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

ИНСТИТУТ ЭНЕРГЕТИКИ И ЭЛЕКТРОТЕХНИКИ
(институт)
Промышленная электроника
(кафедра)
11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)
Промышленная электроника
(профиль)

БАКАЛАВРСКАЯ РАБОТА

на тему Манипулятор с микроконтроллерным управлением

Студент Д.Ю. Журавский _____
(И.О. Фамилия) (личная подпись)

Руководитель А.В. Прядилов _____
(И.О. Фамилия) (личная подпись)

Консультанты Н.В. Андрюхина _____
(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

Ключевые слова: рука-манипулятор, сервопривод, плата микроконтроллера, последовательный порт.

Бакалаврская работа состоит из шести основных частей, заключения и списка используемых источников. Объем пояснительной записки – 48 страниц, количество рисунков – 41, графическая часть состоит из 6 плакатов формата А1, в списке используемых источников насчитывается 25 наименований.

Предметом выпускной квалификационной работы является разработка и исследование демонстрационного стенда на основе роботизированной руки-манипулятора. Сформулированы цели и задачи проекта. Представлена структурная схема конструкции манипулятора. Во время выполнения бакалаврской работы были выбраны и досконально исследованы все задействованные элементы устройства: сервоприводы, плата микроконтроллера. Основную часть работы над проектом заняли сборка устройства и создание программного кода управления манипулятором с последующей отладкой. Особенностью разработанной программы является способ управления манипулятором с помощью мыши и клавиатуры. Программное обеспечение данного устройства состоит из двух частей: программы, загруженной на плату микроконтроллера и терминальной программы для компьютера, считывающей положение мыши и нажатия на клавиши клавиатуры. Вся информация отправляется и считывается через последовательный порт. Осуществлена оценка экологичности и безопасности исследуемого устройства. Отдельная часть бакалаврской работы дает подробную информацию о проведенном экономическом анализе и стоимости компонентов.

Abstract

Keywords: arm-manipulator, servo drives, microcontroller board, serial port.

Bachelor's thesis consists of six main parts, a conclusion and a list of references. The bachelor’s thesis consists of an explanatory note on 48 pages, including 41 drawings, the graphical part of 6 A1 format posters, and a list of 25 references.

The subject of the bachelor’s thesis is the development and investigation of a demonstration stand based on a robotic arm-manipulator. The goals and objectives of the project are formulated. A structural diagram of the manipulator design is presented. During the execution of the Bachelor thesis, all the elements of the device, such as servo drives, microcontroller board, were selected and thoroughly investigated. The main part of the project work time was spend on the assembly of the device and the creation of a program code for manipulator control with subsequent debugging. The software of this device consists of two parts: a program loaded on the microcontroller board and terminal program for the computer, reading the position of the mouse and pressing the keys of the keyboard. All information is sent and read through the serial port. Environmental and safety assessment of the investigated device was carried out. The special part of the bachelor’s thesis gives details about carried out economic analysis and component costings.

Содержание

Введение

Современный человек в 21 веке во множестве сфер деятельности, например, в промышленности, медицине, строительстве, исследовательской деятельности, быту и т. п., очень сильно стал нуждаться в таких незаменимых «помощниках», как роботы. Именно поэтому в последние годы очень сильно стала развиваться робототехника. Было разработано много разновидностей роботизированных устройств, которые максимально упрощают работу над определенными задачами, существенно повышают качество выполняемой работы и минимизируют вероятность получения производственных травм, вплоть до летального исхода. Одним из примеров подобных устройств является роботизированный манипулятор.

Рисунок 1 – Роботизированный манипулятор на производстве

Роботизированный манипулятор- это устройство, имитирующее движения, подобные движениям человеческой руки, предназначенное для реализации широкого спектра поставленных задач, от простейшего перемещения различных объектов в пространстве до проведение сложных хирургических операций. Существует огромное количество разновидностей манипуляторов, которые

различаются по числу степеней подвижности, используемыми приводами, грузоподъемностью, способом управления, методом установки, наличию возможности к перемещению и соответственно способом перемещения. Все эти критерии выбираются исходя из цели использования манипулятора. Ниже приведены основные виды манипуляторов, различающихся по конструкции:

Декартовы (картезианские) манипуляторы, представляющие собой конструкцию, состоящую из расположенных перпендикулярно друг другу осей движения. Манипуляторы данного типа обладают хорошей жесткостью конструкции из-за малого количества соединений. В тоже время недостатком таких манипуляторов является полное отсутствие гибкости, что сильно ограничивает их в движении. Исправить эти недочеты смогли инженеры создавшие манипуляторы SCARA, которые, в отличии от предшественников, обладали хорошей гибкостью по горизонтали. Описанные выше манипуляторы нашли свое применение в основном на сборочном производстве.

Рисунок 2 – Манипулятор декартового типа

Шарнирные манипуляторы представляют собой несколько звеньев, образующих полярную систему координат. Такой вид конструкции манипулятор обладает очень высокой гибкостью, что позволяет им принимать любые положения внутри рабочей зоны. Шарнирные манипуляторы применяются во многих областях: при сварке, малоинвазивной хирургии и т.д.

Рисунок 3 – Манипулятор шарнирного типа

Параллельные стержневые манипуляторы. Чаще всего представляют собой несколько кинематических цепей, который образуют схему параллелограмма. Такие устройства обладают высокой точностью, возможностью работать на высоких скоростях и огромную степень свободы. Манипуляторы такой конструкции применяются в основном при сортировке и упаковке готовых изделий.

Рисунок 4 – Манипулятор параллельного типа

Однако все описанные выше виды конструкций манипуляторов представляют собой незамкнутую цепь, соединенных между собой звеньев, которые имеют свои названия, аналогичные названиям частей руки человека (плечо, предплечье, локоть, кисть, пальцы, сустав и т.д.)

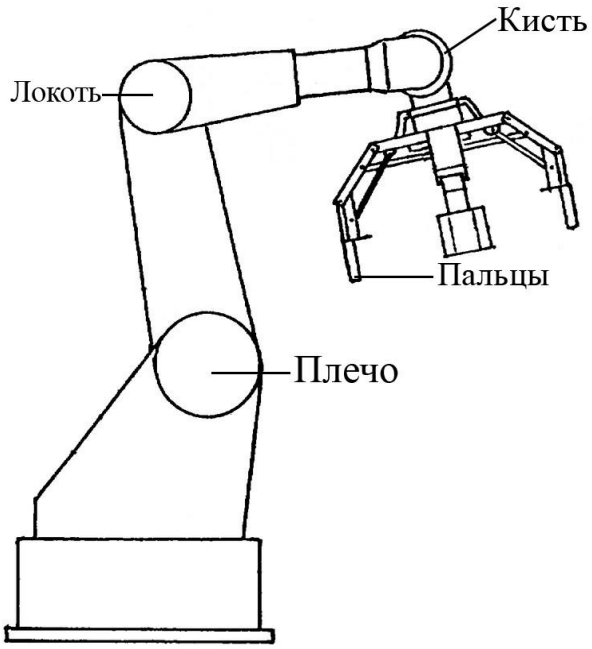

Рисунок 5 – Устройство манипулятора

На сегодняшний день особенно востребованы шарнирные манипуляторы в промышленности, где основным требование является высокая грузоподъемность. С этой задачей хорошо справляются пневматические и гидравлические манипуляторы. В тоже время манипуляторы нужны и для других задач, где не так важна грузоподъемность, но требуется высокая точность, возможность гибкой настройки и управления. Для этих целей больше подходят, не такие известные как их гидравлические и пневматические собратья, манипуляторы на сервоприводах. Данный вид манипуляторов только начал набирать популярность и пока что существует не так много готовых решений, которые соответствуют всем требованиям. В нашем проекте как раз и будет использоваться шарнирный манипулятор на сервоприводах, который мы рассмотрим поподробнее.

1. Состояние вопроса

1.1. Формулирование цели и задач проекта

Цель: Создать демонстрационный стенд роботизированного манипулятора на основе сервоприводов, разработать и реализовать способ управления манипулятором.

Задачи:

1. Проанализировать весь материал так или иначе связанный с нашей темой диплома.
2. Выделить все преимущества и недостатки существующих решений.
3. Изготовление устройства.
4. Разработка и отладка программной части устройства.

1.2. Анализ исходных данных и известных решений

На текущий момент имеется довольно малое количество готовых решений манипуляторов на сервоприводах. Все рассмотренные мною реализации имеют много общего друг с другом: примерно одинаковая конструкция с использованием разных материалов, которые влияют на общий вес устройства, используются схожие микроконтроллеры. Однако есть и небольшие различия: применены разные сервоприводы, которые влияют на максимальный рабочий вес, который может поднять манипулятор и на максимальную степень свободы. Были реализованы разные варианты захватов, от присосок до клешней. Также различаются и способами управления: с помощью потенциометров, джойстика, ультразвуковых датчиков, клавиатуры и мыши и т.д. Из всех существующих решений наиболее перспективным проектом, по моему мнению, являются манипуляторы от компании Ufactory, которая успешно получила финансирование на площадке kickstarter.com и представила уже несколько готовых моделей. Их продукция пользуется огромным

спросом из-за неимения конкурентов в их ценовой категории, а промышленные роботы стоят в десятки раз больше.

Рисунок 6 – Одна из моделей манипулятора от компании Ufactory

Команда инженеров Ufactory уже несколько лет занимаются разработкой и улучшением своих проектов, им удалось добиться очень хороших результатов. Была создана оптимальная конструкция устройства и использованы очень качественные материалы, также разработано высококачественное программное обеспечение для управления и отладки манипуляторов с открытым исходным кодом. Одним из главных достоинств данного робота является его способность к обучению, что дает огромные возможность для людей, хотя бы немного понимающих в программирование, а для тех, кто далек от этого есть возможность повторения манипулятором движений руки хозяина.

Помимо uArm на рынке существуют еще несколько вариантов манипуляторов, которые по тем или иным причинам уступают описанному выше решению, но все же заслуживают внимания: проект Dobot, имеющий уклон в сторону 3D печати, проект Ctrl, напоминающий промышленный манипулятор, уменьшенный в несколько раз. Однако у этих двух проектов есть существенный недостаток для устройств, предназначенных в основном для домашнего использования, их цена:

Dobot (1400 долларов), Ctrl (800 долларов). Существует и наоборот очень дешевый проект под названием LittleArm (60 долларов) 2C, который по своей сути является маленьким манипулятором, направленным на обучение детей основам программирования и робототехники.

Рисунок 7 – Модели манипуляторов Dobot и Ctrl

Проанализировав все рассмотренные выше решения, мы пришли к выводу, что для реализации нашего проекта нужно использовать недорогие легкодоступные компоненты, которые можно быстро и без труда поменять при необходимости, в качестве управления мы выбрали, используемый также в проекте uArm, метод контроля манипулятора с помощью мыши и клавиатуры, так как он предусматривает очень широкий спектр настраиваемых параметров, что позволит реализовать очень гибкую систему управления.

2. Проектный раздел

2.1. Разработка схемы и выбор необходимых компонентов

Чтоб создать задуманный проект, первым делом нужно разработать структурную блок схему устройства, чтобы иметь как можно более подробное представление о том какие компоненты нам нужны. Структурная блок-схема устройства приведена ниже на рисунке 11. Оформление схемы было осуществлено в системе автоматизированного проектирования КОМПАС-3D.

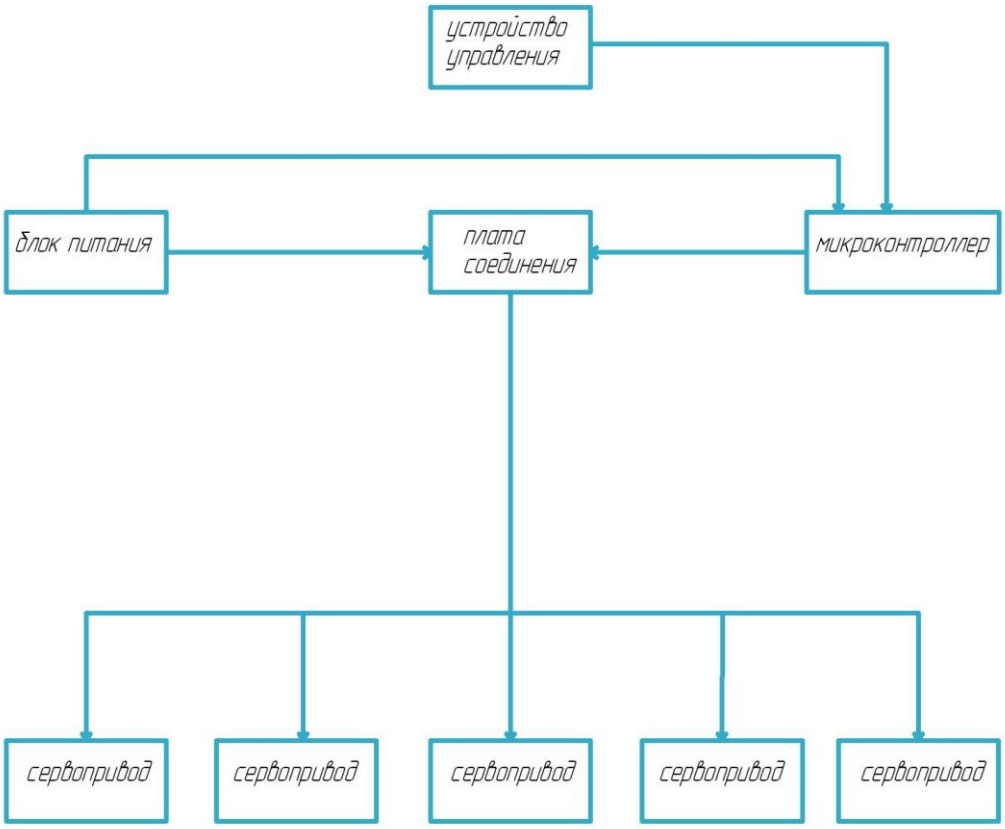

Рисунок 8 – Структурная блок-схема устройства

Для начала необходимо определиться каким же способом мы будем управлять нашим манипулятором. Так как мы хотим создать демонстрационный стенд,

которым сможет пользоваться любой человек, без каких-либо проблем с поиском специальных устройств управления, такие как джойстик, геймпад и т. п., которые не всегда есть под рукой, выбор упал на управление через компьютер с помощью мыши и клавиатуры. Исходя из того, что наш демонстрационный стенд не мобильный и ему не нужны аккумуляторы для передвижения, мы выбрали компьютерный блок питания, который идеально подходит чтобы запитать и сервоприводы, и микроконтроллер. В качестве платы микроконтроллера остановились на Arduino Mega 2560 которая полностью соответствует всем требованиям необходимым для нашего проекта. Данная модель может питаться не только через USB, но также и от блока питания. На борту данной платы стоит микроконтроллер ATmega2560 имеющий 256кБ памяти предназначенной для хранения кода программы. Так же в нашем распоряжение появляется огромное количество пинов ввода/вывода (16 аналоговых и 54 цифровых), порт USB, UART. Поддерживает большинство плат расширения, в том числе Multiservo Shield, который можно использовать в нашем стенде. Одним из главных достоинств данной платформы является известная во всем мире интегрированная среда программирования Arduino IDE. Эта среда использует упрощенный язык программирования C/C++, очень хорошо поддерживается разработчиками, имеет огромное количество библиотек, которые постоянно обновляются.

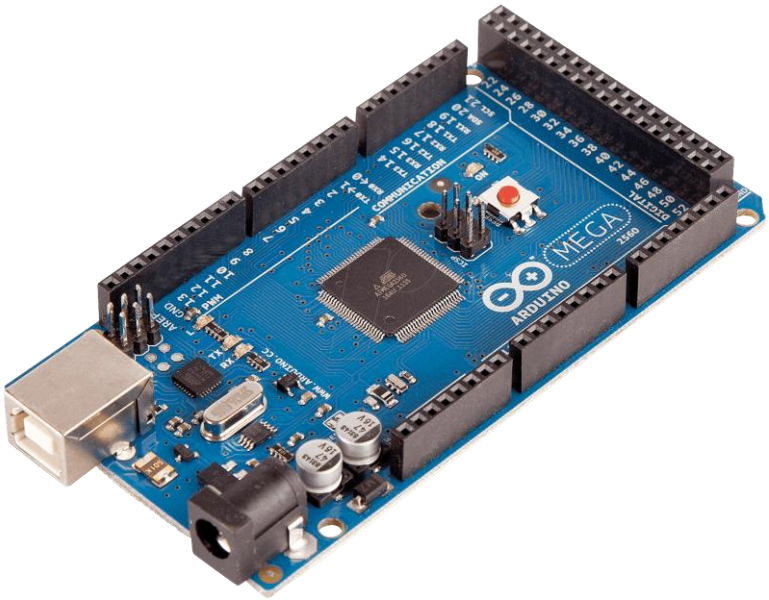

Рисунок 9 – Arduino Mega 2560

Одной из важнейших частей нашего устройства является сервопривод. Для оптимального выбора цена/качество мы рассмотрели множество вариантов. Главным критериями для выбора были: тип механизма обратной связи, в нашем случае мы сделали выбор в сторону цифрового сервопривода, металлический механизм, большой крутящий момент, наличие ударной защиты, высокая точность позиционирования, маленькая мертвая зона, большой угол вращения. Наиболее подходящим для нас оказался сервопривод Tower Pro MG996R, имеющий неплохие характеристики при довольно низкой цене: вес 55г, крутящий момент 9,4 kgf·cm (килограмм силы на сантиметр) при 4,8 В, 11 kgf·cm при 6 В, рабочий вольтаж: 5 – 7,3 В, рабочая скорость: 0,17 с / 60º (5 В), 0,14 с / 60º (6 В), механический механизм, угол вращения в 120 градусов. Для нашего проекта необходимо 5 таких сервоприводов.

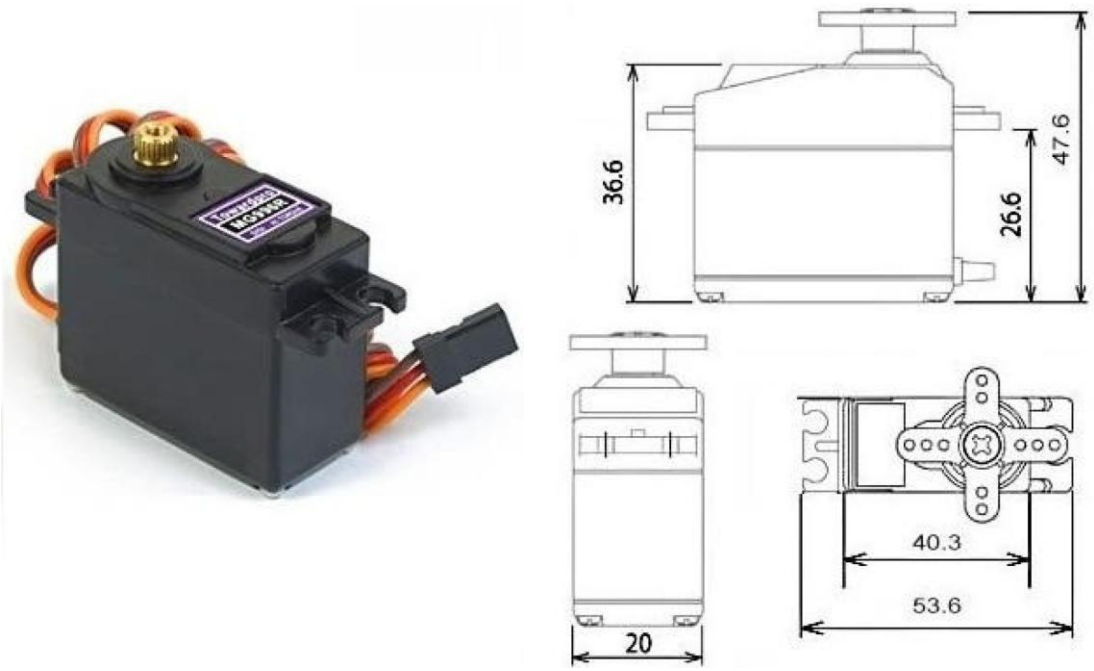

Рисунок 10 – Сервопривод Tower Pro MG996R

Полная схема электрическая принципиальная вместе с перечнем используемых элементов приведены ниже на рисунке 11. Оформление схемы было осуществлено в системе автоматизированного проектирования КОМПАС-3D.

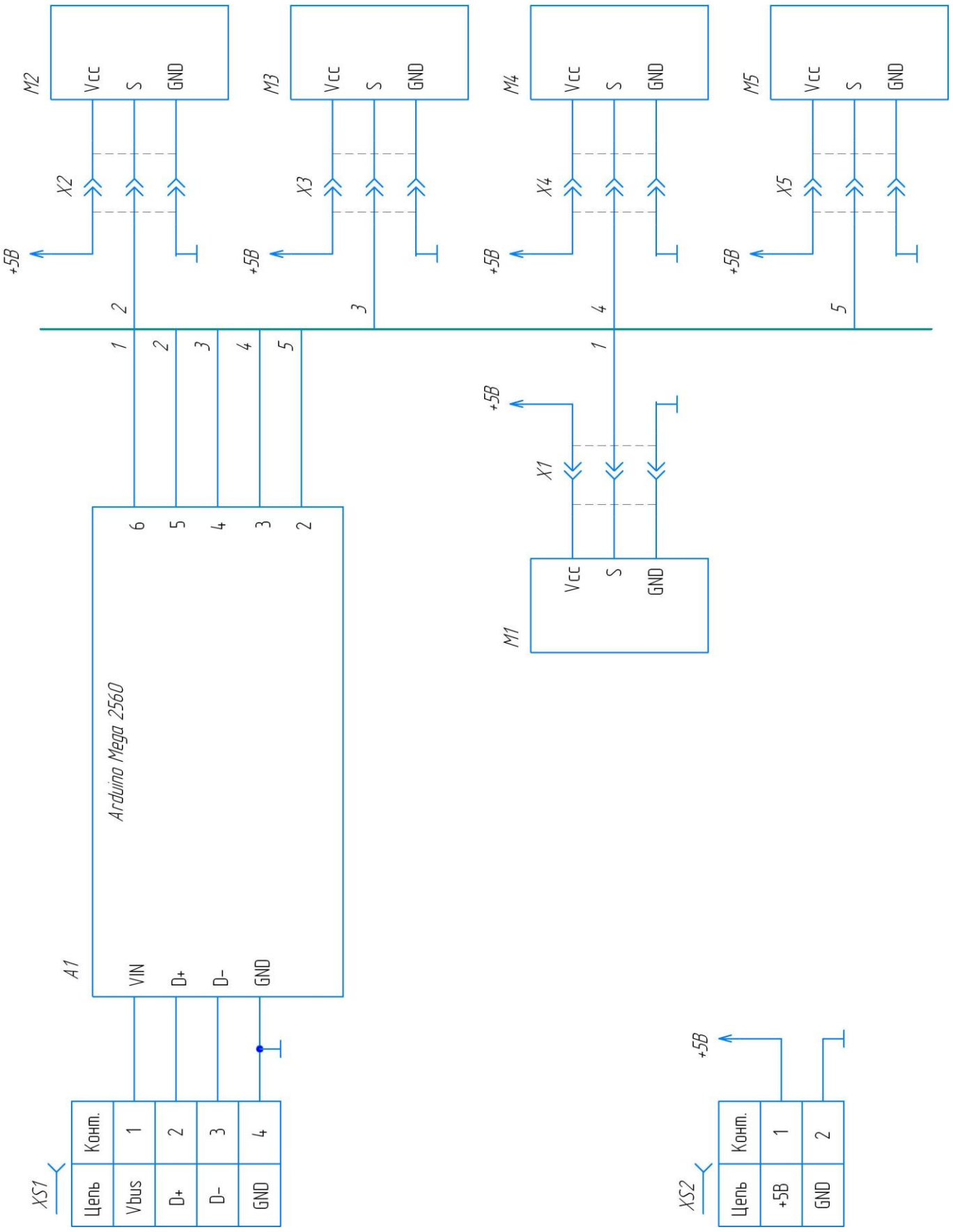

Рисунок 11 – Схема электрическая принципиальная с перечнем элементов

2.2. Разработка конструкции устройства

Так как нашей первоочередной задачей является создание демонстрационного стенда, который будет использоваться для практического изучения программирования манипуляторов, что подразумевает быстрое подключение устройства, возможность изменения конфигурации подключаемого оборудования, легкий доступ ко всем компонентам и т.д., было принято решение сделать полностью открытый стенд, где на одной платформе будет расположены манипулятор и все сопутствующие ему детали: блок питания, плата соединения, плата микроконтроллера управления. В программе Компас 3D создали примерный макет конструкции нашего стенда, отталкиваясь от которого будем изготавливать устройство.

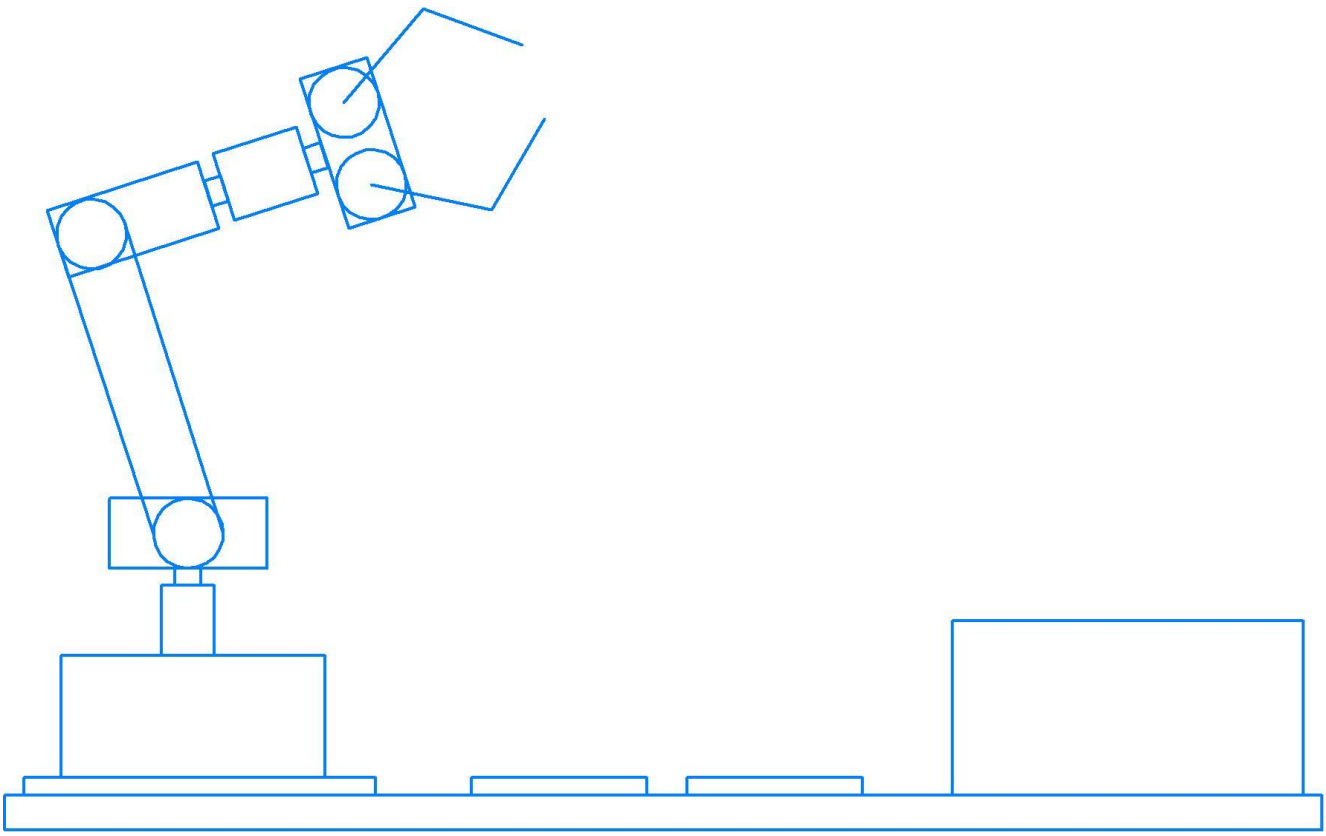

Рисунок 12 – Схема конструкции будущего устройства

2.3. Практическое изготовление устройства

Итак, приступили к сборке устройства. В качестве платформы для нашего стенда, на которой будут расположены все детали, вплоть до блока питания, мы выбрали пластину стеклотекстолита. Чтобы питать наш манипулятор и плату микроконтроллера мы взяли обычный блок питания от компьютера.

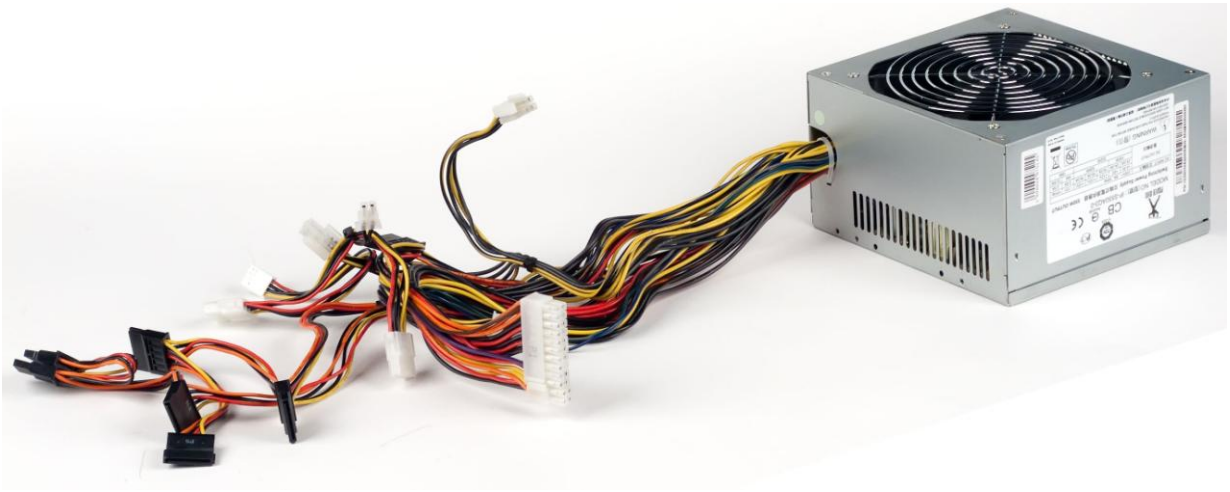

Рисунок 13 – блок питания

Избавились от лишних проводов, оставив только пару проводов GND, 5V для всех сервоприводов и 12V которые могут понадобиться в будущих проектах где может потребоваться внешнее питание для платы ардуино. Также сделали кнопку питания блока для удобного включения / отключения устройства. Закрепили блок питания на пластине предварительно покрасив все в черный цвет. Непосредственно для сборки самого манипулятора мы использовали готовые алюминиевые детали от конструктора, из-за их небольшого веса и наличия готовых отверстий для крепления.

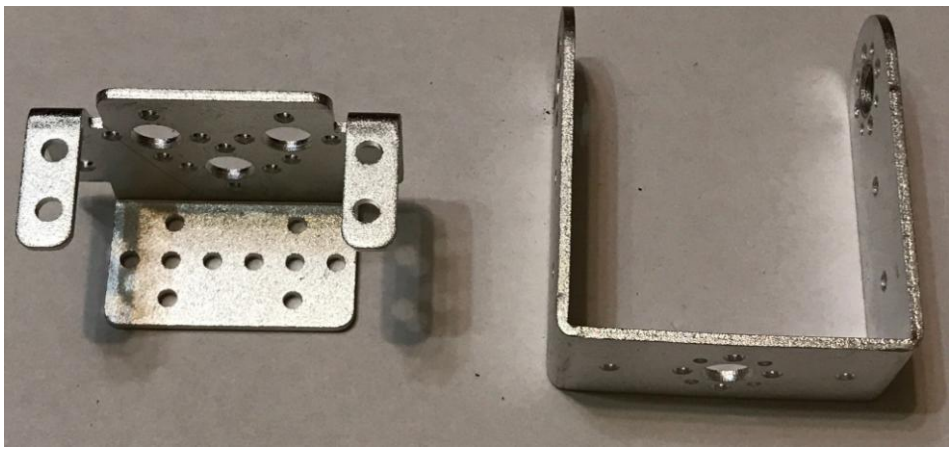

Рисунок 14 – детали каркаса манипулятора

Также мы нашли готовое решение захвата, которое имеет совместимость с нашим сервоприводом, возможность захвата предметов довольно больших габаритов и небольшую стоимость, что позволяет с легкостью поменять его в случае выхода из строя.

Рисунок 15 – захват манипулятора

Соединив все алюминиевые детали, сервоприводы и захват, собрали манипулятор. Первоначально манипулятор был собран на 6 сервоприводах, но увидев, что самый нижний сервопривод, на котором держится вся остальная конструкция, не выдерживает такой вес, было принято решение убрать один сервопривод, отвечающий за второй верхний сгиб манипулятора. Так же пришлось удлинить провода от сервоприводов и датчика чтобы их длины хватало для свободного перемещения по рабочей траектории манипулятора. Проверили каждый сервопривод на работоспособность.

Далее было принято решение создать отдельную плату, которая будет отвечать за общее питание и землю для сервоприводов. Также добавили на плату общие выводы для соединения сигнальных проводов от сервоприводов со входными пинами на плате микроконтроллера.

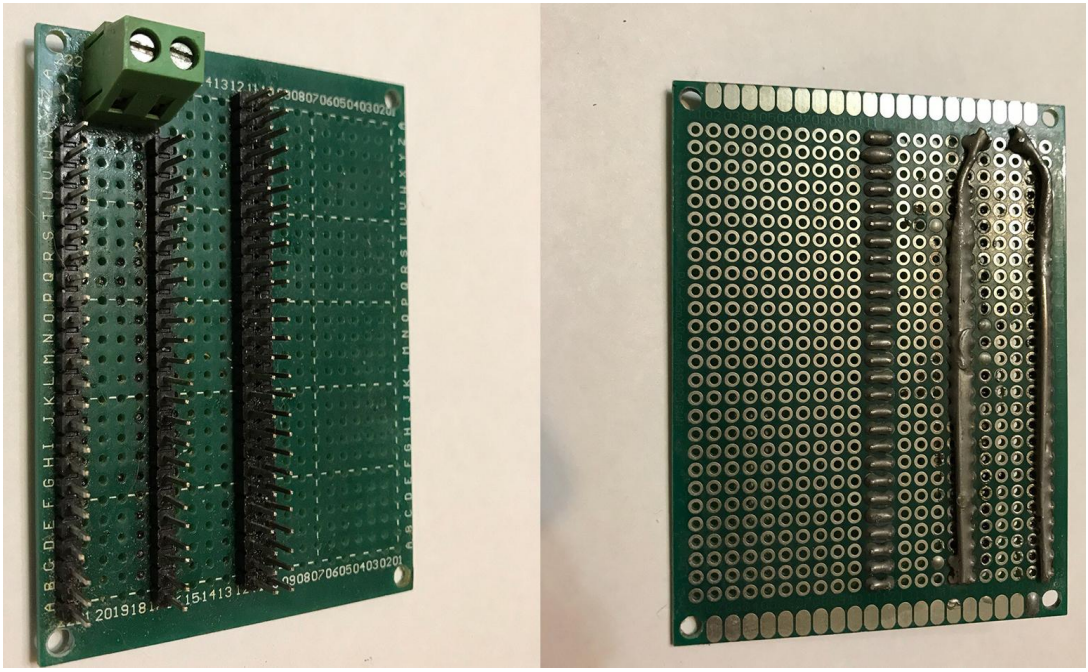

Рисунок 16 – Плата соединения

Разместили собранный манипулятор, плату соединения и плату микроконтроллера на пластине стеклотекстолита, собрали все провода от сервоприводов в один общий жгут, предварительно промаркировав каждый провод соответствующим номером сервопривода, для того чтобы не запутаться в большом количестве проводов. Подключили все провода в соответствующие разъемы. Также было решено дополнительно добавить в наше устройство ультразвуковой дальномер для возможности в дальнейших проектах с нашим стендом реализовать с помощью него управление захватом манипулятора. Сборка демонстрационного стенда завершена. Ниже будет приведено фото готового устройства.

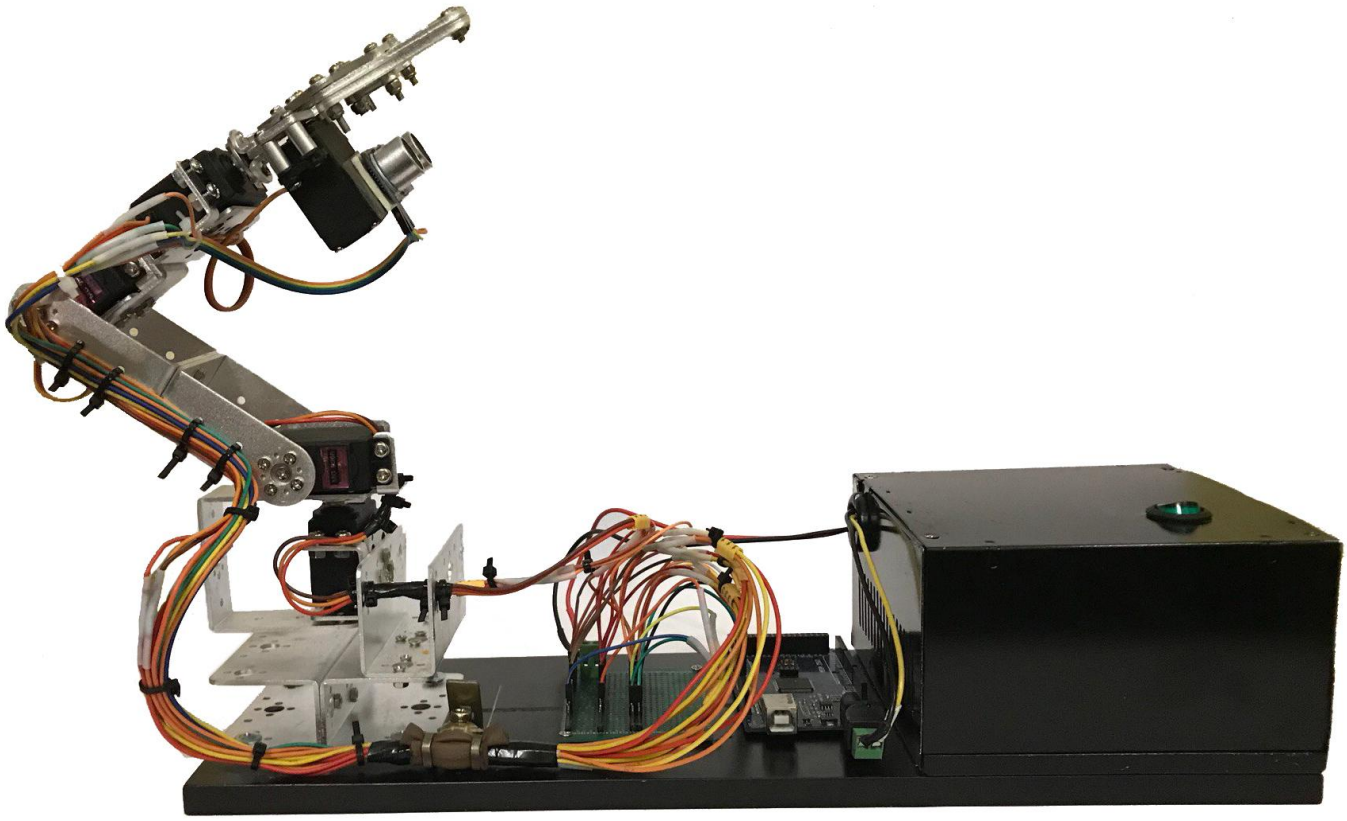

Рисунок 17 – Собранный стенд

3. Программная часть

3.1. Описание алгоритма работы устройства

Алгоритм работы устройства в двух словах представляет собой обработку данных от связанных между собой интерфейсом передачи данных микроконтроллера управления, клавиатуры, мыши и сервоприводов. Обработанные данные нормализуются и посылаются обратно на отработку сервоприводов. Для более детального описания алгоритма работы устройства необходимо рассмотреть принцип работы и устройство каждого компонента, связанного с интерфейсом передачи данных.

Сервопривод – актуатор, который и будет приводить в действие наш манипулятор. Сервопривод или сервомашинка довольно сложное устройство обратной связи по сравнению с обычным электромотором. Представляет из себя: двигатель постоянного тока, шестеренный редуктор, управляющую электронную схему и потенциометр. Шестеренный редуктор используется для понижения скорости мотора, который преобразует большой крутящий момент в более маленький из-за взаимодействия меньшей шестерни с малым количеством зубцов с большей шестерней с соответственно большим количеством зубцов. Электронная схема отвечает за поддержание обратной связи, за получение и сравнение значений со встроенного потенциометра прикрепленного к выходному валу со внешними численными значениями от других управляющих систем и за запуск и остановку мотора. Сервопривод имеет три вывода: VCC – 5 вольт (постоянный ток), GND – земля, PWM (Pulse Width Modulation)/ ШИМ (широтно-импульсная модуляция). – вывод управляющего сигнала.

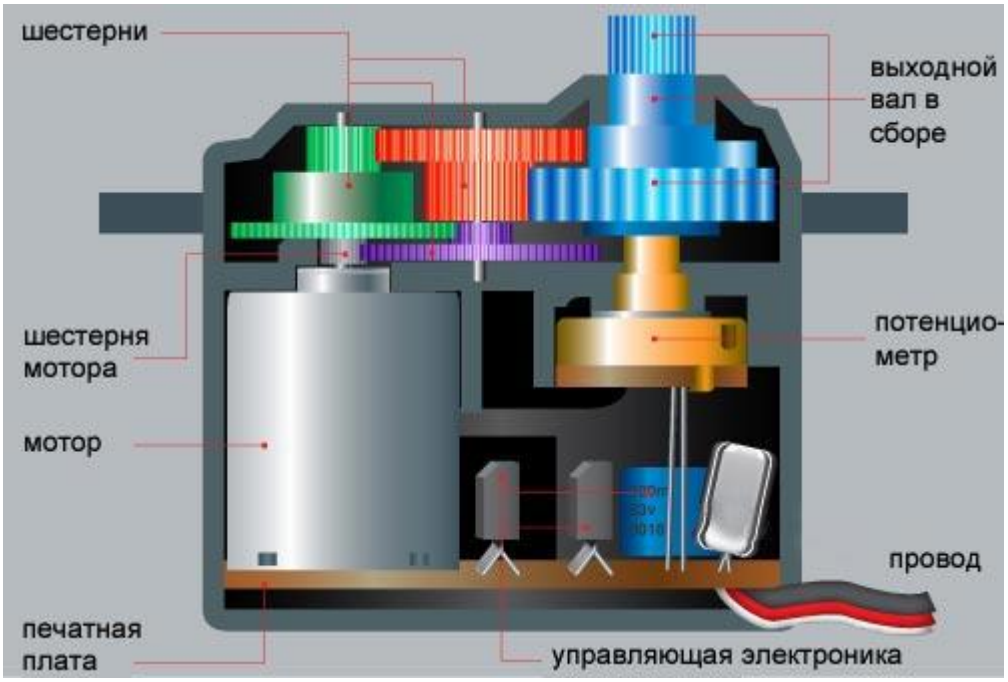

Рисунок 18 – Устройство сервопривода

Одной из основных частей сервопривода является потенциометр, который обладает 3 выводами, из них на два боковых подается питание, между каждым выводом находится резистивное вещество, которое позволяет двигаться ползунку, соединенному со средним выводом, с которого снимается значение напряжения.

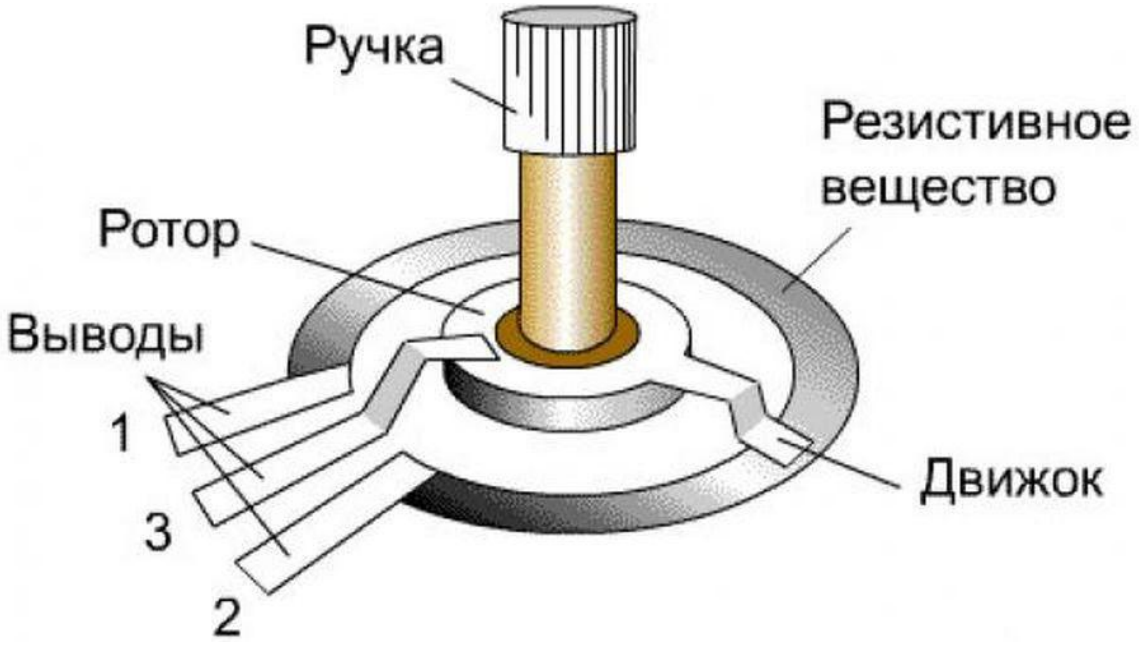

Рисунок 19 – Устройство потенциометра в сервоприводе

Вал сервопривода соединен с валом потенциометра, если мы вращаем одно, то вращается и другое. Когда мы вращаем ручку потенциометра по часовой стрелке от плюса к минусу, увеличивается сопротивление, а напряжение уменьшается. Напряжение будет уменьшаться до минимального, которое зависит от максимального сопротивления потенциометра в нашем сервоприводе, у каждого сервопривода может быть разный номинал максимального сопротивления потенциометра, однако в большинстве моделей используется потенциометры на 5 кОм. Допустим мы хотим повернуть наш сервопривод на 75 градусов, условно в этом положение на выходе потенциометра будет 2 вольта. В какой-то момент времени вал находится в положение, например, 10 градусов что условно равно 0,3 вольт. Чтобы повернуть сервопривод в желаемое положение нам нужно подать на управляющую плату управляющий импульс, который содержит в себе информацию в какое положение мы хотим повернуть вал. В дело вступает электронная начинка на управляющей плате, которая сравнивает показания напряжения на потенциометре при положении вала (0,3 вольта) с требуемыми показаниями (2 вольта), которые пришли на управляющую плату. При несовпадении показаний микропроцессор на управляющей плате принимает решение отправлять с помощью кварцевого генератора управляющие импульсы определенной длины, что позволяет вращать в нужном направление вал потенциометра, пока показания напряжения на нем не сравняются с требуемым напряжением при 75 градусах. То есть положение вала потенциометра зависит от длины импульса. Курс вращение определяется более коротким из двух импульсов. Одна часть управляющей микросхемы производит сигнал, продолжительность которого устанавливается потенциометром, а другая часть сравнивает эти два импульса.

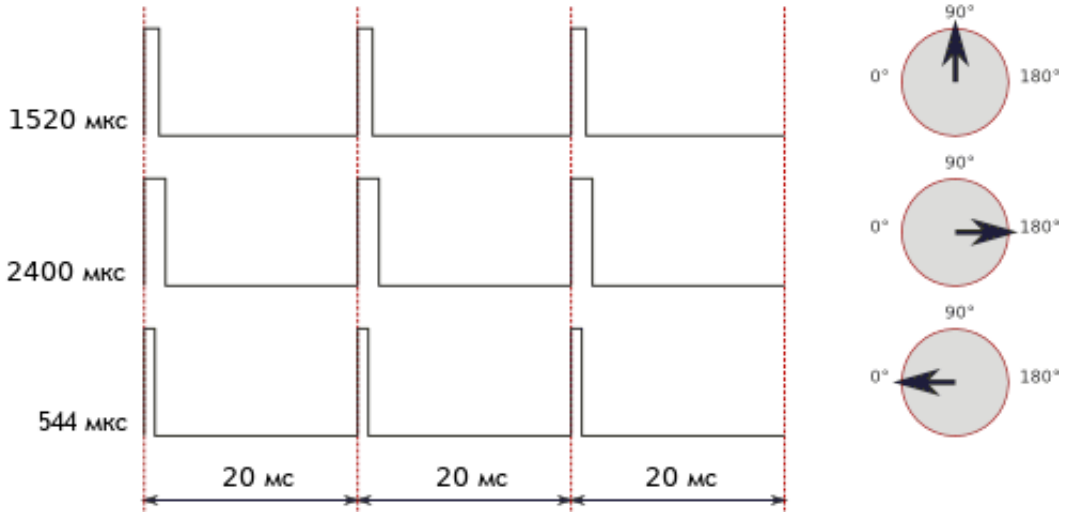

Рисунок 20 – Диаграмма управляющих импульсов сервопривода

На большинстве сервоприводов используются одинаковая частота импульсов в 50 Гц (отправка и получение импульса происходит каждые 20 миллисекунд) и одинаковые границы длительности импульсов: нижняя граница длительности импульса, когда сервопривод находится в положение 0 градусов, равна 544 микросекундам, а верхняя граница – 180 градусов равна 2400 микросекундам. Зная граничные значения, мы можем определить длительность импульса для любого положения сервопривода. Стандартная библиотека Servo.h, которая будет использоваться в нашей программе, как раз имеет общепринятые значения границ.

В тексте выше я упомянул о внешних численных значениях от других управляющих систем. Внешними численными значениями и являются параметры, которые мы можем изменять с помощью мыши и клавиатуры, так же эти параметры изменяет и ультразвуковой датчик. Для этих целей была написана специальная терминальная программа, которая производит вычисления и нормализует параметры со входа мыши и клавиатуры для дальнейшего сравнения с параметрами, заданными в константах в отдельном написанном скетче на платформе arduino. Эти внешние значения мы отправляем на сервоприводы, которые выполняют процессы, которые мы подробно описали выше.

Так же в двух словах опишем принцип работы ультразвукового дальномера HC-SR04, который может нам понадобиться в будущих проектах с нашим устройством. Ультразвуковой дальномер использует в своей работе принцип эхолокации, как у летучих мышей. На плате дальномера расположены два пъезоизлучателя, один работает как излучатель (динамик), а второй как приемник (микрофон). Для подключения датчика используется четыре вывода: VCC – 5 вольт (DC), GND – земля, Trig – Триггер (INPUT), Echo – Эхо (OUTPUT). Интерфейс связи работает следующим образом: на выводе Trig формируется короткий импульс длительностью 10-15 микросекунд, после которого на выходе создается некоторое количество сигналов частотой 40 килогерц, которые посредством динамика будут отправлены в сторону препятствия. В тот же момент времени на выводе Echo образуется логическая единица высокого уровня длинной от 150 микросекунд до 25 миллисекунд. Как только отраженная от препятствия волна дойдет до приемника на выводе Echo появится логический ноль. Расстояние до объекта прямо пропорционально ширине сигнала Echo. Другими словами, зная время присутствия логической единицы высокого уровня на выводе Echo, получиться определить расстояние до какого-либо объекта, воспользовавшись специальной формулой.

3.2. Разработка программной части устройства

Программная платформа для нашего устройства будет состоять из двух частей:

1. Терминальной программы, написанной на языке C# (C Sharp) в программе SharpDevelop 5.1, так как этот язык идеально подходит для быстрого написания настольных приложений.

2. Программы для платы управления (скетч), написанного в программе Arduino IDE, которая специально создана для написания программ для плат Arduino.

Терминальная программа запускается на PC, а скетч загружается на плату управления Arduino Mega.

Программа для платы управления представляет собой алгоритм последовательных действий, представленный ниже.

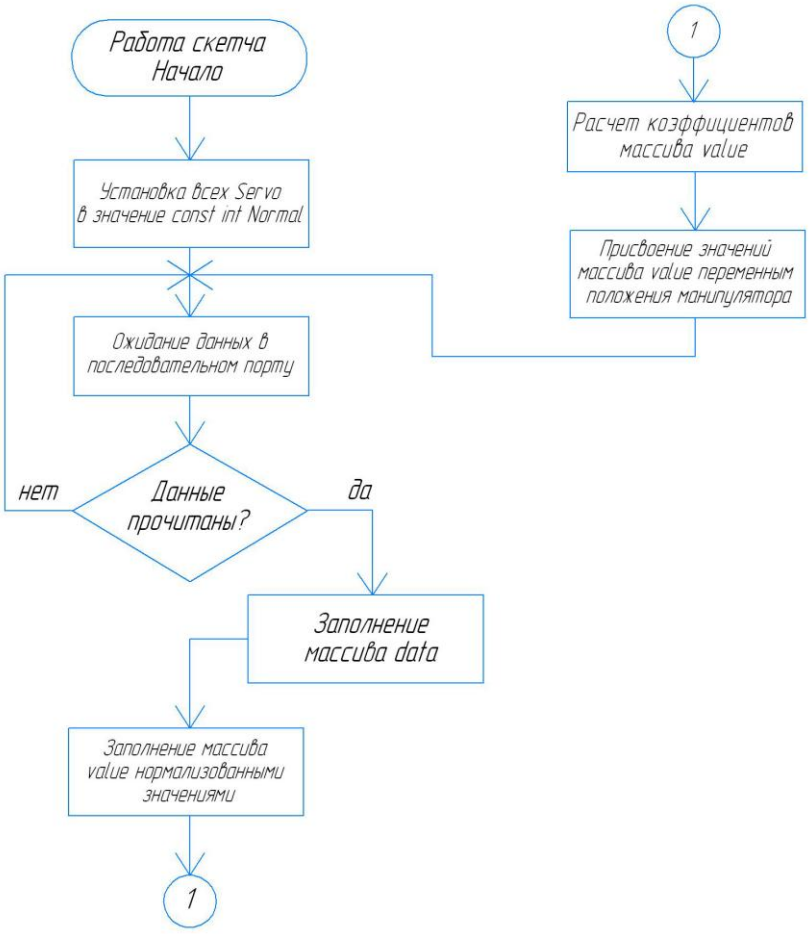

Рисунок 21 – алгоритм работы программы для платы управления

Рассмотрим основные моменты кода. С полным кодом программы можно ознакомиться в приложение А.

```
int data[9];
int value[4];
```

Рисунок 22 – Фрагмент кода программы платы управления

Так как при передаче и считывание с последовательного порта будет задействован большой набор данных нам следует воспользоваться массивами данных. Массив данных это своего рода участок памяти разделенный на некоторое число элементарных ячеек, каждая из которых имеет свой порядковый номер. И чтобы получить доступ к информации, которая храниться в определенной ячейке, нужно всего лишь знать ее номер. В нашем случае нам понадобятся два одномерных массива типа int. Тип данных int используется для хранения числовых данных и занимает 2 байта памяти. Каждому массиву мы присвоили название data и value, также обозначили размеры этих массивов. Массиву data обозначили размер в 9 элементов, а массиву value в 4, тем самым дав инструкцию компилятору на выделение памяти 9*2=18 байт для data и соответственно 4*2=8 байт для value, то есть на каждый элемент массива выделяется по 2 байта.

```
Servo Osnova;
Servo Naklonnizhnii;
Servo Naklonverhnii;
Servo Povorotzahvata;
Servo Zahvat;
```

Рисунок 23 – Фрагмент кода программы платы управления

Создали объекты типа Servo. Данный тип объекта имеет свои свойства, которые уже описали за нас в подключенной библиотеке Servo.h.

```
const int osnovaMin = 4;
const int osnovaMax = 175;
const int osnovaNormal = 80;

const int naklonnizhniiMin = 0;
const int naklonnizhniiMax = 180;
const int naklonnizhniiNormal = 90;

const int naklonverhniiMin = 0;
const int naklonverhniiMax = 180;
const int naklonverhniiNormal = 60;

const int povorotzahvataMin = 0;
const int povorotzahvataMax = 175;
const int povorotzahvataNormal = 90;

const int zahvatMin = 25;
const int zahvatMax = 130;
const int zahvatNormal = 25;
```

Рисунок 24 – Фрагмент кода программы платы управления

Задали константы, которые будут доступны только для чтения, то есть нельзя будет изменить значение переменных. Выбирая между define и const int, выбрали второе, так как для работы с массивами необходимо использовать const по нескольких основным причинам: компилятор выдаст отчет об ошибке, в случае использования константы другого типа, на константы действуют единые правила области видимости переменных в отличие от define. Для каждого сервопривода задано по 3 константы к каждой из который присвоено значение, отвечающее за максимальный, минимальный и начальный угол положения сервоприводов.

```
Osnova.attach(2);
Naklonnizhnii.attach(3);
Naklonverhnii.attach(4);
Povorotzahvata.attach(5);
Zahvat.attach(6);
```

Рисунок 25 – Фрагмент кода программы платы управления

Присоединяем каждый серво к необходимому входу, через который будет производиться управление.

```
if ( (value[0] <= naklonverhniiMax) & (value[0] >= naklonverhniiMin) ){
    Naklonverhnii.write(value[0]);
}

if ( (value[1] <= osnovaMax) & (value[1] >= osnovaMin) ){
    Osnova.write(value[1]);
}

if ( (value[2] <= naklonnizhniiMax) & (value[2] >= naklonnizhniiMin) ){
    Naklonnizhnii.write(value[2]);
}

if ( (value[3] <= povorotzahvataMax) & (value[3] >= povorotzahvataMin) ){
    Povorotzahvata.write(value[3]);
```

Рисунок 26 – Фрагмент кода программы платы управления

Сравниваем все значения value из последовательного порта с константами, если все условия соблюдаются, то выполняется запись на соответствующий порт соответствующего значения value.

Сравнивает значение value[3] с константами povorotzahvataMax и povorotzahvataMin, если условия соблюдаются, то выполняется запись на порт Povorotzahvata значения (value[3].

```
if ( (data[8] == 0x01) & (zahvat <= zahvatMax) ) {
    Zahvat.write(zahvat++);
    data[8] = 0;
}

if ( (data[8] == 0x10) & (zahvat >= zahvatMin) ){
    Zahvat.write(zahvat--);
    data[8] = 0;
```

Рисунок 27 – Фрагмент кода программы платы управления

Сравнивает значение data[8] с 0x01, data[8] с 0x10 (обращением к регистру микропроцессора arduino которое хранит целое число) и сравнивает значения zahvat с zahvatMax и zahvatMin, если все условия соблюдены то на порт Zahvat пишутся значения из переменных, прибавляя единицу в первом случае и вычитая единицу во втором.

data[8] = 0 обнуляет значение 9 члена массива данных.

```
void readSerial(){
    if (Serial.available() == 11)
            if (Serial.read() == 0xFF)
                if (Serial.read() == 0xAA){
                    data[0] = Serial.read();
                    data[1] = Serial.read();
                    data[2] = Serial.read();
                    data[3] = Serial.read();
                    data[4] = Serial.read();
                    data[5] = Serial.read();
                    data[6] = Serial.read();
                    data[7] = Serial.read();
                    data[8] = Serial.read();

                    data[9] = Serial.read();
```

Рисунок 28 – Фрагмент кода программы платы управления

ReadSerial – функция которая отвечает за заполнения массива данных в шестнадцатеричной системе. Включаем процесс, который ждет появление данных в

Serial (в последовательном порте от терминальной программы, отправленный пакет содержит 11 байт). Если данные прочитаны, то выполняется следующий фрагмент кода. Первый байт всегда 0xFF, второй байт всегда 0xAA. Далее заполняются по очереди все 10 членов массива из последовательного порта.

```
void dataToNormal(){

    // naklonnizhnii
    value[0] = naklonnizhniiNormal - data[3];

    //osnova
    value[1] = data[0] - data[1] + osnovaNormal;

    //naklonverhnii
    value[2] = data[4] - data[5] + naklonverhniiNormal;

    //povorotzahvata
    value[3] = data[6] - data[7] + povorotzahvataNormal;
```

Рисунок 29 – Фрагмент кода программы платы управления

Нормализуются значения data: 1) из константы naklonnizhniiNormal вычитаем значения четвертого члена массива data, 2) к разности первого и второго члена массива data прибавляем константу osnovaNormal, 3) к разности пятого и шестого члена массива data прибавляем константу naklonverhniiNormal, 4) к разности седьмого и восьмого члена массива data прибавляем константу povorotzahvataNormal. В двух словах код можно описать так: считываются значения с ком порта, нормализуются и сравниваются с константами.

Терминальная программа.

Исходный код терминальной программы представляет собой решение, которое состоит из нескольких частей. Рассмотрим ту часть, которая считывает положения мышки, состояния клавиш клавиатуры и после отправляет все эти значения на последовательный порт.

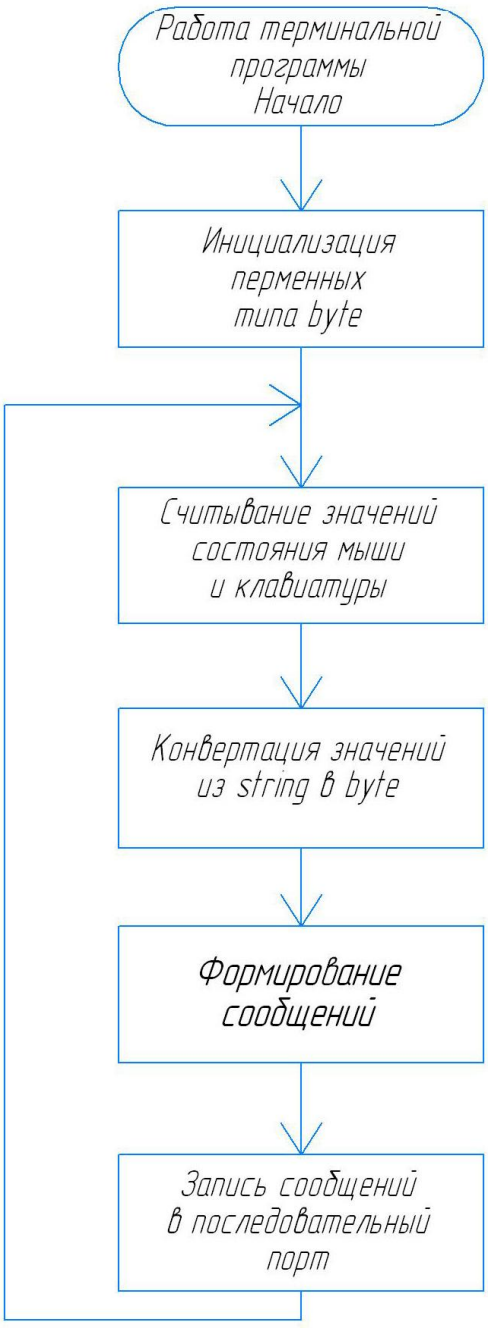

Рисунок 30 – Алгоритм работы терминальной программы

С полным кодом программы можно ознакомиться в приложение В.

```
int xCount = 0;
int yCount = 0;
int zCount = 0;
int rCount = 0;
int xCount_lst = 0;
int yCount_lst = 0;
int zCount_lst = 0;
int rCount_lst = 0;
int speed = 44;
int center = 100;
byte buttonData = 0x00;
byte keyData = 0x00;
byte buttonData_lst = 0x00;
bool release_flag = true;

int xCountMax = 110;
int xCountMin = -110;
int yCountMax = 90;
int yCountMin = -10;
int zCountMax = 60;
int zCountMin = -100;
int rCountMax = 110;
int rCountMin = -110;
```

Рисунок 31 – Фрагмент кода терминальной программы

Инициализировали переменные типа int и назначили им десятичный литерал. Инициализировали переменные типа byte и назначили им шестнадцатеричный литерал. Bool проверяет соблюдение всех логических условий. Также обозначили максимальные и минимальные значения каждой переменной типа int.

```
private void Form2_MouseMove(object sender, MouseEventArgs e)
{
    int min = center - (45-speed);
    int max = center + (45-speed);
    int x = Cursor.Position.X - this.Left - 3;
    int y = Cursor.Position.Y - this.Top - 23;
    if (x < min || x > max || y < min || y > max)
    {
        Cursor.Position = new System.Drawing.Point(this.Left + center + 3, this.Top + center + 23);
    }

    if (x < min) xCount--;
    if (x > max) xCount++;
    if (y < min) yCount++;
    if (y > max) yCount--;
```

Рисунок 32 – Фрагмент кода терминальной программы

Обработчик события движения курсора мыши. Возвращает курсор обратно в центр формы.

```
case Keys.W:
case Keys.Up:
                    keyData |= 0x01;
                    yCount += 1;
                    break;
case Keys.S:
case Keys.Down:
                    keyData |= 0x02;
                    yCount -= 1;
                    break;
case Keys.D:
case Keys.Right:
                    keyData |= 0x08;
                    xCount += 1;
                    break;
case Keys.A:
case Keys.Left:
                    keyData |= 0x04;
                    xCount -= 1;
                    break;
```

Рисунок 33 – Фрагмент кода терминальной программы

Обработчик событий нажатия на клавиши клавиатуры.

```
if (e.KeyData == Keys.W || e.KeyData == Keys.Up)
{
    keyData &= 0xfe;
}
if (e.KeyData == Keys.S || e.KeyData == Keys.Down)
{
    keyData &= 0xfd;
}
if (e.KeyData == Keys.D || e.KeyData == Keys.Right)
{
    keyData &= 0xf7;
}
if (e.KeyData == Keys.A || e.KeyData == Keys.Left)
{
    keyData &= 0xfb;
```

Рисунок 34 – Фрагмент кода терминальной программы

Обработчик событий отжатия клавиш клавиатуры.

В исходном коде терминальной программы, представленного в приложение В, присутствует еще довольно много обработчиков событий, помимо приведенных выше: обработчики события нажатия и отжатия на левую и правую кнопки мыши, нажатия на клавишу Esc, разворачивания списка доступных портов, нажатия на кнопку "Выход", нажатия на кнопку "?" и т. д.

```
private void SendAsHex(string str)
{
    int Len = str.Length;
    byte[] send = new byte[Len / 2];
    int j = 0;
    for (int i = 0; i < Len; i = i + 2, j++)
        send[j] = Convert.ToByte(str.Substring(i, 2), 16);
    serialPort1.Write(send, 0, send.Length);
```

Рисунок 35 – Фрагмент кода терминальной программы

В этом методе происходит конвертация сообщения из типа String в Byte и последующая запись полученного сообщения в последовательный порт.

```
private void sendData()
{
    this.parent.sendData("ff");
    this.parent.sendData("aa");
    this.parent.sendData(((xCount >> 8) & 0x00FF).ToString("x"));
    this.parent.sendData((xCount & 0x00FF).ToString("x"));
    this.parent.sendData(((yCount >> 8) & 0x00FF).ToString("x"));
    this.parent.sendData((yCount & 0x00FF).ToString("x"));
    this.parent.sendData(((zCount >> 8) & 0x00FF).ToString("x"));
    this.parent.sendData((zCount & 0x00FF).ToString("x"));
    this.parent.sendData(((rCount >> 8) & 0x00FF).ToString("x"));
    this.parent.sendData((rCount & 0x00FF).ToString("x"));
    this.parent.sendData(buttonData.ToString("x"));
```

Рисунок 36 – Фрагмент кода терминальной программы

В этом методе происходит формирование сообщений для записи в последовательный порт.

```
private void sComm_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        rxBuf = (byte)serialPort1.ReadByte();
        if (stateMachine == 0)
        {
            if (rxBuf == 0xff) stateMachine = 1;
            else stateMachine = 0;
        }
        else if (stateMachine == 1)
        {
            if (rxBuf == 0xaa) stateMachine = 2;
            else stateMachine = 0;
        }
        else if (stateMachine == 2)
        {
            dataBuf[t++] = rxBuf;
            if (t > 2)
            {
                rxAdd = dataBuf[0];
                rxData = (dataBuf[2] + (dataBuf[1] << 8)) - 32768;
                controlPanel.set_data();
                stateMachine = 0;
                t = 0;
```

Рисунок 37 – Фрагмент кода терминальной программы

В этом методе происходит считывание данных из последовательного порта.

```
public void sendData(string input)
{
    try
    {
        if (!serialPort1.IsOpen)
            btOpen_Click(null, null);

        string TempStr = string.Empty;
        TempStr = input;
        TempStr = DelSpace(TempStr);
        SendAsHex(TempStr);
    }
    catch (Exception err)
```

Рисунок 38 – Фрагмент кода терминальной программы

В этом методе происходит подготовка сообщений для записи их в последовательный порт. Вызов метода SendAsHex для записи подготовленных сообщений в последовательный порт.

Собранное решение представляет собой программу, состоящую из двух окон.

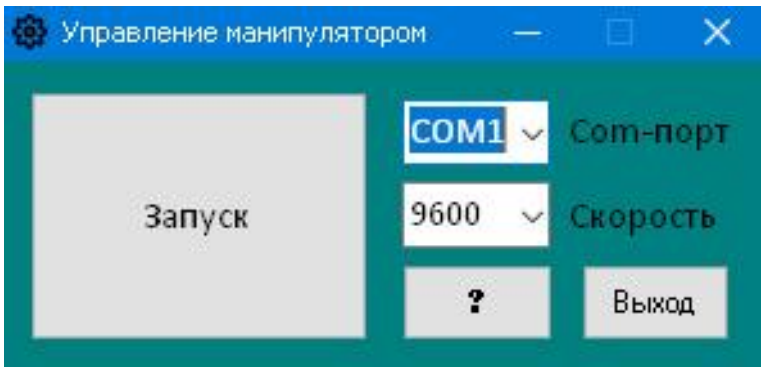

Рисунок 39 – Первое окно терминальной программы

Первое окно отвечает за выбор активного последовательного порта, за выбор скорости передачи данных, вызов справки. При нажатии на кнопку «Запуск» вызывается второе окно.

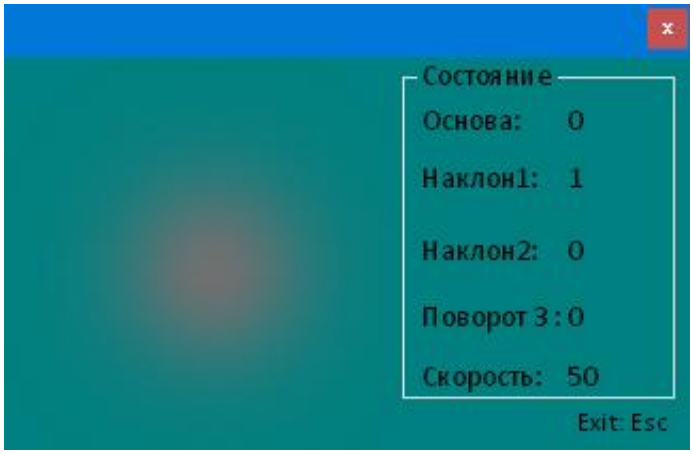

Рисунок 40 – Второе окно терминальной программы

Второе окно отвечает непосредственно за управление манипулятором, также можно увеличить скорость движения манипулятора. В графе состояния выводятся текущие положения каждого сервопривода

4. Экспериментальная часть

4.1. Инструкция по работе с устройством

1. Произвести осмотр стенда перед началом использования на наличие каких-либо повреждений.
2. Вставить шнур в блок питания.
3. Подключить плату ардуино к компьютеру.
4. Загрузить программу на плату управления.
5. Запустить файл программы Управление манипулятором.exe.
6. Выбрать задействованный последовательный порт в первом окне программы.
7. Для получения справки нажмите на вопросительный знак в первом окне программы.
8. Нажать кнопку включения/выключения на блоке питания.
9. Нажать на кнопку «Запуск» в первом окне программы для управления манипулятором.

   В появившемся окне осуществляется непосредственно управление манипулятором:

   1) с помощью движений мыши по оси x изменяется положение Servo Naklonverhnii, по оси y изменяется положение Servo Osnova.

   2) при нажатии на клавиатуре клавиш Q и E изменяется положение Servo Naklonnizhnii

   3) клавиши Z и X отвечают за изменение положения Servo Povorotzahvata

   4) левая и правая кнопки мыши отвечают за положение Servo Zahvat

   5) для увеличения скорости сервоприводов используются кнопки + и –
10. Чтобы закрыть окно управления манипулятором нажмите клавишу Esc на клавиатуре.
11. Чтобы закрыть стартовое окно нажмите на кнопку «Выход».

12. Для завершения работы с устройством, так же нужно нажать кнопку включения/выключения на блоке питания.
13. После завершения работы произвести осмотр стенда на наличие каких-либо повреждений. При обнаружение каких-либо повреждение дальнейшая эксплуатация устройства запрещена до момента устранения неполадок.

4.2. Проверка и отладка программной части устройства

Произвели проверку и окончательную отладку программной части. При управление манипулятором были замечены некоторые небольшие резкие рывки сервоприводов что приводит к колебанию всей конструкции манипулятора. Это происходит из-за того, что сервоприводы работают не с достаточно высокой точностью и на малые углы поворота могут не отреагировать. В скетче выставили наиболее стабильные значения начального, минимального и максимального положения каждого сервопривода, для устранения дребезга сервомоторов, добились более плавной работы манипулятора. В коде терминальной программы так же были опытным путем найдены и выставлены оптимальные значения переменных каждой оси управления и скорости управления каждым из сервоприводов, что позволило наиболее точно и без каких-либо трудностей производить управление манипулятором. Для проверки стабильности терминальной программы, во время работы с манипулятором внезапно отключали активный COM порт Arduino от компьютера, манипулятор остался в том же положении и никаких действий самостоятельно не производил.

5. Безопасность и экологичность устройства

При сборке и эксплуатации исследуемого устройства применялось различное оборудование, во время использования которого необходимо соблюдать определенные правила безопасности для предотвращения таких чрезвычайных ситуаций как короткое замыкание а в следствие и пожар: полную проверку состояния инструментов перед использованием, в частности электрических на наличие дефектов в кабелях питания, при обнаружении кабеля с оголенной частью, запрещена дальнейшая эксплуатация устройства до момента устранения дефекта.

При использование электрического оборудования, предназначенного для сверления технологических отверстий и резки материала, используемого при сборке устройства, необходимо тщательно следить за состоянием расходных материалов (сверла, диски и т.п.), использовать защитное снаряжение для предотвращения травм (защитные очки или маску, перчатки и одежду из плотной ткани). При обнаружении неисправности во время использования инструмента, необходимо немедленно отключить его от сети. Во время эксплуатации паяльного оборудования так же нужно соблюдать технику безопасности: не браться за жало паяльника, не оставлять включенным без присмотра, держать в удаленности от легковоспламеняющихся материалов, рекомендуется пользоваться респиратором для защиты дыхательных путей от вредных испарений. Респиратор необходимо менять по истечению срока допустимой эксплуатации, указанного в регламенте или при существенном снижение рабочих характеристик респиратора. При соприкосновении флюса с открытым участком кожи необходимо тщательно промыть пораженное место водой. Если при работе с паяльником был получен ожог, нужно незамедлительно обратиться за медицинской помощью, желательно к врачу. Также для предотвращения чрезвычайных ситуаций необходимо проводить инструктаж о пожарной безопасности перед использование устройства.

Помещение где проводится сборка и эксплуатация устройства должно удовлетворять всем требованиям к микроклимату СанПиН: температура,

вентиляция, уровень влажности должны поддерживаться в допустимых пределах. Особенно необходимо иметь хорошее освещение, удовлетворяющее требованиям СНиП, для предотвращения утомляемости и вследствие травматизма. Уровень шума и вибраций от спроектированного устройства удовлетворяет всем нормам и никак не влияет на человека. Для управления манипулятором в нашем проекте используется персональный компьютер, следовательно, имеется такой вредный фактор как электромагнитное излучение, которое отрицательно влияет на внимание, память и зрение.

Конструкция рабочего места должна поддерживать удобное положение при работе со стендом, чтобы избежать утомления мышц и развития заболеваний опорно-двигательного аппарата. Монитор должен располагаться на рабочем месте перед глазами пользователя в пределах от 500 мм до 700 мм. Клавиатуру и мышь необходимо установить на рабочей поверхности на расстояние примерно 200 мм от края поверхности.

6. Экономическая часть

Смета расходов на компоненты, которые были использованы при сборке данного проекта представлена ниже.

№	Наименование	Кол.	Стоимость, р.
1	Сервопривод Tower Pro MG996R	5	2250
3	ATmega2560-16AU CH340G	1	500
4	Детали каркаса манипулятора	10	1350
5	Захват для MG996R	1	300
6	Макетная плата односторонняя	1	100
7	Разъём PLS-40 штекер прямой	4	40
8	Кабель DuPont	100	250
9	Разъем K-128-2P	1	10
10	Маркер кабельный REXANT 12-6061	7	14
11	Компьютерный блок питания	1	400
12	Выключатель SC768 2P 6A	1	25
13	Набор термоусадки	1	50
			Итого: 5289 р.

Рисунок 41- Таблица расходов на компоненты

Заключение

Во время реализации выпускной квалификационной работы было осуществлено изучение всех существующих на сегодняшний день решений, связанных с темой нашего проекта. Проанализировав все преимущества и недостатки каждого решения, были выбраны наиболее перспективные для нас варианты. Исходя из выбранных способов реализации, был произведен тщательный подбор наиболее оптимальных компонентов, необходимых для нашего проекта. Далее была разработана схема конструкции стенда и описан весь процесс сборки. Показан принцип работы всех основных компонентов, задействованных в исследуемом устройстве. Создана терминальная программа управления манипулятором и программа для платы с полным управлением через компьютер, описан весь основной код данных программ. Разработанный код был испытан на построенной модели манипулятора и впоследствии доработан. Составлена подробная инструкции по эксплуатации разработанного стенда. Так же осуществлена оценка безопасности и экологичности устройства. Была подготовлена смета на все задействованные при сборке устройства компоненты. Итогом выпускной квалификационной работы являются неплохие результаты управления манипулятором. Главным преимуществом является возможность применения разработанной программы для других устройств с разными вариациями конструкции и используемых компонентов. Однако заметны небольшие рывки при движении конструкции манипулятора, это происходит из-за особенностей выбранных сервоприводов. Разработанный манипулятор может эффективно использоваться в качестве демонстрационного стенда для изучения задач кинематики, динамики роботов и их программирования.

Список используемых источников

1. Основы разработки и программирования робототехнических систем: учебное пособие / Сорокин С.В., Солдатенко И.С. – Тверь: Твер. гос. ун-т, 2017. – 157 с.

2. Глибин Е.С. Программирование электронных устройств: электронное учеб. пособие / Глибин Е.С., Прядилов А.В. – Тольятти: Изд-во ТГУ, 2014.: 1 оптический диск

3. Интегральные роботы: сборник статей / Поздняк Г. Е. – М.: Мир, 1973. – 213 с.

4. Промышленные роботы. Кинематика, динамика, контроль и управление. Серия «Библиотека инженера» / Булгаков А. Г., Воробьев В. А. – М.: СОЛОН-ПРЕСС, 2008. – 488 с.: ил.

5. Моделирование, планирование траекторий и управление движение робота-манипулятора / Пол Р. – пер. с англ. М.: Наука, 1976. – 104 с.

6. Программирование микронотроллерных плат Arduino / Соммер У. – Санкт-Петербург: БХВ-Петербург, 2012. – 256с.

7. C Sharp 6.0 Справочник. Полное описание языка, 6-е изд. / Албахари Д., Албахари Б. – пер. с англ. – М.: ООО «И. Д. Вильямс», 2016 – 1040с.

8. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. - М.: Издательский дом Додэка-XXI, 2007. - 592 с.

9. Шпак Ю. А. Программирование на языке С для AVR и PIC микроконтроллеров. – Москва: Додэка-XXI, МК-Пресс, 2007.

10. Уроки робототехники. Конструкция, движение и управление. / Филиппов С. А. – М.: Лаборатория знаний, 2017. – 176 с.

11. Интернет магазин электронных компонентов «Импульс» [Электронный ресурс]. Режим доступа: https://www.impulsi.ru (04.06.2018).

12. Справочник языка программирования Arduino [Электронный ресурс]. Режим доступа: http://arduino.ru/Reference

13. Справочник по C# [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference

14. HC-SR04 User Guide [Электронный ресурс]: документация. – Режим: доступа http://www.mpja.com/download/hc-sr04_ultra.modguide.pdf

15. MG996R Servo [Электронный ресурс]: документация. – Режим доступа: http://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf

16. Arduino Mega 2560 datasheet [Электронный ресурс]: документация. –Режим доступа: http://microelectronicos.com/datasheets/arduinomega2560.pdf

17. Как домашний настольный манипулятор может помочь в DIY [Электронный ресурс]: статья. Режим доступа: https://geektimes.com/post/284962

18. Robotic Arm Inverse Kinematics on Arduino [Электронный ресурс]: форум. Режим доступа: https://www.circuitsathome.com/mcu/robotic-arm-inverse-kinematics-on-arduino

19. Uarm манипуляторы [Электронный ресурс]. https://www.ufactory.cc/#/en

20. Работа с последовательным портом Arduino [Электронный ресурс]: статья. Режим доступа: http://www.customelectronics.ru/arduino-rabota-s-com-portom

21. Elias Eliot, Tihomir Latinovic, Dayal R. Parchi, J. Srinivas. Design, Analysis and Fabrication of an Articulated Mobile Manipulator // Department of Industrial Design National Institute of Technology, Rourkela Odisha, India, 2013. URL: https://www.researchgate.net/publication/275770788

22. Paula Useche Murillo, Robinson Jimenez Moreno, Oscar F Avilés S. Individual Robotic Arms Manipulator Control Employing Electromyographic Signals Acquired by Myo Armbands // International Journal of Applied Engineering Research Volume 11, Number 23, 2016. URL: https://www.researchgate.net/publication/314079352

23. Pieter Dijkshoorn. Designing a generic software architecture for ARW manipulators // University of Twente, Enschede, Netherlands, November 2016. URL: https://www.ram.ewi.utwente.nl/calendar/details/designing-a-generic-software-architecture-for-arw-manipulators.html

24. Mohd Ashiq Kamaril Yusoffa, Reza Ezuan Saminb, Babul Salam Kader Ibrahimc. Wireless Mobile Robotic Arm // Faculty of Electrical and Electronics Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, Malaysia, 2012. URL: https://www.researchgate.net/publication/271890997

25. Mohd Aliffa, Shujiro Dohtaa, Tetsuya Akagia, Hui Lia. Development of a simple-structured pneumatic robot arm and its control using low-cost embedded controller // Okayama University of Science 1-1 Ridai-cho, Kita-Ku, Okayama, Japan, 2012. URL: https://www.sciencedirect.com/science/article/pii/S1877705812025398

Приложение А

```
#include <Servo.h>

void readSerial(void);
void dataToNormal(void);

int data[9];
int value[4];

Servo Osnova;
Servo Naklonnizhnii;
Servo Naklonverhnii;
Servo Povorotzahvata;
Servo Zahvat;

const int osnovaMin = 4;
const int osnovaMax = 175;
const int osnovaNormal = 80;

const int naklonnizhniiMin = 0;
const int naklonnizhniiMax = 180;
const int naklonnizhniiNormal = 90;

const int naklonverhniiMin = 0;
const int naklonverhniiMax = 180;
const int naklonverhniiNormal = 60;

const int povorotzahvataMin = 0;
const int povorotzahvataMax = 175;
const int povorotzahvataNormal = 90;

const int zahvatMin = 25;
const int zahvatMax = 130;
const int zahvatNormal = 25;


int zahvat = zahvatNormal;

void setup(){
```

```
    Osnova.attach(2);
    Naklonnizhnii.attach(3);
    Naklonverhnii.attach(4);
    Povorotzahvata.attach(5);
    Zahvat.attach(6);

    Zahvat.write(zahvatNormal);
    Serial.begin(9600);
}

void loop(){
    readSerial();
    dataToNormal();

    if ( (value[0] <= naklonverhniiMax) & (value[0] >= naklonverhniiMin) ){
        Naklonverhnii.write(value[0]);
    }

    if ( (value[1] <= osnovaMax) & (value[1] >= osnovaMin) ){
        Osnova.write(value[1]);
    }

    if ( (value[2] <= naklonnizhniiMax) & (value[2] >= naklonnizhniiMin) ){
        Naklonnizhnii.write(value[2]);
    }

    if ( (value[3] <= povorotzahvataMax) & (value[3] >= povorotzahvataMin) ){
        Povorotzahvata.write(value[3]);
    }

    if ( (data[8] == 0x01) & (zahvat <= zahvatMax) ) {
        Zahvat.write(zahvat++);
        data[8] = 0;
    }

    if ( (data[8] == 0x10) & (zahvat >= zahvatMin) ){
        Zahvat.write(zahvat--);
        data[8] = 0;
    }

}
```

```
void readSerial(){
   if (Serial.available() == 11)
         if (Serial.read() == 0xFF)
            if (Serial.read() == 0xAA){
               data[0] = Serial.read();
               data[1] = Serial.read();
               data[2] = Serial.read();
               data[3] = Serial.read();
               data[4] = Serial.read();
               data[5] = Serial.read();
               data[6] = Serial.read();
               data[7] = Serial.read();
               data[8] = Serial.read();

               data[9] = Serial.read();
            }
}

void dataToNormal(){

   // naklonnizhnii
   value[0] = naklonnizhniiNormal - data[3];

   //osnova
   value[1] = data[0] - data[1] + osnovaNormal;

   //naklonverhnii
   value[2] = data[4] - data[5] + naklonverhniiNormal;

   //povorotzahvata
   value[3] = data[6] - data[7] + povorotzahvataNormal;
}
```

Приложение В

Form 1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace MouseControl
{
   public partial class Form1 : Form
   {
      bool IsReceving = false; // Создание переменной логического типа

      public Form1()
      {
         InitializeComponent(); // Вызов метода, отвечающего за отрисовку формы
      }

      private void SendAsHex(string str)      // В этом методе происходит конвертация сообщения из типа String в Byte и последующая запись полученного сообщения в SerialPort
      {
         int Len = str.Length;
         byte[] send = new byte[Len / 2];
         int j = 0;
         for (int i = 0; i < Len; i = i + 2, j++)
            send[j] = Convert.ToByte(str.Substring(i, 2), 16);
         serialPort1.Write(send, 0, send.Length);
      }

      private string DelSpace(string str) // В этом методе происходит очистка сообщения от пробелов. Переменная TempStr всегда имеет четное количество символов.
      {
         string TempStr = string.Empty;
         int Len = str.Length;
         for (int i = 0; i < Len; i++)
         {
            if (str[i] != ' ')
               TempStr += str[i];
         }
         Len = TempStr.Length;
         if (Len % 2 != 0)
            TempStr = '0' + TempStr;
         return TempStr;
      }
```

```
        private void ReOpenPort() // Метод для переоткрытия порта SerialPort
        {
            try
            {
                btClose_Click(null, null);
                if (!serialPort1.IsOpen)
                    btOpen_Click(null, null);
            }
            catch (Exception Err)
            {
                controlPanel.Close();
                MessageBox.Show(Err.Message, "Управление манипулятором");
            }
        }

        private void Form1_Load(object sender, EventArgs e) // В этом методе происходит получение
имен всех доступных Serial портов
        {
            try
            {
                foreach (string com in System.IO.Ports.SerialPort.GetPortNames())
                    this.cmPort.Items.Add(com);
                cmPort.SelectedIndex = 0;
            }
            catch
            {
                controlPanel.Close();
                MessageBox.Show("Не найден последовательный порт!", "Управление манипулятором");
            }
        }

        private void OpenPort() // В этом методе происходит конфигурация и открытие порта
        {
            serialPort1.WriteTimeout = 1000;  //Написать тайм-аут Установка таймаута на запись)
 serialPort1.ReadTimeout = 1000;   //Тайм-аут чтения, там же. (Установка таймаута на чтение)
            serialPort1.NewLine = "\r\n";               serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.sComm_DataReceived);
            serialPort1.PortName = cmPort.Text;
            serialPort1.BaudRate = int.Parse(cmBaudRate.Text);
            serialPort1.DataBits = 8;
            serialPort1.Parity = (Parity)Enum.Parse(typeof(Parity), "None");
            serialPort1.StopBits = (StopBits)Enum.Parse(typeof(StopBits), "1");
            serialPort1.Open();
        }

        private void ClosePort() // В этом методе происходит очистка порта с последующим его
закрытием
        {

            serialPort1.DataReceived -= this.sComm_DataReceived;              while (IsReceving)
                Application.DoEvents();     //Обработка последовательного приема системных сообщений и
других событий.
```

```
            serialPort1.Close();

        }

        public static Form2 controlPanel = null; // Создание экземпляра класса Form2

        private void btOpen_Click(object sender, EventArgs e) // Обработчик события нажатия на кнопку "Запуск"
        {
            try
            {
                OpenPort(); // Вызов метода для открытия порта

                //Изменения конфигурации первой формы и создание и активация второй формы
                if (serialPort1.IsOpen)
                {
                    btClose.Enabled = true;
                    btOpen.Enabled = false;
                    cmPort.Enabled = false;
                    cmBaudRate.Enabled = false;

                    if (controlPanel == null)
                    {
                        controlPanel = new Form2(this);
                        controlPanel.Show();
                    }
                    else
                    {
                        controlPanel.Activate();
                    }
                }
            }
            catch (Exception er)
            {
                ClosePort();
                MessageBox.Show("Не удалось открыть порт!" + er.Message, "Управление манипулятором");
            }
        }

        private void cmPort_MouseDown(object sender, MouseEventArgs e) // Обработка события разворачивания списка доступных портов
        {
            try
            {
                cmPort.Items.Clear();
                foreach (string com in System.IO.Ports.SerialPort.GetPortNames())
                    this.cmPort.Items.Add(com);
            }
            catch
            {
                controlPanel.Form2Closing();
                MessageBox.Show("Не найден последовательный порт!", "Управление манипулятором");
```

```
        }
    }

    // Блок создания переменных
    int stateMachine = 0;
    int t = 0;
    public static byte rxBuf;
    public static int rxData;
    public static byte rxAdd;
    byte[] dataBuf = new byte[3];
    //Конец блока создания переменных

    private void sComm_DataReceived(object sender, SerialDataReceivedEventArgs e) // (В этом
методе происходит считывание данных из Serial порта)
    {
        try
        {
            rxBuf = (byte)serialPort1.ReadByte();
            if (stateMachine == 0)
            {
                if (rxBuf == 0xff) stateMachine = 1;
                else stateMachine = 0;
            }
            else if (stateMachine == 1)
            {
                if (rxBuf == 0xaa) stateMachine = 2;
                else stateMachine = 0;
            }
            else if (stateMachine == 2)
            {
                dataBuf[t++] = rxBuf;
                if (t > 2)              //получение 3 байта данных
                {
                    rxAdd = dataBuf[0];
                    rxData = (dataBuf[2] + (dataBuf[1] << 8)) - 32768;                    //
                    controlPanel.set_data();
                    stateMachine = 0;
                    t = 0;
                }
            }
        }
        catch (Exception Err)
        {
            controlPanel.Form2Closing();
            MessageBox.Show(Err.Message, "Управление манипулятором");
        }
    }

    public void sendData(string input)  // В этом методе происходит подготовка сообщений для
записи их в Serial порт.
    {
        try
        {
```

```
            if (!serialPort1.IsOpen)
                btOpen_Click(null, null);

            string TempStr = string.Empty;
            TempStr = input;
            TempStr = DelSpace(TempStr);
            SendAsHex(TempStr);  // Вызов метода SendAsHex для записи подготовленных сообщений в Serial порт
        }
        catch (Exception err)
        {
            controlPanel.Form2Closing();
            MessageBox.Show(err.Message, "Управление манипулятором");
        }
    }

    public void btClose_Click(object sender, EventArgs e) // Обработчик события нажатия на кнопку "Close Port"
    {
        ClosePort();
        if (!serialPort1.IsOpen)
        {
            btOpen.Enabled = true;
            btClose.Enabled = false;
            cmPort.Enabled = true;
            cmBaudRate.Enabled = true;
        }
    }

    private void exit_Click(object sender, EventArgs e) // Обработчик события нажатия на кнопку "Выход"
    {
        btClose_Click(null, null);
        this.Close();
    }

    private void logo_Click(object sender, EventArgs e) // Обработчик события нажатия на иконку программы на форме
    {
        System.Diagnostics.Process.Start("");
    }

    private void button1_Click(object sender, EventArgs e) // Обработчик события нажатия на кнопку "?"
    {
        MessageBox.Show(
            "Инструкция по управлению : " + Environment.NewLine +
            " " + Environment.NewLine +
            "1. " + "Наклон верхний: W & S" + Environment.NewLine +
            " " + Environment.NewLine +
            "2. " + "Наклон нижний: E & Q" + Environment.NewLine +
            " " + Environment.NewLine +
            "3. " + "Основа: D & A" + Environment.NewLine +
```

```
            " " + Environment.NewLine +
            "4. " + "Поворот захвата: Z & X" + Environment.NewLine +
            " " + Environment.NewLine +
            "5. " + "Захват: ЛКМ & ПКМ" + Environment.NewLine +
            "");
        }

        private void comPort_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Form 2

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

namespace MouseControl
{
    public partial class Form2 : Form
    {
        int xCount = 0; // инициализировали переменную типа int и назначили ей десятичный литерал.
        int yCount = 0;
        int zCount = 0;
        int rCount = 0;
        int xCount_lst = 0;
        int yCount_lst = 0;
        int zCount_lst = 0;
        int rCount_lst = 0;
        int speed = 44;
        int center = 100;
        byte buttonData = 0x00;  // инициализировали переменную типа byte и назначили ей
шестнадцатеричный литерал.              // H->L  4:Xbutton2(+) 3:Xbutton1(-) 2:Right 1:Middle 0:Left
        byte keyData = 0x00;                // H->L  5:- 4:+ 3:D 2:A 1:S 0:W
        byte buttonData_lst = 0x00;
        bool release_flag = true;           // проверка логических условий

        int xCountMax = 110;                // максимальное значение переменной
        int xCountMin = -110;               // минимальное значение переменной
        int yCountMax = 90;
        int yCountMin = -10;
        int zCountMax = 60;
        int zCountMin = -100;
        int rCountMax = 110;
        int rCountMin = -110;
```

```
        public Form1 parent;
        public Form2(Form1 parent) // Конструктор для инициализации компонентов формы и создания обработчиков событий
        {
            InitializeComponent();
            this.parent = parent;
            this.MouseMove += new System.Windows.Forms.MouseEventHandler(Form2_MouseMove);
            this.MouseDown += new System.Windows.Forms.MouseEventHandler(Form2_MouseDown);
            this.MouseUp += new System.Windows.Forms.MouseEventHandler(Form2_MouseUp);
            this.KeyDown += new System.Windows.Forms.KeyEventHandler(Form2_KeyDown);
            this.KeyUp += new System.Windows.Forms.KeyEventHandler(Form2_KeyUp);
        }

        private void Form2_MouseMove(object sender, MouseEventArgs e)  // Обработчик события движения курсора. Возвращает курсор обратно в центр формы
        {
            int min = center - (45-speed);
            int max = center + (45-speed);
            int x = Cursor.Position.X - this.Left - 3;
            int y = Cursor.Position.Y - this.Top - 23;
            if (x < min || x > max || y < min || y > max)
            {
                Cursor.Position = new System.Drawing.Point(this.Left + center + 3, this.Top + center + 23);
            }

            if (x < min) xCount--;
            if (x > max) xCount++;
            if (y < min) yCount++;
            if (y > max) yCount--;

        }

        private void Form2_MouseDown(object sender, MouseEventArgs e)  // Обработчик события нажатия на кнопки мыши
        {

            if (e.Button == MouseButtons.Left)
            {
                buttonData &= 0xfd;

                buttonData |= 0x01;
            }
            if (e.Button == MouseButtons.Left && e.Clicks == 2)
            {
                buttonData &= 0xfe;
                buttonData |= 0x02;
            }
            if (e.Button == MouseButtons.Middle)
            {
                buttonData |= 0x04;
            }
            if (e.Button == MouseButtons.Middle && e.Clicks == 2)
            {
```

```
            buttonData |= 0x08;
        }
        if(e.Button == MouseButtons.Right)
        {
            buttonData |= 0x10;
            r_label.Text = "*";
        }
        if (e.Button == MouseButtons.Right && e.Clicks == 2)
        {
            buttonData |= 0x20;
        }
        if (e.Button == MouseButtons.XButton1)
        {
            speed = --speed < 1 ? 1 : speed;
            label11.Text = Convert.ToString(speed);
        }
        if (e.Button == MouseButtons.XButton2)
        {
            speed = ++speed > 50 ? 50 : speed;
            label11.Text = Convert.ToString(speed);
        }
    }

    private void Form2_MouseUp(object sender, MouseEventArgs e)   // Обработчик события отжатия на кнопки мыши
    {
        if (e.Button == MouseButtons.Left)
        {
            buttonData &= 0xfc;
        }
        if (e.Button == MouseButtons.Middle)
        {
            buttonData &= 0xf3;

        }
        if (e.Button == MouseButtons.Right)
        {
            buttonData &= 0xcf;
            r_label.Text = " ";

        }
    }
    private void Form2_Load(object sender, EventArgs e)   // Установка курсора в нужную точку на форме
    {
        Cursor.Clip = new Rectangle(this.Location.X + 33, this.Location.Y + 53, 130, 130);
    }

    private void exit_bt_Click(object sender, EventArgs e)      // (Обработчик события нажатия на кнопку Esc)
    {
        Form1.controlPanel = null;
        Cursor.Clip = Rectangle.Empty;
```

```
            this.parent.btClose_Click(null, null);
            this.Close();
        }

        public void set_data()
        {
            switch (Form1.rxAdd)
            {
                case 0x01: xCountMax = Form1.rxData; break;
                case 0x02: xCountMin = Form1.rxData; break;
                case 0x03: yCountMax = Form1.rxData; break;
                case 0x04: yCountMin = Form1.rxData; break;
                case 0x11: xCount = Form1.rxData; break;
                case 0x12: yCount = Form1.rxData; break;
                default: break;
            }
        }

        private void sendData()      // В этом методе происходит формирование сообщений для записи в
Serial порт
        {
            this.parent.sendData("ff");
            this.parent.sendData("aa");
            this.parent.sendData(((xCount >> 8) & 0x00FF).ToString("x"));
            this.parent.sendData((xCount & 0x00FF).ToString("x"));
            this.parent.sendData(((yCount >> 8) & 0x00FF).ToString("x"));
            this.parent.sendData((yCount & 0x00FF).ToString("x"));
            this.parent.sendData(((zCount >> 8) & 0x00FF).ToString("x"));
            this.parent.sendData((zCount & 0x00FF).ToString("x"));
            this.parent.sendData(((rCount >> 8) & 0x00FF).ToString("x"));
            this.parent.sendData((rCount & 0x00FF).ToString("x"));
            this.parent.sendData(buttonData.ToString("x"));
        }

        private void timer1_Tick(object sender, EventArgs e)      // Обработчик события такта таймера
        {
            try
            {
                if (xCount_lst != xCount || yCount_lst != yCount || zCount_lst != zCount || rCount_lst != rCount
|| ( buttonData & 0xff) != 0x00)
                {
                    xCount = xCount < xCountMin ? xCountMin : xCount;
                    xCount = xCount > xCountMax ? xCountMax : xCount;
                    yCount = yCount < yCountMin ? yCountMin : yCount;
                    yCount = yCount > yCountMax ? yCountMax : yCount;
                    zCount = zCount < zCountMin ? zCountMin : zCount;
                    zCount = zCount > zCountMax ? zCountMax : zCount;
                    rCount = rCount < rCountMin ? rCountMin : rCount;
                    rCount = rCount > rCountMax ? rCountMax : rCount;

                    label4.Text = Convert.ToString(xCount);
                    label5.Text = Convert.ToString(yCount);
                    label1.Text = Convert.ToString(zCount);
```

```
            label20.Text = Convert.ToString(rCount);
            sendData();

            xCount_lst = xCount;
            yCount_lst = yCount;
            zCount_lst = zCount;
            rCount_lst = rCount;
            buttonData_lst = buttonData;
        }
      }
      catch (Exception Err)
      {
        MessageBox.Show("Timing errors!" + Err.Message, "MouseControl");
        timer1.Enabled = false;
        Form2Closing();
      }
    }

    private void Form2_KeyDown(object sender, KeyEventArgs e)  // Обработчик событий нажатия клавиш клавиатуры
    {
      switch (e.KeyData)
      {
        case Keys.Oemplus:
                    speed = ++speed > 50 ? 50 : speed;
                    label11.Text = Convert.ToString(speed);
                    break;                                          // Увеличение уменьшение скорости
        case Keys.OemMinus:
                    speed = --speed < 1 ? 1 : speed;
                    label11.Text = Convert.ToString(speed);
                    break;
        case Keys.W:                                                 // вверх Naklonverhnii
                    keyData |= 0x01;
                    yCount += 1;
                    break;
        case Keys.S:                                                 // вниз Naklonverhnii
                    keyData |= 0x02;
                    yCount -= 1;
                    break;
        case Keys.D:                                                 // вправо Osnova
                    keyData |= 0x08;
                    xCount += 1;
                    break;
        case Keys.A:                                                 // влево Osnova
                    keyData |= 0x04;
                    xCount -= 1;
                    break;
        case Keys.Q:                                                 // вниз Naklonnizhnii
                    buttonData &= 0x7f;
                    buttonData |= 0x40;
                    zCount -= 1;
                    break;
        case Keys.E:                                                 // вверх Naklonnizhnii
```

```
                buttonData &= 0xbf;
                buttonData |= 0x80;
                zCount += 1;
                break;

            case Keys.Z:                                    // влево Povorotzahvata
                buttonData &= 0x7f;
                buttonData |= 0x50;
                rCount -= 1;
                break;
            case Keys.X:                                    // вправо Povorotzahvata
                buttonData &= 0xbf;
                buttonData |= 0x90;
                rCount += 1;
                break;

        }
    }

    private void Form2_KeyUp(object sender, KeyEventArgs e)                                    //
Обработчик событий отжатия клавиш клавиатуры
    {


        if (e.KeyData == Keys.Z || e.KeyData == Keys.X)
        {
            buttonData &= 0x3f;
         //   label1.Text = "None";
        }
        if (e.KeyData == Keys.Q || e.KeyData == Keys.E)
        {
            buttonData &= 0x2f;
        //    label1.Text = "None";
        //    r_label.Text = "F";
        }

        if (e.KeyData == Keys.W)
        {
            keyData &= 0xfe;
        }
        if (e.KeyData == Keys.S)
        {
            keyData &= 0xfd;
        }
        if (e.KeyData == Keys.D)
        {
            keyData &= 0xf7;
        }
        if (e.KeyData == Keys.A)
        {
            keyData &= 0xfb;
        }
    }
```

```
        private void Form2_FormClosing(object sender, FormClosingEventArgs
e)                                  // Приватный метод для закрытия формы
        {
            timer1.Enabled = false;
            Form1.controlPanel = null;
            Cursor.Clip = Rectangle.Empty;
            this.parent.btClose_Click(null, null);
        }

        public void Form2Closing()                          // Публичный метод для закрытия формы
        {
            timer1.Enabled = false;
            Form1.controlPanel = null;
            Cursor.Clip = Rectangle.Empty;
            this.parent.btClose_Click(null, null);
            this.Close();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label20_Click(object sender, EventArgs e)
        {

        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }
    }
}
```

Поз. обозначение	Наименование	Кол.	Примечание
A1	Arduino Mega 2560	1	
M1–M5	Сервопривод MG996R	5	
	Соединители		
X1–X5	PLS-3	5	
XS1	USB Female Type B Connector	1	
XS2	K-128-2P	1	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Журавский Д.Ю		
Пров.		Прядилов А.В.		
Н.контр.				
Утв.		Шевцов А.А.		

18.110304.107.05.03 ПЭ

Управляющее устройство

Лит.	Лист	Листов
		1

ТГУ ЭЛб-1401

Копировал

Формат А4