МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт энергетики и электротехники
(институт)

Кафедра «Промышленная электроника»

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

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

БАКАЛАВРСКАЯ РАБОТА

на тему Учебный стенд «Автоматизированный транспортно− складской комплекс»

Студент(ка)	Е.В. Булаева		
	(И.О. Фамилия)		(личная подпись)
Руководитель	Д.Г. Токарев		
	(И.О. Фамилия)		(личная подпись)
Консультанты			
	(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)

«_____»_____2016 г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт энергетики и электротехники

(институт)

Кафедра «Промышленная электроника»

УТВЕРЖДАЮ

Зав.кафедрой «Промышленная электроника»

_____ А.А. Шевцов

(подпись) (И.О. Фамилия)

«____»_____2016г.

ЗАДАНИЕ

на выполнение бакалаврской работы

Студент Булаева Екатерина Владимировна

1.Тема Учебный стенд «Автоматизированный транспортно− складской комплекс»

2. Срок сдачи студентом законченной бакалаврской работы 24.06.2016 г.

3. Исходные данные к бакалаврской работе структурная схема

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов, разделов)

Введение

1) Общие сведения
2) Обзор и выбор микроконтроллера
3) Общие сведения об Arduino
4) Разработка циклограммы учебного стенда «Автоматизированный транспортно− складской комплекс»
5) Разработка алгоритма и блок-схемы работы учебного стенда

Заключение

Список использованной литературы

Приложение

5. Ориентировочный перечень графического и иллюстративного материала

1) Изображение учебного стенда «Автоматизированный транспортно− складской комплекс» 1 лист А1
2) Эскиз учебного стенда 1 лист А1

3) Структурная схема 1 лист А1
4) Циклограмма работы комплекса 1 лист А1
5) Таблица соответствия выводов и переменных 1 лист А1
6) Общая блок–схема 1 лист А1

6. Консультанты по разделам

7. Дата выдачи задания « 30» мая 2016 г.

Руководитель бакалаврской работы

	Д.Г. Токарев
(подпись)	(И.О. Фамилия)

Задание принял к исполнению

	Е.В. Булаева
(подпись)	(И.О. Фамилия)

Аннотация

В данной бакалаврской работе (далее БР) разработана программная часть учебного стенда "Автоматизированный транспортно− складской комплекс" на основании структурной схемы, которая используется как исходные данные. Представлен обзор и выбор по параметрам экономичности и эффективности внедрения в коммплекс микроконтроллера, среды разработки и языка программирования, а также подробно рассмотрен вопрос подключаемых библиотек и их функций, используемых для управления такими компонентами как сервопривод, шаговый двигатель, датчик цвета. Помимо этого разработана циклограмма работы комплекса, содержащая в себе описание тактов работы комплекса, список оборудований, и состояния работы каждого исполнительного механиза в зависимости от такта работы. На базе этого реализован принцип управления каждым устройством комплекса, который в дальнейшем был сформирован в общую подробную блок− схему, которая в свою очередь позволила создать программный код управления учебным стендом "Автоматизированный транспортно− складской комплекс".

БР выполнена на 59 страницах машинописного текста, и графическая часть в объеме 6 листов А1.

Содержание

Введение

Резкое стремление развития современной промышленной автоматизации, рост количества компаний, работающих в этой отрасли, привели к постоянному расширению набора требований, предъявляемых к системам транспортировки, сортировки и тестирования различных микросхем, устройств и модулей. В условиях жесткой экономической конкуренции и высокой стоимости оборудования особую важность приобретает задача – повышение эффективности процесса разработки новых установок, а так же уменьшение затрачиваемых средств на их реализацию. Решение проблемы предполагает выбор наиболее простых, дешевых и эффективных компонентов комплекса и способов их управления. Среди них упрощение алгоритма работы комплекса и ускорение процесса перехода от модели программного комплекса к реализации программного кода.

Целью БР является разработка программной части учебного стенда "Автоматизированный транспортно- складской комплекс", которая включает определение алгоритма работы каждого модуля, реализацию программного обеспечения, программной среды для управления комплексом, который в свою очередь выполняет такие промышленные операции как перемещение детали, сорировка по цвету и складирование. Для достижения поставленной цели необходимо:

1) разработать обобщенную информационно- логическую модель комплекса в виде алгоритма работы;
2) разработать схематично обобщенную структуру построения программного кода в виде блок- схемы на основании циклограммы;
3) выбрать программную среду управления элементами комплекса.

1 Общие сведения

В современной промышленности развиты комплексы промышленной автоматизации. Существуют ведущие производители роботизированных комплексов. Они выпускают специализированное оборудование. Например, компания FESTO. Её подразделение FESTO Didactic выпускает ряд специализированных устройств, учет компонентов которых моделируют те или иные фрагменты технологических операций. Стоимость подобных комплексов от компании FESTO начинается от 3.5 млн рублей., и работают они с помощью гидравлической системы. В связи с финансовой неспособностью приобретения оборудования такого уровня поставлена задача в разработке комплексного устройства, моделирующего основные элементы транспортно− сортировочной системы, который не теряя эффективность и простоту сможет существенно сократить расходы на создание и материалы комплекса, а так− же упростить алгоритм работы.

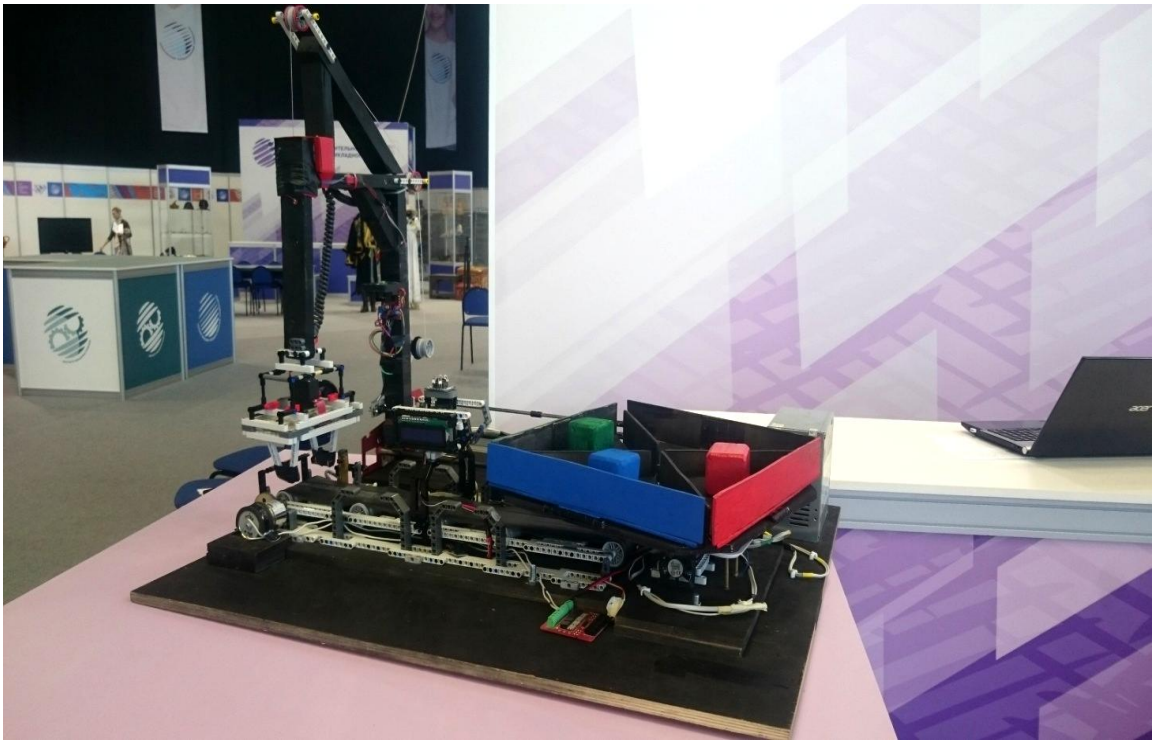


Рисунок 1.1 − Учебный стенд "Автоматизированный транспортно− складской комплекс"

Спроектированный учебный стенд "Автоматизированный транспортно– складской комплекс", изображенный на рисунке 1.1, представляет собой рабочую модель автоматизированной складской системы, которая сортирует и перемещает детали по цвету. Установка предназначена для наглядного представления практической реализации возможностей современных микроконтроллеров и привлечения абитуриентов на технические специальности.

Данный стенд, эскиз которого представлен на рисунке 1.2, включает в себя: манипулятор, транспортную ленту и накопительный барабан– склад.

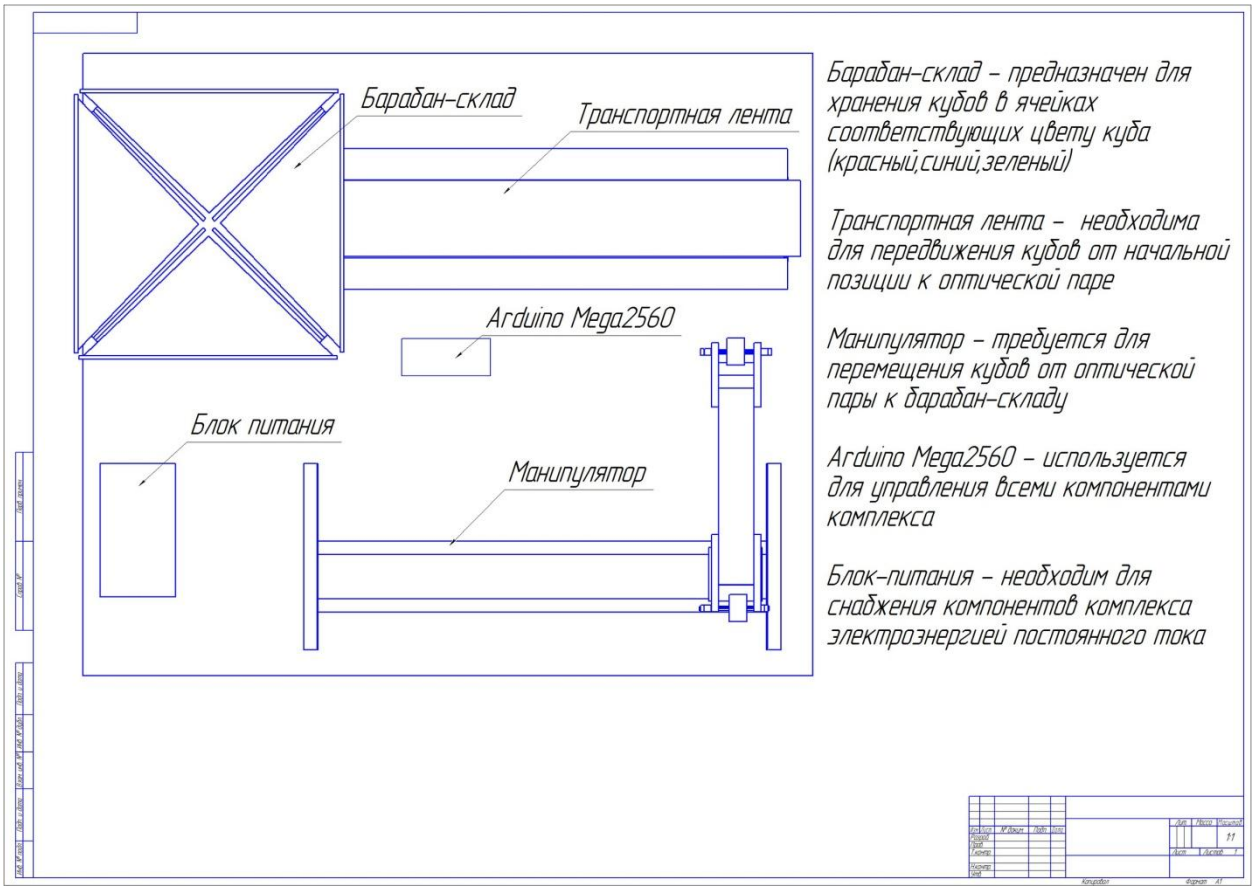


Рисунок 1.2 – Эскиз учебного стенда

Система управления состоит из микроконтроллера, который в свою очередь управляет различными элементами учебного стенда: двигателями (шаговый, серво– , постоянного тока), датчиками (цвета, размера), LCD дисплеем, оптической парой. Структурная схема "Автоматизированного транспортно– складского комплекса" показана на рисунке 1.3.

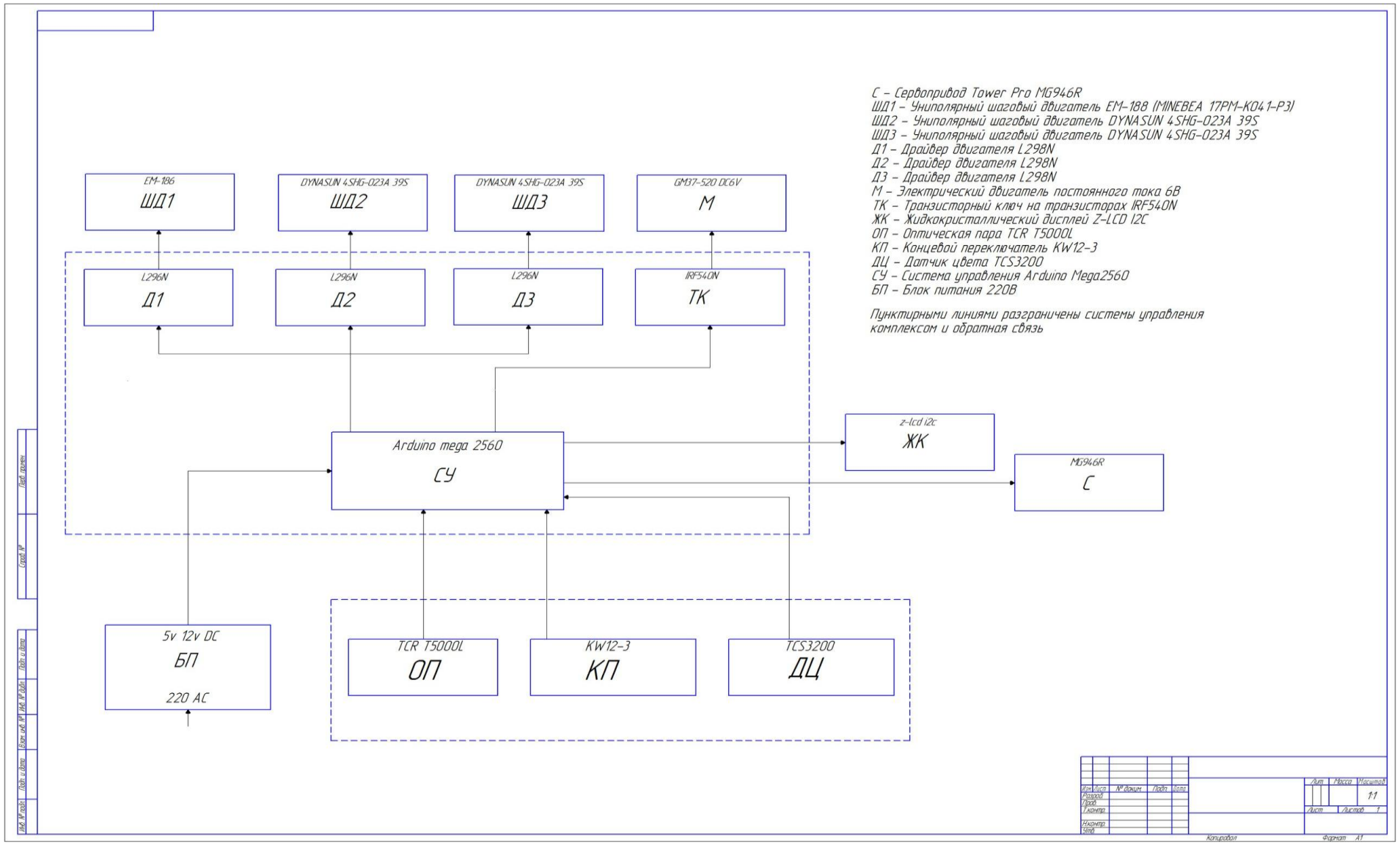


Рисунок 1.3 – Структурная схема

Здесь:

1) ШД1 – шаговый двигатель предназначен для работы с барабан– складом. Благодаря шаговому двигателю склад имеет возможность поворачиваться на программно заданную ячейку в зависимости от цвета куба;

2) ШД2 – шаговый двигатель, управляющий движением манипулятора в вертикальной плоскости, т.е. выполняет подъем и опускание манипулятора;

3) ШД3 – шаговый двигатель служит для перемещения манипулятора по горизонтальной плоскости от точки взятия куба и до первоначального положения куба на транспортной ленте;

4) М – электрический двигатель постоянного тока обеспечивает работу транспортной ленты;

5) С – сервопривод отвечает за работу схвата манипулятора, т.е. разжимает либо сжимает схват для того, чтобы взять куб;

6) ДЦ – датчик цвета установлен для определения цвета куба, помещаемый в барабан– склад;

7) ОП – оптическая пара, расположенная в конце транспортной ленты, необходима для остановки работы транспортной ленты;

8) ЖК – Жидкокристаллический дисплей отображает название цвета куба;

9) КП – концевой переключатель предназначенный для остановки ШД2 на определенной высоте;

10) СУ – система управления Arduino Mega 2560, обеспечивающая управление всеми компонентами системы;

11) БП – вторичный источник электропитания, необходимый для снабжения узлов комплекса электроэнергией постоянного тока, путём преобразования сетевого напряжения до требуемых значений;

12) Д1– Д3 – драйвер двигателя L298N, позволяющий управлять ШД1– ШД3;

13) ТК – транзисторный ключ на транзисторах необходимый для включения и отключения М.

2 Обзор микроконтроллеров

2.1 Выбор микроконтроллера

Во время разработки учебного стенда возникает необходимость использовать микроконтроллер, т.е. микросхему, предназначенную для управления различными электронными устройствами для реализации ряда функциональных возможностей. Для того, чтобы верно выбрать микроконтроллер, обязательно нужно понять какие факторы будут учитываться при отборе. Задача данной БР выбрать бюджетный микроконтроллер (для минимизации общей стоимости комплекса), но в то же время соответствующий требованиям стенда, т.е. по производительности, стоимости, простоте применения и т.д. Вследствие этого проанализируем ряд микроконтроллеров различных фирм: Arduino, Raspberry, Strela, Iskra.

Существует 2 категории плат для разработки, представленные в таблице 2.1:

Таблица 2.1 – Категории плат

Платы на микроконтроллере (MCU, MicroController Unit)	Одноплатные компьютеры (SoC, System on a Chip)
Типичный представитель — Arduino ADK	Типичный представитель — Beaglebone Black

Если рассматривать платы с точки зрения возможностей, то в этом случае выигрывают одноплатные компьютеры, выполняющие программы в операционной системы (ОП), так как у них есть отличительная особенность – большая эффективность и широкий мультимедийный потенциал. К сожалению, микроконтроллеры этим похвастаться не могут, ведь они в состоянии только выполнить одну задачу, зато прекрасно с этим справляются. Помимо этого, на рынке представлены и гибридные платформы, которые совмещают в себе процессор и микроконтроллер. Смысл заключается в том, чтобы распределить задачи на 2 устройства по сложности, то есть микроконтроллеру достается

процедура управления различными двигателями, реле, датчиками, а процессору выполнять наиболее сложные функции, такие как обработку медиафайлов, выход в сеть.

Кроме того, можно реализовать их совместное взаимодействие через общие интерфейсы, если приобрести по одной плате из каждой категории. Но как отмечалось ранее, по заданию на БР необходимо выбрать ту плату, которая была бы наиболее производительной и простой в подключении, но в тоже время и дешевой. К сожалению, у одноплатных компьютеров на рынке высокая стоимость, что приводит к увеличению затрат на создание учебного комплекса. Ниже приведена таблица, которая поможет сравнить и выбрать специализированную плату, которая сильно выделяется среди прочих особенностями удовлетворяющими разработку комплекса.

Таблица 2.2 – Характеристики микроконтроллеров и одноплатных компьютеров

	Микроконтроллер	Одноплатный компьютер
Производительность	1 ядро, десятки– сотни МГц, десятки КБ оперативки, десятки– сотни КБ постоянной памяти.	1 или более ядер, сотни– тысячи МГц, сотни МБ оперативки, гигабайты постоянной памяти.
Многозадачность	Отсутствует	Присутствует. Управляется ОС.
Скорость реакции в проектах критичных к времени	Высокий контроль над временем и длительностью подачи сигналов.	Из– за многозадачности критический процесс может пропустить своё время.
Выбор языков программирования	Ограниченный. Чаще C/C++.	Python, JavaScript, Bash и др.
Возможности для работы со звуком	На мощных микроконтроллерах возможен синтез звука. Для работы с MP3/OGG/WAV нужны дополнительные модули.	Поддержка MP3/OGG/WAV на уровне ОС. Аудиовыход HDMI и/или разъём 3,5 мм.

Используя описание и характеристики микроконтроллеров и одноплатных компьютеров, которые были исследованы в прошлых разделах, составим список наиболее подходящих для данного стенда с учетом их достоинств и недостатков.

Микроконтроллер Arduino Uno представлен на рисунке 2.1.

Рисунок 2.1 – Arduino Uno

Процессор с частотой 16 МГц, объем постоянной памяти 32 КБ и оперативной составляет 2 КБ, наличие 20 портов ввода– вывода, из них 6 аналоговых входов и 6 каналов ШИМ, 2 аппаратных прерывания. Данная модель способна решать любые миссии по исправному управлению исполнительными устройствами.

Достоинства:

1) Огромное количество онлайн– уроков, созданных библиотек, легкая в освоение среда разработки Arduino IDE с языком на базе C++.

2) Требуемое напряжение в 5 В, есть возможность установки плат расширения, наличие аналоговых входов – это все дает возможность контактировать с разной периферией, датчиками, двигателями.

Микроконтроллер Arduino Leonardo представлен на рисунке 2.2.

Рисунок 2.2 – Arduino Leonardo

Похожа на Arduino Uno, но имеет микроконтроллер ATmega32u4. Несмотря на это есть несколько отличительных черт, хотя и находится в том же классе.

Достоинства:

1)Количество аналоговых входов увеличено до 12, а так же ШИМ каналов до 7 и пинов с аппаратным прерыванием до 5. Так же имеются обособленные независимые serial− интерфейсы для USB и UART.

2) Arduino Leonardo, подключенный через USB порт к компьютеру, может быть клавиатурой или мышью.

Недостатки:

1) Несовместимость с некоторыми платами расширения, так как распиновка отличается от Arduino Uno.

Микроконтроллер Arduino Micro представлен на рисунке 2.3.

Рисунок 2.3 − Arduino Micro

Есть схожесть с Arduino Leonardo, но выполнение в более компактной форме.

Достоинства:

1) Компактность. Всего 48×18 мм.

Недостатки:

1) Ввиду маленького размера, установка плат расширения станет затруднительной. Потребуется макетная плата с дополнительными модулями и проводами.

Микроконтроллер Arduino Mega представлен на рисунке2.4.

Рисунок 2.4 – Arduino Mega

Создана на базе мощного микроконтроллера той же архитектуры как и у Arduino Uno. Подходит для наиболее затруднительных задач, где требуется изобилие выводов и большой объем памяти.

Достоинства:

1) Объем памяти достигает 256 КБ, а оперативной 8 КБ. Наличие портов увеличено до 60, из данных 16 аналоговых, а 15 с ШИМ.

Недостатки:

1) Габариты составляют 101×53 мм, оказываясь больше чем у Arduino Uno.

Микроконтроллер Arduino Due представлен на рисунке 2.5.

Рисунок 2.5 – Arduino Due

Реализована на микроконтроллере Cortex– M3,это говорит о том, что производительность платы выше чем у остальных ее предшественников. А по габаритам напоминает Arduino Mega.

Достоинства:

1) Обладает процессором с 84 МГц частотой и памятью в объеме 512 КБ. Количество пинов увеличено до 66, из них 12 являются аналоговыми входами, а другие 12 поддерживают ШИМ.

Отметим, что 66 пинов имеют возможность быть настроенными, как аппаратные прерывания.

2) Допустимость взаимодействия с автомобильной электроникой и создание сети из Due благодаря встроенному контроллеру шины CAN. Присутствие двух каналов ЦАП для синтезирования стереозвука с частотой 4,88 Гц.

Недостатки:

1) К сожалению,3,3 В – напряжение данной платы, а не всем удобное 5 В. Для ликвидации этой загвоздки неизбежна установка преобразователей уровня напряжений.

Микроконтроллер Iskra JS представлен на рисунке 2.6.

Рисунок 2.6 – Iskra JS

Программирование для этой платы становится реализованным на языке JavaScript, а не на привычном Си++. К тому же плата расположена на ядре Espruino.

Достоинства:

1) Сомнение по поводу сложности JavaScript развеивается, если принять тот факт, что программирование становится явно компактнее и легче особенно для создания массивов данных, веб– интерфейсов.

2) Наличие внушительного микроконтроллера Cortex M4 с частотой 168 МГц, объема флеш– памяти 1 МБ, а оперативной 192 КБ. Присутствие 2

аналоговых выхода, десятки портов с ШИМ, по нескольку портов I²C, SPI, UART.

3) Также уровень напряжения платы составляет 3,3 В, но есть особенность

заключающаяся в подключение пятивольтовой периферии тривиально.

Недостатки:

1) В виду низкой популярности довольно сложно найти готовые библиотеки. Именно по этой причине программирование на ней становится сложным, так как придется самостоятельно все реализовывать. Кроме того, этот микроконтроллер имеет другую среду разработки.

2) Приобретение данной модели затруднительно вследствие низкого спроса.

Микроконтроллер Strela представлен на рисунке 2.7.

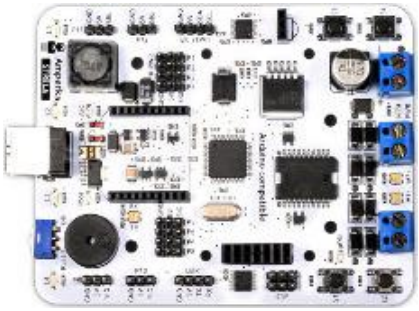


Рисунок 2.7 – Strela

Несмотря на другого производителя, Strela программируется через среду разработки Arduino IDE и состоит из того же микроконтроллера, что и Arduino Leonardo.

Достоинства:

1) Наличие слотов для ЖК– экрана и модуля беспроводной связи, 4 кнопок, разъемов для сервопривода и светодиодов. Вдобавок для двух двигателей предназначен встроенный драйвер.

2) Использование широкого спектра аккумуляторов с помощью мощного регулятора питания.

3) Допустимо присоединение вспомогательных датчиков и модулей благодаря 11 входов– выходов в виде 3– контактных разъёмов в виде 3– контактных разъёмов.

Недостатки:

1) Отсутствие колодок для установки плат расширения.

2) Из– за отличной от Arduino Leonardo нумерации контактов используются иные процедуры для работы с выводами платы.

3) Высокая стоимость.

Одноплатный компьютер Raspberry Pi 2 Model B представлен на рисунке 2.8.

Рисунок 2.8 – Raspberry Pi 2 Model B

Данная модель имеет немалую популярность среди разработчиков. Для задач требующих вычислительные ресурсы в этой плате предусмотрено четыре ядра по 900 МГц каждый, объем оперативной памяти равной 1 Гб и полноценную ОС Linux, основанную на Debian.

Достоинства:

1) Довольно много информации о том, как программировать, из этого следует, что использование платы не составляет труда.

2) Благодаря наличию портов HDMI, 3,5 мм аудио, 4 USB можно легко управлять монитором, колонками, клавиатурой, мышью и другими USB– устройствами. В дополнение этого предусмотрена беспроводная связь через модули BLE и WiFi.

Недостатки:

1) Нехватка АЦП сказывается на затрудненности подключения аналоговых сенсоров. Решение этой проблемы доступно в использовании внешних дополнительных компонентов.

2) Усложненная работа с периферией за неимением более 1 ШИМ− канала.
3) Высокая стоимость

Изучая характеристики каждого микроконтроллера важными критериями отбора являются: стоимость, большое количество выводов, большой объем памяти для загрузки программ, практичная платформа быстрой разработки электронных устройств для новичков, простота языка программирования и большой спрос на рынке. Всем этим требованиям отвечает Arduino Mega 2560.

3 Общие сведения об Arduino

Arduino является небольшой платой, состоящая из собственной памяти и процессора. Помимо этого плата имеет контакты, посредством которых можно подключиться к различным устройствам: сервоприводам, шаговые двигателям, дисплеям, сенсорам и многим другим. Управление этими компонентами осуществляется благодаря программе, которая выполняет заданный алгоритм. Для обеспечения этого программный код через среду разработки Arduino IDE необходимо загрузить в процессор Arduino.

Высокий спрос и популярность Arduino получила в основном из− за того, что она легка в использовании, т.к. не требует глубоких знаний в программировании, ведь освоить азы в работе с Arduino можно за несколько часов. А в этом помогут большое количество онлайн− уроков, статьей, обзоров и заметок на русском языке в сети.

Язык программирования для плат Arduino является привычный C++ с процедурами для управления контактами ввода/вывода. Для того, чтобы начать работать с Arduino требуется всего лишь бесплатная среда разработки Arduino IDE и подключенный микроконтроллер к компьютеру через USB− порт. Несмотря на легкость в эксплуатации можно усложнить задачу и программировать через другие среды разработки, например, Eclipse, Visual Studio, или даже командную строку.

Кроме всего прочего, Arduino может похвастаться присутствием плат для расширения или по другому "шилдами"(от анг.shields). Идея их установки состоит в усовершенствовании Arduino. Например, для необходимости доступа к сети и интернету потребуется "Ethernet Shield", в случае управления моторами "Motor Shield", а для получения координат "модуль GPS" и др.

3.1 Библиотеки Arduino необходимые для управления ШД

Библиотека — это набор дополнительных команд, который позволяет вводить программу в упрощенном формате. Для управления шаговыми двигателями существуют 2 библиотеки Arduino, отличающиеся друг от друга сложностью и скоростью управления.

1) Библиотека Stepper – #include <Stepper.h>

Эта библиотека поможет управлять такими типами шаговых двигателей (далее ШД), как биполярный и униполярный. Несмотря на это могут понадобиться определенные электронные компоненты в зависимости от типа ШД и способа его подключения. Так же Stepper поддерживает только полношаговый метод управления и имеет сильно ограниченные возможности. Использовать ее можно только в очень простых приложениях, в которых используется только один мотор.

Основные функции:

1. Stepper(steps, pin1, pin2)

2. Stepper(steps, pin1, pin2, pin3, pin4)

Данная функция предназначена для создания объекта класса Stepper, который в свою очередь соотнесен к одному ШД, присоединенному к микроконтроллеру Arduino. При объявлении переменной класса Stepper конструктор рекомендуется использовать вне setup() и loop(), т.е. в самом начале. Исходя из подключенного числа выходов двигателя двух или четырех, варьируется количество параметров в конструкторе.

3. Stepper.setSpeed(rpms)

Эта функция треубется для установки вращения 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. Stepper.step(steps)

Предназначение этой функции состоит во вращение ШД на заданное количество шагов при скорости, прописанной функцией setSpeed(). Работа этой функции не заканчивается пока вращение ШД не будет завершено, после чего происходит переход к следующей команде.

Предположим, при установке скорости вращения равной 2 RPM (2 оборотам в минуту), и вызове step(200) на двигателе с полным оборотом в 200 шагов, функция буде выполнятся в течении двух минут. Для решения вопроса, касающегося длительной блокировки выполняемого кода скетча, желательно установить значение скорости высоким, а количество шагов за один вызов step() низким.

2) Библиотека AccelStepper – #include <AccelStepper.h>

AccelStepper делает возможным управление ШД, но с отличительными особенности по сравнение от стандартной библиотеки Stepper. Эта библиотека очень хорошо работает совместно с шаговым мотором 28BYJ– 48 (мотор почти не греется), а также поддерживает ускорение, что позволяет заставить мотор вращаться быстрее. Библиотека использует код, не блокирующий шаги и включает немало других приятных особенностей.

Преимущества следующие:

- обеспечение замедления и ускорения ШД;
- обеспечение совместной работы двух и более ШД;
- обеспечение Adafruit Motor Shield;
- обеспечение SparkFun EasyDriver;
- обеспечение очень маленьких скоростей;
- обеспечение субклассов;
- поддержание микрошаговых режимов и различных типов ШД.

Основные функции:

1) Инициализация и установка параметров:

- AccelStepper mystepper(1, pinStep, pinDirection); – ШД, который управляется за счет выделенной платы.

– AccelStepper mystepper(2, pinA, pinB); – Биполярный ШД, который управляется за счет Н– моста.

– AccelStepper mystepper(4, pinA1, pinA2, pinB1, pinB2); – Униполярный ШД, который управляется с помощью четырех транзисторов.

– mystepper.setMaxSpeed(stepsPerSecond); – Так как скорость по умолчанию довольно низкая, то есть необходимость в ее переопределение благодаря установке максимальной скорости. В процессе работы ШД примется ускорятся, пока не достигнет этой максимальной скорости, и замедлится при подходе к концу движения.

– mystepper.setAcceleration(stepsPerSecondSquared); – Регулирование ускорения, в шагах в секунду за секунду.

2) Управление позицией:

– mystepper.moveTo(targetPosition); – Перемещение в абсолютно указанное положение посредством запуская функции run().

– mystepper.move(distance); – Перемещение в относительно указанное положение посредством запуская функции run(). Значение distance может быть больше или меньше нуля.

– mystepper.currentPosition(); – Возвращение текущего абсолютного положение.

– mystepper.distanceToGo(); – Возвращение расстояние до указанного положения для проверки достижения двигателем указанной конечной точки.

– mystepper.run(); – Начало движения ШД, но для продолжения движения ШД следует вызывать функцию заново.

– mystepper.runToPosition(); – Начало движения ШД. Не осуществляет возврата, пока двигатель не остановится.

3) Управление шагами:

– mystepper.setSpeed(stepsPerSecond); – Установка скорости в шагах за секунду путем запуска функции runSpeed().

– mystepper.runSpeed(); – Начало движения ШД, но для продолжения движения ШД следует вызывать функцию снова.

При создании учебного стенда "Автоматизированный транспортно− складской комплекс" были использованы три шаговых двигателя, управляемые через драйвер L298N:

1) униполярный шаговый двигатель EM− 188(MINEBEA 17PM− K041− P3) (далее ШД1);

2) униполярный шаговый двигатель DYNASYN 4SHG− 023A 39S (далее ШД2);

3) униполярный шаговый двигатель DYNASYN 4SHG− 023A 39S (далее ШД3);

С помощью ШД1 управляется барабан− склад, в котором хранятся распределяющиеся по цвету кубы. Для этого управление ШД должно обеспечить плавную работу и торможение при окончании движения привода, и в дополнение к этому минимизацию погрешности в шагах. Вследствие этого было решено использовать библиотеку AccelStepper, так как только она имеет особенность обеспечения замедления и ускорения ШД.

ШД2 приводит в движение манипулятор в горизонтальном положение. Поскольку требования к данному ШД минимальны потому, что он выполняет цикличное действие вправо/влево, то в таком случае библиотека не требуется для его программирования.

ШД3 так же задействует манипулятор, но только уже в вертикальном положение, т.е. опускает и поднимает на определенное расстояние, выраженное в шагах. Задача немного усложнилась, поэтому реализовать ее поможет стандартная библиотека Stepper.

3.2 Библиотеки Arduino необходимые для управления сервопривод

Помимо ШД в комплексе установлен сервопривод (далее С) марки Tower Pro MG946R. К сожалению выбор в библиотеках отсутствует, поэтому решено использовать ниже описанную библиотеку Servo.

1) Библиотека управления сервоприводом − Servo − #include <Servo.h>

Данная библиотека для сервопривода содержит набор функций необходимых для управления двигателем. Существуют 2 вида сервоприводов:

− имеющие возможность поворота привода только на заданный угол от 0 до 180 градусов;

− позволяющие совершать полный оборот, т.е. 360 градусов с заданной скоростью.

Библиотека справляется с управлением 12− ю сервоприводами на большом количестве плат Arduino, кроме Arduino Mega, так как на ней количество увеличилось до 48.

С учетом того, что выбран микроконтроллер Arduino Mega2560, нет недостатка в невозможности использования 9 и 10 выходов в режиме ШИМ как на других платах Arduino. Кроме того, плата Mega подразумевает подключение до 12 сервоприводов без потери функционала ШИМ. Но если количество двигателей увеличено до 23, то в таком случае станет невозможным использование 11 и 12 выходов для ШИМ.

Основные функции:

1. Servo.attach()

Подключение библиотеки к заданному выходу, с которого реализуется управление приводом. На более ранних версиях Arduino библиотека Servo поддерживала управления только с помощью 9 и 10 портов.

2. Servo.write()

Считывание значения для управления приводом. Для стандартного сервопривода это угол поворота. Для привода постоянного вращения, функция задает скорость вращения (0 − для максимальной скорости вращения в одну сторону, 180 − для максимальной скорости в другую сторону и около 90 для неподвижного состояния).

3. Servo.writeMicroseconds()

Передача значения для управления сервоприводом в микросекундах (uS), устанавливая угол поворота на это значение. Для стандартного привода значения следующие:

1000 – максимальный поворот против часовой стрелки;

2000 – максимальный поворот по часовой стрелке;

1500 – остановка посередине.

Некоторые производители не придерживаются стандартных значений, и такие приводы могут управляться значениями от 700 до 2300. Тем не менее, следует избегать постоянного использования привода на значениях больше допустимых.

Приводы постоянного вращения реагируют на данную команду подобно реакции на функцию write().

4. Servo.read()

Считывание значения текущего положения сервопривода (значение записанное последним вызовом функции write()).

5. Servo.attached()

Определение есть ли привязка к приводу через вывод. Возвращает true если есть привязка к выводу, а в противном случае – false .

6. Servo.detach()

Отключение вывода от библиотеки Servo. В случае отключенных переменных Servo – выводы 9 и 10 можно использовать для PWM– вывода через analogWrite().

3.3 Библиотека Arduino необходимая для управления дисплея

Так как в стенде необходимо выводить цвет куба, требовалось использовать ЖК– дисплей с удобной библиотекой, не затрудняющей программирование. В Arduino есть стандартная библиотека LiquidCrystal, о которой ниже представлен обзор.

1) Библиотека – LiquidCrystal – #include <LiquidCrystal.h>

Эта библиотека предоставляет микроконтроллеру управлять различными жидкокристаллическими дисплеями (далее ЖК), созданными на базе известного чипсета Hitachi HD44780 (или совместимого). Данная библиотека

поддерживает несколько режимов работы, такие как 4− х, так и 8− битный, а это означает, что есть возможность использовать 4 или 8 линий данных, совместно с управляющими линиями RS, Enable и RW).

Основные функции:

1. LiquidCrystal()

Создание переменной типа liquidcrystal. Управление ЖК осуществляется по 4− х или по 8− проводной схеме. В первом случае необходимо проигнорировать параметры d0− d3, оставив соответствующие выводы не подключенными. Вывод RW можно также не подключать к Arduino и соединить его напрямую с землей. В этом случае параметр rw в функции можно не указывать.

2. Begin()

Инициализация интерфейса для согласования с ЖК и установка размеров (ширина и высота) области вывода экрана. При работе с ЖК, функции begin() полагается вызываться первой и предшествовать другим командам из библиотеки liquidcrystal.

3. Clear()

Очищение ЖК и перемещение указателя в верхний левый угол.

4. SetCursor()

Установка позиции указателя на ЖК, т.е. место, в которой будет выводиться последующий текст.

5. Write()

Вывод символов на ЖК− дисплей.

6. Print()

Вывод текста на ЖК− дисплей.

7. Cursor()

Отображение на ЖК курсор: символ подчеркивания в том месте, где будет выводится следующий символ.

8. Display()

Включение ЖК после того, как он был выключен функцией nodisplay(). После выполнения этой функции на экране отобразится текст (и курсор), которые были на нем до выключения.

9. NoDisplay()

Выключение ЖК. Текст, отображаемый на экране, сохраняется в памяти.

10. leftToRight()

Установка по умолчанию режима вывода текста на ЖК слева− направо. Это значит, что выводимые на экран символы будут появляться слева направо, не удаляя предыдущий текст.

11. rightToLeft()

Установка режим вывода текста на LCD справа− налево. Это значит, что выводимые на экран символы будут появляться справа налево, не удаляя предыдущий текст.

2) Библиотека − LiquidCrystal_I2C − #include <LiquidCrystal_I2C.h>

Неофициальная версия библиотеки LiquidCrystal с особенностями, например, упрощающие подключение, минимизация использованных выводов и простое программирование. Данная библиотека содержит те же функции, что и стандартная, но есть отличие в инициализации дисплея. Если в стандартной библиотеке требовалось установить параметры дисплея, где первое число — адрес устройства, остальные — назначаются управляющими пинами переходника под конкретный дисплей, то в LiquidCrystal_I2C эта проблема отсутствует достаточно указать адрес устройства и определенный тип дисплея, 16 символов в 2 строки.

Единственный минус библиотеки так это необходимость всегда писать название правильно иначе компилятор Arduino IDE не воспримет ее и посчитает ошибкой в синтаксисе.

Несмотря на это выбор пал на библиотеку LiquidCrystal_I2C ввиду того, что важным критерием было быстрота освоения и отсутствие большого количества подключаемых проводов.

4 Разработка циклограммы учебного стенда «Автоматизированный складской комплекс»

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

1. Движения:

Для выполнения одного цикла работы стенда «Автоматизированный транспортно-складской комплекс» необходимы следующие движения (переходы):

– движение транспортной ленты;

– пересечение деталью линии оптопары;

– остановка движения транспортной ленты;

– нажатие скользящего контакта на направляющей схвата манипулятора;

– разжимание схвата подъемного манипулятора;

– опускание схвата подъемного манипулятора;

– сжимание схвата подъемного манипулятора;

– поднятие схвата подъемного манипулятора;

– перемещение подъемного манипулятора на позицию над датчиком цвета(позиция2);

– остановка подъемного манипулятора;

– определение цвета детали с помощью датчика цвета;

– вращение барабанного склада на заданное количество шагов в соответствии с определенным цветом;

– перемещения подъемного манипулятора на позицию над барабанным складом(позиция3);

– разжимание схвата подъемного манипулятора;

2. В формировании заданного цикла участвуют следующие механизмы:

Подъемного манипулятора

– схват (зажим),

– механизм опускания\поднятия схвата,

– механизм перемещения подъемного манипулятора по направляющей;

Транспортной ленты

– оптопара (датчик линии),

– механизм движения транспортной ленты,

– датчик цвета;

Барабанного склада

– механизм поворота склада.

3. Исходное положение оборудования и его механизмов:

– детали находятся на транспортной ленте в произвольном порядке;

– начальное положение подъемного манипулятора задается относительно места крепления оптопары (позиция1);

– схват подъемного манипулятора разжат и находится на максимально допустимой высоте;

– барабанный склад находится в нулевой позиции.

На основе этого разработана циклограмма, изображенная на рисунке 4.1

Оборудование	Исполнительный механизм	Состояние	Такты										Описание тактов
			1	2	3	4	5	6	7	8	9	10	
Подъёмный манипулятор	схват (зажим)	Сжат				+	+	+	+	+			1) Начальное положение подъемного манипулятора задается относительно места крепления оптопары(позиция1), схват подъемного манипулятора разжат и находится на максимально допустимой высоте, барабанный склад находится в нулевой позиции(позиция 0); 2)Линия оптопары пересечена. 3)Механизм опускания/поднимания схвата опускается чтобы взять деталь. 4)Схват сжимается . 5)Механизм опускания/поднимания схвата поднимается. 6)Манипулятор перемещается на позицию 2. 7)Датчик цвета определяет цвет. Барабанный склад поворачивается на заданное количество шагов в зависимости от определенного значения цвета. 8) Манипулятор перемещается на позицию 3. 9)Схват разжимается и деталь падает в ячейку барабанного склада в соответствии с ее цветом. 10)Начальное положение системы.
		Разжат	+		+						+	+	
	механизм опускания\поднимания схвата	Поднят	+				+	+	+	+	+	+	
		Опущен			+	+							
	механизм перемещения подъемного манипулятора по направляющей	Позиция1	+		+	+	+					+	
		Позиция2						+	+				
		Позиция3								+	+		
Транспортная лента	Оптопара	Есть сигнал		+	+	+							
		Нет сигнала	+				+	+	+	+	+	+	
	Механизм движения транспортной ленты	Включен	+									+	
		Выключен			+	+	+	+	+	+	+		
	Датчик цвета	Включен						+	+				
		Выключен	+		+	+	+			+	+	+	
Барабанный склад	механизм поворота склада	Позиция 0	+	+	+	+	+	+				+	
		Позиция1							+				
		Позиция2											
		Позиция3											

Рисунок 4.1 – Циклограмма

5 Разработка алгоритма и блок− схемы работы учебного стенда

5.1 Разработка алгоритма

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

1) Начальное положение кубика задается на транспортной ленте вручную;

2) Производится подключения питания от сети 220 В через компьютерный блок питания, далее контроллер Arduino подключается через зарядный блок питания мобильного телефона (5 В). При подключении стенда к сети включаются в работу: транспортная лента, дисплей, датчик цвета, оптопара;

3) Кубик перемещается по движущейся ленте до точки работы манипулятора, где факт наличия детали определяется с помощью оптопары, работающей в отражающем режиме;

4) По изменению интенсивности светового потока, приходящего на фотоприемник, система управления, собранная на микроконтроллере ATmega2560 определяет факт прихода заготовки и выдает команду управления подъемом манипулятора, а так же отключения транспортной ленты и диодов;

5) Если кнопка, находящаяся на нижней части направляющей не нажата, то схват подъемного механизма разжимается и поднимается до кнопки. При нажатии кнопки подъемный механизм опускается до середины кубика. Далее, схват зажимает кубик и поднимается вверх на заданную высоту;

6) При достижении схватом определенной высоты, в работу включается шаговый двигатель, который перемещает подъемный манипулятор до датчика цвета, закрепленного в центре над транспортной лентой;

7) После того, как датчик цвета определил цвет кубика, его название выводится на дисплее, который располагается на конструкции рядом с датчиком цвета;

8) Подъемный манипулятор перемещается до барабан− склада;

9) В зависимости от цвета кубика барабан– склад поворачивается на определенный угол;

10) Схват разжимается и кубик падает в ячейку своего цвета;

11) Подъемный манипулятор и барабан– склад возвращаются в начальное положение, и в работу включаются диоды и транспортная лента;

12) Цикл продолжается, пока барабан– склад не будет заполнен всеми кубиками;

13) После заполнения барабан– склада всеми кубиками они заново выкладываются вручную на транспортную ленту.

5.2 Используемые переменные

Для того, что бы реализовать блок– схему, а в дальнейшем программный код управления учебным стендом необходимо определить переменные.

– Переменные для ШД1

Двигатель ШД1 управляется за счет драйвера L298N, на который поступают сигналы от микроконтроллера Arduino Mega 2560.Так как в самом начале программы требуется настройка режима работы двигателя, а библиотека выбранная для него является AccelStepper, то первым делом необходимо обозначит режим работы двигателя.

1) #define HALFSTEP 8

#define – директива, позволяющая дать имя константе перед тем как программа будет скомпилирована. Определенные этой директивой константы не занимают программной памяти, поскольку компилятор заменяет все обращения к ним их значениями на этапе компиляции, соответственно они служат исключительно для удобства программиста и улучшения читаемости текста программы.

HALFSTEP 8 указывает на полушаговый режим: 8– шаговая управляющая сигнальная последовательность. Данный режим считается рекомендованным.

При подключение ШД1 используются 4 вывода, обозначенные переменными:

2) #define motorPin1 24

3) #define motorPin2 28

4) #define motorPin3 26

5) #define motorPin4 30

После каждой переменной расположено двухзначное число, являющимся номером порта на плате Arduino Mega 2560.

После того, как переменные выводов объявлены, остается прописать переменную, благодаря которой будет происходить управление шаговым двигателем.

6) stepper1

Данная переменная позволяет осуществлять управление двигателем посредством необходимых функций, прописанных после stepper1.Например, stepper1.setSpeed(350) указывает на то, что после того как компилятор прочитает данную строчку, скорость двигателя примет значение 350.

– Переменные для ШД2

Двигатель ШД2 также управляется за счет драйвера L298N, но программирование совершается без библиотеки. С помощью функции digitalWrite для каждого вывода микроконтроллер отправляет сигнал драйверу, а он на обмотки двигателя подает определенные значение HIGH или LOW. Для полного оборота надо совершить несколько циклов по 4 шага поэтом чередую значения в правильном порядке происходит вращения ШД2. Перед каждой сменой обмоток прописана следующая переменная:

1) pause

Данная переменная является задержкой для того, чтобы регулировать скорость двигателя.

– Переменные для ШД3

ШД3 программируется при поддержке библиотеки Stepper, в которой можно опустить объявление переменных и прописать их в объявление уже самого класса.

1) stepsPerRevolution

Переменная необходима для определения количества шагов ШД3 за один оборот.

2) myStepper

Эта переменная указывает программе под каким "именем" находится двигатель и с легкостью управляет им. Но также как и в библиотеке AccelStepper после myStepper необходимо прописать определенную функцию для управления двигателем.

- Переменные для С

Управление происходит посредством библиотеки Servo, позволяющей осуществлять программное управление сервоприводами. Для этого заводится переменная типа Servo, в данном случае:

1) myservo2

- Переменные ДЦ

Датчик цвета имеет 5 выводов, для которых прописываются следующие переменные:

1) s0

2) s1

3) s2

4) s3

5) out

Все эти переменные необходимы для определения цвета куба, путем считывания длины сигнала на заданном порту.

Кроме того, для точного определения цвета куба объявляется следующая переменная:

6) float_color

В данную переменную сохраняются значения цветов куба, например, для красного значение 0, для синего 1, а для зеленого 2.Так как необходимо правильное определение цвета без погрешностей, то ДЦ считывает цвет куба 10 раз. Несомненно имеет место быть погрешности вследствие плохого освещения, поэтому программа может считывать данный куб как зеленым, так и синем, в худшем случае не определить цвет, тогда программа укажет на красный цвет (автоматически). Пример, дан зеленый куб, программа 10 раз считает цвет, допустим, сначала покажет зеленый – 2, потом синий – 1, зеленый – 2, красный – 0 и так до 10 раз. Все считанные значения цветов складываются и делятся на 10. В итоге получится среднее значение цвета, которое в дальнейшем будет сравниваться с тем диапазоном чисел, в котором находится определенный цвет. После того как программа даст окончательный вердикт номер куба, переменная которого ниже, выведется на экране ЖК.

7) int_color

Эта переменная служит для того, чтобы проще было корректировать точно определения цвета.

8) red

9) green

10) blue

Данные переменные служат для хранения значения цветов кубов. Так же они заносятся в массив для удобства вывода на ЖК. В них записывается значение, которое получается в результате считывания длины сигнала на определенном порту в зависимости от цвета. После чего, программа проверяет, какой сигнал длиннее для того, чтобы определить цвет.

– Переменные ТЛ

Транспортная лента приводится в движение благодаря электрическому двигателю постоянно тока 6 В. Действие ее не отличается особой сложностью, поэтому переменная не требуется. Нужно всего лишь воспользоваться функцией pinMode() и прописать в ней номер порта платы и режим

подключение OUTPUT или INPUT. В данном случае функция выглядит так pinMode(4, OUTPUT).

– Переменные ЖК

Для ЖК используется неофициальная библиотека, упрощающая некоторые функции программирования. В виду этого требуется всего 1 переменная, указывающая на обращение программы к ЖК.

1) lcd

– Переменные КП

Концевой переключатель или кнопка был установлен для ограничения диапазона линейного движения манипулятора при движении вверх. Он сигнализирует о приближении схвата к краю рабочей поверхности, чтобы предотвратить выход его за рамки конструкции. В таком же случае как и ТЛ, переменная не требуется. При проверке нажата кнопка или нет, можно воспользоваться функцией digitalRead(6), в скобках которой указывается номер порта платы для считывания. Функция считывает значение с заданного входа, если кнопка нажата HIGH, иначе LOW.

– Переменные ОП

Оптическая пара нужна для определения расстояния от куба до ОП. Т.е. если куб находится далеко до ОП, то программа сообщит об отсутствии куба, иначе о присутствии. Для этого требуется переменная, хранящая значения в виде расстояния.

1) tcrt

В эту переменную заносятся данные после того как ДЦ считал значения с определенного вывода микроконтроллера, номер которого находится в следующей переменной, подключенного к ДЦ.

2) led

Переменная, обозначающая номер пина Arduino Mega, к которому подключен ДЦ.

Кроме выше описанных переменных есть дополнительные, которые необходимы для более простой реализации программного кода.

1) key1 – ключ для таймера, который имеет начальное значение false. Требуется для отключения или включения таймера.

2) timer – переменная таймера, необходимая для корректной работы ОП.

5.3 Разработка таблицы соответствия выводов и переменных

В таблице 5.1 приведен перечень переменных, номеров выводов Arduino Mega2560 и подключаемых к ним устройств. Из таблицы видно, что используется большинство цифровых выводов, чем аналоговых.

Таблица 5.1 – Соответствия выводов и переменных

Устройство	Выводы	Номер порта на плате Arduino	Переменная	
ТЛ	1	4	Отсутствует	
КП	1	6	Отсутствует	
С, управляющий схватом	1	7	myservo2	
ОП	1	11, A0	led tcrt	
ШД1, управляющий барабан – складом	IN1	24	motorPin1	stepper1 Halfstep 8
	IN3	26	motorPin3	
	IN2	28	motorPin2	
	IN4	30	motorPin4	
ШД2, управляющий манипулятором	IN4	34	input4	pause
	IN2	36	input2	
	IN3	38	input3	
	IN1	40	input1	
ШД3, управляющий подъем/опускание схвата	IN1	42	myStepper stepsPerRevolution	
	IN2	44		
	IN3	46		
	IN4	48		
ДЦ	1	A1	s1	float float_color int int_color red green blue
	2	A2	s2	
	3	A3	s3	
	4	A4	s0	
	5	A5	out	
ЖК	1	20, 21	lcd	

5.4 Разработка блок− схем

Исходя из алгоритма работы и определения переменных, общая блок− схема работа комплекса изображена на риунок 5.1.

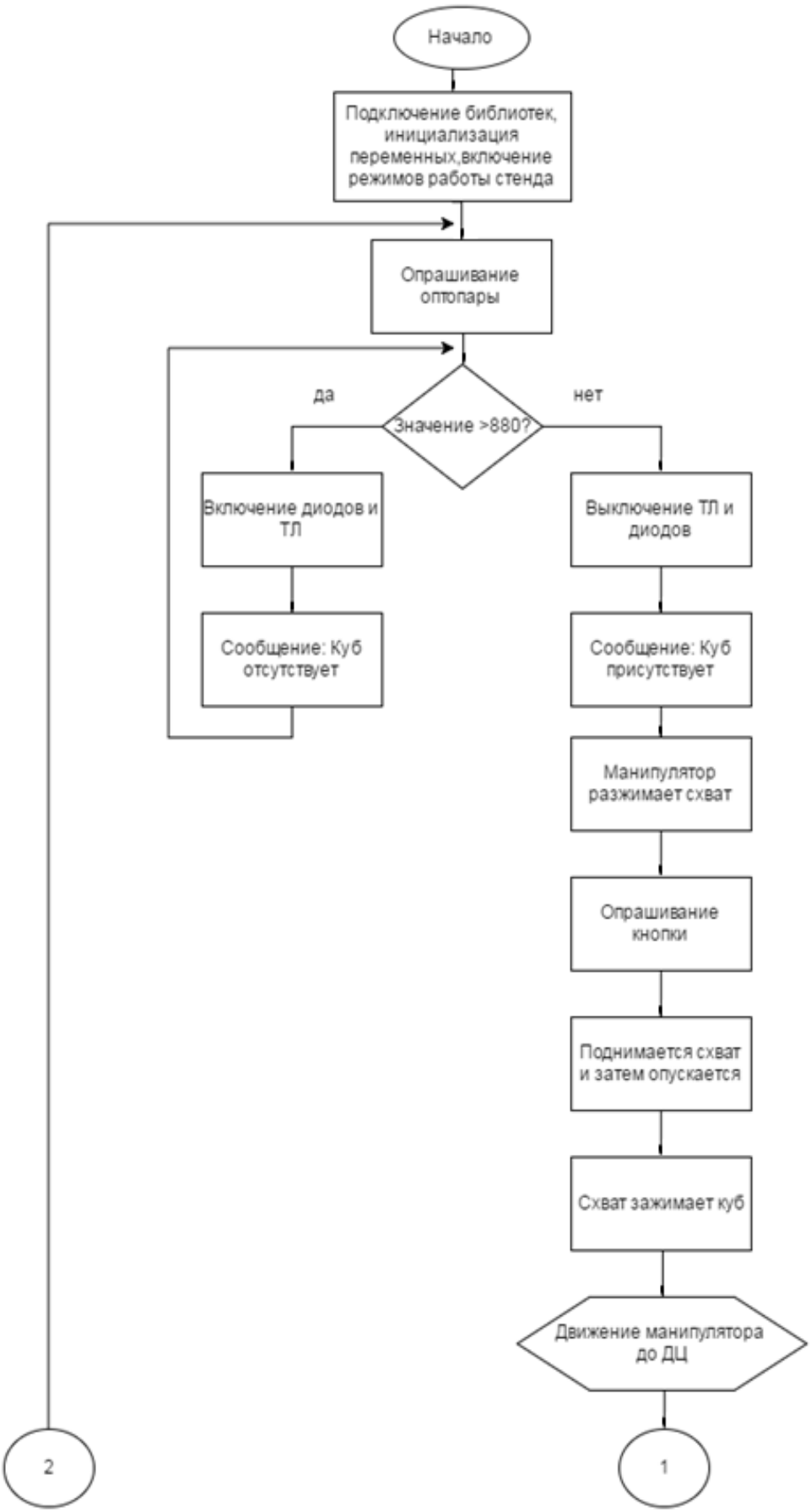
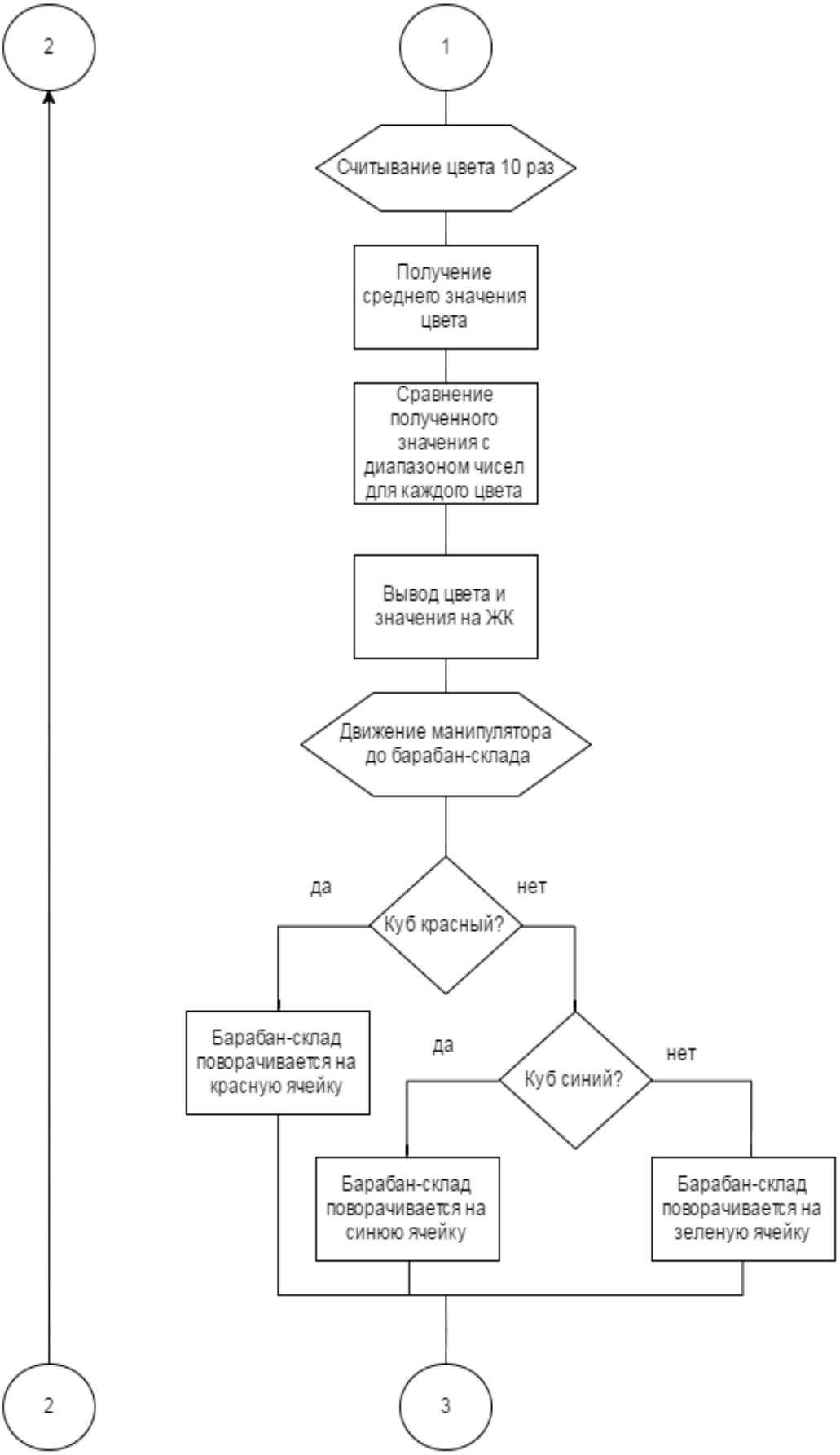
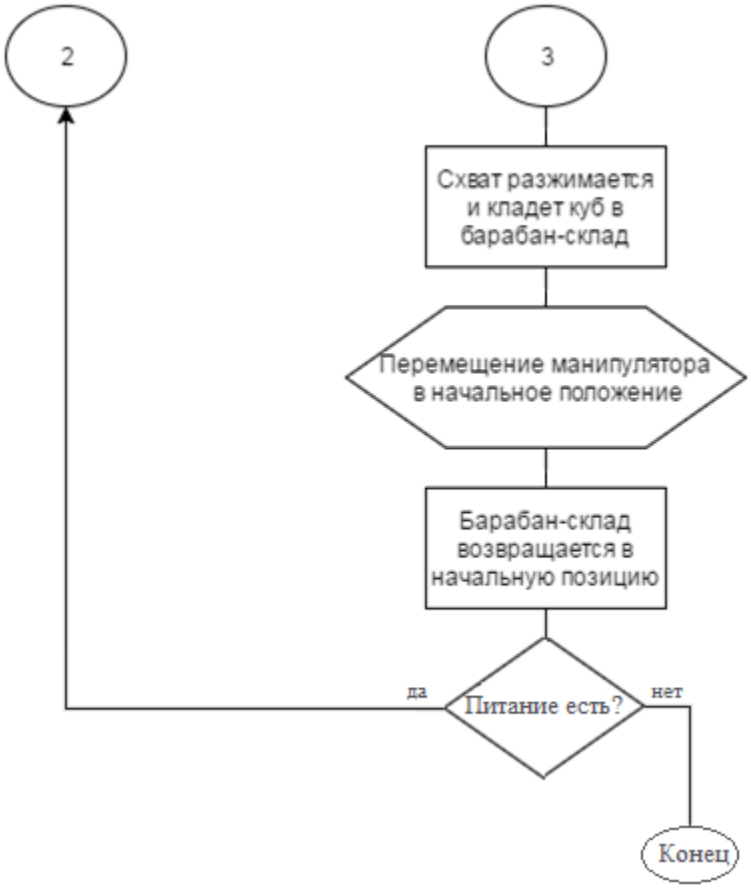


Рисунок 5.1 – Общая блок-схема учебного стена



Продолжение рисунка 5.1 – Общая блок– схема учебного стена



Продолжение рисунка 5.1 – Общая блок– схема учебного стена

Более подробная блок – схема стенда представленна на рисунке 5.2.

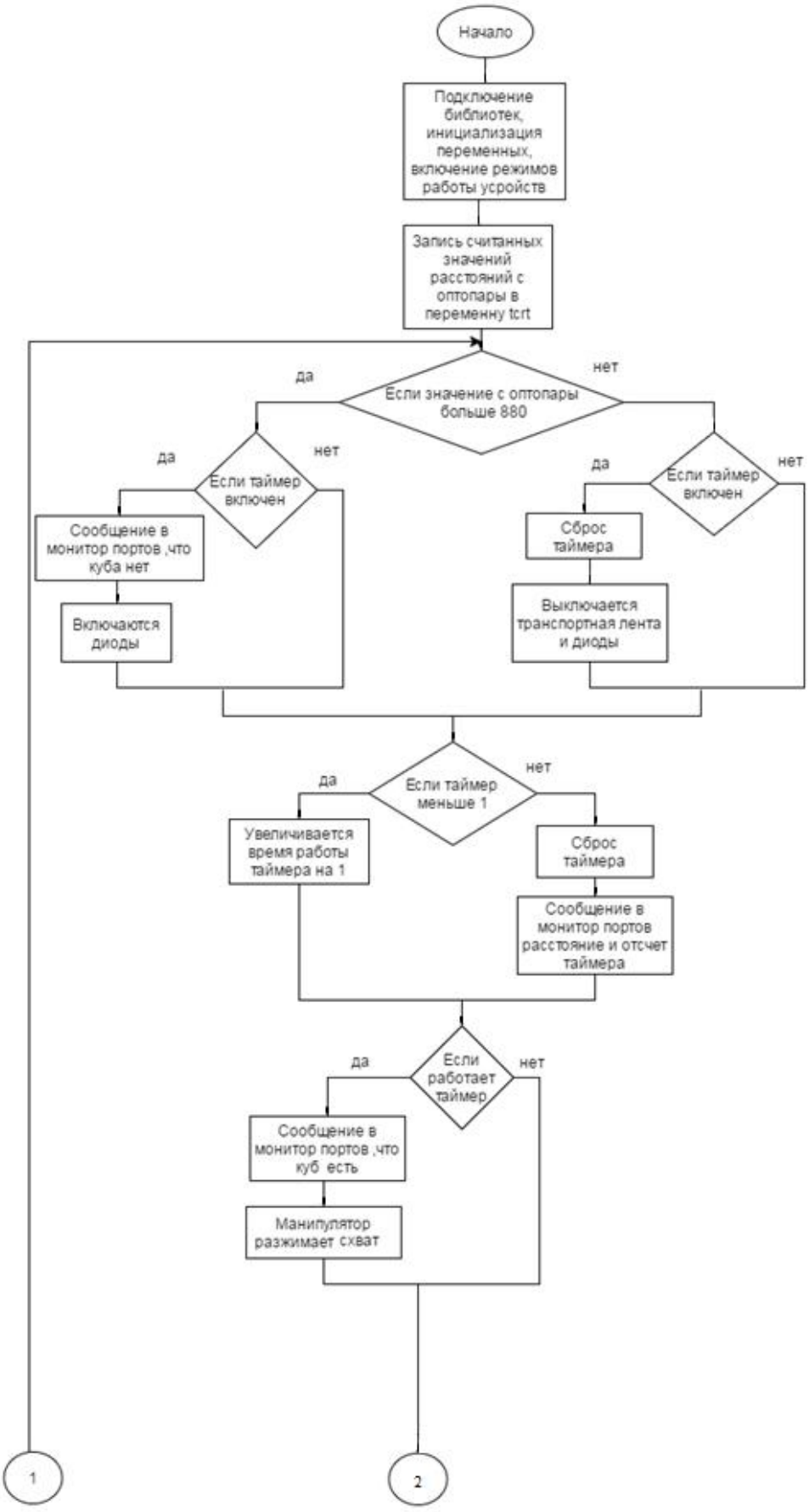
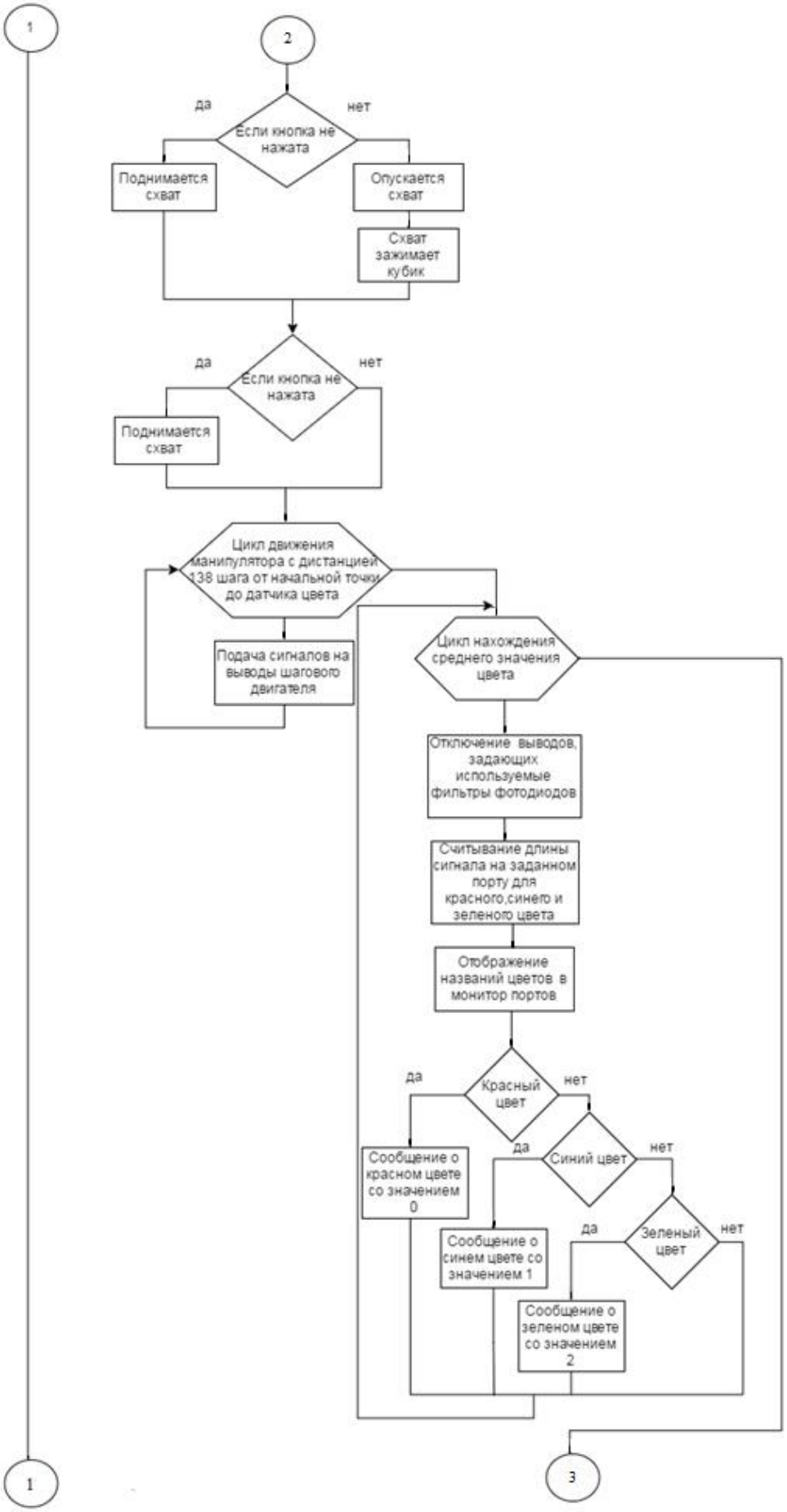
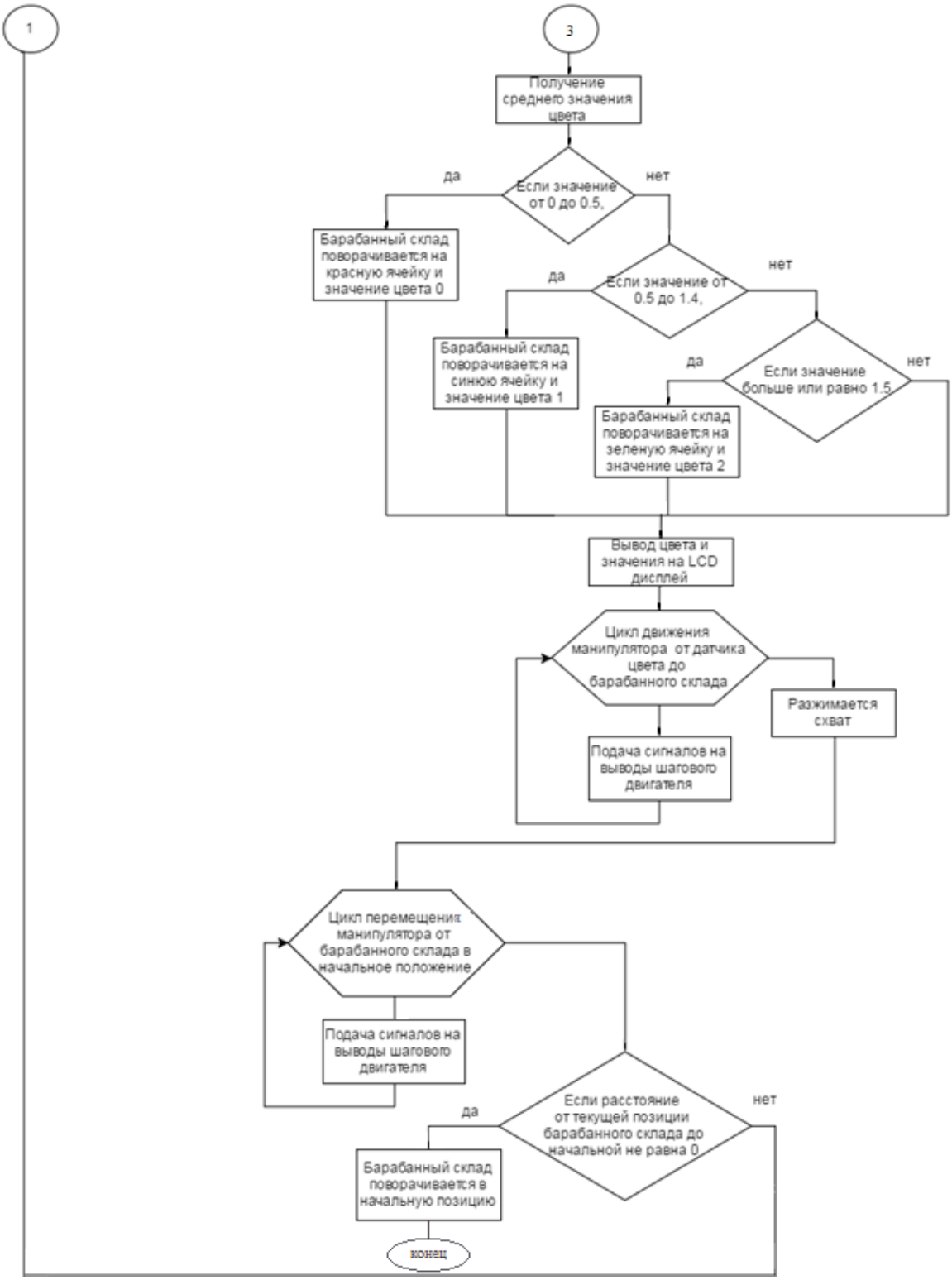


Рисунок 5.2 − Блок− схема стенда



Продолжение рисунка 5.2 – Блок– схема стенда



Продолжение рисунка 5.2 – Блок– схема стенда

Данная блок– схема подробно описывает действия во всем процессе работы "Автоматизированного транспортно– складского комплекса". Поэтому это упростило задачу разработки программного кода, который представлен в приложение 1.

Заключение

В результате проделанной работы были получены следующие результаты:

1) произведен обзор с учетом достоинств и недостатков, имеющихся на рынке микроконтроллеров;

2) исходя из требований к комплексу, выбран микроконтроллер марки Arduino Mega 2560;

3) совершено обоснование использования среды разработки Arduino IDE и языка программирования Си++;

4) произведен обзор существующих библиотек с учетом их назначения и функций для среды разработки Arduino IDE;

5) в зависимости от используемых двигателей выбраны библиотеки, упрощающие реализацию управления посредством написания программного кода:

- для шагового двигателя – ШД2, управляющим манипулятор в вертикальном положение подобрана библиотека Stepper;
- для шагового двигателя – ШД1, управляющим барабанным складом избрана библиотека AccelStepper;
- для сервопривода – С, управляющим сжатием схвата отобрана библиотека Servo;
- для жидкокристаллического дисплея – ЖК, выявлена библиотека LiquidCrystal_I2C ;
- для остальных компонентов комплекса решено программирование без помощи библиотек.

6) разработана циклограмма процессов учебного стенда;

7) на основание циклограммы выработан алгоритм действий стенда;

8) исследованы переменные, необходимые для функционирования каждого компонента комплекса;

9) сформирована таблица, содержащая перечень используемых выводов Arduino Mega 2560 со списком компонентов и относящимся к ним переменным величинам;

10) составлены две блок – схемы: общая и подробная.

Любая разработка не гарантирует отсутствие различных неполадок и проблем. При реализации программного кода были успешно решены такие затруднения как: скачкообразные сигналы с датчика оптопары, медленная работа и дребезжание шагового двигателя и другие.

Список использованной литературы

1. Баранов, В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н.Баранов – М.: Издательский дом «Додэка– XXI», 2004. – Мировая электроника) – 288 с.
2. Канцедал, С.А. Алгоритмизация и программирование : Учебное пособие / С.А. Канцедал. – М.: ИД ФОРУМ, НИЦ ИНФРА– М, 2013. – 352 с.
3. Головин, И.Г. Языки и методы программирования: Учебник для студентов учреждений высшего профессионального образования / И.Г. Головин, И.А. Волкова. – М.: ИЦ Академия, 2012. – 304 с.
4. Брукс, Ф. Проектирование процесса проектирования: записки компьютерного эксперта / Ф. Брукс; Пер. с англ. К.А. Птицын. – М.: Вильямс, 2013. – 464 с.
5. Гришин, А.В. Промышленные информационные системы и сети: практическое руководство / А.В. Гришин. – М.: Радио и связь, 2010. – 176 с.
6. Лавровская, О.Б. Технические средства информатизации. Практикум: Учебное пособие для студ. учреждений сред. проф. образования / О.Б. Лавровская. – М.: ИЦ Академия, 2012. – 208 с.
7. Гребенюк, Е.И. Технические средства информатизации: Учебник для студентов среднего профессионального образования / Е.И. Гребенюк, Н.А. Гребенюк. – М.: ИЦ Академия, 2011. – 352 с.
8. Петрова, А.М. Автоматическое управление: Учебное пособие / А.М. Петрова. – М.: Форум, 2010. – 240 с.
9. Бессонов, Л.А. Теоретические основы электротехники. Электрические цепи: Учебник для бакалавров / Л.А. Бессонов. – М.: Юрайт, 2013. – 701 с.
10. Ефимов, И.Е. Основы микроэлектроники / И.Е. Ефимов, И.Я. Козырь. – М.: Высшая школа, 1983.

11. Калашников, В.И. Электроника и микропроцессорная техника: Учебник для студ. учреждений высш. проф. обр. / В.И. Калашников, С.В. Нефедов. – М.: ИЦ Академия, 2012. – 368 с.
12. Партала, О.Н. Цифровая электроника / О.Н. Партала. – М.: Наука, 2001. – 224 с.
13. Аппаратная часть платформы Arduino [Электронный ресурс]. URL: http://arduino.ru/Hardware (дата обращения: 05.05.2016).
14. Платы и модули – Arduino [Электронный ресурс]. URL: http://amperka.ru/collection/arduino (дата обращения: 21.03.2016).
15. Arduino [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Arduino (дата обращения: 7.04.2016).
16. Что такое Arduino [Электронный ресурс]. URL: http://amperka.ru/page/what– is– arduino (дата обращения: 11.04.2016).
17. Курс «Arduino для начинающих» [Электронный ресурс]. URL: http://edurobots.ru/kurs– arduino– dlya– nachinayushhix/ (дата обращения: 3.05.2016).
18. Arduino [Электронный ресурс]. URL: http://robocraft.ru/blog/arduino/ (дата обращения: 12.04.2016).
19. Arduino: теория и практика [Электронный ресурс]. URL: http://www.cyberforum.ru/arduino/ (дата обращения: 17.05.2016).
20. Разработка– Выбираем микроконтроллер вместе [Электронный ресурс]. URL: https://habrahabr.ru/post/122030/ (дата обращения: 1.05.2016).

Приложение 1

Листинг программы автоматизированного транспортно− складского комплекса

```
#include <Servo.h> // Библиотека сервопривода, управляемая сжатием схавата
#include<AccelStepper.h> // Библиотека шагового двигателя, управляемая барабанным складом
#include <Stepper_28BYJ.h> // Библиотека шагового двигателя, управляемая опусканием/поднятием схвата
#include <LiquidCrystal_I2C.h> // Библиотека для работы с дисплеем
 #define HALFSTEP 8  // Включение полушагового режима работы шагового двигателя, управляемого барабанным складом
#define motorPin1  24 // Номер порта на плате Arduino подключаемый к порту IN1 на драйвере L298N
#define motorPin2  28 // Номер порта на плате Arduino подключаемый к порту IN2 на драйвере L298N
#define motorPin3  26 // Номер порта на плате Arduino подключаемый к порту IN3 на драйвере L298N
#define motorPin4  30 // Номер порта на плате Arduino подключаемый к порту IN4 на драйвере L298N

AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);
// Создание объекта класса шагового двигателя с указанием полушагового режима работы двигателя и подключаемых номеров портов на плате Arduino к портам на драйвере L298N
const int stepsPerRevolution = 32;  // Количество шагов шагового двигателя за один оборот
```

Stepper_28BYJ myStepper(stepsPerRevolution, 42,46,44,48); // Инициализация шагового двигателя с 32 шагами на оборот, управление обмотками через 42,46,44,48 цифровые выходы

// Выводы шаг.двигателя

```
int input1 = 40;
int input2 = 36;
int input3 = 38;
int input4 = 34;
```

Servo myservo2; // Создание объекта класса Servo для сервопривода

LiquidCrystal_I2C lcd(0x27,16,2); // Установка размерности дисплея

// Инициализация портов датчика цвета

```
const int s0 = A4;
const int s1 = A1;
const int s2 = A2;
const int s3 = A3;
const int out = A5;
```

const int led = 11; // Вывод подключения датчика расстояния

int tcrt;// Переменная для хранение данных с датчика расстояния

int pause = 5;// Переменная паузы между коммутациями шагового двигателя

bool key1 = false; // Ключ для таймера с начальным положение false

float timer = 0; // Переменная таймера

float float_color = 0; // Переменная значения цвета

int int_color =0; // Переменная цвета

char* color_text[] = {"red","blue","green"}; // Массив цветов

// Переменные, хранящие значения цветов

```
int red = 0;
int green = 0;
int blue = 0;
```

```
        void setup()
 {
  stepper1.setMaxSpeed(300.0); // Установка максимальной скорости работы барабанного склада
stepper1.setAcceleration(200.0); // Установка ускорения барабанного склада
stepper1.setSpeed(350); // Установка начальной скорости барабанного склада
myStepper.setSpeed(600); // Установка скорости схвата
 lcd.init(); // Инициализация lcd дисплея
lcd.backlight(); // Включение подсветки дисплея
// Устанвливает режим работы датчика цвета
          pinMode(s0, OUTPUT);
          pinMode(s1, OUTPUT);
          pinMode(s2, OUTPUT);
          pinMode(s3, OUTPUT);
          pinMode(out, INPUT);
Serial.begin(9600);   // Инициируется последовательное соединение и задается скорость передачи данных
// Подача сигнала на выводы датчика цвета
          digitalWrite(s0, HIGH);
         digitalWrite(s1, HIGH);
myservo2.attach(7);  // Подключение сервопривода к порту 7
stepper1.setCurrentPosition(0); // Установка начального положения барабанного склада
pinMode(4, OUTPUT); // Подключение диода к выводу
 pinMode(led, OUTPUT); // Подключение датчика расстояния к выводу
//Подключение выводов шагового двигателя
         pinMode(input1,OUTPUT);
         pinMode(input2,OUTPUT);
         pinMode(input3,OUTPUT);
         pinMode(input4,OUTPUT);
```

```
  }

 void loop()
{
        tcrt = analogRead(A0); // Сохранение в переменную считываемого
значения из выхода датчика расстояния
        delay (50);// Частота срабатывания датчика

        if (tcrt > 880) // Если расстояние большее 880
{
         if (!key1) // Таймер включен
{
Serial.print("Kybica net:"); // Сообщение в монитор портов ,что куба нет
digitalWrite(4, HIGH);// Включаются транспортная лента
 }
 }
        else // Иначе расстояние меньше 880
{
         if (!key1) timer = 0;// Сброс таймера
         key1= true;// Сброс ключа таймера
        digitalWrite(4, LOW); // Выключается транспортная лента
}
if (timer < 1) // Если таймер меньше 1
timer += 1; // Увеличивается время работы таймера на 1
else  key1 = false; // Иначе сбросить ключ таймера
Serial.println(tcrt); // Сообщение в монитор портов расстояние
Serial.println(timer); // Сообщение в монитор портов отсчет таймера
if (key1) // Если работает таймер

{
```

```
        Serial.print("Kybic est:"); // Сообщение в монитор портов ,что куб есть
        myservo2.write(0);// Разжимается схват
        delay(500); // Время для выполнение действия схвата
if (!digitalRead(6)) //Если кнопка отпущена(не нажата)
{
        myStepper.step(992); // Поднимается  схват
}
delay(1000);// Пауза после поднятия схвата
myStepper.step(− 1000);// Опускается схват
delay(500); // Время для опускания схвата
myservo2.write(100);// Схват зажимает кубик
delay(500); // Время для взятия кубика
if (!digitalRead(6)) //Если кнопка отпущена(не нажата)
{
        myStepper.step(992);//Поднимается схват
}
        delay(1000); );// Пауза после поднятия схвата
//Цикл движения манипулятора с дистанцией 138 шага от начальной точки до
датчика цвета
        for (int i=0;i<138;i++)
{
         digitalWrite(input1,HIGH);// пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,HIGH);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
```

```
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,HIGH);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,HIGH);//  пин 4
        delay(pause);
  }
// Цикл нахождения среднего значения цвета( для высокой точности
определения цвета)
for (int i = 0; i < 10; i++)
{
        color(); //Вызов программы цвета
         //Отображение значений цвета в серийный монитор последовательного
порта
                  Serial.print("RED :");
                 Serial.print(red, DEC);
                 Serial.print(" GREEN : ");
                  Serial.print(green, DEC);
                  Serial.print(" BLUE : ");
                 Serial.print(blue, DEC);
                  Serial.println();
  //Проверка обнаружения красного цвета
        if (red < blue && red < green && red > 20)
 {
         Serial.println("RED");  // Сообщение о красном цвете
```

```
 }
        // Проверка обнаружения синего цвета
         else if (blue < red && blue < green)
 {
         Serial.println("BLUE");  // Сообщение о синем цвете
        float_color += 1; // Увеличение значения цвета на 1
 }
         // Проверка обнаружения зеленого цвета
         else if (green < red && green < blue)
 {
         Serial.println("GREEN");  // Сообщение о зеленом цвете
         float_color += 2; // Увеличение значения цвета на 2
 }
 Serial.println();// Отображение цвета
 delay(100); // Пауза
}
float_color /= 10.0;// Получение среднего значения цвета
if (float_color < 0.5) // Если значение от 0 до 0.5,то красный цвет
{
    int_color = 0; // Числовое определение красного куба
     stepper1.runToNewPosition(− 930); // Барабанный склад поворачивается на
ячейку красного цвета
}
        if ((float_color >= 0.5)&&(float_color < 1.3)) // Если значение от 0.5 до
1.4,то синий цвет
{
            int_color = 1; // Числовое определение синего куба
       stepper1.runToNewPosition(− 390); // Барабанный склад поворачивается на
ячейку синего цвета
}
```

```
        if (float_color >= 1.3) // Если значение больше или равно 1.5, то зеленый цвет
{

  int_color = 2;// Числовое определение зеленого куба
    stepper1.runToNewPosition (150); // Барабанный склад поворачивается на ячейку зеленого цвета
 }
        lcd.clear(); // Очистка дисплея
        delay(40); // Пауза
        lcd.setCursor(6, 0); // Установка курсора в начало первой строки
        lcd.print(color_text[int_color]); // Вывод значения цвета
        lcd.setCursor(6, 1); //Установка курсора в начало второй строки
        lcd.print(float_color); // Вывод числового значения цвета
        delay(100); // Пауза
for (int i=0;i<115;i++) // Движение манипулятора  от датчика цвета до барабанного склада
{
         digitalWrite(input1,HIGH);// пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,HIGH);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
```

```
        digitalWrite(input3,HIGH);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,HIGH);//  пин 4
        delay(pause);
}
        delay(800); // Пауза
        myservo2.write(0); // Разжимается схват
        delay(2000); // Пауза
for (int i=0;i<253;i++)// Перемещение шагового двигателя от барабанного склада
в начальное положение
{
        digitalWrite(input1,HIGH);// пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,HIGH);//  пин 4
        delay(pause);
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,LOW);//  пин 2
        digitalWrite(input3,HIGH);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
```

```
        digitalWrite(input1,LOW);//  пин 1
        digitalWrite(input2,HIGH);//  пин 2
        digitalWrite(input3,LOW);//  пин 3
        digitalWrite(input4,LOW);//  пин 4
        delay(pause);
}
         if(stepper1.distanceToGo()!= 0) // Если расстояние от текущей позиции
        барабанного склада до начальной не равна 0
        stepper1.runToNewPosition (0); // Барабанный склад поворачивается в
        начальную позицию
}
}
void color()  // Функция цвета
{
       // Отключение  выводов, задающих используемые фильтры фотодиодов
        digitalWrite(s2, LOW);
        digitalWrite(s3, LOW);
        red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  //
Считывание длины сигнала на заданном порту для красного цвета
        digitalWrite(s3, HIGH);  // Включение вывода датчика цвета
        blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  //
Считывание длины сигнала на заданном порту для синего цвета
        digitalWrite(s2, HIGH);  // Включение вывода датчика цвета
        green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  //
Считывание длины сигнала на   заданном порту для зеленого цвета
}
```