Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа теоретической механики

Работа допущена к защите

Директор ВШТМ, д.ф.-м.н., чл.-корр. РАН

_____ А.М. Кривцов

«___»_____2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

РАЗРАБОТКА МАНИПУЛЯТОРА С МЯГКИМ СХВАТОМ

по направлению 01.04.03 Механика и математическое моделирование
по образовательной программе
01.04.03_03 Механика и цифровое производство

Выполнил
студент гр. 3640103/80301 Р.А. Султан

Руководитель
доцент ВШТМ, к.ф.-м.н. М.Б. Бабенков

Санкт-Петербург

2020

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
Институт прикладной математики и механики
Высшая школа теоретической механики

УТВЕРЖДАЮ

Директор ВШТМ, д.ф.-м.н., чл.-корр. РАН

А.М. Кривцов

« » 2020 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы

студенту Султану Руслану Азизовичу, 3640103/80301
фамилия, имя, отчество (при наличии), номер группы

1. Тема работы: Разработка манипулятора с мягким схватом

2. Срок сдачи студентом законченной работы: 08.06.2020 г.

3. Исходные данные по работе: радиус рабочей зоны манипулятора ~400 мм.; грузоподъемность: 200 г., максимальный диаметр схватываемого объекта: 50 мм.

4. Содержание работы (перечень подлежащих разработке вопросов): Разработка 3D-модели манипулятора, разработка структурной схемы, разработка управляющей программы, разработка программы для разпознавания координат объекта по фотографии, разработка мягкого схвата, создание прототипа манипулятора с мягким схватом

5. Перечень графического материала (с указанием обязательных чертежей): кинематическая схема манипулятора, структурная схема манипулятора

6. Консультанты по работе: отсутствуют

7. Дата выдачи задания: 22.01.2020 г.

Руководитель ВКР _____ М.Б. Бабенков
(подпись) инициалы, фамилия

Задание принял к исполнению 22.01.2020 г.
(дата)

Студент _____ Р.А. Султан
(подпись) инициалы, фамилия

РЕФЕРАТ

На 46 с., 31 рисунок, 2 таблицы

РОБОТОТЕХНИКА, КОНСТРУИРОВАНИЕ, МАНИПУЛЯТОР, МЯГКИЙ СХВАТ, 3D-МОДЕЛИРОВАНИЕ, 3D-ПЕЧАТЬ, ПИТОН 3, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, АРДУИНО.

Тема выпускной квалификационной работы: «Разработка манипулятора с мягким схватом».

В данной работе изложен процесс разработки манипулятора с мягким схватом. Проведен аналитический обзор манипуляторов с мягкими схватами. Решено использовать классический жесткий схват с дополнительными съемными деформируемыми пластинами. Разработана 3D-модель манипулятора, произведена анимация его движения, смоделирована деформация мягкого схвата. Создан действующий прототип манипулятора с мягким схватом. Решена обратная задача кинематика для манипулятора с 4 степенями подвижности. Разработаны управляющая программа манипулятора, программа для распознавания координат объекта по фотографии с использованием компьютерного зрения. Произведены тестовые запуски манипулятора, показывающие работоспособность конструкции как самого манипулятора, так и мягкого схвата.

THE ABSTRACT

46 pages, 31 pictures, 2 tables

ROBOTICS, DESIGN, MANIPULATOR, SOFT GRIPPER, 3D MODELING, 3D PRINTING, PYTHON 3, COMPUTER VISION, ARDUINO

Theme of the final qualification work: "Development of a manipulator with a soft gripper".

In this paper, the development process of manipulator with soft gripper is described. An analytical review of manipulators with soft grippers was carried out. It’s decided to use a classic rigid gripper with additional removable deformable

plates. A 3D model of the manipulator was developed, its movement was animated, and deformation of the soft gripper was modeled. A working prototype of a soft gripper manipulator has been created. The inverse kinematics problem for a manipulator with 4 degrees of mobility is solved. A manipulator control program, a program for recognizing the coordinates of an object from a photograph using computer vision, has been developed. Test runs of the manipulator were made, showing the operability of the design of both the manipulator itself and the soft gripper.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Манипуляторы с «мягкими» схватами могут использоваться для манипулирования твердыми, хрупкими и мягкими объектами произвольной формы. Большинство «мягких» схватов работают на основе пневматики. Такие схваты содержат расширяющиеся или изгибающиеся захватные элементы, выполненные в виде камер из эластичного материала, например резины, изменение размера (объема) которых, при подачи в их внутреннюю полость сжатого воздуха, обеспечивается зажим детали.

Также при создании универсальных захватных устройств могут использоваться многозвенные механизмы, способные совершать волнообразные движения. В таких механизмах звенья приводятся в движение двумя тросами через систему роликов. Захватное устройство может удерживать объект любой формы, причем давление распределяется равномерно по всей линии контакта его звеньев с объектом.

Области применения таких схватов: пищевая промышленность, машиностроение, приборостроение, электронная промышленность, коллаборативная робототехника, оптовая и розничная торговля.

Цели работы: разработать 3D-модель манипулятора, разработать электрическую схему манипулятора, разработать управляющую программу, разработать программу для разпознавания координат объекта по фотографии, разработкать модель мягкого схвата, создать прототип манипулятора с мягким схватом.

ГЛАВА 1. АНАЛИТИЧЕСКИЙ ОБЗОР МАНИПУЛЯТОРОВ С МЯГКИМИ СХВАТАМИ

1.1. Схват Soft Robotics

Схват Soft Robotics (рис. 1.1) сделан из специальной резины и снабжен пневматическим приводом. Внешне он напоминает кисть с несколькими «пальцами» (количество «пальцев» может отличаться в зависимости от вида схвата). Когда схват манипулятора оказывается над предметом, сжатый воздух подается в «пальцы», и они сжимаются, надежно охватывая цель. После того как объект перемещен в нужную точку, воздух сбрасывается, и захват разжимается.

Благодаря своей мягкости, «пальцы» манипулятора легко приспосабливаются к предметам любой формы, не повреждая их. С одинаковой легкостью разработка Soft Robotics перемещает различные продукты питания, электронные компоненты и др. Это рациональное решение для линий, где нужно захватывать разнообразные по форме и весу предметы: вместо того чтобы тщательно обучать робота обращаться с каждым видом продукции, достаточно установить универсальный схват [14].

Рис. 1.1. Манипулятор с схватом Soft Robotics

Рис. 1.2. Схваты «Soft Robotics»

1.2. Схват по модели щупальца осьминога от «Festo»

Бионический захват состоит из мягкой силиконовой структуры, которой можно управлять пневматически. Если на него подается сжатый воздух, щупальце сгибается внутрь и может обхватывать соответствующий предмет, облегая его. Как и в случае с его естественной моделью, внутри силиконового щупальца расположены два ряда присосок. В то время как маленькие присоски на конце захвата работают пассивно, к более крупным присоскам может быть приложен вакуум, с помощью которого объект надежно прикрепляется к захвату. Это означает, что TentacleGripper может подбирать и удерживать предметы различной формы [10].

Благодаря своему мягкому материалу искусственное щупальце не только способно захватить аккуратно и безопасно. Оно также соответствует строгим критериям компонента мягкой робототехники и, таким образом, имеет большой потенциал для коллаборативной робототехники в будущем.

Для этого «Festo» тестирует захват на двух пневматических облегченных роботах одновременно, которые также были разработаны в Bionic Learning Network: BionicMotionRobot и BionicCobot. Оба робота являются полностью гибкими и могут быть бесконечно усилены с точки зрения их кинематики. Поэтому они могут взаимодействовать напрямую с людьми. Даже в случае

столкновения они безвредны и не должны быть защищены от рабочего, как обычные заводские роботы. Схват по модели щупальца осьминога от «Festo» представлен на рис. 1.3.

Рис. 1.3. Схват по модели щупальца осьминога от «Festo»

1.3. «Мягкий схват» для работы со стеклянными изделиями

Каждый промышленный робот – манипулятор состоит из двух основных компонентов: манипулятора и устройства управления. Первый отвечает за все необходимые движения, второй – за управление ими. Описывая конструкцию промышленного робота, трудно удержаться от сравнения его с человеком. Каждый промышленный робот имеет мозг – блок управления и механическую часть, включающую тело и руку. Тело робота – это, как правило, массивное основание (станина), а рука – многозвенный рычажный механизм – манипулятор. Чтобы рука могла совершать положенное ей многообразие движений, она имеет мышцы – привод. Задача мышц – преобразовывать сигналы блока управления в механические перемещения руки. На конце механической руки находится захватное устройство – схват.

Большинство промышленных роботов имеет 1 руку, но существуют и роботы, обладающие 2, 3 и более руками. Взглянув на руки промышленного робота, нетрудно определить сферу применения робота. Например, клешни из трёх крюков для круглых поковок, присоски, как у осьминога, для стеклянных листов, ковш для сыпучих материалов и т.д. Ещё проще разобраться с

функциями робота, если руки его снабжены специализированным инструментом: сверлом, краскопультом, сварочным оборудованием и др.

На выставке НТТМ-82 можно было наблюдать роботы, манипулирующие электролампами (рис. 1.4) [3]. Один из роботов имел захват в виде резиновых гофрированных «хоботков». Когда в кисть подавался воздух, «хоботки», раздуваясь, изгибались и захватывали лампу за тонкостенную стеклянную колбу осторожно, но прочно.

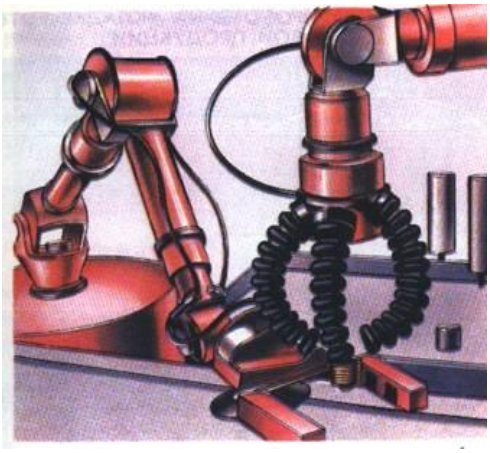

Рис. 1.4. «Мягкий схват» робота – манипулятора для работы с хрупкими стеклянными изделиями

1.4. Универсальные захваты

При создании универсальных захватных устройств для манипулирования твердыми, хрупкими и мягкими объектами произвольной формы могут использоваться многозвенные механизмы, способные совершать волнообразные движения. В таких механизмах звенья приводятся в движение двумя тросами через систему роликов. Захватное устройство может удерживать объекты разных форм, причем давление распределяется практически равномерно по всей линии контакта его звеньев с объектом. Плотное облегание по всем точкам контура обеспечивается как для выпуклых, так и для вогнутых профилей деталей. Необходимым условием являются размерные ограничения на звенья, которые должны представлять собой малые сегменты, не имеющие резких изломов по длине (рис. 1.5, а) [4].

Наличие двух приводных тросов объясняется их назначением: один разжимает звенья захватного устройства, если требуется освободить объект, другой обеспечивает их сжимание при захвате детали. Таким образом, устройство не имеет отдельных сервоприводов для каждого из звеньев, а приводится в действие при помощи натяжения общих для всех частей тросов. Это упрощает конструкцию захватного устройства в целом и повышает надежность его работы (рис. 1.5, а).

Равномерное давление на поверхность захватываемой детали, возможности регулирования усилия захвата в широком диапазоне и работы захватного устройства с объектами сложной формы обусловливают широкое применение многозвенных механических захватных устройств при обслуживании технологических процессов в разных отраслях промышленности. Однако, наличие передаточных звеньев, необходимость установки приводного элемента двухстороннего действия, а также использование тросовых передач снижают надежность захватных устройств и усложняют конструкцию. Кроме того, конструктивные ограничения при выборе минимального размера элементов многозвенной кинематической цепи приводят к возникновению дискретных точек касания звеньев с поверхностью захватываемого объекта и, как следствие, к концентрации нагрузок на отдельных участках поверхности объекта.

Этих недостатков можно избежать, если зажимной рабочий элемент выполнить в виде упругой гибкой ленты, концы которой жестко закреплены к корпусу 3 захватного устройства (рис. 1.5, б). Шток 4 пневматического привода 5 присоединен к среднему перегибу ленты, 2 крайних перегиба которой служат губками захватного устройства. Вся конструкция крепится к руке 6 манипулятора.

Описанное захватное устройство предназначено для захвата объектов, имеющих плоские поверхности, с которыми наилучшим образом взаимодействует эластичная лента. Наиболее универсальны захватные устройства с эластично-схватывающими зажимными элементами,

позволяющими производить захват объектов произвольных форм, жесткости, изготовленных из различных материалов. Такие захватные устройства обеспечивают необходимую податливость, адаптивность к форме объекта, равномерное распределение усилия зажима по всей его поверхности, т. е. оптимальные условия наложения удерживающих связей.

Захватное устройство с эластично-охватывающими рабочими элементами (рис. 1.5, в) состоит из корпуса 2 с размещенными в нем камерами 1 и силораспределительными нагнетателями 3, каналов 4 для подвода рабочего тела в полости камер 1 и амортизирующего сильфона 5. Захватное устройство крепится к руке 6 (или ее кисти) манипулятора.

К недостаткам рассмотренных универсальных камерных захватных устройств с эластично-охватывающими зажимными элементами относится избыточное давление на поверхность захватываемых объектов при наложении удерживающих связей. В ряде случаев, особенно при манипулировании объектов из хрупких материалов, избыточное давление недопустимо, а процесс удержания должен осуществляться за счет гравитационных сил (веса объекта). Вместе с тем сложная форма деталей и отсутствие четко выраженных базирующих поверхностей не допускают применения призматических, штыревых и других гравитационных захватных устройств.

Применение камерных захватных устройств с гранулированными наполнителями (рис. 1.5, в) также не всегда решает задачу захвата тел произвольной формы и жесткости, так как они не обеспечивают плотного прилегания материала рабочей камеры к поверхности захватываемого объекта, что приводит к неравномерным нагрузкам на эту поверхность. Кроме того, гранулированный сыпучий наполнитель не позволяет изменять усилия захвата. Поставленная задача может быть решена при использовании универсальных камерных захватных устройств с зажимными элементами в виде эластичных тонкостенных камер, заполненных электрореологической или магнитно-реологической жидкостью и снабженных соответственно управляемыми источниками электрического или магнитного поля.

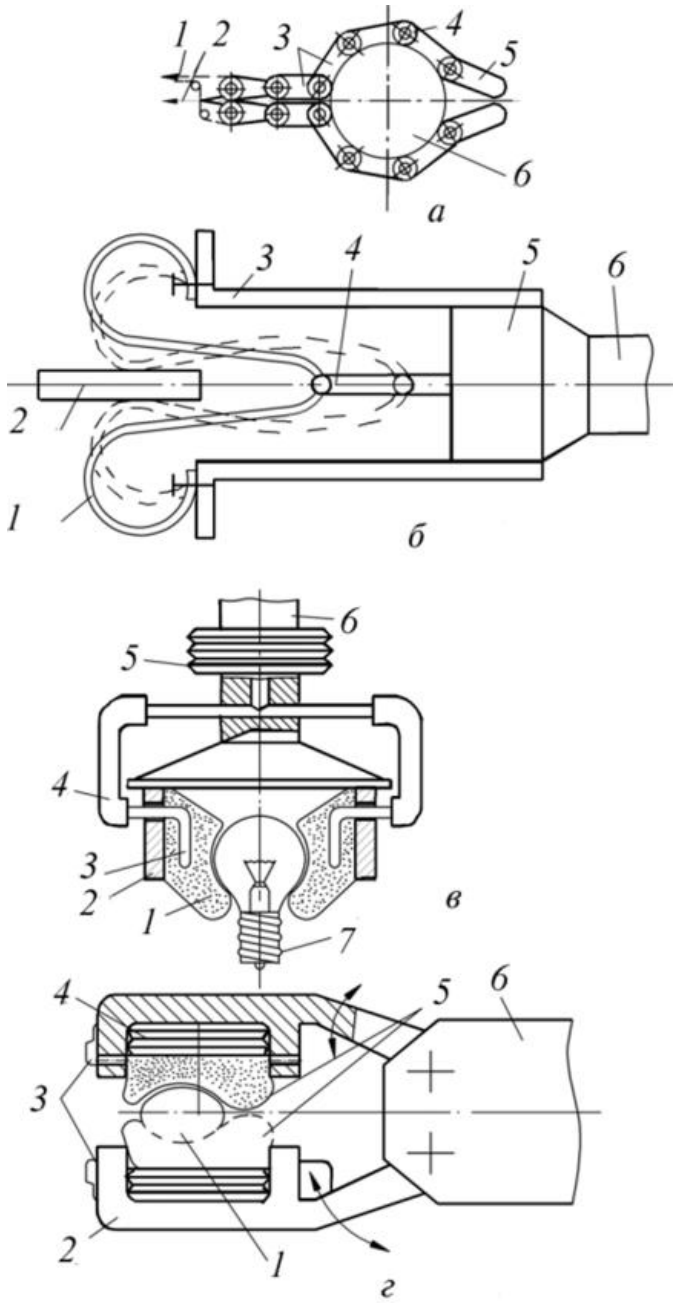

Рис. 1.5. Универсальные захватные устройства:
а - многозвенное; б - пружинное; в - с силораспределяющим наполнителем; г - с эластичными схватывающими элементами

1.5. Мягкие схваты компании OnRobot

Внедрение автоматизации в 20-м веке в значительной степени сформировало индустриальный мир, и роботы уже много лет используются в производстве. Сегодня небольшие, доступные по цене коллаборативные роботы-манипуляторы и легкие промышленные роботы вновь

революционизируют индустриальный мир, предоставляя производителям гибкость и повышенную производительность при меньших затратах.

Компания OnRobot разработала мягкий схват (рис. 1.6), у которого есть универсальное основание SG BASE PART и различные силиконовые (из жесткого и мягкого силикона) насадки SG SILICONE TOOLS [12].

Рис. 1.6. Мягкий схват компании OnRobot, состоящий из универсального основания и различных насадок

С помощью эластичных силиконовых насадок SG захват может обрабатывать самые разные заготовки для самых разных областей применения. При работе с одной и той же заготовкой применение различных насадок могут частично перекрываться, но насадки имеют разные характеристики, а также индивидуальную эффективность для данной заготовки. У некоторых конструкций насадок SG в верхней части захвата есть мягкая силиконовая часть. Эти насадки лучше подходят для обработки хрупких заготовок и/или заготовок с большой разницей в размерах по сравнению с инструментами из твердого силикона. Это связано с более «прощающей» природой мягкой части. Пользователь может испытывать снижение полезной нагрузки по сравнению с жесткими силиконовыми насадками. Чтобы правильно обрабатывать заготовку, пользователь должен знать некоторые параметры, которые определяются общими условиями заготовки и ее представления в приложении. Это помогает определить, какой инструмент выбрать, и фактическую ширину захвата на нем. Общий обзор таких параметров: форма, размеры, вес, шероховатость,

хрупкость, ориентация захвата/размещения. В табл. 1 представлены характеристики заготовок, используемых с насадкой SG-a-H. За шероховатость, равную 1 принята шероховатость полированной поверхности (например, полированного металла); 5 – «текстурированной» поверхности (например, картона); 10 – «грубой» поверхности (например, металла после пескоструйной обработки).

Таблица 1

Характеристики заготовок

Пример материала	Заготовка	Размеры, мм	Вес, г	Шерохо-ватость	Форма	Фактическая ширина захвата, мм
Гладкое дерево (шлифованное)	Круглый пруток	27	32	5	Цилиндр	20
Полированный металл	Алюминиевый куб	35x25	512	1	Квадрат	15
«Грубый» металл	Алюминиевый цилиндр	60	490	8	Цилиндр	55
Пластик	ПЭТ бутылка POM-C POM-C	65 50 50	431 221 1410	1 2 2	Цилиндр Цилиндр Цилиндр	50 42 15
Стекло	Стеклянный стакан	68	238	1	Цилиндр	50
Органический материал	Помидор Гриб Виноградина	54 40 20	92 8 7	2 10 10	Круглая Круглая Овальная	53 39 16
Углеволокно	Цилиндр из углеволокна	38	48	7	Цилиндр	29

1.6. Итог аналитического обзора

В результате аналитического обзора было решено создавать манипулятор с мягким схватом без использования пневматики, так как для создания пневматического схвата потребуется подключение дополнительного

оборудования такого, как насос, ресивер, распределитель и т.д, что значительно усложнит и удорожит конструкцию. Решено использовать классический жесткий схват с дополнительными деформируемыми пластинами.

ГЛАВА 2. РАЗРАБОТКА КОНСТРУКЦИИ МАНИПУЛЯТОРА

2.1. Кинематическая схема

Манипулятор имеет 4 степени подвижности, 3 – для наведения схвата в нужную точку, 1 – для ориентации схвата перпендикулярно поверхности стола. Манипулятор состоит из основания и стола 0, первой степени подвижности 1, второй степени подвижности 2, третьей степени подвижности 3, четвертой степени подвижности 4, схвата 5.

Кинематическая схема манипулятора представлена на рис. 2.1.

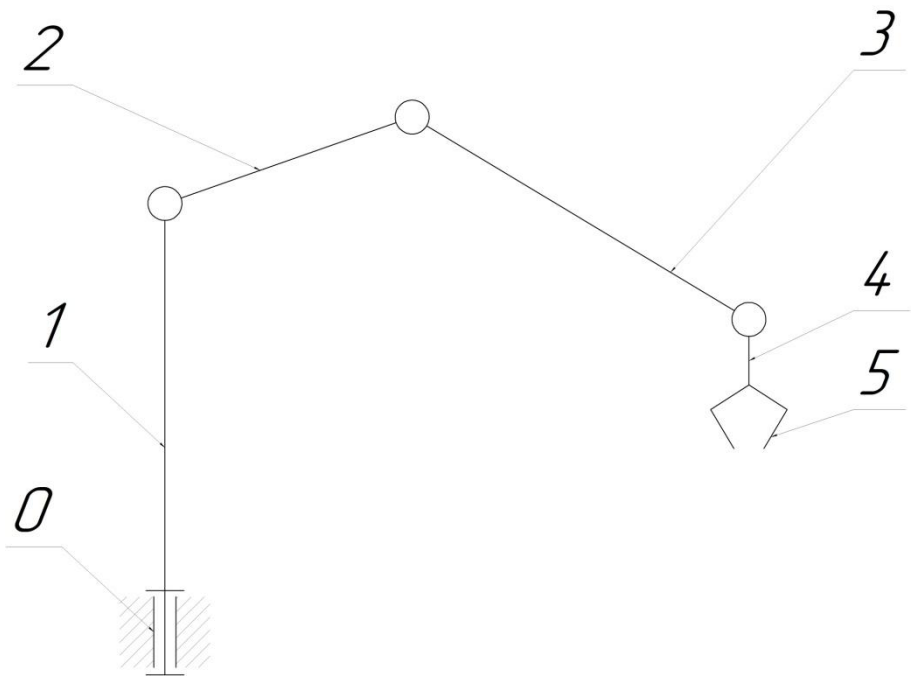

Рис. 2.1. Кинематическая схема манипулятора

2.2. Разработка 3D-модели манипулятора

Для создания прототипа манипулятора потребовалась разработка 3D-модели, которая представлена на рисунке 2.2. 3D-модель создавалась в программном комплексе САПР (Системе автоматизированного проектирования) SolidWorks, различные силовые расчеты проводились в его модуле SolidWorks Simulation, анимация движения создавалась при помощи

модуля Solidworks Motion. Рендеринг (создание фотореалистичного изображения) производился при помощи программы KeyShot 8.

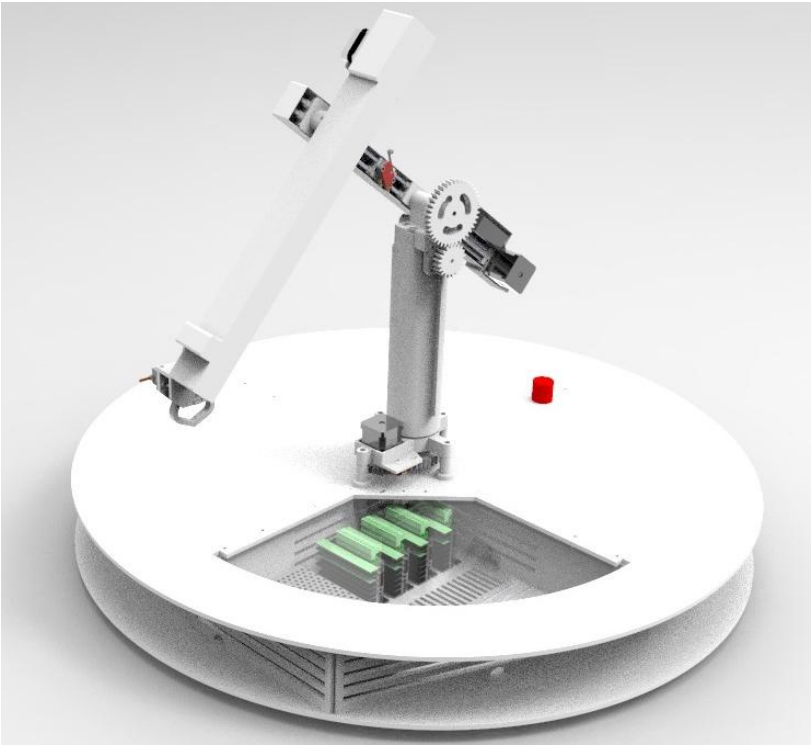

Рис. 2.2. 3D-модель манипулятора

2.3. Описание конструкции манипулятора

Стол манипулятора служит для размещения на его поверхности объектов для захвата. Также стол имеет отсек для расположения в нем электронных компонентов манипулятора, таких как блока питания, драйверов шаговых двигателей, микроконтроллера и макетной платы. Стол манипулятора был выполнен из фанеры толщиной 6 мм при помощи лазерной резки.

К поверхности стола крепится основание манипулятора (рис. 2.3). Вращение с шагового двигателя основания Nema 17HS4401 (момент удержания 4 кг*см) передается через 1-ступенчатую зубчатую передачу с передаточным отношением 2 на первую степень подвижности. Далее вращение с шагового двигателя 1-й степени подвижности Nema17 HS4401 (момент удержания 4 кг*см) передается через 1-ступенчатую зубчатую передачу с передаточным отношением 2 на вторую степень подвижности. Вторая степень подвижности

выполнена в виде рычага с противовесом (в качестве противовеса используется двигатель и дополнительный груз) для уравновешивания сил тяжести, действующих на руку манипулятора. Вращение с шагового двигателя 2-й степени подвижности Nema 17HS4401 (момент удержания 4 кг*см) передается через 1-ступенчатую ременную передачу с передаточным отношением 2,4 на третью степень подвижности. Третья степень подвижности также выполнена в виде рычага с противовесом (в качестве противовеса используется двигатель) для уравновешивания сил тяжести, действующих на руку манипулятора. Вращение с шагового двигателя 3-й степени подвижности Nema 17HS2408 (момент удержания 4 кг*см) передается через 1-ступенчатую ременную передачу с передаточным отношением 2 на четвертую степень подвижности. Четвертая степень подвижности интегрирована со схватом манипулятора. Губки схвата манипулятора приводятся в движение при помощи сервомотора TOWER PRO MG-995 (момент 10 кг/см, угол поворота 180 градусов). На губки схвата манипулятора крепятся насадки с деформируемыми пластинами из резины, которые выполняют роль мягкого схвата.

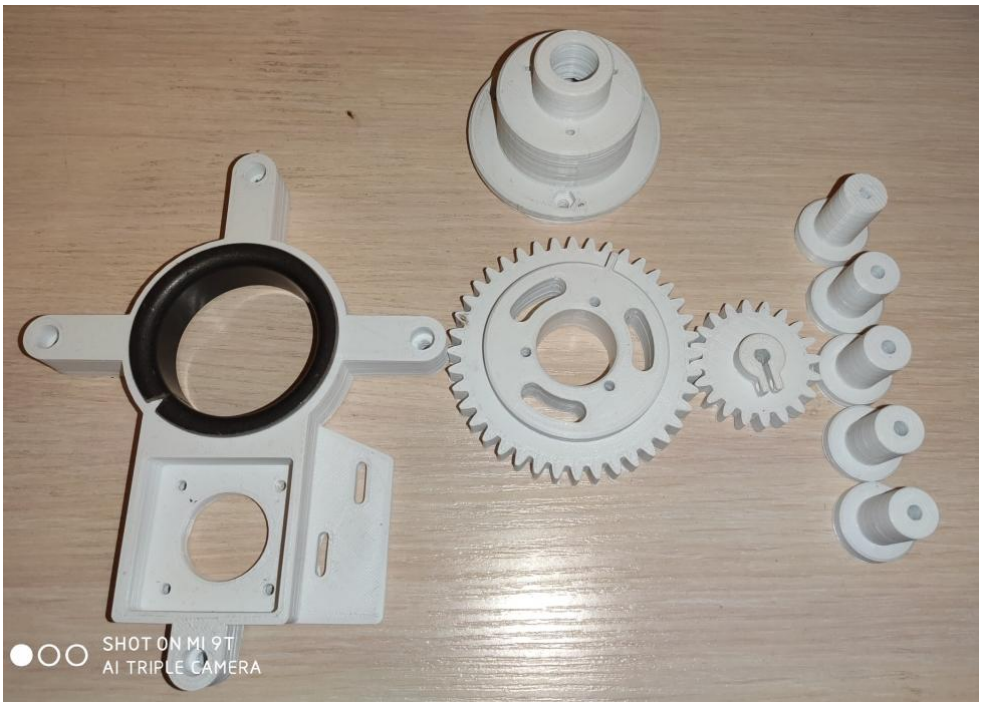

Рис. 2.3. Компоненты основания манипулятора

Большинство компонентов манипулятора были выполнены при помощи печати на 3D-принтере из ABS-пластика, который обладает повышенной ударопрочностью, высокой эластичностью, высокой долговечностью в отсутствии прямого солнечного света. Также данный пластик легко поддается механической обработке [15]. Для уменьшения сил трения и износа пластиковых компонентов были использованы подшипники скольжения, которые в сравнении с подшипниками качения имеют меньшие габариты, но больший коэффициент трения (что допустимо при работе на небольших скоростях). Выбор крепежных элементов, используемых при сборке манипулятора, осуществлялся при помощи справочника В.И. Анурьева [1]. Подбор допусков на изготовление некоторых деталей манипулятора осуществлялся при помощи справочника В.И. Анухина [2]. Для установки манипулятора в начальную позицию используется 4 концевых датчика (по 1 на каждую степень подвижности). Для ориентации в пространстве губок схвата дополнительного датчика не требуется, т.к. приводом схвата является сервомотор, который «знает» свой угол поворота. На рис. 2.4 представлен готовый прототип манипулятора.

Рис. 2.4. Прототип манипулятора

Над столом на кронштейне (рис. 2.5) установлена веб-камера, подключенная к компьютеру и передающая на него фотографию стола сверху. По этой фотографии вычисляются координаты объекта для захвата.

Рис. 2.5. Веб-камера, установленная на кронштейне

В табл. 2 представлены характеристики разработанного прототипа манипулятора.

Таблица 2

Характеристики прототипа манипулятора с мягким схватом

№	Характеристика	Значение
1	Радиус рабочей зоны манипулятора (по поверхности стола)	400 мм
2	Максимальная высота подъема схвата манипулятора	500 мм
3	Грузоподъемность	250 г
4	Точность позиционирования	±10 мм
5	Количество степеней подвижности	4

2.4. Структурная схема манипулятора

Управление манипулятора осуществляется через контроллер Arduino Mega 2560R3. При помощи Arduino IDE на него записана циклически выполняющаяся управляющая программа. Контроллер подает импульсы на

драйвера шаговых двигателей TB 6600, отвечающих за вращение шаговых двигателей четырех степеней подвижности. Один импульс соответствует повороту шагового двигателя на 1/200 оборота при полном шаге (возможно использовать деление шага, тогда 1 импульсу может соответствовать поворот шагового двигателя на 1/400, 1/800, 1/1600, 1/3200 и 1/6400 оборота) Также с контроллера на драйвер поступает информация о направлении вращения (Сигнал DIR — Потенциальный сигнал, сигнал направления. Логическая единица — ШД вращается по часовой стрелке, ноль — ШД вращается против часовой стрелки, или наоборот, инвертировать сигнал DIR обычно можно либо из программы управления или поменять местами подключение фаз ШД в разъеме подключения в драйвере). Также в данном типе драйвера шагового двигателя возможно регулировка тока от 0,5 до 3,5 А, что позволяет регулировать момент на шаговых двигателях. Питание драйверов осуществляется при помощи импульсного блока питания 20 В. Для вращения трех степеней подвижности использовались шаговые двигатели Nema 17HS4401 (момент удержания 4 кг*см), для вращения 4-ой - Nema 17HS2408 (момент удержания 1,2 кг*см). Приводом схвата является сервомотор. Также к контроллеру подключены концевые датчики для 4-х степеней подвижности, которые служат для установки манипулятора в начальную позицию. Контроллер соединен с компьютером по USB порту, через который осуществляется питание и передается информация о координатах (x,y) объекта для схватывания (координаты вычисляются по изображению, полученному с веб-камеры). Дополнительное питание контроллера осуществляется через блок питания 5 В.

Структурная схема манипулятора представлена на рисунке 2.6.

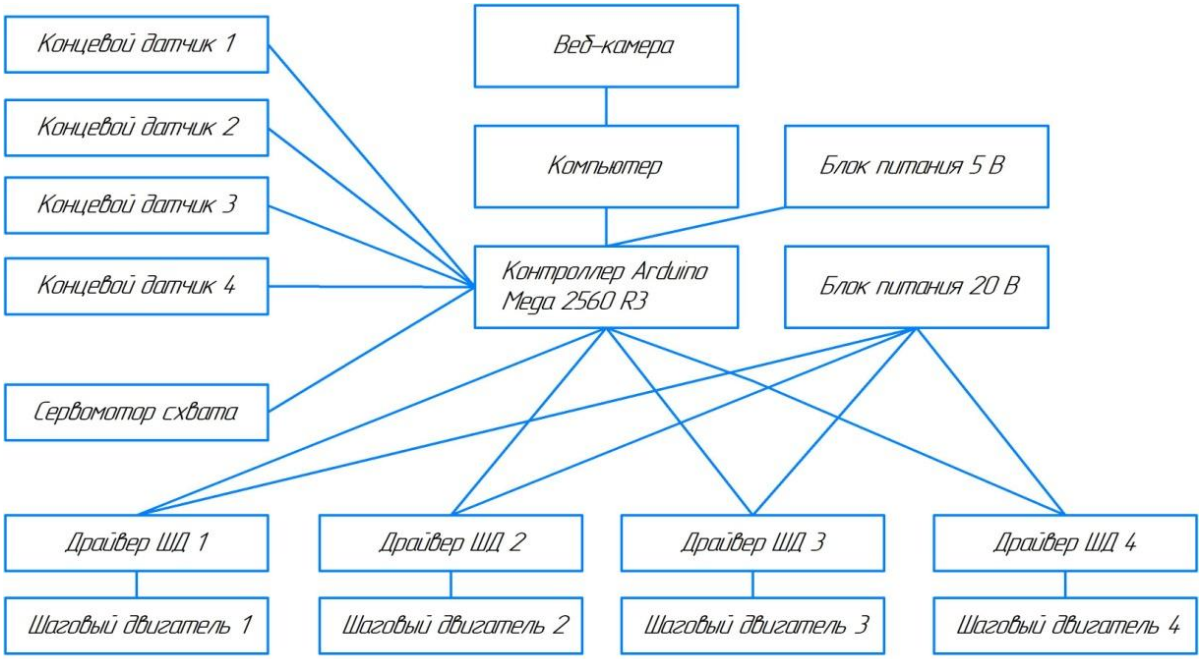

Рис. 2.6. Структурная схема манипулятора

2.5. Разработка мягкого схвата

В результате аналитического обзора в качестве мягкого схвата было решего использовать классический жесткий схват с дополнительными деформируемыми пластинами. Было предложено 2 варианта конструкции: резиновая пластина крепится на поверхность жесткого схвата всей площадью, т.е. задняя поверхность деформируемой пластины полностью заделана; резиновая пластина крепится на поверхность жесткого схвата частично, т.е. задняя поверхность деформируемой пластины частично заделана. При помощи Solidworks Simulation были смоделированы деформации 2 предложенных типов конструкций под нагрузкой 10Н. Результаты расчета приведены на рисунках 2.7, 2.8.

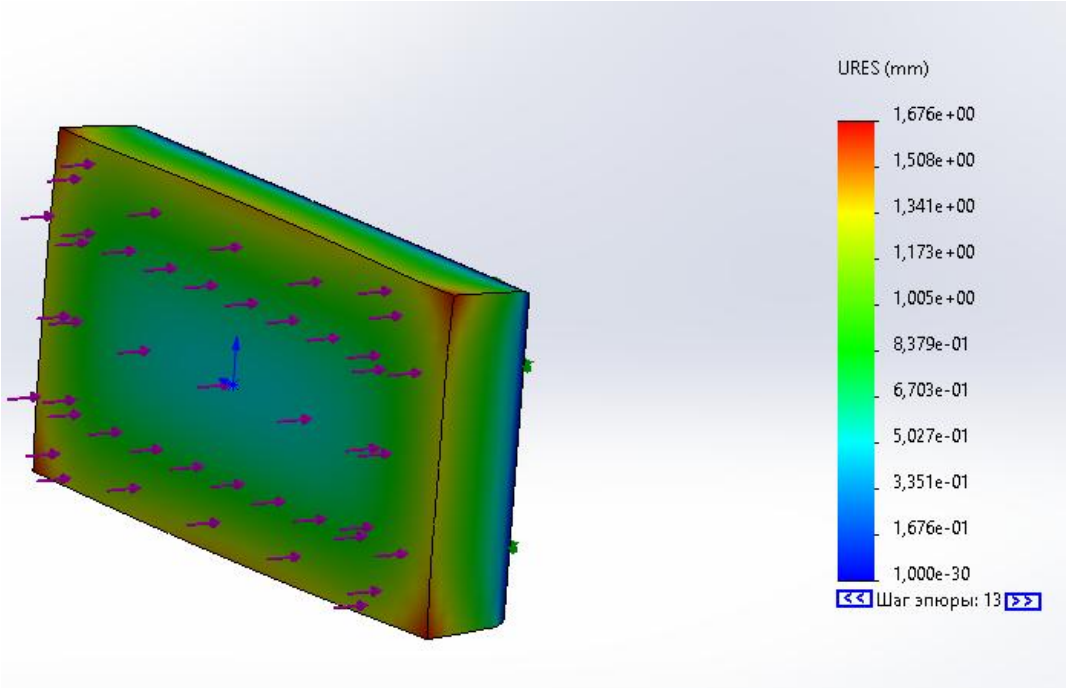

Рис. 2.7. Деформация пластины 60x30x10 мм из резины с полностью заделанной задней стенкой под нагрузкой 10Н

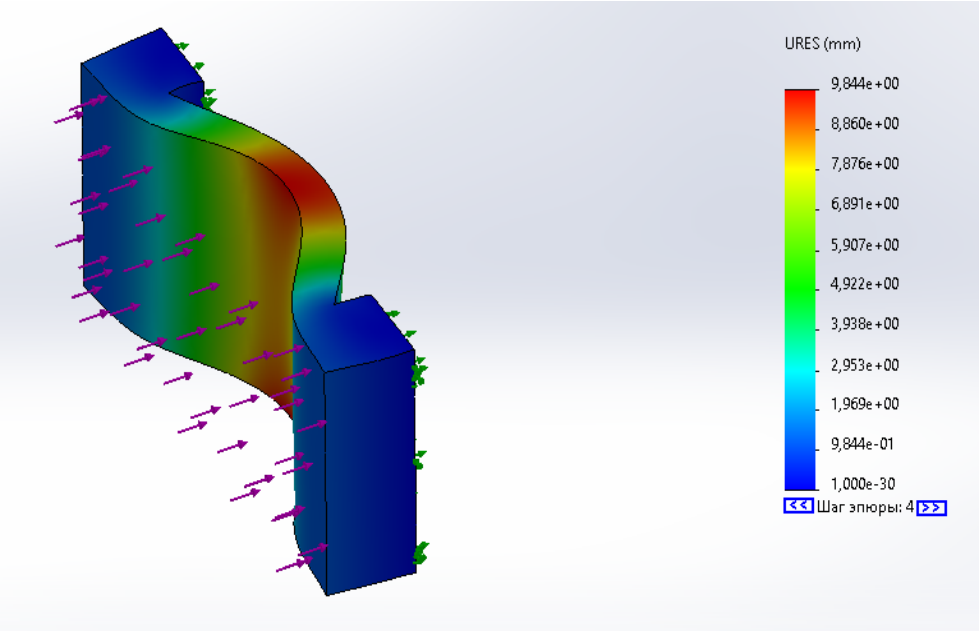

Рис. 2.8. Деформация пластины из резины с частично заделанной задней стенкой под нагрузкой 10Н

Расчеты показали, что деформация резиновой пластины с полностью заделанной задней стенкой недостаточна (порядка 1 мм при 10 Н распределенной нагрузки). При частично заделанной задней стенке деформация составляет порядка 10 мм при той же нагрузке. Таким образом для создания

мягкого схвата больше подходит пластина с частично заделанной задней стенкой. Были разработаны и произведены съемные насадки для схвата, которые представлены на рис. 2.9. Каркас насадок выполнен из алюминиевого профиля, деформируемая часть выполнена из листовой резины. Крепление к губками жесткого схвата осуществляется винтами.

Рис. 2.9. Съемные «мягкие» насадки для схвата

ГЛАВА 3. СОЗДАНИЕ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

3.1. Решение обратной задачи кинематики

Обратная задача кинематики состоит в определении углов поворота φ1, φ2, φ3, φ4, звеньев манипулятора, определяемых по координатам Sx, Sy, Sz рабочего органа манипулятора. Схема с обозначением углов поворота звеньев манипулятора и их длин a,b,c,d представлена на рис. 3.1.

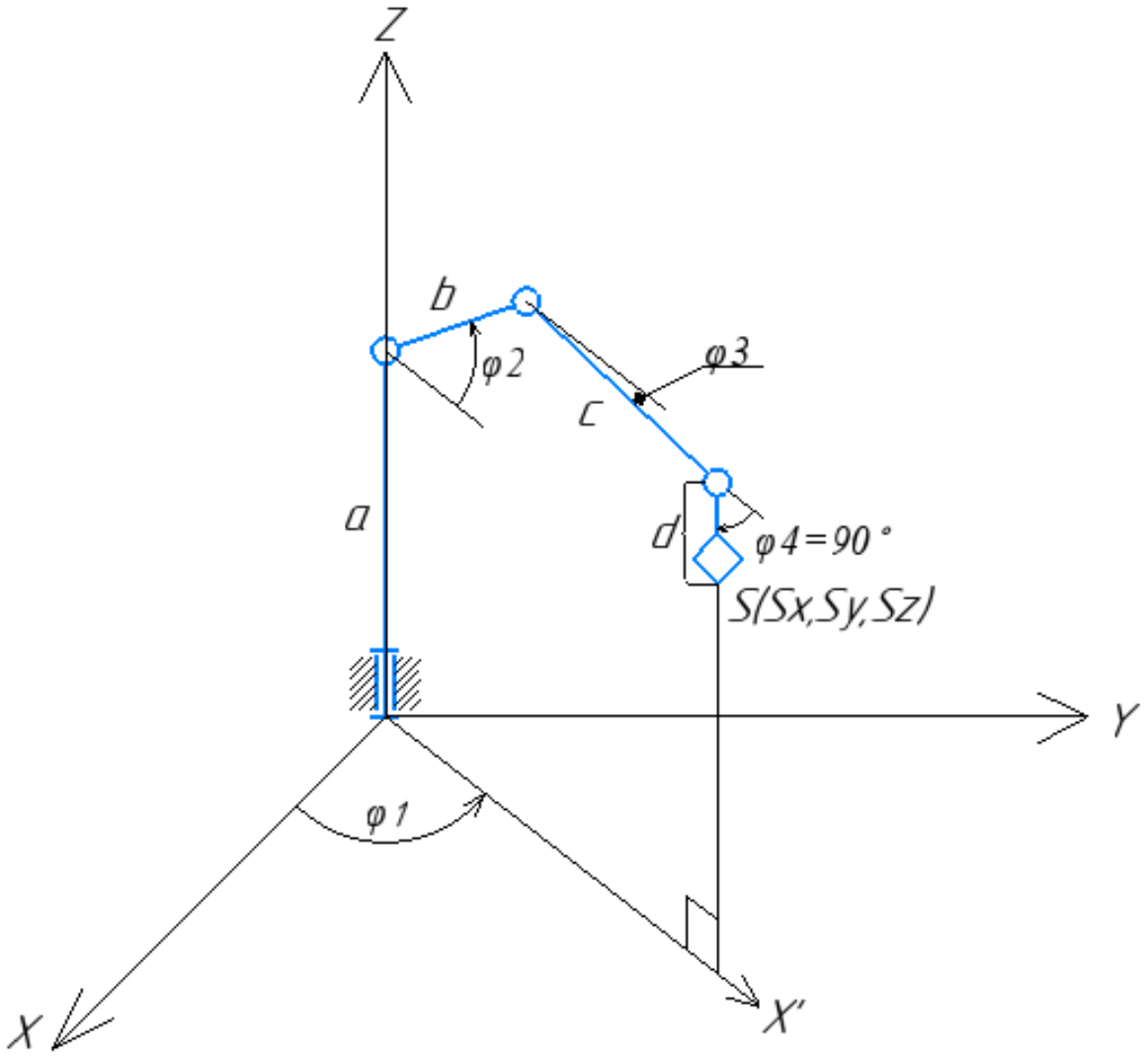

Рис. 3.1. Схема с обозначением углов поворота звеньев и их длин

Угол поворота первого звена манипултятора определяется по формуле (3.1) .

$$\varphi 1 = \mathrm{atan2}(\mathrm{Sy}, \mathrm{Sx}) \tag{3.1}$$

$$atan2(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) \text{ если } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi \text{ если } x < 0 \text{ и } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi \text{ если } x < 0 \text{ и } y < 0 \\ +\frac{\pi}{2} \text{ если } x = 0 \text{ и } y > 0 \\ -\frac{\pi}{2} \text{ если } x = 0 \text{ и } y < 0 \\ \text{неопределенно если } x = 0 \text{ и } y = 0 \end{cases} \quad (3.2)$$

Для нахождения углов поворота φ2, φ3 второго и третьего звеньев манипулятора решается система уравнений (3.3).

$$\begin{cases} b \cdot \sin(\varphi 2) - \mathrm{c} \cdot \sin(\varphi 3) = Sz - a + d \\ b \cdot \cos(\varphi 2) + \mathrm{c} \cdot \cos(\varphi 3) = \left|\frac{Sx}{\cos(\varphi 1)}\right| \end{cases} \quad (3.3)$$

Примем $\mathrm{z}' = \mathrm{Sz} - \mathrm{a} + \mathrm{d}$; $\mathrm{x}' = |\mathrm{Sx}/\mathrm{co\,s}(\varphi 1)|$.

Возведем уравнения системы (3.3) в квадрат (3.4).

$$\begin{cases} \mathrm{b}^2 \cdot \sin^2(\varphi 2) - 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \sin(\varphi 2) \cdot \sin(\varphi 3) + \mathrm{c}^2 \cdot \sin^2(\varphi 3) = (\mathrm{z}')^2 \\ \mathrm{b}^2 \cdot \cos^2(\varphi 2) + 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \cos(\varphi 2) \cdot \cos(\varphi 3) + \mathrm{c}^2 \cdot \cos^2(\varphi 3) = (\mathrm{x}')^2 \end{cases} \quad (3.4)$$

Сложим оба уравнения вместе (3.5).

$$b^2 \cdot \sin^2(\varphi 2) + b^2 \cdot \cos^2(\varphi 2) - 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \sin(\varphi 2) \cdot \sin(\varphi 3) + 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot$$
$$\cdot \cos(\varphi 2) \cdot \cos(\varphi 3) + \mathrm{c}^2 \cdot \cos^2(\varphi 3) + \mathrm{c}^2 \cdot \sin^2(\varphi 3) = \mathrm{b}^2 + \mathrm{c}^2 + 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \quad (3.5)$$
$$\cdot \cos(\varphi 2) \cdot \cos(\varphi 3) - 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \sin(\varphi 2) \cdot \sin(\varphi 3) = (\mathrm{z}')^2 + (\mathrm{x}')^2$$

Преобразуем, используя формулу косинуса суммы (3.6).

$$2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \cos(\varphi 2 + \varphi 3) = (\mathrm{z}')^2 + (\mathrm{x}')^2 - \mathrm{b}^2 - \mathrm{c}^2 \quad (3.6)$$

После дальнейших преобразований получим (3.7).

$$\cos(\varphi 2 + \varphi 3) = \frac{(\mathrm{z}')^2 + (\mathrm{x}')^2 - \mathrm{b}^2 - \mathrm{c}^2}{2 \cdot \mathrm{b} \cdot \mathrm{c}} \quad (3.7)$$

Таким образом, сумма углов поворота второго и третьего звена φ2+φ3 (3.8) является арккосинусом выражения, полученного в (3.7).

$$\varphi 2 + \varphi 3 = \arccos\left(\frac{(\mathrm{z}')^2 + (\mathrm{x}')^2 - \mathrm{b}^2 - \mathrm{c}^2}{2 \cdot \mathrm{b} \cdot \mathrm{c}}\right) \quad (3.8)$$

Обозначим φ23=φ2+φ3. Далее (3.9) преобразуем первое уравнение системы (3.3) для нахождения угла φ2.

$$\begin{gathered} b \cdot \sin(\varphi 2) - c \cdot \sin(\varphi 23 - \varphi 2) = b \cdot \sin(\varphi 2) + c \cdot \sin(\varphi 2 - \varphi 23) = \\ = b \cdot \sin(\varphi 2) + c \cdot \sin b \cdot \sin(\varphi 2) + c \cdot (\sin(\varphi 2) \cdot \cos(\varphi 23) - \cos(\varphi 2) \cdot \\ \cdot \sin\varphi 23 = z' \end{gathered} \tag{3.9}$$

Далее (3.10) поделим полученное выражение на c и приведем слагаемые.

$$\begin{gathered} \frac{b}{c} \cdot \sin(\varphi 2) + \sin(\varphi 2) \cdot \cos(\varphi 23) - \cos(\varphi 2) \cdot \sin(\varphi 23) = \\ = \sin(\varphi 2)\left(\frac{b}{c} + \cos(\varphi 23)\right) - \cos(\varphi 2) \cdot \sin(\varphi 23) = \frac{z'}{c} \end{gathered} \tag{3.10}$$

Затем (3.11) воспользуемся формулами универсальной тригонометрической подстановки (выражением косинуса и синуса угла через тангенс половины угла).

$$\begin{gathered} 2 \cdot \mathrm{tg}\left(\frac{\varphi 2}{2}\right)\left(\frac{b}{c} + \cos(\varphi 23)\right) - \left(1 - \mathrm{tg}^2\left(\frac{\varphi 2}{2}\right)\right) \cdot \sin(\varphi 23) - \\ - \frac{z'}{c}\left(1 + \mathrm{tg}^2\left(\frac{\varphi 2}{2}\right)\right) = 0 \end{gathered} \tag{3.11}$$

Обозначим $m = b/c + \cos(\varphi 23)$; $l = \sin(\varphi 23)$; $k = z'/c$. Получим формулу (3.12).

$$\begin{gathered} 2 \cdot m \cdot \mathrm{tg}\left(\frac{\varphi 2}{2}\right) - \left(1 - \mathrm{tg}^2\left(\frac{\varphi 2}{2}\right)\right) \cdot l - k \cdot \left(1 + \mathrm{tg}^2\left(\frac{\varphi 2}{2}\right)\right) \\ = (l - k) \cdot \mathrm{tg}^2\left(\frac{\varphi 2}{2}\right) + 2 \cdot m \cdot \mathrm{tg}\left(\frac{\varphi 2}{2}\right) - (k + l) = 0 \end{gathered} \tag{3.12}$$

Найдем решение данного квадратного уравнения (3.13).

$$\mathrm{tg}\left(\frac{\varphi 2}{2}\right) = \frac{-2 \cdot m \pm \sqrt{4 \cdot m^2 + 4(l - k)(l + k)}}{2(l - k)} = \frac{-m \pm \sqrt{m^2 + l^2 - k^2}}{l - k} \tag{3.13}$$

Таким образом, угол φ2 равен удвоенному арктангенсу выражения, полученного в (3.13).

$$\varphi 2 = 2 \cdot \mathrm{arctg}\left(\frac{-m \pm \sqrt{m^2 + l^2 - k^2}}{l - k}\right) \tag{3.14}$$

Т.к. схват должен постоянно быть перпендикулярным столу то угол поворота четвертого звена составляет 90°.

Полное решение обратной задачи кинематики с учетом диапазонов углов (диапазоны определяются геометрически при анализе конструкции манипулятора) представлено в системе (3.15). Если полученные углы не попадают в диапазон, то позиция не достижима. Решение системы (3.3) также было проверено при помощи функции Given-Find в программе MathCad. Полученные 2-мя методами решения совпали, что говорит о правильности решения системы.

$$\begin{cases} \varphi 1 = \mathrm{atan2}(\mathrm{Sy}, \mathrm{Sx}) \\ 0° \leq \varphi 1 \leq 360° \\ \varphi 2 = 2 \cdot \mathrm{arctg}\left(\dfrac{-\mathrm{m} \pm \sqrt{\mathrm{m}^2 + \mathrm{l}^2 - \mathrm{k}^2}}{\mathrm{l} - \mathrm{k}}\right) \\ -45° \leq \varphi 2 \leq 45° \\ \varphi 3 = \arccos\left(\dfrac{(\mathrm{z}')^2 + (\mathrm{x}')^2 - \mathrm{b}^2 - \mathrm{c}^2}{2 \cdot \mathrm{b} \cdot \mathrm{c}}\right) - \varphi 2 \\ -20° \leq \varphi 3 \leq 120° \\ \varphi 4 = 90° \end{cases} \tag{3.15}$$

3.2. Выбор платформы

Для создания управляющей программы была выбрана платформа Arduino.

Arduino - это электронная платформа с открытым исходным кодом, основанная на простом в использовании аппаратном и программном обеспечении. Платы Arduino могут считывать входные данные - свет на сенсоре, палец на кнопке или сообщение в Твиттере - и превращать его в выходной сигнал - активировать мотор, включать светодиод и публиковать что-то в Интернете. Можно сообщить своей плате, что делать, отправив набор инструкций микроконтроллеру на плате. Для этого используется язык программирования Arduino (на основе Wiring) и программное обеспечение Arduino (IDE (Integrated Development Environment) - Интегрированная среда разработки) на основе Processing.

Благодаря простому и доступному пользовательскому интерфейсу Arduino используется в тысячах различных проектов и приложений. Программное обеспечение Arduino простое в использовании для начинающих, но достаточно гибкое для опытных пользователей. Оно работает на операционных системах: Mac, Windows и Linux.

Есть много других микроконтроллеров и платформ микроконтроллеров, доступных для физических вычислений. Parallax Basic Stamp, Netmedia BX-24, Phidgets, Handyboard MIT и многие другие предлагают аналогичную функциональность. Все эти инструменты скрывают сложные детали программирования микроконтроллера и упаковывают их в простой в использовании инструмент. Arduino также упрощает процесс работы с микроконтроллерами, но предлагает учителям, студентам и любителям некоторые преимущества перед другими системами:

1. Дешевизна. Платы Arduino относительно недороги по сравнению с другими платформами микроконтроллеров. Самая дешевая версия модуля Arduino может быть собрана вручную, и даже предварительно собранные модули Arduino стоят менее 50 долларов.
2. Кроссплатформенность. Программное обеспечение Arduino (IDE) работает в операционных системах Windows, Macintosh OSX и Linux. Большинство микроконтроллерных систем ограничены Windows.
3. Простая, понятная среда программирования - Arduino Software (IDE) проста в использовании для новичков, но достаточно гибка для опытных пользователей, чтобы также воспользоваться ее преимуществами.
4. Программное обеспечение с открытым исходным кодом и расширяемое программное обеспечение. Программное обеспечение Arduino выпускается в виде инструментов с открытым исходным кодом, доступных для расширения опытными программистами. Язык может быть расширен с помощью библиотек C ++, и люди, желающие понять технические детали, могут перейти от Arduino к языку программирования

AVR-C, на котором он основан. Точно так же можно добавить код AVR-C непосредственно в программы Arduino.

5. Открытый исходный код и расширяемое аппаратное обеспечение. Схемы плат Arduino публикуются под лицензией Creative Commons, поэтому опытные конструкторы схем могут создать собственную версию модуля, расширяя его и улучшая его. Даже относительно неопытные пользователи могут создать макетную версию модуля, чтобы понять, как он работает, и сэкономить деньги.

3.3. Описание управляющей программы

В начале выполнения программы происходит импортирование библиотеки для работы с сервомотором. Также определяются переменные, отвечающие за пины (направление и импульс) подключаемых шаговых двигателей, концевых датчиков. Происходит инициализация переменных для работы 4-х степеней подвижности: «isPosReset» - переменная, принимающая истинное значение, когда степень подвижности выставляется в нулевую позицию; «initPos» - начальный угол поворота степени подвижности манипулятора; «currentPos» - текущий угол поворота степени подвижности манипулятора; «i» - передаточное отношение между шаговым двигателем и выходным валом; «stepsPerRev» - количество импульсов, требуемых поворота вала шагового двигателя на 360 градусов; «dir» - переменная, отвечающая за направление вращения шагового двигателя; «delay» - задержка в миллисекундах; a,b,c,d – длины звеньев манипулятора; X,Y,Z – массивы для хранения координат.

```
//defines pins
#include <Servo.h> //используем библиотеку для работы с сервоприводом
//#include <math.h>
Servo gripperServo; //объявляем переменную gripperServo типа Servo
#define stepPin1 2  //PUL -Pulse
#define dirPin1 3 //DIR -Direction
//#define enPin1 4  //ENA -Enable
#define stepPin2 4  //PUL -Pulse
#define dirPin2 5 //DIR -Direction
#define stepPin3 6  //PUL -Pulse
#define dirPin3 7 //DIR -Direction
#define stepPin4 8  //PUL -Pulse
#define dirPin4 9 //DIR -Direction
#define button1 10
#define button2 11
#define button3 12
#define button4 13
bool isPosReset1;          bool isPosReset2;          bool isPosReset3;                          bool isPosReset4;
float initPos1 = 0;        float initPos2 = 47;       float initPos3 = 0;                        float initPos4 = 68;
float currentPos1;         float currentPos2;         float currentPos3;                         float currentPos4;
int i1 = 2;                int i2 = 2;                int i3 = 24;/*increased 10 times*/         int i4 = 2;
int stepsPerRev1 = 3200;   int stepsPerRev2 = 3200;   int stepsPerRev3 = 320;/*reduced 10 times*/ int stepsPerRev4 = 3200;
bool dir1;                 bool dir2;                 bool dir3;                                 bool dir4;
int delay1 = 1500;         int delay2 = 1500;         int delay3 = 1500;                         int delay4 = 1500;
int a = 364; int b = 195; int c = 298; int d = 100; //lengthes of mechanism links

int X [21]; //points coordinates
int Y [21];
int Z [21];
int G [21];
int sim = 0;
bool arePointsReached = false;
```

Рис. 3.2. Инициализация переменных

Далее выполняется функция setup(). Эта функция вызывается, когда стартует скетч. Используется для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д. Функция setup запускает только один раз, после каждой подачи питания или сброса платы Arduino. Режим работы заданного вход/выхода(pin) как входа или как выхода устанавливается при помощи функции pinMode() [8]. Выводы концевых выключателей button1, button2, button3, button4 включаются в режиме INPUT_PULLUP, что означает чтение информации с данных пинов с использованием подтягивающих резисторов, что позволяет избежать помех. Выводы шаговых двигателей отвечающих за направление и за шаг stepPin1, dirPin1 – stepPin4, dirPin4 включены в режиме OUTPUT. Это значит, что на данные выходы подаются импульсы. Далее переменным, показывающим, выставлена ли стартовая позиция, присваивается значение «ложь». Потом задается направление вращения шаговых двигателей, сервомотор привязывается к 22 пину и его губки сводятся командой servo.write(140), где

140-это угол поворота сервомотора. Данный фрагмент кода представлен на рис. 3.3.

```
void setup() {
  Serial.begin(9600);
  Serial.setTimeout(2000);
  //Sets the pins as Outputs
  pinMode(button1, INPUT_PULLUP);
  pinMode(button2, INPUT_PULLUP);
  pinMode(button3, INPUT_PULLUP);
  pinMode(button4, INPUT_PULLUP);
  pinMode(stepPin1, OUTPUT);
  pinMode(dirPin1, OUTPUT);
  pinMode(stepPin2, OUTPUT);
  pinMode(dirPin2, OUTPUT);
  pinMode(stepPin3, OUTPUT);
  pinMode(dirPin3, OUTPUT);
  pinMode(stepPin4, OUTPUT);
  pinMode(dirPin4, OUTPUT);
  isPosReset1 = false;
  isPosReset2 = false;
  isPosReset3 = false;
  isPosReset4 = false;
  //for stepper motor HIGH-CW;LOW-CCW
  //for face side of stepper motor
  dir1 = LOW;//for output shaft HIGH-CCW;LOW-CW
  dir2 = LOW;//for output shaft HIGH-CCW;LOW-CW
  dir3 = LOW;//for output shaft HIGH-CW;LOW-CCW
  dir4 = LOW;//for output shaft HIGH-CW;LOW-CCW
  gripperServo.attach(22); //привязываем привод к порту 22
  gripperServo.write(140);
}
```

Рис. 3.3. Функция setup()

Далее (см. рис. 3.4) следует выставление стартовой позиции. Чтобы избежать столкновения со столом, сначала выставляется вторая степень подвижности, происходит вращение до нажатия концевого датчика (чтение значения датчика происходит при помощи команды digitalRead() [7]). Аналогичным образом выставляются первая и третья степени подвижности. Выставление четвертой степени подвижности происходит при вращении степени до нажатия концевого датчика, а затем поворотом в противоположную сторону на 70 градусов, чтобы обеспечить перпендикулярность схвата поверхности стола.

```
void loop() {
  ///Setting to start position
  if (!isPosReset2) {
    digitalWrite(dirPin2, dir2);
    while (true) {
      if (digitalRead(button2) == HIGH)break;
      digitalWrite(stepPin2, HIGH);
      delayMicroseconds(delay2);
      digitalWrite(stepPin2, LOW);
      delayMicroseconds(delay2);
    }
    isPosReset2 = true;
    currentPos2 = initPos2;
  }
  if (!isPosReset1) {
    digitalWrite(dirPin1, dir1);
    while (true) {
      if (digitalRead(button1) == HIGH)break;
      digitalWrite(stepPin1, HIGH);
      delayMicroseconds(delay1);
      digitalWrite(stepPin1, LOW);
      delayMicroseconds(delay1);
    }
    isPosReset1 = true;
    currentPos1 = initPos1;
  }
  if (!isPosReset3) {
    digitalWrite(dirPin3, dir3);
    while (true) {
      if (digitalRead(button3) == HIGH)break;
      digitalWrite(stepPin3, HIGH);
      delayMicroseconds(delay3);
      digitalWrite(stepPin3, LOW);
      delayMicroseconds(delay3);
    }
    isPosReset3 = true;
    currentPos3 = initPos3;
  }
  if (!isPosReset4) {
    while (true) {
      if (digitalRead(button4) == HIGH)break;
      digitalWrite(stepPin4, HIGH);
      delayMicroseconds(delay4);
      digitalWrite(stepPin4, LOW);
      delayMicroseconds(delay4);
    }
    isPosReset4 = true;
    dir4 = !dir4;
    digitalWrite(dirPin4, dir4);
    for (int i = 0; i < initPos4 * i4 * stepsPerRev4 / 360; i++) {
      digitalWrite(stepPin4, HIGH);
      delayMicroseconds(delay4);
      digitalWrite(stepPin4, LOW);
      delayMicroseconds(delay4);
```

Рис. 3.4. Выставление стартовой позиции

Затем следует чтение координат (x,y) с последовательного порта (сначала проверяется доступность порта командой Serial.available() [9]) при помощи метода Serial.ParseInt() [6]. Координате z присваивается значение 100 мм, что соответствует положению схвата над объектом. Потом задается точка с координатой z 40 мм, соответствующая положению объекта. Также задается дополнительная точка сброса объекта. За угол раскрытия схвата в каждой точке отвечает массив G. Значение угла 140 градусов соответствует закрытому схвату, 40 градусов – раскрытому. Заполнение массивов с координатами точек представлено на рис. 3.5.

```
if (!arePointsReached) {
  while (Serial.available())
  {
    siz = Serial.parseInt();
    if (siz > 7) siz = 7;
    for (int i = 0; i < 3 * siz; i++) {
      if (i % 3 == 0) {
        X[i] = Serial.parseInt();
        Y[i] = Serial.parseInt();
        Z[i] = 100;
        G[i] = 40;
      }
      if (i % 3 == 1) {
        X[i] = X[i - 1];
        Y[i] = Y[i - 1];
        Z[i] = 30;
        G[i] = 140;
      }
      if (i % 3 == 2) {
        X[i] = 300;
        Y[i] = 1;
        Z[i] = 60;
        G[i] = 40;
      }
    }
  }
```

Рис. 3.5. Заполнение массивов с координатами точек

Далее следует решение обратной задачи кинематики (см. рис. 3.6) (вычисление углов поворота звеньев манипулятора по координатам x,y,z рабочего органа манипулятора) для полученных точек.

```
for (int i = 0; i < 3*siz; i++) {
  float angle1;     float targetPos1;   long stepsNumber1;
  float angle2;     float targetPos2;   long stepsNumber2;
  float angle3;     float targetPos3;   long stepsNumber3;
  float targetPos4;                     long stepsNumber4;
  angle1 = atan2(Y[i], X[i]);
  Serial.print("angle1=");
  Serial.println(angle1 * 180 / PI);
  float x1 = X[i] / cos(angle1);
  int z1 = Z[i] + d - a;
  float n = (square(x1) + square(z1) - square(b) - square(c)) / (2.0 * b * c);
  float angle23 = acos(n);
  Serial.print("angle2+angle3=");
  Serial.println(angle23);
  float m = (float)b / c + n;
  float l = sin(angle23);
  float k = (float)z1 / c;
  angle2 = 360 / PI * atan((-m + sqrt(square(m) + square(l) - square(k))) / (l - k));
  Serial.print("angle2=");
  Serial.println(angle2);
  angle3 = angle23 * 180 / PI - angle2;
  Serial.print("angle3=");
  Serial.println(angle3);
  targetPos4 = 90 + angle3;
  Serial.print("targetPos4=");
  Serial.println(targetPos4);
```

Рис. 3.6. Решение обратной задачи кинематики

После решения обратной задачи кинематики следует проверка полученных углов на условие достижимости: для угла поворота второго звена угол должен составлять от -47 до 47 градусов, для третьей – от -22 до 110 градусов, для четвертой от 0 до 190 градусов. Далее вычисляется угол и направление поворота относительно текущей позиции для достижения требуемой позиции. Также вычисляется количество шагов для каждого шагового двигателя, выбирается максимальное число шагов. Затем в цикле попеременно на каждый драйвер двигателя подаются импульсы. Таким образом, обеспечивается плавность движения. При достижении целевой позиции переменным, отвечающим за текущую позицию, присваивается целевое значение. Также происходит развод губок схвата на нужный угол. Конец программы представлен на рис. 3.7.

```
    if (angle2 >= -47 && angle2 <= 47 && angle3 >= -22 &&
        angle3 <= 110 && targetPos4 >= 0 && targetPos4 <= 190) {
      if (X[i] < 0 && Y[i] >= 0)targetPos1 = 450 - 180 * angle1 / PI;
      else targetPos1 = 90 - 180 * angle1 / PI;
      Serial.print("targetPos1=");
      Serial.println(targetPos1);
      stepsNumber1 = (long)abs(targetPos1 - currentPos1) * i1 * stepsPerRev1 / 360;
      Serial.print("stepsNumber1=");
      Serial.println(stepsNumber1);
      if (targetPos1 - currentPos1 > 0)
        dir1 = HIGH;
      else dir1 = LOW;
      digitalWrite(dirPin1, dir1);
      targetPos2 = angle2;
      stepsNumber2 = (long)abs(targetPos2 - currentPos2) * i2 * stepsPerRev2 / 360;
      Serial.print("stepsNumber2=");
      Serial.println(stepsNumber2);
      if (targetPos2 - currentPos2 < 0)
        dir2 = HIGH;
      else dir2 = LOW;
      digitalWrite(dirPin2, dir2);
      targetPos3 = angle23 * 180 / PI - 25;
      Serial.print("targetPos3=");
      Serial.println(targetPos3);
      stepsNumber3 = (long)abs(targetPos3 - currentPos3) * i3 * stepsPerRev3 / 360;
      Serial.print("stepsNumber3=");
      Serial.println(stepsNumber3);
        if (targetPos3 - currentPos3 > 0)
          dir3 = HIGH;
        else dir3 = LOW;
        digitalWrite(dirPin3, dir3);
        stepsNumber4 = (long)abs(targetPos4 - currentPos4) * i4 * stepsPerRev4 / 360;
        Serial.print("stepsNumber4=");
        Serial.println(stepsNumber4);
        if (targetPos4 - currentPos4 > 0)
          dir4 = HIGH;
        else dir4 = LOW;
        digitalWrite(dirPin4, dir4);
        int maxSteps = max(max(stepsNumber1, stepsNumber2), max(stepsNumber3, stepsNumber4));
        Serial.print("maxSteps=");
        Serial.println(maxSteps);
        for (int j = 0; j < maxSteps; j++) {
          if (j < stepsNumber1) digitalWrite(stepPin1, HIGH);
          if (j < stepsNumber2) digitalWrite(stepPin2, HIGH);
          if (j < stepsNumber3) digitalWrite(stepPin3, HIGH);
          if (j < stepsNumber4) digitalWrite(stepPin4, HIGH);
          delayMicroseconds(delay1);
          if (j < stepsNumber1) digitalWrite(stepPin1, LOW);
          if (j < stepsNumber2) digitalWrite(stepPin2, LOW);
          if (j < stepsNumber3) digitalWrite(stepPin3, LOW);
          if (j < stepsNumber4) digitalWrite(stepPin4, LOW);
          delayMicroseconds(delay1);
        }
        currentPos1 = targetPos1;
        currentPos2 = targetPos2;
        currentPos3 = targetPos3;
        currentPos4 = targetPos4;
        gripperServo.write(G[i]);
        delay(1000);
      }
      else Serial.println("Position is unreachable!");
    }
  }
  arePointsReached = true;
}
```

Рис. 3.7. Конец программы

3.4. Определение координат объекта

Для определения координат объекта для схвата использовалось компьютерное зрение. Для реализации компьютерного зрения был выбран язык программирования Python 3. Приложение писалось в Jupyter Notebook. Jupyter Notebook - это веб-приложение с открытым исходным кодом, которое позволяет создавать и обмениваться документами, которые содержат живой код, уравнения, визуализации и текст повествования [13]. Использование включает в себя: очистку и преобразование данных, численное моделирование, статистическое моделирование, визуализацию данных, машинное обучение и многое другое. В процессе написания использовалась библиотека компьютерного зрения и машинного обучения с открытым исходным кодом OpenCV (называется cv2 для Python [11]). Также использовались библиотека numpy для работы с многомерными массивами, math для использования некоторых математических функций, serial для работы с последовательным портом, time для выставления задержки по времени, PIL для работы с изображениями. Процесс импорта необходимых библиотек и написания функции для вычисления расстояния между 2 точками представлен на рис. 3.8.

```
In [2]: import numpy as np
        import cv2
        import math
        import serial
        import time
        from PIL import Image
        import PIL.ImageOps
        from matplotlib import pyplot as plt
        def euclid(x1,y1,x2,y2):
            return math.sqrt((x1-x2)**2+(y1-y2)**2)
```

Рис. 3.8. Импорт библиотек

Далее делается фотография с веб-камеры, установленной сверху стола. Создается объект camera при помощи функции cv2.VideoCapture(1), затем делается снимок и сохраняется в файл. В конце работы веб-камеры происходит ее удаление. Далее следует обрезка снимка под габариты стола, изменение

размера фотографии на 450x450 пикселей. Таким образом, т.к. диаметр стола составляет 900 мм, то одному пикселю соответсвует 2 мм. Далее на фотографию накладываются прямоугольники и линии, схожие по цвету со столом. Это осуществляется с целью скрыть манипулятор (манипулятор при фотографировании всегда находится в одной позиции, поэтому это можно реализовать простым накладыванием прямоугольников) и некоторые элементы стола, которые могут быть идентифицированы как объекты для схвата. Процесс фотографирования и преобразования снимка показан на рисунке 3.9. Начальное и преобразованное изображение можно наблюдать на рисунке 3.10.

```
In [3]: camera = cv2.VideoCapture(1)
        return_value, camImage = camera.read()
        cv2.imwrite('camImage.png', camImage)
        del(camera)
```

```
In [13]: initImage = Image.open('camImage.png')
         plt.imshow(initImage)
         plt.show()
         left = 164
         top = 10
         right = 634
         bottom = 480
         # (It will not change orginal image)
         cropImage = initImage.crop((left, top, right, bottom))
         basewidth = 450
         hsize=450
         resizedImage = cropImage.resize((basewidth,hsize), Image.ANTIALIAS)
         image=cv2.cvtColor(np.array(resizedImage), cv2.COLOR_RGB2BGR)
         cv2.rectangle(image, (145, 80), (250, 160), (98,162,194), -1)
         cv2.rectangle(image, (180, 160), (262, 185), (98,162,194), -1)
         cv2.rectangle(image, (160, 185), (280, 262), (98,162,194), -1)
         cv2.rectangle(image, (177, 262), (245, 345), (98,162,194), -1)
         cv2.rectangle(image, (370, 175), (390,210), (98,162,194), -1)
         cv2.rectangle(image, (210, 0), (225,15), (98,162,194), -1)
         cv2.rectangle(image, (435, 220), (450,230), (98,162,194), -1)
         cv2.line(image, (280, 255), (366,310), (98,162,194), 3)
         cv2.line(image, (366,310), (364,318), (98,162,194), 3)
         cv2.line(image, (364,318), (368,321), (98,162,194), 3)
         image=cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
         plt.imshow(image)
         plt.show()
         image=cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
```

Рис. 3.9. Процесс создания и преобразования снимка

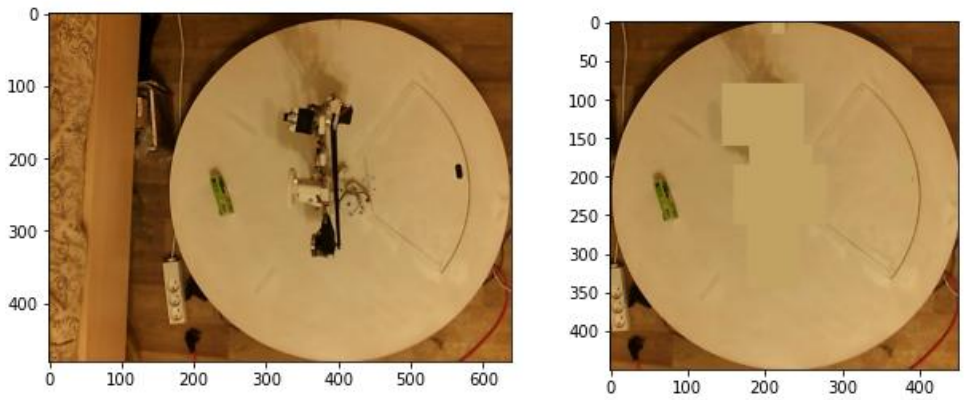

Рис. 3.10. Начальное и преобразованное изображение

Далее у преобразованного изображения вырезается фон. Для этого создается пустое изображение из нулей такой же размерности, как и входное изображение. Затем маркируются вручную единицами фоновая часть изображения. Также маркируется область интереса числами, отличными от 1. После установки маркеров, используется алгоритм водоразделов для создания промаркированного изображения [5]. Области с маркером 1 присваивается черный цвет, областям с маркерами, отличными от 1, присваивается белый цвет. Применяется созданная маска к исходному изображению.

```
In [2]: marker = np.zeros_like(image[:,:,0]).astype(np.int32)
        marker[25][25] = 1
        marker[25][425] = 1
        marker[425][25] = 1
        marker[425][425] = 1
        marker[430][430] = 1
        marker[390][390] = 1
        marker[20][225] = 20
        marker[430][225] = 20
        marker[225][20] = 20
        marker[225][430] = 20
        marked = cv2.watershed(image, marker)
        plt.imshow(marked, cmap='gray')
        plt.show()
        marked[marked == 1] = 0
        marked[marked > 1] = 255
        kernel = np.ones((3,3),np.uint8)
        dilation = cv2.dilate(marked.astype(np.float32), kernel, iterations = 1)
        plt.imshow(dilation, cmap='gray')
        plt.show()
        finalImage = cv2.bitwise_and(image, image, mask=dilation.astype(np.uint8))
        finalImage =cv2.cvtColor(finalImage, cv2.COLOR_BGR2RGB)
        plt.imshow(finalImage)
        plt.show()
```

Рис. 3.11. Вырезка фона

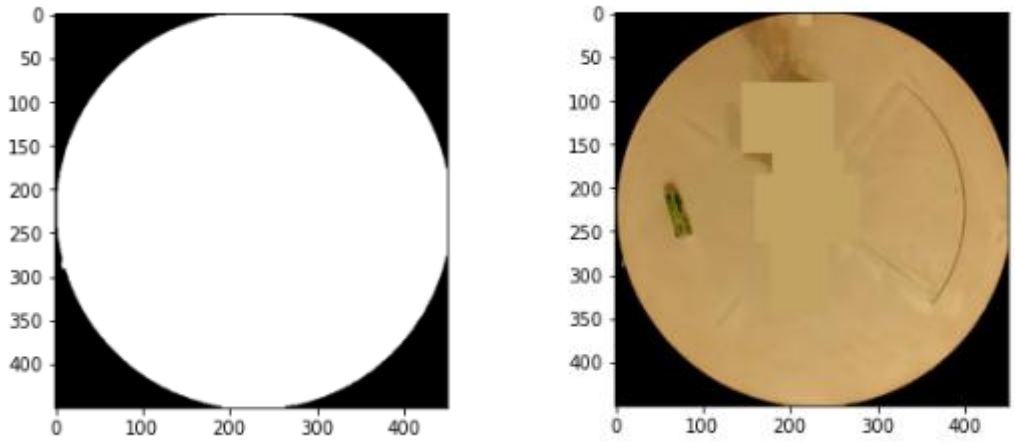

Рис. 3.12. Изображение с вырезанным фоном

На рисунке 3.13 применены различные виды порогов к исходному изображению. Был выбран BINARY_INV (рис. 3.14) для того, чтобы инвертировать цвета. Пикселям черно-белого изображения со значениями от 0 до 90 (значение порога, которое можно регулировать) будет присвоен белый цвет, от 90 до 255 – черный.

```
In [8]: #Showing different types of thresholds
        grayImg =cv2.cvtColor(finalImage, cv2.COLOR_RGB2GRAY)
        ret,thresh1 = cv2.threshold(grayImg,127,255,cv2.THRESH_BINARY)
        ret,thresh2 = cv2.threshold(grayImg,127,255,cv2.THRESH_BINARY_INV)
        ret,thresh3 = cv2.threshold(grayImg,127,255,cv2.THRESH_TRUNC)
        ret,thresh4 = cv2.threshold(grayImg,127,255,cv2.THRESH_TOZERO)
        ret,thresh5 = cv2.threshold(grayImg,127,255,cv2.THRESH_TOZERO_INV)

        titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
        images = [grayImg, thresh1, thresh2, thresh3, thresh4, thresh5]

        for i in range(6):
            plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
            plt.title(titles[i])
            plt.xticks([]),plt.yticks([])
        plt.show()
```

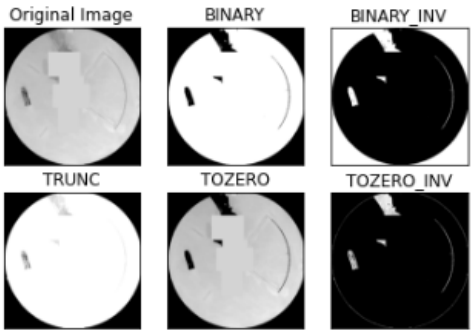

Рис. 3.13. Разные варианты порогов

```
In [7]: # Convert to grayscale and threshold
        imGray = cv2.cvtColor(finalImage,cv2.COLOR_RGB2GRAY)
        ret,thresh = cv2.threshold(imGray,90,255,cv2.THRESH_BINARY_INV)
        plt.imshow(thresh,'gray')
        plt.show()
```

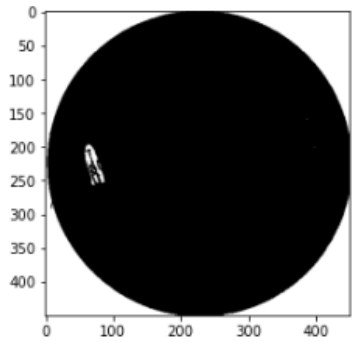

Рис. 3.14. Порог BINARY_INV

Далее происходит поиск контуров при помощи функции cv2.findContours(…) и их отрисовка зеленым цветом при помощи функции cv2.drawContours(…) (рис. 3.15). Далее найденные контуры фильтруются по радиусу (должен быть от 5 до 25 пикселей или от 10 до 50 мм). Также контур должнен находиться внутри стола, поэтому расстояние от центра контура до центра стола должно быть меньше радиуса стола 450 мм. Также фильтруются контуры, находящиеся на маленьком расстоянии друг от друга (меньше 50 мм), т.к. вероятнее всего эти контуры являются одним объектом. Отфильтрованные контуры записываются в результирующий список res. Преобразованные координаты (x,y) центров контуров передаются по последовательному порту (USB) на Arduino (рис. 3.16).

```
In [300]: ## Find contours, draw on image and save
contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
resizedImage = cv2.cvtColor(np.array(resizedImage),cv2.COLOR_RGB2BGR)
cv2.drawContours(resizedImage, contours, -1, (0,255,0), 3)
cv2.imwrite('result.png',resizedImage)
resizedImage = cv2.cvtColor(resizedImage,cv2.COLOR_BGR2RGB)
plt.imshow(resizedImage)
plt.show()
```

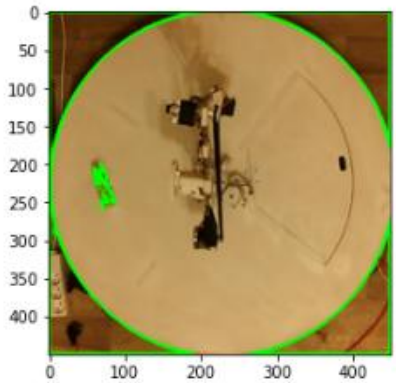

Рис. 3.15. Поиск и отрисовка контуров

```
In [301]: #Print all contours and filtering
res=[]
for cnt in contours:
   (x,y),radius = cv2.minEnclosingCircle(cnt)
   center = (int(x),int(y))
   radius = int(radius)
   print('Contour: centre {},{}, radius {}'.format(round(x),round(y),radius))
   if radius>5 and radius<25 and euclid(x,y,225,225)<220:
        flag=False
        if len(res)==0:
            res.append(cnt)
        for cnt1 in res:
           (x1,y1),radius1 = cv2.minEnclosingCircle(cnt1)
           if euclid(x,y,x1,y1)<25:
             flag=True
        if flag==False:
            res.append(cnt)
```

```
Contour: centre 356,363, radius 126
Contour: centre 449,274, radius 0
Contour: centre 78,251, radius 0
Contour: centre 70,252, radius 4
Contour: centre 86,248, radius 4
```

```
In [303]: ser = serial.Serial('COM3', 9600)
time.sleep(3)
ser.write(bytes(str(len(res)), 'utf-8'))
ser.write(b';')
for cnt in res:
   (x,y),radius = cv2.minEnclosingCircle(cnt)
   center = (int(x),int(y))
   radius = int(radius)
   ser.write(bytes(str(2*round(x-225)),'utf-8'))
   ser.write(b';')
   ser.write(bytes(str(2*round(225-y)),'utf-8'))
```

Рис. 3.16. Фильтрация контуров и передача преобразованных координат центров контуров по последовательному порту

ЗАКЛЮЧЕНИЕ

Разработка материалов для выпускной квалификационной работы магистра на тему «разработка манипулятора с мягким схватом» выполнена в соответствии с требованиями задания.

Проведен аналитический обзор различных видов манипуляторов с мягкими схватами. На основе данного обзора было решено проектировать мягкий схват в качестве дополнительных съемных пластин из легко деформируемого материала (например, резины), т.к. данный тип схвата прост в изготовлении и не требует подключение пневматики или гидравлики. Более того, данный тип схвата можно обратно превратить в классический жесткий схват, т.к. пластины выполнены в съемном исполнении.

Создана кинематическая, структурная схемы манипулятора, а также разработана его 3D-модель. Также выполнены анимация движения, собран действующий прототип. Проведены испытания, показывающие работоспособность конструкции манипулятора и мягкого схвата. Характеристики манипулятора с мягким схватом соответствую требованиям задания, а именно

Из недостатков разработанного прототипа манипулятора с мягким схватом можно отметить недостаточную жесткость конструкции и недостаточную точность позиционирования, а также долгую калибровку камеры.

В дальнейшем для исправления имеющихся недостатков и улучшения теущих планируется сделать конструкцию манипулятора более жесткой. Этого можно достичь, используя другие виды пластика, а также усовершенствовав саму конструкцию (например, добавлением ребер жесткости). Также требуется облегчить конструкцию, что увеличит общую динамику, т.е. позволит двигаться с большими скоростями и ускорениями. Шаговые двигатели целесообразно заменить сервомоторами, что увеличит общую точность позиционирования и позволит избавиться от концевых датчиков и драйверов

шаговых двигателей. Планируется разработать другие прототипы мягкого схвата, используя технологию 3D-печати (применяя FLEX-пластик или другой аналогичный пластик). Для уменьшения времени калибровки камеры планируется выполнить кронштейн камеры, крепящимся непосредственно к столу манипулятора, что увеличит точность позиционирования камеры относительно стола. Рассматривается возможность доработки программы распознавания координат объекта по фотографии для работы с видео, а также для распознавания 3-х координат объекта (x,y,z) вместо 2-х (x,y) (для этого потребуется установить камеру под углом к столу).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Анурьев В.И. Справочник конструктора-машиностроителя: В 3 т. Т. 1. – 8-е изд., перераб. и доп. Под ред. И.Н. Жестковой. – М.: Машиностроение, 2001. – 920 с.

2. Анухин В.И. Допуски и посадки. Выбор и расчет, указание на чертежах: Учеб. пособие. 2-е изд., перераб. и доп. СПб.: Изд-во СПбГТУ, 2001. – 220 с.

3. Мацкевич В.В. Занимательная анатомия роботов. Изд.: Радио и связь, 1998. – 160 с.

4. Москвичев А.А., Кварталов А.Р., Устинов Б.В. Захватные устройства промышленных роботов и манипуляторов: Учебное пособие. Изд.: Инфра-М, 2015. – 180 с.

5. Alexander Mordvintsev & Abid K. Image Segmentation with Watershed Algorithm. 2013. [Электронный ресурс] – https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html

6. Arduino.cc. Статья: Функция parseInt(). 2016. [Электронный ресурс] – https://www.arduino.cc/reference/en/language/functions/communication/serial/parseint

7. Arduino.ru. Статья: Функция digitalRead(). 2016. [Электронный ресурс] – http://arduino.ru/Reference/DigitalRead

8. Arduino.ru. Статья: Функция pinMode(). 2016. [Электронный ресурс] – http://arduino.ru/Reference/PinMode

9. Arduino.ru. Статья: Serial.available(). 2016. [Электронный ресурс] – http://arduino.ru/Reference/Serial/Available

10. FESTO. Статья: TentacleGripper. 2017. [Электронный ресурс] – https://www.festo.com/group/en/cms/12745.htm

11. Ghoneim S. Статья: OpenCV-Python Cheat Sheet: From Importing Images to Face Detection. 2019. [Электронный ресурс] –

https://heartbeat.fritz.ai/opencv-python-cheat-sheet-from-importing-images-to-face-detection-52919da36433

12. OnRobot. SG BASE PART AND SG SILICONE TOOLS. 2018. [Электронный ресурс] – https://onrobot.com/sites/default/files/documents/Datasheet_SG_v1.0_EN.pdf

13. Pryke B. Статья: Jupyter Notebook for Beginners: A Tutorial. 2019. [Электронный ресурс] – https://www.dataquest.io/blog/jupyter-notebook-tutorial/

14. Robo-hunter. Статья: Деликатный захват предметов. 2019. [Электронный ресурс] – https://robo-hunter.com/news/delikatnii-zahvat-predmetov-v-soft-robotics-sozdayt-pnevmaticheskie-manipulyatori-dlya-promishlennih-robotov13666

15. Luke Spencer. Статья: PLA Vs. ABS Plastic Filament – Unlock The Pros And Cons. 2019. [Электронный ресурс] – https://3dknowledge.com/pla-vs-abs/