Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет энергетический
Кафедра автоматизации производственных процессов и электротехники
Направление подготовки 15.03.04 - Автоматизация технологических процессов и производств
Профиль Автоматизация технологических процессов и производств в энергетике

ДОПУСТИТЬ К ЗАЩИТЕ
И.о.зав. кафедрой
_____ О.В. Скрипко
«06» июля 2020 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR

Исполнитель студент группы 641об	01.07.2020 (подпись, дата)	М.С. Потемкин
Руководитель профессор, д-р техн.наук	01.07.2020 г. (подпись, дата)	О.В. Скрипко
Консультант по безопасности и экологичности доцент, канд физ.-мат. наук	02.07.2020 (подпись, дата)	В.Н. Аверьянов
Нормоконтроль профессор, д-р техн.наук	01.07.2020 (подпись, дата)	О.В. Скрипко

Благовещенск 2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет Энергетический
Кафедра автоматизации производственных процессов и электротехники

УТВЕРЖДАЮ
И.о.зав. кафедрой
_____О.В. Скрипко
«__06__» ___июня___ 2020 г.

ЗАДАНИЕ

К выпускной квалификационной работе студента *641 группы Потемкина Михаила Сергеевича*

1. Тема выпускной квалификационной работы: Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR

(утверждена приказом от 30.04.2020. № 810-уч)

2. Срок сдачи студентом законченного проекта: *1 июля 2020 года.*

3. Исходные данные к выпускной квалификационной работе: *1) ФГОС направления подготовки бакалавров 15.03.04 Автоматизации технологических процессов и производств; 2) Учебный план направления подготовки бакалавров 15.03.04 Автоматизации технологических процессов и производств; 3) Техническое задание.*

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов):

1) Техническое описание робота-манипулятора и имеющихся систем управления;

2) Разработка системы управления на микроконтроллерной платформе Arduino mega 2560;

3) Разработка протокола обмена данными между ПК и МК;

4) Разработка ручного управления роботом с использованием матричной клавиатуры;

5) Разработка интерфейса пользователя для управления роботом в ручном режиме, дискретном, позиционном и автоматическом режимах.

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.):

Лист 1: Робот-манипулятор «УР-4».

Лист 2: Система управления роботом.

Лист 3: Принципиальная схема системы управления роботом.

Лист 4: Алгоритм работы протокола обмена данными.

Лист 5: Ручное управление роботом.

Лист 6: Оконное приложение управления роботом.

6. Дата выдачи задания 10.03.2020

Руководитель выпускной квалификационной работы Скрипко Ольга Валерьевна, профессор кафедры АПП и Э, доктор техн. наук.

Задание принял к исполнению (дата): 10.03.2020 _____

(подпись студента)

РЕФЕРАТ

Бакалаврская работа содержит 79 с., 67 рисунков, 10 таблиц, 12 источников.

МИКРОКОНТРОЛЛЕР, ПРОТОКОЛ ОБМЕНА ДАННЫМИ, STEP, DIR, ДРАЙВЕР, ПК, ШАГОВЫЙ ДВИГАТЕЛЬ, РОБОТ-МАНИПУЛЯТОР, АЛГОРИТМ, КОНЦЕВЫЕ ДАТЧИКИ, МАТРИЧНАЯ КЛАВИАТУРА

В работе объектом исследования и последующей модернизации является учебный робот–манипулятор «УР-4». Данная работа направлена на разработку автоматизированной системы управления на базе микроконтроллера, с учетом всех минусов предыдущей системы.

Целью работы разработка протокола обмена данными между ПК и МК, и алгоритмов управления с ПК и в ручном режиме.

В ходе выполнения работы был разработан протокол обмена данными между МК и ПК, реализован протокол STEP/DIR для управления шаговыми двигателями. Создано ПО для управления звеньями робота с ПК.

Результатом разработки является система управления роботом–манипулятором на базе микроконтроллерной платформы Arduino mega 2560. Также данная система управления предоставляет роботу перемещать схват по непрерывному движению.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

СУР – система управления роботом;

USB – universal serial bus;

МК – микроконтроллер;

ПО – программное обеспечение;

AVR – семейство микроконтроллеров фирмы Atmel;

ШД – шаговый двигатель;

ДШД – драйвер шагового двигателя;

ПК – персональный компьютер;

УР – учебный робот.

ВВЕДЕНИЕ

Научной проблемой проекта является разработка системы управления роботом (СУР) на базе микроконтроллера, поддерживающую обмен данными между микроконтроллером и персональным компьютером через USB порт, предоставляющая возможность пользователю программировать траекторию движения робота.

Целесообразность работы заключается в разработке новой системы управления на МК. Создание ПО производится с целью управления роботом с компьютера.

Основанием выполнение проекта является:

- знание архитектуры микроконтроллеров AVR, струуктуру языков программирования;
- умение программировать в средах Arduino IDE микроконтроллерную платформу Arduino и в среде Microsoft visual studio создавать оконные приложения, приобретенное в студенческом конструкторском бюро;
- грант на разработку СУР роботом манипулятором.

Техническая база и программные средства для выполнения проекта:

- робот манипулятор «УР-4»;
- драйверы шаговых двигателей TB6560-V2;
- микроконтроллер фирмы Arduino – Arduino mega 2560;
- среда программирования микроконтроллеров фирмы Arduino - Arduino IDE 1.8.12;
- Microsoft visual studio 2012 – средство для разработки оконных приложений windows forms c#.

Актуальность данного проекта заключается в следующем, исходная система управления роботом манипулятором на базе котроллера OMRONCP1L-M30DT-D, по типу движения, относится к цикловой, но

движение робота происходит одновременно только по двум осям. Для программирования движения используются языки: релейных диаграмм LD и язык функциональных блок-схем FBD. Данная модернизация робота состояла в переводе СУР на дискретное позиционное управление движением по всем координатам. Для чего была выбрана CNC-система Mach 3, использующая G-коды. Далее была создана система управления на базе микроконтроллера PIC16 и создано оконное приложение для управления роботом, отказавшись от предыдущей системы использовавшую CNC – систему. Настоящая модернизация заключается в разработке микроконтроллерной СУР, осуществляющей дискретное позиционное по всем координатам непрерывное движение, с открытым и понятным протоколом управления шаговыми двигателями. Позволяющая программировать траекторию движения с использованием функций API Windows и распространённых языков программирования.

Новизна проекта состоит в том, что для обмена данными между компьютером и микроконтроллером используется USB интерфейс, в отличие от предыдущих СУ, что является удобнее в использовании. СУР представленная в проекте осуществляет дискретное позиционное по всем координатам непрерывное движение, позволяющая пользователю программировать траекторию движения.

Робот-манипулятор является лабораторным стендом предназначенным для изучения технических характеристик и построения систем автоматизации. Учебный робот предоставляет возможность управлять схватом в ручном, дискретном, позиционном и автоматическом режимах.

Практическая значимость данного проекта заключается в совмещении работы компьютера и микроконтроллера, посредством обмена данными по USB порту, для управления роботом манипулятором «УР-4».

1 ОПИСАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ

В данном проекте объектом автоматизации является лабораторный стенд «Средства автоматизации и управления робота манипулятора «САУ-РОБОТ», предназначенный для наглядного изучения технических характеристик, а также построения системы автоматизации на базе робота манипулятора. Внешний вид робота изображен на рисунке 1.

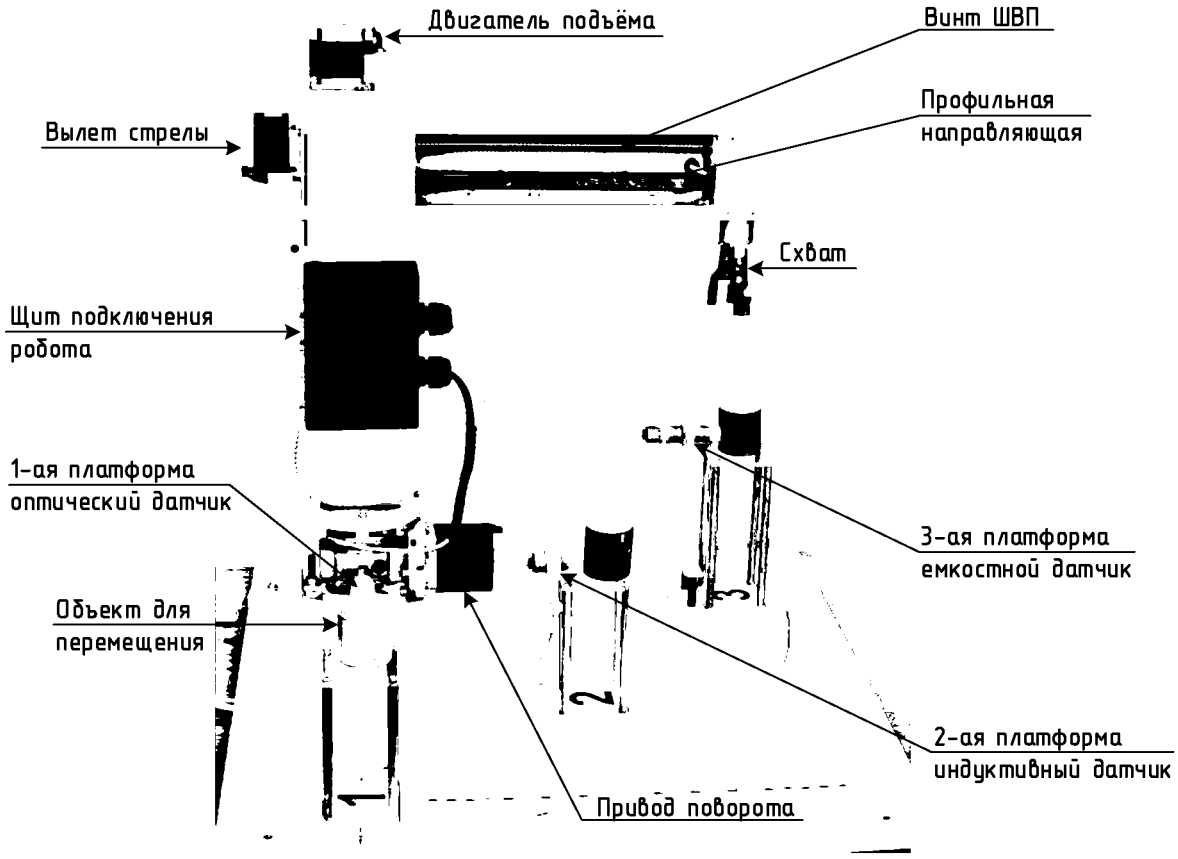


Рисунок 1 - Учебный робот

«УР - 4» установлен на металлическую платформу с координатной сеткой с нанесенной на ней системой координат. Рабочее пространство робота оснащено различными датчиками технологической информации. «УР - 4» имеет возможность работать в цилиндрических координатах, для чего на платформе нанесены углы поворота платформы(см. рис. 2).

Робот – манипулятор может осуществлять движение по трем осям: вылет стрелы – ось X; спуск/подъем стрелы – ось Y; поворот стрелы вокруг своей оси – ось Z. Также приводить в движение схват робота.

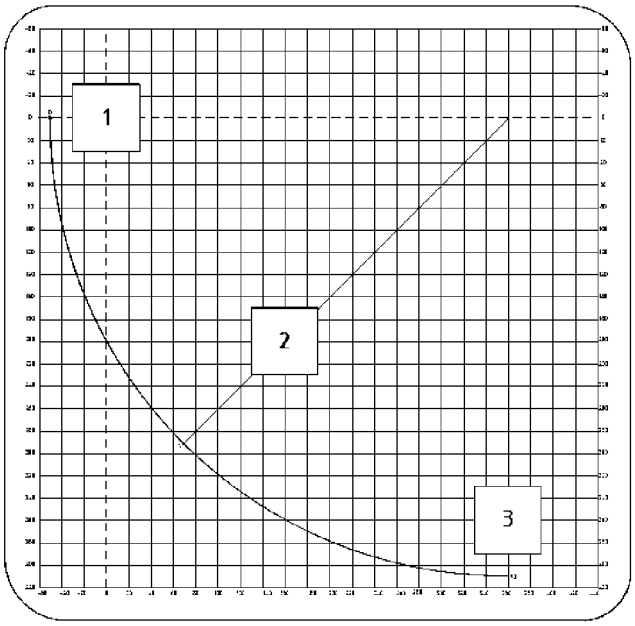


Рисунок 2 – Платформа

За спуск/подъем и вылет стрелы отвечают шаговые двигатели GD57STH56-2808A.

Таблица 1 - Технические характеристики серии GD57STH56

Наименование	Значение
Угловой шаг	0,9°
Погрешность углового шага	±5% (полный шаг, без нагрузки)
Погрешность сопротивления	±10%
Погрешность индуктивности	±20%
Повышение температуры	80°CMax.(рабочий ток, 2 фазы)
Рабочая температура	-20°C~+50°C
Сопротивление изоляции	100MΩMin. ,500VDC
Диэлектрическая прочность	500VAC for one minute
Радиальное биение вала	0,02Max. (450 g-load)
Осевое биение вала	0,08Max. (450 g-load)
Максимальная Радиальная сила	75 N
Максимум. Осевое усилие	15 N

Таблица 2 - Электромеханические характеристики

Наименование	Значение	Ед. измерения
Напряжение питания	4,5	В
Ток /фаза	3	А
Сопротивление/фаза	2,25	Ω(Ом)
Индуктивность/фаза	2,5	мГн
Крутящий момент	12,6	кг•см
Кол-во выводов	8	Шт.
Момент инерции ротора	300	г-см^2
Вес	0,7	кг
Момент удержания	0,4	кг•см

За поворот стрелы вокруг своей оси отвечает шаговый двигатель GD57STH76 марки 2006A.

Таблица 3 - Электромеханические характеристики GD57STH76-2006A

Наименование	Значение	Ед. измерения
Напряжение питания	4,5	В
Ток /фаза	2	А
Сопротивление/фаза	2,25	Ω(Ом)
Индуктивность/фаза	3,6	мГн
Крутящий момент	13,5	кг•см
Кол-во выводов	6	Шт.
Момент инерции ротора	480	г-см^2
Вес	1	кг
Момент удержания	0,68	кг•см

Схват робота приводится в движение гибридным шаговым двигателем Nema 17 42CM06.

Таблица 4 - Технические характеристики Nema 17 42CM06

Наименование	Значение
Угловой шаг	1,9°
Погрешность углового шага	±5% (полный шаг, без нагрузки)
Удерживающий момент	0,34 *Н•м*
Номинальный ток	1,5 А
Индуктивность/фаза	5 мГн
Сопротивление/фаза	2,6 Ω(Ом)
Повышение температуры	80°CMax.(рабочий ток, 2 фазы)
Рабочая температура	-20°C~+50°C
Сопротивление изоляции	100MΩMin. ,500VDC
Вес	0,34 кг

Из выше сказанного можно сделать вывод о том, что звенья робота-манипулятора осуществляют движения с помощью шаговых двигателей установленных на осях робота.

Кинематическая схема робота см. ПРИЛОЖЕНИЕ А.

1.1 Система управления предложенная Пилецким П.Е.

В 2017 году была собрана система управления ШД представленная на рисунке 3. Каждый ШД подключен непосредственно к своему драйверу шагового двигателя. Драйвер управляет ШД по протоколу STEP/DIR. Сигнал STEP – шаг, один импульсный сигнал инициирует поворот двигателя на один шаг. Сигнал DIR – сигнал задающий направления вращения двигателя. Данные сигналы поступают с ПК, проходя через плату коммутации драйверов шаговых двигателей, все управление осуществляется с помощью программного обеспечения, для ЧПУ – станков, MACH3. Подключение данной системы управления к компьютеру осуществляется через LPT-порт.

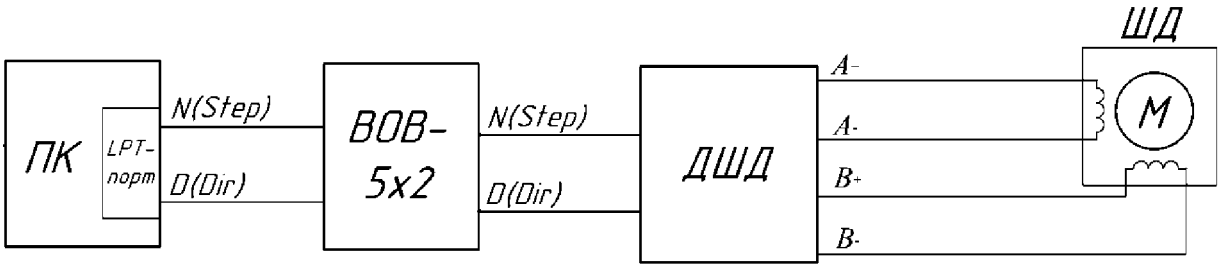


Рисунок 3 - СУ ШД

Схема включает в себя:

- ДШД – драйвер шагового двигателяTB6560-1;
- ШД – шаговый двигатель;
- BOB 5x2 – плата коммутации драйверов шаговых двигателей;
- A+,A-,B+ и B- – выводы обмоток двигателя;
- N – число шагов (сигнал Step);
- D – направление вращения двигателя (сигнал Dir);
- *LPT*-порт– параллельный порт ПК;
- ПК – персональный компьютер.

Плата коммутации драйверов шаговых двигателей – BOB-5x2, визуальный вид представлен на рисунке 4. Подключается к ПК осуществляется через LPT-порт. Поддерживает управление пятью осями. Имеет пять дискретных входов и один аналоговый выход 0-10 В. Так же есть возможность управления силовой нагрузкой с помощью реле. Все управляющие входы оптоизолированы. Выводы платы показаны на рисунке 5, назначение выводов в таблице 5 [1].

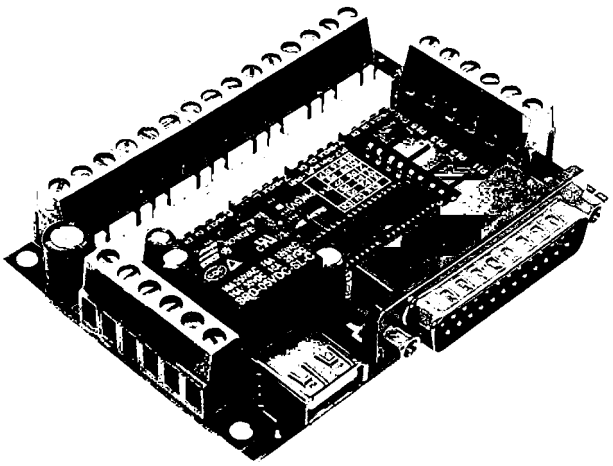


Рисунок 4 - Плата BOB 5x2

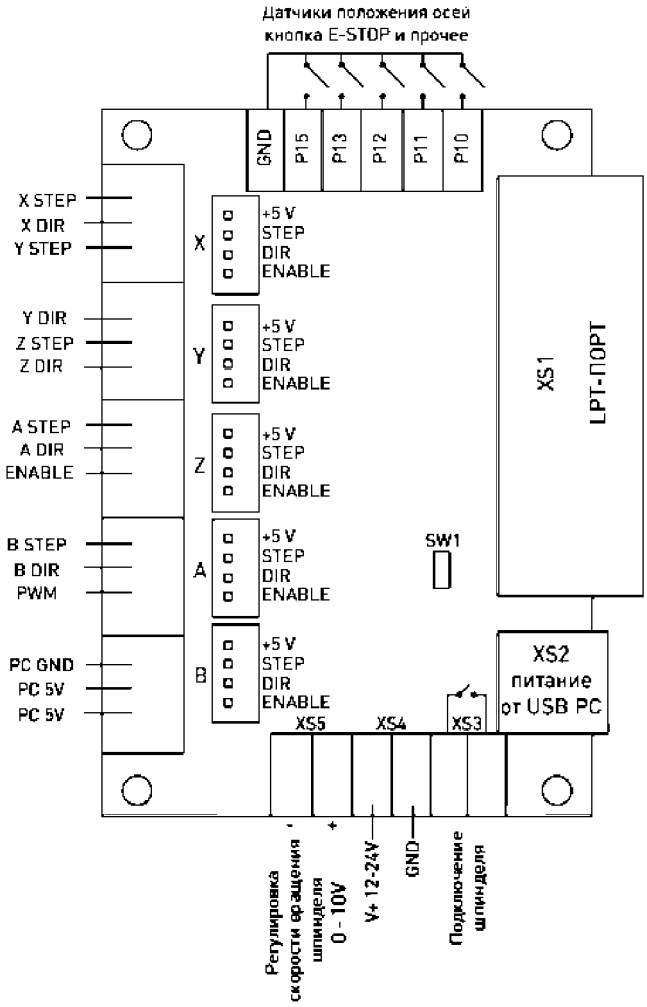


Рисунок 5 – Выводы модуля

Таблица 5 - Назначение выводов

Вывод	Назначение
XS1	LPT порт для подключения к компьютеру
XS2	Подача питания от компьютера через USB
XS3	Подключение шпинделя
XS4	Подключение питания
12-24V	Напряжение питания (V+)
GND	Напряжение питания (GND)
XS5	Регулировка скорости вращения шпинделя
XS6	Подключение датчиков положения осей,сигнала Estop и др
PC5V	Вывод питания от PC (+5V)
XCLK	X STEP
XDIR	X DIR
YCLK	Y STEP
YDIR	Y DIR
ZCLK	Z STEP
ZDIR	Z DIR
ACLK	A STEP
ADIR	A DIR
BCLK	B STEP
BDIR	B DIR
EN	Общий ENABLE
PWM	Выход ШИМ

Драйвер шагового двигателя TB6560-V2– это электронное устройство, которое управляет шаговым двигателем по результатам полученных сигналов управления, визуальный вид представлен на рисунке 6. Основные характеристики данных драйверов, представлены в таблице 6.

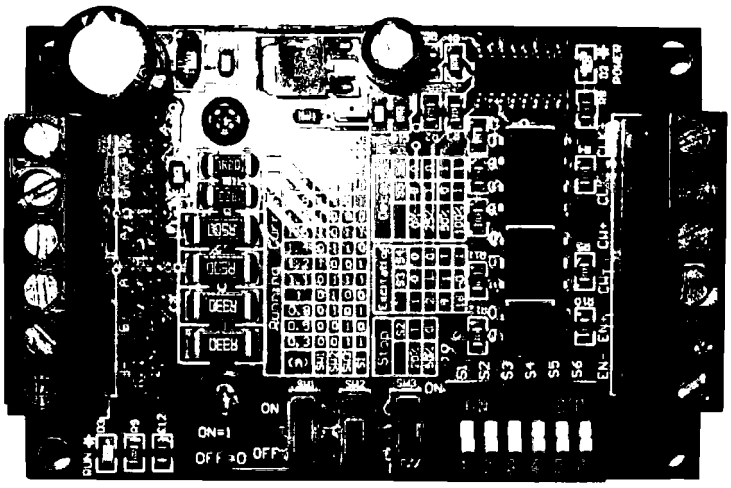


Рисунок 6 - Драйвер TB6560-V2

Таблица 6 - Технические характеристики драйвера TB6560-V2

Наименование показателя	Значение
1	2
Напряжение питания.	10 … 35 В Рекомендуется 24 В.
Ток фазы	0,3 … 3 А .Устанавливается переключателями на плате. Всего 14 ступеней регулировки.
Режимы	шаговый; полу шаговый; микро шаговый 1/8 шага; микро шаговый 1/16 шага. Режим задается переключателями на плате.
Интерфейс	Оптоизолированный STEP/DIR /ENABLE.

1	2
Максимальный ток сигналов CW (DIR) и EN (ENABLE)	50 мА
Максимальный ток сигнала CLK (STEP)	20 мА
Ток сигналов CW (DIR), EN (ENABLE), CLK (STEP) при 5 В на входе (ограничен резисторами модуля)	11 мА
Максимальная частота сигнала CLK (STEP)	15 кГц
Минимальная длительность импульса сигнала CLK (STEP)	7,7 мкс
Защита от перегрева	170 °С
Рабочая температура	- 10 … + 45 °С
Габариты	75 х 50 х 35 мм
Вес	73 г

Цоколевка драйвера и назначение выходов представлены на рисунке 7 и таблице 7 соответственно.

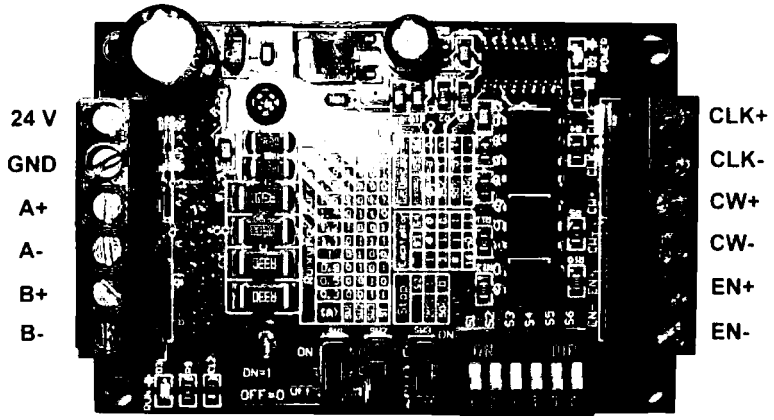


Рисунок 7 - Цоколевка драйвера TB6560-V2

Таблица 7 - Назначение выводов драйвера TB6560-V2

Вывод	Назначение
CLK+	Контакты тактового сигнала STEP. Ток ограничен резистором 330 Ом.
CLK -	
CW+	Контакты сигнала направления вращения DIR. Ток ограничен резистором 330 Ом.
CW-	
EN+	Контакты сигнала разрешения работы ENABLE. Ток ограничен резистором 330 Ом.
EN-	
B-	Фазная обмотка B
B+	
A-	Фазная обмотка A
A+	
GND	Общий провод
24 V	Положительный сигнал питания

1.2 Система управления предложенная Бова Д.Е.

В 2018 году была модернизирована система управления ШД путем добавления микроконтроллера PIC16F870. Разработанная система на МК из семейства PIC поддерживает три режима управления: ручной, автоматический и покоординатный. Также имеет возможность управления всеми осями робота. Структурная схема системы представлена на рисунке 8. В данной системе управления для формирования задержек с большой точностью в микросекундах применяется микроконтроллер.

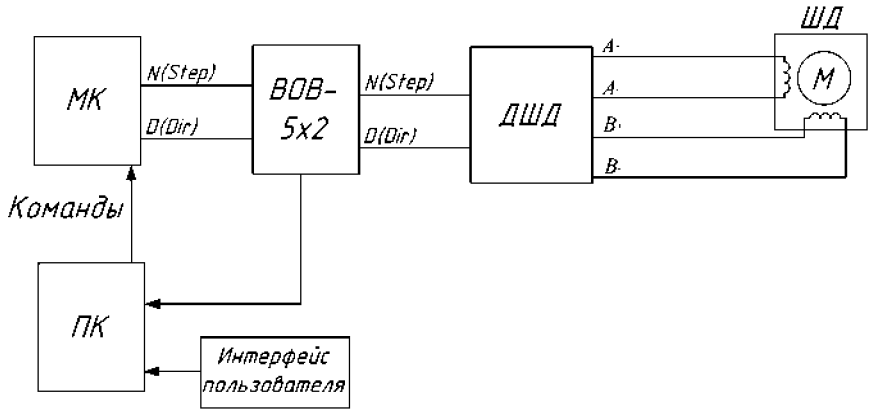


Рисунок 8 - СУ ШД с МК PIC16F870

Схема включает в себя:

- ДШД - драйвер шагового двигателя;
- ШД - шаговый двигатель;
- BOB 5x2 - плата коммутации драйверов шаговых двигателей;
- A+, A-, B+ и B- - выводы обмоток двигателя;
- N - число шагов (сигнал Step);
- D - направление вращения двигателя (сигнал Dir);
- ПК - персональный компьютер;
- МК – микроконтроллер PIC.

В данной системе управления микроконтроллер применяется для реализации схемы управления протоколом STEP/DIR. На рисунке 9 представлен микроконтроллер PIC16F870, в таблице 8 представлены технические характеристики микроконтроллера.

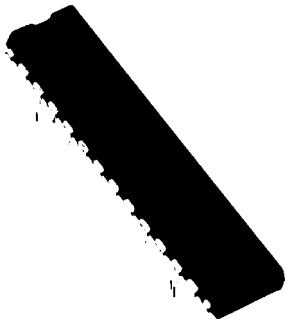


Рисунок 9 - PIC16F870

Таблица 8 - Технические характеристики PIC16F870

Наименование показателя	Значение
Ядро	PIC16
Ширина шины данных	8 bit
Максимальная тактовая частота	20 мГц
Размер программной памяти	3,5 кБ
Размер ОЗУ данных	128 Б
Количество входов/выходов	22
Рабочее напряжение питания	2 – 5,5 В
Тип корпуса	DIP - 28
Тип памяти программ	FlashStandardPackQty
Доступные аналоговые/цифровые каналы	5 шт
Разрядность АЦП	10 bit

2 РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ НА МИКРОКОНТРОЛЛЕРЕ AVR

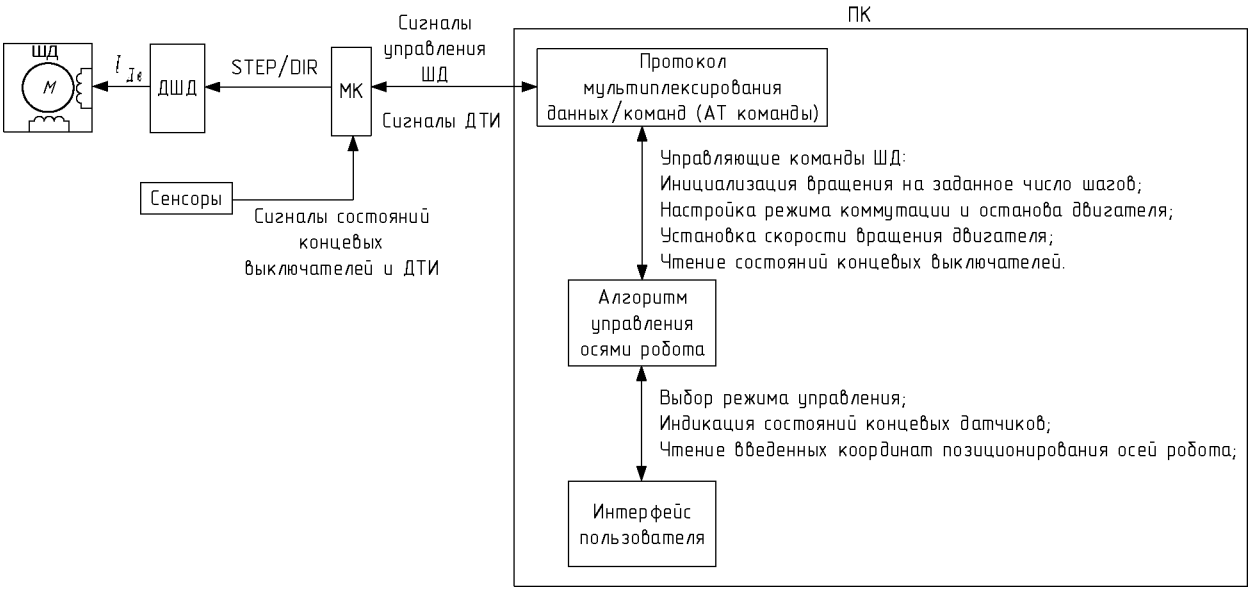


Рисунок 10 – Структурная схема разрабатываемой системы управления роботом манипулятором

В разрабатываемой системе управления добавлены датчики концевых выключателей противоположны начальному положению робота, заменен двигатель на схвате робота.

Основными компонентами системы управления являются: микроконтроллер, драйверы шаговых двигателей, датчики концевые и технологической информации, персональный компьютер.

Микроконтроллер используется для подачи сигналов на ДШД, т.е. сигнал шага в виде единичного импульса, сигнал направления вращения вала двигателя и сигнал разрешения работы двигателя. Также микроконтроллер опрашивает сенсоры, ведет обмен данными с ПК используя текстовый протокол обмена данными, отправляя компьютеру информацию ДТИ и концевых выключателей. Еще одной функцией является передача числа шагов шагового двигателя в протокол STEP/DIR, на выходе которого осуществляется подача сигналов на ДШД.

Драйвера на основании полученных сигналов от микроконтроллера осуществляет коммутацию обмоток двигателей, которые приводят в движение робота по осям: X, Y, Z и схват.

На персональном компьютере находится оконное приложение для управления роботом, которое предоставляет пользователю: выбор режима работы, управление в ручном режиме, позиционирование осей робота, индикацию состояний датчиков концевых и технологической информации. В приложении содержатся алгоритмы работы управления осями робота, который обрабатывает полученные сигналы позиционирования робота в ручном и координатных режимах в число шагов и направления вращения шагового двигателя. Компьютер поддерживает связь с микроконтроллером через USB порт.

В данном проекте микроконтроллерная платформа Arduino mega 2560, визуальный вид представлен на рисунке 11.

Arduino – это маленькое электронное устройство, состоящее из одной печатной платы, которое способно управлять разными датчиками, электродвигателями, освещением, передавать и принимать данные.

«Мозг» Arduino – это микроконтроллер семейства AVR. Микроконтроллер представляет из себя микропроцессор с памятью и различными периферийными устройствами, реализованный на одной микросхеме. Фактически это однокристальный микрокомпьютер, который способен выполнять относительно простые задачи [2,3].

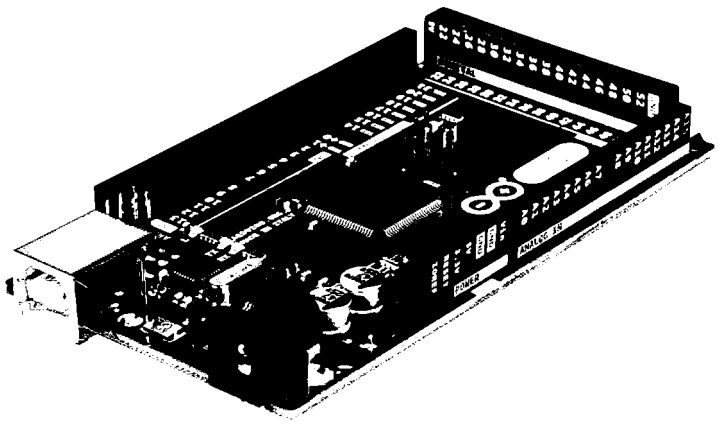


Рисунок 11 - Arduino mega 2560

Таблица 9 - Характеристики Arduino mega 2560

Наименование показателя	Значение
Микроконтроллер	ATmega2560
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	54 (из них 15 могут использоваться в качестве ШИМ - выходов)
Аналоговые входы	16
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода 3.3V	50 мА
Flash - память	256 КБ (ATmega2560) из которых 8 КБ используется загрузчиком
SPAM	8 КБ (ATmega2560)
EEPROM	4 КБ (ATmega2560)
Тактовая частота	16 Мгц

Функциональная схема системы управления см. ПРИЛОЖЕНИЕ Б.

3 ПРОТОКОЛ STEP/DIR ДЛЯ УПРАВЛЕНИЯ ШАГОВЫМИ ДВИГАТЕЛЯМИ

3.1 Назначение протокола STEP/DIR

В нашем случае для управления ШД используются драйверы шаговых двигателей TB6560-V2, подробное описание драйвера представлено в разделе 1.

Управление шаговыми двигателями осуществляется по протоколу STEP/DIR. Для управления используются три сигнала

Сигнал STEP - шаг. Каждый импульс инициирует поворот двигателя на один шаг. Если драйвер работает в полу шаговом или микро шаговом режимах, то поворот происходит не на физический шаг двигателя, а на часть шага, определяемого режимом. Для полу шагового режима это половина физического шага, для микро шагового – микро шаг. Частота следования импульсов, определяющая скорость вращения вала двигателя.

Сигнал DIR – сигнал задающий направления вращения двигателя. При высоком уровне сигнала двигатель вращается по часовой стрелке, при низком против часовой. Сигнал DIR должен формироваться раньше формирования импульсов сигнала STEP.

Сигнал ENABLE – сигнал разрешения работы драйвера. Разрешающий уровень сигнала ENABLE – низкий, т.е. отсутствие напряжения, запрещающий естественно наоборот.

Исходя из всего выше сказанного можно составить временную диаграмму работы протокола STEP/DIR (см. рис. 12), в дальнейшем на основании этой диаграммы будет написано программное обеспечение для управления шаговыми двигателями.

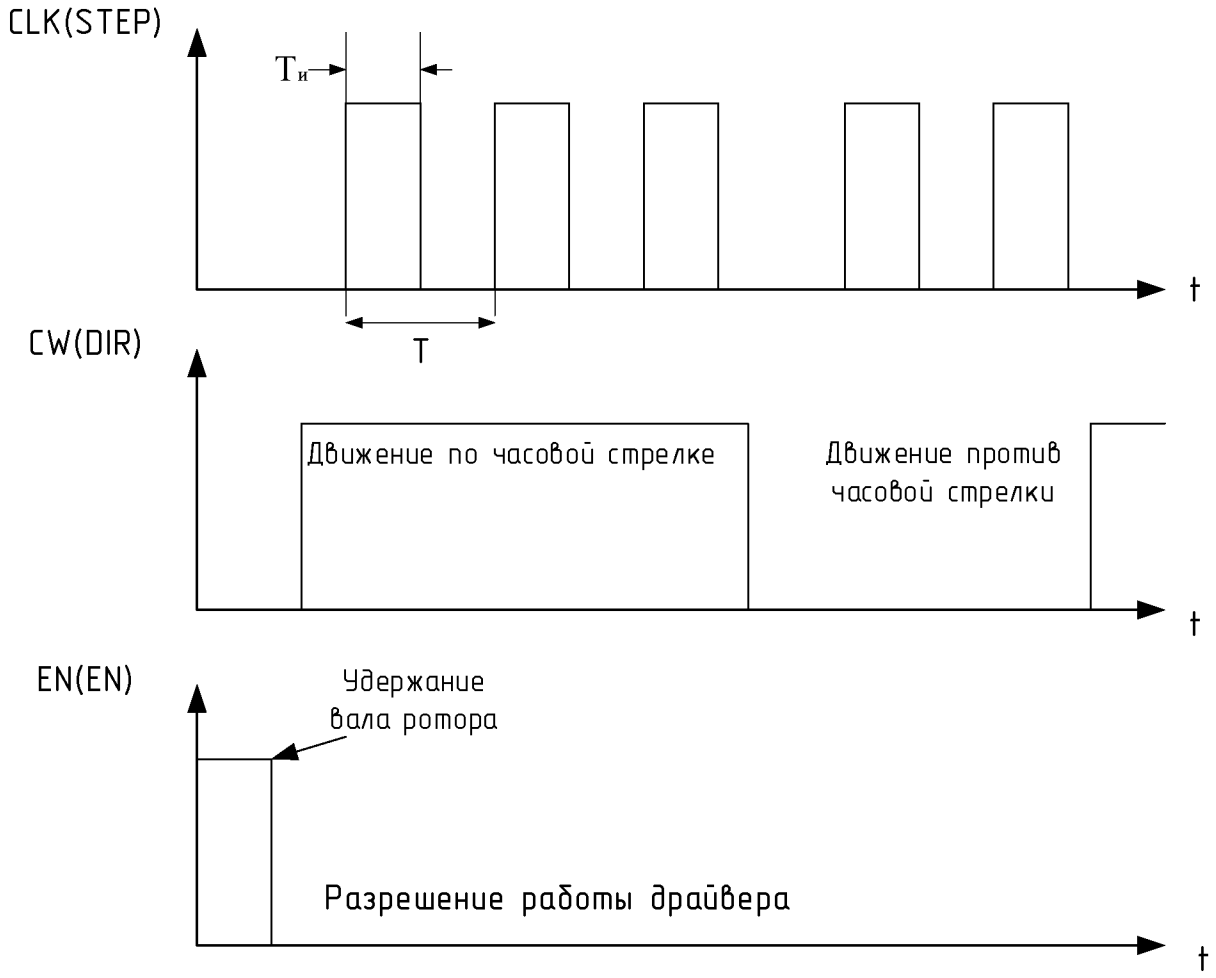


Рисунок 12 - Временная диаграмма протокола STEP/DIR

Схема включает в себя:

- CW(DIR) – сигнал определяющий направление вращения двигателя (при высоком уровне сигнала двигатель вращается по часовой стрелке, при низком уровне – в противоположную сторону);
- CLK(STEP) – шаг, это тактирующий сигнал, один импульс говорит о том, что ротор двигателя надо повернуть на один шаг;
- EN(EN) – сигнал удержания ротора двигателя после совершения шага.
- Ти – длительность импульса сигнала;
- Т – период импульса сигнала.

3.2 Описание методов библиотеки StepDirDriver

Для управления шаговыми двигателями с микроконтроллера была взята библиотека StepDirDriver с интернет ресурса - mypractic.ru Уроки Arduino.[4] Библиотека содержит класс StepDirDriver с следующими public методами:

- функция формирующая управляющие сигналы для коммутации фаз двигателя;
- функция для поворота вала на заданное число шагов;
- функция для формирования сигнала удержания вала двигателя при останове;
- функция установки делителя частоты коммутации фаз двигателя;
- чтение оставшихся шагов.

3.3 Описание методов класса StepDirDriver

1. Функция формирующая управляющие сигналы для коммутации фаз двигателя

В данной функции происходит выработка управляющих сигналов, определяющие коммутацию фаз. Данную функцию необходимо вызывать в параллельном процессе работы МК, т.е. по прерыванию таймера. Частота вызова функции control() вместе с делителем, задаваемым функцией setDivider, определяет скорость вращения двигателя. Программный код представлен на рисунке ниже (см. рис. 13).

```
void  StepDirDriver::control()
 {
   // делитель частоты коммутации
   if ( _steps == 0 )
   {
     // двигатель остановлен
     _dividerCount= 65534;  // сброс делителя частоты
     _prevSteps = 0;
     return;
     }
     // двигатель не остановлен
     _dividerCount++;
     if ( _dividerCount < _divider ) return;
       _dividerCount= 0;
   if ( _prevSteps != 0 )
   {
     if ( _steps > 0 ) _steps--; // вращение против часовой стрелки
     else              _steps++; // вращение по часовой стрелке
   }
   _prevSteps = _steps;
   if ( _steps != 0 )
   {
     // сделать шаг
     digitalWrite(_pinStep, HIGH);
     delayMicroseconds(10);  // 10 мкс
     digitalWrite(_pinStep, LOW);
   }
   else
   {
     if ( _fixStop == 0 ) digitalWrite(_pinEn, HIGH);
   }
 }
```

Рисунок 13 - Метод void control()

2. Функция для поворота вала на заданное число шагов

Данный метод нужен предназначен для передачи числа шагов методу void control(). В данном методе класса происходит анализ пришедшего числа шагов на знак числа, т.е. если мы передали методу отрицательное число шагов, то устанавливается низкий сигнал на выходе DIR, если положительное то высокий сигнал. Программный код представлен на рисунке ниже (см. рис. 14).

```
void  StepDirDriver::step(long steps)
{
    // блокировка
  if ( (steps == 0) && (_fixStop == 0) ) digitalWrite(_pinEn, HIGH);
  else digitalWrite(_pinEn, LOW);
    // направление вращения
  if ( steps < 0 ) digitalWrite(_pinDir, LOW);
  else digitalWrite(_pinDir, HIGH);
    noInterrupts();
  _steps= steps;
  interrupts();
}
```

Рисунок 14 - метод void step(long steps)

3. Функция для формирования сигнала удержания вала двигателя при останове

Данный метод нужен для настройки состояния двигателя при остановке.

- Если fixStop = 1, то при остановке вращения вала на обмотки двигателя подается ток удержания, положение ротора зафиксировано;
- Если fixStop = 0, напряжение с обмоток двигателя при остановке снимается.

В данном методе происходит перегрузка операторов. Код программы представлен на рисунке 15.

4. Функция установки делителя частоты коммутации фаз двигателя

Данный метод задает коэффициент частоты коммутации фаз двигателя, тем самым определяет скорость вращения вала.

В данном методе происходит перегрузка операторов. Код программы представлен на рисунке 15.

```
void  StepDirDriver::setMode(int fixStop)
{
  _fixStop= fixStop;
}

void  StepDirDriver::setDivider(int divider)
{
  _divider= divider;
}
```

Рисунок 15 - Методы void setMode(int fixStop) и void setDivider(int divider)

5. Чтение оставшихся шагов

В данном методе происходит запрет прерываний и чтение переменной _steps, которая считается в методе control(), далее прерывания разрешаются. Код программы представлен на рисунке 16.

```
int StepDirDriver::readSteps()
{
  long stp;
  noInterrupts();
  stp= _steps;
  interrupts();
  return(stp);
}
```

Рисунок 16 - метод readSteps()

В конструкторе класса StepDirDriver, происходит перегрузка номеров выходов, конфигурация выходов и установка начальных значений, установка начальных состояний параметров модификатора private.

4 РАЗРАБОТКА ПРОТОКОЛА СВЯЗИ ПК – МИКРОКОНТРОЛЛЕР

4.1 Описание текстового протокола

Протокол обмена данными – набор правил, соглашений которые определяют обмен данными между различными программами. В проекте используется текстовый протокол на основе AT команд для передачи данных с приложения верхнего уровня (ПК) на микроконтроллер и наоборот.

AT команды представляют собой тестовый протокол обмена данными начиная с символов «AT», в переводе с английского слова (attention) внимание.

Стандартным ответом на команду AT является последовательность символов «OK».

В протоколе обмена данными между ПК и микроконтроллером использованы следующие команды:

- вращения вала двигателя на заданное число шагов;
- настройка режима фиксации вала двигателя при остановке;
- установка скорости вращения двигателя;
- чтение состояний концевых выключателей и датчиков технологической информации.

Для функций протокола Step/Dir, написаны соответствующие AT команды. Каждая команда начинается с символов «AT», далее следует последовательность символов, чисел, параметров. Заканчивать команду принято символами перевода строки «\r\n», где символы «\r» и «\n» возврат в начало и переход на новую строку соответственно. В нашем случае мы будем использовать эти символы в десятеричной форме, где 13 соответствует перемещению курсора в начало строки, а 10 соответствует переходу на новую строку.

Исходя из выше сказанного создан следующий набор AT команд:

Таблица 10 - Набор используемых команд

Команда	Описание
1	2
– ATSX – ATSY – ATSZ – ATSG	Данная команда передает число шагов по осям X, Y, Z и схвата методу step() класса StepDirDriver. Команда представляет последовательную запись символов («ATSX=» , n_steps, 13, 10), где n_steps заданное число шагов.
– ATM	Команда задает режим фиксации вала двигателя при остановке по осям X, Y, Z и схвата методу setMode() класса StepDirDriver. Команда представляет последовательную запись символов («ATM=», 00001111, 13, 10), где нулевой разряд двоичного числа соответствует управляющему сигналу по оси X, первый разряд по оси Y, второй разряд по оси Z, третий для схвата. 1 – фиксация вала двигателя при остановке, 0 – отсутствие фиксации вала двигателя при остановке;

Продолжение таблицы 10

1	2
– ATRX1, ATRX2 – ATRY1, ATRY2 – ATRZ1, ATRZ2 – ATRG1, ATRG2	Команда предназначена для ручного управления осями робота с ПК. Команда представляет последовательную запись символов («ATRX», 1, 13, 10), где «X» – ось X робота, 1 – вращение вала двигателя по часовой стрелке, 2 – вращение вала двигателя против часовой стрелки.
– ATDX – ATDY – ATDZ – ATDG	Данная команда предназначена для настройки частоты переключения фаз двигателей по осям X, Y, Z и схвата. Команда представляет последовательную запись символов («ATDX=» , delitel, 13, 10), где delitel делитель частоты коммутации фаз двигателя.
– ATL?	Команда предназначена для опроса состояния концевых датчиков. Команда представляет последовательную запись символов («ATL?», 13, 10).

4.2 Разработка алгоритма управления осями робота по АТ командам

Для того чтобы реализовать управления осями робота с использованием текстового протокола нужно представить задачу в виде отдельных простых блоков выполняющих полное законченное действие.

Рассмотрим алгоритм работы программы.

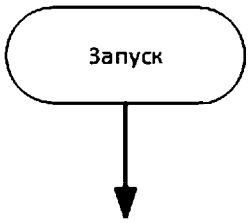


Рисунок 17 - Начало алгоритма

Программа начинается с запуска и переходит к выполнению первых команд.

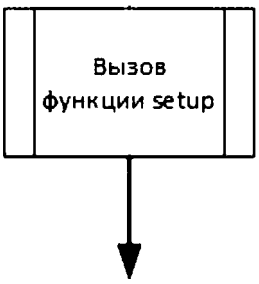


Рисунок 18 - Инициализация программы

Программа начинается с запуска таймера, установки режима фиксации вала двигателя при остановке, установка частоты коммутации фаз двигателя, запуска последовательного порта МК.

Далее следует алгоритм основного цикла программы. В основном цикле проверяется последовательность пришедших на микроконтроллер символов с последовательного порта, в случае совпадения последовательности символов с имеющиеся командой, выполняется алгоритм действий управления шаговыми двигателями соответствующие пришедшей команде. Рассмотрим более детально работу основного цикла программы.

Рисунок 19 - Обработка пришедших данных в порт UART

Работа основного цикла программы начинается с проверки поступления данных на вход UART. Как только данные пришли на порт UART происходит запись этих данных в массив Text, одновременно ведется проверка истечения тайм-аута. Время тайм-аута предоставляет возможность записи символов в массив.

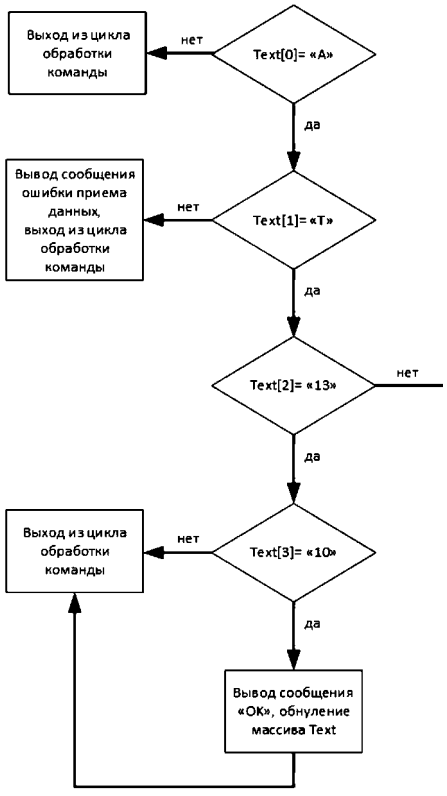


Рисунок 20 - Обработка команды «AT»

В приведенном выше блоке показан алгоритм обработки команды «AT» соответствующей команде для проверки связи между МК и ПК. После того как истекает время тайм-аута начинается обработка элементов массива на совпадение последовательности символов соответствующих загруженным командам. При несовпадении последовательности символов происходит выход из подпрограммы обработки элементов массива Text, при совпадении последовательности символов с заданной командой происходит вывод сообщения «OK» об успешной обработки команды в последовательный порт и обнуление содержимого массива Text.

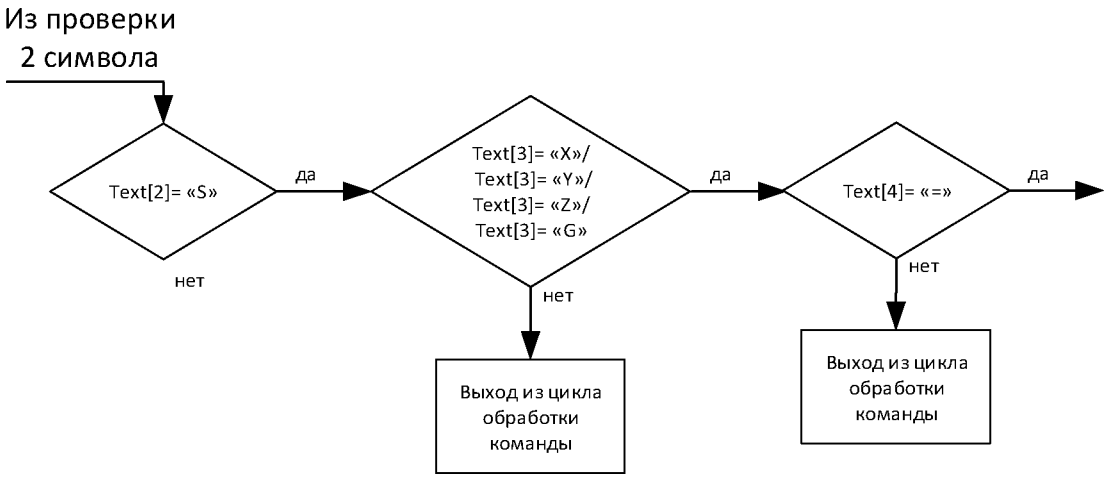


Рисунок 21 - Обработка команды ATSX, ATSY, ATSZ, ATSG

Если второй символ массива не совпадает с кодом символа возврата каретки – «13», происходит проверка элемента массива на совпадение с символом «S». При совпадении идет проверка на совпадении 3 элемента массива на символ «X», «Y», «Z» или «G», при несовпадении не с одним из символов происходит выход из подпрограммы обработки массива Text. После пришедшей последовательности символов должен следовать знак «=», если его нет в пришедшей команде происходит выход из подпрограммы обработки массива Text.

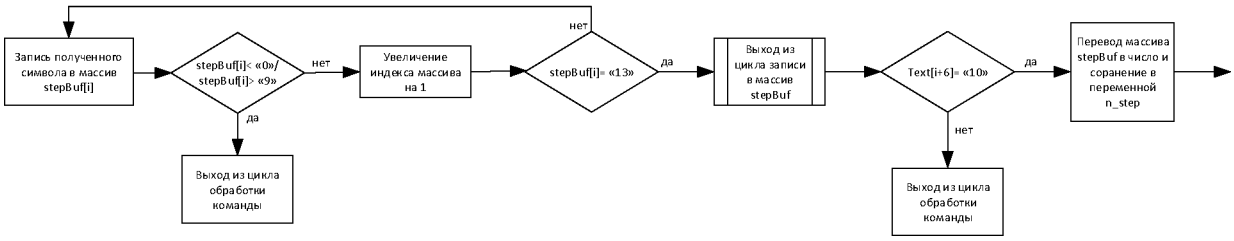


Рисунок 22 - Обработка числа шагов

Далее после того как пришел знак равно следует число шагов. Число шагов представляет собой последовательность цифр и эта последовательность поэлементно записывается в массив stepBuf. Как только приходит код символа «13», запись в массив stepBuf прекращается и далее следует проверки наличия символа перехода на новую строку кодом которого является десятеричный код «10», если он отсутствует, команда пришла неполной и происходит выход из подпрограммы обработки массива Text. Если команда пришла полностью происходит перевод массива stepBuf в число и сохранение в переменной n_step.

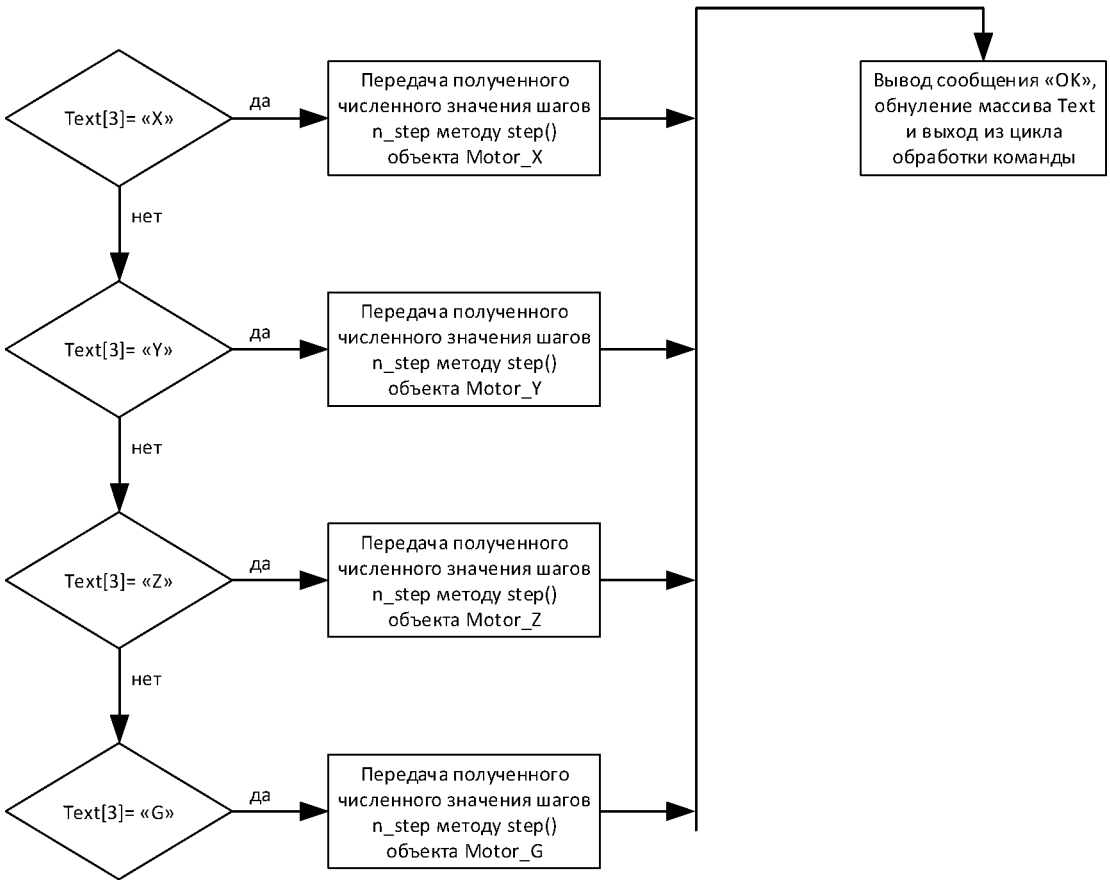


Рисунок 23 - Передача числа шагов методу step()

После удачной обработки числа следует передача числа шагов n_step. Для передачи значения нужному объекту класса проверяется 3 символ массива Text, в зависимости от полученного символа значение n_step записывается в метод step() объектов класса StepDirDriver: Motor_X, Motor_Y, Motor_Z или Motor_G. Тем самым заставляет вращаться вал двигателя. Далее происходит вывод сообщения «OK» об успешной обработки команды в последовательный порт и обнуление содержимого массива Text.

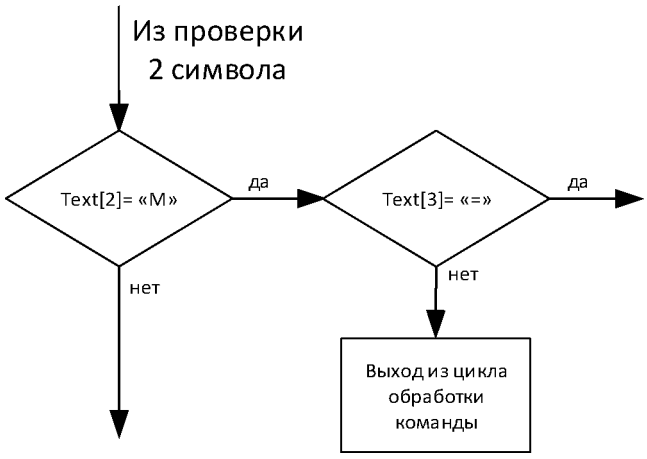


Рисунок 24 - Обработка команды ATM

Если второй символ массива не совпадает с кодом символа «S», происходит проверка элемента массива на совпадение с символом «M». При несовпадении символа ни с одним из вышеуказанных происходит дальше проверка на совпадение с следующим символом. При совпадении идет проверка на совпадении 3 элемента массива на символ «=», если его нет в пришедшей команде происходит выход из подпрограммы обработки массива Text.

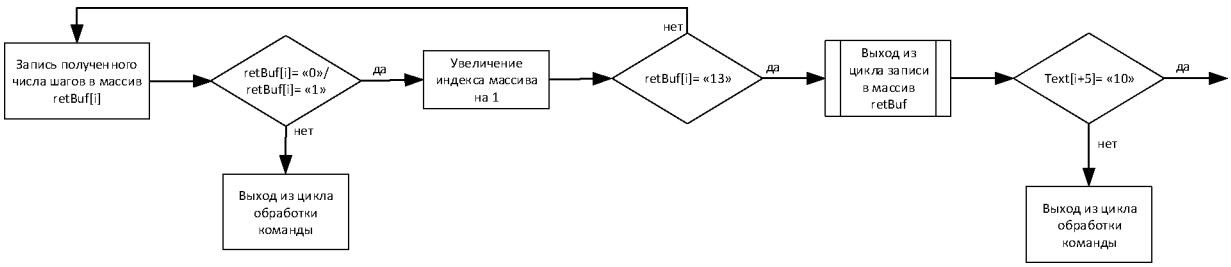


Рисунок 25 – Обработка числа установки режима останова вала двигателя

Далее после того как пришел знак равно следует двоичное число соответствующее настройки режима фиксации валов двигателей по осям робота, соответствие двоичного числа и настроек двигателей на осях X, Y, Z, G приведено на рис.26. Двоичное число представляет собой последовательность цифр и эта последовательность поэлементно записывается в массив retBuf. Как только приходит код символа «13», запись в массив retBuf прекращается и далее следует проверки наличия символа перехода на новую строку кодом которого является десятеричный код «10», если он отсутствует, команда пришла неполной и происходит выход из подпрограммы обработки массива Text.

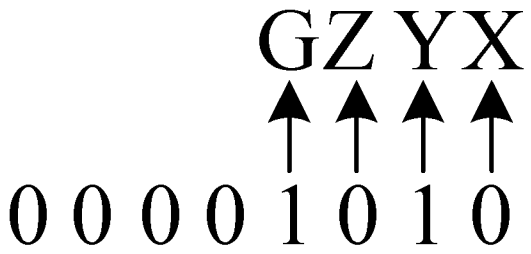


Рисунок 26 - Соответствие двоичного числа осям робота

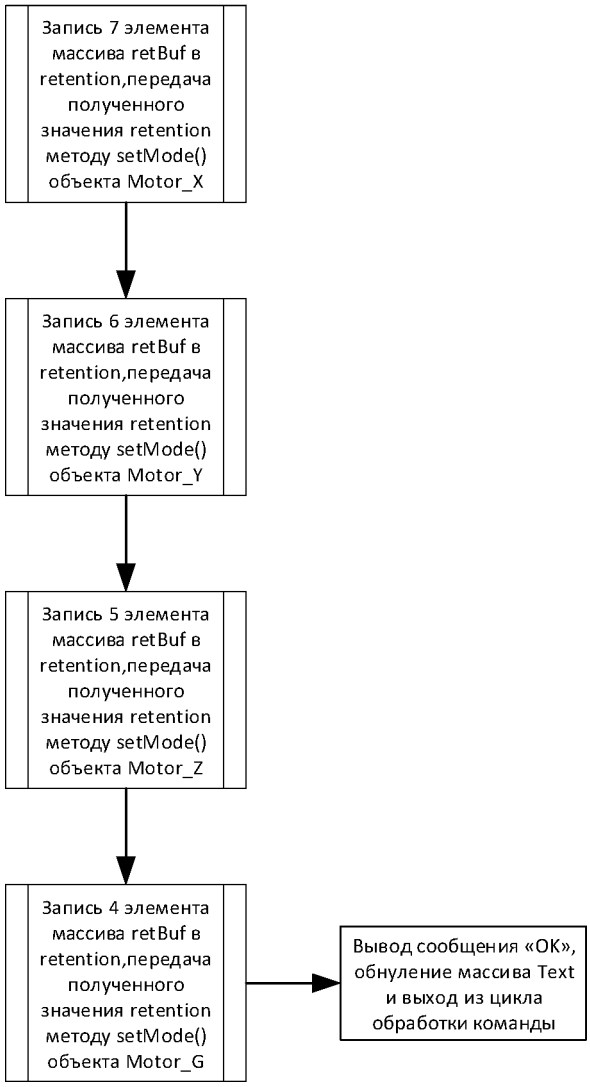


Рисунок 27 - Передача числа шагов методу setMode()

Если команда пришла полностью, происходит передача элементов массива 7-4 в метод setMode() объектов класса StepDirDriver: Motor_X, Motor_Y, Motor_Z и Motor_G. Тем самым формируется управлявший сигнал enable на выходе МК. Далее происходит вывод сообщения «OK» об успешной обработки команды в последовательный порт и обнуление содержимого массива Text.

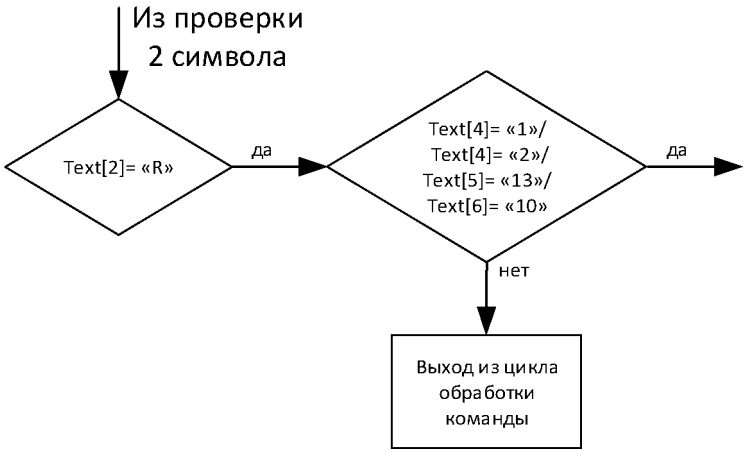


Рисунок 28 - Обработка команды ATR

Если второй символ массива не совпадает с кодом символа «M», происходит проверка элемента массива на совпадение с символом «R». При несовпадении символа ни с одним из вышеуказанных происходит выход из подпрограммы обработки массива Text. При совпадении идет проверка на совпадении 4 элемента массива на символы «1», «2», 5 элемента на символ возврата каретки и 6 элемента на символ перехода на новую строку, если одного из символов нет в пришедшей команде происходит выход из подпрограммы обработки массива Text.

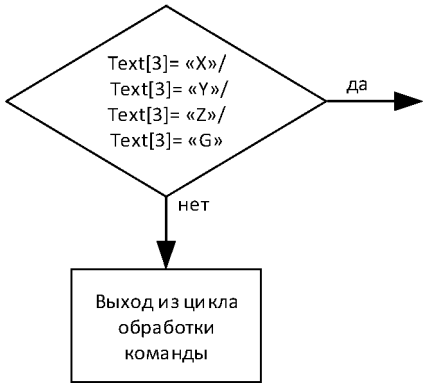


Рисунок 29 - Проверка 4 символа команды на соответствие одной из осей

После того как команда прошла проверку на полноту команды, происходит проверка 3 элемента массива на соответствие символа одной из управляемых осей. В случае неправильно пришедшей команды происходит выход из обработки массива Text, содержащего эту команду.

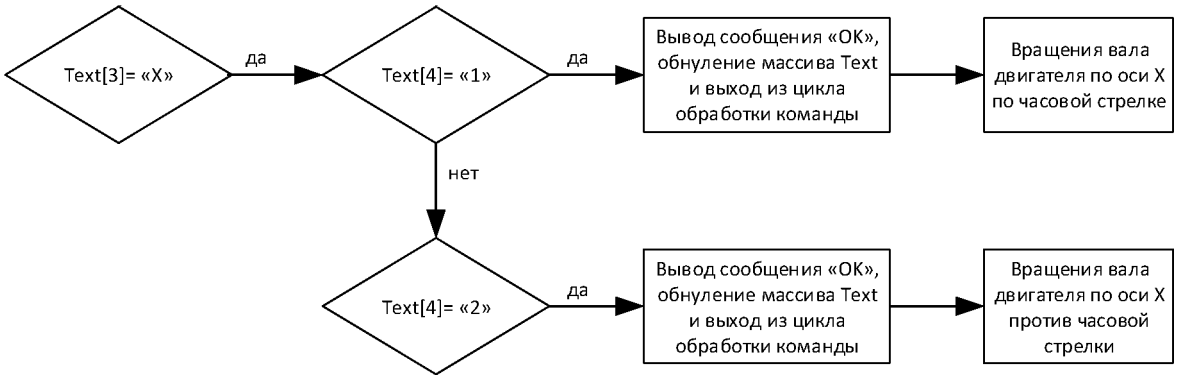


Рисунок 30 – Вращение вала двигателя по оси X

После того как команда была проверена на соответствие одной из набора AT команд (см. табл. 10), происходит управление валом двигателя по оси, направление которой хранится в 3 элементе массива Text, в данном случае X. После проверяется 4 символ массива Text (5 символ команды), если он равен 1 или 2, то происходит вывод сообщения «OK» об успешной обработки команды в последовательный порт и обнуление содержимого массива Text. Далее в зависимости от того какое число содержится в 4 элементе массива Text, происходит вращение вала двигателя по часовой и против часовой стрелки. На рисунке 30 приведет алгоритм обработки команд: ATRX1 и ATRX2, для команд ATRY1, ATRY2, ATRZ1, ATRZ2 алгоритм аналогичен.

Если второй символ массива не совпадает с кодом символа «R», происходит проверка элемента массива на совпадение с символом «D». При совпадении идет проверка на совпадении 3 элемента массива на символы «X», «Y», «Z», «D», и проверка 4 элемента массива на символ «=» если одного из символов нет в пришедшей команде происходит выход из подпрограммы обработки массива Text.

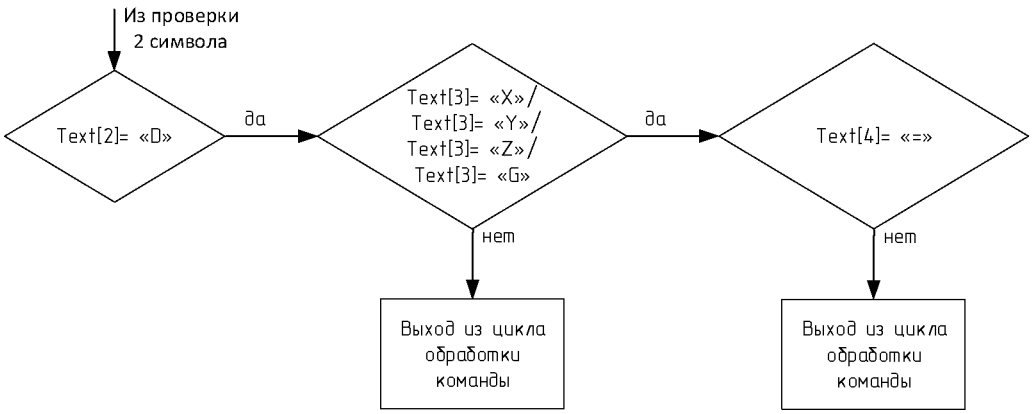


Рисунок 31 - Обработка команды ATD

Далее после того как пришел знак равно следует числовое значение делителя частоты коммутации обмоток двигателя. Число делителя представляет собой последовательность цифр и эта последовательность поэлементно записывается в массив delBuf. Как только приходит код символа «13», запись в массив delBuf прекращается и далее следует проверки наличия символа перехода на новую строку кодом которого является десятеричный код «10», если он отсутствует, команда пришла неполной и происходит выход из подпрограммы обработки массива Text. Если команда пришла полностью происходит перевод массива delBuf в число и сохранение в переменной delitel.

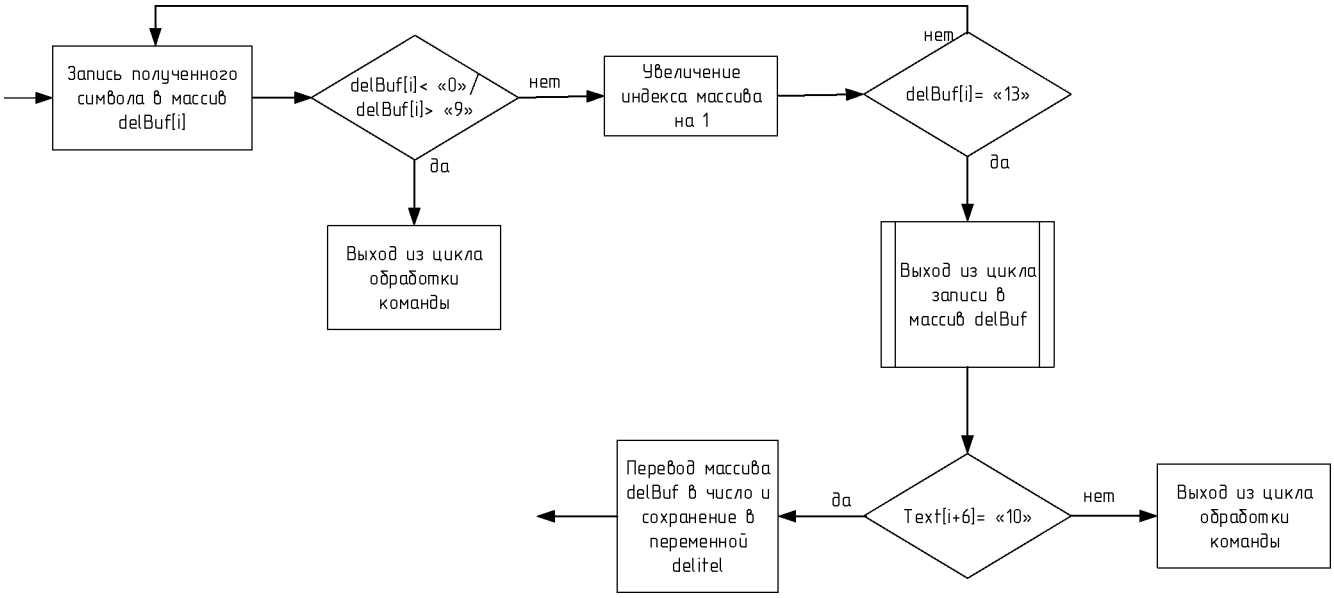


Рисунок 32 - Обработка числа делителя частоты

После удачной обработки числа следует передача числового значения делителя частоты коммутации обмоток двигателя. Для передачи значения

нужному объекту класса проверяется 3 символ массива Text, в зависимости от полученного символа значение n_step записывается в метод setDivider() объектов класса StepDirDriver: Motor_X, Motor_Y, Motor_Z или Motor_G. Тем самым устанавливает период коммутации обмоток двигателя. Далее происходит вывод сообщения «OK» об успешной обработки команды в последовательный порт и обнуление содержимого массива Text.

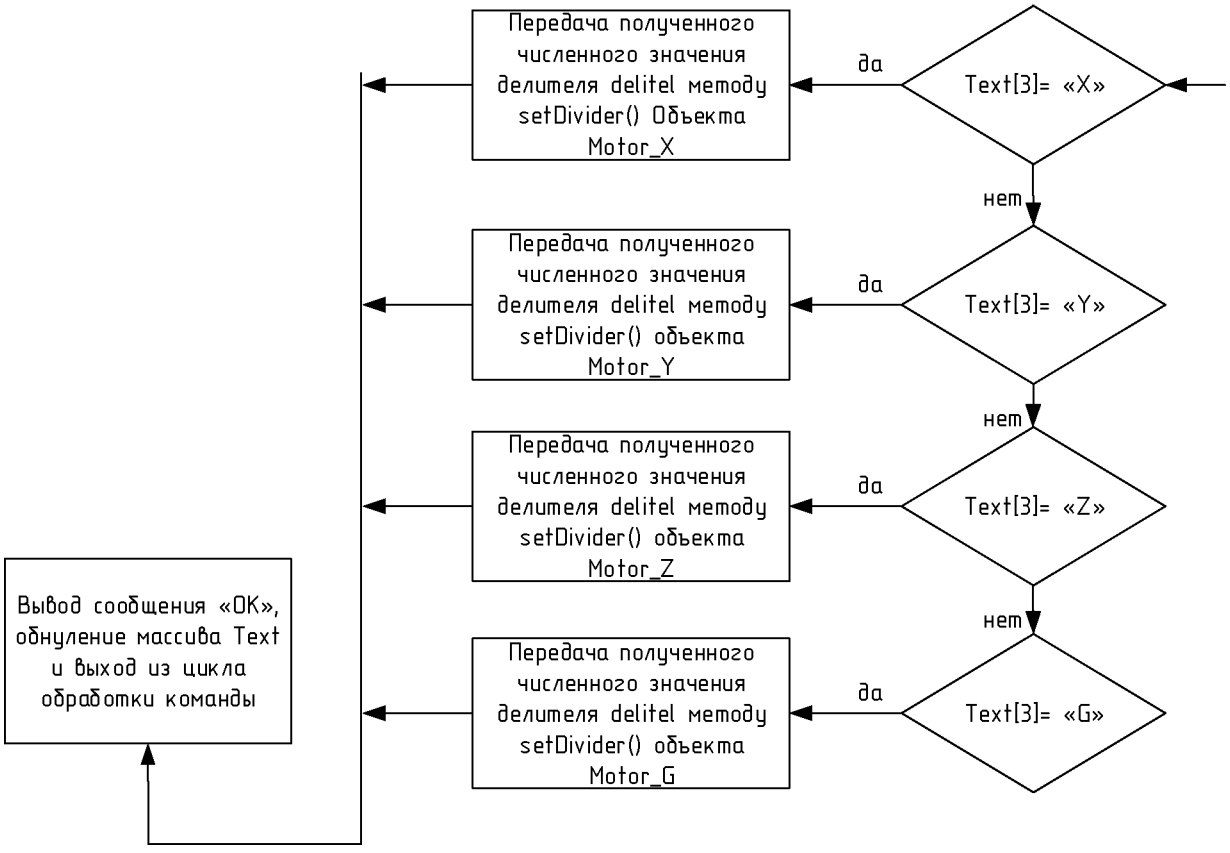


Рисунок 33 - Передача числового значения делителя частоты коммутации обмоток двигателя методу setDivider()

Если второй символ массива не совпадает с кодом символа «D», происходит проверка элемента массива на совпадение с символом «L». При несовпадении символа ни с одним из вышеуказанных происходит выход из подпрограммы обработки массива Text. При совпадении идет проверка на совпадении 4 и 5 элементов массива на символы возврата каретки и на символ перехода на новую строку соответственно, если одного из символов нет в пришедшей команде происходит выход из подпрограммы обработки массива Text.

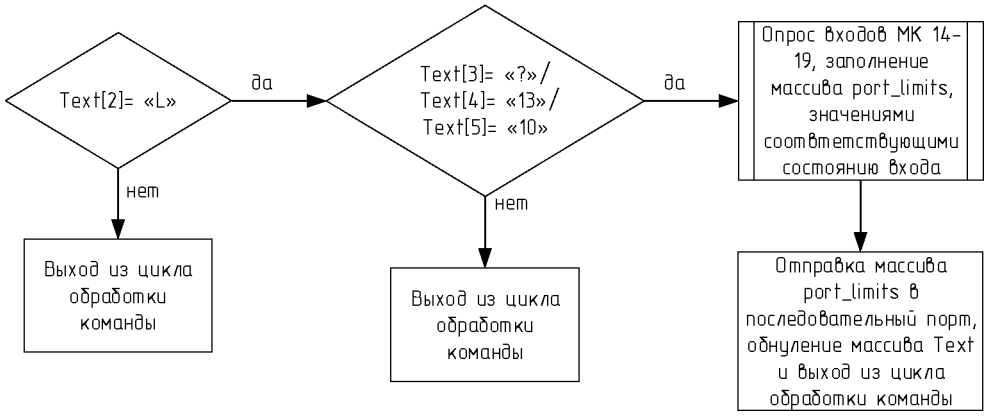


Рисунок 34 - Обработка команды ATL?

После того как команда пришла полностью микроконтроллер опрашивает состояния входов 14-19, к которым подключены концевые датчики. После опроса происходит запись их состояний в массив port_limits следующим образом: если датчик сработал в массив записывается 1, если не сработал – 0. Далее происходит отправка массива port_limits в последовательный порт.

Полный алгоритм программы см. ПРИЛОЖЕНИЕ Г.

4.3 Разработка исходного текста программы

Работа программы (см. рис. 17) начинается с объявления библиотек и переменных.

```
#include <TimerOne.h>
#include <StepDirDriver.h>
int flagX, flagY, flagZ;
char Text[20];      // тестовый буфер
char  stepBuf[10]; // буфер для кол-ва шагов
char retBuf[10];   //буфер для установки удержания ротора
char delBuf[5];   //буфер для установки делителя частоты вращения
char port_limits[5];// буфер для хранения состояния концевых датчиков
int delitel;
int s;  // индекс массива Text[20]
int i; // индекс массива stepBuf[10],retBuf[10]
#define TIME_OUT 200 // время тайм-аута между командами (* 0,25 мс)
#define TIME_OUT_KEYPAD 200 // время тайм-аута между командами (* 0,25 мс)
int  timeOutCount; // счетчик времени между приемом данных
int timeOutCount_but;// счетчик времени обработки матричной клавиатуры
int s3; //код третьего элемента массива Text[20]
int s4; //код четвертого элемента массива Text[20]
long n_step = 0; //число шагов
int retention = 1; //удержание ротора
int state_button = 0;//флаг состояний на интерфейсе пользователя
int st;// номер опрашиваемого столбца
char portState[3]= {0xF7,0xEF,0xDF};
char inputState[3]={0x01,0x02,0x04};
char symbol;//Переменная для хранения символа
int j;// номер опрашиваемой строки
const char key [3][3]=
{{'1','4','7'},
 {'2','5','8'},
 {'3','6','9'}};
StepDirDriver Motor_X(2, 3, 4);//создаем объект типа StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver Motor_Y(5, 6, 7);//создаем объект типа StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver Motor_Z(8, 9, 10);//создаем объект типа StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver Motor_G(11, 12, 13);//создаем объект типа StepDirDriver, задаем выводы для сигналов(step,dir,enable)
```

Рисунок 35 - Запуск программы

При запуске программы объявляются библиотеки TimerOne.h для работы таймера и StepDirDriver.h для формирования управляющих сигналов для драйвера шагового двигателя который отвечает за коммутацию фаз двигателя. Далее происходит объявление массивов, переменных и присвоение им начальных значений. После объявления переменных создаются объекты класса StepDirDriver - Motor_X, Motor_Y, Motor_Z и Motor_G, при этом передаем конструктору номера выводов сигналов step, dir, enable (см. рис. 35).

Далее переходим к подпрограмме инициализации системы (см. рис. 18).

```
36 void setup()
37 {
38   Serial.begin(9600);//Запуск последовательного порта
39   Timer1.initialize(250);  // инициализация таймера 1, период 250 мкс
40   Timer1.attachInterrupt(timerInterrupt, 250);  // задаем обработчик прерываний
41   Motor_X.setMode(retention); // задаем режимы коммутации фаз и остановки
42   Motor_Y.setMode(retention); // задаем режимы коммутации фаз и остановки
43   Motor_Z.setMode(retention); // задаем режимы коммутации фаз и остановки
44   Motor_G.setMode(retention); // задаем режимы коммутации фаз и остановки
45   Motor_X.setDivider(10); // установка делителя частоты для коммутации фаз
46   Motor_Y.setDivider(10); // установка делителя частоты для коммутации фаз
47   Motor_Z.setDivider(10); // установка делителя частоты для коммутации фаз
48   Motor_G.setDivider(10); // установка делителя частоты для коммутации фаз
49   Motor_X.step(0); // инициирует поворот ротора на заданное число шагов
50   Motor_Y.step(0); // инициирует поворот ротора на заданное число шагов
51   Motor_Z.step(0); // инициирует поворот ротора на заданное число шагов
52   Motor_G.step(0); // инициирует поворот ротора на заданное число шагов
53   DDRF|=(0<<0)|(0<<1)|(0<<2)|(1<<3)|(1<<4)|(1<<5);//Конфигурация портов к которым подключена матричная клавиатура
54   PORTF|=(1<<0)|(1<<1)|(1<<2);//Подключение подтягивающих резисторов по строкам клавиатуры
55   pinMode(14, INPUT);//1 Концевой датчик по оси X
56   pinMode(15, INPUT);//2 Концевой датчик по оси X
57   pinMode(16, INPUT);//1 Концевой датчик по оси Y
58   pinMode(17, INPUT);//2 Концевой датчик по оси Y
59   pinMode(18, INPUT);//1 Концевой датчик по оси Z
60   pinMode(19, INPUT);//2 Концевой датчик по оси Z
61 }
62
```

Рисунок 36 - Функция void setup()

После объявления библиотек и переменных, назначаем режим работы системы и выполнение настроечных команд. В функции void setup() происходит запуск работы последовательного порта (UART) и задаем ему скорость обмена данными 9600 бит/сек.. Далее происходит инициализация таймера, которому передается значение 250, соответствующее периоду наступления прерывания по таймеру (в мкс.), задается обработчик прерываний. Устанавливается режим удержания ротора двигателей, задается делитель частоты коммутации фаз и задается начальное значения числа шагов шаговым двигателям. Также происходит конфигурация портов

микроконтроллера, к которым подключены датчики концевых положений на вход.

Далее переходим к выполнению основного цикла программы (см. рис. 19).

```
void loop()
{
  if (Serial.available() == 0)
  {
    timeOutCount = 0;
    s = 0;
  }
    while (Serial.available())
    {
      char sym = Serial.read();
      Text[s] = sym;
      s++;
      delay(16);
    }
```

Рисунок 37 - Запись данных с UART в массив

В начале основного цикла программы проверяется наличие байтов в последовательном порте МК, если порт пуст обнуляется счетчик тайм-аута и индекс массива Text. Если поступили символы и пока они поступают в порт происходит запись символов в массив Text, между поступлениями символов удерживается пауза в 16 мс. Задержка необходима т.к. UART работает медленнее, чем CPU микроконтроллера. Исходя из этого составим временную диаграмму текстового протокола (см. рис. 38).

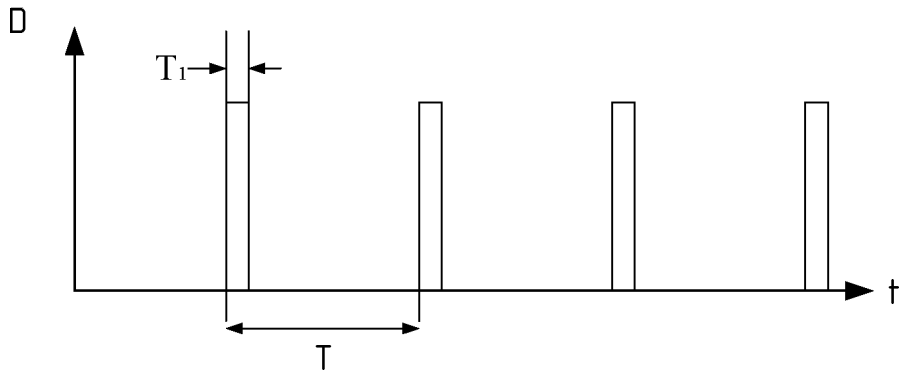


Рисунок 38 - Временная диаграмма текстового протокола

Схема включает в себя:

– D – сигнал определяющий прием байта по uart;

– T_1 – длительность получения байта;

– T – период записи байта в массив Text[].

Переходим к обработке поступившей команды от ПК, первой обрабатывается команда проверки связи между МК и ПК (см. рис. 20).

```
if ( timeOutCount > TIME_OUT )
{
  while (true)
  {
    if ( Text[0] != 'A') break; // ошибка
    if ( Text[1] != 'T')
    {
      Serial.print("Bad\r\n");
      break;
    }
    s3 = Text[2]; // третий символ команды
    if ( s3 == 13 )
    {
      if ( Text[3] != 10 ) break; // ошибка
      Serial.print("OK\r\n");
      for (int q = 0; q < s; q++)//Здесь происходит обнуление массива в котором хранится
      {                          //команда, т.к. следующая команда может интерпретироваться
        Text[q] = 0;             //неправильно из – за оставшегося мусора от предыдущей.
      }
      break;
    }
```

Рисунок 39 - Проверка команды «AT»

Если время тайм-аута истекло и данные не поступают в порт, начинается проверка элементов массива Text на последовательность символов одной из набора AT команд (см. табл.10). Проверка начинается с нулевого элемента, если он не соответствует символу «A», значит пришла некорректная команда и выходит из подпрограммы обработки массива, иначе если совпадает проверяется первый элемент массива на соответствие символу «T», при несоответствии символа выводится в порт сообщение об ошибке «Bad» и выходим из подпрограммы обработки массива. Далее ведется проверка на наличие кода символа возврата каретки «13», при несоответствии выходим из подпрограммы, по аналогии проверяется код символа перехода на новую строку «10». При успешной обработки команды в последовательный порт выводится сообщение «OK» и обнуляется массив Text, содержащий полученную команду.

Если второй элемент массива Text не соответствует коду символа возврата каретки, то проверяется на соответствие команд ATSX, ATSY, ATSZ,

ATSG – передачи числа шагов методу step() класса StepDirDriver для объектов Motor_X, Motor_Y, Motor_Z и Motor_G соответствующих двигателям по оси X, Y, Z, G (см. рис. 40).

```
else if (s3 == 'S')
{
  s4 = Text[3]; // четвертый символ команды
  if ((s4 == 'X')||(s4 == 'Y')||(s4 == 'Z')||(s4 == 'G'))
  {
    if (Text[4] != '=')break;
```

Рисунок 40 - Обработка команды ATSX

В данном фрагменте программы при соответствии третьего символа (второго элемента) массива коду символа «S», проверяется четвертый символ, если символ X, Y, Z, G, значит команда соответствует одной из списка (см. табл.10) и начинается проверка четвертого элемента массива на символ «=», при его отсутствии происходит выход из подпрограммы обработки массива (см. рис. 21).

```
i = 0;
while (i < 8)
{
  stepBuf[i] = Text[i + 5];
  if ( stepBuf[i] == 13 ) break; // конец команды
  if ((stepBuf[i] < '0') || (stepBuf[i] > '9')) break;
  i++;
}
if ( (stepBuf[i] != 13) || (i > 7) ) break; // ошибка
stepBuf[i] = 0;
if ( Text[i + 6] != 10 ) break;
n_step=atoi(stepBuf);
```

Рисунок 41 - Обработка числа шагов

После того как был идентифицирован символ «=», следует запись символов, соответствующих числу шагов, из массива Text, в котором хранится команда в массив stepBuf, где хранится число шагов для ШД. При записи в массив stepBuf ведется проверка на код символа возврата каретки и проверка пришедших символов на цифры. При обнаружении кода символа возврата каретки «13», происходит выход из цикла записи данных в массив stepBuf.

Далее проверяется окончание команды и в переменную n_step записывается массив конвертированный в целочисленный тип int (см. рис. 22).

```
    switch(s4)
    {
      case 'X':
        Motor_X.step(n_step);
        break;
      case 'Y':
        Motor_Y.step(n_step);
        break;
      case 'Z':
        Motor_Z.step(n_step);
        break;
      case 'G':
        Motor_G.step(n_step);
        break;
    }
    Serial.print("OK\r\n");
    for (int q = 0; q < s; q++)
  {
    Text[q] = 0;
  }
    break;
  }
  break;
}
```

Рисунок 42 - Передача числа шагов методу setMode()

В последнем шаге выполнения команды передачи шагов для ШД, выполняется содержание третьего элемента массива, т.е. для двигателя какой оси были переданы число шагов, при совпадении символа n_step передается методу step() одному из объектов библиотеки StepDirDriver. После успешного выполнения команды в последовательный порт выводится сообщение «OK» и обнуляется массив Text, содержащий полученную команду и происходит выход из обработки команды. (см. рис. 23).

Если же второй элемент массива Text соответствует коду символа «M», проверяется третий элемент массива на символ «=». (см. рис. 24).

```
else if (s3 == 'M')
{
    if (Text[3] != '=')break;
    i = 0;
    while (i < 9)
    {
      retBuf[i] = Text[i + 4];
      if ( retBuf[i] == 13 ) break; // конец команды
      if((retBuf[i]!='0')&&(retBuf[i]!='1')) break;
      i++;
    }
    if ( (retBuf[i] != 13) || (i > 8) ) break; // ошибка
```

Рисунок 43 - Обработка числа установки режима останова вала двигателя

После символа «=» следует последовательность символов представляющих двоичное число (см. рис. 26) настроек фиксации валов двигателей на осях робота, которое записывается в массив retBuf, где хранится двоичное число для настройки работы двигателей. При записи в массив retBuf ведется проверка на код символа возврата каретки и проверка пришедших символов на цифры «0» или «1». При обнаружении кода символа возврата каретки «13», происходит выход из цикла записи данных в массив retBuf. Далее проверяется окончание команды (см. рис. 25).

```
      retention=retBuf[7]-48;
      Motor_X.setMode(retention);
      retention=retBuf[6]-48;
      Motor_Y.setMode(retention);
      retention=retBuf[5]-48;
      Motor_Z.setMode(retention);
      retention=retBuf[4]-48;
      Motor_G.setMode(retention);
      Serial.print("OK\r\n");
      for (int q = 0; q < s; q++)
    {
      Text[q] = 0;
    }
      break;
  }
  break;
}
```

Рисунок 44 - Передача числа шагов методу setMode()

В переменную retention записывается число 7 элемента массива, после чего передается методу setMode() объекту Motor_X библиотеки StepDirDriver, по аналогии записываются числа 6, 5 и 4 элементов массива объектам

Motor_Y, Motor_Z, Motor_G (см. рис. 26), в конце в последовательный порт выводится сообщение «OK» и обнуляется массив Text, содержащий полученную команду и происходит выход из обработки команды (см. рис. 26).

Если вторым элементом массива Text является символ соответствующий коду символа «R», последует проверка на содержание 4 элемента массива символов «1» или «2», указывающие на направление вращения вала двигателя, и проверка 5, 6 элементов на символы возврата каретки и возвращения на новую строку. При наличие всех символов, считается что команда пришла полной. Происходит анализ 4 символа команды на совпадение элемента соответствующего одной из осей робота «X», «Y», «Z» или «G». Если 4 символ не является одним из элементов происходит выход из обработки массива Text (см. рис. 28, 29).

```
else if (s3 == 'R')
{
  if ((Text[4] != '1') && (Text[4] != '2') || (Text[5] != 13) || (Text[6] != 10))break;
  if ((Text[3] == 'X') || (Text[3] == 'Y') || (Text[3] == 'Z') || (Text[3] == 'G'))
```

Рисунок 45 - Обработка команды ATR

Следом снова происходит проверка 4 элемента команды, для передачи управления на соответствующую ось. Проверяется 5 элемент команды на символы «1» или «2», для совершения вращения вала двигателя в нужном направлении (см. рис 30). Ниже приведен код для передачи вращения по оси X, для остальных осей алгоритм действий аналогичен. После успешной обработки команды в последовательный порт выводится сообщение «OK» и обнуляется массив Text, содержащий полученную команду и происходит выход из обработки команды

```
if (Text[3] == 'X')
{
  switch (Text[4])
  {
    case '1':
      state_button = 1;
      break;
    case '2':
      state_button = 2;
      break;
  }
}
```

Рисунок 46 - Передача вращения вала двигателя по оси X

Переменная state_button предназначена для хранения номера и направления вращаемого двигателя, state_button=1 – двигатель вращается по часовой стрелке по оси X, 2 - двигатель вращается против часовой стрелки по оси X; 3 – двигатель вращается по часовой стрелке по оси Y, 4 - двигатель вращается против часовой стрелки по оси Y; 5 – двигатель вращается по часовой стрелке по оси Z, 6 - двигатель вращается против часовой стрелки по оси Z.

После обработке команды происходит проверка state_button и если он не равен 0 то происходит вращения одного из двигателей и проверка состояний концевых выключателей до того момента пока в порт UART микроконтроллера не придёт какой - нибудь символ (см. рис. 37).

```
while (state_button == 1)
{
  n_step++;
  Motor_X.step(n_step);
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_X.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(14)==HIGH)
  {
    Serial.print("end\r\n");
    n_step=0;
    Motor_X.step(n_step);
    state_button = 0;
    flagX=1;
  }
  if(flagX==2)
  {
    if(digitalRead(15)==HIGH)
    {
      Serial.print("st!\r\n");
      flagX=0;
    }
  }
}
```

Рисунок 47 - Вращение вала двигателя по оси X

Если вторым элементом массива Text является символ соответствующий коду символа «D», последует проверка на содержание 3 элемента массива символов «X», «Y», «Z» или «G», указывающие на двигатель установленный на соответствующей оси. После чего проверяется 4 элемент массива на символ равно (см. рис. 31).

```
 else if (s3 == 'D')
{
  s4 = Text[3]; // четвертый символ команды
  if ((s4 != 'X') && (s4 != 'Y') && (s4 != 'Z') && (s4 != 'G'))break;
  if(Text[4]!='=') break;
```

Рисунок 48 - Обработка команды ATDX

После того как был идентифицирован символ «=», следует запись символов, соответствующих числу делителя частоты коммутации обмоток двигателя, из массива Text, в котором хранится команда в массив delBuf, где хранится числовое значение делителя частоты. При записи в массив delBuf ведется проверка на код символа возврата каретки и проверка пришедших символов на цифры. При обнаружении кода символа возврата каретки «13», происходит выход из цикла записи данных в массив delBuf. Далее проверяется

окончание команды и в переменную delitel записывается массив конвертированный в целочисленный тип int (см. рис. 32).

```
i=0;
    while(i<5)
  {
    delBuf[i]=Text[i+5];
    if(delBuf[i]==13) break;//Конец команды
    if((delBuf[i]<'0')||(stepBuf[i]>'9')) break;
    i++;
  }
  if((delBuf[i]!=13)||(i>6)) break;//Ошибка
  delBuf[i]=0;
  if(Text[i+6]!=10) break;
  delitel=atoi(delBuf);
```

Рисунок 49 - Обработка делителя частоты

В последнем шаге выполнения команды передачи делителя частоты, выполняется проверка содержание третьего элемента массива, т.е. для двигателя какой оси были переданы число шагов, при совпадении символа, delitel передается методу stetDivider() одному из объектов библиотеки StepDirDriver. После успешного выполнения команды в последовательный порт выводится сообщение «OK» и обнуляется массив Text, содержащий полученную команду и происходит выход из обработки команды. (см. рис. 33).

```
    switch (s4)
{
  case 'X':
    Motor_X.setDivider(delitel);
    break;
  case 'Y':
    Motor_Y.setDivider(delitel);
    break;
  case 'Z':
    Motor_Z.setDivider(delitel);
    break;
  case 'G':
    Motor_G.setDivider(delitel);
    break;
}
Serial.print("OK\r\n");
for (int q = 0; q < s; q++)
{
  Text[q] = 0;
}
break;
```

Рисунок 50 - Передача делителя частоты методу setDivider()

Если второй элемент массива Text соответствует коду символа «L», проверяется третий элемент массива на символ «?». При несовпадении второго

элемента ни с одним из символов команд представленных в списке (см. табл.10), происходит выход из обработки массива. Аналогично происходит при несовпадении третьего элемента массива с символом «?» (см. рис. 34).

```
else if (s3 == 'L')
{
 if ((Text[3] != '?')|| (Text[4] != 13) || (Text[5] != 10))break;
 if (digitalRead(14)==HIGH)port_limits[0]='1';
      else if (digitalRead(15)==HIGH)port_limits[1]='1';
      else if (digitalRead(16)==HIGH)port_limits[2]='1';
      else if (digitalRead(17)==HIGH)port_limits[3]='1';
      else if (digitalRead(18)==HIGH)port_limits[4]='1';
      else if (digitalRead(19)==HIGH)port_limits[5]='1';
      Serial.print(port_limits);
      for (int q = 0; q < s; q++)
     {
       Text[q] = 0;
     }
     break;
```

Рисунок 51 - Обработка команды ATL?

После успешного распознавания данной команды проверяются выходы микроконтроллера к которым подключены концевые датчики и заполняется массив port_limits.

5 РАЗРАБОТКА РУЧНОГО УПРАВЛЕНИЯ ШД С ПОМОЩЬЮ МАТРИЧНОЙ КЛАВИАТУРЫ

Для управления роботом в ручном режиме будем использовать матричную клавиатуру 3x3, это количество кнопок достаточно для управления всеми осями робота и схватом. Подключение матричной клавиатуры к Arduino приведено в приложении В.

Для управления матричной клавиатуры объявим следующие переменные: TIME_OUT_KEYPAD – время тайм – аута проверки состойний кнопок матричной клавиатуры; timeOutCount_but – счетчик времени, увеличивающийся в прерываниях по таймеру, по достижению переменной timeOutCount_but значения TIME_OUT_KEYPAD происходит проверка состояний кнопок клавиатуры; массив key хранящий номера кнопок; переменная symbol для хранения номера нажатой кнопки; переменные st и j хранящие номера опрашиваемых столбца и строки соответственно. Код программы приведен на рисунке 52.

```
#define TIME_OUT_KEYPAD 200 // время тайм-аута между командами (* 0,25 мс)
int timeOutCount_but;// счетчик времени обработки матричной клавиатуры
int st;// номер опрашиваемого столбца
char portState[3]= {0xF7,0xEF,0xDF};
char inputState[3]={0x01,0x02,0x04};
char symbol;//Переменная для хранения символа
int j;// номер опрашиваемой строки
const char key [3][3]=
{{'1','4','7'},
 {'2','5','8'},
 {'3','6','9'}};
```

Рисунок 52 - Объявление переменных для работы матричной клавиатуры

Матричную клавиатуру подключаем к аналоговым выходам Arduino, аналоговые выходы соответствую цифровым выходам микроконтроллера Atmega2560. Будем работать с ними как с цифровыми с использованием языка С для AVR. На рисунке 53 представлено соответствие выходов Arduino и Atmega2560.

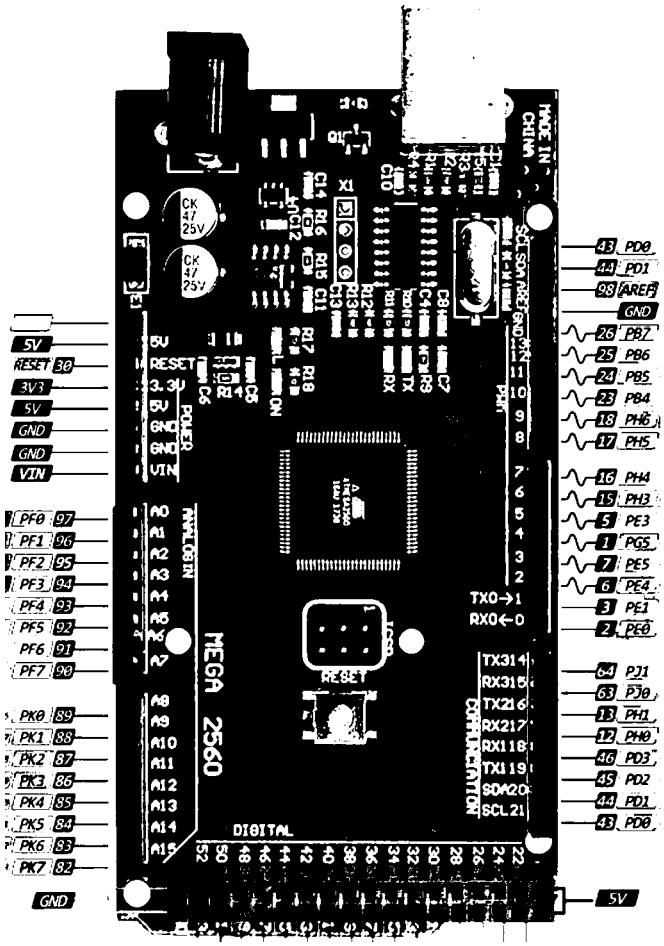


Рисунок 53 - Распиновка Arduino mega 2560

При конфигурации портов настраиваем выходы к которым подключены столбцы клавиатуры на выход, строки на вход с подтяжкой внутренних резисторов. Код программы приведен на рисунке 54.

```
DDRF|=(0<<0)|(0<<1)|(0<<2)|(1<<3)|(1<<4)|(1<<5);,
PORTF|=(1<<0)|(1<<1)|(1<<2);//Подключение подтяг
```

Рисунок 54 - Конфигурация порта С на МК

В основной программе проверяется не достигло ли значение переменной timeOutCount_but значения 200, соответствующее 50 мс. При достижении этого значения начинается сканирование клавиатуры, по столбцам подаются нули. При подаче нуля на первый столбец проверяются выходы МК, к которым подключены строки на низкий уровень сигнала. При нахождении входа в состоянии 0 выбирается символ из массива key с элементами соответствующими проверяемому столбцу и строки на которой был

обнаружен низкий сигнал, далее символ, т.е. номер кнопки сохраняется в переменной symbol. В конце проверки клавиатуры выставляется высокий сигнал на выходе соответствующему проверяемому столбцу и обнуление счетчика времени [5]. Код программы приведен на рисунке 55.

```
if (timeOutCount_but > TIME_OUT_KEYPAD)
{
  symbol=0;
  for(st=0; st<3; st++)
  {
    PORTF=portState[st];
    for(j=0; j<3; j++)
    {
      if(((PINF&inputState[j])==0))
      {
        symbol=key [st][j];
      }
    }
    PORTF|=(1<<st);
    timeOutCount_but=0;
  }
}
```

Рисунок 55 - Проверка состояний кнопок матричной клавиатуры

После обнаружения нажатой кнопки ведется проверка, какая клавиша нажата при нахождении нажатой кнопки устанавливается соответствующий сигнал DIR и вызывается функция коммутирующая фазы двигателя, ниже (см. рис. 48) представлен код для вращения оси X при нажатии на кнопку 4 влево, 6 вправо.

```
if(symbol=='4')
{
      digitalWrite(3,1);
      Step_X();
}
if(symbol=='6')
{
      digitalWrite(3,0);
      Step_X();
}
```

Рисунок 56 - Управление осью X

Принципиальную схему устройства управления звеньями робота см. ПРИЛОЖЕНИЕ В. Полный листинг программы на МК см. ПРИЛОЖЕНИЕ И. Алгоритм опроса матричной клавиатуры см. ПРИЛОЖЕНИЕ Д.

6 РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ НА ПК

Разработка интерфейса пользователя производится в программной среде Microsoft Visual Studio на языке C# windows forms. Microsoft Visual Studio - линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Window Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.[6]

Для работы с микроконтроллером через последовательный порт следует использовать ресурсы класса SerialPort, который в свою очередь предоставляет средства для синхронного и управляемого событиями ввода – вывода, для доступа к состоянию подключения – отключения устройства, а также для доступа к свойствам драйвера последовательного порта. Для использования объектов класса SerialPort в шапке программы пропишем строку `using System.IO.Ports;` Пространство имен System.IO.Ports содержит классы для управления последовательными портами [7].

6.1 Создание стартового окна

Визуальный вид стартового окна программы представлен на рисунке 57.

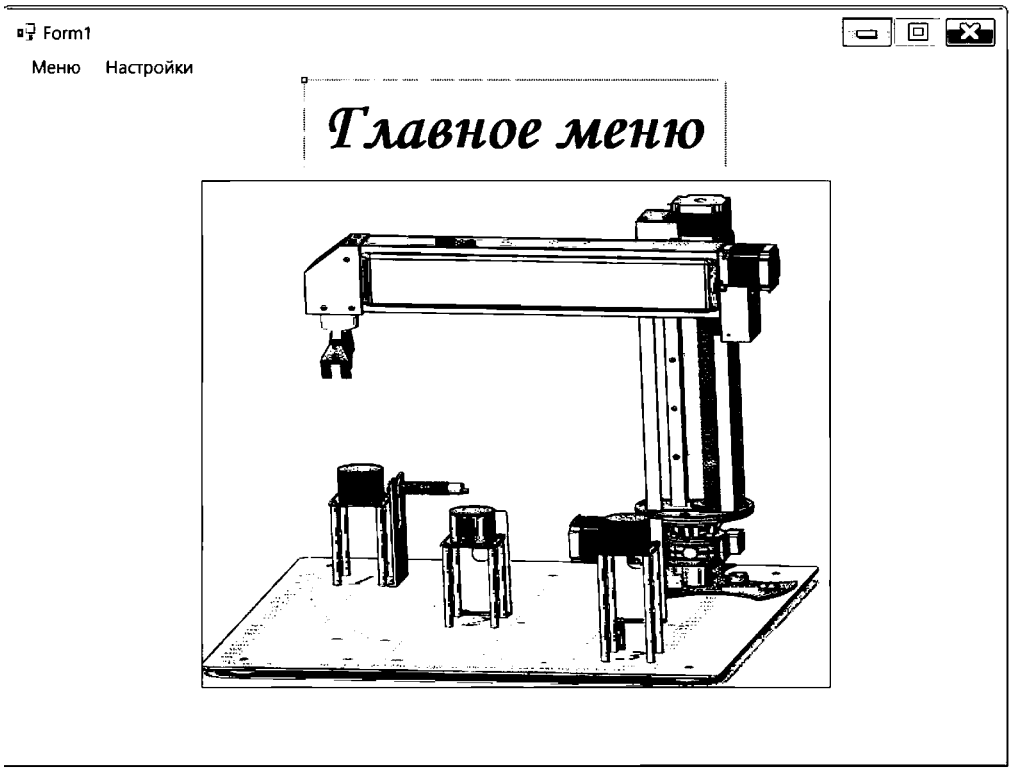


Рисунок 57 - Стартовое окно программы управления роботом

С использованием компонентов «Label», «MenuStrip» и «rectangleShape» составим визуальное представление стартового окна приложения(см. рис. 57), в компоненте «Label» пропишем текст «Главное меню», компонент «rectangleShape» растянем по центу экрана и в вкладке - фоновое изображение вставим изображение робота – манипулятора. В компоненте «MenuStrip» сделаем две вкладки: «Меню», содержащую четыре подпункта для выбора режимов управления: ручной, дискретный, позиционный, автоматический; «Настройки», для настройки работы двигателей. На рисунке 58 показаны вкладки «MenuStrip» [8].

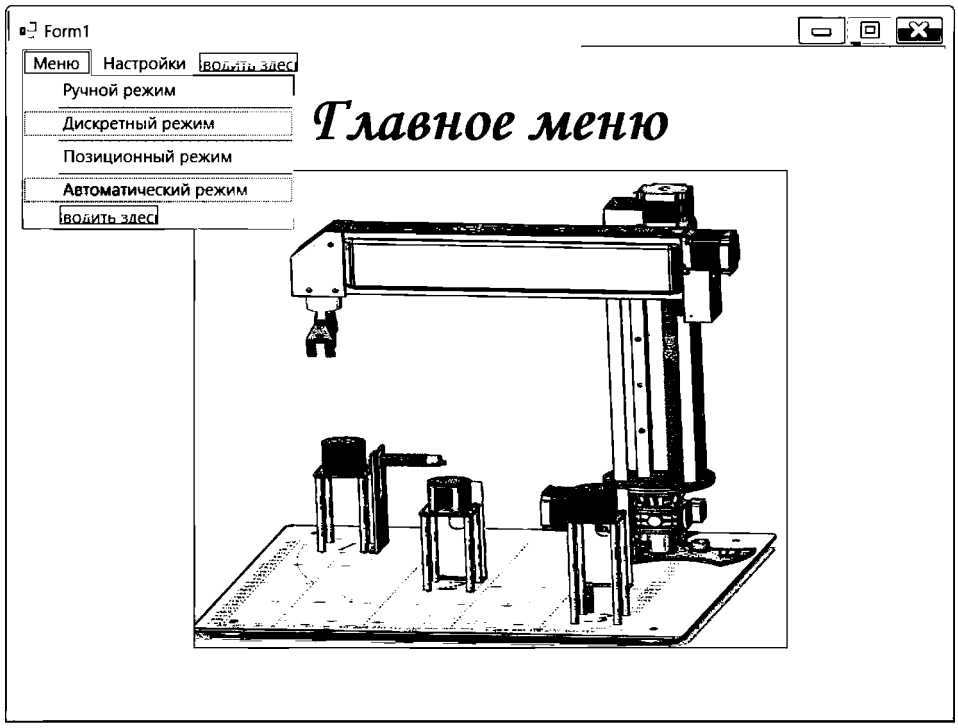


Рисунок 58 - Вкладки меню

После того как стартовое окно создано можно переходить к визуальному созданию и последующему программированию режимов управления роботом.

6.2 Создание окна ручного режима

Для управления роботом в ручном режиме пользователю представляется окно (см. рис. 59), в этом окне изображен объект управления, рядом с стрелками, иллюстрирующими направления движения робота, расположены буквы, указывающие название осей: X, Y, Z. Также рядом с буквами расположены кнопки с надписью «>>»или «<<», при нажатии на них звенья робота осуществляют движение в указанном направлении. Рядом с стрелками находтся индикаторы конечных выключателей, которые отображены закрашенными кружками. Если круг зеленного цвета, то конечный выключатель по этой оси не сработал и движение в данную сторону возможно, если же цвет красный, то выключатель сработал и движение запрещено.

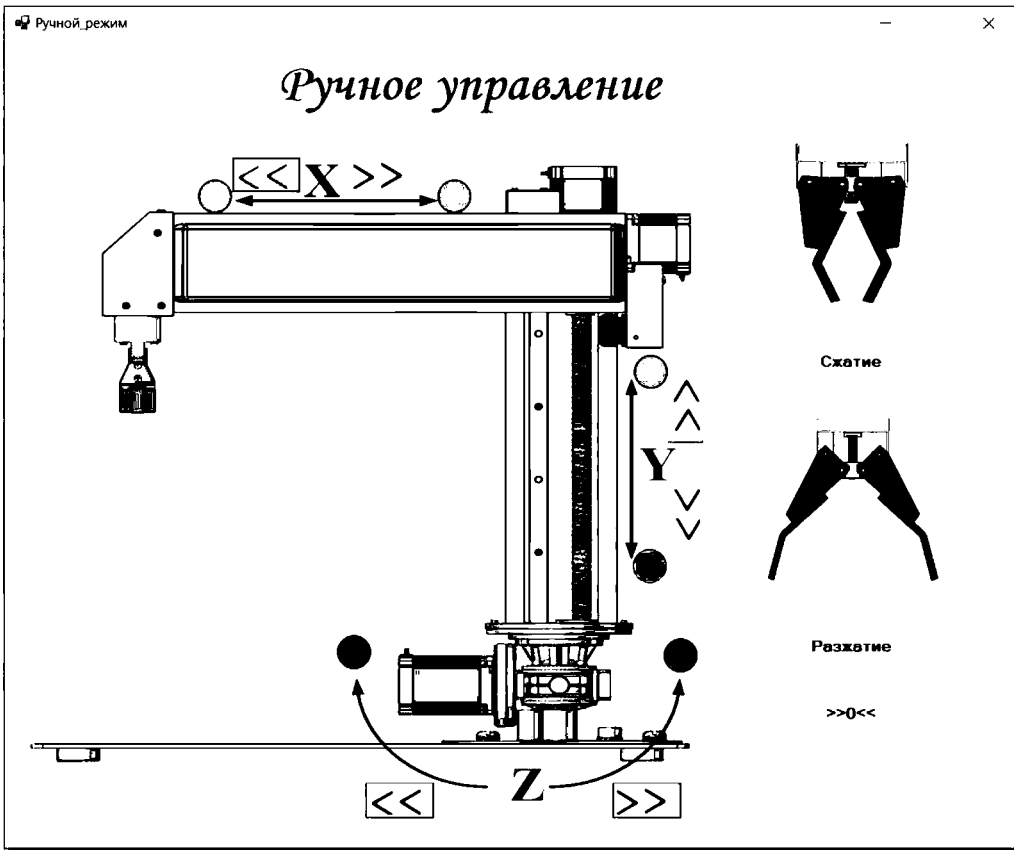


Рисунок 59 - Окно ручного управления

При запуске окна в микроконтроллер отправляется команда «ATL?\r\n» для опроса состояний концевых датчиков робота, в ответ от микроконтроллера приходит двоичное число. За каждым разрядом двоичного числа закреплен датчик, 0 – датчик не сработал, 1 датчик сработал, также происходит индикация указателей конечных выключателей, изображенных в виде кружков, в зависимости от состояния датчика. Ниже приведен код начального запуска окна.

```
private void Ручной_режим_Load(object sender, EventArgs e)
{
    shapeContainer1.BringToFront();
    port.Write("ATL?\r\n");
    Thread.Sleep(100);
    String answer = port.ReadExisting();
    if (answer == "")
    {
        port.Write("ATL?\r\n");
        Thread.Sleep(100);
        answer = port.ReadExisting();
        if (answer == "")
        {
            MessageBox.Show("Обрыв связи!");
            return;
        }
    }
    if (answer != "")
    {
        char[] limits = answer.ToCharArray();
        if (limits[0] == '1') ovalShape1.FillColor = Color.Red;
        else ovalShape1.FillColor = Color.Lime;
        if (limits[1] == '1') ovalShape3.FillColor = Color.Red;
        else ovalShape3.FillColor = Color.Lime;
        if (limits[2] == '1') ovalShape2.FillColor = Color.Red;
        else ovalShape2.FillColor = Color.Lime;
        if (limits[3] == '1') ovalShape4.FillColor = Color.Red;
        else ovalShape4.FillColor = Color.Lime;
        if (limits[4] == '1') ovalShape6.FillColor = Color.Red;
        else ovalShape6.FillColor = Color.Lime;
        if (limits[5] == '1') ovalShape5.FillColor = Color.Red;
        else ovalShape5.FillColor = Color.Lime;
    }
}
```

Рисунок 60 - Запуск окна ручного управления

При нажатии на одну из кнопок, с надписью «>>»или «<<», происходит отправка соответствующих команд текстового протокола в микроконтроллер. При нажатии на кнопку событие button_MouseDown, в котором отправляется команда и ожидается ответ подтверждения выполнения команды, если ответ не пришел команда посылается второй раз, при отсутствии ответа во второй раз выводится сообщение «Обрыв связи!». Если приходит ответ от микроконтроллера запускается таймер, который через короткие промежутки времени опрашивает порт и проверяет не сработал ли конечный выключатель оси, которой в данный момент идет управление. Если выключатель сработал, то движение прекращается, и происходит индикация их состояний (см. рис. 61). При отпускании кнопки происходит отправка символа «1», для остановки движения оси робота и остановка работы таймера. Ниже приведен код программы для управления движением оси X, для осей Y, Z и схвата программа аналогична (см. рис. 62).

```
private void timer1_Tick(object sender, EventArgs e)
{
    switch (port.ReadExisting())
    {
        case "st\r\n":
            ovalShape1.FillColor = Color.Red;
            break;
        case "end\r\n":
            ovalShape2.FillColor = Color.Red;
            break;
        case "st!\r\n":
            ovalShape1.FillColor = Color.Lime;
            break;
        case "end!\r\n":
            ovalShape2.FillColor = Color.Lime;
            break;
        case "up\r\n":
            ovalShape3.FillColor = Color.Red;
            break;
        case "down\r\n":
            ovalShape4.FillColor = Color.Red;
            break;
        case "up!\r\n":
            ovalShape3.FillColor = Color.Lime;
            break;
        case "down!\r\n":
            ovalShape4.FillColor = Color.Lime;
            break;
        case "left\r\n":
            ovalShape5.FillColor = Color.Red;
            break;
        case "right\r\n":
            ovalShape6.FillColor = Color.Red;
            break;
        case "left!\r\n":
            ovalShape5.FillColor = Color.Lime;
            break;
        case "right!\r\n":
            ovalShape6.FillColor = Color.Lime;
            break;
    }
```

Рисунок 61 - Обработчик таймера

```
private void button1_MouseDown(object sender, MouseEventArgs e)
{
    port.Write("ATRX2\r\n");
    Thread.Sleep(50);
    String answer = port.ReadExisting();
    if (answer != "OK\r\n")
    {
        port.Write("ATRX2\r\n");
        Thread.Sleep(50);
        answer = port.ReadExisting();
        if (answer != "OK\r\n")
        {
            MessageBox.Show("Обрыв связи!");
        }
    }
    timer1.Start();
}
private void button1_Click(object sender, EventArgs e)
{
    port.Write("1");
    timer1.Stop();
}
private void button3_MouseDown(object sender, MouseEventArgs e)
{
    port.Write("ATRX1\r\n");
    Thread.Sleep(50);
    String answer = port.ReadExisting();
    if (answer != "OK\r\n")
    {
        port.Write("ATRX1\r\n");
        Thread.Sleep(50);
        answer = port.ReadExisting();
        if (answer != "OK\r\n")
        {
            MessageBox.Show("Обрыв связи!");
        }
    }
    timer1.Start();
}
private void button3_Click(object sender, EventArgs e)
{
    port.Write("1");
    timer1.Stop();
}
```

Рисунок 62 - Обработчик нажатия кнопок «>>», «<<» по оси X

6.3 Создание окна дискретного режима

Для управления роботом – манипулятором в дискретном режиме «от точки к точке», пользователю предоставляется окно в котором отображаются текущее положение робота, заданное положение и кнопки управления: «Задать», «>>0<<», «Сброс», «Остановить». Визуальный вид окна представлен на рисунке 63. При нажатии на кнопку «Задать» происходит извлечение числа из элемента textbox, извлеченное число проверяется на допустимы предел, который указан в окне рядом с полем для ввода значения у каждой оси, затем если значение допустимое идет обработка текущего

значения с заданным для перемещения звена в соответствующую сторону и умножается на коэффициент, для перевода из заданных расстояний перемещения осей, угла поворота, в количество импульсов, которые подаются на шаговый для вращения вала. Также после отправки команды ожидается ответ от микроконтроллера, при отсутствии ответа выводится сообщение «Обрыв связи!», после успешного выполнения команд в поле с текущими значениями переносятся из заданных значений. Ниже приведен код программы обработки заданного значения по оси X (см. рис. 64), по остальным осям робота и схвата обработка аналогична.

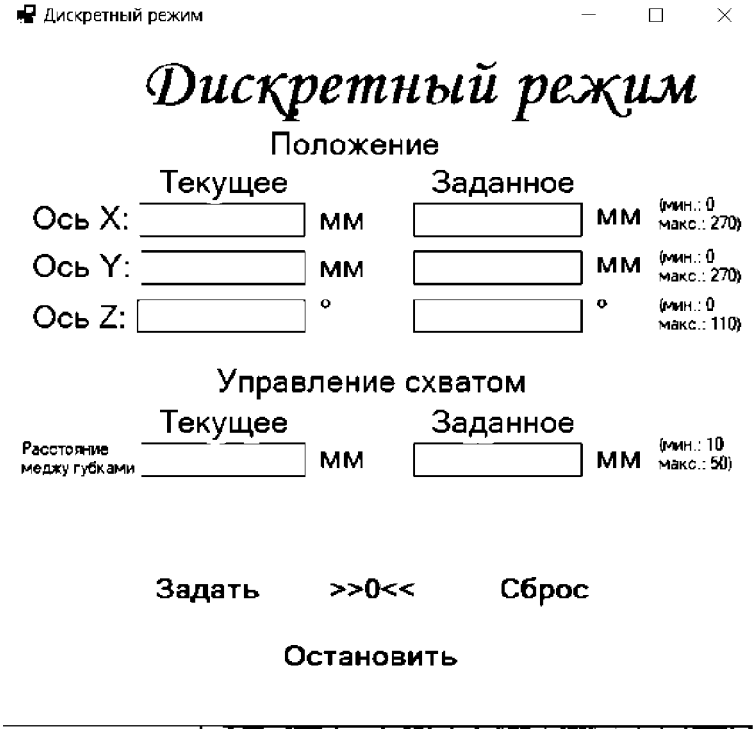


Рисунок 63 - Окно дискретного режима

```
/*-------ось X-------------------------*/
int K_X=Convert.ToInt32(textBox2.Text);//Коэффициент перевод мм в число шагов
 n_step_X= Convert.ToInt32(textBox8.Text) * K_X;//число шагов
if (Convert.ToInt32(textBox8.Text) >= 0 && Convert.ToInt32(textBox8.Text) <= 270)
{
   if ((Convert.ToInt32(textBox8.Text) >=Convert.ToInt32(textBox1.Text)))
    {
        n_step_X = Convert.ToInt32(textBox8.Text) - Convert.ToInt32(textBox1.Text);
        n_step_X=n_step_X* K_X;
    }
   else if ((Convert.ToInt32(textBox8.Text) < Convert.ToInt32(textBox1.Text)))
   {
       n_step_X = Convert.ToInt32(textBox1.Text) - Convert.ToInt32(textBox8.Text);
       n_step_X = -n_step_X * K_X;
   }
    String comand_X = "ATSX=" + Convert.ToString(n_step_X)+"\r"+"\n";
    port.Write(comand_X);
    Thread.Sleep(200);
    String answer = port.ReadExisting();
    if (answer != "OK\r\n")
    {
        port.Write(comand_X);
        Thread.Sleep(200);
        answer = port.ReadExisting();
        if (answer != "OK\r\n")
        {
            MessageBox.Show("Обрыв связи!");
            return;
        }
}
    textBox1.Text = textBox8.Text;
}
else
{
    MessageBox.Show("Неправильно заданное значение по оси X");
}
```

Рисунок 64 - Обработка заданного значения по оси X

При нажатии на кнопку «Сброс» происходит сброс заданных значений. Нажимая кнопку «Остановить» отправляются команды передачи нулевого числа шагов для остановки работы шаговых двигателей.

6.4 Создание окна настроек

Окно настроек предназначено для установки режима работы шагового двигателя, установка фиксации вала двигателя, настройки скорости вращения вала по всем осям робота. Визуальный вид окна представлен на рисунке 65.

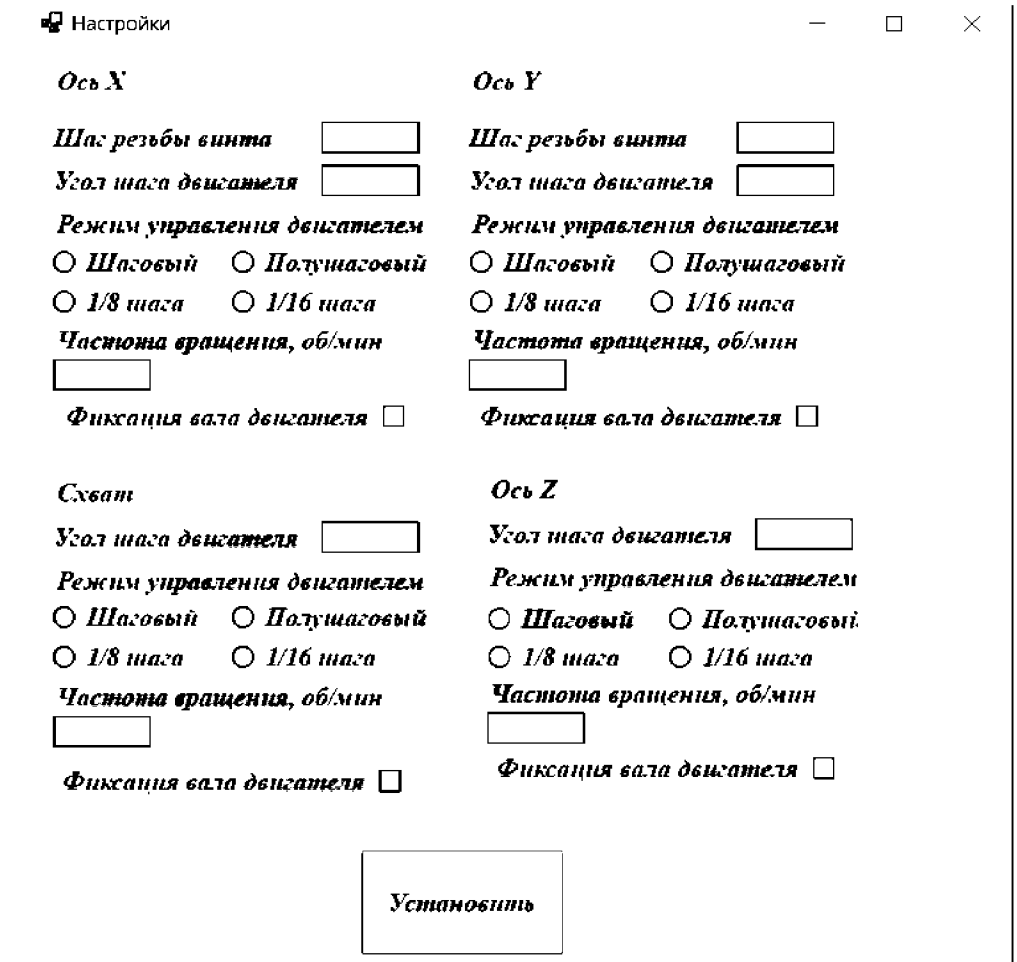


Рисунок 65 - Окно настроек

После ввода всех необходимых настроек происходит проверка на заполнение всех необходимых полей настройки, при пропуске какого-нибудь поля выходит сообщение предупреждения (см. рис. 66).

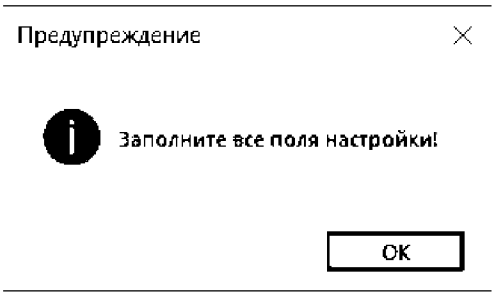


Рисунок 66 - Сообщение предупреждения

Если же все обязательные поля заполнены начинается обработка данных с полей. В переменные h_X, phi_X, rotation_frequency_X и step_X записываются данные с полей шаг резьбы винта, угол шага двигателя, частота вращения и режим управления двигателем соответственно. Далее идет расчет

коэффициента для перевода из заданного расстояния перемещения осей и угла поворота в количество импульсов для шагового двигателя, который используется в дискретном управлении. Также в соответствии с введенным значением скорости вращения вала двигателя и установленными настройками шагового двигателя происходит расчет делителя частоты коммутации фаз двигателя. И отправляется команда «ATDX=delitel_X\r\n» в микроконтроллер, по аналогии происходит управление по остальным осям заменяя 4 символ команды на название соответствующей оси: Y, Z, G. Далее считываются состояние элементов checkbox у каждой оси с помощью которых указываются настройки удержания вала двигателя и записываются в массив retention, после чего массив формируется в команду «ATM=retention\r\n» и отправляется в микроконтроллер. Ниже приведен код программы обработки нажатия на кнопку.

```
/*----------Ось_X--------------------------*/
h_X = Convert.ToDouble(textBox5.Text);
phi_X = Convert.ToDouble(textBox2.Text);
rotation_frequency_X = Convert.ToInt32(textBox3.Text);
if (radioButton16.Checked) step_X= 1;
if (radioButton14.Checked) step_X= 2;
if (radioButton15.Checked) step_X= 8;
if (radioButton13.Checked) step_X= 16;
K_X =Convert.ToInt32(360 / (h_X * phi_X))*step_X;
delitel_X = Convert.ToInt32(60000 / (rotation_frequency_X*step_X*0.25*(360/phi_X)));
String comand = "ATDX=" + delitel_X + "\r\n";
port.Write(comand);
Thread.Sleep(150);
String answer = port.ReadExisting();
if (answer != "OK\r\n")
{
    port.Write(comand);
    Thread.Sleep(150);
    answer = port.ReadExisting();
    if (answer != "OK\r\n")
    {
        MessageBox.Show("Обрыв связи!");
    }
}
if(checkBox17.Checked==true) retention[7]='1';
else retention[7] = '0';

String ret = new String(retention);
String com = "ATM=" + ret + "\r\n";
port.Write(com);
Thread.Sleep(250);
String answ = port.ReadExisting();
if (answ != "OK\r\n")
{
    port.Write(com);
    Thread.Sleep(250);
    answ = port.ReadExisting();
    if (answ != "OK\r\n")
    {
        MessageBox.Show("Обрыв связи!")
    }
}
```

Рисунок 67 - Обработка нажатия кнопки «Установить»

Визуальный вид окон интерфейса пользователя см. ПРИЛОЖЕНИЕ Е.

Листинг программы для ПК см. ПРИЛОЖЕНИЕ И.

7 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

7.1 Основные возможные опасные процессы при эксплуатации манипулятора

1. Робот - манипулятор, представляющий собой многозвенный механизм, обладающий несколькими степенями подвижности и имеющий возможность перемещения звеньев с достаточно высокой скоростью, может оказаться в любой точке рабочей зоны неожиданно для рабочего, причинив ему весьма серьезную травму. Во избежание нанесения вреда следует исключить нахождение людей в рабочем пространстве робота – манипулятора [9];

2. Поведение робота, определяемое управляющей программой и качеством элементов устройства управления, в случае ошибок в программе либо сбоев работы концевых датчиков может стать вообще непредсказуемым. При выполнении роботом алгоритма управляющей программы, может произойти сбой при выполнении одного из циклов программы, связанных с опросом концевых датчиков или количеством выдаваемых импульсов, для коммутации обмоток двигателей. Также при управлении роботом пользователем т.е. в ручном режиме возможен сбой системы в случае выхода из строя концевых датчиков. При возникновении одной из вышеприведенных неисправностей следует отключить стенд, оповестить администрацию.

3. Манипулирование на значительных скоростях объектами, обладающими большими массами, при ненадежном их удержании либо ошибочном раскрытии захватного устройства представляет опасность травмирования выпавшим объектом.

4. При пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и

жизни человека. Во избежание поражения электрическим током необходимо твердо знать и выполнять правила безопасного пользования электроэнергией.

5. При коротком замыкании электропроводки необходимо отключить блок питания СУ путем вытаскивания питающего провода из розетки. И устранить неисправность в системе управлении роботом.

6. При возникновении возгорании на стенде его элементов или проводки, необходимо обесточить стенд. При малом возгорании накрыть место горения плотной тканью. При большом возгорании использовать огнетушитель.Если огонь перекинулся на другие объекты, и нет возможности потушить пожар, необходимо покинуть помещение. Позвонить в Единую службу спасения 01 (для операторов сотовой связи 01,112).

7.2 Инструкция по технике безопасности при эксплуатации робота манипулятора

При эксплуатации установки необходимо перед ее включением проверить:

- подключение всех проводов к элементам робота и к щиту подключения робота;

- осмотреть вилку и провод блока питания на наличие пробоев и других неисправностей;

- проверить правильность подключения датчиков и драйверов шаговых двигателей к микроконтроллеру, а также подключения драйверов к шаговым двигателям.

После предварительной проверки подключаются блоки питания шаговых двигателей и концевых датчиков к сети. Затем подключается и запускается компьютер, на котором установлено ПО управления роботом. Дождавшись полной загрузки компьютера, к нему подключается микроконтроллер в свободный USB–порт.

Затем на ПК запускается программа для управления роботом. Перед началом работы робота рекомендуется настроить режим работы шаговых

двигателей на осях робота для точности позиционирования в дискретном и позиционном режимах. В ручном режиме пользователю предоставляется визуальный вид робота и индикация датчиков конечных положений. При отсутствии сигналов на одном из датчиков требуется проверить соединение датчиков и МК, и далее производится проверка исправности блока питания и датчиков, в случае неустроенности неисправности следует сообщить администрации о данной проблеме. Если же при управлении роботом оказывается неработоспособной одна из осей робота, аналогично следует правильность подключений двигателей к драйверам и далее к МК, далее проверить исправность блока питания и двигателей, в случае неустроенности неисправности следует сообщить администрации о данной проблеме.

После завершения работы с роботом необходимо:

закрыть ПО управления роботом на ПК;

выключить ПК;

отключить питание системы управления роботом манипулятором.

7.3 Требования электробезопасности

При пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека

Во избежание поражения электрическим током необходимо твердо знать и выполнять следующие правила безопасного пользования электроэнергией:

1. Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. При обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

2. Во избежание повреждения изоляции проводов и возникновения коротких замыканий не разрешается:

а) вешать что-либо на провода;

б) закрашивать и белить шнуры и провода;

в) закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;

г) выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

3. Для исключения поражения электрическим током запрещается:

а) часто включать и выключать компьютер без необходимости;

б) прикасаться к экрану и к тыльной стороне блоков компьютера;

в) работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;

г) работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе

д) класть на средства вычислительной техники и периферийном оборудовании посторонние предметы.

3. Запрещается под напряжением очищать от пыли и загрязнения электрооборудование.

4. Запрещается проверять работоспособность электрооборудования в неприспособленных для эксплуатации помещениях с токопроводящими полами, сырых, не позволяющих заземлить доступные металлические части.

5. Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований.

6. Недопустимо под напряжением проводить ремонт средств вычислительной техники и периферийного оборудования.

7. Во избежание поражения электрическим током, при пользовании электроприборами нельзя касаться одновременно каких-либо трубопроводов, батарей отопления, металлических конструкций, соединенных с землей.

8. При обнаружении оборвавшегося провода необходимо немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей. Прикосновение к проводу опасно для жизни.

Во всех случаях поражения человека электрическим током немедленно вызывают врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему [10].

7.4 Требования по обеспечению пожарной безопасности

На рабочем месте запрещается иметь огнеопасные вещества

В помещениях запрещается:

а) зажигать огонь;

б) включать электрооборудование, если в помещении пахнет газом;

в) курить;

г) сушить что-либо на отопительных приборах;

д) закрывать вентиляционные отверстия в электроаппаратуре

Источниками воспламенения являются:

а) искра при разряде статического электричества

б) искры от электрооборудования

в) искры от удара и трения

г) открытое пламя

При возникновении пожароопасной ситуации или пожара персонал должен немедленно принять необходимые меры для его ликвидации, одновременно оповестить о пожаре администрацию.

Помещения с электрооборудованием должны быть оснащены огнетушителями типа ОУ-2, ОУБ-3 или ОП-3 [11,12].

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана автоматизированная система управления роботом – манипулятором на базе микроконтроллера, мною была предложена система управления с использованием UART интерфейса на МК и USB порта на персональном компьютере.

Был разработан протокол обмена данными между ПК и МК предусматривающий следующие команды:

- вращения вала двигателя на заданное число шагов;
- настройка режима фиксации вала двигателя при остановке;
- установка скорости вращения двигателя;
- чтение состояний концевых выключателей и датчиков технологической информации.

Для безопасного управления роботом манипулятором были добавлены датчики концевых положений на каждую ось, т.е. теперь пользователю не нужно следить за тем, чтобы робот не заехал за необрабатываемое крайнее положение.

Также было разработано программное обеспечение для управления роботом – манипулятором, которое предусматривает несколько режимов управления и настройка режима работы двигателей установленных на звеньях робота. Полный листинг программы на МК и ПК приведены в приложении З и приложении Ж соответственно.

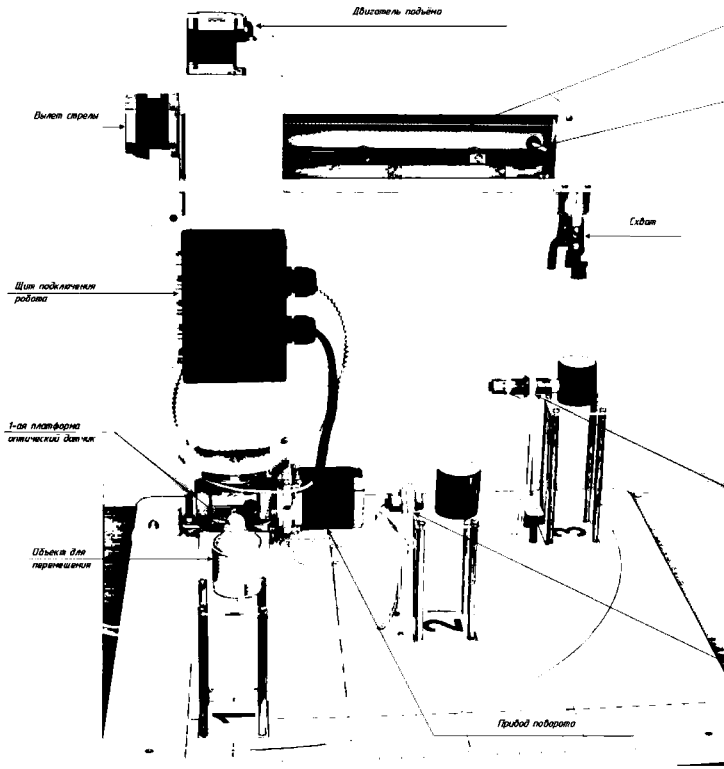
Данная система управления имеет возможность модернизации. Перспективой следующей модернизации является создание имитационной модели, отражающей движение робота в настоящем времени, которая будет отображена в приложении верхнего уровня.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

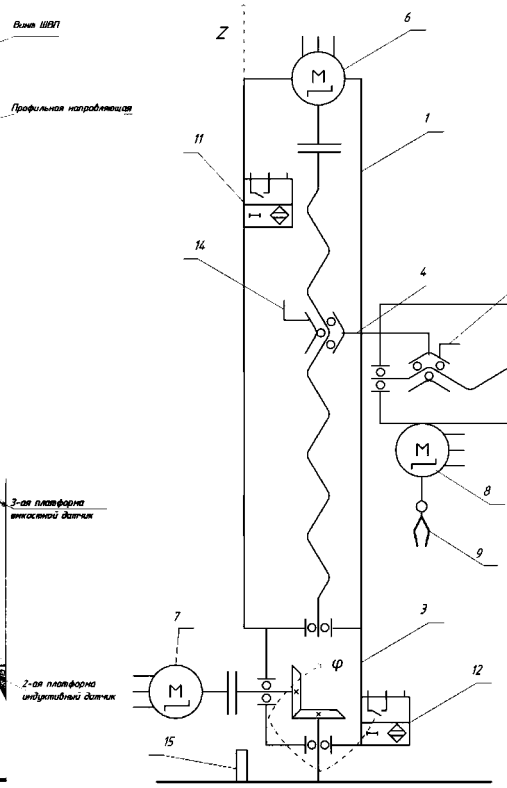
1. Purelogic.ru: Плата коммутации драйверов ШД/СД [Электронный ресурс]: © Purelogic R&D.– 2007 .– Режим доступа: https://purelogic.ru/-files/downloads/doc/Controller/BOB-5x2_Purelogic.pdf/. (Дата обращения-15.03.2020);
2. KakProsto.ru: Что такое Arduino и что с ним можно делать [Электронный ресурс]: KakProsto.ru: как просто сделать все. Режим доступа: https://www.kakprosto.ru/kak-920473-chto-takoe-arduino-i-chto-s-nim-mozhno-sdelat (Дата обращения-15.03.2020);
3. Arduino.ru: Что такое Ардуино? [Электронный ресурс]: Аппаратная платформа Arduino | Arduino.ru Режим доступа: http://arduino.ru/About (Дата обращения-15.03.2020);
4. Mypractic.ru: Уроки программирования Ардуино. [Электронный ресурс]: © ОБОРУДОВАНИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ.-2015.-Режим доступа: http://mypractic.ru/uroki-programmirovaniya-arduino-navigaciya-po-urokam (Дата обращения-24.02.2020);
5. AVR-START.RU: Опрос матричной клавиатуры [Электронный ресурс]: © AVR-START.RU.-2013.-Режим доступа: http://avr-start.ru/?p=1244 (Дата обращения-22.03.2020);
6. ru.wikipedia.org: Microsoft Visual Studio. [Электронный ресурс]: Wikipedia®. Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio (Дата обращения-2.05.2020);
7. Microsoft : SerialPort Class. [Электронный ресурс]: Microsoft.com. Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8 (Дата обращения-2.05.2020);

8. C# 2008 и платформа .NET 3.5 для профессионалов. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2009 – 1392 с.;
9. Библиотекарь.Ру: Основы робототехники. [Электронный ресурс]: bibliotekar.ru. Режим доступа: http://www.bibliotekar.ru/7-robot/69.htm (Дата обращения-15.06.2020);
10. Техника безопасности: Требования по технике безопасности при работе с вычислительной техникой. [Электронный ресурс]: saletty.narod.ru. Режим доступа: www.saletty.narod.ru/index.htm (Дата обращения-15.06.2020);
11. Fireman.club: Огнетушитель углекислотный ОУ-2 (ВСЕ): описание и приведение в действие. [Электронный ресурс]: Fireman.club. Режим доступа: https://fireman.club/statyi-polzovateley/ognetushitel-uglekislotnyie-ou-2-bce-ttx-i-privedenie-v-deystvie/ (Дата обращения-15.06.2020);
12. Fireman.club: Огнетушитель углекислотный ОУ-3 (ВСЕ): описание и приведение в действие. [Электронный ресурс]: Fireman.club. Режим доступа: https://fireman.club/statyi-polzovateley/ognetushitel-uglekislotnyiy-ou-3-bce-opisanie-i-privedenie-v-deystvie/ (Дата обращения-15.06.2020).

1. Внешний вид «УР-4»



2. Кинематическая схема робота



Поз. Обозн.	Наименование	Кол.	Примеч.
1	Башня, обеспечивает движение вдоль оси		
	Z (по вертикали)	1	
2	Рабочая стрела, обеспечивает движение		
	вдоль оси r (по горизонтали)	1	
3	Опорно-поворотное устройство, вращение башни		
	по азимуту φ	1	
4	Каретка с ШВП осей Z и r	1	
5	ШД (GD57STH56-2808A) вылета стрела	1	
6	ШД (GD57STH56-2808A) спуска/подъёма	1	
7	ШД (GD57STH56-2006A) поворота	1	
8	ШД (Nema 17 42CM06) схвата	1	
9	Схват	1	
10	Индуктивный датчик (LJBA3-2-Z/BX)		
	конечного положения по оси X	1	
11	Индуктивный датчик (LJBA3-2-Z/BX)		
	конечного положения по оси Y	1	
12	Индуктивный датчик (LJBA3-2-Z/BX)		
	конечного положения по оси Z	1	
13	Флажок конечного положения по оси X	1	
14	Флажок конечного положения по оси Y	1	
15	Флажок конечного положения по оси Z	1	

Изм.	Лист	№ докум.	Подп.	Дата	ВКР 164.016.15.03.04.СХ
Разраб.		Потапенко Н.Г.			Робот-манипулятор «УР-4»
Провер.		Сарычев В.В.			
Т.контр.		Сарычев В.В.			
Рецензент					Разработана автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR
Н.контр.		Сарычев В.В.			
Утвердил		Сарычев В.В.			

Литера	Масса	Масштаб
	1	

Лист 1 | Листов 6

АмГУ
Кафедра АПИиЭ

ПРИЛОЖЕНИЕ А

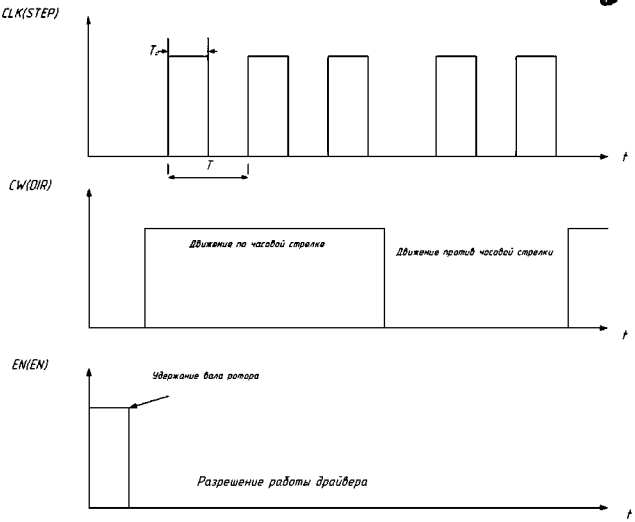
1. Список команд текстового протокола

Команда	Описание
- ATSX - ATSY - ATSZ - ATSG	Данная команда передает число шагов по осям X, Y, Z и сгвата методу step() класса StepDirDriver. Команда представляет последовательную запись символов («ATSX=» , n_steps, 13, 10), где n_steps заданное число шагов
- ATM	Команда задает режим фиксации вала двигателя при остановке по осям X, Y, Z и сгвата методу setMode() класса StepDirDriver. Команда представляет последовательную запись символов («ATM=», 00001111, 13, 10), где нулевой разряд двоичного числа соответствует управляющему сигналу по оси X, первый разряд по оси Y, второй разряд по оси Z, третий для сгвата. 1 – фиксация вала двигателя при остановке, 0 – отсутствие фиксации вала двигателя при остановке.
- ATRX1, ATRX2 - ATRY1, ATRY2 - ATRZ1, ATRZ2 - ATRG1, ATRG2	Команда предназначена для ручного управления осями робота с ПК. Команда представляет последовательную запись символов («ATRX», 1, 13, 10), где «X» – ось X робота, 1 – вращение вала двигателя по часовой стрелке, 2 – вращения вала двигателя против часовой стрелки
- ATDX - ATDY - ATDZ - ATDG	Данная команда предназначена для настройки частоты переключения фаз двигателей по осям X, Y, Z и сгвата. Команда представляет последовательную запись символов («ATDX=», delitel, 13, 10), где delitel делитель частоты коммутации фаз двигателя
- ATL?	Команда предназначена для опроса состояния концевых датчиков. Команда представляет последовательную запись символов («ATL?», 13, 10)

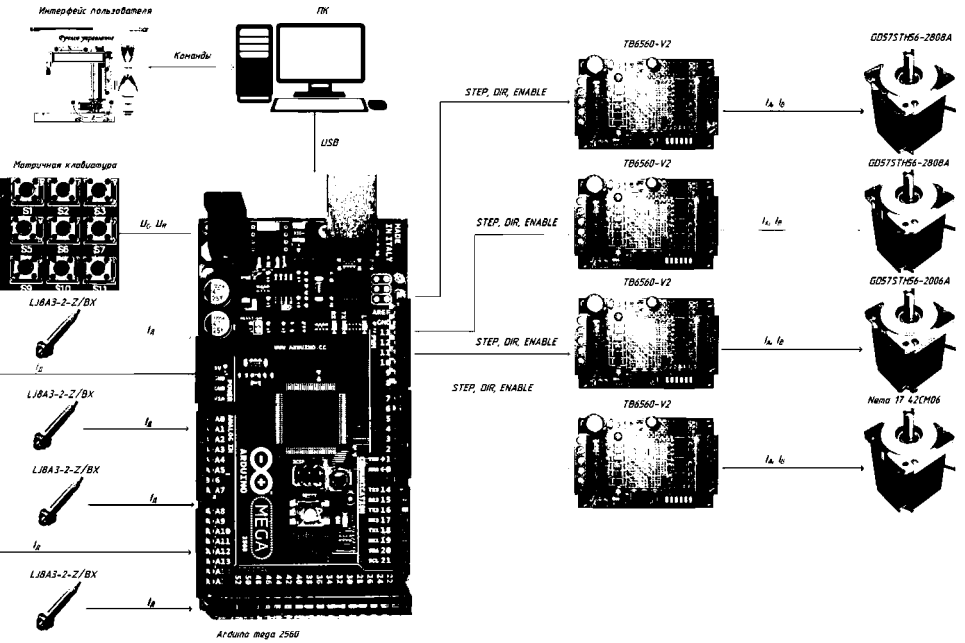
LJ8A3-2-Z/BX

LJ8A3-2-Z/BX

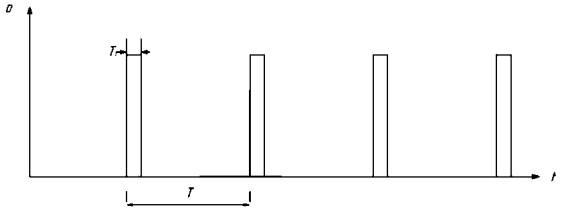
3. Временная диаграмма протокола Step/Dir



2. Функциональная схема управления роботом



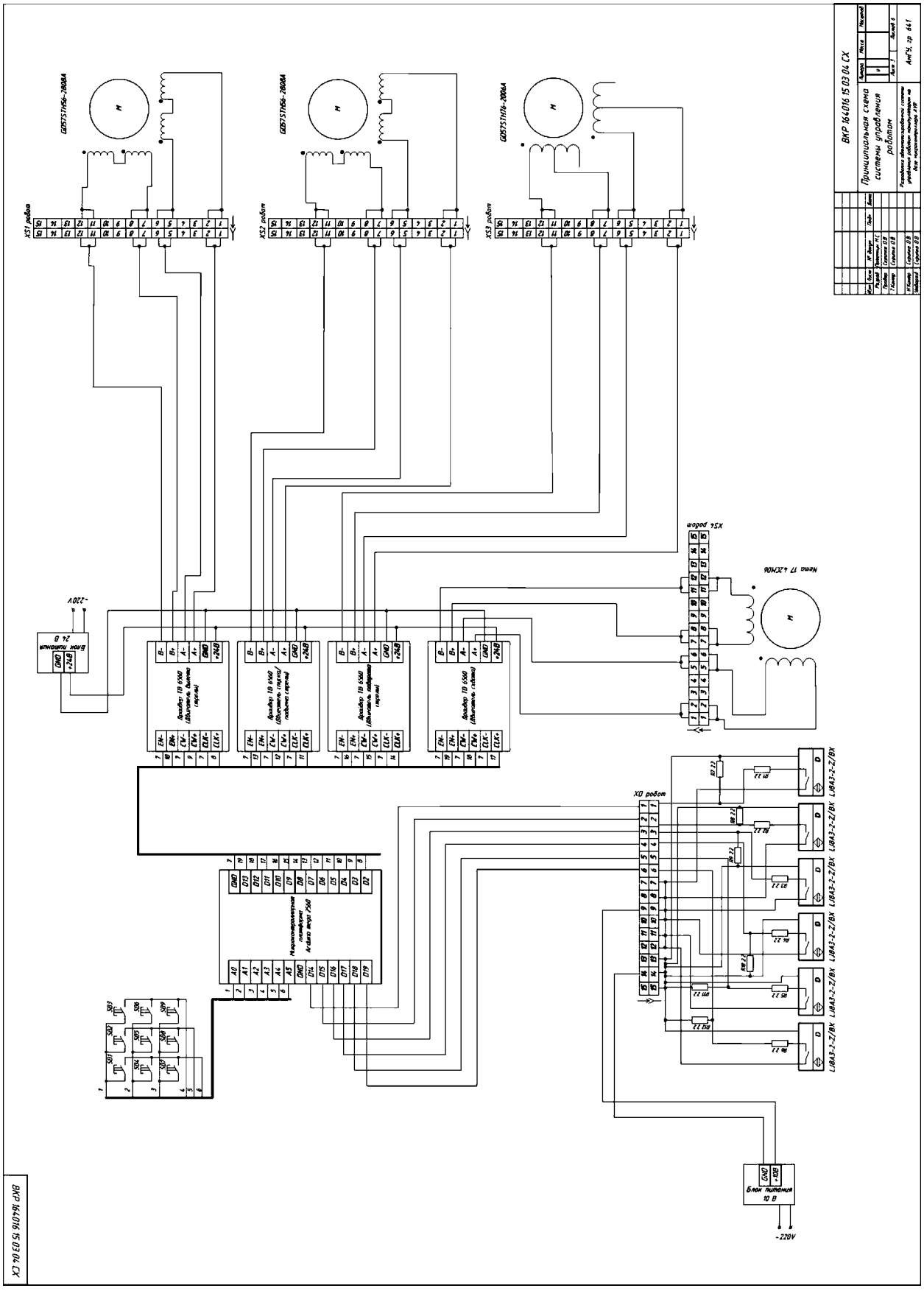
4. Временная диаграмма текстового протокола



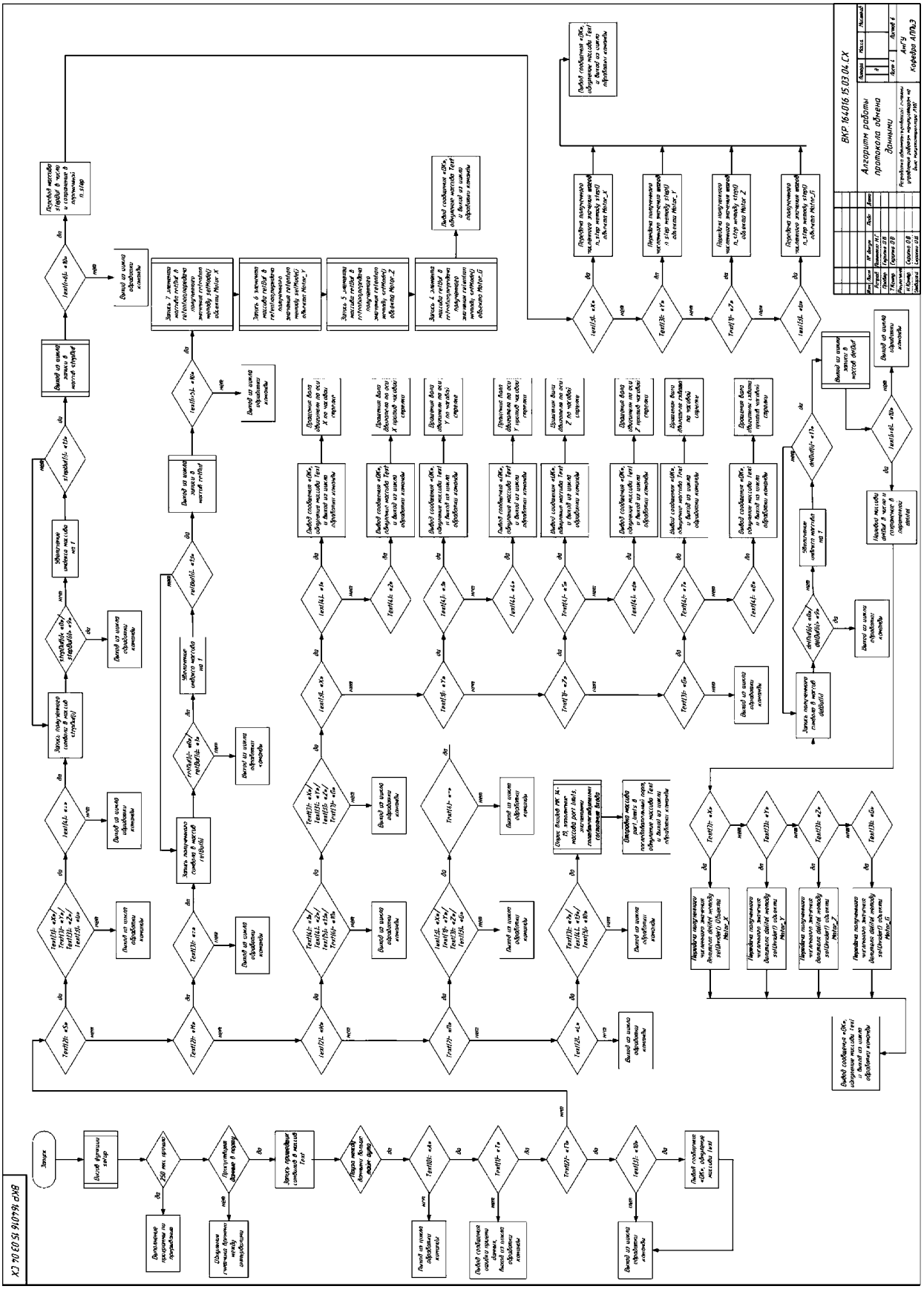
					ВКР 164016 15 03 04 СХ	
Изм.	Лист	№ докум.	Подп.	Дата	Система управления роботом Временные диаграммы	Литера: Масса: Масштаб:
Разраб.		Ананьин Н.Г.				
Провер.		Сергеева О.В.				Лист 2 Листов 6
Т.Контр.		Сергеева О.В.				
Реценз.					Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ Кафедра АПиЭ
Н.Контр.		Сергеева О.В.				
Утвердил		Горлова О.В.				

ПРИЛОЖЕНИЕ Б

ПРИЛОЖЕНИЕ В

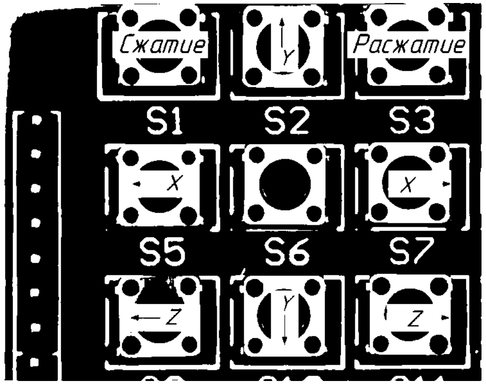


ПРИЛОЖЕНИЕ Г



ПРИЛОЖЕНИЕ Д

1. Назначение кнопок на матричной клавиатуре



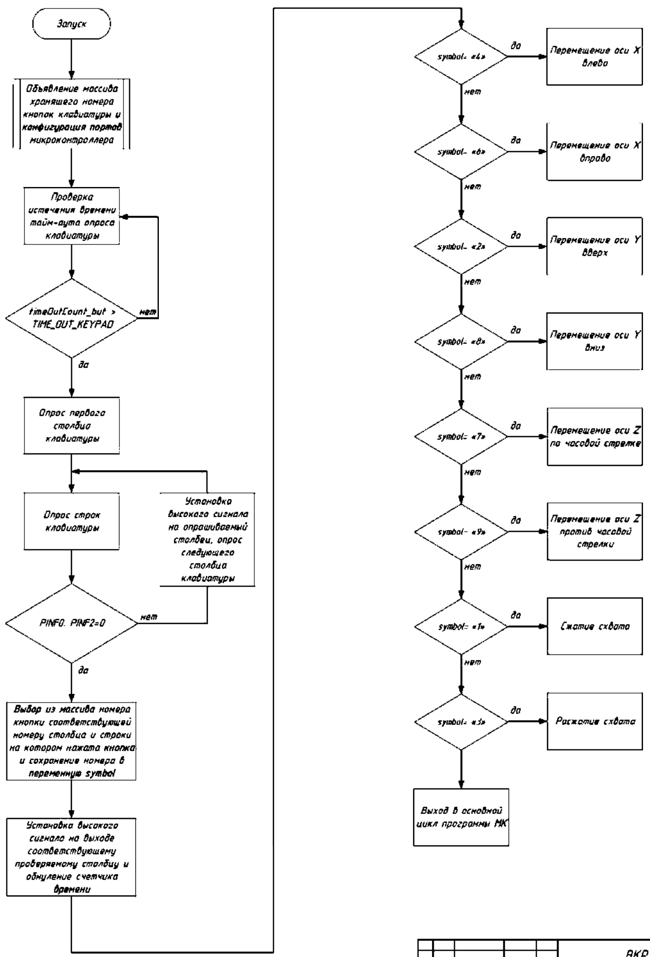
2. Назначение кнопок

«Сжатие» «Расжатие» - управление схватом робота;
«<-X» «X->» - перемещение стрелы робота по оси X;

↑Y Y↓ - перемещение стрелы робота по оси Y;

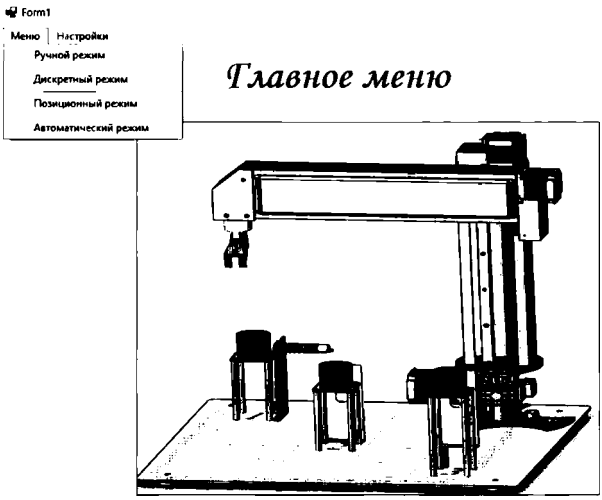
«<-Z» «Z->» - поворот робота по оси Z.

3. Алгоритм управления роботом с матричной клавиатуры

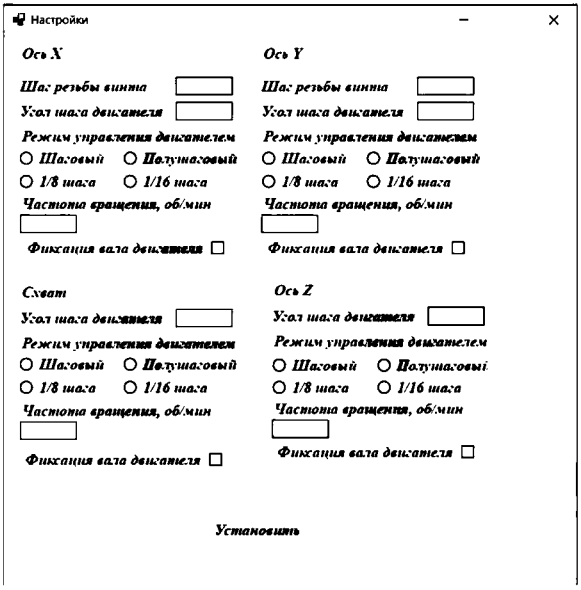


					ВКР 164016 15.03.04.СХ		
Изм.	Лист	№ докум.	Подп.	Дата	Ручное управление роботом	Литера	Масса	Масштаб
Разраб.		[illegible] Н.С.						
Провер.		Скрипко О.В.				Лист 5	Листов 6	
Т.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ Кафедра АППиЭ		
Рецензент								
Н.Контр.		Скрипко О.В.						
Утверд.		Скрипко О.В.						

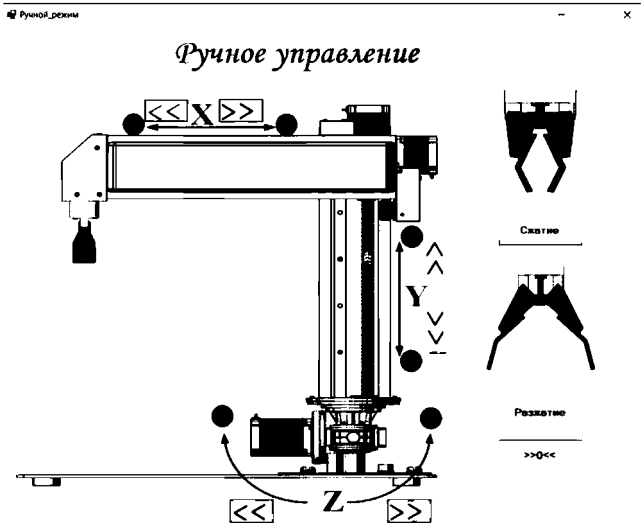
1. Стартовое окно



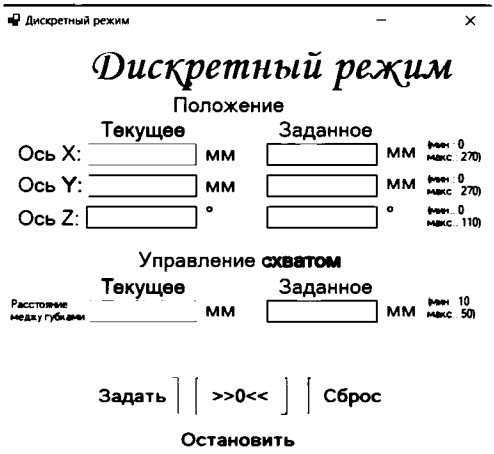
3. Окно настроек



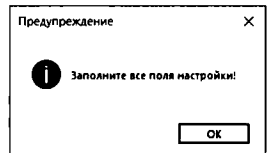
2. Окно ручного управления



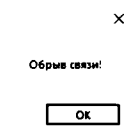
4. Окно дискретного режима



5. Окно предупреждения о некорректной работе в настройках



6. Окно предупреждения об обрыве связи между МК и ПК



						ВКР 164016.15.03.04. СХ		
							Литера	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата	Оконное приложение управления роботом				
Разраб.		Романенко Н.Е.							
Провер.		Сарапина О.В.							
Т.Контр.		Сарапина О.В.					Лист 6	Листов 6	
Нормоконтр.					Разработка автоматизированной системы управления роботом-манипулятором на базе тензодатчиков КУКА		АмГУ Кафедра АПЭ		
Утвердил		Сарапина О.В.							

ПРИЛОЖЕНИЕ Е

ПРИЛОЖЕНИЕ Ж

Техническое задание

1 Введение

Данное ТЗ распространяется на исследование учебного робота-манипулятора «УР-4», разработка системы управления на основе МК используя протокол STEP/DIR для работы с шаговыми двигателями и создание программного обеспечения для управления учебным роботом-манипулятора по протоколу AT команд.

Заказчик: Федеральное государственное бюджетное образовательное учреждение высшего образования Амурский государственный университет (ФГБОУ ВО АмГУ).

Исполнитель: Потемкин М. С.

Система разрабатывается на основании следующих документов:

• ФГОС направления подготовки бакалавров 15.03.04 автоматизации технологических процессов и производств;

• Учебный план направления подготовки бакалавров 15.03.04 автоматизации технологических процессов и производств;

Плановые сроки начала и окончания работ по созданию системы:

Начало: 13.01.2020

Окончание: 01.03.2020

2 Характеристики объекта автоматизации

Объектом автоматизации является учебный робот-манипулятор «УР-4», предназначенный для наглядного изучения технических характеристик и построения систем автоматизации.

Робот установлен на металлическую платформу с координатной сеткой с нанесенной на ней системой координат. На платформе установлено три площадки с тремя различными датчиками технологической информации, на каждую площадку по одному датчику. На осях робота установлены концевые индуктивные датчики, на каждой оси по два датчика.

Основными компонентами системы управления являются: микроконтроллер, драйверы шаговых двигателей, датчики концевые и технологической информации, персональный компьютер.

Микроконтроллер используется для подачи сигналов на ДШД. Также микроконтроллер опрашивает концевые датчики, ведет обмен данными с ПК отправляя компьютеру информацию ДТИ и концевых выключателей. Еще одной функцией является передача числа шагов шагового двигателя в протокол STEP/DIR, на выходе которого осуществляется подача сигналов на ДШД.

Драйвера на основании полученных сигналов от микроконтроллера осуществляет коммутацию обмоток двигателей, которые приводят в движение робота по осям: X, Y, Z.

На персональном компьютере находится оконное приложение для управления роботом, которое содержит интерфейс пользователя для управления роботом.

Робот-манипулятор эксплуатируется в учебной лаборатории, металлическая платформа с роботом установлена на специальном стеллаже. Диапазон рабочих температур: от +10 до +35 °С, при влажности не более 80%.

3 Цели создания системы

Разработка новой системы управления с помощью МК AVR. Совмещение работы микроконтроллера и компьютера для последующего управления ШД и считывания состояний датчиков.

Разработка программного обеспечения для управления роботом-манипулятором, начиная с приведения алгоритмов работы программы и разработки исходного текста, завершая тестированием на практике разработанного ПО.

Исследование имеющейся системы управления роботом-манипулятором приводится для изучения принципов управления шаговыми

двигателями и обработки сигналов с датчиков концевых и технологической информации.

Добавление в имеющуюся систему энкодеров для создания обратной связи МК с шаговыми двигателями.

Установка дополнительных концевых выключателей.

Совершенствование лабораторной базы кафедры.

4 Функции и задачи создаваемой АСУ

Основными компонентами системы управления являются: микроконтроллер, драйверы шаговых двигателей, датчики концевые и технологической информации, персональный компьютер.

Микроконтроллер используется для подачи сигналов на ДШД, т.е. сигнал шага в виде единичного импульса, сигнал направления вращения вала двигателя и сигнал разрешения работы двигателя. Также микроконтроллер опрашивает сенсоры, ведет обмен данными с ПК используя AT команды, отправляя компьютеру информацию ДТИ и концевых выключателей. Еще одной функцией является передача числа шагов шагового двигателя в протокол STEP/DIR, на выходе которого осуществляется подача сигналов на ДШД.

Драйвера на основании полученных сигналов от микроконтроллера осуществляет коммутацию обмоток двигателей, которые приводят в движение робота по осям: X, Y, Z и схват.

На персональном компьютере находится оконное приложение для управления роботом, которое предоставляет пользователю: выбор режима работы, управление в ручном режиме, позиционирование осей робота, индикацию состояний датчиков концевых и технологической информации. В приложении содержатся алгоритмы работы управления осями робота, который обрабатывает полученные сигналы позиционирования робота в ручном и координатных режимах в число шагов и направления вращения шагового двигателя. Протокол AT команд по которому осуществляется обмен

данными с микроконтроллером. Компьютер поддерживает связь с микроконтроллером через последовательный USB порт.

Задачи проекта:

1. Создание структурной схемы работы системы МК и ПК;
2. Разработка протокола управления шаговыми двигателями;
3. Разработка протокола обмена данными между ПК и МК;
4. Создание ручного управления роботом
5. Разработка интерфейса пользователя для управления роботом с ПК

ПРИЛОЖЕНИЕ И

Листинг программы на Arduino

```
#include <TimerOne.h>
#include <StepDirDriver.h>
int flagX, flagY, flagZ;
char Text[20];     // тестовый буфер
char  stepBuf[10]; // буфер для кол-ва шагов
char retBuf[10];   //буфер для установки удержания ротора
char delBuf[5];   //буфер для установки делителя частоты вращения
char  port_limits[5];//  буфер  для  хранения  состояния  концевых
датчиков
int delitel;
int s;  // индекс массива Text[20]
int i; // индекс массива stepBuf[10],retBuf[10]
#define TIME_OUT 200 // время тайм-аута между командами (* 0,25
мс)
#define TIME_OUT_KEYPAD 200 // время тайм-аута между командами (*
0,25 мс)
int  timeOutCount; // счетчик времени между приемом данных
int  timeOutCount_but;//  счетчик  времени  обработки  матричной
клавиатуры
int s3; //код третьего элемента массива Text[20]
int s4; //код четвертого элемента массива Text[20]
long n_step = 0; //число шагов
int retention = 1; //удержание ротора
int state_button = 0;//Флаг состояний на интерфейсе пользователя
int st;// номер опрашиваемого столбца
char portState[3]= {0xF7,0xEF,0xDF};
char inputState[3]={0x01,0x02,0x04};
char symbol;//Переменная для хранения символа
int j;// номер опрашиваемой строки
const char key [3][3]=
{{'1','4','7'},
 {'2','5','8'},
 {'3','6','9'}};
StepDirDriver   Motor_X(2,    3,    4);//создаем    объект    типа
StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver   Motor_Y(5,    6,    7);//создаем    объект    типа
StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver   Motor_Z(8,    9,    10);//создаем    объект    типа
StepDirDriver, задаем выводы для сигналов(step,dir,enable)
StepDirDriver   Motor_G(11,   12,   13);//создаем   объект   типа
StepDirDriver, задаем выводы для сигналов(step,dir,enable)


void setup()
{
  Serial.begin(9600);//Запуск последовательного порта
```

```
  Timer1.initialize(250);  // инициализация таймера 1, период 250
мкс
  Timer1.attachInterrupt(timerInterrupt,   250);      //   задаем
обработчик прерываний
  Motor_X.setMode(retention); // задаем режимы коммутации фаз и
остановки
  Motor_Y.setMode(retention); // задаем режимы коммутации фаз и
остановки
  Motor_Z.setMode(retention); // задаем режимы коммутации фаз и
остановки
  Motor_G.setMode(retention); // задаем режимы коммутации фаз и
остановки
  Motor_X.setDivider(10);  //  установка  делителя  частоты  для
коммутации фаз
  Motor_Y.setDivider(10);  //  установка  делителя  частоты  для
коммутации фаз
  Motor_Z.setDivider(10);  //  установка  делителя  частоты  для
коммутации фаз
  Motor_G.setDivider(10);  //  установка  делителя  частоты  для
коммутации фаз
  Motor_X.step(0); // инициирует поворот ротора на заданное число
шагов
  Motor_Y.step(0); // инициирует поворот ротора на заданное число
шагов
  Motor_Z.step(0); // инициирует поворот ротора на заданное число
шагов
  Motor_G.step(0); // инициирует поворот ротора на заданное число
шагов
  DDRF|=(0<<0)|(0<<1)|(0<<2)|(1<<3)|(1<<4)|(1<<5);//Конфигурация
портов к которым подключена матричная клавиатура
  PORTF|=(1<<0)|(1<<1)|(1<<2);//Подключение         подтягивающих
резисторов по строкам клавиатуры
  pinMode(14, INPUT);//1 Концевой датчик по оси X
  pinMode(15, INPUT);//2 Концевой датчик по оси X
  pinMode(16, INPUT);//1 Концевой датчик по оси Y
  pinMode(17, INPUT);//2 Концевой датчик по оси Y
  pinMode(18, INPUT);//1 Концевой датчик по оси Z
  pinMode(19, INPUT);//2 Концевой датчик по оси Z
}

void loop()
{
  if (Serial.available() == 0)
  {
    timeOutCount = 0;
    s = 0;
  }
  while (Serial.available())
  {
    state_button = 0;
    char sym = Serial.read();
```

```
    Text[s] = sym;
    s++;
    delay(16);
  }
if (timeOutCount_but > TIME_OUT_KEYPAD)
{
  symbol=0;
  for(st=0; st<3; st++)
  {
    PORTF=portState[st];
    for(j=0; j<3; j++)
    {
      if(((PINF&inputState[j])==0))
      {
        Serial.println(key [st][j]);
        symbol=key [st][j];
      }
    }
    PORTF|=(1<<st);
    timeOutCount_but=0;
  }
}
if(symbol=='4')
{
      digitalWrite(3,1);
      Step_X();
}
if(symbol=='6')
{
      digitalWrite(3,0);
      Step_X();
}
if(symbol=='2')
{
  digitalWrite(6,1);
  Step_Y();
}
if(symbol=='8')
{
  digitalWrite(6,0);
  Step_Y();
}
if(symbol=='7')
{
  digitalWrite(8,1);
  Step_Z();
}
if(symbol=='9')
{
  digitalWrite(8,0);
  Step_Z();
```

```
}
if(symbol=='1')
{
  digitalWrite(12,1);
  Step_G();
}
if(symbol=='3')
{
  digitalWrite(12,0);
  Step_G();
}
  if ( timeOutCount > TIME_OUT )
  {
    while (true)
    {
      if ( Text[0] != 'A') break; // ошибка
      if ( Text[1] != 'T')
      {
        Serial.print("Bad\r\n");
        break;
      }
      s3 = Text[2]; // третий символ команды
      if ( s3 == 13 )
      {
        if ( Text[3] != 10 ) break; // ошибка
        Serial.print("OK\r\n");
        for (int q = 0; q < s; q++)//Здесь происходит обнуление
массива в которм хранится
        {    //команда,    т.к.    следующая    команда    может
интерпретироваться
          Text[q] = 0;           //неправильно из - за оставшегося
мусора от предыдущей.
        }
        break;
      }
      else if (s3 == 'S')
      {
        s4 = Text[3]; // четвертый символ команды
        if ((s4 == 'X') || (s4 == 'Y') || (s4 == 'Z') || (s4 ==
'G'))
        {
          if(Text[4]!='=') break;
            i=0;
            if(Text[5]!='-')
            {
                while(i<8)
              {
                stepBuf[i]=Text[i+5];
                if(stepBuf[i]==13) break;//Конец команды
                if((stepBuf[i]<'0')||(stepBuf[i]>'9')) break;
                i++;
```

```
        }
        if((stepBuf[i]!=13)||(i>7)) break;//Ошибка
        stepBuf[i]=0;
        if(Text[i+6]!=10) break;
        n_step=atoi(stepBuf);
      }
      if(Text[5]=='-')
      {
          while(i<8)
        {
          stepBuf[i]=Text[i+6];
          if(stepBuf[i]==13) break;//Конец команды
          if((stepBuf[i]<'0')||(stepBuf[i]>'9')) break;
          i++;
        }
        if((stepBuf[i]!=13)||(i>7)) break;//Ошибка
        stepBuf[i]=0;
        if(Text[i+7]!=10) break;
        n_step=-atoi(stepBuf);
      }
    switch (s4)
    {
      case 'X':
        Motor_X.step(n_step);
        break;
      case 'Y':
        Motor_Y.step(n_step);
        break;
      case 'Z':
        Motor_Z.step(n_step);
        break;
      case 'G':
        Motor_G.step(n_step);
        break;
    }
    Serial.print("OK\r\n");
    for (int q = 0; q < s; q++)
    {
      Text[q] = 0;
    }
    break;
  }
  break;
}
else if (s3 == 'M')
{
  if (Text[3] != '=')break;
  i = 0;
  while (i < 9)
  {
    retBuf[i] = Text[i + 4];
```

```
          if ( retBuf[i] == 13 ) break; // конец команды
          if ((retBuf[i] != '0') && (retBuf[i] != '1')) break;
          i++;
        }
        if ( (retBuf[i] != 13) || (i > 8) ) break; // ошибка
        retBuf[i] = 0;
        if ( Text[i + 5] != 10 ) break;
        retention = retBuf[7] - 48;
        Motor_X.setMode(retention);
        retention = retBuf[6] - 48;
        Motor_Y.setMode(retention);
        retention = retBuf[5] - 48;
        Motor_Z.setMode(retention);
        retention = retBuf[4] - 48;
        Motor_G.setMode(retention);
        Serial.print("OK\r\n");
        for (int q = 0; q < s; q++)
        {
          Text[q] = 0;
        }
        break;
      }
      else if (s3 == 'R')
      {
        if ((Text[4] != '1') && (Text[4] != '2') || (Text[5] !=
13) || (Text[6] != 10))break;
        if ((Text[3] == 'X') || (Text[3] == 'Y') || (Text[3] ==
'Z') || (Text[3] == 'G'))
        {
          if (Text[3] == 'X')
          {
            switch (Text[4])
            {
              case '1':
                state_button = 1;
                break;
              case '2':
                state_button = 2;
                break;
            }
          }
          else if (Text[3] == 'Y')
          {
            switch (Text[4])
            {
              case '1':
                state_button = 3;
                break;
              case '2':
                state_button = 4;
                break;
```

```
            }
          }
          else if (Text[3] == 'Z')
          {
            switch (Text[4])
            {
              case '1':
                state_button = 5;
                break;
              case '2':
                state_button = 6;
                break;
            }
          }
          else if (Text[3] == 'G')
          {
            switch (Text[4])
            {
              case '1':
                state_button = 7;
                break;
              case '2':
                state_button = 8;
                break;
            }
          }
          Serial.print("OK\r\n");
          for (int q = 0; q < s; q++)
          {
            Text[q] = 0;
          }
        }
        break;
      }
       else if (s3 == 'D')
      {
        s4 = Text[3]; // четвертый символ команды
        if ((s4 != 'X') && (s4 != 'Y') && (s4 != 'Z') && (s4 !=
'G'))break;
        if(Text[4]!='=') break;
            i=0;
                while(i<5)
              {
                delBuf[i]=Text[i+5];
                if(delBuf[i]==13) break;//Конец команды
                if((delBuf[i]<'0')||(stepBuf[i]>'9')) break;
                i++;
              }
              if((delBuf[i]!=13)||(i>6)) break;//Ошибка
              delBuf[i]=0;
              if(Text[i+6]!=10) break;
```

```
            delitel=atoi(delBuf);
            switch (s4)
          {
            case 'X':
              Motor_X.setDivider(delitel);
              break;
            case 'Y':
              Motor_Y.setDivider(delitel);
              break;
            case 'Z':
              Motor_Z.setDivider(delitel);
              break;
            case 'G':
              Motor_G.setDivider(delitel);
              break;
          }
          Serial.print("OK\r\n");
          for (int q = 0; q < s; q++)
          {
            Text[q] = 0;
          }
          break;
    }
     else if (s3 == 'L')
     {
      if ((Text[3] != '?')|| (Text[4] != 13) || (Text[5] !=
10))break;
      if (digitalRead(14)==HIGH)port_limits[0]='1';
           else if (digitalRead(15)==HIGH)port_limits[1]='1';
           else if (digitalRead(16)==HIGH)port_limits[2]='1';
           else if (digitalRead(17)==HIGH)port_limits[3]='1';
           else if (digitalRead(18)==HIGH)port_limits[4]='1';
           else if (digitalRead(19)==HIGH)port_limits[5]='1';
           Serial.print(port_limits);
           for (int q = 0; q < s; q++)
          {
            Text[q] = 0;
          }
          break;
     }
    break;
    }
    while (state_button == 1)
    {
      n_step++;
      Motor_X.step(n_step);
      if (Serial.available() != 0)
      {
        n_step=0;
        Motor_X.step(n_step);
        state_button = 0;
```

```
    break;
  }
  else if (digitalRead(14)==HIGH)
  {
    Serial.print("end\r\n");
    n_step=0;
    Motor_X.step(n_step);
    state_button = 0;
    flagX=1;
  }
  if(flagX==2)
  {
    if(digitalRead(15)==HIGH)
    {
      Serial.print("st!\r\n");
      flagX=0;
    }
  }
}
while (state_button == 2)
{
  n_step--;
  Motor_X.step(n_step);
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_X.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(15)==HIGH)
  {
    Serial.print("st\r\n");
    n_step=0;
    Motor_X.step(n_step);
    state_button = 0;
    flagX=2;
  }
  if(flagX==2)
  {
    if(digitalRead(14)==HIGH)
    {
      Serial.print("end!\r\n");
      flagX=0;
    }
  }
}
while (state_button == 3)
{
  n_step++;
  Motor_Y.step(n_step);
```

```
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_Y.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(16)==HIGH)
  {
    Serial.print("down\r\n");
    n_step=0;
    Motor_Y.step(n_step);
    state_button = 0;
    flagY=1;
  }
  if(flagY==2)
  {
    if(digitalRead(17)==HIGH)
    {
      Serial.print("up!\r\n");
      flagY=0;
    }
  }
}
while (state_button == 4)
{
  n_step--;
  Motor_Y.step(n_step);
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_Y.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(17)==HIGH)
  {
    Serial.print("up\r\n");
    n_step=0;
    Motor_Y.step(n_step);
    state_button = 0;
    flagY=2;
  }
  if(flagY==2)
  {
    if(digitalRead(16)==HIGH)
    {
      Serial.print("down!\r\n");
      flagY=0;
    }
  }
```

```
}
while (state_button == 5)
{
  n_step++;
  Motor_Z.step(n_step);
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_Z.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(18)==HIGH)
  {
    Serial.print("left\r\n");
    n_step=0;
    Motor_Z.step(n_step);
    state_button = 0;
    flagZ=1;
  }
  if(flagZ==2)
  {
    if(digitalRead(19)==HIGH)
    {
      Serial.print("right!\r\n");
      flagZ=0;
    }
  }
}
while (state_button == 6)
{
  n_step--;
  Motor_Z.step(n_step);
  if (Serial.available() != 0)
  {
    n_step=0;
    Motor_Z.step(n_step);
    state_button = 0;
    break;
  }
  else if (digitalRead(19)==HIGH)
  {
    Serial.print("right\r\n");
    n_step=0;
    Motor_Z.step(n_step);
    state_button = 0;
    flagZ=2;
  }
  if(flagZ==2)
  {
    if(digitalRead(18)==HIGH)
```

```
        {
          Serial.print("left!\r\n");
          flagZ=0;
        }
      }
    }
    while (state_button == 7)
    {
      n_step++;
      Motor_G.step(n_step);
      if (Serial.available() != 0)
      {
        n_step=0;
        Motor_G.step(n_step);
        break;
      }
    }
    while (state_button == 8)
    {
      n_step--;
      Motor_G.step(n_step);
      if (Serial.available() != 0)
      {
        n_step=0;
        Motor_G.step(n_step);
        break;
      }
    }
    timeOutCount = 0;
  }
}
void Step_X()
{
      digitalWrite(2, HIGH);
      delayMicroseconds(10);
      digitalWrite(2, LOW);
      delayMicroseconds(2500);
}
void Step_Y()
{
      digitalWrite(5, HIGH);
      delayMicroseconds(10);
      digitalWrite(5, LOW);
      delayMicroseconds(2500);
}
void Step_Z()
{
   digitalWrite(8, HIGH);
      delayMicroseconds(10);
      digitalWrite(8, LOW);
      delayMicroseconds(2500);
```

```
}
void Step_G()
{
   digitalWrite(11, HIGH);
      delayMicroseconds(10);
      digitalWrite(11, LOW);
      delayMicroseconds(2500);
}
void  timerInterrupt()
{
  Motor_X.control(); // управление двигателем
  Motor_Y.control(); // управление двигателем
  Motor_Z.control(); // управление двигателем
  Motor_G.control(); // управление двигателем
  timeOutCount++; // счетчик времени между приемом данных
  timeOutCount_but++;
}
```

ПРИЛОЖЕНИЕ К

Листинг программы ручного режима

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;   //Подключение класса для работы с COM портом

namespace WindowsFormsApplication4
{
    public partial class Ручной_режим : Form
    {
        private SerialPort port;
        public Ручной_режим()
        {
            InitializeComponent();
            Init();
        }
        private void Init()
        {
            port = new SerialPort();
            port.BaudRate = 9600;
            port.PortName = "COM3";
            port.Open();
        }
        private void Ручной_режим_Load(object sender, EventArgs e)
        {
            shapeContainer1.BringToFront();
            port.Write("ATL?\r\n");
            Thread.Sleep(100);
            String answer = port.ReadExisting();
            if (answer == "")
            {
                port.Write("ATL?\r\n");
                Thread.Sleep(100);
                answer = port.ReadExisting();
                if (answer == "")
                {
                    MessageBox.Show("Обрыв связи!");
                    return;
                }
            }
```

```
            if (answer != "")
            {
                char[] limits = answer.ToCharArray();
                if  (limits[0]  ==  '1')  ovalShape1.FillColor  =
Color.Red;
                else ovalShape1.FillColor = Color.Lime;
                if  (limits[1]  ==  '1')  ovalShape3.FillColor  =
Color.Red;
                else ovalShape3.FillColor = Color.Lime;
                if  (limits[2]  ==  '1')  ovalShape2.FillColor  =
Color.Red;
                else ovalShape2.FillColor = Color.Lime;
                if  (limits[3]  ==  '1')  ovalShape4.FillColor  =
Color.Red;
                else ovalShape4.FillColor = Color.Lime;
                if  (limits[4]  ==  '1')  ovalShape6.FillColor  =
Color.Red;
                else ovalShape6.FillColor = Color.Lime;
                if  (limits[5]  ==  '1')  ovalShape5.FillColor  =
Color.Red;
                else ovalShape5.FillColor = Color.Lime;
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private     void     button1_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRX2\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRX2\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private     void     button3_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRX1\r\n");
```

```
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRX1\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private     void     button5_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRY1\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRY1\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private     void     button2_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRY2\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
```

```
            {
                port.Write("ATRY2\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private void button8_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private     void     button8_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRZ1\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRZ1\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private     void     button9_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRZ2\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRZ2\r\n");
                Thread.Sleep(150);
```

```
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
            timer1.Start();
        }
        private void button9_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
            timer1.Stop();
        }
        private void button7_Click(object sender, EventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
        }
        private     void     button7_MouseDown(object     sender,
MouseEventArgs e)
        {
            port.Write("ATRG1\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRG1\r\n");
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
        }
        private     void     button10_MouseDown(object     sender,
MouseEventArgs e)
        {
            Thread.Sleep(100);
            port.Write("1");
        }
        private void button10_Click(object sender, EventArgs e)
        {
            port.Write("ATRG2\r\n");
            Thread.Sleep(150);
            String answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                port.Write("ATRG2\r\n");
                Thread.Sleep(150);
```

```
            answer = port.ReadExisting();
            if (answer != "OK\r\n")
            {
                MessageBox.Show("Обрыв связи!");
            }
        }
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        switch (port.ReadExisting())
        {
            case "st\r\n":
                ovalShape1.FillColor = Color.Red;
                break;
            case "end\r\n":
                ovalShape2.FillColor = Color.Red;
                break;
            case "st!\r\n":
                ovalShape1.FillColor = Color.Lime;
                break;
            case "end!\r\n":
                ovalShape2.FillColor = Color.Lime;
                break;
            case "up\r\n":
                ovalShape3.FillColor = Color.Red;
                break;
            case "down\r\n":
                ovalShape4.FillColor = Color.Red;
                break;
            case "up!\r\n":
                ovalShape3.FillColor = Color.Lime;
                break;
            case "down!\r\n":
                ovalShape4.FillColor = Color.Lime;
                break;
            case "left\r\n":
                ovalShape5.FillColor = Color.Red;
                break;
            case "right\r\n":
                ovalShape6.FillColor = Color.Red;
                break;
            case "left!\r\n":
                ovalShape5.FillColor = Color.Lime;
                break;
            case "right!\r\n":
                ovalShape6.FillColor = Color.Lime;
                break;
        }
    }
```

```
        private   void   Ручной_режим_FormClosed(object   sender,
FormClosedEventArgs e)
        {
            Form1 F1 = new Form1();
            port.Close();
            F1.textBox1.Text = this.textBox1.Text;
            F1.textBox2.Text = this.textBox2.Text;
            F1.textBox3.Text = this.textBox3.Text;
            F1.textBox4.Text = this.textBox4.Text;
            F1.Visible = true;
        }
    }
}
```

Листинг программы дискретного режима

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using  System.IO.Ports;   //Подключение  класса  для  работы  с  COM
портом

namespace WindowsFormsApplication4
{
    public partial class Дискретный_режим : Form
    {
        //int t;
        int n_step_X;
        SerialPort port;//Создание объекта класса
        public Дискретный_режим()
        {
            InitializeComponent();
            Init();//Вызов метода init
        }
        private void Init()
        {
            port = new SerialPort();
            port.BaudRate = 9600;
            port.PortName = "COM3";
            port.Open();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            /*-------ось X-------------------------*/
```

```
            int  K_X=Convert.ToInt32(textBox2.Text);//Коэффициент
перевод мм в число шагов
             n_step_X=      Convert.ToInt32(textBox8.Text)      *
K_X;//число шагов
            if    (Convert.ToInt32(textBox8.Text)    >=    0    &&
Convert.ToInt32(textBox8.Text) <= 270)
            {
               if                ((Convert.ToInt32(textBox8.Text)
>=Convert.ToInt32(textBox1.Text)))
                {
                    n_step_X  =  Convert.ToInt32(textBox8.Text)  -
Convert.ToInt32(textBox1.Text);
                    n_step_X=n_step_X* K_X;
                }
               else    if    ((Convert.ToInt32(textBox8.Text)    <
Convert.ToInt32(textBox1.Text)))
                {
                   n_step_X  =  Convert.ToInt32(textBox1.Text)  -
Convert.ToInt32(textBox8.Text);
                   n_step_X = -n_step_X * K_X;
                }
                String      comand_X      =      "ATSX="      +
Convert.ToString(n_step_X)+"\r"+"\n";
                port.Write(comand_X);
                Thread.Sleep(200);
                String answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand_X);
                    Thread.Sleep(200);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                        return;
                    }
            }
                textBox1.Text = textBox8.Text;
            }
            else
            {
                MessageBox.Show("Неправильно заданное значение по
оси X");
            }
                /*-------ось Y-------------------------*/
            int K_Y=Convert.ToInt32(textBox3.Text);
            int n_step_Y = Convert.ToInt32(textBox6.Text) * K_Y;
            if    (Convert.ToInt32(textBox6.Text)    >=    0    &&
Convert.ToInt32(textBox6.Text) <= 270)
            {
```

```
                if      ((Convert.ToInt32(textBox6.Text)      >=
Convert.ToInt32(textBox5.Text)))
                {
                    n_step_Y  =  Convert.ToInt32(textBox6.Text)  -
Convert.ToInt32(textBox5.Text);
                    n_step_Y = n_step_Y * K_Y;
                }
                else   if   ((Convert.ToInt32(textBox6.Text)   <
Convert.ToInt32(textBox5.Text)))
                {
                    n_step_Y  =  Convert.ToInt32(textBox5.Text)  -
Convert.ToInt32(textBox6.Text);
                    n_step_Y = -n_step_Y * K_Y;
                }
                String      comand_Y      =      "ATSY="      +
Convert.ToString(n_step_Y) + "\r" + "\n";
                port.Write(comand_Y);
                Thread.Sleep(200);
                String answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand_Y);
                    Thread.Sleep(200);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                        return;
                    }
                }
                textBox5.Text = textBox6.Text;
            }
            else
            {
                MessageBox.Show("Неправильно заданное значение по
оси Y");
            }
                /*-------ось Z-------------------------*/
            int K_Z=Convert.ToInt32(textBox11.Text);
            int n_step_Z = Convert.ToInt32(textBox7.Text) * K_Z;
            if    (Convert.ToInt32(textBox7.Text)    >=    0    &&
Convert.ToInt32(textBox7.Text) <= 110)
            {
                if      ((Convert.ToInt32(textBox7.Text)      >=
Convert.ToInt32(textBox4.Text)))
                {
                    n_step_Z  =  Convert.ToInt32(textBox7.Text)  -
Convert.ToInt32(textBox7.Text);
                    n_step_Z = n_step_Z * K_Z;
                }
```

```
                else   if   ((Convert.ToInt32(textBox7.Text)   <
Convert.ToInt32(textBox4.Text)))
                {
                    n_step_Z  =  Convert.ToInt32(textBox4.Text)  -
Convert.ToInt32(textBox7.Text);
                    n_step_Z = -n_step_Z * K_Z;
                }
                String      comand_Y      =      "ATSZ="      +
Convert.ToString(n_step_Z) + "\r" + "\n";
                port.Write(comand_Z);
                Thread.Sleep(200);
                String answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand_Z);
                    Thread.Sleep(200);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                        return;
                    }
                }
                textBox4.Text = textBox7.Text;
            }
            else
            {
                MessageBox.Show("Неправильно заданное значение по
оси Z");
            }

        }

        private  void  Дискретный_режим_FormClosed(object  sender,
FormClosedEventArgs e)
        {
            Form1 F1 = new Form1();
            port.Close();
            F1.textBox1.Text = this.textBox2.Text;
            F1.textBox2.Text = this.textBox3.Text;
            F1.textBox3.Text = this.textBox11.Text;
            F1.textBox4.Text = this.textBox9.Text;
            F1.Visible = true;
        }
        private void button4_Click(object sender, EventArgs e)
        {
            port.Write("ATSX=0\r\n");
            Thread.Sleep(100);
            port.Write("ATSY=0\r\n");
            Thread.Sleep(100);
            port.Write("ATSZ=0\r\n");
```

```
        }
    }
}
```

Листинг программы настроек

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;   //Подключение класса для работы с COM
портом

namespace WindowsFormsApplication4
{
    public partial class Настройки : Form
    {
        double h_X, phi_X;
        int step_X, K_X, delitel_X, rotation_frequency_X;
        char[] retention = { '0', '0', '0', '0', '0', '0', '0',
'0' };
        double h_Y, phi_Y;
        int step_Y, K_Y, delitel_Y, rotation_frequency_Y;
        double phi_Z;
        int step_Z, K_Z, delitel_Z, rotation_frequency_Z;
        double phi_G;
        int step_G, delitel_G, rotation_frequency_G;
        SerialPort port;//Создание объекта класса
        public Настройки()
        {
            InitializeComponent();
            Init();//Вызов метода init
        }
        private void Init()
        {
            port = new SerialPort();
            port.BaudRate = 9600;
            port.PortName = "COM3";
            port.Open();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            if ((textBox5.Text == "") || (textBox2.Text == "") ||
(textBox3.Text == "") ||
                (textBox11.Text == "") || (textBox6.Text == "") ||
(textBox4.Text == "") ||
```

```
                (textBox10.Text == "") || (textBox9.Text == "") ||
(textBox8.Text == "") ||
                (textBox7.Text == "") || (!radioButton16.Checked)
&& (!radioButton15.Checked) &&
                (!radioButton14.Checked)                          &&
(!radioButton13.Checked) || (!radioButton12.Checked) &&
                (!radioButton11.Checked)                          &&
(!radioButton10.Checked) && (!radioButton9.Checked) ||
                (!radioButton8.Checked)                           &&
(!radioButton7.Checked) && (!radioButton6.Checked) &&
                (!radioButton2.Checked)                           ||
(!radioButton4.Checked) && (!radioButton3.Checked) &&
                (!radioButton5.Checked)                           &&
(!radioButton1.Checked))
            {
                MessageBox.Show("Заполните все поля настройки!",
"Предупреждение",                          MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else
            {
                /*----------Ось_X--------------------------*/
                h_X = Convert.ToDouble(textBox5.Text);
                phi_X = Convert.ToDouble(textBox2.Text);
                rotation_frequency_X                              =
Convert.ToInt32(textBox3.Text);
                if (radioButton16.Checked) step_X= 1;
                if (radioButton14.Checked) step_X= 2;
                if (radioButton15.Checked) step_X= 8;
                if (radioButton13.Checked) step_X= 16;
                K_X =Convert.ToInt32(360 / (h_X * phi_X))*step_X;
                delitel_X     =     Convert.ToInt32(60000     /
(rotation_frequency_X*step_X*0.25*(360/phi_X)));
                String comand = "ATDX=" + delitel_X + "\r\n";
                port.Write(comand);
                Thread.Sleep(150);
                String answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand);
                    Thread.Sleep(150);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                    }
                }
                if(checkBox17.Checked==true) retention[7]='1';
                else retention[7] = '0';
                /*----------Ось_Y--------------------------*/
                h_Y = Convert.ToDouble(textBox11.Text);
```

```
                phi_Y = Convert.ToDouble(textBox6.Text);
                rotation_frequency_Y                                =
Convert.ToInt32(textBox4.Text);
                if (radioButton8.Checked) step_Y = 1;
                if (radioButton6.Checked) step_Y = 2;
                if (radioButton7.Checked) step_Y = 8;
                if (radioButton2.Checked) step_Y = 16;
                K_Y  =  Convert.ToInt32(360  /  (h_Y  *  phi_Y))  *
step_Y;
                delitel_Y     =     Convert.ToInt32(60000     /
(rotation_frequency_Y * step_Y * 0.25 * (360 / phi_Y)));
                comand = "ATDY=" + delitel_Y + "\r\n";
                port.Write(comand);
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand);
                    Thread.Sleep(150);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                    }
                }
                if  (checkBox18.Checked  ==  true)  retention[6]  =
'1';
                else retention[6] = '0';
                /*----------Ось_Z--------------------------*/
                phi_Z = Convert.ToDouble(textBox8.Text);
                rotation_frequency_Z                                =
Convert.ToInt32(textBox7.Text);
                if (radioButton1.Checked) step_Z = 1;
                if (radioButton4.Checked) step_Z = 2;
                if (radioButton3.Checked) step_Z = 8;
                if (radioButton5.Checked) step_Z = 16;
                K_Z = step_Z * 56;
                delitel_Z     =     Convert.ToInt32(60000     /
(rotation_frequency_Z * step_Z * 0.25 * (360 / phi_Z)));
                comand = "ATDZ=" + delitel_Y + "\r\n";
                port.Write(comand);
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand);
                    Thread.Sleep(150);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
```

```
                    }
                }
                if (checkBox20.Checked == true) retention[5] =
'1';
                else retention[5] = '0';
                /*----------Схват--------------------------*/
                phi_G = Convert.ToDouble(textBox10.Text);
                rotation_frequency_G                          =
Convert.ToInt32(textBox9.Text);
                if (radioButton12.Checked) step_G = 1;
                if (radioButton10.Checked) step_G = 2;
                if (radioButton11.Checked) step_G = 8;
                if (radioButton9.Checked) step_G = 16;
                delitel_G     =     Convert.ToInt32(60000     /
(rotation_frequency_G * step_G * 0.25 * (360 / phi_G)));
                comand = "ATDG=" + delitel_Y + "\r\n";
                port.Write(comand);
                Thread.Sleep(150);
                answer = port.ReadExisting();
                if (answer != "OK\r\n")
                {
                    port.Write(comand);
                    Thread.Sleep(150);
                    answer = port.ReadExisting();
                    if (answer != "OK\r\n")
                    {
                        MessageBox.Show("Обрыв связи!");
                    }
                }
                if (checkBox19.Checked == true) retention[4] =
'1';
                else retention[4] = '0';
            }
            String ret = new String(retention);
            String com = "ATM=" + ret + "\r\n";
            port.Write(com);
            Thread.Sleep(250);
            String answ = port.ReadExisting();
            if (answ != "OK\r\n")
            {
                port.Write(com);
                Thread.Sleep(250);
                answ = port.ReadExisting();
                if (answ != "OK\r\n")
                {
                    MessageBox.Show("Обрыв связи!");
                }
            }
        }
        private    void    Настройки_FormClosed(object    sender,
FormClosedEventArgs e)
```

```
        {
            Form1 F1 = new Form1();
            port.Close();
            F1.textBox1.Text = Convert.ToString(K_X);
            F1.textBox2.Text = Convert.ToString(K_Y);
            F1.textBox3.Text = Convert.ToString(K_Z);
            F1.textBox4.Text = Convert.ToString(step_G);
            /*---Сохранение настроек----------*/
            Properties.Settings.Default.textBox5                =
this.textBox5.Text;
            Properties.Settings.Default.textBox2                =
this.textBox2.Text;
            Properties.Settings.Default.radioradioButton16=
this.radioButton16.Checked;
            Properties.Settings.Default.radioradioButton15      =
this.radioButton15.Checked;
            Properties.Settings.Default.radioradioButton14      =
this.radioButton14.Checked;
            Properties.Settings.Default.radioradioButton13      =
this.radioButton13.Checked;
            Properties.Settings.Default.textBox3                =
this.textBox3.Text;
            Properties.Settings.Default.checkBoxSave17          =
checkBox17.Checked;

            Properties.Settings.Default.textBox11               =
this.textBox11.Text;
            Properties.Settings.Default.textBox6                =
this.textBox6.Text;
            Properties.Settings.Default.radioradioButton8       =
this.radioButton8.Checked;
            Properties.Settings.Default.radioradioButton7       =
this.radioButton7.Checked;
            Properties.Settings.Default.radioradioButton6       =
this.radioButton6.Checked;
            Properties.Settings.Default.radioradioButton2       =
this.radioButton2.Checked;
            Properties.Settings.Default.textBox4                =
this.textBox4.Text;
            Properties.Settings.Default.checkBoxSave18          =
checkBox18.Checked;

            Properties.Settings.Default.textBox10               =
this.textBox10.Text;
            Properties.Settings.Default.radioradioButton12      =
this.radioButton12.Checked;
            Properties.Settings.Default.radioradioButton11      =
this.radioButton11.Checked;
            Properties.Settings.Default.radioradioButton10      =
this.radioButton10.Checked;
```

```
            Properties.Settings.Default.radioradioButton9       =
this.radioButton9.Checked;
            Properties.Settings.Default.textBox9                =
this.textBox9.Text;
            Properties.Settings.Default.checkBoxSave19          =
checkBox19.Checked;

            Properties.Settings.Default.textBox8                =
this.textBox8.Text;
            Properties.Settings.Default.radioradioButton1       =
this.radioButton1.Checked;
            Properties.Settings.Default.radioradioButton3       =
this.radioButton3.Checked;
            Properties.Settings.Default.radioradioButton4       =
this.radioButton4.Checked;
            Properties.Settings.Default.radioradioButton5       =
this.radioButton5.Checked;
            Properties.Settings.Default.textBox7                =
this.textBox7.Text;
            Properties.Settings.Default.checkBoxSave20          =
checkBox20.Checked;
            Properties.Settings.Default.Save();
            F1.Visible = true;
        }
        private void LoadSettings()
        {
            textBox5.Text = Properties.Settings.Default.textBox5;
            textBox2.Text = Properties.Settings.Default.textBox2;
            radioButton16.Checked                               =
Properties.Settings.Default.radioradioButton16;
            radioButton15.Checked                               =
Properties.Settings.Default.radioradioButton15;
            radioButton14.Checked                               =
Properties.Settings.Default.radioradioButton14;
            radioButton13.Checked                               =
Properties.Settings.Default.radioradioButton13;
            textBox3.Text = Properties.Settings.Default.textBox3;
            checkBox17.Checked                                  =
Properties.Settings.Default.checkBoxSave17;

            textBox11.Text                                      =
Properties.Settings.Default.textBox11;
            textBox6.Text = Properties.Settings.Default.textBox6;
            radioButton8.Checked                                =
Properties.Settings.Default.radioradioButton8;
            radioButton7.Checked                                =
Properties.Settings.Default.radioradioButton7;
            radioButton6.Checked                                =
Properties.Settings.Default.radioradioButton6;
            radioButton2.Checked                                =
Properties.Settings.Default.radioradioButton2;
```

```
            textBox4.Text = Properties.Settings.Default.textBox4;
            checkBox18.Checked                                    =
Properties.Settings.Default.checkBoxSave18;

            textBox10.Text                                        =
Properties.Settings.Default.textBox10;
            radioButton12.Checked                                 =
Properties.Settings.Default.radioradioButton12;
            radioButton11.Checked                                 =
Properties.Settings.Default.radioradioButton11;
            radioButton10.Checked                                 =
Properties.Settings.Default.radioradioButton10;
            radioButton9.Checked                                  =
Properties.Settings.Default.radioradioButton9;
            textBox9.Text = Properties.Settings.Default.textBox9;
            checkBox19.Checked                                    =
Properties.Settings.Default.checkBoxSave19;

            textBox8.Text = Properties.Settings.Default.textBox8;
            radioButton1.Checked                                  =
Properties.Settings.Default.radioradioButton1;
            radioButton3.Checked                                  =
Properties.Settings.Default.radioradioButton3;
            radioButton4.Checked                                  =
Properties.Settings.Default.radioradioButton4;
            radioButton5.Checked                                  =
Properties.Settings.Default.radioradioButton5;
            textBox7.Text = Properties.Settings.Default.textBox7;
            checkBox20.Checked                                    =
Properties.Settings.Default.checkBoxSave20;
        }

        private void Настройки_Load(object sender, EventArgs e)
        {
            this.LoadSettings();
        }
    }
}
```

Листинг программы меню

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication4
```

```
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private  void  ручнойРежимToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Ручной_режим Ручной = new Ручной_режим();
            Ручной.textBox1.Text = this.textBox1.Text;
            Ручной.textBox2.Text = this.textBox2.Text;
            Ручной.textBox3.Text = this.textBox3.Text;
            Ручной.textBox4.Text = this.textBox4.Text;
            this.Visible = false;
            Ручной.ShowDialog();
        }

        private                                              void
дискретныйРежимToolStripMenuItem_Click(object  sender,  EventArgs
e)
        {
            Дискретный_режим Дискретный = new Дискретный_режим();
            Дискретный.textBox2.Text = this.textBox1.Text;
            Дискретный.textBox3.Text = this.textBox2.Text;
            Дискретный.textBox11.Text = this.textBox3.Text;
            Дискретный.textBox9.Text = this.textBox4.Text;
            this.Visible = false;
            Дискретный.ShowDialog();
        }

        private                                              void
позиционныйРежимToolStripMenuItem_Click(object  sender,  EventArgs
e)
        {
            Позиционный_режим       Позиционный       =       new
Позиционный_режим();
            Позиционный.ShowDialog();
        }

        private   void   настройкиToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Настройки настройки = new Настройки();
            this.Visible = false;
            настройки.ShowDialog();
        }

        private     void     Form1_FormClosed(object     sender,
FormClosedEventArgs e)
```

```
            {
              System.Diagnostics.Process.GetCurrentProcess().Kill();
            }
        }
    }
```

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Амурский государственный университет»
(ФГБОУ ВО «АмГУ»)

ПРИКАЗ

11.10.2019 г. Благовещенск № 291-ОД

О результатах конкурса студенческих грантов АмГУ

На основании решения научно-технического совета АмГУ от 10.10.2019, протокол № 1

п р и к а з ы в а ю:

1. Утвердить список студентов, выигравших студенческие гранты на 2019-2020 учебный год со сроком реализации 8 месяцев (с 01.10.2019 по 31.05.2020), с общим объемом финансирования одного гранта 20000 руб. (2500 руб. в месяц):

- факультет математики и информатики:
Афанасов Леонид Сергеевич, 852-ом гр.;
Демьяненко Александр Евгеньевич, 857-ом гр.;
Колтунов Николай Сергеевич, 852-ом гр.;
Поправка Светлана Тимофеевна, 952-ом гр.;
Юшкевич Полина Андреевна, 852-ом гр.;

- факультет дизайна и технологии:
Грищенко Анастасия Евгеньевна, 486-ос гр.;
Каньшина Юлия Витальевна, 682-об гр.;
Тимошенко Анна Владимировна, 682-об гр.;

- энергетический факультет:
Лисогурская Лидия Николаевна, Лисогурский Иван Александрович, 842-ом1 гр. (совместный проект);
Потемкин Михаил Сергеевич, 641-об гр.;
Сазонова Наталья Евгеньевна, 742-об1 гр.;

- филологический факультет:
Колесников Семен Витальевич, 994-ом гр.;
Филимонов Антон Владимирович, 698-об(2) гр.;

122

- **факультет международных отношений:**
  Аганина Елена Дмитриевна, 730-об гр.;
  Кирпикова Виталия Вадимовна, 631-об(2) гр.;
  Копысов Максим Александрович, 831-ом гр.;
  Кучерявый Александр Юрьевич, 631-об(1) гр.;
  Мулер Варвара Евгеньевна, 631-об(1) гр.;
  Нестерова Екатерина Петровна, 831-ом гр.;
  Савченко Иван Сергеевич, 930-ом гр.;

- **факультет социальных наук:**
  Ефремова Анастасия Евгеньевна, 661-об гр.;
  Казанцева Мелания Олеговна, 566-ос гр.;
  Кальницкая Янина Владимировна, 861-ом гр.;
  Конфедератова Лилия Станиславовна, 761-об гр.;
  Косицына Дарья Сергеевна, 566-ос гр.;
  Лепцан Василий Романович, 762-об гр.;

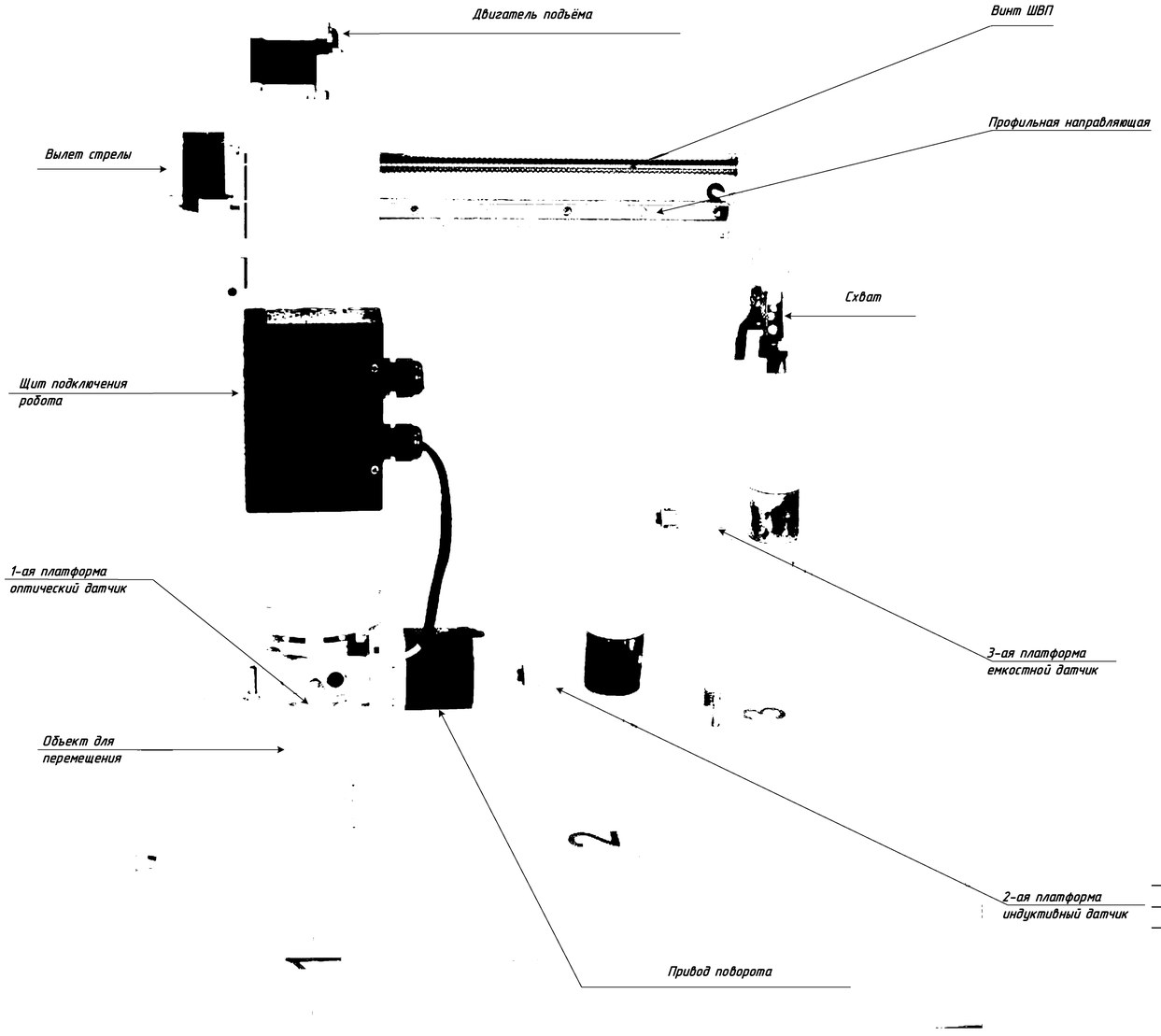
- **экономический факультет:**
  Ликай Виктория Константиновна, 673-об1 гр.;
  Орехова Анна Александровна, 871-озм гр.;
  Петренко Екатерина Андреевна, 671-об гр.;
  Пустовая Ольга Сергеевна, 672-об гр.;
  Сиренко Любовь Юрьевна, 578-ос гр.

2. Контроль за исполнением приказа оставляю за собой.

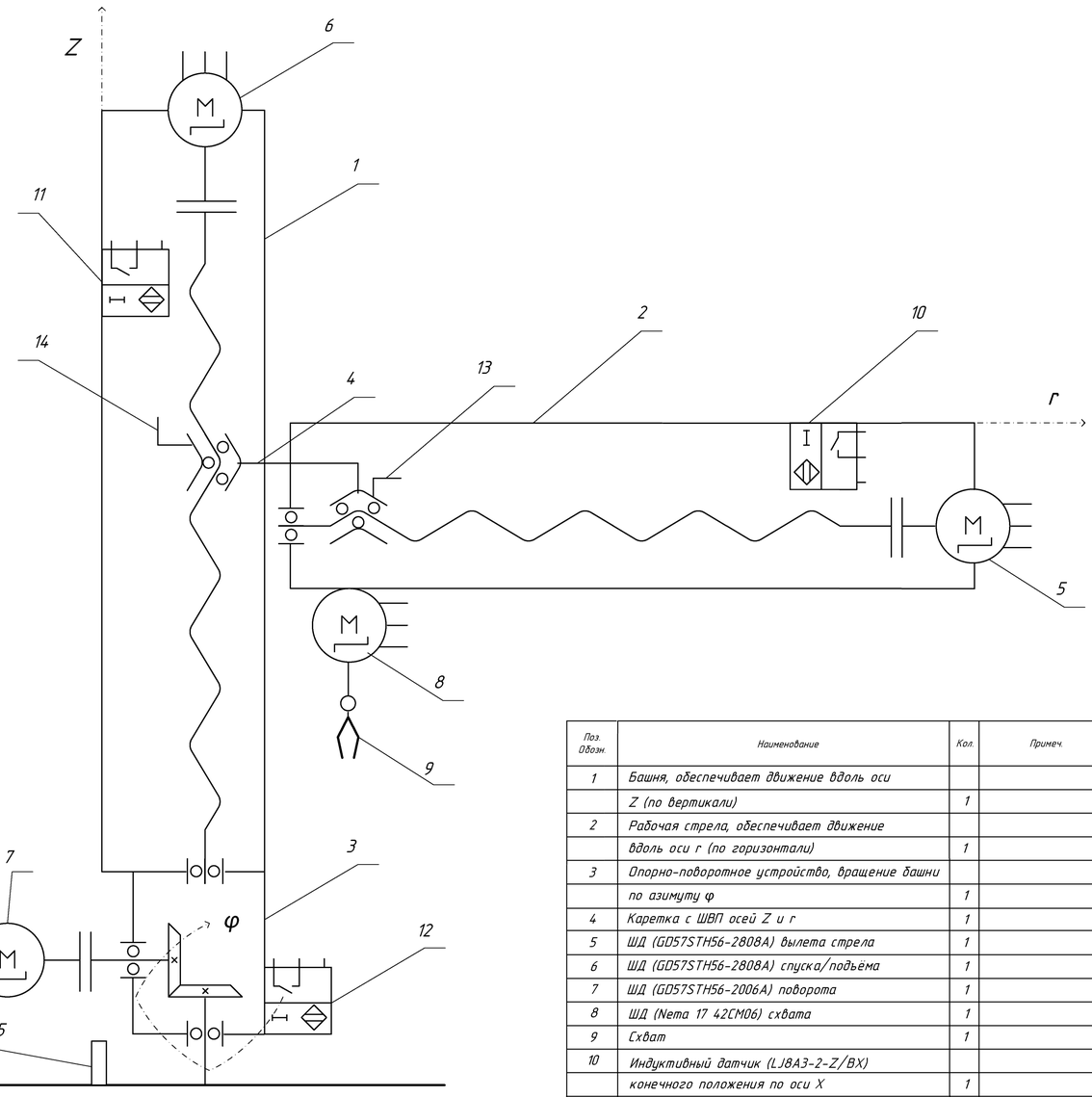
Врио ректора А.В. Лейфа

ПРИЛОЖЕНИЕ 1

1. Внешний вид «УР-4»



2. Кинематическая схема робота



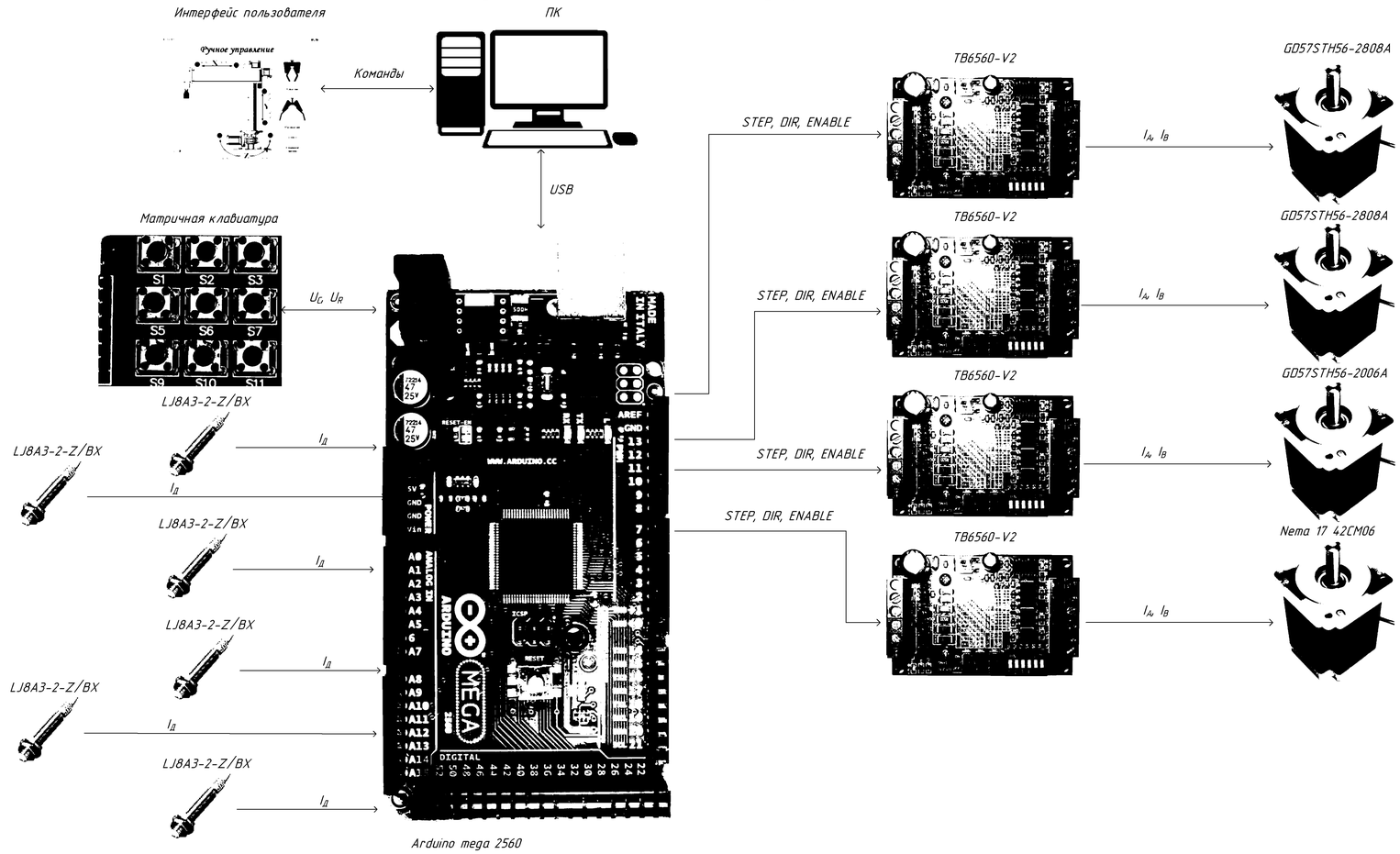
Поз. Обозн.	Наименование	Кол.	Примеч.
1	Башня, обеспечивает движение вдоль оси		
	Z (по вертикали)	1	
2	Рабочая стрела, обеспечивает движение		
	вдоль оси г (по горизонтали)	1	
3	Опорно-поворотное устройство, вращение башни		
	по азимуту φ	1	
4	Каретка с ШВП осей Z и г	1	
5	ШД (GD57STH56-2808A) вылета стрела	1	
6	ШД (GD57STH56-2808A) спуска/подъёма	1	
7	ШД (GD57STH56-2006A) поворота	1	
8	ШД (Nema 17 42CM06) схвата	1	
9	Схват	1	
10	Индуктивный датчик (LJ8A3-2-Z/BX)		
	конечного положения по оси X	1	
11	Индуктивный датчик (LJ8A3-2-Z/BX)		
	конечного положения по оси Y	1	
12	Индуктивный датчик (LJ8A3-2-Z/BX)		
	конечного положения по оси Z	1	
13	Флажок конечного положения по оси X	1	
14	Флажок конечного положения по оси Y	1	
15	Флажок конечного положения по оси Z	1	

					ВКР.164016.15.03.04.СХ	
Изм.	Лист	№ докум.	Подп.	Дата		
Разраб.		Потемкин М.С.			Робот-манипулятор «УР-4».	Литера: У; Масса: ; Масштаб:
Провер.		Скрипко О.В.				Лист 1; Листов 6
Т.Контр.		Скрипко О.В.				
Н.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641
Утвержд.		Скрипко О.В.				

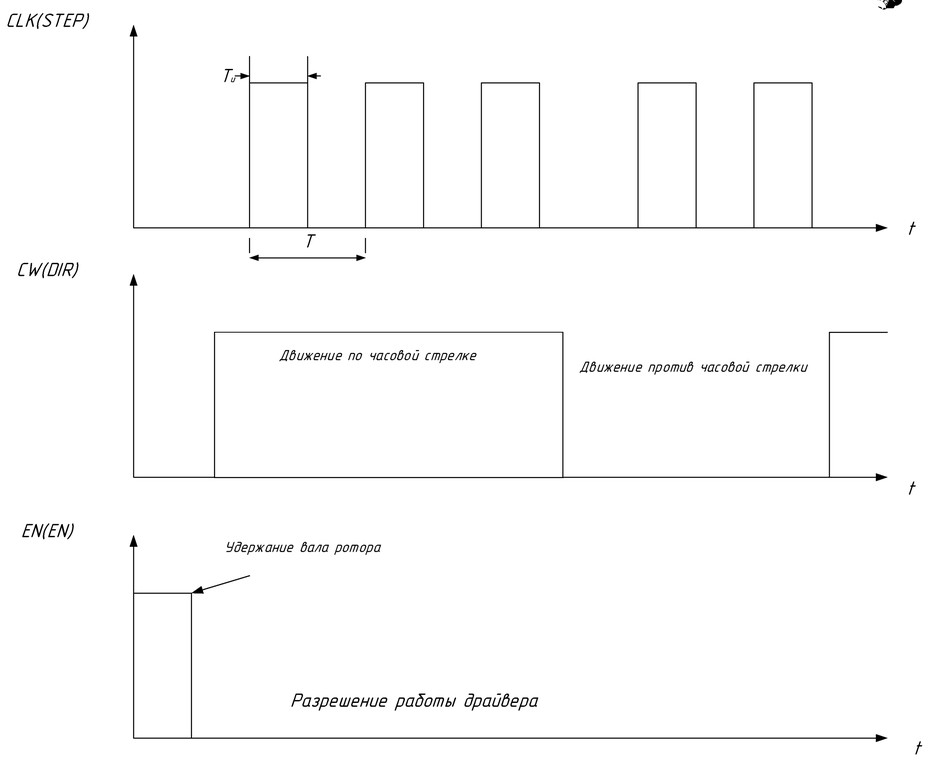
1. Список команд текстового протокола

Команда	Описание
- ATSX - ATSY - ATSZ - ATSG	Данная команда передает число шагов по осям X, Y, Z и схвата методу step() класса StepDirDriver. Команда представляет последовательную запись символов («ATSX=» , n_steps, 13, 10), где n_steps заданное число шагов.
- ATM	Команда задает режим фиксации вала двигателя при остановке по осям X, Y, Z и схвата методу setMode() класса StepDirDriver. Команда представляет последовательную запись символов («ATM=», 00001111, 13, 10), где нулевой разряд двоичного числа соответствует управляющему сигналу по оси X, первый разряд по оси Y, второй разряд по оси Z, третий для схвата. 1 – фиксация вала двигателя при остановке, 0 – отсутствие фиксации вала двигателя при остановке;
- ATRX1, ATRX2 - ATRY1, ATRY2 - ATRZ1, ATRZ2 - ATRG1, ATRG2	Команда предназначена для ручного управления осями робота с ПК. Команда представляет последовательную запись символов («ATRX», 1, 13, 10), где «X» – ось X робота, 1 – вращение вала двигателя по часовой стрелке, 2 – вращение вала двигателя против часовой стрелки.
- ATDX - ATDY - ATDZ - ATDG	Данная команда предназначена для настройки частоты переключения фаз двигателей по осям X, Y, Z и схвата. Команда представляет последовательную запись символов («ATDX=», delitel, 13, 10), где delitel делитель частоты коммутации фаз двигателя.
- ATL?	Команда предназначена для опроса состояния концевых датчиков. Команда представляет последовательную запись символов («ATL?», 13, 10).

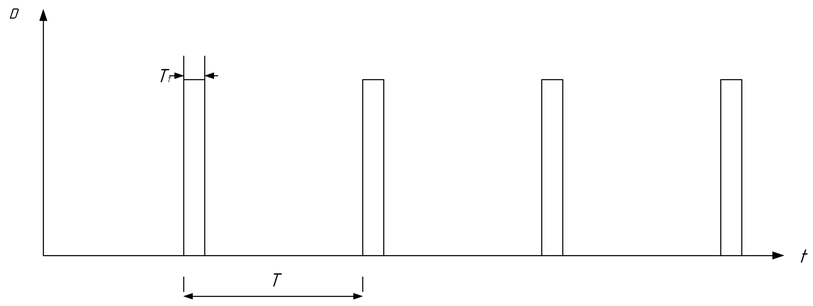
2. Функциональная схема управления роботом



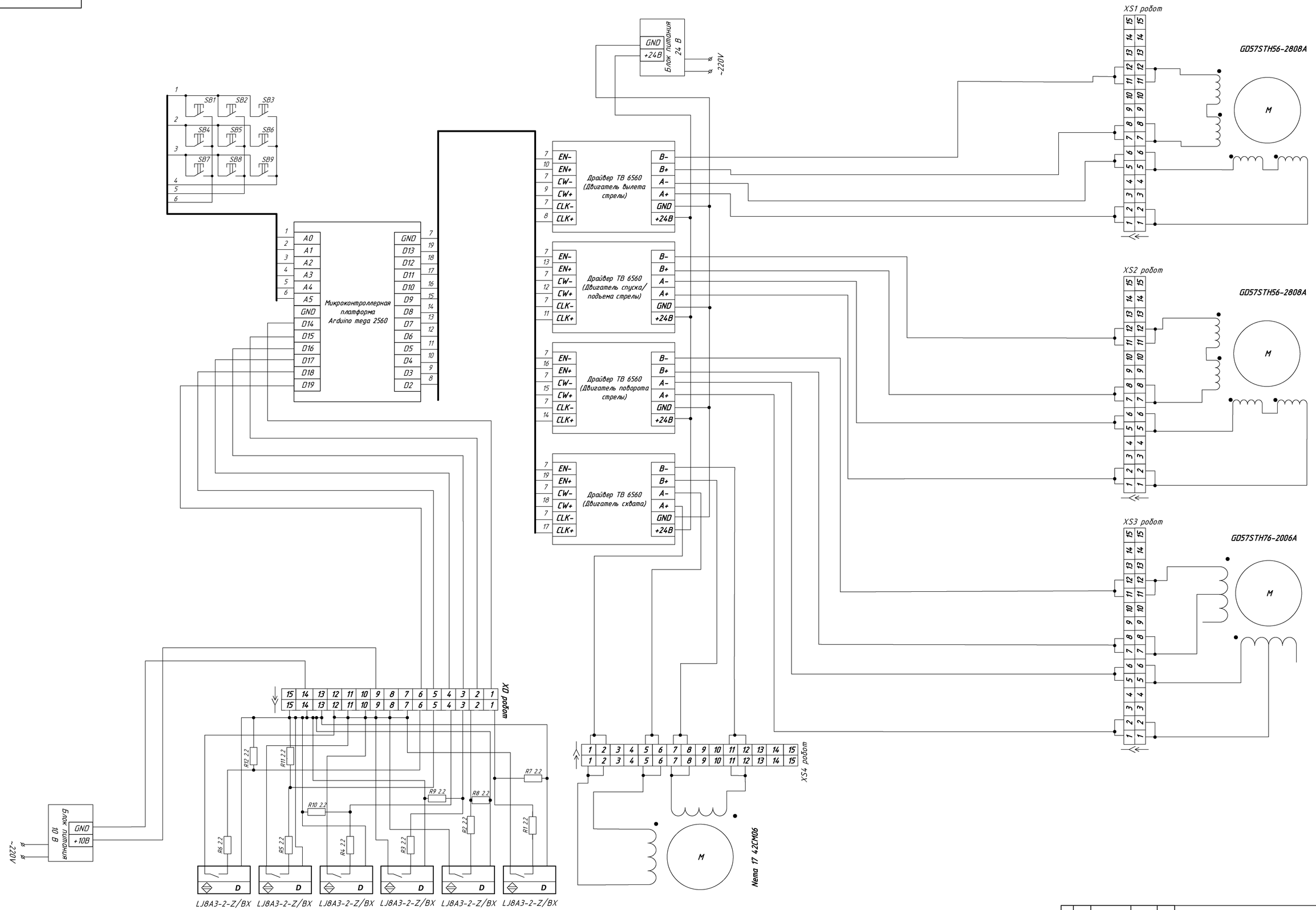
3. Временная диаграмма протокола Step/Dir



4. Временная диаграмма текстового протокола



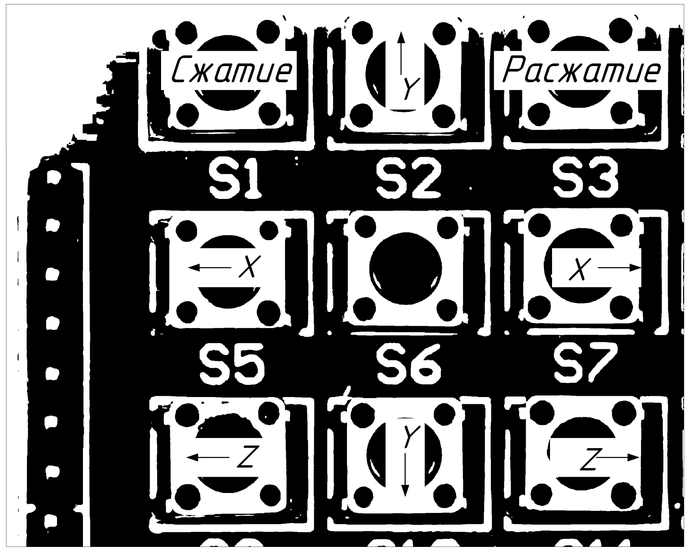
					ВКР.164016.15.03.04.СХ		
Изм.	Лист	№ докум.	Подп.	Дата	Система управления роботом Временные диаграммы	Литера: У	Масса / Масштаб
Разраб.		Потемкин М.С.				Лист 2	Листов 6
Провер.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641	
Т.Контр.		Скрипко О.В.					
Н.Контр.		Скрипко О.В.					
Утвержд.		Скрипко О.В.					



					ВКР.164016.15.03.04.СХ	
Изм.	Лист	№ докум.	Подп.	Дата	Принципиальная схема системы управления роботом	Литера: У; Масса; Масштаб
Разраб.		Потемкин М.С.				Лист 3; Листов 6
Провер.		Скрипко О.В.				
Т.Контр.		Скрипко О.В.				
Н.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641
Утвержд.		Скрипко О.В.				

Изм	Лист	№ докум.	Подп.	Дата	ВКР.164016.15.03.04.СХ		
Разраб.		Потемкин М.С.			Алгоритм работы протокола обмена данными	Литера: у	Масса / Масштаб
Провер.		Скрипко О.В.				Лист 4	Листов 6
Т.Контр.		Скрипко О.В.					
Н.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641	
Утвержд.		Скрипко О.В.					

1. Назначение кнопок на матричной клавиатуре



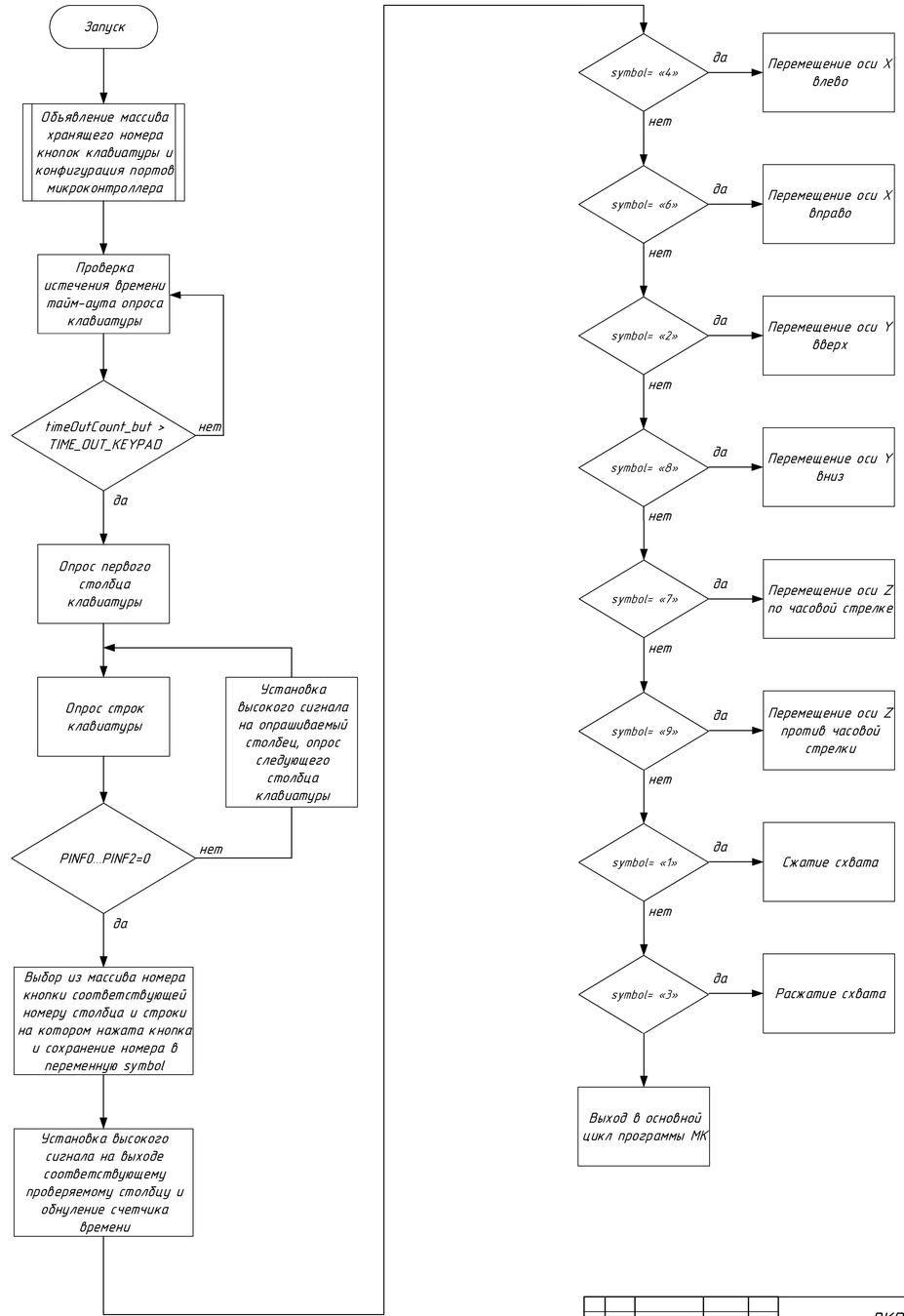
2. Назначение кнопок

«Сжатие» «Расжатие» – управление схватом робота;
«<-X» «X->» – перемещение стрелы робота по оси X;

Y Y – перемещение стрелы робота по оси Y;

«<-Z» «Z->» – поворот робота по оси Z.

3. Алгоритм управления роботом с матричной клавиатуры

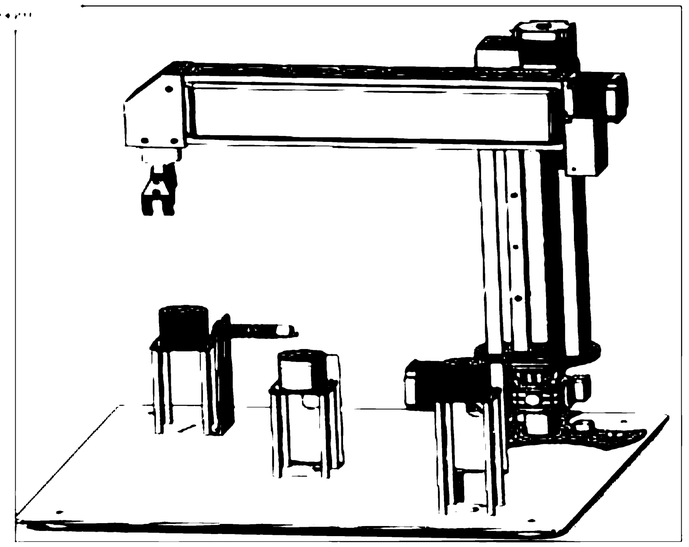


					ВКР.164016.15.03.04.СХ		
Изм	Лист	№ докум.	Подп.	Дата	Ручное управление роботом	Литера: У	Масса:	Масштаб:
Разраб.		Потемкин М.С.						
Провер.		Скрипко О.В.				Лист 5	Листов 6	
Т.Контр.		Скрипко О.В.						
Н.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641		
Утвержд.		Скрипко О.В.						

1. Стартовое окно

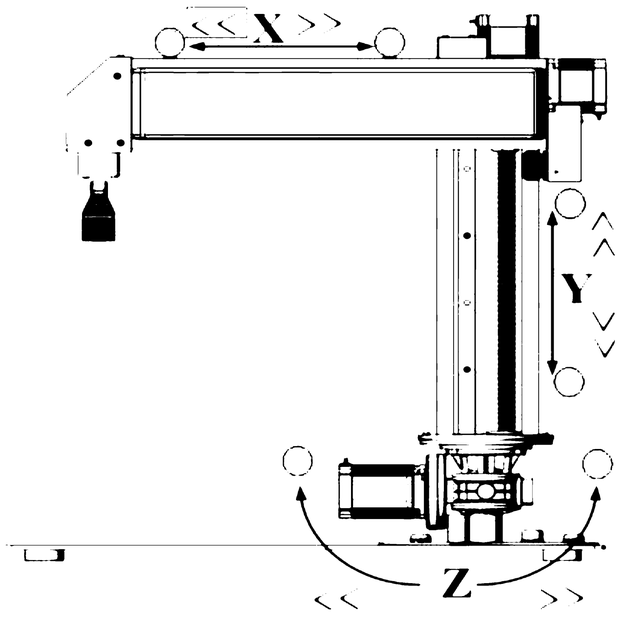
Меню Настройки

Главное меню



2. Окно ручного управления

Ручное управление



Сжатие

Разжатие

>>0<<

3. Окно настроек

Ось X

Шаг резьбы винта

Угол шага двигателя

Режим управления двигателем

Шаговый Полушаговый

1/8 шага 1/16 шага

Частота вращения, об/мин

Фиксация вала двигателя

Ось Y

Шаг резьбы винта

Угол шага двигателя

Режим управления двигателем

Шаговый Полушаговый

1/8 шага 1/16 шага

Частота вращения, об/мин

Фиксация вала двигателя

Схват

Угол шага двигателя

Режим управления двигателем

Шаговый Полушаговый

1/8 шага 1/16 шага

Частота вращения, об/мин

Фиксация вала двигателя

Ось Z

Угол шага двигателя

Режим управления двигателем

Шаговый Полушаговый

1/8 шага 1/16 шага

Частота вращения, об/мин

Фиксация вала двигателя

Установить

4. Окно дискретного режима

Дискретный режим

Положение

Текущее Заданное

Ось X: мм мм

Ось Y: мм мм

Ось Z:

Управление схватом

Текущее Заданное

мм мм

Задать >>0<< Сброс

Остановить

5. Окно предупреждения о некорректной работе в настройках

Ok

6. Окно предупреждения об обрыве связи между МК и ПК

Ok

					ВКР.164016.15.03.04.СХ		
Изм.	Лист	№ докум.	Подп.	Дата	Оконное приложение управления роботом	Литера	Масса	Масштаб
Разраб.		Потемкин М.С.				У		
Провер.		Скрипко О.В.				Лист 6	Листов 6	
Т.Контр.		Скрипко О.В.						
Н.Контр.		Скрипко О.В.			Разработка автоматизированной системы управления роботом манипулятором на базе микроконтроллера AVR	АмГУ, гр. 641		
Утвержд.		Скрипко О.В.						