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

Институт энергетики и электротехники
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование кафедры)

27.03.04 Управление в технических системах
(код и наименование направления подготовки, специальности)

Системы и технические средства автоматизации и управления
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка программного обеспечения лабораторного стенда «Автоматизированная складская система»

Студент	С.В. Ерусланов	
	(И.О. Фамилия)	(личная подпись)
Руководитель	Е.С. Глибин.	
	(И.О. Фамилия)	(личная подпись)
Консультант	Н.В. Ященко.	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.А. Шевцов
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2018

Аннотация

Название бакалаврской работы - Разработка программного обеспечения лабораторного стенда «Автоматизированная складская система»

Целью работы является разработка программного обеспечения для лабораторного стенда для автоматизации процесса хранения.

Бакалаврская работа посвящена обзору микроконтроллеров, поиску наиболее подходящего контроллера и написанию программы для лабораторного стенда, которая должна управлять несколькими датчиками, сервомоторами, шаговыми двигателями и двигателями постоянного тока.

Начнем с постановки задачи, а затем логично перейдем к ее возможным решениям. Мы подробно изучили существующие микроконтроллеры для создания лабораторного стенда для обучения в университете. Мы проанализировали их преимущества и недостатки.

В целом, результаты показывают, что вычислительная платформа Arduino подходит для нашего проекта, который широко используется в учебных заведениях для обучения программированию.

Мы разработали циклограмму стенда и схему выводов микроконтроллера и создали блок-схему программы. В бакалаврской работе была разработана программа на языке программирования C в Arduino.

В заключение я хотел бы подчеркнуть, что все задачи разработки программного обеспечения и выбора микроконтроллера для лабораторного стенда завершена бакалаврская работа выполнена на 54 страницах машинописного текста, и графическая часть в объеме 6 листов А1.

Abstract

The title of the graduation work is the Development of the Laboratory Stand for Software. "Automated Warehouse System".

The aim of the work is development of the laboratory stand software for automation of the storage process.

The graduation work is devoted to the review of microcontrollers, searching for the most suitable controller and writing a program for the laboratory stand, which must control several sensors, servo motors, stepper motors and DC motors.

We start with the statement of the problem and then logically pass over to its possible solutions. We study in detail the existing microcontrollers for the creation of a laboratory stand for studying at university. We analyze their advantages and disadvantages.

Overall, the results suggest that the Arduino computing platform is suitable for our project, which is widely used in educational institutions for programming training.

We have developed a timing diagram of the stand and a circuit of conclusions of the microcontroller and have created the block diagram of the program. In the graduation work, a program was developed in the programming language C in Arduino.

In conclusion, I would like to emphasize that all the tasks of developing software and choosing a microcontroller for a laboratory bench have been completed.

Содержание

Введение

Название бакалаврской работы - Разработка программного обеспечения лабораторного стенда «Автоматизированная складская система»

Целью работы является разработка программного обеспечения для лабораторного стенда для автоматизации процесса хранения.

Бакалаврская работа посвящена обзору микроконтроллеров, поиску наиболее подходящего контроллера и написанию программы для лабораторного стенда, которая должна управлять несколькими датчиками, сервомоторами, шаговыми двигателями и двигателями постоянного тока.

Начнем с постановки задачи, а затем логично перейдем к ее возможным решениям. Мы подробно изучили существующие микроконтроллеры для создания лабораторного стенда для обучения в университете. Мы проанализировали их преимущества и недостатки.

В целом, результаты показывают, что вычислительная платформа Arduino подходит для нашего проекта, который широко используется в учебных заведениях для обучения программированию.

Мы разработали циклограмму стенда и схему выводов микроконтроллера и создали блок-схему программы. В бакалаврской работе была разработана программа на языке программирования C в Arduino.

В заключение я хотел бы подчеркнуть, что все задачи разработки программного обеспечения и выбора контроллера для управления лабораторным стендом завершены.

1. Общие сведения

Спроектированный лабораторный стенд "Автоматизированная складская система", изображенный на рисунке 1, представляет собой рабочую модель автоматизированной складской системы, которая сортирует детали в зависимости от цвета. Установка предназначена для наглядного представления практической реализации возможностей современных контроллеров и привлечения абитуриентов на технические специальности.

Рисунок 1 - Лабораторный стенд «Автоматизированная складская система»

Данный стенд состоит из робота-манипулятора рисунок 5.1, транспортной ленты рисунок 5.4.1, барабан-склада

Система управления состоит из контроллера, который в свою очередь управляет различными элементами стенда: двигателями (серво, шаговый, постоянного тока), оптической парой, датчиком цвета

2. Обзор и выбор контролера

Для разработки управляющей программы автоматической складской системы. Была выбрана среда разработки Arduino IDE. С помощью программной среды Arduino IDE, можно, основываясь лишь на знаниях языка C++, решать самые разные творческие задачи, связанные с программированием.

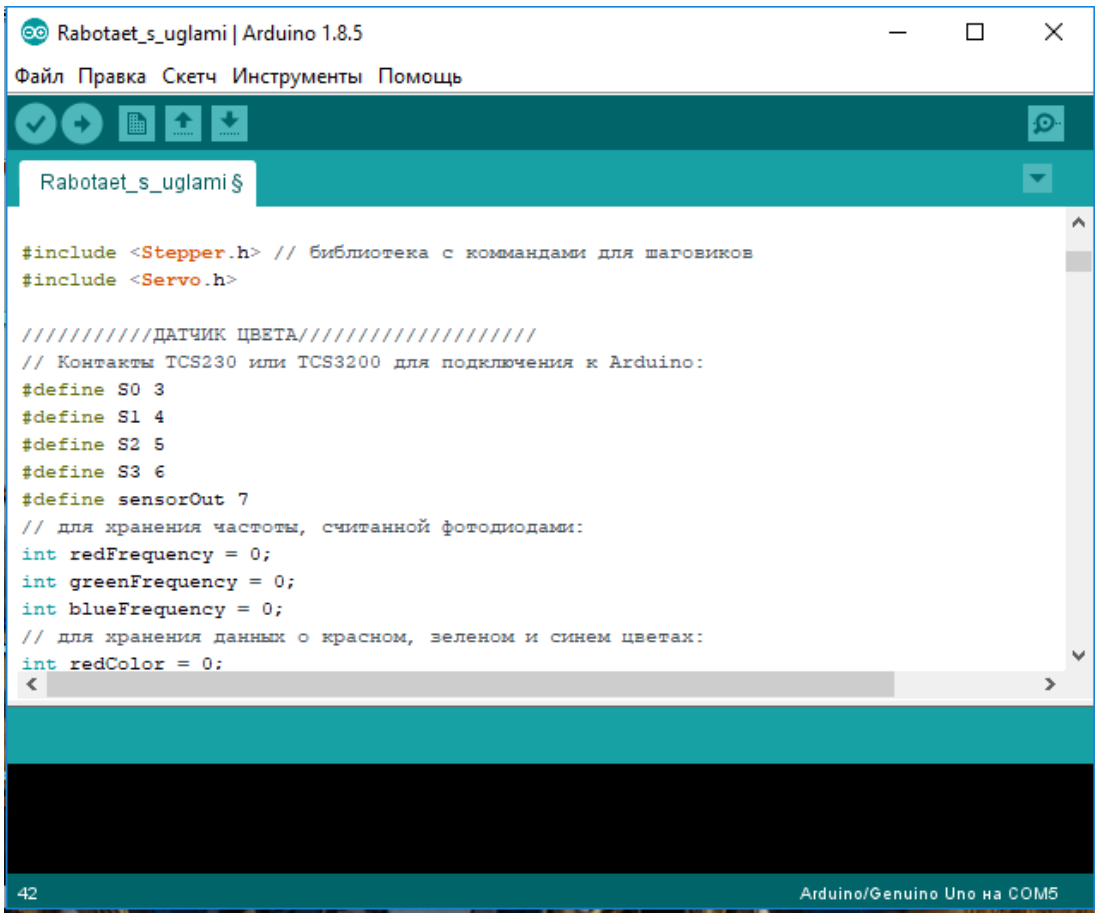


Рисунок 2.1– Окно разработки Arduino IDE.

Благодаря Arduino IDE можно легко запрограммировать одноименные контроллеры. На сегодняшний день с помощью Arduino конструируют всевозможные обучающие, интерактивные, экспериментальные, развлекательные проекты.

Типичный контроллер Arduino сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути,

это однокристальный компьютер, способный выполнять относительно простые задачи.

Отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами. Рассмотрим и сравним основные наиболее популярные микроконтроллеры

Arduino Pro Mini – рисунок 2.2, самая маленькая плата в серии Arduino, размеры которой составляют 1,8x3,3 см. Процессор ATmega168 с тактовой частотой 8 МГц, объемом постоянной памяти 16 КБ и оперативной 1 КБ. Имеет 14 цифровых входов и выходов,6 аналоговых входов

Достоинства:

1) Небольшие размеры, удобен во многих проектах, где мало места для размещения контроллера.
2) Рабочее напряжение питания 3 В, что позволяет питать устройство от двух батареек типа AAA

Недостатки:

1) Нету встроенного программатора, для программирования данного контроллера необходимо использовать внешние программаторы.
2) Маломощный процессор с низкой частотой и малой памятью который не рассчитан на большие вычислительные процессы.

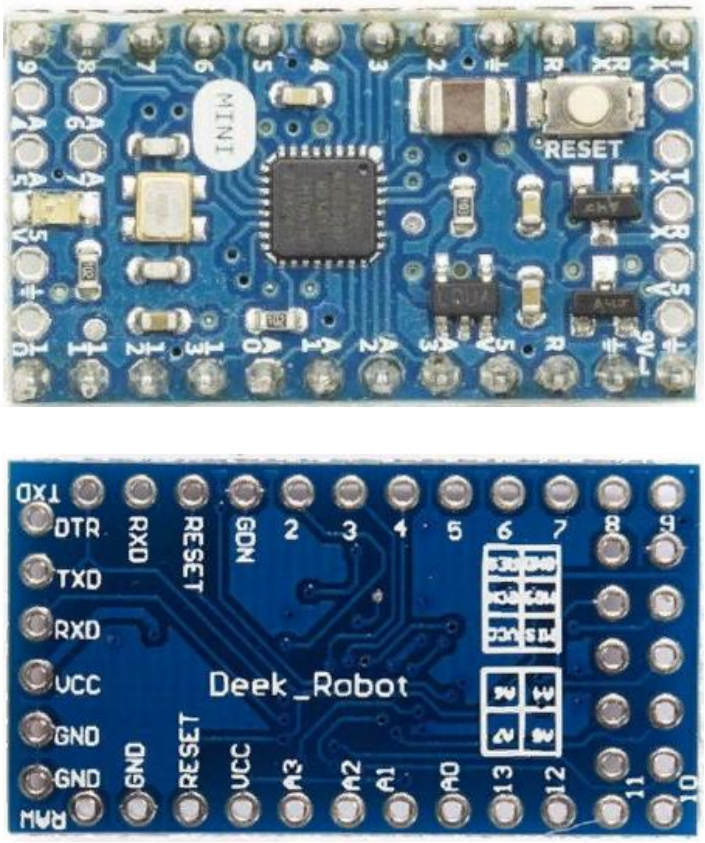


Рисунок 2.2– Arduino Pro Mini.

Arduino Nano 3.0 – рисунок 2.3, еще одна плата с размерами 1.85 см x 4.2 см. Построен на процессоре ATmega328 с тактовой частотой 16 МГц, объемом постоянной памяти 32 КБ и оперативной 2 КБ. Имеет 14 цифровых входов и выходов,8 аналоговых входов. Рабочее напряжение питания 5В. Несмотря на свои размеры данная плата оснащена процессором, который используется в более старших контроллерах. Имеет 8 аналоговых входов, что на 2 больше чем у старшего контроллера Arduino Uno. Так же эта плата уже имеет уже встроенный программатор, который позволяет не использовать внешние устройства для программирования данного контроллера. Данная плата является одним из самых популярных из серии контроллеров Arduino благодаря своей низкой стоимости и небольшим размерам.

Достоинства:

1) Небольшие размеры

2) Встроенный программатор
3) Процессор ATmega328

Недостатки:

1) Ввиду своих размера, установка расширительных плат, таких станет затруднительной. Потребуется макетная плата с дополнительными проводами и модулями.

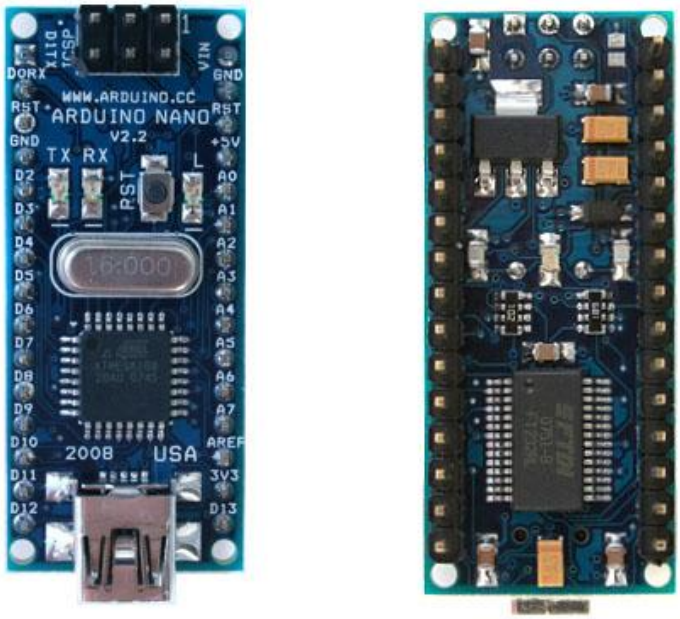


Рисунок 2.3 – Arduino Nano.

Arduino Uno – рисунок 2.4, построена на все том же процессоре ATmega328. Размеры платы 6.9 см x 5.3 см. Рабочее напряжение питания 5В. Имеет 14 цифровых входов и выходов, 6 аналоговых входов. Данная плата поддерживает дополнительные платы расширения.

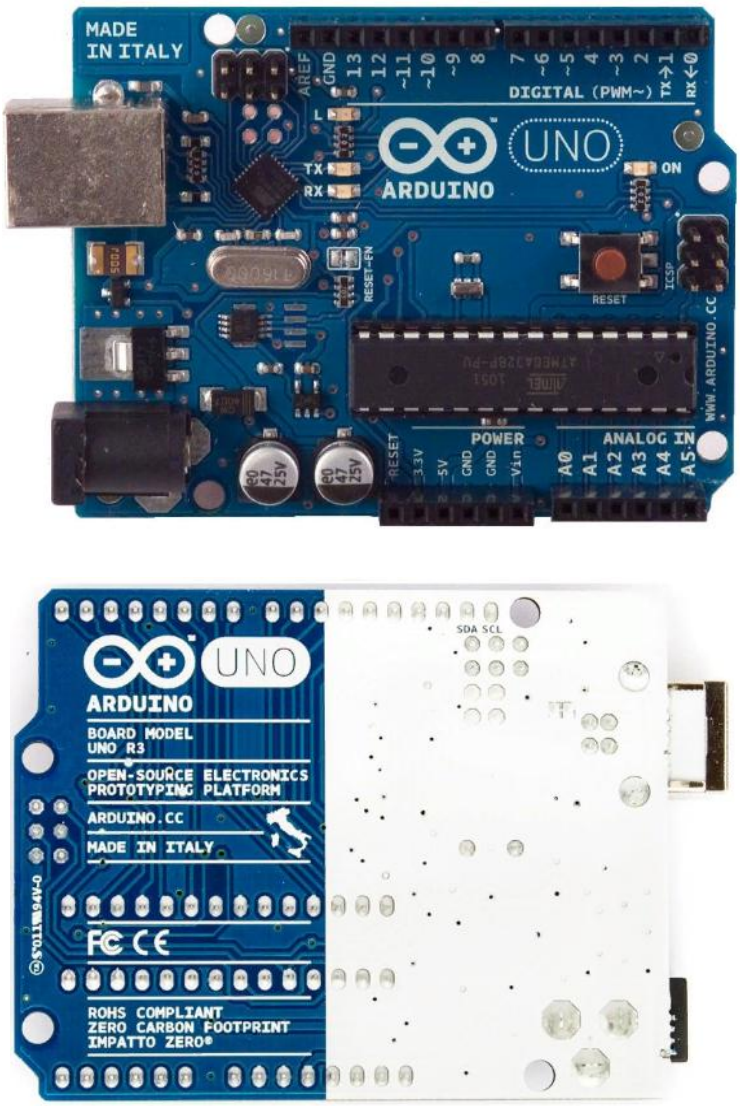


Рисунок 2.4 – Arduino Uno.

Arduino Mega 2560 - Самая большая плата из серии, построена на процессоре ATmega2560, с тактовой частотой 16 МГц, объемом постоянной памяти 256 КБ и оперативной 8 КБ. Имеет 54 цифровых входов и выходов, 16 аналоговых входов. Рабочее напряжение питания 5В.

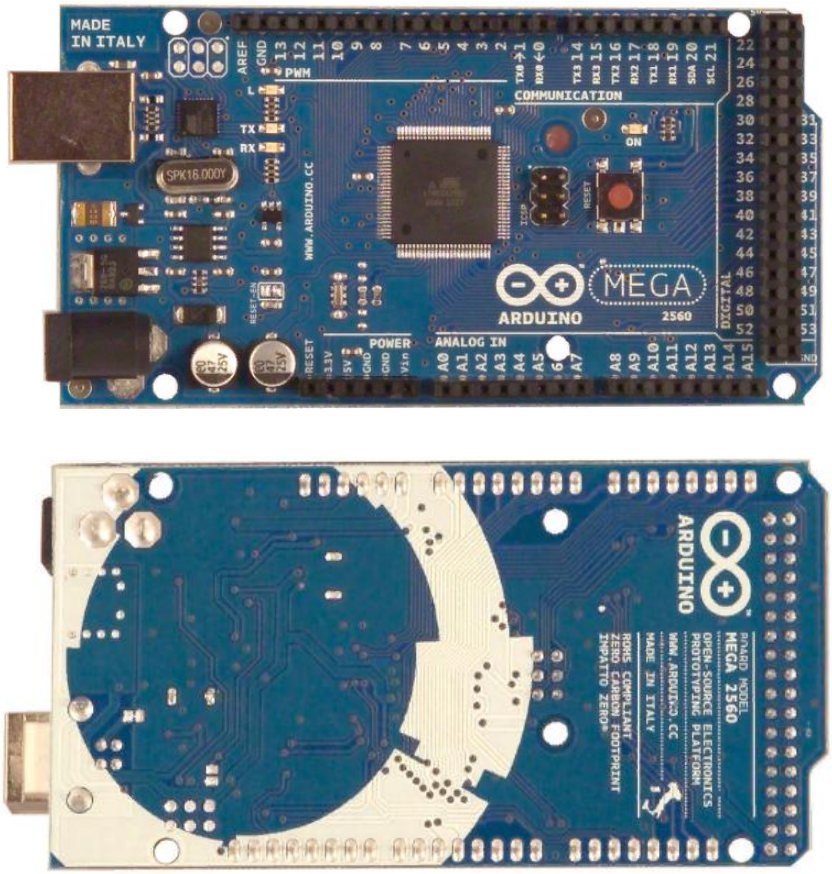


Рисунок 2.5 – Arduino Mega 2560.

Arduino Due – плата реализована на процессоре Cortex− M3. Производительность данного контролера выше чем у остальных предшественников. А по габаритам схожа с Arduino Mega.2560 Тактовая частота процессора 84 МГц, объемом постоянной памяти 512 КБ, оперативной 96 КБ. Плата имеет 54 цифровых входов и выходов, 12 аналоговых входов. Рабочее напряжение питания 5В.

Рисунок 2.6 – Arduino Due.

Стенд использует 1 аналоговый вход для получения информации с оптопары, 15 выходов 4 из которых используются для считывания информации с датчика цвета, 1 для считывания показания с концевого выключателя. 7 цифровых выходов 4 из которых используются для управления сервоприводами Робота-Манипулятора, 4 используются для управления шаговым двигателем, отвечающим за продольное перемещение Манипулятора и 4 используются для управления шаговым двигателем барабан-склада. Исходя из этих данных сравним более подробно микроконтроллеры и выберем наиболее подходящий для нашего проекта

Таблица 1 - Сравнение микроконтроллеров.

Наименование (Микроконтроллер)	Тактовая частота МГц	Объем ПЗУ Кб	Объем ОЗУ Кб	Количество Аналоговых выходов	Количество Цифровых входов/выходов	Стоимость руб.
Arduino Pro Mini (ATmega168)	8	16	1	8 (4 из которых имеют выводы)	14 (6 из которых работают с ШИМ сигналом)	277 □
Arduino Nano 3.0 (ATmega328)	16	32	2	8	14 (6 из которых работают с ШИМ сигналом)	307 □
Arduino Uno (ATmega328)	16	32	2	6	14 (6 которых работают с ШИМ сигналом)	419 □
Arduino Mega 2560 (ATmega2560)	16	256	8	16	54 (14 из которых работают с ШИМ сигналом)	949 □
Arduino Due (AT91SAM3X8E)	84	512	96	12	54 (на 12 из которых работают с ШИМ сигналом)	1290 □

***цены на контролеры приведены с сайта mcustore.ru за май 2018 года.**

Исходя из таблицы 1 нам подходят Arduino Mega 2560 на микропроцессоре ATmega2560 и Arduino Due на микропроцессоре AT91SAM3X8E.

Из данных таблицы 1 видно, что под данное условие наиболее точно попадает микроконтроллер Arduino Mega 2560 за 949 рублей и микроконтроллер Arduino Due за 1290 рублей. Характеристики остальных микропроцессоров считаются недостаточными, поэтому не принимаются к дальнейшему рассмотрению. Так как при решении поставленной задачи разница, будет ли микропроцессор работать на частоте 84 МГц или 16 МГц не является существенной, а вот цена на микроконтроллер Arduino Due в 1,36 раза дороже Arduino Mega, то было принято решение выбрать более дешевый вариант.

3. Разработка циклограммы лабораторного стенда «Автоматизированный складской комплекс»

Рассмотрим циклограмму учебного стенда, описывающую используемые механизмы, движение каждого узла стенда, а также исходное положение оборудования в начале работы.

1. Движения: Для выполнения одного цикла работы стенда «Автоматизированной складской системы» необходимы следующие движения (переходы):
   – движение транспортной ленты;
   – пересечение кубика линии оптопары;
   – остановка работы транспортной ленты;
   – занимание нулевой позиции (позиция 1) манипулятором;
   – перемещение манипулятора в право до нажатия на концевой выключатель;
   – подвод схвата манипулятора к транспортной ленте (позиция 2);
   – сжатие схвата манипулятора;
   – возвращение манипулятора в нулевое положение (позиция 1);
   – перемещение манипулятора к датчику цвета;
   – подвод схвата манипулятора к датчику цвета (позиция 3);
   – определения цвета кубика при помощи датчика цвета;
   – возвращение манипулятора в нулевую позицию (позиция 1);
   – перемещение манипулятора в крайнее левое положение;
   – поворот манипулятора на 180 градусов к барабан-складу;
   – вращение барабан-склада на заданное количество шагов в зависимости ранее определенного цвета;
   – подвод схвата манипулятора к ячейке барабан склада (позиция 4);
   – разжатие схвата манипулятора;

2. В формировании заданного цикла участвуют следующие механизмы:

   Робота-манипулятора

   – схват манипулятора,

   – механизм манипулятора

   – механизм перемещения робота-манипулятора по направляющей;

   Транспортировочной ленты

   – оптическая пара (датчик расстояния),

   – механизм движения транспортной ленты,

   – датчик определения цвета;

   Барабан склада

   – механизм вращения склада.

3. Начальное положение оборудования и его механизмов:

   – кубики находятся на транспортировочной ленте в произвольном порядке;

   – начальное положение манипулятора задается в крайнее левое положение;

   – схват подъемного манипулятора разжат;

   – барабан-склад находится в исходном положении.

На основе этого алгоритма разработана циклограмма работы стенда, изображенная на рисунке 3

Оборудование	Исполнительный механизм	Состояние	Такты														Описание тактов
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Робот манипулятор	Схват	Сжат					+	+	+	+	+	+	+	+			1) Нчальное положение транспортная Робот-манипулятор находится в крайнем левом положении(позиция 1) барабан-склад находится в 0 позиции 2) Кубик персек линию оптопары, схват разжат, манипулятор занял позицию 1 3) Манипулятор перемещается в право к концевому выключателю 4) Концевой выключатель нажат, робот-манипулятор подводит схват к транспортной ленте (позиция 2) 5) Сжимается схват, манипулятор возвращается в позицию 1 6) Манипулятор перемещается к датчику цвета (позиция 3) 7) Манипулятор подводит схват к датчику цвета (позиция 3), включается датчик цвета 8) Манипулятор занимает позицию 1 9) Манипулятор перемещается в крайнее левое положение 10) Манипулятор поворачивается на 180 градусов (позиция 5), барабан-склад занимает позицию в зависимости от цвета кубика определенного ранее 11) Манипулятор подводит схват к ячейке барабан-склада (позиция 4) 12) Схват разжимается 13) Манипулятор занимает позицию 1 14) Начальное положение
		Разжат	+	+	+	+									+	+	
	Механизм робота-манипулятора	Позиция 1		+	+		+	+		+	+				+	+	
		Позиция 2				+											
		Позиция 3							+								
		Позиция 4											+	+			
		Позиция 5										+	+	+			
	Механизм перемещения подъемного манипулятора по направляющей	Позиция 1	+	+							+	+	+	+	+	+	
		Позиция 2			+	+	+										
		Позиция 3						+	+	+							
	Концевой выключатель	Есть сигнал				+	+										
		Нет сигнал	+	+	+			+	+	+	+	+	+	+	+	+	
Транспортная лента	Механизм транспортной ленты	Включен	+														
		Выключен		+	+	+	+	+	+	+	+	+	+	+	+	+	
	Оптрон	Есть сигнал		+	+	+	+										
		Нет сигнала	+					+	+	+	+	+	+	+	+	+	
	Датчик цвета	Включен							+								
		Выключен	+	+	+	+	+	+		+	+	+	+	+	+	+	
Барабан склад	Механизм поворота барабан-склада	Позиция 0	+	+	+	+	+	+	+	+	+					+	
		Позиция 2											+				
		Позиция 3															
		Позиция 4															

Рисунок 3 – Циклограмма работы стенда.

4. Разработка алгоритма и блок-схемы работы учебного стенда.

На основе циклограммы, разработанной в предыдущем разделе, алгоритм работы стенда состоит из следующих пунктов:

1) Начальное положение кубика задается на конвейерной ленте вручную;

2) Производится включение компьютерного блока питания, который в свою очередь питает все элементы стенда. При подключении стенда к сети включаются в работу: контроллер, конвейерная лента, оптопара;

3) Кубик перемещается по транспортной ленте оптопары;

4) По изменению интенсивности светового потока до определенных значений, приходящего на фотоприемник, микроконтроллер Arduino Mega2560 определяет факт прихода кубика и выдает команду для остановки транспортной ленты;

5) Манипулятор занимает нулевое положение и с помощью шагового двигателя перемещается до концевого выключателя;

6) При нажатие концевого выключателя останавливается шаговый двигатель

7) Манипулятор при помощи сервоприводов хватает кубик и возвращается в свое нулевое положение

8) Манипулятор перемещается в лево, на определенное количество оборотов, к датчику цвета и останавливается

9) Манипулятор при помощи сервоприводов подносит кубик к датчику цвета

10) Датчик цвета определяет цвет, и манипулятор возвращается в свое нулевое положение

11) Манипулятор перемещается влево до начального положения и поворачивается к барабан-складу;

12) В зависимости от определенного ранее цвета кубика барабан-склад поворачивается на необходимый угол;

13) Манипулятор при помощи сервоприводов подносит кубик к ячейке барабан-склада и кладет его;

14) Манипулятор возвращается в нулевое положение и поворачивается к транспортной ленте;

15) Цикл продолжается, пока барабан− склад не будет заполнен всеми кубиками;

16) После заполнения барабан− склада всеми кубиками они заново выкладываются вручную на транспортную ленту;

Исходя из определения переменных и алгоритма работы, общая блок−схема работа стенда изображена на рисунке 4.

На основе этой блок схемы был написан код управляющей программы для «Автоматизированной складской системы»

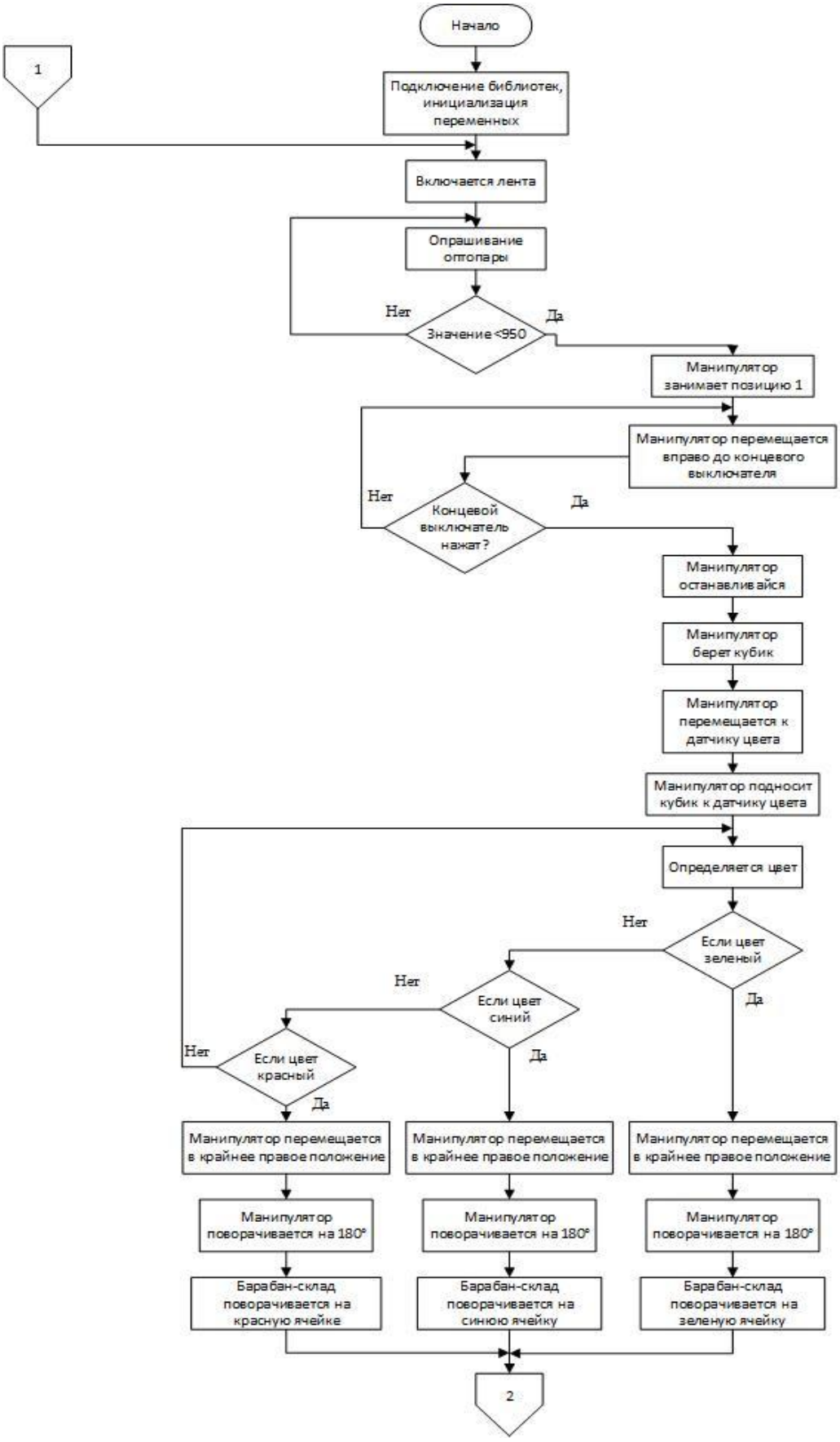
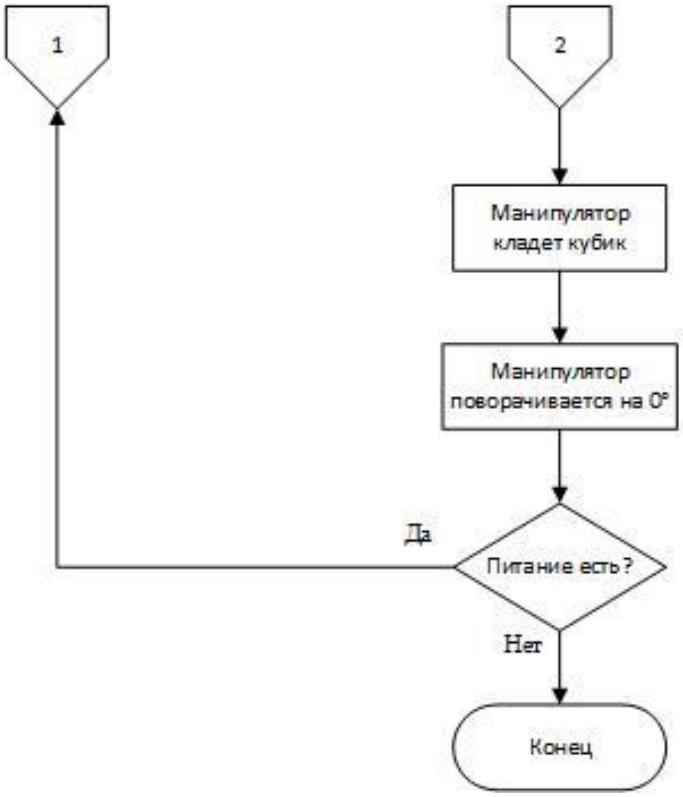


Рисунок 4 - Общая блок-схема работы стенда.



Продолжение рисунка 4 - Общая блок-схема работы стенда.

5. Разработка алгоритма управления стендом

5.1 Робот-манипулятор

Рассмотрим отдельно робот-манипулятор (рисунок 5.1). Он приводится в движение при помощи трех сервомоторов.

Рисунок 5.1– Робот-Манипулятор.

Сервомотор – это привод, вал которого может встать в заданное положение или поддерживать заданную скорость вращения. Другими словами, можно задать скорость и частоту вращения вала сервомотору через контролер.

5.1.1 Подключение сервомотора к Arduino

Сервомотор имеет 3 контакта рисунок 5.1.1.1

Gnd – (черный провод) минусовой контакт, подключается любому GND выходу контроллера или к «-» аккумулятора, если сервопривод питается от аккумулятора.

Vcc – (красный провод) плюсовой контакт, подключается к любому 5в выходу контроллера либо «+» аккумулятора

S – (Белый провод) управляющий провод, подключается к PWM (ШИМ) выходу Arduino

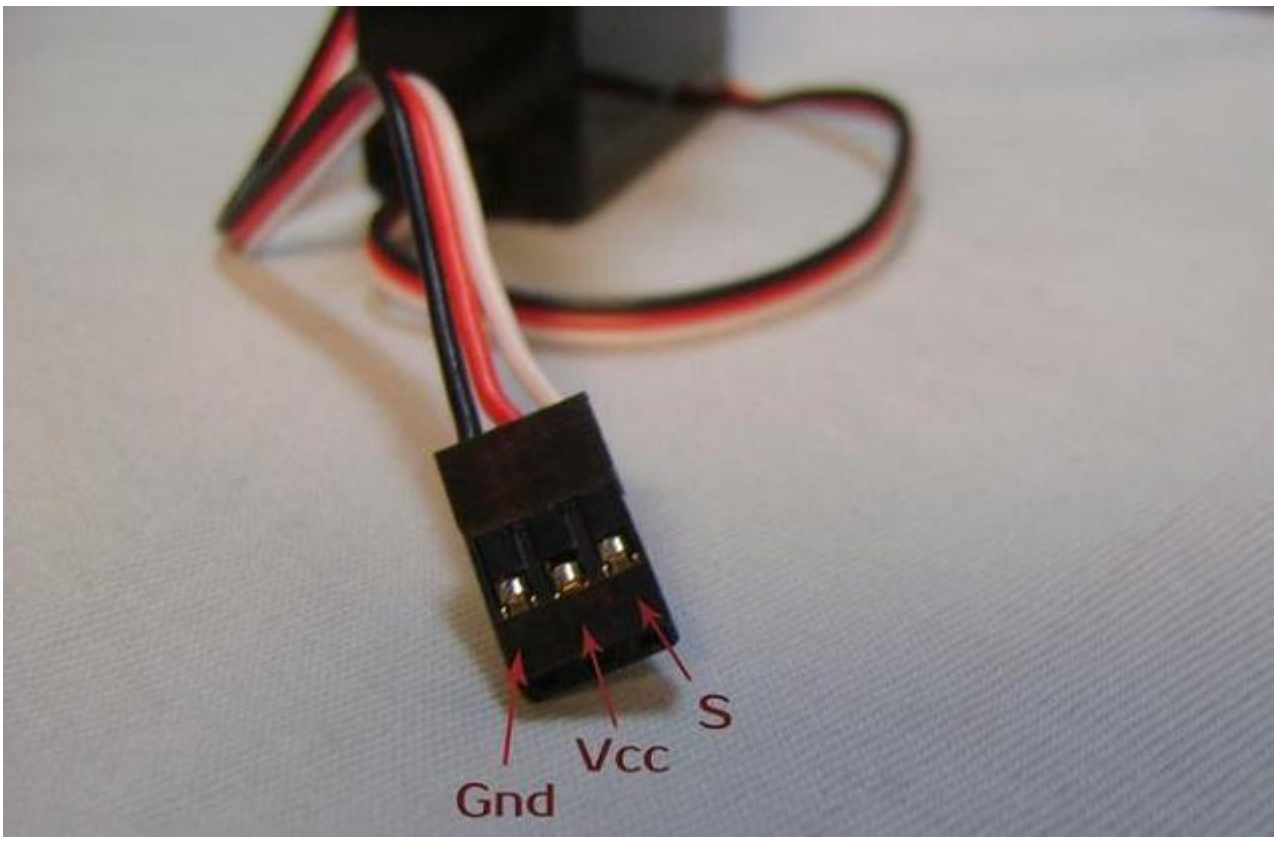


Рисунок 5.1.1.1 – Выводы контактов сервомотора.

Для работы Робота-Манипулятора (рисунок 5.1) необходимо точно настроить работу четырех сервомоторов. Так как сервомоторы не имеют обратной связи для настройки точных углов необходимые для работы робота-манипулятора было решено задавать углы через потенциометры. Для этого мы воспользовались стандартной библиотекой Servo.h

Основные функции библиотеки Servo.h

1. Servo.attach() Подключение библиотеки к заданному пину, с которого происходит управление сервоприводом.

2. Servo.write() Установка угла для поворота вала сервопривода в градусах.

3. Servo.writeMicroseconds() Передача значения для управления сревомотором в микросекундах, устанавливая угол поворота на это значение. Для стандартного сервомотора значения следующие:

1000 – максимальный поворот против часовой стрелки;

1500 – остановка посередине.

2000 – максимальный поворот по часовой стрелке;

4. Servo.read() Считывание значения текущего положения сервомотора (значение установленное последним вызовом функции write()).

5. Servo.attached() Определение есть ли привязка сервомотора к пину. Возвращает true если есть привязка к пину, а в противном случае – false .

6. Servo.detach() Отключение вывода от библиотеки Servo.h В случае отключенных переменных Servo – пины котроллера 9 и 10 можно использовать для ШИМа– вывода через analogWrite().

Ниже приведен пример для управления сервомотором через потенциометр.

Скетч для считывания значений углов с потенциометра.

```
#include <Servo.h>      // подключение библиотеки Servo.h

Servo servo;                // создание объекта servo

int servo_pot = A0;       // аналоговый вход A0 для подключения потенциометра
int val;                        // переменная для хранения значения потенциометра

void setup()
{
Serial.begin(9600);     // открытие последовательного порта
servo.attach(5);          // подключение левого сервопривода к 5 цифровому выходу
 }
```

```
void loop()
{
val = analogRead(servo_pot);                  // считывание значения с потенциометра
(значение от 0 до 1023)
val = map(val, 0, 1023, 0, 180);   // масштабируем значение к интервалу 0-180
levo.write(val);                              // поворот сервопривода на заданный угол
Serial.println(" levo ");
Serial.print(val);                    // вывод на монитор значения угла потенциометра
        }
```

Схема подключения показана ниже на рисунке 5.1.1.2

Таким образом экспериментально было получены углы для четырех позиций манипулятора таблица 5.1.1, которые раннее были показаны на циклограмме работы стенда.

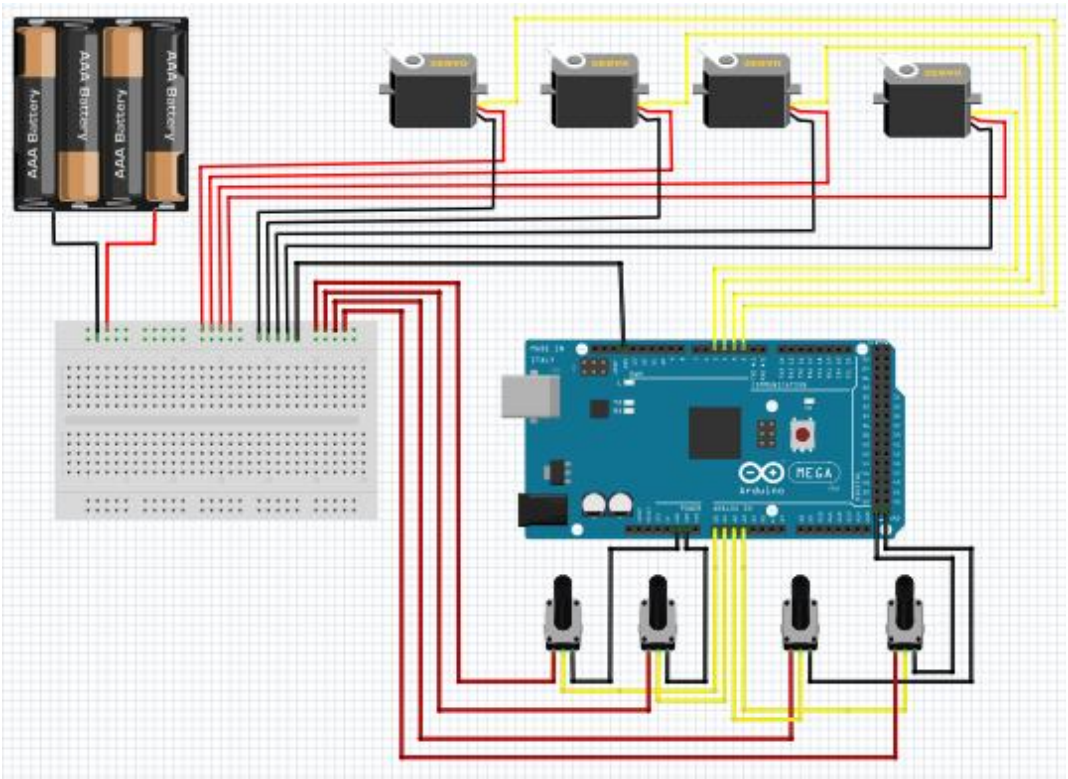


Рисунок 5.1.1.2 - Схема подключения 4 сервомоторов и потенциометров к Arduino.

Таблица 5.1.1 – Значения углов сервомоторов для каждой позиции манипулятора.

Сервомоторы	Позиция 1	Позиция 2	Позиция 3	Позиция 4
Правый	164	87	112	85
Левый	19	54	27	28
Центральный	27	27	27	113
Схват	120	70	70	120

В ходе настройки работы робота-манипулятора выяснилось, что стандартная скорость работы сервомотора оказалась слишком высокой, из-за этого позиционирование робота была с погрешностями. Так как в стандартной библиотеке Servo.h не предусмотрено регулировка скорости работы сервомотора было решено воспользоваться библиотекой VarSpeedServo.h

Особенности данной библиотеки:

- Поддержка до 8 сервоприводов.
- Поддерживает одновременное асинхронное движение всех сервоприводов.
- Можно задавать скорость хода.
- Функция write() инициирует перемещение и при необходимости может дождаться завершения перемещения перед возвращением.
- Можно задать сервомотору последовательность ходов (где каждый ход имеет положение свою скорость)

Основные функции библиотеки Servo.h

1. VarSpeedServo myServo; — создание объекта для серводвигателя.

2. slowmove (newpos, speed) – установка скорости сервомотора.

newpos – установка позиция сервомотора.

speed – скорость сервомотора.

4. myServo.attach (mainPin, ServoMin, ServoMax); - подключение объекта.

myServo к пину контроллера.

mainPin – пин куда подключен сервопривод.

ServoMin – (опционально) ширина импульса в микросекундах устанавливающая положение вала сервомотора в 0 градусов.

ServoMax - (опционально) ширина импульса в микросекундах устанавливающая положение вала сервомотора в 180 градусов.

Ниже, на рисунке 5.1.1.3, показана блок схема взятия кубика манипулятором с транспортной ленты.

Рисунок 5.1.1.3 – Блок схема работы 1 цикла руки-манипулятора.

Ниже приведен пример, с использованием данной библиотеки. В данном примере робот-манипулятор берет кубик с транспортной ленты и возвращается в исходное положение.

Скетч взятия кубика манипулятором с транспортной ленты.

```
#include <VarSpeedServo.h> // подключение библиотеки
////////////ОБЪЯЕВЛЕНИЕ ПЕРЕМЕННЫХ//////////////
VarSpeedServo centr;
VarSpeedServo levo;
VarSpeedServo pravo;
VarSpeedServo gripper;
void setup() {
  //////////ПОДКЛЮЧЕНИЕ СЕРВОПРИВОДОВ//////////
  gripper.attach(14);
  centr.attach(15);
  pravo.attach(16);
  levo.attach(17);
}
void loop()
{
  int SPEED = 40; //переменная для задания скорости работы сервопривода (от 0 до 250)
  //////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1/////////
  pravo.write(164, SPEED);
  delay(500);
  levo.write(19, SPEED);
  delay(500);
  centr.write(0, SPEED);
  delay(500);

  /////// МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 2/////////
```

```
 griffer.write(120);
 delay(500);
 levo.write(54, SPEED);
 delay(500);
 pravo.write(87, SPEED);
 delay(500);
 gripper.write(40, SPEED);
 delay(500);
}
```

5.2 Шаговые двигатели

Далее рассмотрим отдельно работу шаговых двигателей В данном стенде используются 2 шаговых двигателя ЕМ-188 (рисунок 5.2)

Рисунок 5.2 – Шаговый двигатель EM-188.

ШД1 отвечает за поворот барабан-склада. ШД2 отвечает за перемещение робота-манипулятора относительно стола.

ШД1 в зависимости от определенного цвета поворачивается на определенный угол.

ШД2 работает по следующему алгоритму:

- ШД2 работает по часовой стрелке, до тех пор, пока робот-манипулятор не нажал на концевой выключатель
- ШД2 делает 1300 шагов в обратную сторону, тем самым перемещая манипулятор к датчику цвета
- ШД2 делает 700 шагов против часовой стрелки, тем самым возвращает манипулятор в начальное положение

Для управления шаговым двигателем через контроллер, в отличии от сервомотора необходим внешний драйвер.

Для управления шаговыми двигателями мы воспользовались стандартной библиотекой Stepper.h Данная библиотека предоставляет удобный интерфейс для управления униполяными и биполярными шаговыми двигателями.

Функции библиотеки:

1. myStepper(steps, pin 1, pin 2)

2. myStepper(steps, pin 1, pin 2, pin 3, pin 4) Данная функция создает объект класса Stepper, который в свою очередь соотнесен к шаговому двигателю присоединенному к контроллеру Arduino. Переменную класса Stepper рекомендуется объявлять в конструктор вне setup() и loop().

3. myStepper.setSpeed(rpms) Эта функция устанавливает вращения вала двигателя в оборотах в минуту (RPMs – rotations per minute). Однако, она не предназначена для посредственного вращения ШД. Благодаря ей программа при вызове функции step() только сможет распознать установленную скорость вращения.

4. myStepper.step(steps)

Данная функция задает вращение вала шагового двигателя на определенное количество шагов, на скорости, устанавливая скорость функцией setSpeed(). Эта функция блокирующая, то есть пока вращение двигателя не закончится цикл программы не продолжится. Например, если скорость вращения установлена на 2 RPM (2 оборот в минуту), и вызвано step(200) на двигателе с полным оборотом в 200 шагов, то выполнение этой функции займет две минуту. Правильнее чтобы скорость вращения вала была высокой, а один вызов функции step() делалось пару шагов, чтобы избежать длительной блокировки выполнения цикла скетча.

Ниже представлен скетч для выполнения 1 оборота вала шагового двигателя по часовой и против часовой стрелки.

Скетч с использованием библиотеки Stepper.h

```
#include <Stepper.h>
const int stepsPerRevolution = 200;   // количество шагов за 1 оборот вала шагового двигателя
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); // создание объекта myStapper типа Stepper
void setup() {
  myStepper.setSpeed(60);   // установка скорости (60 об/мин)
  }
void loop() {
  myStepper.step(stepsPerRevolution);  // 1 оборот шагового двигателя по часовой стрелке
  delay(500);               // пауза
  myStepper.step(-stepsPerRevolution); // 1 оборот шагового двигателя против часовой стрелке
  delay(500);    }          // пауза
```

5.3 Датчик цвета

Для определения цвета кубиков был выбран датчик цвета TCS3200. Данный датчик преобразует интенсивность цветового спектра в выходной сигнал различной частоты

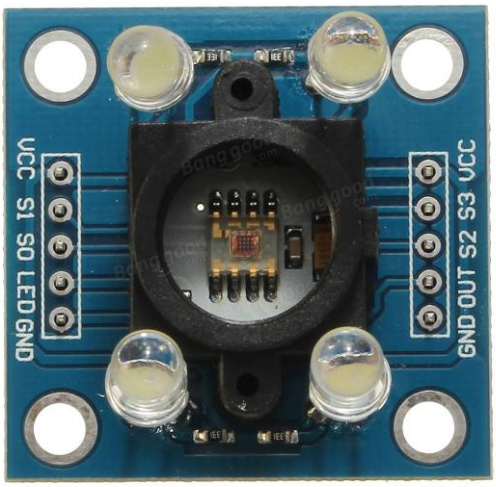


Рисунок 5.3 – Датчик цвета TCS3200.

Таблица 5.3 - Распиновка выводов микросхемы.

Название контакта	Вход/выход сигнала	Описание
VCC	Вход	Питание 5В.
S1	Вход	Масштабирование частоты
S0	Вход	
LED	Вход	Включение 4х светодиодов
GND	Вход	Заземление
OUT	Выход	Вывод частоты
S2	Вход	Выбор типа фотодиодов
S3	Вход	

Скетч определения цвета при помощи датчика цвета TCS3200

```
// назначаем имена для портов
#define s2  2
#define s3  3
#define out 4
// освобождаем память для переменных
byte red = 0;
byte green = 0;
byte blue = 0;
void setup()
{
 pinMode(s2, OUTPUT);
 pinMode(s3, OUTPUT);
 pinMode(out, INPUT);
 Serial.begin(9600);
}
 void loop()
{
 color();
// выводим значения цветов
 Serial.print("color RED :" + String(red));
 Serial.print("color GREEN : " + String(green));
 Serial.println("color BLUE : " + String(blue));
 delay(500);
}
 void color()
```

```
{
 // если 2 и 3 порты отключить, то получим значение красного цвета
 digitalWrite(s2, LOW);
 digitalWrite(s3, LOW);
 red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
 // если 3 порт включить, а 2 отключить, то получим синий цвет
 digitalWrite(s3, HIGH);
 blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
 // если 2 включить, а 3 отключить, то получим зеленый цвет
 digitalWrite(s2, HIGH);
 green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
}
```

5.4 Транспортная лента

Рассмотрим отдельно транспортную ленту стенда рисунок.

Данный лента приводится в движение при помощи двигателя постоянного тока 6BGM37B-52 рисунок. 5.4.2.1

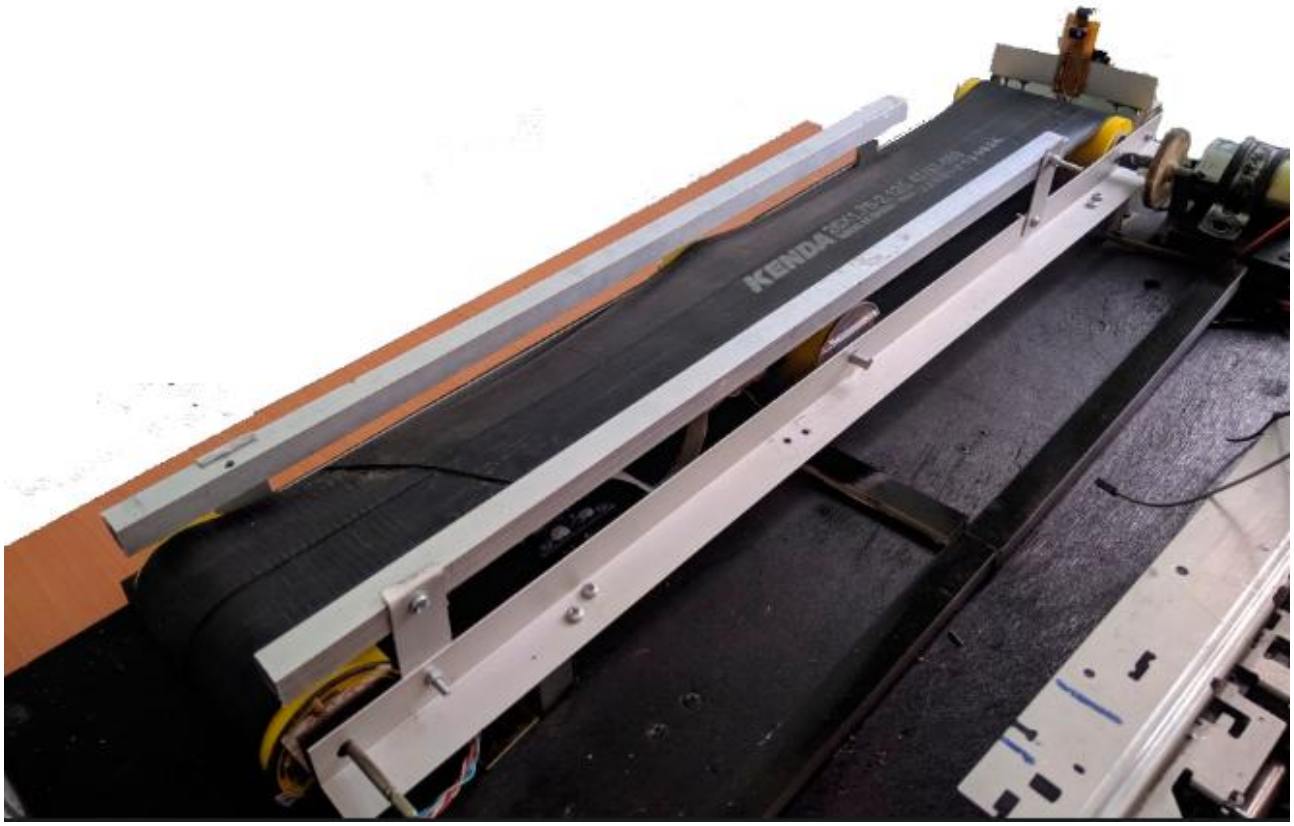


Рисунок 5.4.1 – Транспортная лента.

Алгоритм работы транспортной ленты.

1) Опрашивается оптическая пара.
2) Если значения, полученные оптопарой меньше 900, лента останавливается.
3) Если значения, полученные оптопарой больше 900, лента продолжает свою работу.

Блок схема работы показана на рисунке 5.4.2.

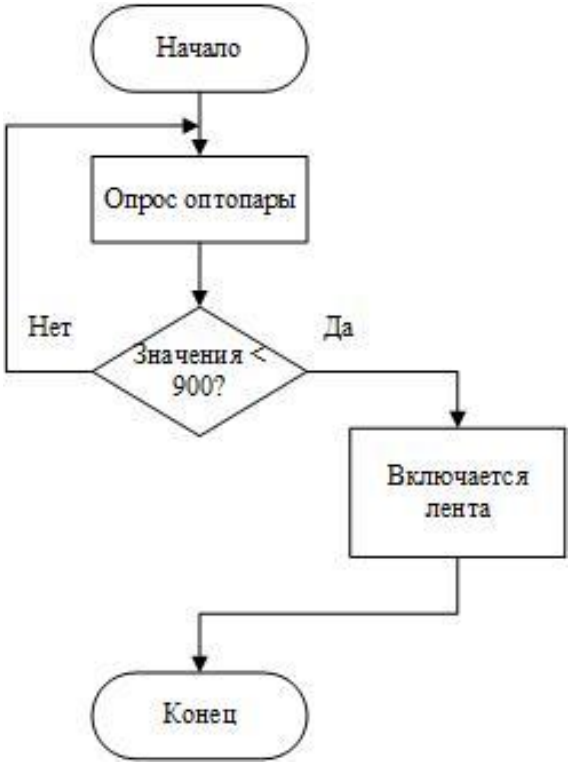


Рисунок 5.4.2 – Блок схема работы транспортной ленты.

5.4.1 Оптическая пара

В лабораторном стенде складской системе для остановки конвейерной ленты используется оптическая пара (оптопара) TCRT5000L, рисунок 5.4.1.1. Если свет, излучённый светодиодом, отражается от доехавшей до конца конвейерной ленты кубика и попадает обратно на фототранзистор, датчик изменяет выходной сигнал, тем самым контроллер получает сигнал о наличии в конце конвейерной ленты детали и останавливает работу ленты. Иначе двигатель конвейерной ленты продолжает работу.

Рисунок 5.4.1.1 – датчик TCRT5000L.

Описание выводов оптической пары показана на рисунке 5.4.1.2.

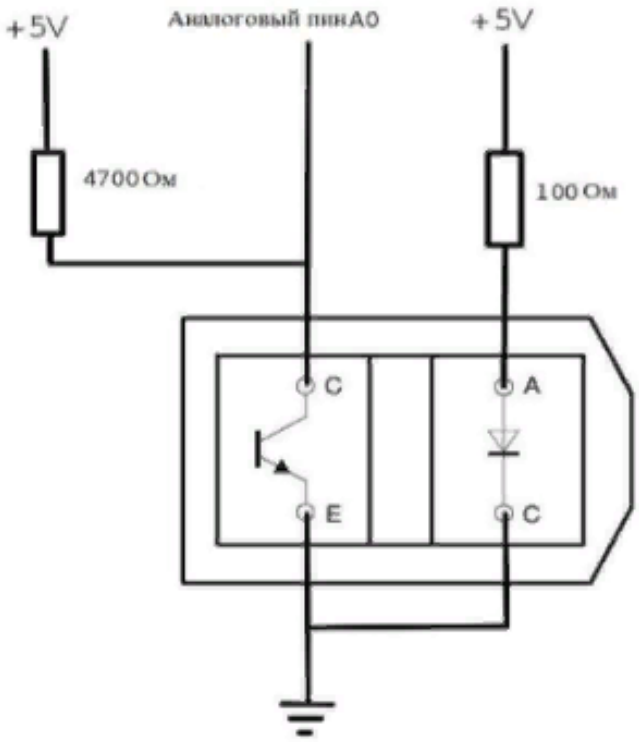


Рисунок 5.4.1.2 – схема контактов датчика TCRT5000L

Скетч опроса оптопары

```
const int ledznach = A0;    // оптопара подклбченна к аналоговому входу
void setup()
{
  Serial.begin(9600);        // установка скорости работы порта
}
void loop()
    {
   int ledznach
   Serial.println(ledznach);  //вывод значений на экран
   delay (10);                  //время обновления показаний
     }
```

5.4.2 Двигатель транспортной ленты

Для работы конвейерной ленты используется электрический двигатель постоянного тока 6BGM37B-52, рисунок 5.4.2.1.

Рисунок 5.4.2.1 - Двигатель постоянного тока 6BGM37B-52.

Данный двигатель при работе потребляет от 700-900 мА. Поэтому, чтобы не перегружать контроллер, было решено управлять двигателем через МОП транзистор IRF540N, рисунок 5.4.2.2 а) внешний вид.

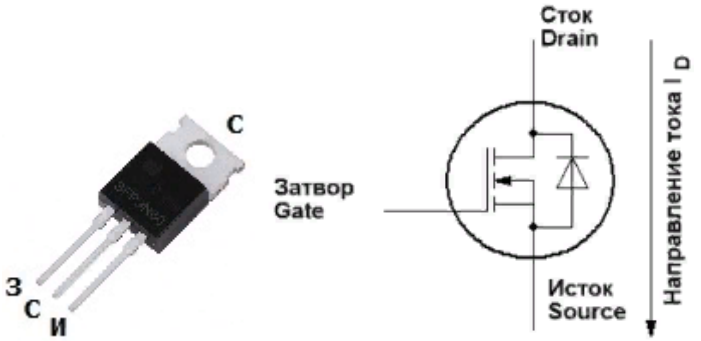


Рисунок 5.4.2.2– МОП транзистор IRF540N а) внешний вид б) цоколевка

Управление МОП транзистором происходит, подавая напряжение на контакт затвора, рисунок 5.4.2.2 б) цоколевка.

Пример управления электрическим двигателем через МОП транзистор.

```
void setup()
{
 pinMode(lenta, OUTPUT);   //выход сигнала из ардуины для двигателя ленты
}
void loop()
   {
  digitalWrite(lenta, HIGH);
  delay(500);
  digitalWrite(lenta, LOW);
  delay(500);
   }
```

Данный скетч периодически включает подает сигнал на «Затвор» транзистора, тем самым открывает его.

6. Управляющая программа

Для управления всей системой необходимо объединить программные коды всех представленных модулей и датчиков в одну программу, написанную в среде разработки Arduino IDE, добавив недостающие программные строки для правильного функционирования стенда.

Управляющая программа

```
// Сервопривод центральный подключен к цифровому выходу 15
// Сервопривод правый подключен к цифровому выходу 16
// Сервопривод левый подключен к цифровому выходу 17
// Сервопривод схвата подключен к цифровому выходу 14
// Контакты датчиков цветов
// S0-3 S1-4 S2-5 S3-6 OUT-7
// ОПТОПАРА A0
// Лента 13
// Шаговик ШД1 8,9,10,11
// Шаговик ШД2 22,24,26,28

#include <Stepper.h>       // библиотека с командами для шаговиков
#include <VarSpeedServo.h> // библиотека с командами для сервомоторов
#include <AccelStepper.h>  // библиотека с командами для шаговиков

//////////ДАТЧИК ЦВЕТА//////////////////
// Контакты TCS230 или TCS3200 для подключения к Arduino:
#define S0 3
#define S1 4
#define S2 5
```

```
#define S3 6

#define sensorOut 7


#define HALFSTEP 8  // Включение полушагового режима работы шагового двигателя, управляемого барабанным складом

#define Pin1  24 // Номер порта на плате Arduino подключаемый к порту IN1 на драйвере L298N

#define Pin2  28 // Номер порта на плате Arduino подключаемый к порту IN2 на драйвере L298N

#define Pin3  26 // Номер порта на плате Arduino подключаемый к порту IN3 на драйвере L298N

#define Pin4  30 // Номер порта на плате Arduino подключаемый к порту IN4 на драйвере L298N


// для хранения частоты, считанной фотодиодами:

int redFrequency = 0;

int greenFrequency = 0;

int blueFrequency = 0;

// для хранения данных о красном, зеленом и синем цветах:

int redColor = 0;

int greenColor = 0;

int blueColor = 0;

//переменная скорости сервоприводов (от 0 до 250)

int SPEED = 40;

//////////////ЛЕНТА+ШАГОВИК+ОПТОПАРА//////////////////////

const int led = A0; //концевик ленты

const int stepsPerRevolution = 320; // Количество шагов шагового двигателя за один оборот

int lenta = 13; //управление двигателем ленты
```

```
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);    // Выводы ШД1
//////////////////СЕРВОПРИВОДЫ//////////////////////
VarSpeedServo centr; // объект сервопривод №1
VarSpeedServo levo;  // объект сервопривод №2
VarSpeedServo pravo; // объект сервопривод №3
VarSpeedServo griffer; // объект сервопривод №4
int levo_pos = 0;    // переменный для хранения положения сервопривода
int pravo_pos = 0;    // переменный для хранения положения сервопривода
int centr_pos = 0;     // переменный для хранения положения сервопривода
AccelStepper myStepper2(HALFSTEP, Pin1, Pin3, Pin2, Pin4); // Создание переменной ШД2 с указанием полу шагового режима работы
void setup() {
  ////////КОНТАКТЫ ДАТЧИКА ЦВЕТА///////////
  // выставляем контакты S0, S1, S2 и S3 в режим OUTPUT:
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  // выставляем контакт sensorOut в режим INPUT:
  pinMode(sensorOut, INPUT);
  // задаем масштабирование частоты на 20%:
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
  // запускаем последовательную коммуникацию:
  Serial.begin(9600);
  //////////ПОДКЛЮЧЕНИЕ СЕРВОПРИВОДОВ//////////////////
  gripper.attach(14);
```

```
  centr.attach(15);
  pravo.attach(16);
  levo.attach(17);
  //////////УСТАНОВКА РАБОТЫ ШД////////////
  myStepper.setSpeed(60);           // Установка начальной скорости ШД1
  myStepper2.setMaxSpeed(300.0);     // Установка максимальной скорости работы ШД2
  myStepper2.setAcceleration(200.0); // Установка ускорения ШД2
  myStepper2.setSpeed(350);          // Установка начальной скорости ШД2
  myStepper2.setCurrentPosition(0);  // Установка начальной позиции для ШД2
}
void loop()
{
  ////////////НАСТРОЙКА ДАТЧИКА ЦВЕТА///////
  // настраиваем датчик таким образом, чтобы считывать данные
  // с фотодиодов с красным фильтром:
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  // считываем выходную частоту:
  redFrequency = pulseIn(sensorOut, LOW);
  // подгоняем считанное значение к диапазону 0-255;
  redColor = map(redFrequency, 125, 170, 500, 0);
  // печатаем значение для красного цвета:
  delay(100);
  // настраиваем датчик таким образом, чтобы считывать данные
  // с фотодиодов с зеленым фильтром:
  digitalWrite(S2, HIGH);
```

```
  digitalWrite(S3, HIGH);
  // считываем выходную частоту:
  greenFrequency = pulseIn(sensorOut, LOW);
  // подгоняем считанное значение к диапазону 0-255;
  greenColor = map(greenFrequency, 246, 270, 500, 0);
  // выводим значение для зеленого цвета:
  delay(100);
  // настраиваем датчик таким образом, чтобы считывать данные
  // с фотодиодов с синим фильтром:
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  // считываем выходную частоту:
  blueFrequency = pulseIn(sensorOut, LOW);
  // подгоняем считанное значение к диапазону 0-255;
  blueColor = map(blueFrequency, 202, 270, 500, 0);
  // печатаем значение для синего цвета:
  delay(100);
  ////////////ЦИКЛ РАБОТА ЛЕНТЫ///////////////
  int ledznaach = analogRead(led);  // Задаем переменную ledznaach для
считывания показаний
  if
  (ledznaach > 900)
    digitalWrite(lenta, HIGH);   // Включение ленты
  else
  {
    digitalWrite(lenta, LOW);   // Выключение ленты
    ////////////КОНЕЦ ЦИКЛА ЛЕНТЫ//////////////////
```

```
//////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1////////
pravo.write(164, SPEED);      // Передвинься на следующую позицию
delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
levo.write(19, SPEED);        // Передвинься на следующую позицию
delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
centr.write(27, SPEED); // Передвинься на следующую позицию
delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
//////////////ЦИКЛ ПРИНТЕРА//////////////////
myStepper.step(-2000);  // Количество шагов в правою сторону до кубика
delay(2000);          // Пауза
//////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 2////////
griffer.write(120, SPEED);
delay(500);

levo.write(54, SPEED);  // Передвинься на следующую позицию
delay(30);        // Пауза 500 мс, чтобы сервопривод достиг позиции
pravo.write(87, SPEED);

delay(2000);          // Пауза после команды
////////СХВАТ БЕРЕТ КУБИК///////
griffer.write(40, SPEED);
delay(500);
//////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1////////
pravo.write(164, SPEED);      // Передвинься на следующую позицию
delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
levo.write(19, SPEED);        // Передвинься на следующую позицию
delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
```

```
    myStepper.step(1300);        // Количество шагов в левую сторону до датчика цвета
    delay(2000);                 // Пауза
    //////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 3////////
    levo.write(27, SPEED);  // Передвинься на следующую позицию
    delay(30);          // Регулирует скорость сервы
    pravo.write(112, SPEED);          // Передвинься на следующую позицию
    delay(30);                    // ждет 15 мс, чтобы сервопривод достиг позиции
  }
  ////////ОПРЕДЕЛЯЕНИЕ ЦИКЛА///////
  // смотрим, какой цвет определился и выводим
  // соответствующее сообщение в мониторе порта:
  /////////ЕСЛИ ЦВЕТ КРАСНЫЙ///////
  if (redColor > greenColor && redColor > blueColor) {
    Serial.println(" - RED detected!");  //  " - это КРАСНЫЙ!"
    //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1////////////
    pravo.write(164, SPEED);       // Передвинься на следующую позицию
    delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
    levo.write(19, SPEED);         // Передвинься на следующую позицию
    delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
    myStepper.step(300);         // Возврат манипулятора на исходную позицию
    delay(2000);                 // Пауза
    ////////ПОВОРАЧИВАЕТЯ НА К БАРАБАН СКЛАДУ//////////
    centr.write(180, SPEED);       // Передвинься на следующую позицию
    delay(20);                  // Регулирует скорость сервы
    //////БАРАБАН СКЛАД ПОВОРАЧИВАЕТСЯ НА ЯЧЕЙКЕ КРАСНОГО ЦВЕТА/////
```

```
  myStepper2.runToNewPosition(−  930);
  //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 4/////////
  levo.write(28, SPEED);  // Передвинься на следующую позицию
  delay(500);          // Регулирует скорость сервы
  pravo.write(85, SPEED);  // Передвинься на следующую позицию
  delay(500);                    // Пауза 500 мс, чтобы сервопривод достиг позиции
//////////РАЗЖИМАЕТСЯ СХВАТ//////////
  gripper.write(120, SPEED);
  delay(500);
  //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1/////////
  pravo.write(164, SPEED); // Передвинься на следующую позицию
  delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
  levo.write(19, SPEED);         // Передвинься на следующую позицию
  delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
  ////////ПОВОРАЧИВАЕТЯ НА ЛЕНТЕ//////////
  centr.write(0, SPEED);      // Передвинься на следующую позицию
  delay(500);                 // Пауза 500 мс, чтобы сервопривод достиг позиции
 }
 //////////ЕСЛИ ЦВЕТ ЗЕЛЕНЫЙ///////////
 if (greenColor > redColor && greenColor > blueColor) {
  Serial.println(" - GREEN detected!");  //  " - это ЗЕЛЕНЫЙ!"
  //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1///////////
  pravo.write(164, SPEED);       // Передвинься на следующую позицию
  delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
  levo.write(19, SPEED);         // Передвинься на следующую позицию
  delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
  myStepper.step(300);         // Возврат манипулятора на исходную позицию
```

```
  delay(2000);              // Пауза
  ////////ПОВОРАЧИВАЕТЯ НА К БАРАБАН СКЛАДУ//////////
  centr.write(180, SPEED);      // Передвинься на следующую позицию
  delay(500);               // Пауза 500 мс, чтобы сервопривод достиг позиции
  //////БАРАБАН СКЛАД ПОВОРАЧИВАЕТСЯ НА ЯЧЕЙКУ ЗЕЛЕНОГО ЦВЕТА///////
  myStepper2.runToNewPosition (150);
  //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 4/////////
  levo.write(28, SPEED);  // Передвинься на следующую позицию
  delay(500);         // Пауза 500 мс, чтобы сервопривод достиг позиции
  pravo.write(85, SPEED);           // Передвинься на следующую позицию
  delay(500);                 // Пауза 500 мс, чтобы сервопривод достиг позиции
  //////////РАЗЖИМАЕТСЯ СХВАТ//////////
  gripper.write(120, SPEED);
  delay(500);
  //////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1/////////
  pravo.write(164, SPEED); // Передвинься на следующую позицию
  delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
  levo.write(19, SPEED);        // Передвинься на следующую позицию
  delay(500);           // Пауза 500 мс, чтобы сервопривод достиг позиции
  ////////ПОВОРАЧИВАЕТЯ НА ЛЕНТЕ//////////
  centr.write(0, SPEED);      // Передвинься на следующую позицию
  delay(500);               // Пауза 500 мс, чтобы сервопривод достиг позиции
 }
 //////////ЕСЛИ ЦВЕТ СИНИЙ////////////
 if (blueColor > redColor && blueColor > greenColor) {
  Serial.println(" - BLUE detected!");  //  " - это СИНИЙ!"
```

```
//////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1////////////
pravo.write(164, SPEED);       // Передвинься на следующую позицию
delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
levo.write(19, SPEED);         // Передвинься на следующую позицию
delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
myStepper.step(300);         // Возврат манипулятора на исходную позицию
delay(2000);               // Пауза
////////ПОВОРАЧИВАЕТЯ НА К БАРАБАН СКЛАДУ//////////
centr.write(180, SPEED);       // Передвинься на следующую позицию
delay(500);               // Пауза 500 мс, чтобы сервопривод достиг позиции
//////БАРАБАН СКЛАД ПОВОРАЧИВАЕТСЯ НА ЯЧЕЙКУ СИНЕГО ЦВЕТА///////
myStepper2.runToNewPosition(−  390);
//////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 4/////////
levo.write(28, SPEED);  // Передвинься на следующую позицию
delay(500);         // Пауза 500 мс, чтобы сервопривод достиг позиции
pravo.write(85, SPEED);           // Передвинься на следующую позицию
delay(500);          // Пауза 500 мс, чтобы сервопривод достиг позиции
//////////РАЗЖИМАЕТСЯ СХВАТ//////////
gripper.write(120, SPEED);
delay(500);
//////////МАНИПУЛЯТОР ЗАНИМАЕТ ПОЗИЦИЮ 1/////////
pravo.write(164, SPEED); // Передвинься на следующую позицию
delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
levo.write(19, SPEED);         // Передвинься на следующую позицию
delay(500);            // Пауза 500 мс, чтобы сервопривод достиг позиции
////////ПОВОРАЧИВАЕТЯ НА ЛЕНТЕ//////////
```

```
    centr.write(0, SPEED);      // Передвинься на следующую позицию
    delay(500);                 // Пауза 500 мс, чтобы сервопривод достиг позиции
  }
if
(myStepper2.distanceToGo()!= 0)   // Если барабан склад не находится в начальной позиции. то
   myStepper2.runToNewPosition (0); // Барабан склад поворачивается в начальную позицию
  delay(5000); // Пауза после выполнения цикла


}
```

Заключение

В данной бакалаврской квалификационной работе были выполнены следующие задачи:

1) Выбран котроллер Arduino Mega 2560 для управления стендом.
2) Написан алгоритм работы всего стенда.
3) Составлена блок-схема и циклограмма работы стенда.
4) Разобраны библиотеки для управления шаговыми и серво двигателями.
5) Разработана управляющая программа лабораторного стенда

Так же был полностью собран стенд «Автоматизированная складская система» При разработке программного кода были решены возникшие проблемы такие как: плохое позиционирование робота-манипулятора, скачкообразные сигналы с датчика оптической пары, дребезжание шаговых двигателей и медленная работа.

Список используемой литературы

1. Канцедал, С.А. Алгоритмизация и программирование : Учебное пособие / С.А. Канцедал. – М.: ИД ФОРУМ, НИЦ ИНФРА– М, 2013. – 352 с.
2. Петрова, А.М. Автоматическое управление: Учебное пособие / А.М. Петрова. – М.: Форум, 2010. – 240 с.
3. Using Arduino as a Platform for Programming, Design and Measurement in a Freshman Engineering, 2011, Gerald W. Recktenwald, David E. Hall
4. Atmel. [Электронный ресурс]: документация. – режим доступа: http://www.alldatasheet.com/datasheet-pdf/pdf/392279/ATMEL/ATMEGA168PAPU.html
5. SG90 9g Micro Servo [Электронный ресурс]: документация – режим доступа: http://www.micropik.com/PDF/SG90Servo.pdf
6. Глибин Е.С. Программирование электронных устройств : электронное учеб. пособие / Е.С. Глибин, А.В. Прядилов. – Тольяттти : Изд-во ТГУ, 2014.: 1 оптический диск.
7. Михеев, В.П., Просандеев, А.В. Датчики и детекторы. Учебное пособие / В.П. Михеев, А.В. Просандеев – М.: МИФИ. Москва 2007. – 172с
8. Вальков, В.Б. Автоматизированные системы управления технологическими процессами / В.Б. Вальков. – Л.: Политехника. Липецк 2011. – 269 с
9. Управление шаговым двигателем с Ардуины [Электронный ресурс] https://lesson.iarduino.ru/page/upravlenie-shagovym-dvigatelem-s-arduiny
10. Библиотека Servo [Электронный ресурс] http://arduino.ru/Reference/Library/Servo
11. Подключение датчика цвета к Ардуино [Электронный ресурс] http://роботехника18.рф/датчика-цвета-ардуино-подключение/
12. Подключение и управление манипулятором [Электронный ресурс] https://lesson.iarduino.ru/page/sborka-robota-manipulyatora-chast-2-elektronika/
13. Машиностроение. Энциклопедия. Т.1-4. Автоматическое управление. Теория, 2000,
14. Роботы и автоматизация производства. 1989, Р. Асфаль.

15. Комплексная автоматизация производственных процессов. 1973, Г.А. Шаумян

16. Putting the Fun in Programming Fundamentals - Robots Make Programs Tangibl, 2013, Dr. Todd R. Hamrick, Dr. Robin A.M. Hensel.

17. A Review of Embedded Systems Education in the Arduino Age: Lessons Learned and Future Directions, 2017, Mohammed El-Abd.

18. Controlling AC motor using Arduino microcontroller, 2017, Donald S Zinger, Nithesh R. Nannuri.

19. Performance comparison of robotic arm using Arduino and Matlab ANFIS, 2015, Naser Alanabi , Dr. Jyoti Shrivastava.

20. Teaching Joint-Level Robot Programming with a New Robotics Software Tool, 2017, Fernando Gonzalez, Janusz Zalewski.