использования данной работы студентами в качестве учебного стенда или для иных целей. Полный код программы, а также блок схемы алгоритма работы приведены в приложениях.

В результате была получена работающая система управления совмещённая с макетом технологического производства, которую возможно использовать в образовательных целях. Для системы управления с целью обеспечения её защиты от воздействия внешней среды был изготовлен корпус из пластикового короба, а также расписаны рекомендации по эксплуатации данного макета.

Список используемой литературы

1. Бегишев И.Р., Хисамова З.И. «Искусственный интеллект и робототехника. Глоссарий понятий»: - Проспект, 2021. - 49 с.
2. Белов, А.В. «Микроконтроллеры AVR: от азов программирования до создания практических устройств» / А.В. Белов. - СПб.: Наука и техника, 2016. - 544 с.
3. Белов А.В. «Разработка устройств на микроконтроллерах AVR: шагаем от "чайника" до профи»: Книга / А.В. Белов. - СПб.: Наука и техника, 2013. - 528 с.
4. Бизнес-журнал. Объединенная межрегиональная редакция Сбор всех частей // Бизнес-журнал: Пензенская область. - 2016. - №4. - С. 96.
5. Булгаков А.Г. «Промышленные роботы: кинематика, динамика, контроль и управление»: - М.: Солон-Пресс, 2017. - 488 с.
6. Бьерн Страуструп. «Язык программирования C++»: - 369 с.
7. Водовозов А.М. «Микроконтроллеры для систем автоматики»: Учебное пособие / А.М. Водовозов. - Вологда: ВоГТУ, 2002. - 123 с.
8. ГОСТ 2.701—2008 «Единая система конструкторской документации. Схемы. Виды и типы. Общие требования к выполнению»: – Введён 01.07.2009 г.
9. ГОСТ 30097-93 «Роботы промышленные. Системы координат и направления движений»: – Введён 01.01.1996 г.
10. ГОСТ Р 59282-2020 «Системы управления складом»: – Введён 01.04.2021 г.
11. Евстифеев А.В. «Микроконтроллеры AVR семейства Tiny фирмы ATMEL»: Руководство пользователя / А.В. Евстифеев. - М.: ДМК, 2015. - 426 с.
12. Евстифеев А.В. «Микроконтроллеры AVR семейств Mega»: Руководство пользователя / А.В. Евстифеев. - М.: ДМК, 2015. - 588 с.
13. Козырев Ю.Г. «Применение промышленных роботов».

Учебное пособие. - М.: "Проспект", 2013. - 358 с.

14. Момот М. В. «Мобильные роботы на базе Arduino»: - 2-е изд. - СПб.: БХВ-Петербург, 2018. - 336 с.9.

15. Петин В.А., Биняковский А.А. «Практическая энциклопедия Arduino»: - 2-е изд. - ДМК-Пресс, 2020. - 166 с.

16. Почанин Ю.С. «Робототехника в промышленности»: - SelfPub, 2022. - 330 с.

17. Рачков М.Ю. «Автоматизация производства. Учебник для СПО»: - 2-е изд. - Юрайт, 2018. - 181 с.

18. Ревич Ю.В. «Занимательная электроника»: - 5-е изд. - СПб.: БХВ-Петербург, 2018. - 672 с.

19. Улли Соммер «Программирование Микроконтроллерных Плат Arduino/Freeduino»: - СПб.: БХВ-Петербург, 2012. - 256 с.

20. Юревич Е.И. «Основы робототехники»: - 4-е изд. - СПб.: БХВ-Петербург, 2018. - 304 с.

21. Charles Platt Encyclopedia of Electronic Components Volume 1. - O'Reilly Media, Incorporated, 2012. - 278 с.

22. Gordon McComb «Arduino Robot Bonanza»: - McGraw-Hill Companies, 2013. - 416 с

23. Herb Schildt's «C++: A Beginner's Guide, Second Edition»: - 2-е, иллюстрированное изд. - McGraw-Hill Prof Med/Tech, 2003. - 576 с.

24. Jeremy Blum «Exploring Arduino Tools and Techniques for Engineering Wizardry»: - Wiley, 2013. - 384 с.

25. John-David Warren, Josh Adams, Harald Molle «Arduino Robotics»: - Apress, 2011. - 628 с.

Приложение А

Структурная схема устройства

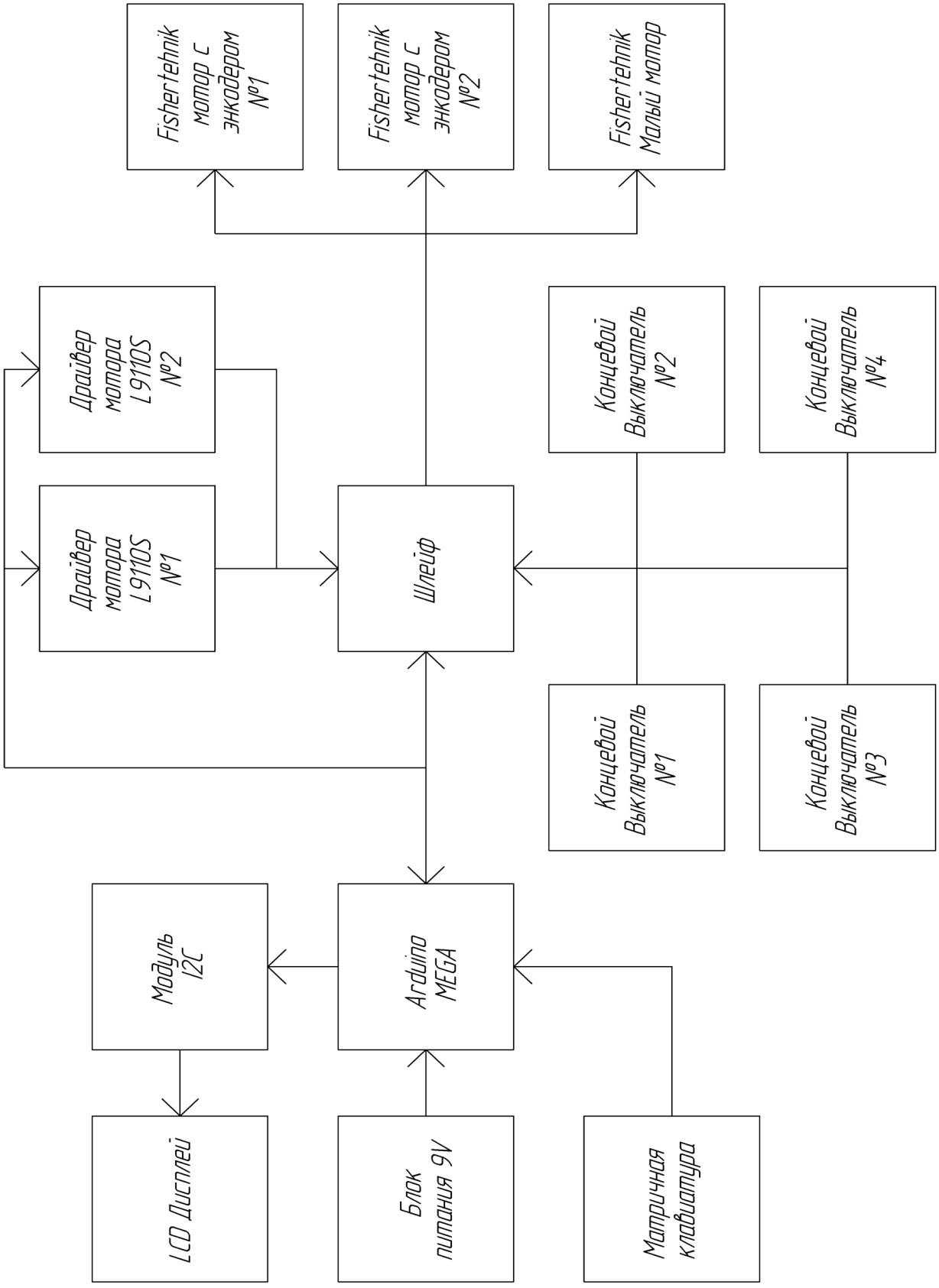

Рисунок А.1 – Структурная схема устройства

Приложение Б

Принципиальная электрическая схема устройства

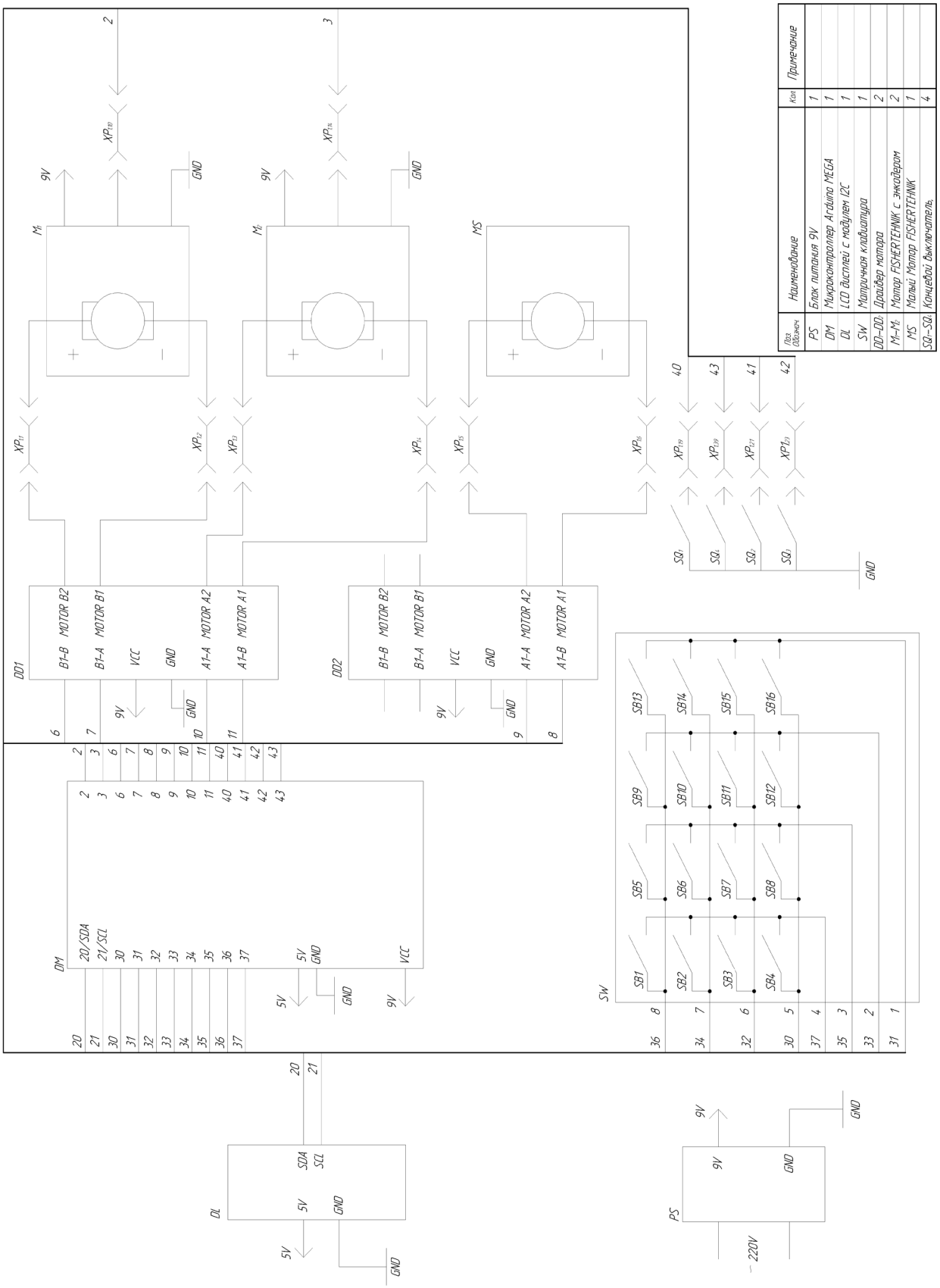

Рисунок Б.1 – Принципиальная электрическая схема устройства

Приложение В

Руководство по эксплуатации

Подключите питание к системе управления.

Для управления макетом используйте матричную клавиатуру.

Клавиши 1, 2, 3, 4, 5, 6 отвечают за автоматическое управление манипулятором. Отсек 1 является отсеком приёма, куда помещают объект, который необходимо транспортировать в другой отсек. Отсеки 2, 3, 4, 5, 6, являются отсеками хранения, куда можно поместить объект из отсека приёма. Разместите принимаемый объект в зоне досягаемости манипулятора.

Для того, чтобы взять объект с отсека приёма используйте клавишу 1. Как только объект будет захвачен, на дисплей будет выведена следующая надпись “Object captured” и “Press 0”.

Выполните указание с дисплея. Нажмите клавишу ноль и верните манипулятор в начальное положение. Как только манипулятор вернётся в положение по умолчанию, необходимо выбрать место, куда манипулятор выгрузит принятый объект.

Кнопка * позволит остановить движение моторов в текущем положении.

Для переключения режима управления в ручной режим используйте кнопку с символом #. Это заблокирует команды автоматического управления и разблокирует доступ к командам ручного управления на клавишах 7, 8, 9, A, B, C, D. Клавиши 7, 8, 9, позволяют выбрать мотор для управления.

Чтобы изменить дальность перемещения манипулятора используйте клавиши A и B. Для перемещения манипулятора используйте клавиши C и D.

Клавишу 0 вернёт манипулятор в состоянии по умолчанию. Однако, перед использованием данной клавиши, для нивелирования вероятности возникновения ошибки рекомендуется сместить положение манипулятора с начальных позиций на всех осях.

Крайне не желательно вытаскивать шлейф из разъёма со стороны устройства управления, чтобы не сломать устройство управления.

Приложение Г

Временные диаграммы

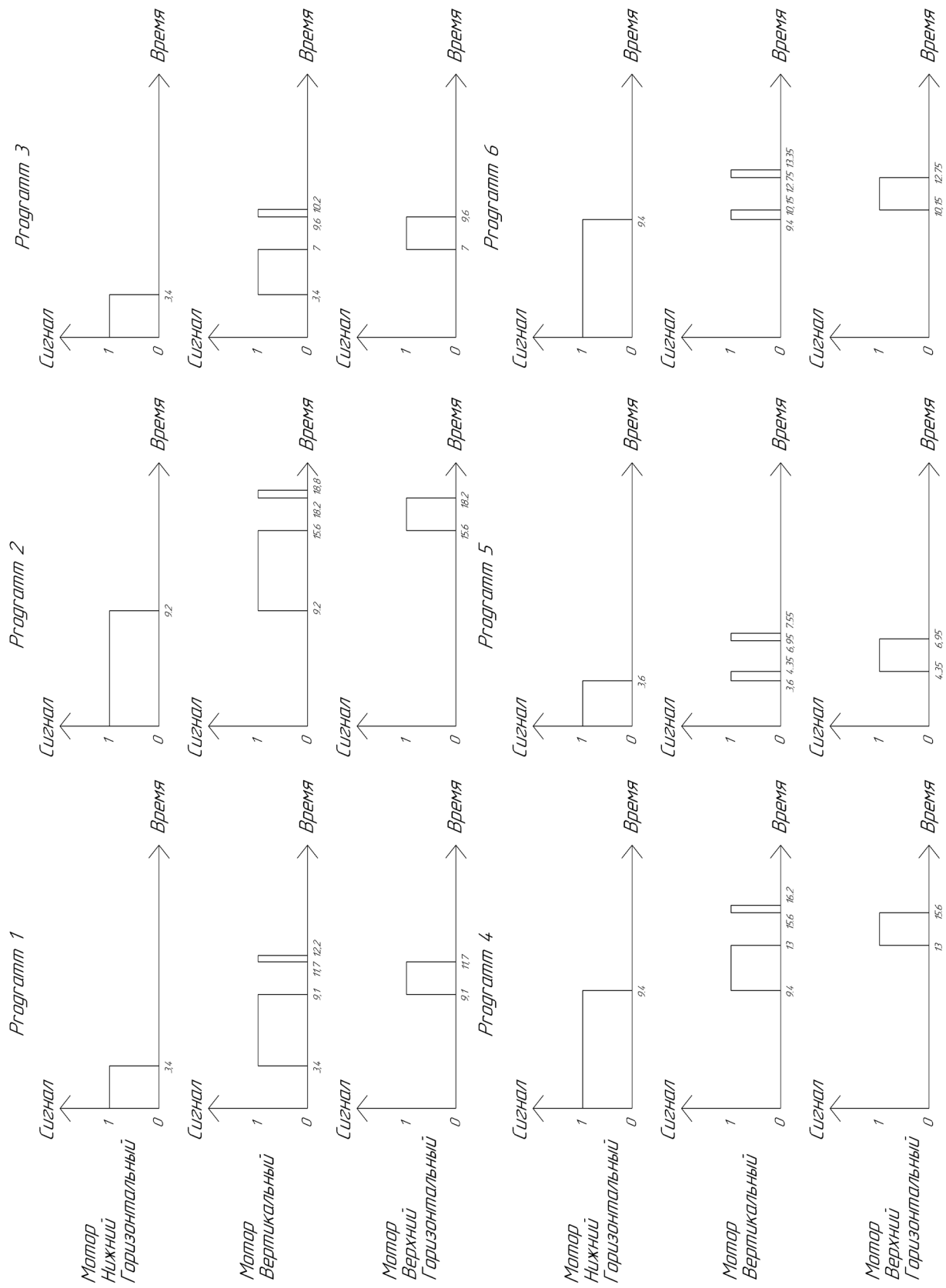

Рисунок Г.1 – Временная диаграмма

Приложение Д

Программный код алгоритма работы

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Stepper.h>
#define USE_BUTTON
#define Btn1 40
#define Btn2 41
#define Btn3 42
#define Btn4 43

uint8_t per = 0;
uint8_t old_per = 1;
volatile int counter = 0;
volatile bool intFlag = false;

//Моторы
#define MC1 6
#define MC2 7
int velocity = 0;

#define MC3 8
#define MC4 9
int velocity2 = 0;

#define MC5 10
#define MC6 11
int velocity3 = 0;

//Монитор
LiquidCrystal_I2C lcd(0x27, 16, 2);

//Клавиатура
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
```

```
};
byte rowPins[ROWS] = {36, 34, 32, 30};
byte colPins[COLS] = {37, 35, 33, 31};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);

  // Подключение кнопок
  pinMode(Btn1, INPUT_PULLUP);
  pinMode(Btn2, INPUT_PULLUP);
  pinMode(Btn3, INPUT_PULLUP);
  pinMode(Btn4, INPUT_PULLUP);

  // Подключение дисплея
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Choose");
  lcd.setCursor(0, 1);
  lcd.print("Programs");

  //Подключение клавиатуры
  Serial.println("Keypad initialized");
  customKeypad.setHoldTime(100);

  //Подключение двигателей
  pinMode(MC1, OUTPUT);
  pinMode(MC2, OUTPUT);

  pinMode(MC3, OUTPUT);
  pinMode(MC4, OUTPUT);

  pinMode(MC5, OUTPUT);
  pinMode(MC6, OUTPUT);



}

boolean block = false;
boolean hand = false;
```

```
int STATUSFLAG = 0;
boolean motor1 = false;
boolean motor2 = false;
boolean motor3 = false;

boolean comand0 = false;
boolean buttonN1 = false;
boolean buttonN2 = false;

int timedelay = 500;
int value = 50;

void loop() {
  bool BtnState1 = !digitalRead(Btn1);
  bool BtnState2 = !digitalRead(Btn2);
  bool BtnState3 = !digitalRead(Btn3);
  bool BtnState4 = !digitalRead(Btn4);
  char customKey = customKeypad.getKey();

  if (customKey == '1' && block == false && hand == false) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Program 1");

    digitalWrite(MC5, LOW);
    digitalWrite(MC6, HIGH);
    delay(3400);
    digitalWrite(MC5, LOW);
    digitalWrite(MC6, LOW);

    digitalWrite(MC1, LOW);
    digitalWrite(MC2, HIGH);
    delay(6000);
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, LOW);

    digitalWrite(MC3, LOW);
    digitalWrite(MC4, HIGH);
    delay(2600);
    digitalWrite(MC3, LOW);
    digitalWrite(MC4, LOW);
```

```
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  delay(500);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Object captured");
  lcd.setCursor(0, 1);
  lcd.print("Press 0");
}

if (customKey == '2' && block == false && hand == false) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Program 2");

  digitalWrite(MC5, LOW);
  digitalWrite(MC6, HIGH);
  delay(9200);
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, LOW);

  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  delay(6300);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  digitalWrite(MC3, LOW);
  digitalWrite(MC4, HIGH);
  delay(2600);
  digitalWrite(MC3, LOW);
  digitalWrite(MC4, LOW);

  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  delay(600);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);
```

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Object delivered");
  lcd.setCursor(0, 1);
  lcd.print("Press 0");
 }

 if (customKey == '3' && block == false && hand == false) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Program 3");

  digitalWrite(MC5, LOW);
  digitalWrite(MC6, HIGH);
  delay(3400);
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, LOW);

  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  delay(3600);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  digitalWrite(MC3, LOW);
  digitalWrite(MC4, HIGH);
  delay(2600);
  digitalWrite(MC3, LOW);
  digitalWrite(MC4, LOW);

  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  delay(500);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Object delivered");
  lcd.setCursor(0, 1);
  lcd.print("Press 0");
```

```
}

if (customKey == '4' && block == false && hand == false) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Program 4");

  digitalWrite(MC5, LOW);
  digitalWrite(MC6, HIGH);
  delay(9400);
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, LOW);

  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  delay(3700);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  digitalWrite(MC3, LOW);
  digitalWrite(MC4, HIGH);
  delay(2600);
  digitalWrite(MC3, LOW);
  digitalWrite(MC4, LOW);

  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  delay(500);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Object delivered");
  lcd.setCursor(0, 1);
  lcd.print("Press 0");
}

if (customKey == '5' && block == false && hand == false) {
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
 lcd.print("Program 5");

 digitalWrite(MC5, LOW);
 digitalWrite(MC6, HIGH);
 delay(3600);
 digitalWrite(MC5, LOW);
 digitalWrite(MC6, LOW);

 digitalWrite(MC1, LOW);
 digitalWrite(MC2, HIGH);
 delay(750);
 digitalWrite(MC1, LOW);
 digitalWrite(MC2, LOW);

 digitalWrite(MC3, LOW);
 digitalWrite(MC4, HIGH);
 delay(2600);
 digitalWrite(MC3, LOW);
 digitalWrite(MC4, LOW);

 digitalWrite(MC1, HIGH);
 digitalWrite(MC2, LOW);
 delay(600);
 digitalWrite(MC1, LOW);
 digitalWrite(MC2, LOW);

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Object delivered");
 lcd.setCursor(0, 1);
 lcd.print("Press 0");
}

if (customKey == '6' && block == false && hand == false) {
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Program 6");

 digitalWrite(MC5, LOW);
 digitalWrite(MC6, HIGH);
 delay(9400);
```

```
  digitalWrite(MC5, LOW);
  digitalWrite(MC6, LOW);

  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  delay(750);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  digitalWrite(MC3, LOW);
  digitalWrite(MC4, HIGH);
  delay(2600);
  digitalWrite(MC3, LOW);
  digitalWrite(MC4, LOW);

  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  delay(600);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Object delivered");
  lcd.setCursor(0, 1);
  lcd.print("Press 0");
 }

 if (customKey == '7' && block == false && hand == true) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Choose M1");
  motor1 = true;
  motor2 = false;
  motor3 = false;
 }
 if (customKey == '8' && block == false && hand == true) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Choose M2");
```

```
  motor1 = false;
  motor2 = true;
  motor3 = false;
}
if (customKey == '9' && block == false && hand == true) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Choose M3");
  motor1 = false;
  motor2 = false;
  motor3 = true;
}

if (customKey == '0' && block == false) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("By default");
  Yreverse();
}



if (customKey == '*') {
  if (!block) {
    block = true;
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Block control");
    //    Останавливаем моторы
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, LOW);
    digitalWrite(MC3, LOW);
    digitalWrite(MC4, LOW);
    digitalWrite(MC5, LOW);
    digitalWrite(MC6, LOW);
  }
  else if (block) {
    block = false; // Разблокировано
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Choose");
    lcd.setCursor(0, 1);
```

```
    lcd.print("Programs");
   }
}

if (customKey == '#' && block == false) {
  if (!hand) {
    hand = true;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Hand control");
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, LOW);
    digitalWrite(MC3, LOW);
    digitalWrite(MC4, LOW);
    digitalWrite(MC5, LOW);
    digitalWrite(MC6, LOW);
  }
  else if (hand) {
    hand = false;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Choose");
    lcd.setCursor(0, 1);
    lcd.print("Programs");
  }
}

if (customKey == 'A' && block == false && hand == true) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("timedelay+50");
  timedelay += 50;
  lcd.setCursor(0, 1);
  lcd.print(timedelay);
}
if (customKey == 'B' && block == false && hand == true) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("timedelay-50");
  timedelay -= 50;
  lcd.setCursor(0, 1);
```

```
  lcd.print(timedelay);
}

if (customKey == 'C' && block == false && hand == true && motor1 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Up value 1");
  digitalWrite(MC5, 0);
  digitalWrite(MC6, 10);
  delay(timedelay);
  digitalWrite(MC5, 0);
  digitalWrite(MC6, 0);
  lcd.setCursor(0, 1);
  lcd.print("          ");
}
if (customKey == 'C' && block == false && hand == true && motor2 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Up value 2");
  digitalWrite(MC1, 0);
  digitalWrite(MC2, 10);
  delay(timedelay);
  digitalWrite(MC1, 0);
  digitalWrite(MC2, 0);
  lcd.setCursor(0, 1);
  lcd.print("          ");
}
if (customKey == 'C' && block == false && hand == true && motor3 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Up value 3");
  digitalWrite(MC3, 0);
  digitalWrite(MC4, 10);
  delay(timedelay);
  digitalWrite(MC3, 0);
  digitalWrite(MC4, 0);
  lcd.setCursor(0, 1);
  lcd.print("          ");
}

if (customKey == 'D' && block == false && hand == true && motor1 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Down value 1");
  digitalWrite(MC5, 10);
```

```
  digitalWrite(MC6, 0);
  delay(timedelay);
  digitalWrite(MC5, 0);
  digitalWrite(MC6, 0);
  lcd.setCursor(0, 1);
  lcd.print("            ");
 }
 if (customKey == 'D' && block == false && hand == true && motor2 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Down value 2");
  digitalWrite(MC1, 10);
  digitalWrite(MC2, 0);
  delay(timedelay);
  digitalWrite(MC1, 0);
  digitalWrite(MC2, 0);
  lcd.setCursor(0, 1);
  lcd.print("            ");
 }
 if (customKey == 'D' && block == false && hand == true && motor3 == true) {
  lcd.setCursor(0, 1);
  lcd.print("Down value 3");
  digitalWrite(MC3, 10);
  digitalWrite(MC4, 0);
  delay(timedelay);
  digitalWrite(MC3, 0);
  digitalWrite(MC4, 0);
  lcd.setCursor(0, 1);
  lcd.print("            ");
 }
}

void Yreverse() {
 digitalWrite(MC3, HIGH);
 digitalWrite(MC4, LOW);
 while (digitalRead(Btn2) == 1) {
  delay (1);
 }
 digitalWrite(MC3, LOW);
 Zreverse();
}
```

```
void Zreverse() {
  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  while (digitalRead(Btn3) == 1) {
    delay (1);
  }
  digitalWrite(MC1, LOW);
  Xreverse();
}

void Xreverse() {
  digitalWrite(MC5, HIGH);
  digitalWrite(MC6, LOW);
  while (digitalRead(Btn1) == 1) {
    delay (1);
  }
  digitalWrite(MC5, LOW);
}
```

Приложение Е

Блок схема алгоритма програм

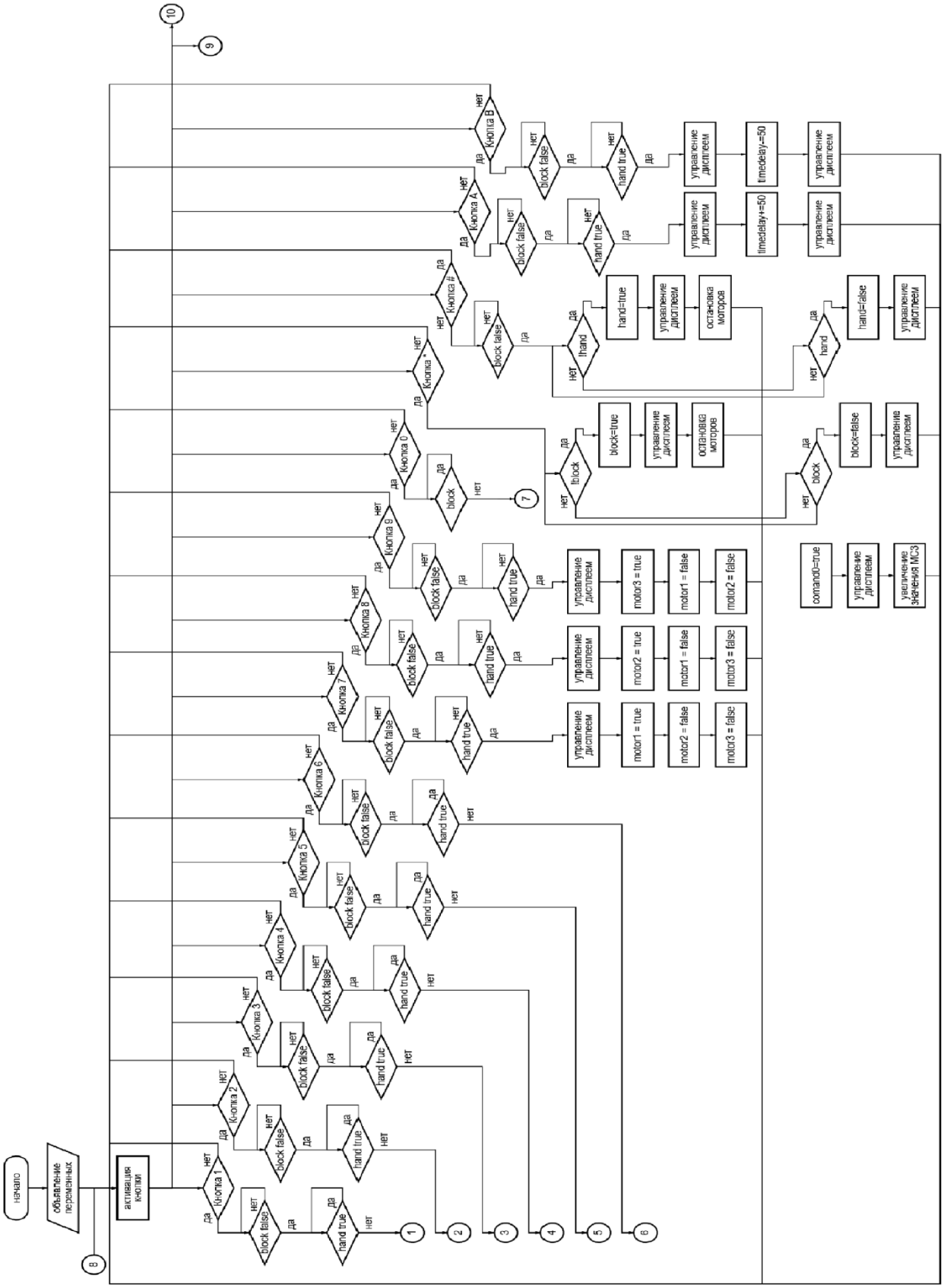

Рисунок Е.1 – Блок схема программы

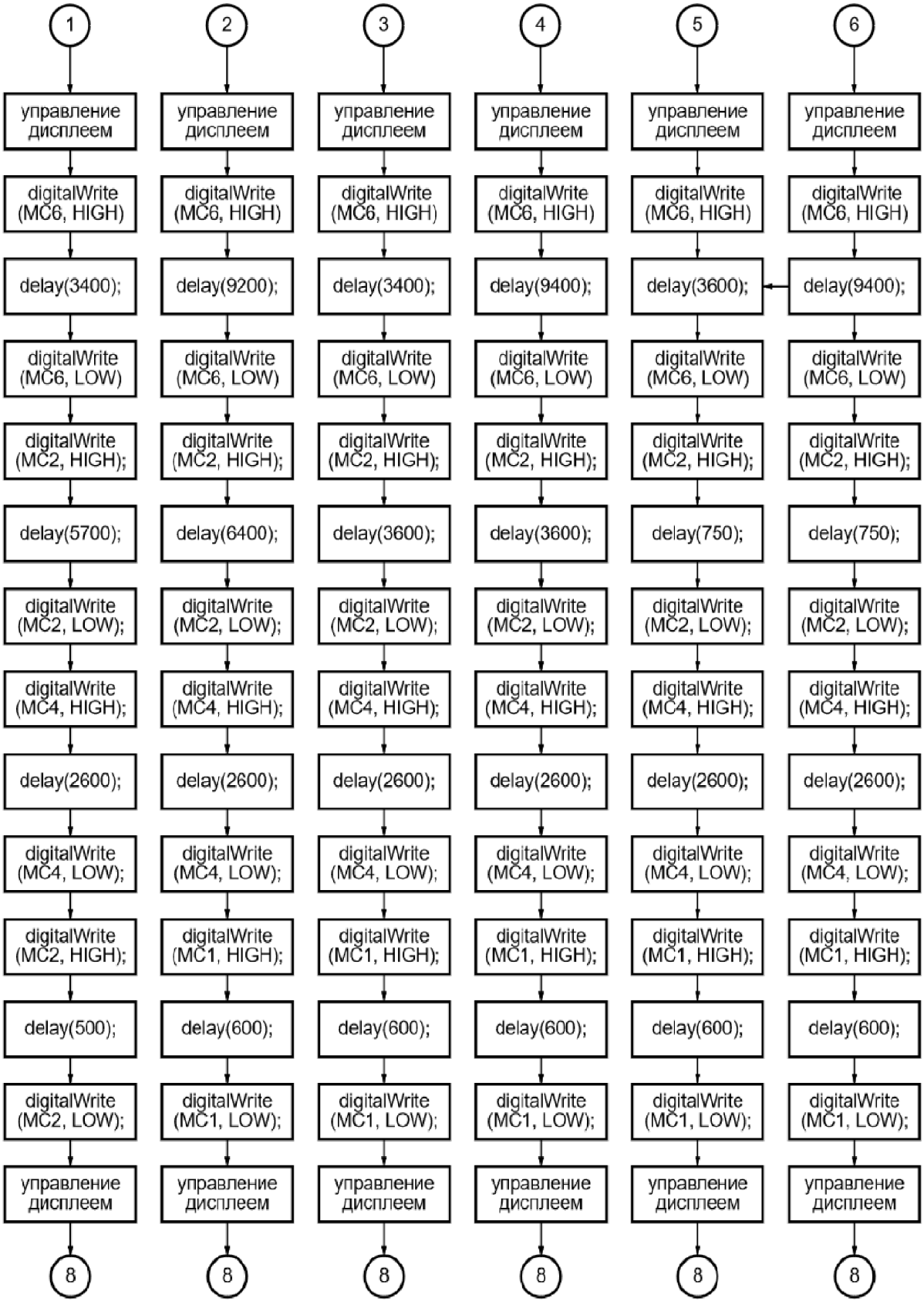

Рисунок Е.2 – Блок схема программы

Рисунок Е.3 – Блок схема программы

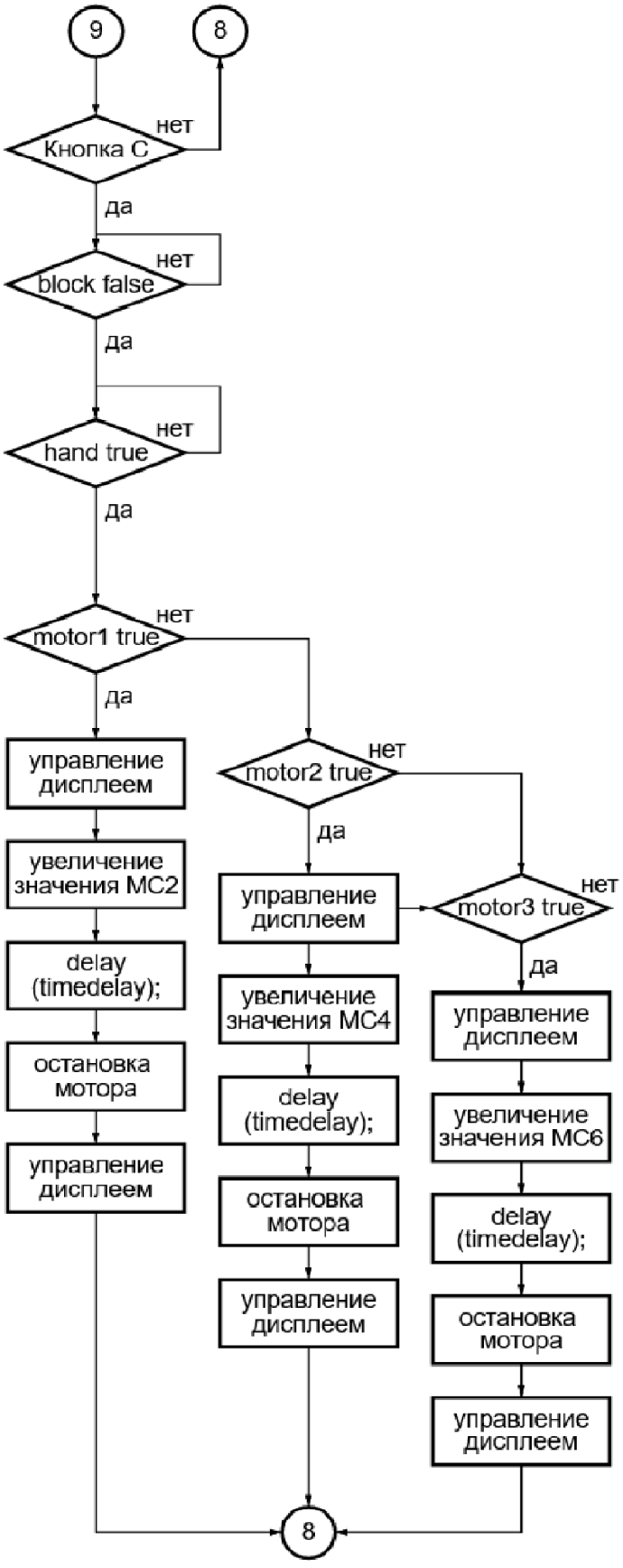

Рисунок Е.4 – Блок схема программы

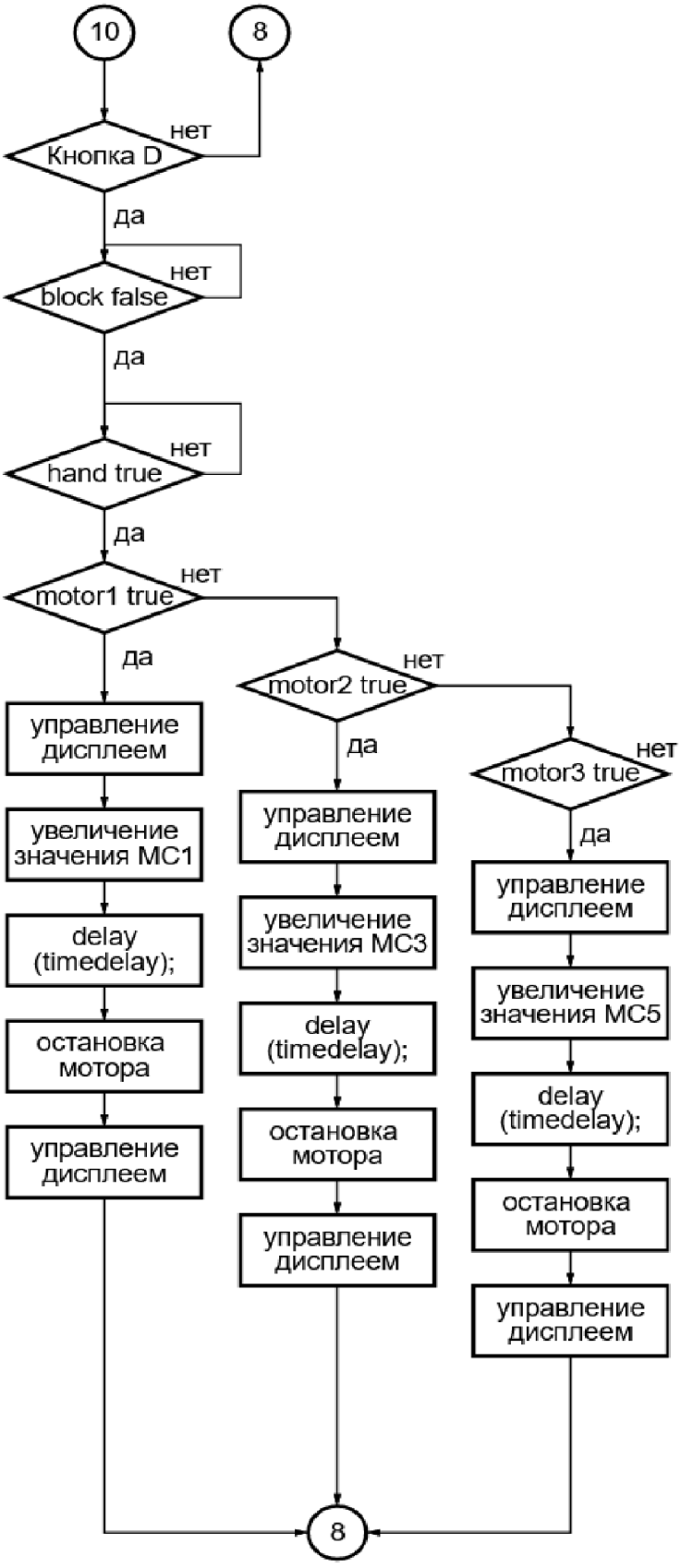

Рисунок Е.5 – Блок схема программы