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

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.03.01 – Информатика и вычислительная техника
Профиль: Автоматизированные системы обработки информации и управления

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____ 2016 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка программируемого виртуального макета робота «InMoov»

Исполнитель
студент группы 253-об _____ А.Р. Сычёв
(подпись, дата)

Руководитель
профессор, доктор техн. наук _____ И.Е. Еремин
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.В. Романико
(подпись, дата)

Благовещенск 2016

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

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____ 2016 г.

З А Д А Н И Е

К бакалаврской работе студента Сычёва Алексея Рустамовича

1. Тема бакалаврской работы: Разработка программируемого виртуального макета робота «InMoov»

(утверждено приказом от 19.04.2006г. № 503-УЧ)

2. Срок сдачи студентом законченной работы 29.05.2006 г.

3. Исходные данные к бакалаврской работе: отчет по преддипломной практике, ГОСТы, справочная литература, ресурсы Интернет.

4. Содержание дипломной работы:

общая характеристика открытого проекта InMoov (комплектация корпуса робота InMoov, общий принцип работы электрооборудования, характеристика аппаратной вычислительной платформы);

характеристика функциональных подсистем (подсистема визуализации робота InMoov, подсистема физического управления виртуальным макетом, подсистема конвертации управляющих сигналов в код Arduino);

характеристика обеспечивающих подсистем (информационное обеспечение, программное обеспечение, техническое обеспечение);

разработка виртуального макета робота InMoov (сборка 3D модели робота InMoov, разработка скелетной анимации подвижных частей робота, разработка

графического интерфейса пользователя, Реализация конвертора для связи виртуальной и физической моделей).

5. Перечень материалов приложения: листинг программного обеспечения, листинг программы, реализующей управление виртуальным макетом при помощи контроллера, листинг программы для конвертации управляющих сигналов в код Arduino.

6. Консультанты по дипломной работе (с указанием относящихся к ним разделов):

нормоконтроль – В.В. Романико, инженер кафедры.

7. Дата выдачи задания 02.02.2006 г.

Руководитель бакалаврской работы: Еремин Илья Евгеньевич, профессор, доктор техн. наук _____

Задание принял к исполнению: 02.02.2006 г.

_____ А.Р. Сычёв

Благовещенск 2016

РЕФЕРАТ

Бакалаврская работа содержит 62 с., 45 рисунков, 15 источников, 1 приложение.

BLENDER, INMOOV, 3D-МОДЕЛЬ, ВИРТУАЛЬНЫЙ МАКЕТ, ФИЗИЧЕСКАЯ МОДЕЛЬ, MICROSOFT VISUAL STUDIO, UNITY 3D

Объектом исследования данной бакалаврской работы является трехмерная модель робота InMoov.

Целью данной бакалаврской работы является разработка виртуального программируемого макета робота InMoov. Ход работы разделен на четыре этапа.

На первом этапе производится исследование предметной области, формулирование основных понятий, общее описание робота InMoov и его принципа работы.

На следующем этапе дана характеристика функциональных подсистем необходимых для реализации поставленной цели.

Третьим этапом выделяются необходимые обеспечивающие подсистемы, дается их характеристика.

На четвертом этапе производится непосредственно разработка виртуального макета робота InMoov, которую можно разделить на следующие пункты:

1) сборка 3D модели робота InMoov;
2) разработка скелетной анимации подвижных частей робота;
3) разработка подсистемы для управления контроллером;
4) разработка графического интерфейса пользователя;
5) реализация конвертора для связи виртуальной и физической моделей.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЭВМ – Электронная вычислительная машина;

ПО – Программное обеспечение;

ПК – Персональный компьютер;

ИС – Информационная система;

УВМ – Управляющая цифровая вычислительная машина.

ВВЕДЕНИЕ

На протяжении всего своего развития человечество старалось изготовить предметы, механизмы, орудия, машины упрощающие труд и обеспечивающие защиту от какой-либо угрозы. Эволюция современного общества и производства дала толчок к возникновению и развитию нового класса машин – роботов – и соответствующего научного направления – робототехники.

Робототехника на сегодняшний день является неотъемлемой частью технического прогресса, усиленно развивающейся научно-технической дисциплиной, изучающей как теорию, методы расчета и конструирования роботов, их систем и элементов, так и проблемы комплексной автоматизации производства, и научных исследований с использованием роботов. Предметом робототехники является создание и применение роботов, других средств робототехники и основанных на них технических систем и комплексов различного назначения.

С каждым годом системы становятся более совершенными и эффективными. Но также этому сопутствуют такие факторы как: усложнение системы, повышение цены комплектующих, трудность в изготовлении. В связи с этим целесообразнее всего первоначально смоделировать систему на ЭВМ и протестировать до изготовления, сборки, наладки и использования реального прототипа, чтобы избежать дальнейших ошибок, которые могли бы привести к выводу из строя дорогостоящих комплектующих. Решить данную задачу призваны трехмерное моделирование и программирование.

Трехмерная визуализация объекта позволяет наглядно продемонстрировать все стороны разрабатываемой модели без затрат на производство реального прототипа. Также модификация трехмерной модели намного проще и дешевле. Даже небольшие изменения конструкции реального объекта могут привести к затратам на производство новых деталей. Еще большую наглядность позволяет достичь возможность анимации подвижных частей спроектированной модели.

Трехмерную модель можно экспортировать в форматы, поддерживаемые различными пакетами для разработки графических приложений. В данном слу-

чае благодаря программированию можно задать определенный алгоритм работы разрабатываемой системы, протестировать ее в различных ситуациях для выявления ошибок. Также можно создать систему управления моделью с помощью периферийных устройств и систему для передачи управляющего сигнала на реальный прототип. Таким образом, возможно смоделировать и протестировать практически любую составляющую реального объекта.

Совокупность смоделированных составляющих реального объекта, процессов его функционирования является основой для создания виртуального макета, представляющего собой объект, в котором рассматриваются его определенный характеристики и свойства, необходимые для его дальнейшего изготовления.

Целью данной бакалаврской работы является разработка виртуального макета робота InMoov.

Для достижения поставленной цели необходимо решить следующие задачи:

1) собрать 3D модель робота;
2) разработать скелетную анимацию подвижных частей робота;
3) разработать интерфейс для управления контроллером;
4) спроектировать и разработать графический интерфейс пользователя;
5) реализовать конвертор для связи виртуальной и физической моделей

На данный момент существует лишь один проект направленный на виртуализацию робота InMoov. Он имеет название Virtual InMoov, которое также описывает тематику проекта. Основной задачей данного проекта является виртуализация как самого робота, так и его аппаратной начинки. Разработчиками данного проекта был представлен программный комплекс способный представить всю электронно-цифровую базу робота. В отличие от проекта Virtual InMoov, в данной работе основной упор был сделан на решение задачи управления виртуальным макетом, на возможность его запрограммировать и связать с реальным прототипом.

Работ, направленных на создание виртуального макета работа InMoov представлено очень мало. Выше упомянутый проект Virtual InMoov не применим для

поставленных целей, т.к. в нем не рассматривается интерактивное управление роботом и связь с физической моделью.

Основной задачей работы является минимизация возможных затрат при возникновении каких-либо сбоев и ошибок в работе робота. Разрабатываемый информационный продукт должен дать возможность произвести все необходимые для этого тесты.

Проектируемое программное обеспечение должно послужить виртуальным макетом, нацеленным на визуализацию необходимых процессов, с которыми предстоит встретиться при сборки, наладки и эксплуатации реального прототипа. Также оно должно обеспечить возможность управления роботом по средствам контроллера дистанционно и возможность запрограммировать физическую модель.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ОТКРЫТОГО ПРОЕКТА INMOOV

Для данной дипломной работы в качестве объекта виртуализации был взят общедоступный проект InMoov. Этот проект является моделью робота, части которого изготавливаются на 3D-принтере. Все комплектующие для печати InMoov распространяются по открытой лицензии CC-BY-NC, что делает проект полностью свободным для некоммерческого использования. Разработчиком данного проекта является французский художник Гаэль Лангевин. Область применения изготовленной модели робота различны: возможна эксплуатация на производстве, применение в обучающих целях.

1.1 Комплектация корпуса робота InMoov

Всего комплектующих деталей для печати на 3D принтере насчитывается более 130 штук. Материалом, из которого изготавливаются детали, является пластмасса. Крепление осуществляется за счет болтов, шурупов, саморезов, кареток, нитей и клея. На рисунке 1 изображен уже распечатанный и собранный робот InMoov.

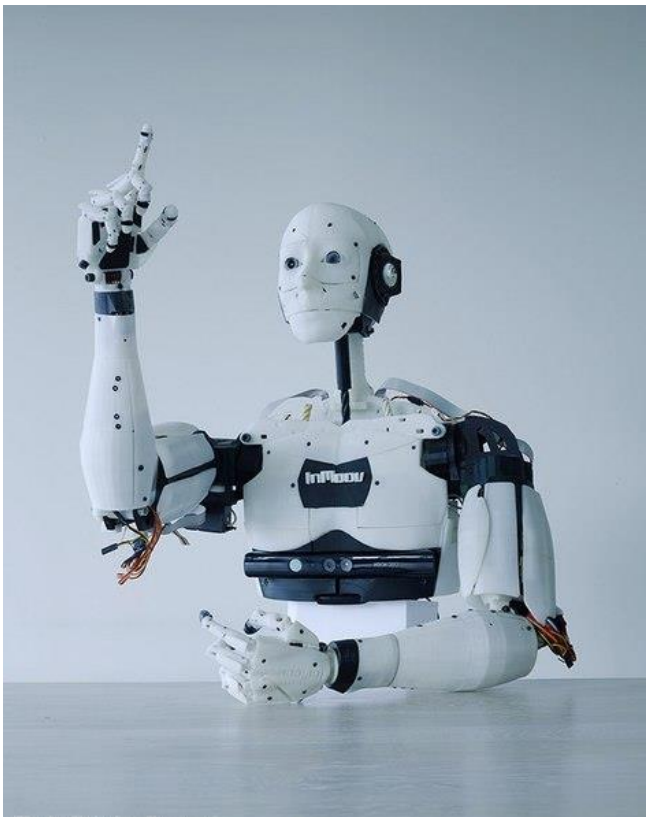

Рисунок 1 – Робот InMoov

Для демонстрации сложности всей конструкции далее будут приведены составные части одной из рук робота.

Каждый палец состоит из трех основных частей: основание, средняя часть и конечная часть. Соединение производится за счет промежуточных частей, «суставов», которых также три штуки. На рисунке 2 изображены перечисленные части одного из пальцев. Под номерами 1, 2, 6 расположены средняя часть, основание и конечная часть соответственно. Под номерами 3, 4, 5 промежуточные части

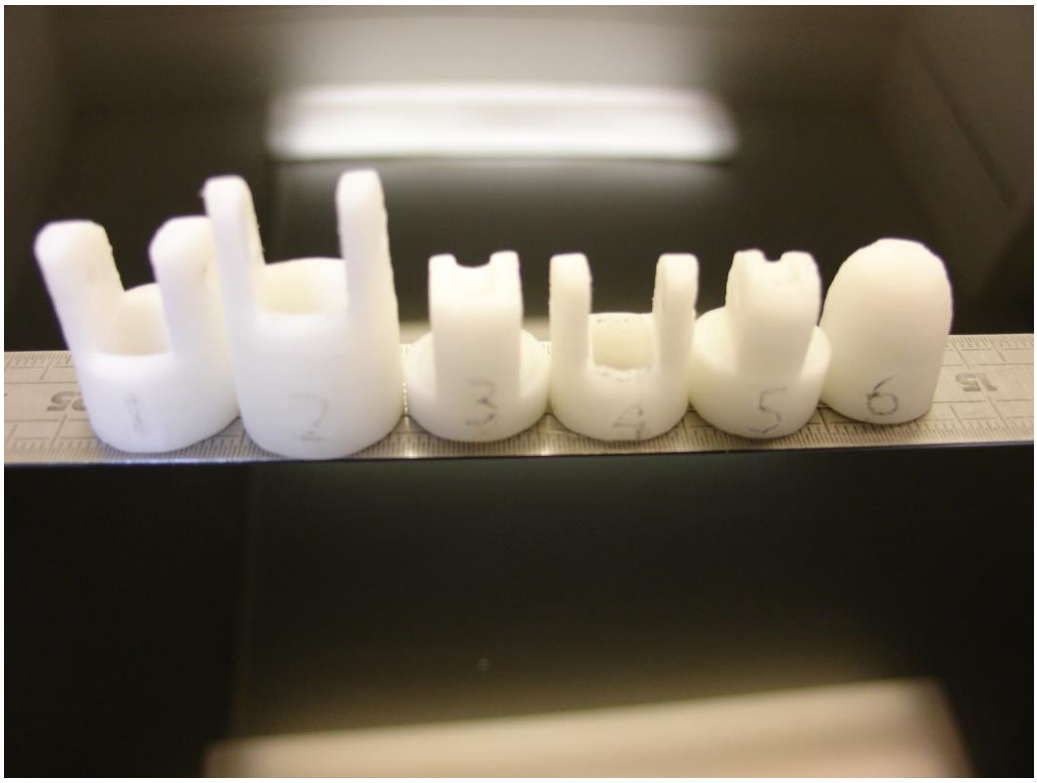

Рисунок 2 – Составные части пальцев

Кисть робота состоит из выше упомянутых пальцев и четырех частей формирующих ладонь: основная часть, служащая ладонью и трех примыкающих суставов для сгибания большого пальца, мизинца и безымянного пальца. На рисунке 3 изображены все четыре части.

На рисунке 3 под номером 1 показана ладонь, под номером 2 показан примыкающий сустав большого пальца, под номером 3 примыкающий сустав безымянного пальца и под номером 4 примыкающий сустав мизинца.

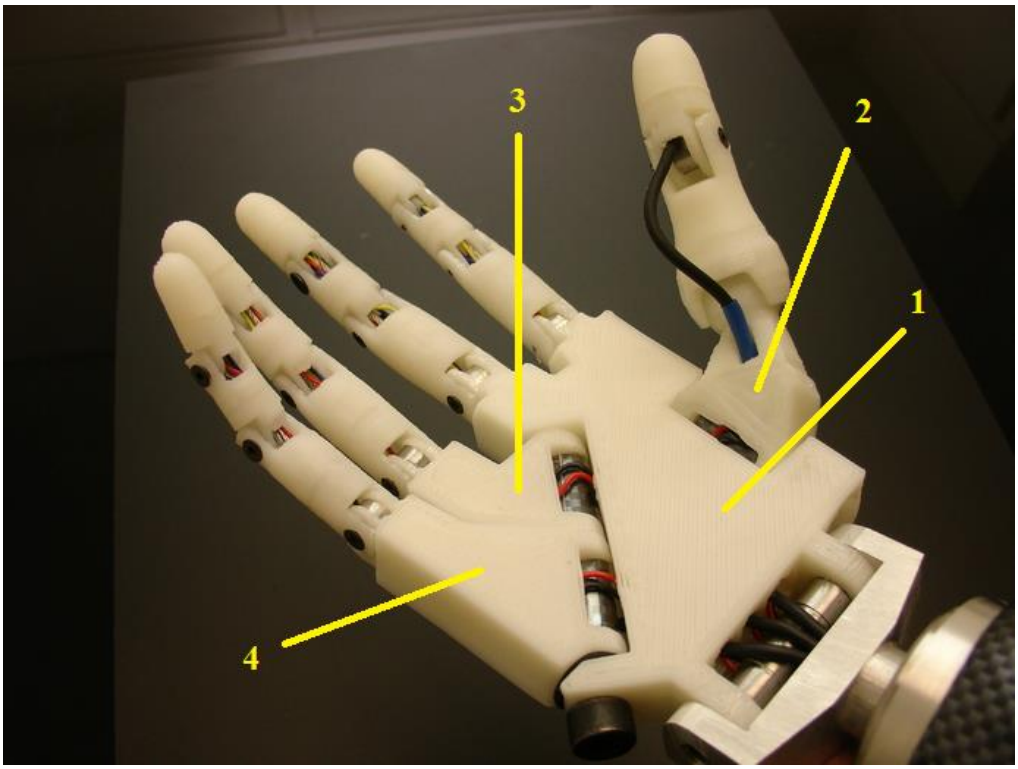

Рисунок 3 – Кисть робота InMoov

На рисунках 4 и 5 изображены нижняя и верхняя часть предплечья. Каждая из них в свою очередь состоит из двух частей. В собранном виде в предплечье образуется полость, в которой должны быть расположены сервоприводы.

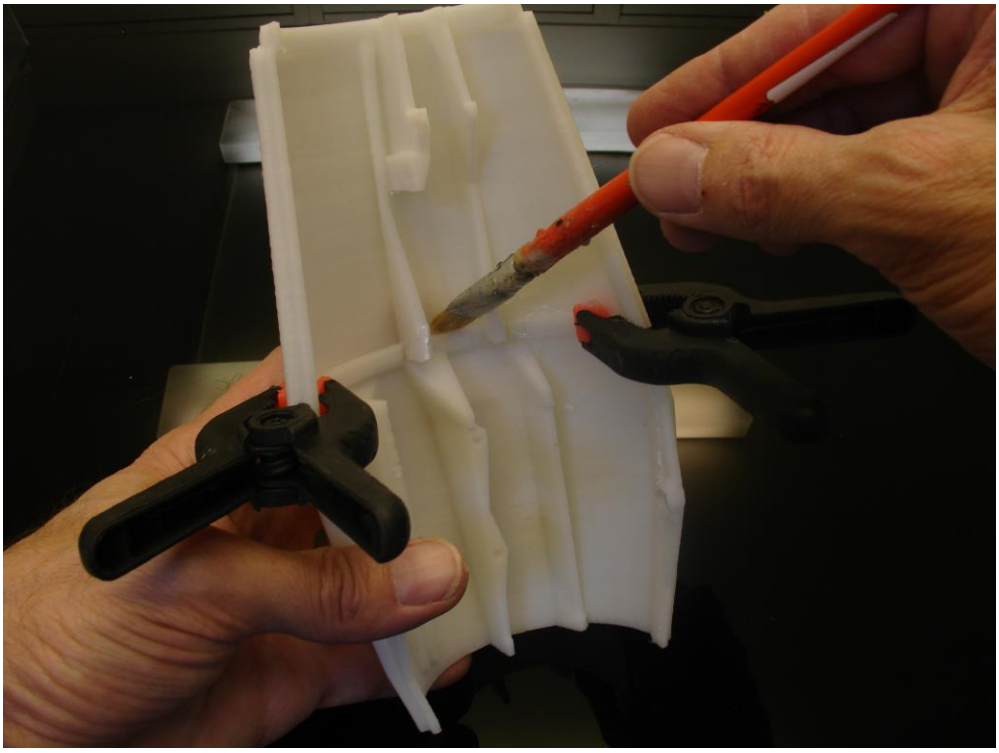

Рисунок 4 – Нижняя часть предплечья

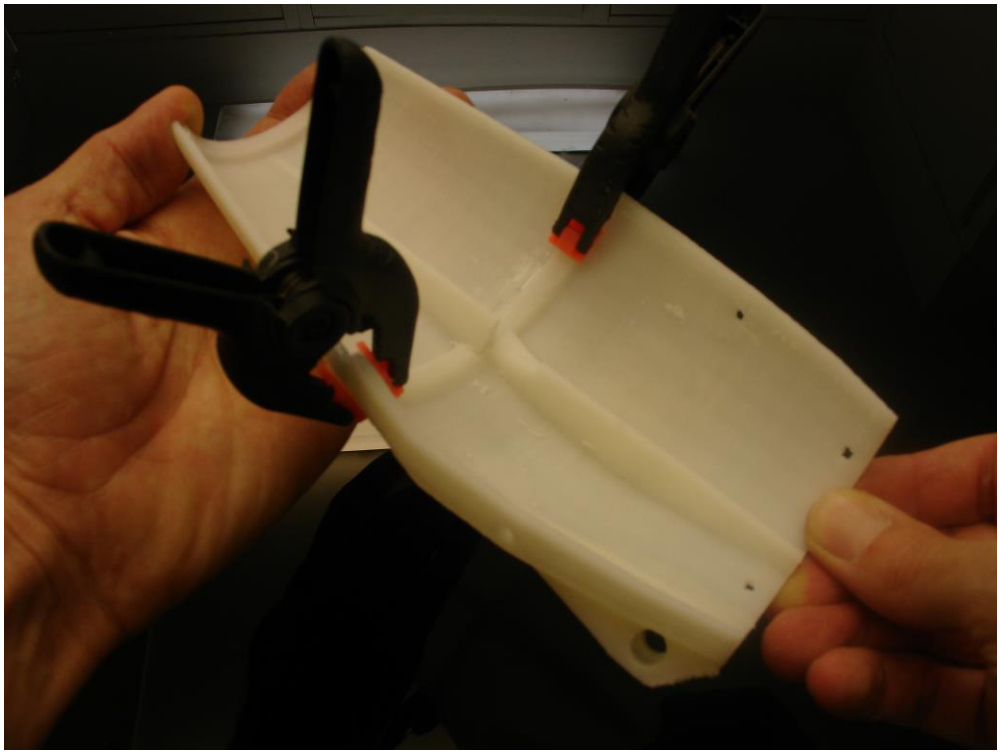

Рисунок 5 – Верхняя часть предплечья

Промежуточной частью между кистью и предплечьем является накладка изображенная на рисунке 6.

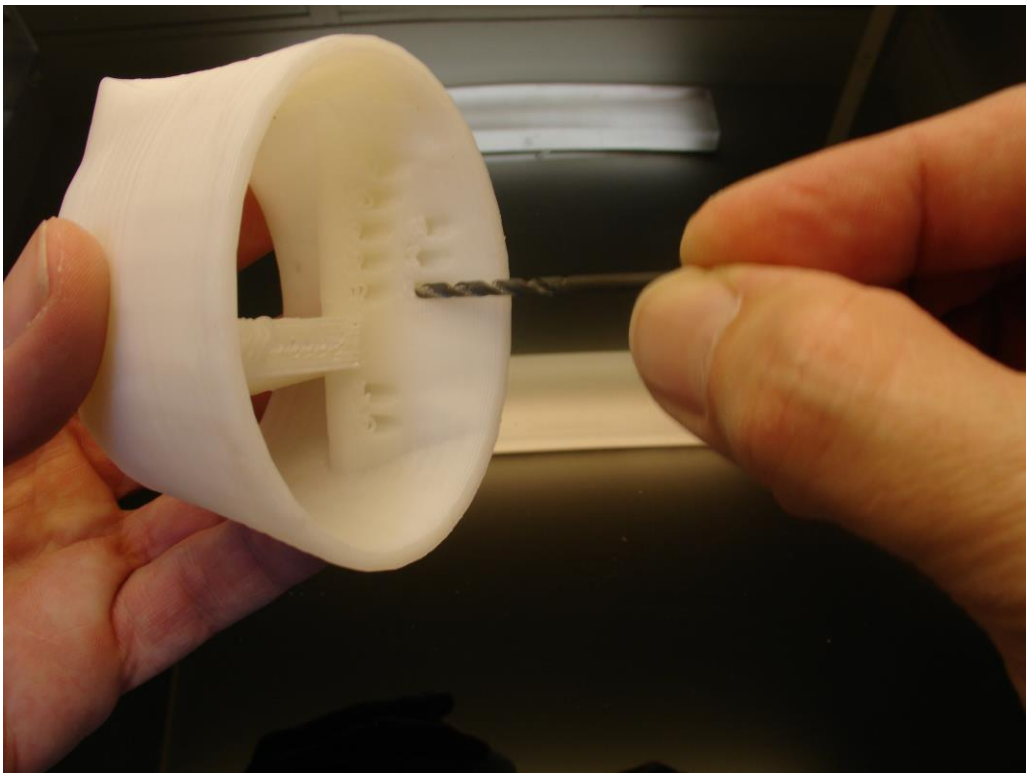

Рисунок 6 – Накладка между кистью и предплечьем

На рисунке 7 изображен локтевой сустав. Сгибание локтя осуществляется за счет вращения винта под номером 1, который вращает шестерню под номером 2. Вращение винта задает сервопривод под номером 3.

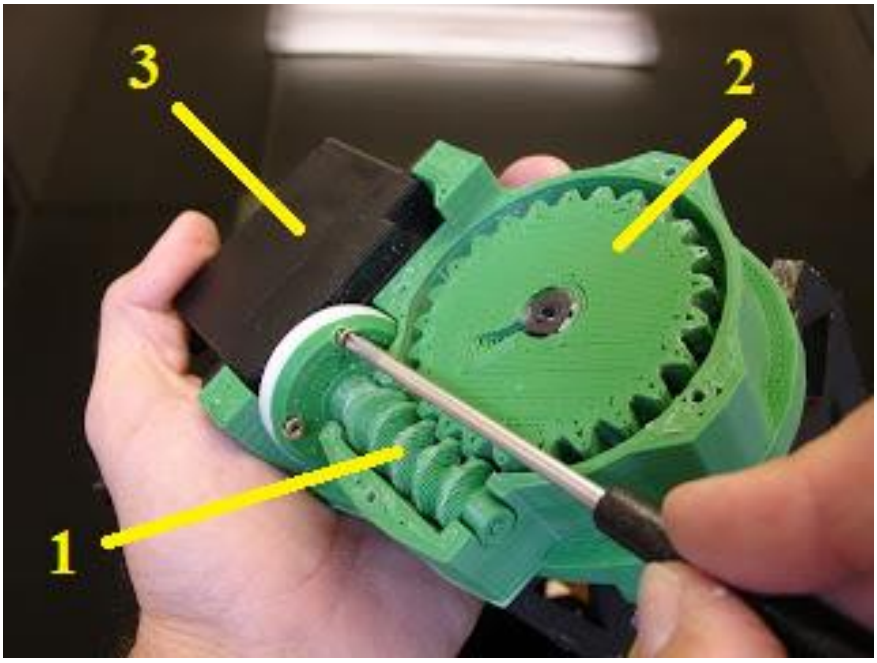

Рисунок 7 – Локтевой сустав

Плечо также, как и предплечье, состоит из верхней и нижней частей и усиливающей конструкции, которые в свою очередь делятся еще на две для удобства печати. На рисунке 8 изображена часть плеча и плечевой сустав уже в собранном виде.

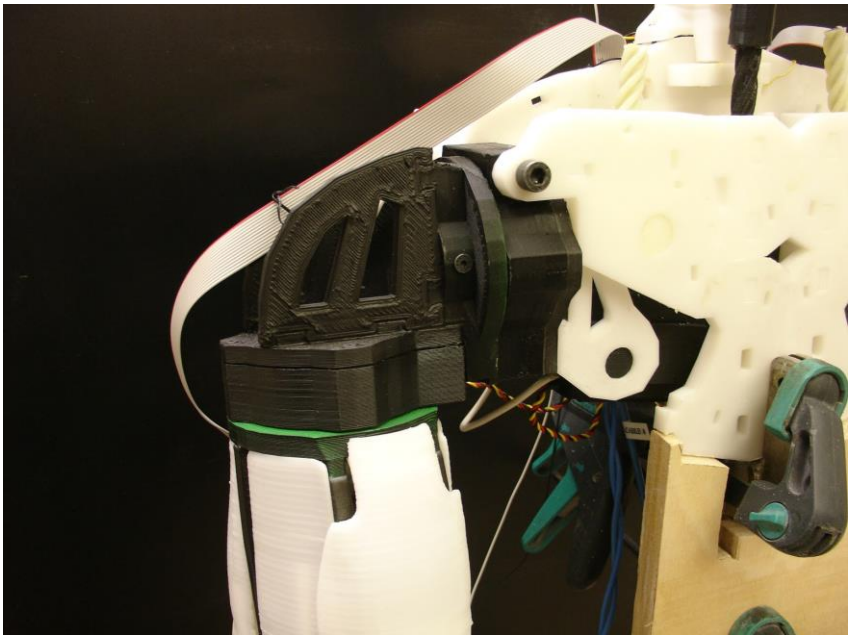

Рисунок 8 – Часть плеча и плечевой сустав

В описанном выше составе деталей руки робота InMoov присутствует сорок пять деталей, которые должные быть распечатаны на 3D-принтере. В общей сумме две руки робота состоят из девяноста деталей, что является основной частью от их общего числа. На рисунках 9 и 10 изображены туловаище и голова робота. Число деталей, из которых они состоят, насчитывает около пятидесяти штук.

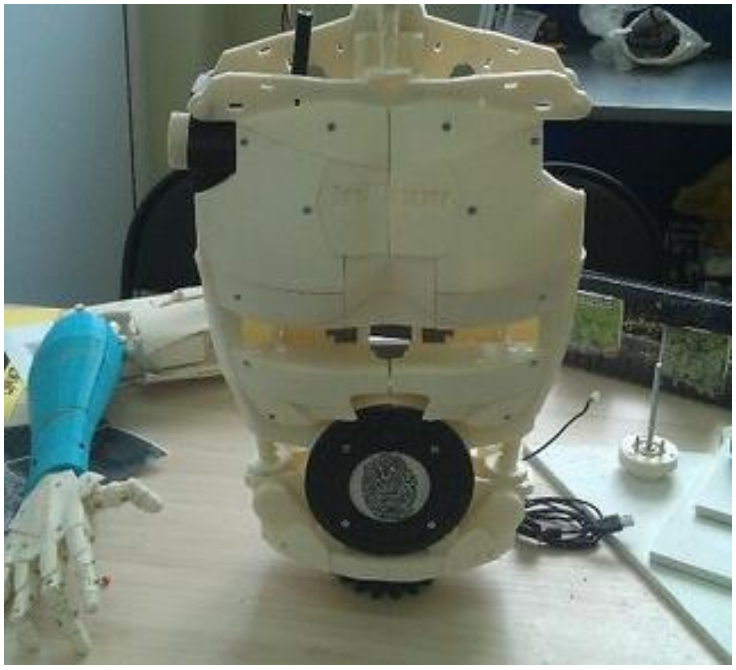

Рисунок 9 – Туловище робота InMoov

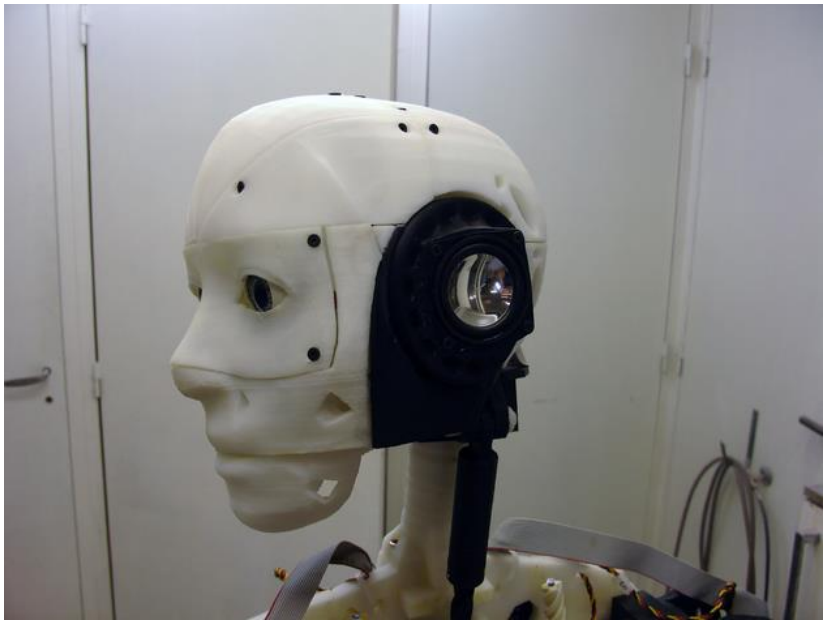

Рисунок 10 – Голова робота InMoov

1.2 Общий принцип работы электрооборудования

За движение подвижных частей робота InMoov отвечают 16 электродвигателей, также называемых сервомоторами или сервоприводами. Каждый из них отвечает за подвижность той или иной части тела робота.

По своей сути сервопривод — это мотор-редуктор, который должен поворачивать выходной вал строго в заданное положение (на угол) и удерживать его там, вопреки сопротивлениям и возмущениям окружающей среды. Нужно это было, в первую очередь, моделистам, для управления положениями различных закрылков, рулей и вертолётных лопастей.

На выходной вал обычно надевают пластиковый рычаг (реже металлический), коромысло, диск или крест, с отверстиями для закрепления тяг рулей высоты, глубины, элеронов или ног робота. На рисунке 11 изображен сервопривод MG995.

Рисунок 11 – Сервопривод MG995

Чтобы четко отслеживать положения вала и воспринимать сигналы управления, сервоприводы имеют в своем составе электронику. Вал с рычагом жёстко связан с двигателем переменного резистора. Резистор подключен в схему контроля и сообщает ей своим текущим сопротивлением о текущем положении вала. На схему, в свою очередь, поступают сигналы управления, сообщающие в какое положение нужно повернуть выходной вал (и резистор соответственно). Схема подаёт питание на моторчик и вращает куда, там останавливается и если что-

нибудь сдвинет рычаг из нужной точки, то вернёт её на место или передвинет её если придёт новая команда.

В простейших аналоговых сервоприводах, угол задаётся длительностью импульсов идущих с определённой частотой. В улучшенных версиях используется протокол I2C, в котором можно узнать текущее положение, текущую нагрузку и скорость движения. Управление сервоприводов довольно простое – у него есть три провода изображенные на рисунке 12.

Рисунок 12 – Контакты подключения сервопривода

Земля – коричневый/черный, питание +5 вольт – красный, сигнальный – оранжевый/желтый/белый.

Управление импульсное – по сигнальному проводу. Особенность состоит в том, что сигнальный провод слаботочный – импульсы можно давать непосредственно с пина микроконтроллера, в отличие от кабеля питания.

Чтобы повернуть сервопривод на нужный угол – нужно на сигнальный вход подавать импульс с нужной длительностью. Чтобы удерживать определённую позицию – импульс должен повторяться. На рисунке 13 показано какой импульс необходим для поворота рычага сервомотора на определенный угол.

Далее рассмотрим сервоприводы, которые непосредственно используются в проекте InMoov.

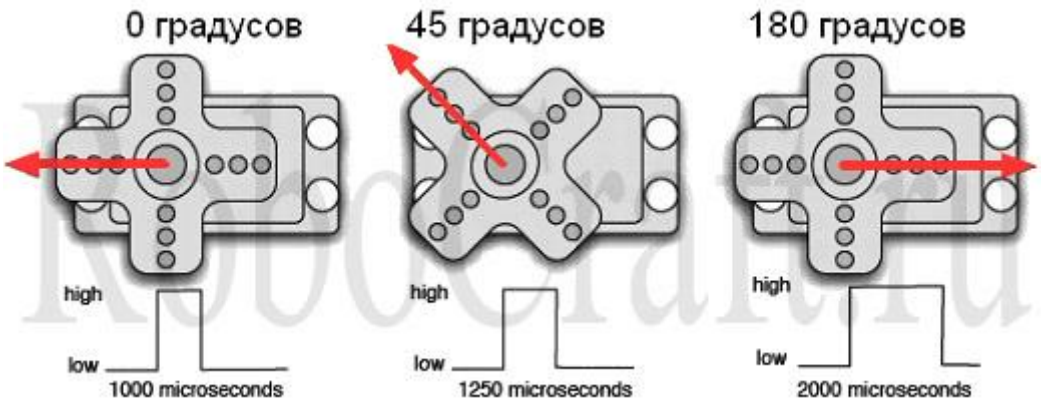

Рисунок 13 – Импульсы для поворота сервомотора

Десять из шестнадцати сервоприводов, используемых в роботе InMoov, предназначены для движения пальцев робота. Для этого используется сервопривод фирмы HobbyKing, модели HK15298, который изображен на рисунке 14.

Рисунок 14 – Сервопривод HK15298

Остальные шесть сервоприводов используются в локтевых и плечевых суставах. В локтевых по одному, а в плечевых по две штуки. Здесь используются сервоприводы фирмы Hitec, модель HS805BB. Главное отличие от предыдущего сервопривода состоит в большей мощности, т.к. он предназначен для движения большей массы руки. Данная модель изображена на рисунке 15.

Рисунок 15 – Сервопривод HS805BB

1.3 Характеристика аппаратной вычислительной платформы

Для автоматизации управления всеми частями робота InMoov в качестве аппаратной вычислительной платформы используется микроконтроллер Arduino.

Микроконтроллер Arduino — аппаратная вычислительная платформа, основными компонентам которой являются простая плата ввода/вывода и среда разработки на языке Wiring (C++).

Плата Arduino состоит из микроконтроллера Atmel AVR (ATmega328 и ATmega168 в новых версиях и ATmega8 в старых) и элементной обвязки для программирования и интеграции с другими схемами. На каждой плате обязательно присутствуют линейный стабилизатор напряжения 5 В и 16 МГц кварцевый генератор (в некоторых версиях керамический резонатор). В микроконтроллер предварительно прошит загрузчик, поэтому внешний программатор не нужен. На рисунке 16 изображена стандартная комплектация платы Arduino.

Arduino – торговая марка аппаратно-программных средств для построения простых систем автоматики и робототехники, ориентированная на непрофессиональных пользователей. Программная часть состоит из бесплатной программной оболочки (IDE) для написания программ, их компиляции и программирования аппаратуры.

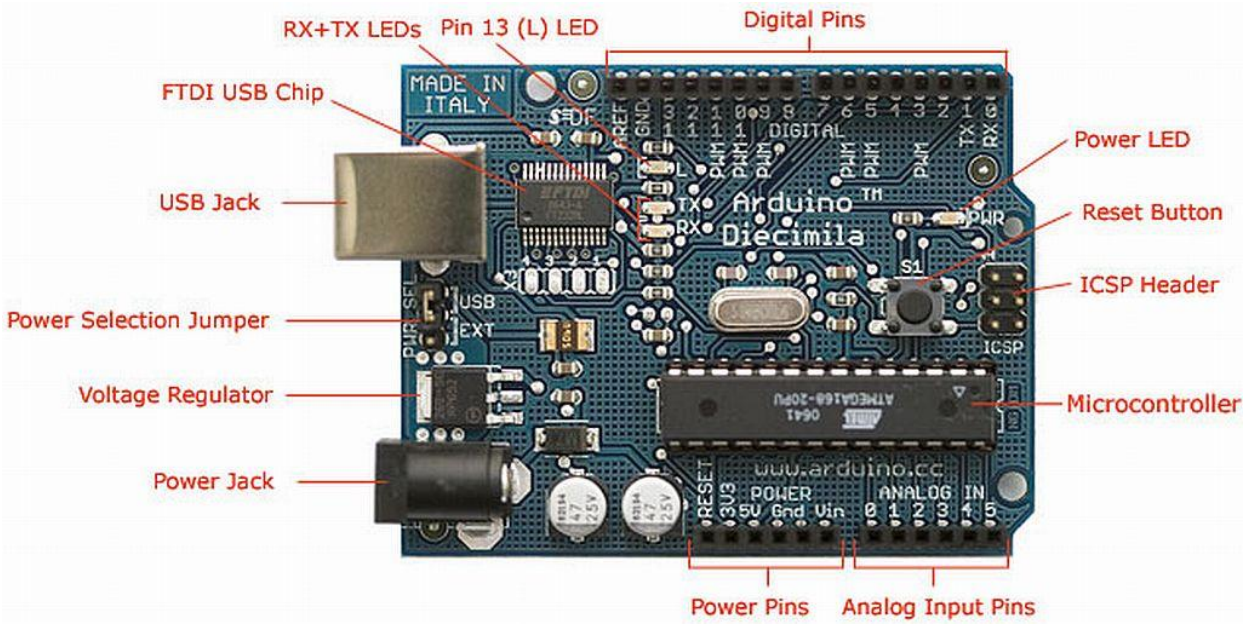

Рисунок 16 – Плата Arduino Diecimila

На концептуальном уровне, все платы программируются через RS-232 (последовательное соединение), но реализация этого способа отличается от версии к версии. Плата Serial Arduino содержит простую инвертирующую схему для конвертирования уровней сигналов RS-232 в уровни ТТЛ, и наоборот. Текущие платы, вроде Diecimila, программируются через USB, что осуществляется благодаря микросхеме конвертера USB-to-serial вроде FTDI FT232. В некоторых вариантах, таких как Arduino Mini или неофициальной Boarduino, для программирования требуется подключение отдельной платы USB-to-serial или кабеля.

Платы Arduino позволяют использовать большую часть I/O выводов микроконтроллера во внешних схемах. Например, в плате Diecimila доступно 14 цифровых вводов/выводов(уровни «LOW» -0В и «HIGH» -5В), 6 из которых могут выдавать ШИМ сигнал, и 6 аналоговых входов(0-5В). Эти выводы доступны в верхней части платы через 0,1 дюймовые разъёмы типа «мама». На рынке доступны несколько внешних плат расширения, известных как «shields». В проекте InMoov используется плата Arduino Uno.

2 ХАРАКТЕРИСТИКА ФУНКЦИОНАЛЬНЫХ ПОДСИСТЕМ

В ходе работы были выделены следующие функциональные подсистемы:

1) подсистема визуализации робота InMoov – 3D-модель робота InMoov, спроектированная и разработанная в соответствии со всеми особенностями объекта виртуализации. Включает в себя трехмерную схему робота без нижних конечностей. Данная функциональная подсистема должна наглядно демонстрировать все особенности физической модели и быть интерактивной.

2) средства физического управления макетом – подсистема обеспечивающая связь между пользователем и виртуальным макетом. Основной задачей этой подсистемы является управление 3D-моделью по средствам контроллера в виде джойстика и мышью.

3) конвертация управляющих сигналов – подсистема, задача которой состоит в кодировке управляющего сигнала и отправке этого сигнала в порт контроллера.

Далее каждая подсистема будет рассмотрена более подробно.

2.1 Подсистема визуализации робота InMoov

Данная подсистема является основой для сборки, наладки и управления роботом InMoov. Виртуальный макет является основной частью данной подсистемы, представляющий собой модель объекта, производство которого планируется, отображающую его функционирование и визуализирующую внешний вид, его структуру. Следует отметить, что виртуальный макет может имитировать как объект целиком, так и некоторые его стороны. При его разработке учитываются те особенности объекта, которые будут важны для его дальнейшего производства. Например, если для производства реального прототипа нет необходимости в моделировании его функционирования, то достаточно создания виртуального макета, визуализирующего его строение и внешний вид.

Виртуальный макет может представлять собой схему, двухмерную или трехмерную графическую модель. Важным моментом является его интерактивность, то есть возможность воздействия на виртуальный макет объекта, способ-

ствующего изменению его состояния. Виртуальный макет создается интерактивным, если это необходимо для изготовления объекта или принятия решения о его производстве.

Виртуальный макет позволяет не только исследовать объект, но и создавать четко подогнанные комплектующие для него. На рисунках 17 – 18 представлены примеры программных комплексов, реализующих виртуализацию различных роботизированных систем.

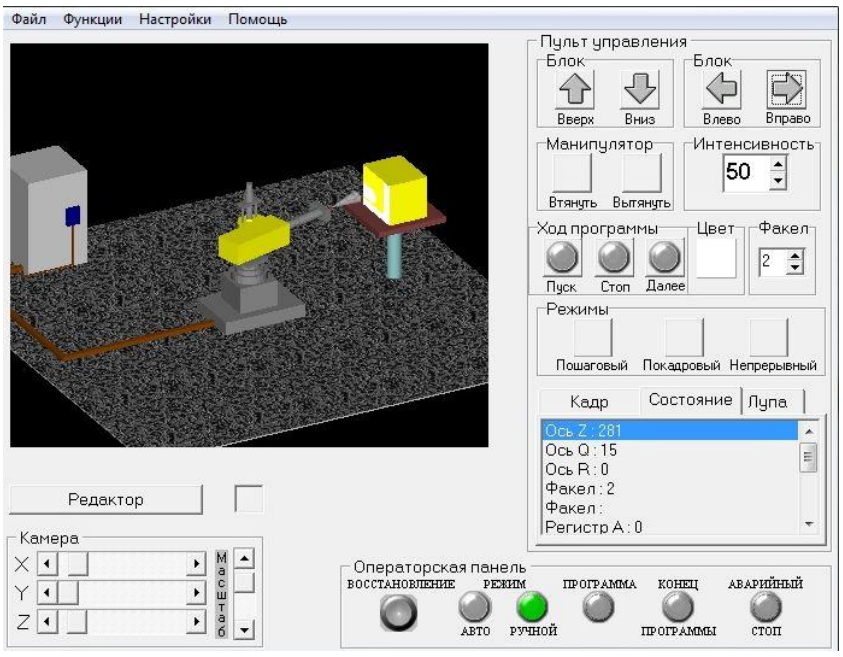

Рисунок 17 – Виртуальный макет покрасочного робота

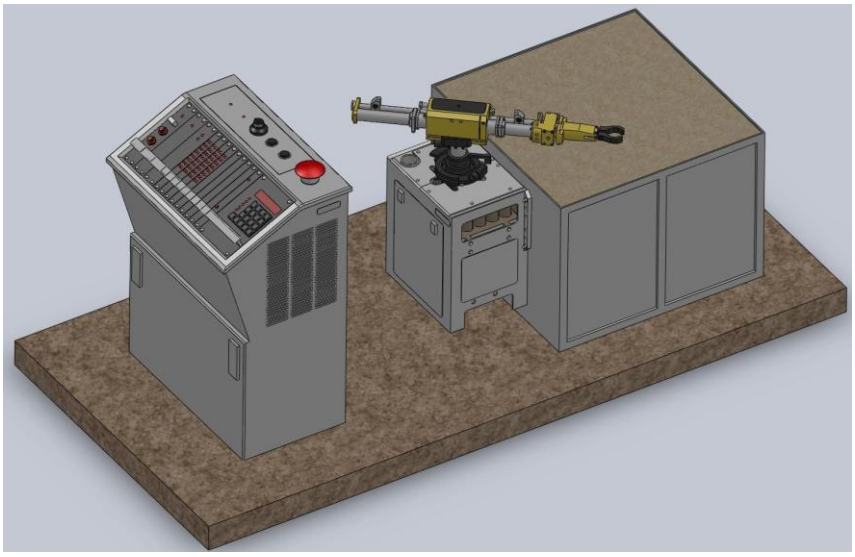

Рисунок 18 – Виртуальный макет пневматического робота

Весомый вклад в развитие программного обеспечения, ориентированного на создание цифровых моделей внесла компания Autodesk. Программные пакеты AliasStudio, Invertor, Productstream и Showcase предназначены для корпоративного использования и часто эксплуатируются совместно. Они позволяют моделировать цифровые модели продуктов еще до их производства. При этом доступ к данным прототипам предоставляется множеству сотрудников организации. Для персонального использования не пригодны, в связи с дороговизной программных пакетов.

В работе ставились задачи, которые должны помогать при сборке, наладке и эксплуатации робота InMoov. Уделялось внимание тем аспектам, которые важны для ее реализации, а именно: строению робота, принципу функционирования его подвижных частей, строению электронной схемы, осуществляющей функционирование реального робота.

Для реализации движения подвижных частей виртуального макета, также необходимо разработать скелетную анимацию.

Скелетная анимация – это способ анимирования трехмерных моделей в моделировании, мультипликации и компьютерных играх. Заключается в том, что разработчик создает скелет, представляющий собой как правило древообразную структуру костей, в которой каждая последующая кость «привязана» к предыдущей, то есть повторяет за ней движения и повороты с учётом иерархии в скелете. Далее каждая вершина модели «привязывается» к какой-либо кости скелета. Таким образом, при движении отдельной кости двигаются и все вершины, привязанные к ней. Благодаря этому задача аниматора сильно упрощается, потому что отпадает необходимость анимировать отдельно каждую вершину модели, а достаточно лишь задавать положение и поворот костей скелета.

Анимационный скелет должен соответствовать всем особенностям объекта виртуализации, в связи чем была построена схема, на основе которой должен быть построен анимационный скелет. Данная схема приведена на рисунке 19.

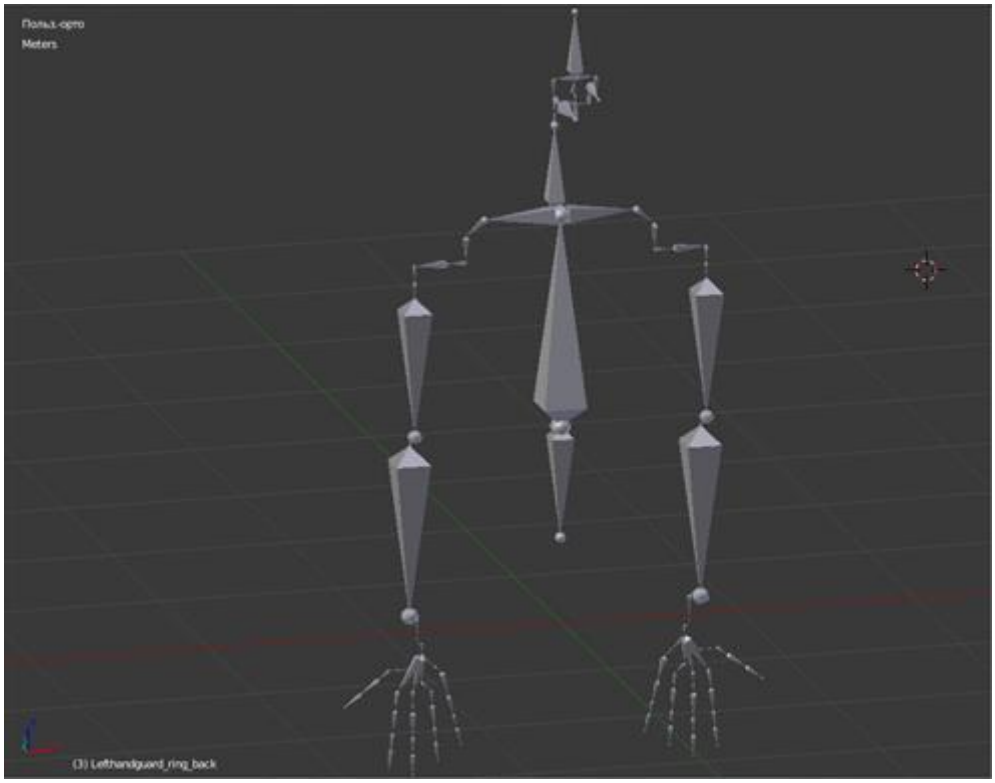

Рисунок 19 – Анимационный скелет

2.2 Подсистема физического управления виртуальным макетом

В ходе выполнения работы над реализацией подсистемы управления виртуальным макетом, было принято решение построить схему, наглядно демонстрирующую принцип работы данной подсистемы. Схема управления виртуальным макетом представлена на рисунке 20.

Рисунок 20 – Схема управления виртуальным макетом

На первом этапе пользователь подает сигнал при помощи управляющего средства в виде джойстика, далее сигнал управления подается на вход программного обеспечения и обрабатывается им, далее обработанный управляющий сигнал воздействует на виртуальный макет. На конечном этапе пользователю демонстрируется результат выполнения его команд.

В связи с этим для реализации данной подсистемы необходимо выбрать управляющее средство, которое удовлетворяет таким требованиям как удобство, эргономичность, возможность дистанционного управления, низкая цена. Учитывая все требования выбор пал на джойстик от фирмы DNS, модель Z-120NP.

Далее необходимо разработать программное обеспечение для обработки сигнала управления и воздействовать на виртуальный макет так как необходимо пользователю. Также система управления должна сконфигурировать массив данных, представляющий собой матрицу A размерностью n×4, где n число строк, зависящее от количества используемых для управления макетом двигателей. Данная матрица представлена формулой (1).

$$A = \begin{pmatrix} a_{1\,1} & a_{1\,2} & a_{1\,3} & a_{1\,4} \\ \dots & \dots & \dots & \dots \\ a_{n\,1} & a_{n\,2} & a_{n\,3} & a_{n\,4} \end{pmatrix} \quad (1)$$

Значения первого столбца матрицы A указывают на номер двигателя участвующего в управлении, второй столбец показывает обозначает угол, второй столбец обозначает конечный угол, третий столбец указывает с какой скоростью должно производиться движение.

Система управления макетом должна оперативно реагировать на поданный пользователем сигнал управления без задержек. Это должно быть достигнуто путем оптимизации разрабатываемого программного обеспечения.

В виду отсутствия математических операций с используемой матрицей, необходимость в математическом обеспечении отсутствует.

2.3 Подсистема конвертации управляющих сигналов в код Arduino

Следующая задача, которая должная быть решена, состоит в том, чтобы связать виртуальную модель с физической. Данная задача должна быть решена путем разработки программного обеспечения конвертирующего управляющий сигнал, поступающий с джойстика, в сигнал понятный для микроконтроллера Arduino.

Для реализации подсистемы конвертации управляющих сигналов была построена блок-схема алгоритма работы программного обеспечения. На рисунке 21 представлена блок-схема алгоритма.

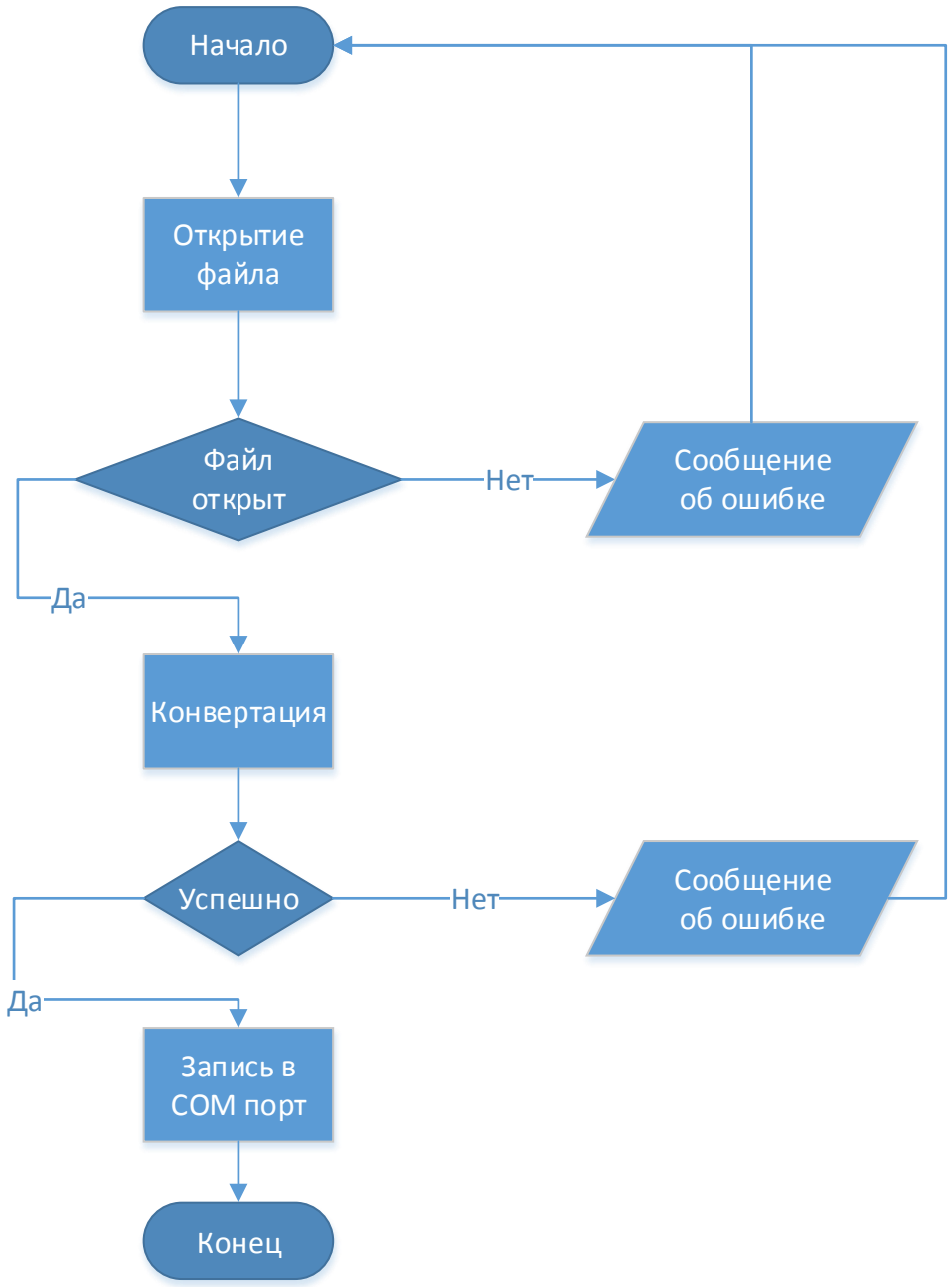

Рисунок 21 – Блок-схема алгоритма конвертации

На первом этапе пользователю необходимо открыть файл сконфигурированный подсистемой управления макетом. В случае возникновения ошибки открытия файла, пользователь должен выбрать другой файл для загрузке. При успешном открытии файла, программа должна конвертировать данные в кодировку ASCII. В случае возникновения ошибки конвертирования, программа сообщит об этом. После успешной конвертации программа будет иметь возможность записать конвертированные данные в COM-порт микроконтроллера Arduino для управления физической моделью.

3 ХАРАКТЕРИСТИКА ОБЕСПЕЧИВАЮЩИХ ПОДСИСТЕМ

В ходе работы было выделено три обеспечивающие подсистемы, а именно:

1) информационное обеспечение – это система концепций, методов и средств, предназначенных для обеспечения пользователей (потребителей) информацией. Назначение подсистемы информационного обеспечения состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений. Информационное обеспечение является основой разработки всего комплекса средств информационной технологии, определяя организационное, техническое и программное обеспечение;

2) программное обеспечение – комплексы программ, ориентированные на пользователей и предназначенные для решения типовых задач обработки информации. Они служат для расширения функциональных возможностей процессов контроля и управления. В программное обеспечение входят пакеты прикладных программ, которые реализуют экономико–математические модели разной степени адекватности, отражающие функционирование реального объекта;

3) техническое обеспечение – представляет комплекс технических средств, предназначенных для обработки данных в ЭИС. В состав комплекса входят электронные вычислительные машины, осуществляющие обработку экономической информации, средства подготовки данных на машинных носителях, средства сбора и регистрации информации, средства передачи данных по каналам связи, средства накопления и хранения данных и выдачи результатной информации, вспомогательное оборудование и организационная техника.

Далее рассмотрим каждую обеспечивающую подсистему в отдельности и выделим необходимые их составляющие.

3.1 Информационное обеспечение

В качестве информационного обеспечения в данной работе выступает сайт открытого проекта InMoov.

В связи с тем, что данный проект является зарубежной разработкой, информация на сайте проекта представлена на английском языке. В связи с этим необ-

ходимо было перевести нужную информацию на русский язык. Для выполнения этой задачи был выбран онлайн-переводчик Яндекс, который также можно отнести к информационным обеспечивающим подсистемам.

На данной сайте представлено большое количество инструкций для печати комплектующих деталей на 3D-принтере, по сборке и наладке робота InMoov, программированию аппаратной вычислительной платформы Arduino.

Также на сайте присутствует форум, на котором можно задать интересующие вопросы при работе с роботом.

Все необходимые компоненты робота InMoov, в виде 3D-моделей, были загружены с сайта проекта. Главная страница сайта представлена на рисунке 22.

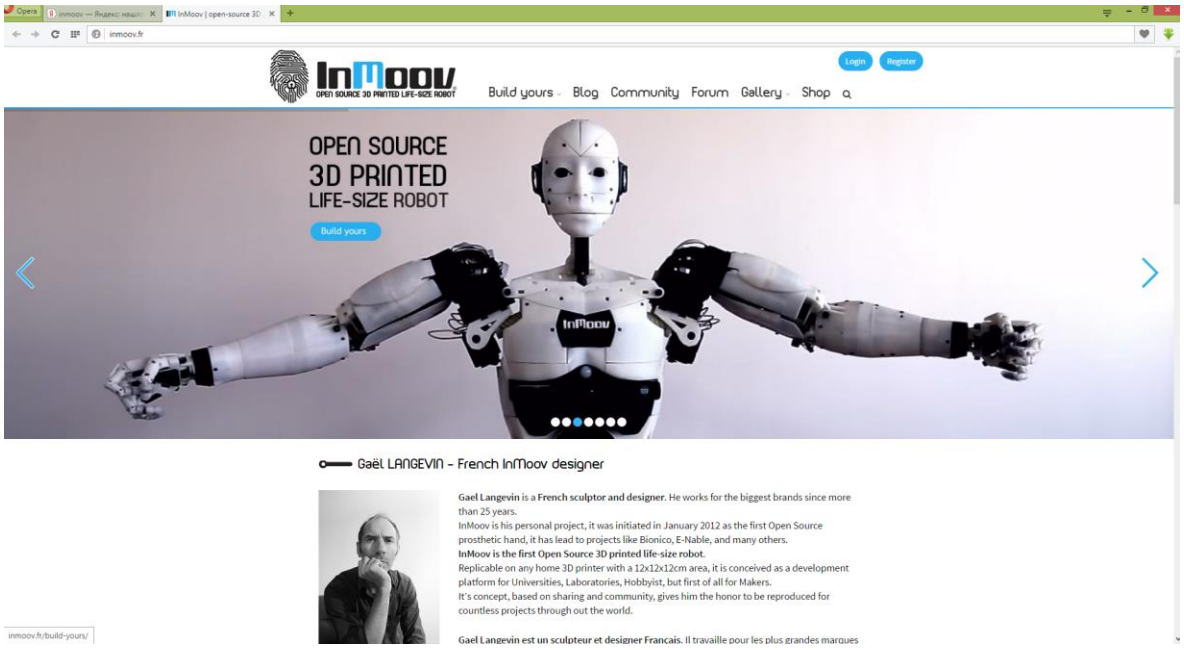

Рисунок 22 – Главная страница сайта проекта InMoov

Также к информационному обеспечению относятся стандарты, которые должны быть учтены при разработке программного обеспечения:

– ГОСТ 19.001-77 – «ЕСПД. Общие положения»;

– ГОСТ 19.004-80 – «ЕСПД. Термины и определения»;

– ГОСТ 19.004-80 – «ЕСПД. Виды программ и программных документов»;

– ГОСТ 19.101-77 – «ЕСПД. Стадии разработки»;

– ГОСТ 19.103-70 – «ЕСПД. Обозначение программ и программных документов»;

– ГОСТ 19.104-78 – «ЕСПД. Основные надписи»;

– ГОСТ 19.402-78 – «ЕСПД. Описание программы»;

– ГОСТ 19.502-78 – «Описание применения. Требования к содержанию и оформлению»;

– ГОСТ 19.505-79 – «Руководство оператора. Требования к содержанию и оформлению»;

– ГОСТ 19.508-79 – «Руководство по техническому обслуживанию. Требования к содержанию и оформлению»;

– ГОСТ 24.301-80 – «Общие требования к выполнению текстовых документов».

3.2 Программное обеспечение

Для реализации данной работы было использовано следующее ПО:

1) Blender – свободный, профессиональный пакет для создания трёхмерной компьютерной графики, включающий в себя средства моделирования, анимации, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов» (Node Compositing), а также для создания интерактивных игр;

2) Visual Studio 2013 – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств;

3) Unity 3D – инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows, OS X;

4) Autodesk 123D Circuits – веб-приложение с довольно хорошей имитацией платформы Arduino, которое позволяет в визуальном режиме прямо из браузера редактировать код и строить схемы без паяльника и проводов, что довольно удобно.

3.2.1 Описание программы Blender

Программа для 3-мерного моделирования Blender - это полноценный бесплатный редактор. Необычный внешний вид приложения свидетельствует о том,

что проект создавался разработчиками "с нуля", они не привязывались к внешнему виду иных подобных утилит.

Технология, реализованная в Blender, позволяет видоизменять интерфейс программы до неузнаваемости. Особенность состоит в том, что во время создания 3-мерной сцены, окно утилиты можно разделить на части, каждая из них будет представлять собой независимое окно с определенным видом 3D сцены, настройками объекта, линейкой временной шкалы.

Количество таких частей ограничивается лишь разрешением экрана. Независимо от их числа, они никогда не будут между собой пересекаться и накладываться друг на друга.

Еще одна положительная сторона Blender - великолепная поддержка "горячих" клавиш. С их помощью можно выполнять любые операции. Разумеется, сочетаний много, поэтому тяжело запомнить их все сразу, но их знание существенно упрощает работу в программе.

Создание 3-мерных моделей выполняется посредством полигональных и NURBS-поверхностей. Приложение также располагает инструментами сплайнового моделирования, а для формирования 3D-объектов используются еще кривые Безье и B-сплайны.

Инструментарий утилиты настолько универсален, что посредством Blender можно воссоздавать составные органические формы. Для этого применяются метаболы и технология "3-мерной лепки" при помощи виртуальных кистей. Изменения в геометрию вносятся в режиме симметрии, что крайне необходимо в процессе моделирования персонажей.

Программа пригодна для создания обычной анимации и работы над оснасткой персонажей. Приложение позволяет формировать скелет и привязывать кости к внешней оболочке. Работа редактора основывается на прямой и инверсной кинематике.

3.2.2 Описание программы Visual Studio 2013

Microsoft Visual Studio – это новая разработка компании Microsoft, позволяющая создавать приложения, работающие на платформе .net. Особенность этой

платформы заключается в широком наборе сервисов, которые доступны в различных языках программирования. При этом сервисы реализуются в виде промежуточного кода, который не зависит от базовой архитектуры. Едва ли не главной целью создания такой платформы было оснащение разработчиков специальными сервисно-ориентированными приложениями, которые могли бы работать на любой платформе, начиная от персонального компьютера и заканчивая мобильным устройством.

Microsoft Visual Studio объединяет в себе огромное количество функций, позволяющих осуществлять разработки для Windows всех версий, в том числе и 8, Интернета, SharePoint, различных мобильных устройств и облачных технологий. В Visual Studio реализуется новая среда разработчика, благодаря которой создавать приложения стало проще. Microsoft Visual Studio - это обновленная и упрощенная программная среда, для которой характерна высокая производительность, причем она не зависит от особенностей оборудования.

Каждая новая версия программы состоит из новейших инструментов и технологий, позволяющих разрабатывать приложения с учетом особенностей и положительных моментов современных платформ. Например, Visual Studio 2013 может поддерживать более ранние версии, в том числе Windows XP и Windows Server 2003. При этом разработчикам открыта дорога к созданию новых и модернизации уже существующих приложений, предназначенных для ранних версий ОС Windows. Стоит отметить, что в процессе использования поддерживаемых системой вариантов исходные файлы, проекты и решения в программе Visual Studio будут работоспособными, но исходный код может нуждаться в изменениях.

Visual Studio Ultimate 2013 представляет собой передовую программу, которая дает возможность любым по размеру командам осуществлять проектирование и создание привлекательных приложений. Благодаря инструментам гибкого планирования можно внедрять методы последовательной разработки и применяться гибкие методологии в темпе, удобном для пользователя.

С помощью расширенных средств моделирования, обнаружения и проектирования можно максимально полно описать систему, которая позволит наиболее удачно реализовать конкретную концепцию архитектуры.

3.2.3 Описание программы Unity 3D

Современный движок с широчайшим набором возможностей для создания качественных видеоигр. Поддерживает все популярные типы игровых устройств (ПК, смартфоны, планшеты, Nintendo Wii, PlayStation, Xbox) и операционных систем (Windows, Android, Mac, iOS). Включает возможность создания игр для интернет-обозревателей (браузеров) на основе Flash и Web Player.

За 10 лет своего существования, Unity3D зарекомендовал себя как надежный, многофункциональный движок, пригодный для профессиональных и начинающих разработчиков. Благодаря постоянному развитию, под знамена Unity встали более 2-х миллионов довольных пользователей, создавших не одну сотню увлекательных компьютерных игр.

Возможности движка объединили в нем самые главные инструменты, сведя к минимум использование сторонних программ. Для более удобной разработки игр, в Unity3D 5 был добавлен ряд полезных функций. К примеру, для создания 3D-моделей можно воспользоваться современным встроенным редактором, не прибегая к использованию других программ. Здесь вы можете без труда создать/обработать физику, шейдеры, звуки, ландшафты, или воспользоваться богатой библиотекой скриптов, чтобы расширить функционал своей игры.

3.3 Техническое обеспечение

Подсистема технического обеспечения представляет собой комплекс технических средств, реализующий процессы сбора, передачи, обработки, отображения и хранения информации. Подсистема технического обеспечения включает в себя универсальные и управляющие цифровые вычислительные машины (УВМ), включая их периферийные устройства, устройства сбора и передачи информации, в том числе и рассмотренные выше системы телемеханики, а также устройства отображения информации диспетчеру. Необходимым условием построения под-

системы технического обеспечения является возможность сопряжения всех видов технических устройств, входящих в эту подсистему.

Для реализации данной работы были выделены следующие технические средства:

1) ноутбук Acer Aspire V3-771G (процессор Intel Core i5-3210M 2.5Ghz, видеокарта NVIDIA GeForce GT 630M, оперативная память DDR3 Memory 4Gb, жесткий диск HDD 750Gb). Данная модель изображена на рисунке 23.

Рисунок 23 – Ноутбук Acer Aspire V3-771G

2) джойстик DNS Z-120NP – является игровым манипулятором в играх любого типа на персональных компьютерах, благодаря двум прецизионным контроллерам, большому количеству кнопок, переключателю направления и поддержке эффектов вибрации, реализуемых двумя встроенными электромоторами.

3) приемопередатчик Bluetooth – предназначен для передачи сигнала с джойстика на ПК. Подключается к компьютеру через USB. Реализует возможность дистанционного управления. Включен в комплект с вышеописанным джойстиком. Джойстик и приемопередатчик изображены на рисунке 24.

Рисунок 24 – Джойстик DNS Z-120NP и приемопередатчик

4 РАЗРАБОТКА ВИРТУАЛЬНОГО МАКЕТА РОБОТА INMOOV

4.1 Сборка 3D-модели робота InMoov

Трехмерная модель улучшает восприятие объекта, при наличии множества отдельных деталей, позволяет собрать их воедино и получить целостное представление о готовом объекте.

В связи с тем, что каждая деталь робота – это отдельный 3D-объект, основной задачей данного этапа работы является сборка каждой отдельной детали в единую модель робота.

Для данной работы были использованы все детали открытого проекта InMoov. С официального сайта был скачан архив с деталями в формате .stl. Пакет для создания компьютерной графики Blender, используемой в качестве программной обеспечивающей системой, не поддерживает напрямую данный формат. Необходимо было сначала конвертировать все детали в формат .blend, поддерживаемый используемым пакетом.

Каждая 3D-модель детали представляет собой многополигональные модели, спроектированные таким образом, чтобы они поддерживались трехмерными принтерами.

Каждая деталь была импортирована в один проект, так как в таком случае проще совершать дальнейшую сборку трехмерной модели робота InMoov. Все детали выполнены в одном масштабе, для дальнейшей сборки никаких дополнительных манипуляций не требовалось.

Все детали выполнены в одном масштабе, для дальнейшей сборки никаких дополнительных манипуляций с ними не требовалось.

Разработка была начата с трехмерной модели головы. В связи с тем, что голова в данной работе не используется в качестве управляемого объекта, было принято решение упростить задачу путем разработки упрощенного варианта.

Упрощение заключалось в том, чтобы голову сделать всего из трех объектов. Первая часть составляет девяносто процентов всей головы, вторая это нижняя челюсть, и третья это глаза.

Для моделирования была взята первая попавшаяся фотография мужчины в интернете в профиль и анфас. Далее было произведено наложение полигональных точек повторяющих контуры лица с фронтальной и боковой стороны. На рисунках 25 и 26 изображен процесс моделирования.

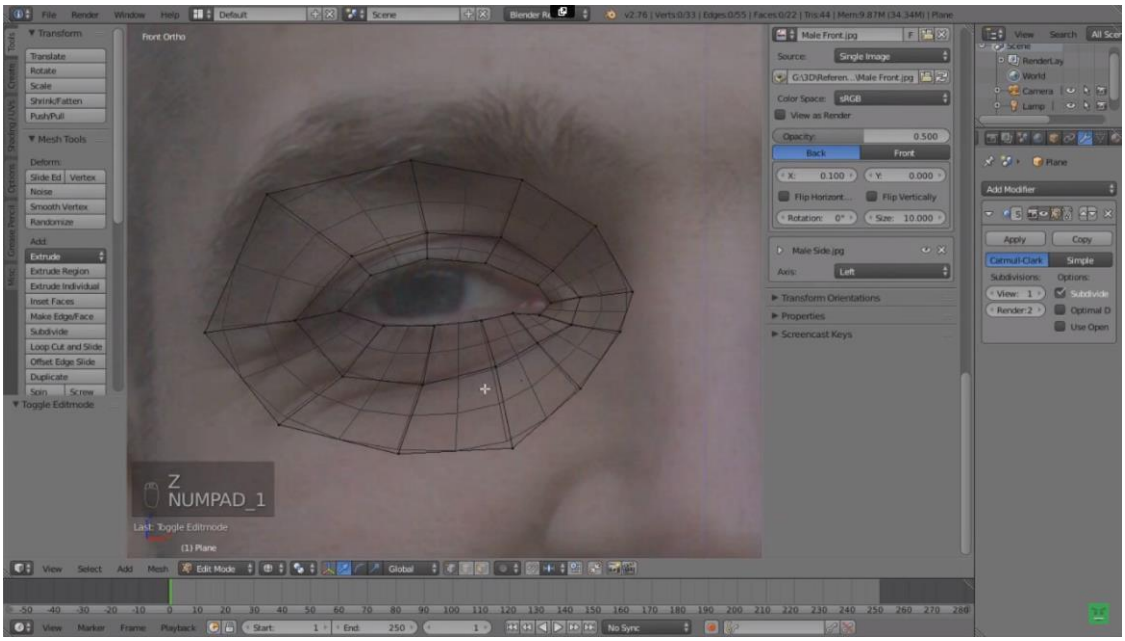

Рисунок 25 – Расстановка полигонов с фронтальной стороны

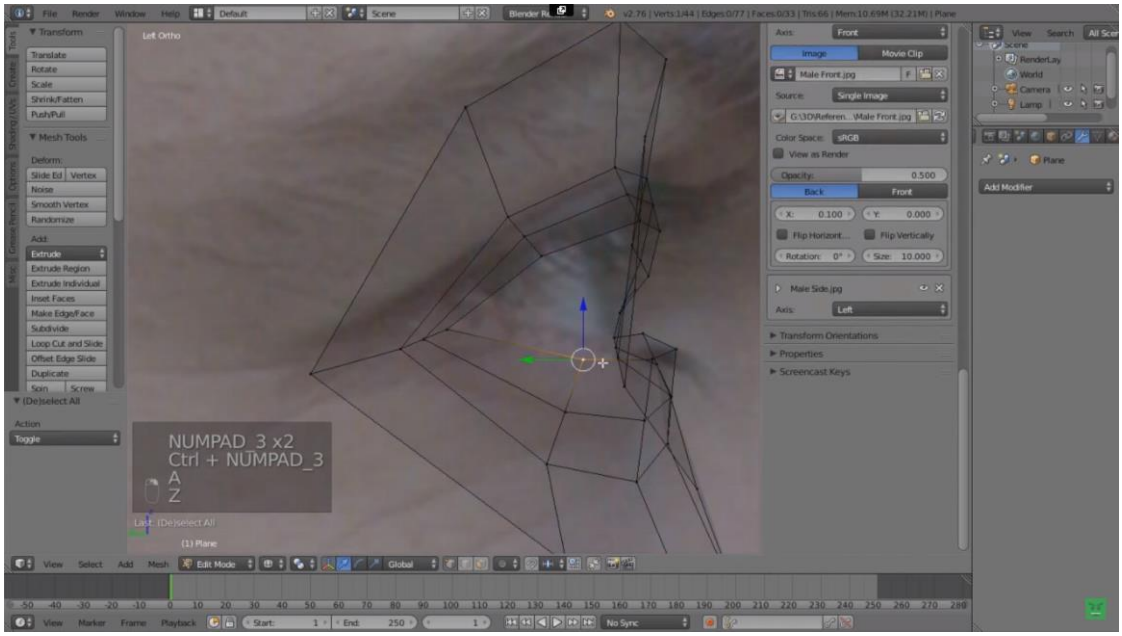

Рисунок 26 – Расстановка полигонов с боковой стороны

После расстановки всех полигонов с одной и с другой стороны, была произведена операция «Mirror» (зеркало), для генерации второй половины лица. После

чего было произведено наложение материала и сглаживание полигонов для устранения неровностей и резких переходов.

Далее было произведено масштабирование смоделированного объекта при помощи инструмента «Scale», в соответствии с размерами импортированных объектов.

Результат моделирования демонстрируется на рисунке 27, на котором изображена готовая основная часть головы для робота InMoov.

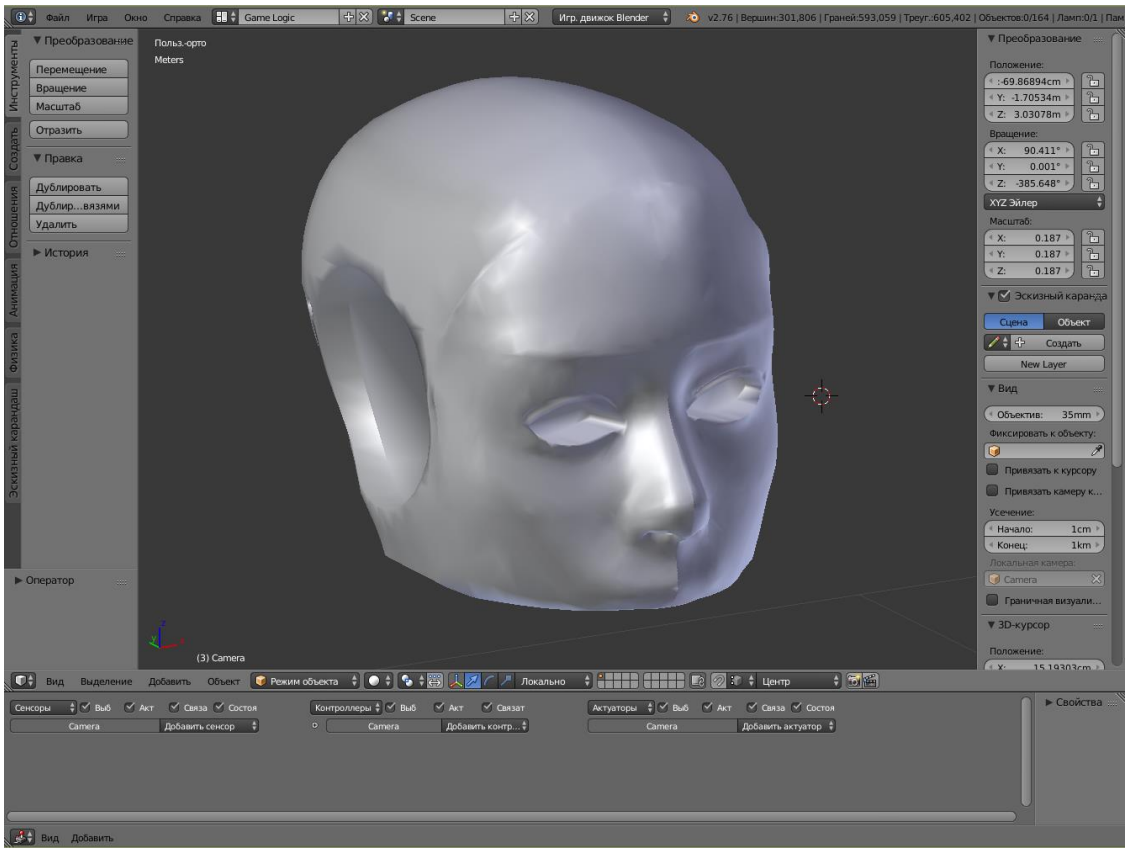

Рисунок 27 – Основная часть головы робота InMoov

Далее таким же образом было смоделирована нижняя челюсть, путем наложения полигональных точек и зеркального отражения второй половины.

В качестве подвижного сустава был смоделирован рычаг имитирующий челюстной сустав.

Результат моделирования показан на рисунке 28.

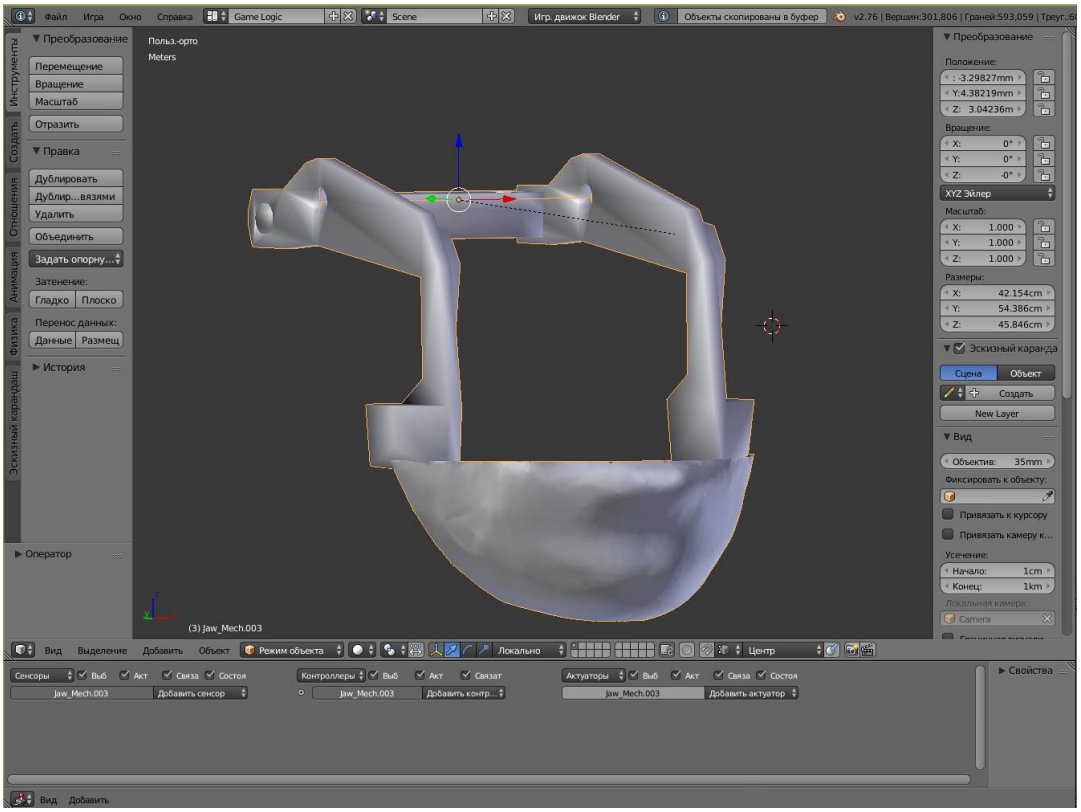

Рисунок 28 – Нижняя челюсть робота InMoov

Глаза были смоделированы при помощи примитивов, а именно сфер. Результат моделирования представлен на рисунке 29.

Рисунок 29 – Глаза робота InMoov

Далее была полностью собрана голова робота. Результат сборки показан на рисунке 30.

Рисунок 30 – Голова робота InMoov

После моделирования головы было принято решение начать сборку руки робота InMoov. Сначала была произведена сборка трехмерных моделей пальцев. Каждая модель пальца представляет собой объект, содержащий шесть элементов – по два элемента на каждую фалангу. Для удобства объекты были разъединены на отдельные элементы.

Сборка производилась путем изменения угла наклона и положения каждого из элементов. После сборки, элементы каждого пальца были заново объединены в один объект – палец.

Результат сборки пальца изображен на рисунке 31.

Рисунок 31 – Палец робота InMoov

Таким образом были собраны все пять пальцев робота. Следующим шагом была сборка трехмерной модели ладони руки, состоящей из четырех компонентов.

Сборка производилась путем изменения положения компонентов модели кисти относительно друг друга. После сборки компоненты были объединены в объект – ладонь.

Трехмерные модели пальцев были собраны воедино с моделью ладони путем изменения положения друг относительно друга в пространстве координат XYZ.

Полученная модель была объединена в группу объектов – кисть, изображенную на рисунке 32.

Сюда же были включены трехмерные модели болтов соединяющих подвижные части кисти.

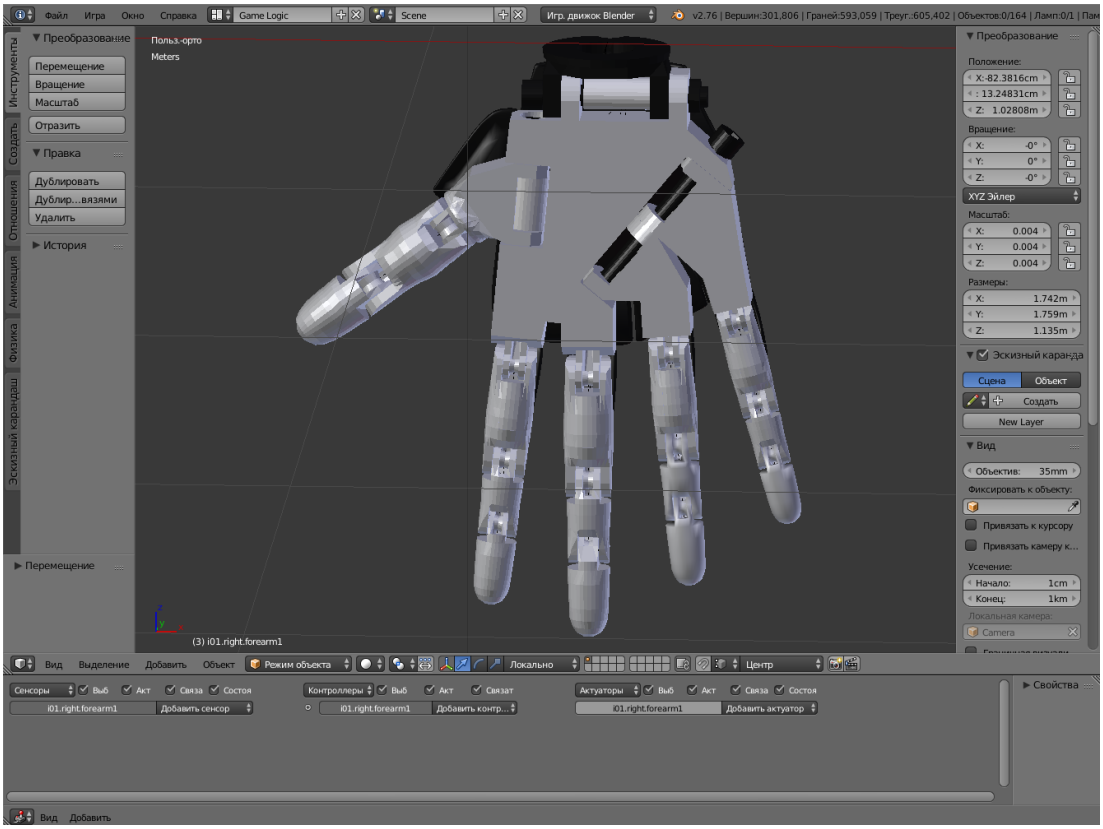

Рисунок 32 – Собранная модель кисти робота InMoov

Далее была произведена сборка трехмерной модели предплечья робота InMoov.

Сборка производилась путем изменения угла наклона и положения друг относительно друга в пространстве координат XYZ с последующим объединением всех деталей предплечья в один объект.

После сборки последовало объединение предплечья и кисти в один объект.

Таким же образом была произведена сборка плеча робота. В итоге трехмерные модели кисти, плеча и предплечья были объединены в единую группу объектов – руку робота InMoov.

Далее была произведена обработка объектов, путем устранения неровностей и стыков сглаживанием и наложением материалов.

На последнем этапе сборки руки было сделано объединение систем координат каждой детали в одну единую систему координат.

Результат сборки продемонстрирован на рисунке 33.

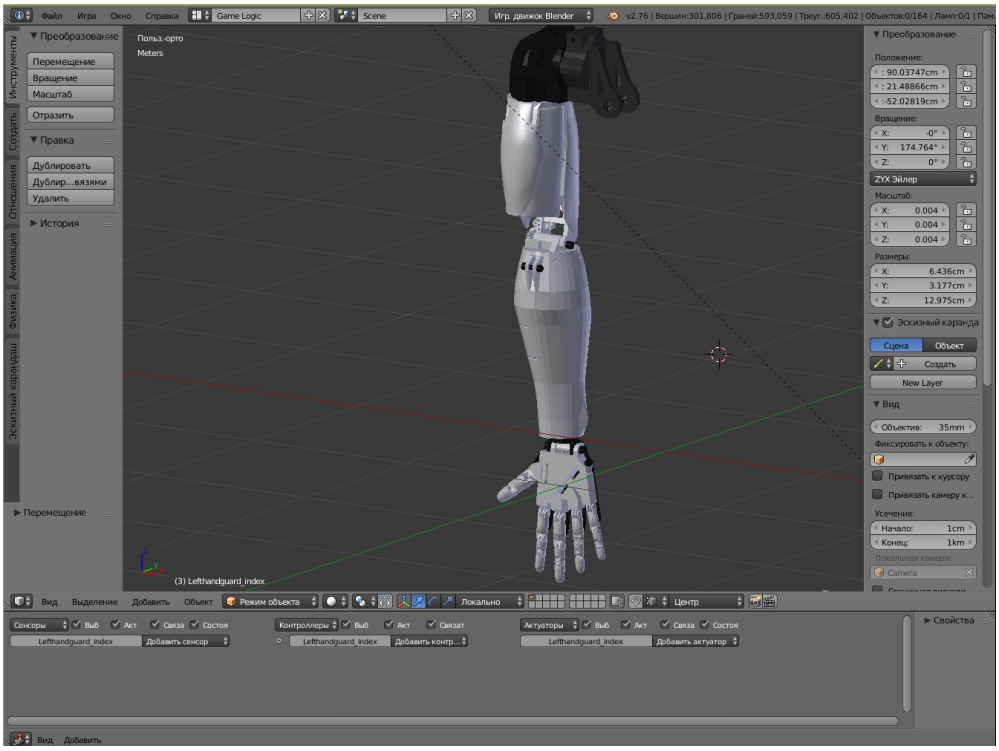

Рисунок 33 – Рука робота InMoov

В связи с тем что в проекте нет отличий левой руки от правой было принято решение воспользоваться инструментов «Зеркало», чтобы упростить задачу. Тем самым была получена и вторая руки робота. Результат на рисунке 34.

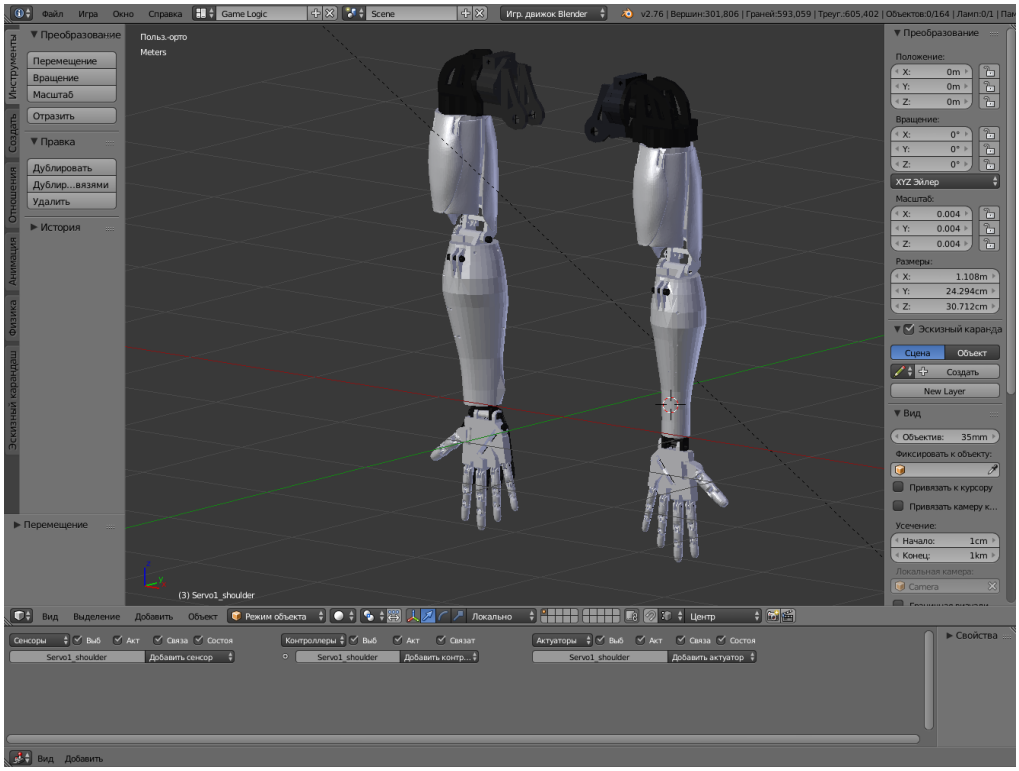

Рисунок 34 – Левая и правая рука робота InMoov

Далее была произведена сборка трехмерной модели туловища робота InMoov. Сборка производилась путем изменения угла наклона и положения друг относительно друга в пространстве координат XYZ с последующим объединением всех деталей туловища в один объект.

После сборки последовало объединение рук, головы и туловища в один объект, образующий всю 3D-модель робота InMoov. На рисунке 35 показана готовая модель всего робота.

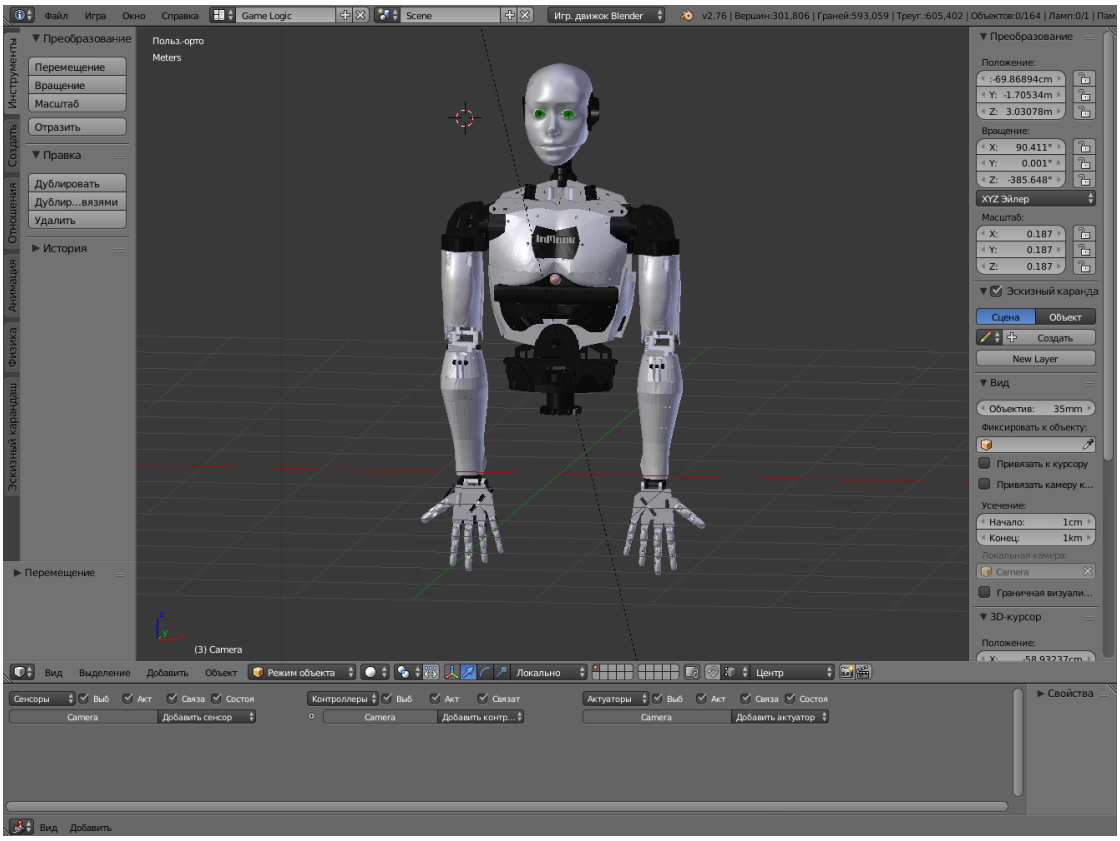

Рисунок 35 – Готовая 3D-модель робота InMoov

4.2 Разработка скелетной анимации подвижных частей робота

Существует два основных типа анимации: анимация с ключевыми кадрами и скелетная анимация. Эти типы анимации используются в различных ситуациях и каждый имеет свои преимущества и недостатки.

В анимации с ключевыми кадрами вы храните статические сетки модели для каждого кадра анимации. Если необходимо будет анимировать модель, по-

требуется экспортировать четыре различных статических сетки и менять рисуемую сетку во время каждого кадра.

Такая анимация называется анимацией с ключевыми кадрами (keyframed) потому что экспортируются только ключевые кадры анимации. Между первой и второй стадиями анимации может быть много промежуточных кадров, используемых для того, чтобы анимация была более плавной. Однако не требуется экспортировать их, потому что они получаются путем интерполяции первого и второго ключевых кадров. Например, в линейной интерполяции, местоположение каждой вершины сетки линейно интерполируется между первым и вторым кадром.

Одним из преимуществ анимации с ключевыми кадрами является быстрота, поскольку в ходе анимации ничего не требуется вычислять. Все кадры анимации хранятся в памяти и во время анимации необходимо только каждый раз менять рисуемую модель. Недостатком этого метода является необходимость хранить все сетки моделей в памяти, чтобы они могли быть быстро нарисованы. Если у модели сотни кадров анимации, ее сетку приходится сохранять сотни раз. В сценах с сотнями анимированных моделей, которые совместно используют одну и ту же анимацию, метод ключевых кадров может быть полезен. Использовать анимацию с ключевыми кадрами в Blender для моделей просто, поскольку в Blender уже есть классы, необходимые для поддержки статических моделей. Следовательно, в Blender можно представить анимированную с использованием ключевых кадров модель как массив статических моделей, используя, например, класс Model.

Другим способом анимации моделей является скелетная анимация. Скелетная анимация – это анимирование 3D-фигуры посредством относительно небольшого количества управляющих элементов, и внешне и по принципу работы, напоминающие скелет.

Для нее необходимо построить скелет модели, состоящий из нескольких костей, и затем присоединить каждую вершину сетки к кости скелета. В результате

при анимации скелета анимируется также и присоединенная к нему сетка, следуя анимации скелета.

Чтобы построить сетку модели, скелет и анимацию, можно использовать различные программы моделирования, поддерживающие скелетную (или костевую) анимацию, такие как 3ds Max, Maya, Blender и т.д. В данной работе используется программа Blender.

После того, как создана модель, надо экспортировать ее в формат, поддерживающий скелетную анимацию. Скелетная анимация является также анимацией с ключевыми кадрами, а это значит, что экспортируются только ключевые кадры скелетной анимации. Как и в анимации с ключевыми кадрами можно интерполировать кадры анимации скелета.

Скелетная анимация обладает несколькими преимуществами по сравнению с анимацией с ключевыми кадрами. Она позволяет легко смешивать анимации, позволяя одновременно применять к модели различные анимации. Скелетная анимация также позволяет соединять кости одного объекта с костями другого.

Например, если есть персонаж, который орудует мечом, можно присоединить кость меча к кости руки персонажа, в результате чего меч будет перемещаться при перемещении руки персонажа.

Сегодня скелетная анимация используется более широко, чем анимация с ключевыми кадрами. Держа это в уме сосредоточимся на скелетной анимации.

На данном этапе необходимо разработать анимационный скелет для робота InMoov.

Целью создания анимационного скелета является возможность анимирования подвижных частей робота, так как необходимо пользователю.

В связи с тем, что робот является подобием человека, процесс создания анимационного скелета упрощается. В пакете Blender присутствует инструмент «Human(Meta-Rig)», показанный на рисунке 36. Данный инструмент позволяет создать анимационный скелет человека.

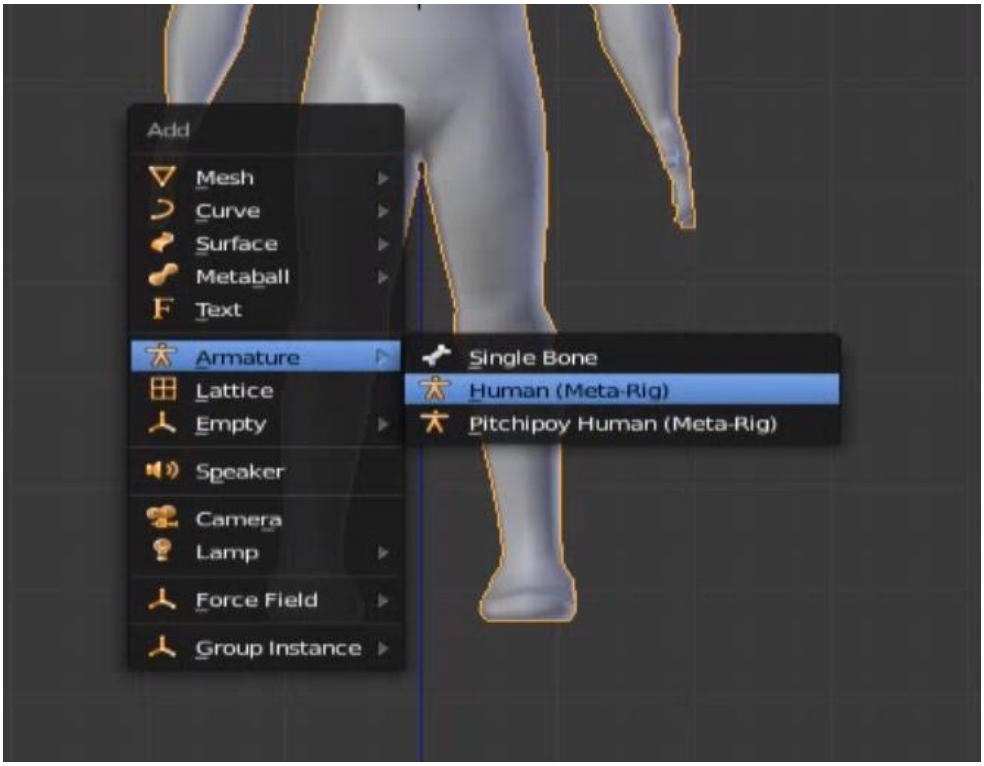

Рисунок 36 – Инструмент «Human(Meta-Rig)»

Результат работы инструмента «Human(Meta-Rig)» продемонстрирован на рисунке 37.

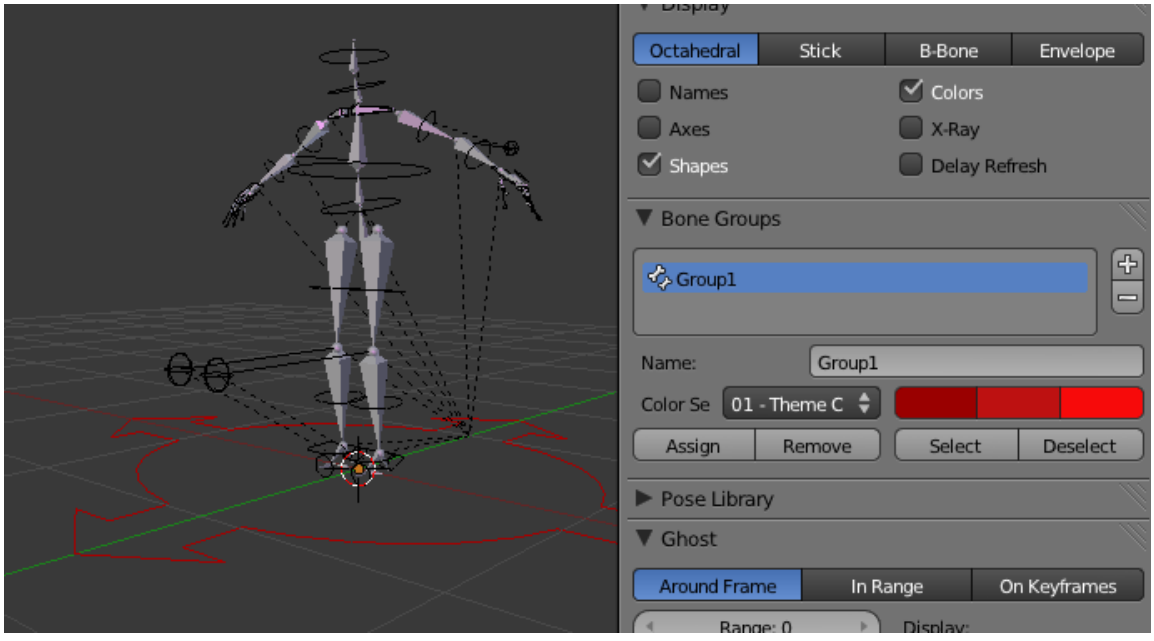

Рисунок 37 – Результат работы инструмента «Human(Meta-Rig)»

В связи с тем, что задача состоит в том, чтобы управлять руками робота, было принято решение удалить нижние части анимационного скелета, отвечающие за движение ног в виду их ненадобности.

Далее необходимо связать каждый элемент анимационного скелета с соответствующему ему 3D-объекту. Это позволит привести в движение необходимый 3D-объект при управлении виртуальным макетом.

На рисунке 38 изображен результат удаления ненужных частей анимационного скелете и соединения каждой кости с соответствующим 3D-объектом.

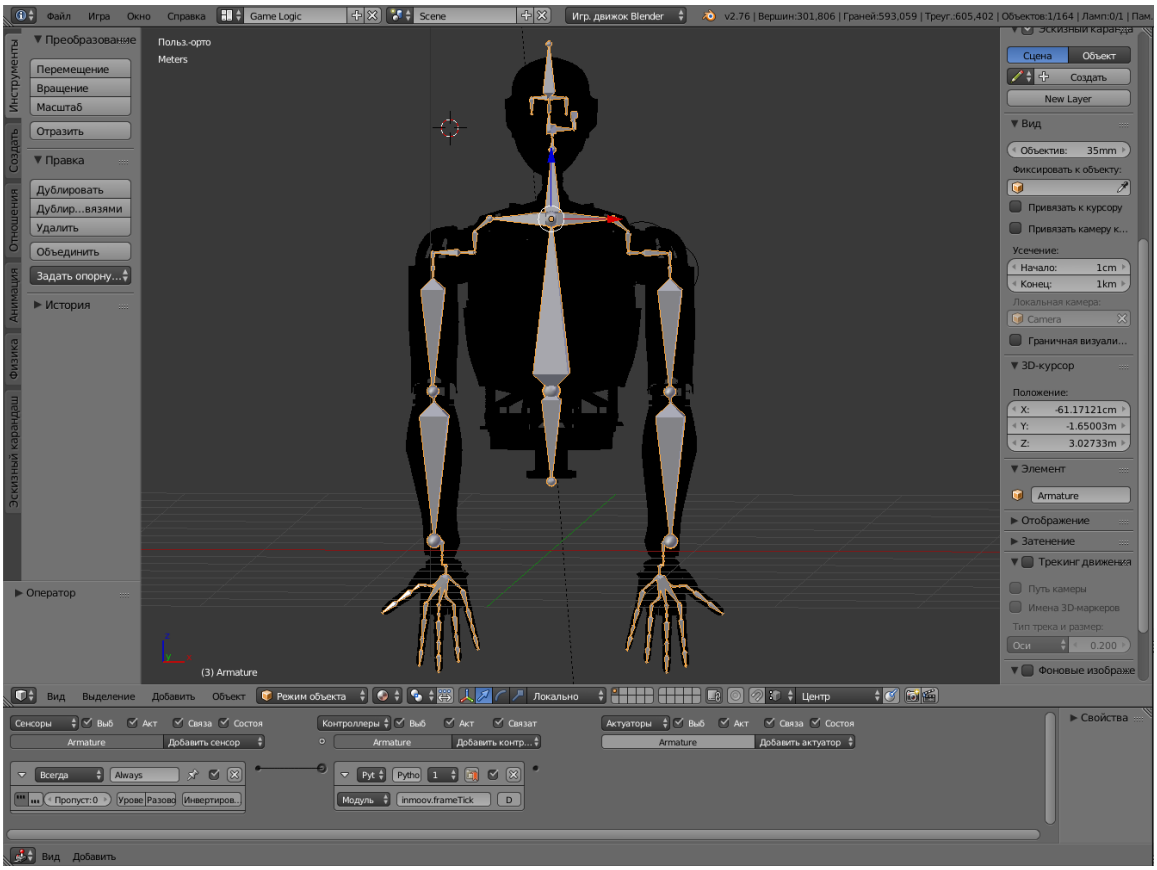

Рисунок 38 – Анимационный скелет робота InMoov

4.3 Разработка подсистемы для управления контроллером

На данном этапе работы необходимо разработать программное обеспечение, реализующее интерфейс для управления контроллером. Данное программное обеспечение было реализовано в движе Unity 3D.

В связи с тем, что для создания графического приложения используется движок Unity 3D, то для программирования необходимо использовать один из двух языков программирования – C# или JavaScript.

C# (произносится «си шарп») – объектно-ориентированный язык программирования. Разработан в 1998 – 2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Языки C# и JavaScript в настоящее время очень распространены среди разработчиков. При выборе языка написания интерфейса необходимо учитывать следующее: C# в подавляющем большинстве случаев используется при написании приложения, которое будет использоваться на компьютерах с операционной системой Windows, тогда как JavaScript подходит под любую платформу, проще в освоении и чаще используется при разработке приложений под мобильные устройства с операционной системой Android.

Наибольшее распространение получил язык C#, пользуется наибольшей поддержкой разработчиков и предназначен для проектирования наиболее сложных приложений.

Так как разрабатываемые приложения предполагается использовать на персональных компьютерах с операционной системой Windows, то был выбран язык программирования C#.

На первом этапе разработки интерфейса управления необходимо было импортировать разработанную 3D-модель робота в проект Unity 3D. Далее необхо-

димо размесить модель в центра сцены и создать камеру. Результат выполнения подготовки сцены в Unity 3D показан на рисунке 39.

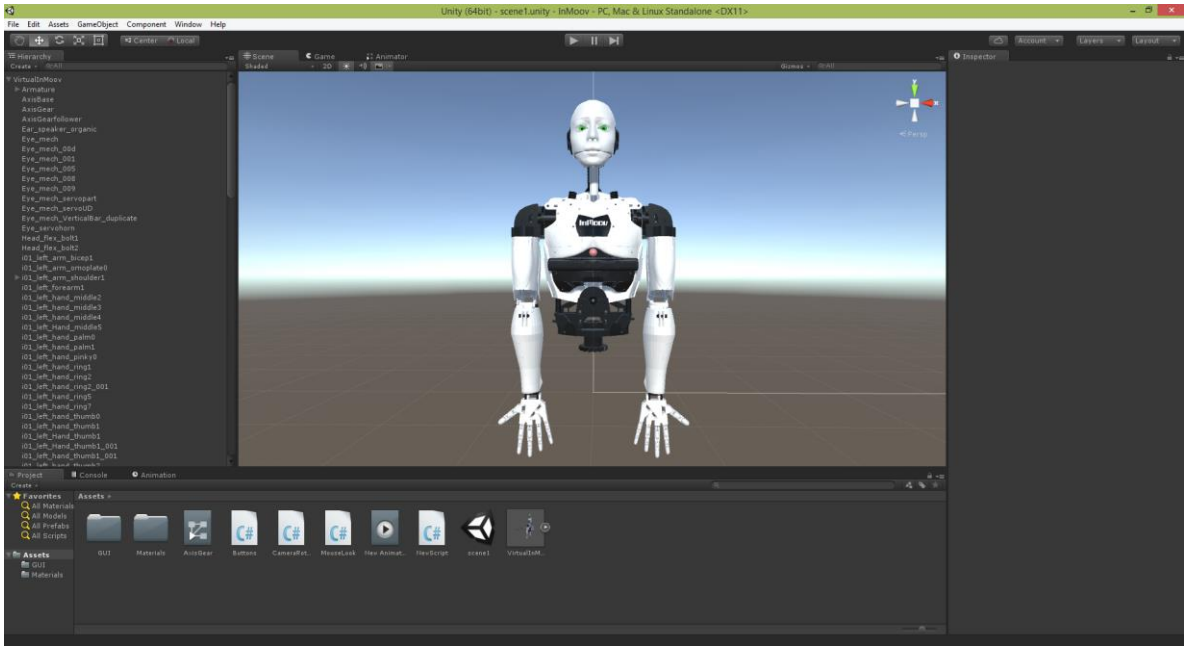

Рисунок 39 – Результат импорта 3D-модели в Unity 3D

Главной особенностью программирования в Unity 3D является наличие функций Start и Update. Функция Start предназначена для установления начальных значений. В функции Update пишется основной код, данная функция является главной и обладает циклическим характером – код, написанный в данной функции, повторяется каждый раз после завершения.

Еще одной особенностью программирования на движке Unity 3D является возможность привязки каждого отдельного элемента скрипта различным объектам. Для этого необходимо объявить отдельную переменную для каждого объекта.

На следующем этапе был написан скрипт «CameraRotateAround» для управления камерой, которые позволяет вращать камеру вокруг объекта с помощью мыши. Блок-схема алгоритма управления камерой изображена на рисунке 40. Листинг данного скрипта представлен в приложении.

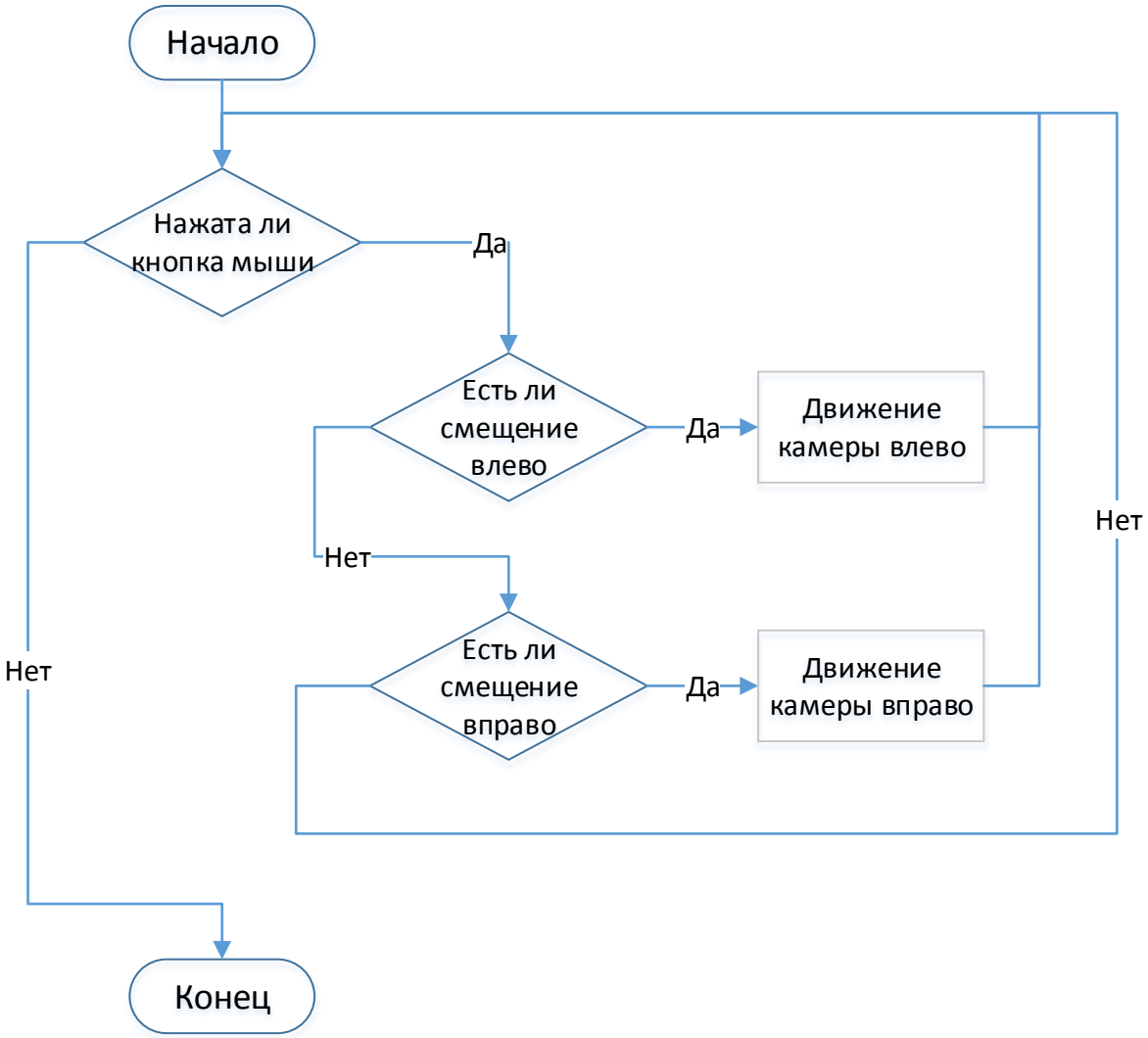

Рисунок 40 – Блок-схема алгоритма управления камерой

Из блок-схемы видно, что принцип работы алгоритма основан на проверке нажатия кнопки мыши и ее смещения, в результате чего выполняются определенные действия.

Алгоритм основан на использовании условного оператора if-else, который позволяет выполнять одни команды, когда поставленное условие истинно, и другие, когда условие ложно.

За проверку нажатия кнопки мыши отвечает функция «GetMouseButton(int n)», класса Input, где параметр функции n – целое число от 0 до 5 обозначающие номера кнопок мыши. Для удобства было принято использовать левую кнопку мыши, которой соответствует номер 0.

Поворот камеры производится за счет смещения ее положения относительно виртуального макета. За определение угла смещения мыши отвечает функция GetAxis(String str) класса Input, где параметр функции str – строка обозначающая ось вращения мыши и может принимать значения «Mouse X» или «MouseY».

Листинг скрипта управляющего камерой представлен в приложении.

Следующим шагом было написание скрипта Controller для управления руками виртуального макета с помощью джойстика. Блок-схема алгоритма управления руками изображена на рисунке 41.

Листинг данного скрипта представлен в приложении.

В данном скрипте также как и в предыдущем использовалась функция GetAxis(String str) класса Input, но в качестве параметров использовались «Vertical» «Horizontal», обозначающие использование изменения угла наклона флиппера на джойстике.

Для определения того, какой именно частью руки нужно управлять, использовался оператор switch. В качестве реализации использовалась функция case. С помощью данной функции возможен выбор действий посредством введения номера операции. В данном случае нужно будет сравнить одну переменную choise с различными значениями и выполнить разные участки кода в зависимости от того, каково значение. Именно это поможет сделать оператор switch.

Дескриптор, handle, описатель – число, которое задает номер какого-либо ресурса (блок памяти, файл, курсор, шрифт, окно и т.п.) при работе с ним. Дескриптор используется обычно при работе через некоторый интерфейс (API), причем смысл значения дескриптора скрыт за этим интерфейсом. Например, дескриптор блока памяти может означать номер ячейки в таблице адресов блоков памяти; поскольку пользователь API работает с дескриптором, а не с указателем, адрес размещения блока памяти может меняться, и это не скажется на работе с ним через дескриптор.

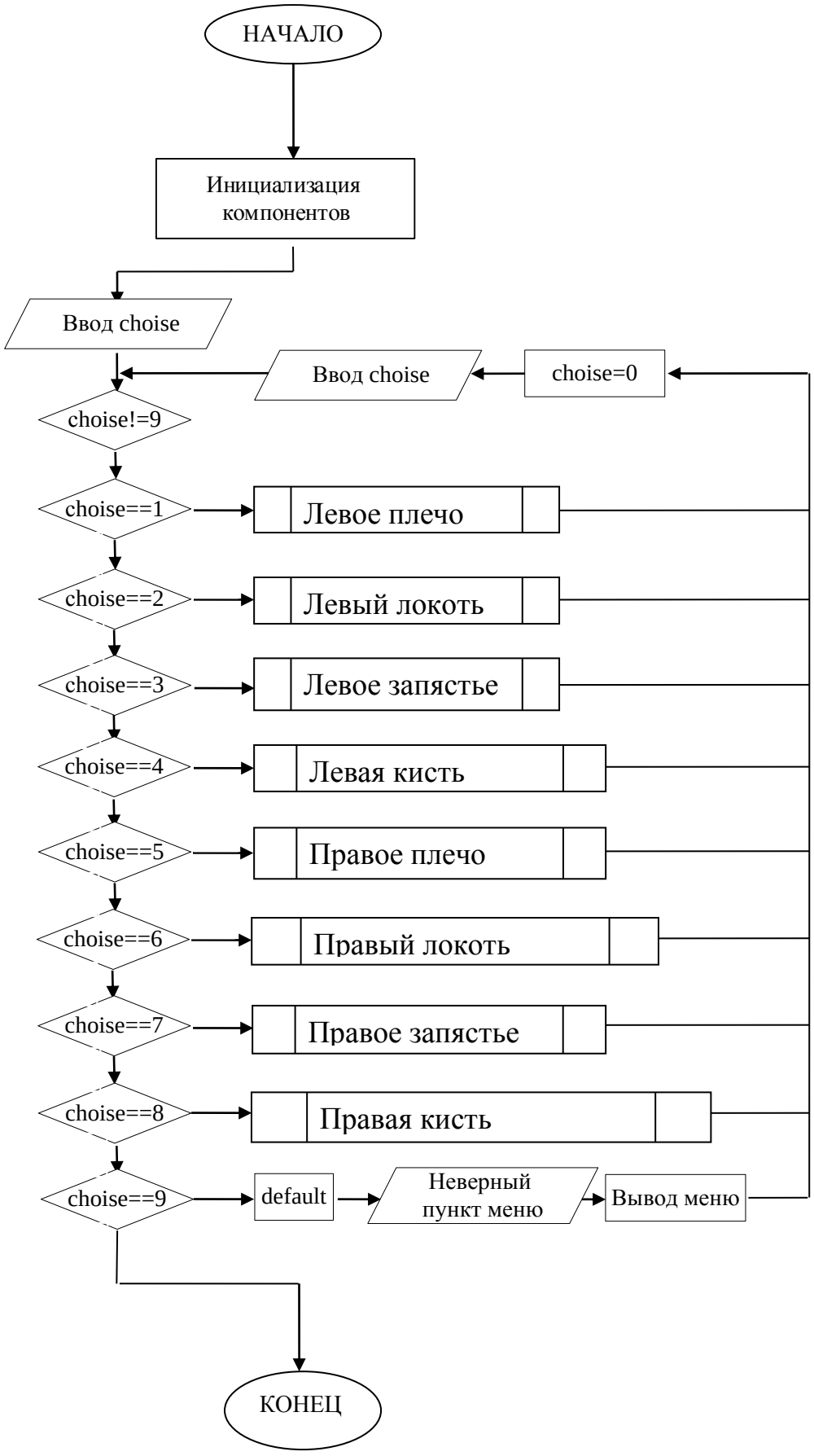

Рисунок 41 – Блок-схема алгоритма управления руками робота

4.4 Разработка графического интерфейса пользователя

На данном этапе работы был разработан графический интерфейс пользователя в Unity 3D. Интерфейс представляет собой два небольших меню, в каждом из которых расположены по четыре кнопки отвечающие за выбор элемента управления.

Методы для работы находятся в классе GUI и GUILayout, методы из этих классов нужно вызывать, только в функции OnGUI и всех вызываемых из неё. GUILayout более автоматизирован. В Unity3D так же, предусмотрены компоненты помогающие гибко настраивать отображения GUI, GUISkin и GUIStyle. GUISkin используется для глобального изменения отображения, а GUIStyle для более локального. Ещё есть GUIContent объект, вмещающий в себя, текст, картинку, и подсказку.

Для вывода данных кнопок был написан скрипт Buttons, листинг которого представлен в приложении. В нем использовалась функция Button(new Rect(int x1, int y1, int x2, int y2)), класса GUI, где x1, y1 – координаты верхнего левого угла кнопки, x2, y2 - координаты нижнего правого угла кнопки. Результат выполнения скрипта показан на рисунке 42.

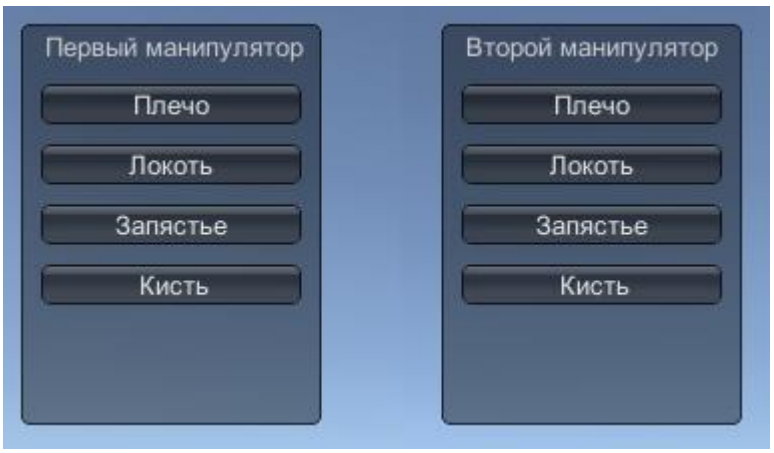

Рисунок 42 – Меню для разрабатываемого приложения

Также, для запуска функции записи данных, была разработана кнопка, которая изменяет свою текстуру после каждого нажатия. Цвет и надпись кнопки символизируют о том идет запись или нет. На рисунке 43.а показана кнопка до

нажатия, на рисунке 43.б после нажатия.

а) б)

Рисунок 43 – Кнопка отвечающая за запись данных

При нажатии на кнопку «Запись», программа начинает записывать в текстовый файл Servo.txt все последующую управляющие сигналы в виде матрицы описанной выше.

Запись в текстовый файл реализована при помощи функции WriteFile(). Функция WriteFile пишет данные в файл с места, обозначенного указателем позиции в файле. Эта функция предназначена и для синхронной, и для асинхронной операции.

Если функция WriteFile завершается успешно, величина возвращаемого значения - не ноль.

Если функция WriteFile завершается с ошибкой, величина возвращаемого значения - ноль. Чтобы получить дополнительные сведения об ошибке, вызовите GetLastError.

4.5 Реализация конвертера для связи виртуальной и физической модели

На данном этапе работы было разработано приложение Converter в среде разработки Microsoft Visual Studio 2013. Данная среда разработки является наиболее популярной среди разработчиков, являясь удобной и одной и самых лучших сред разработок.

Основной задачей конвертора управляющих сигналов, является перевод поданных на объект управления сигналов с джойстика в понятный для микроконтроллера Arduino код.

При нажатии кнопки «Запись» в разработанном графическом приложении производится запись данных в текстовый файл в виде матрицы. Разрабатываемый конвертор должен открыть файл, считать с него данные, конвертировать их

и записать при необходимости в COM-порт.

COM-порт, или последовательный порт, представляет собой двунаправленный последовательный интерфейс, который предназначен для обмена байтовыми данными. В первое время этот порт использовали для подключения терминала, а потом для модема и мыши. Сейчас его принято применять для подключения источника бесперебойного питания, а также для связи с аппаратными средствами обработки вычислительных систем встраиваемого типа.

Буквально 15 лет назад использовался способ подключения устройств к компьютеру посредством специального стандартного разъема, расположенного на задней панели системного блока с применением специального сериального кабеля RS-232. У этого способа имеется множество недостатков. Такой кабель, по современным меркам, предоставляет крайне низкую скорость передачи данных – примерно сотню килобит в секунду. Помимо того, когда производилось физическое соединение разъемов, необходимо было осуществлять выключение оборудования, а сами они крепились друг к другу при помощи винтов, обеспечивающих надежность, при этом их размеры отличались немалой величиной.

В связи с этим функции, которые должно выполнять приложение следующие: открытие файла, конвертация, запись в COM-порт. Для наглядности и упрощения написания программного кода программы была построена блок-схема отвечающая заданным требованиям.

На первом этапе производится открытие файла и чтение информации. При возникновении ошибки, программа сообщит об этом и предложит открыть другой файл.

Далее производится конвертация данных. Если конвертация прошла успешна, то программа сообщит об этом и даст возможность записать результат конвертации в COM-порт, иначе программа сообщит об ошибке.

На рисунке 44 изображена блок-схема алгоритма конвертации реализующая требуемые функции.

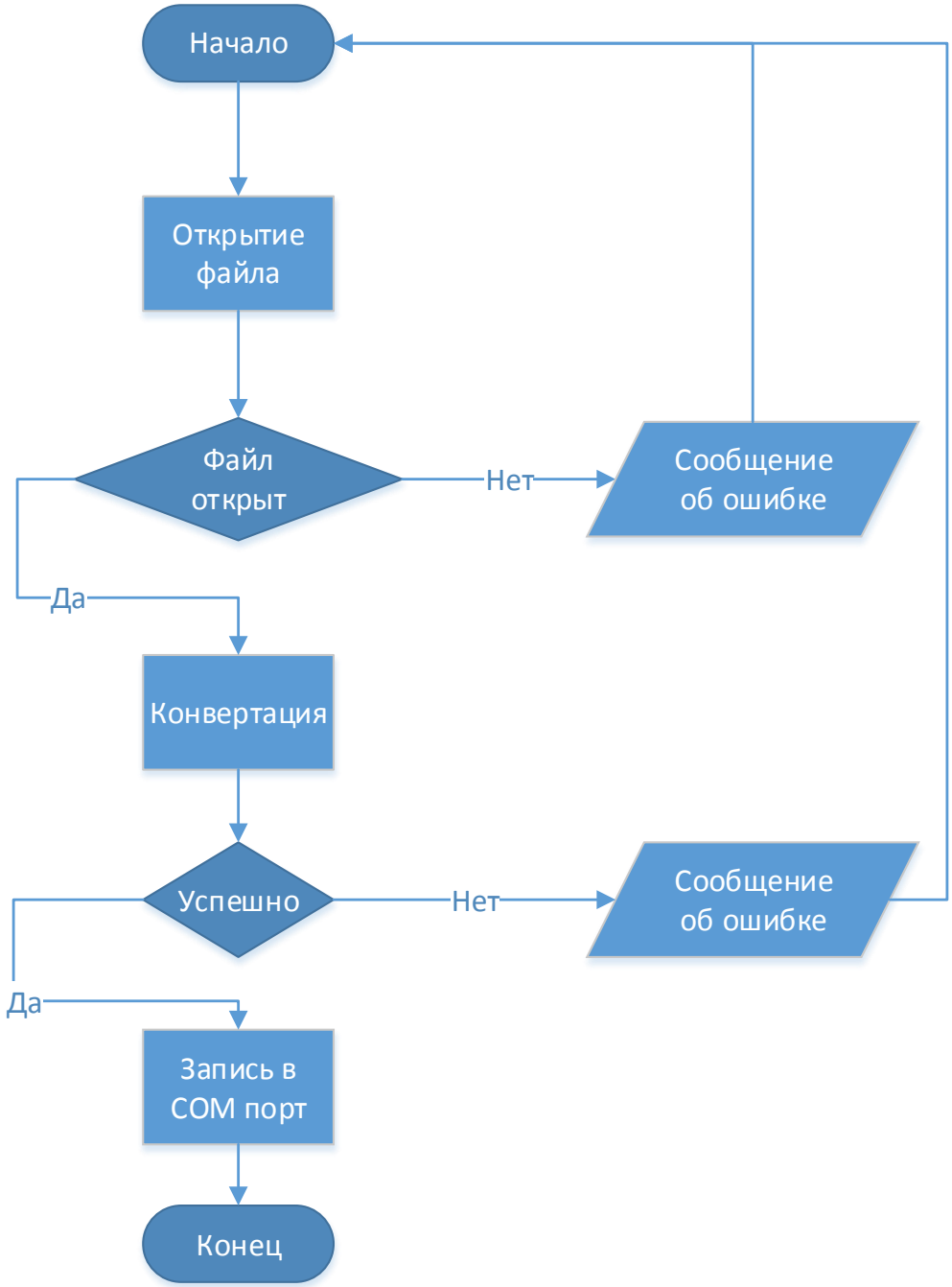

Рисунок 44 – Блок-схема алгоритма конвертации

Выбор языка программирования пал на Visual Basic, ввиду его простоты и гибкости. На первом этапе был разработан графический интерфейс пользователя, включающий два текстовых поля textbox и три кнопки, вызывающие определенные функции.

Первое текстовое поле отображает содержимое входного файла, сконфигурированного приложением, управляющим движением рук. Второе текстовое по-

ле отображает результат выполнения конвертации. Разработанный графический интерфейс представлен на рисунке 45.

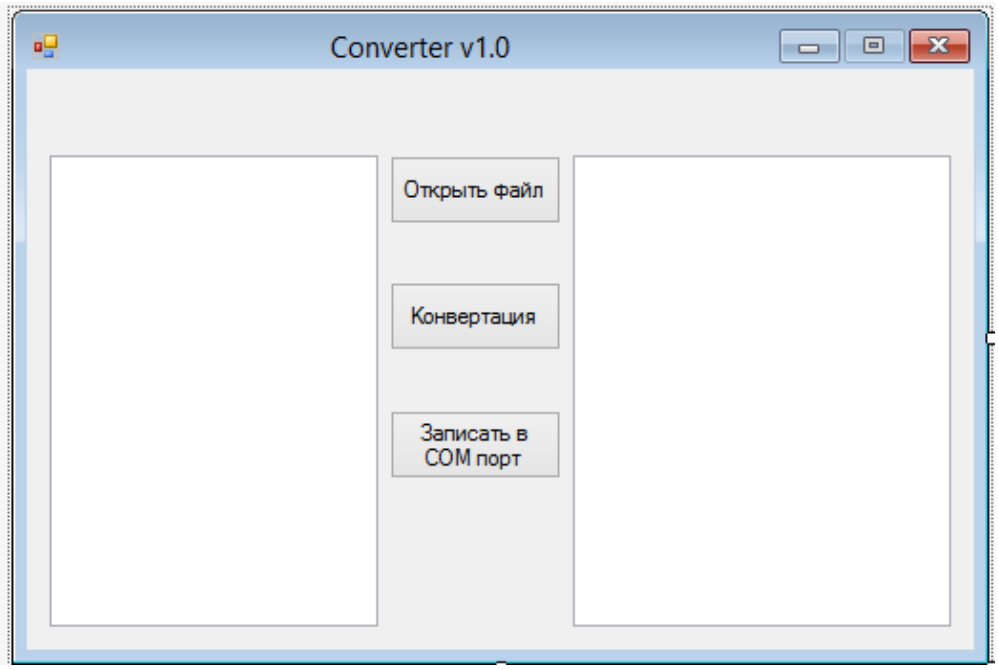

Рисунок 45 – Графический интерфейс программы Converter

Далее была реализована функция открытия файла. Данная функция была реализовано при помощи стандартного окна открытия файлов Windows. В Visual Basic это окно можно вызвать при помощи функции ShowDialog(), класса OpenFileDialog.

Класс OpenFileDialog позволяет пользователям просматривать папки личного компьютера или любого компьютера в сети, а также выбирать файлы, которые требуется открыть. Диалоговое окно возвращает путь и имя файла, который был выбран пользователем.

После того, как пользователь выбирает файл, который требуется открыть, существует два подхода к открытию файла. Если разработчик предпочитает работать с потоками файлов, можно создать экземпляр класса StreamReader, который был использован в данной работе. С другой стороны, для открытия выбранного файла можно использовать метод OpenFile.

Файловый поток – особая переменная, значением которой является файл.

Поток – понятие обозначающее на языке Visual Basic внутренний (логиче-

ский) файл. Это последовательность байтов передаваемых в процессе ввода, вывода и не зависящая от конкретного устройства с которым производится обмен информации. Для поддержки потоков в Visual Basic используется библиотека содержащая иерархию классов, построенную на основе класса ios.

Потоковая модель файлового ввода-вывода была популяризирована во многом благодаря операционной системе Unix. Большая функциональность современных операционных систем унаследовала потоки от Unix, а многие языки программирования унаследовали интерфейс файлового ввода-вывода языка.

Далее был открыт файловый поток для чтения при помощи функции StreamReader(String path), класса System.IO, где параметр path должен обозначать путь к файлу.

Функция StreamReader позволяет легко считывать весь текст или отдельные строки из текстового файла. Она возвращает значение тогда, когда выполнено одно из ниже перечисленных условий:

- операция чтения завершается на читающем конце канала;
- затребованное число байтов прочитано;
- или происходит ошибка.

Если функция StreamReader завершается успешно, величина возвращаемого значения - не ноль. Если функция StreamReader завершается с ошибкой, величина возвращаемого значения - ноль. Чтобы получить дополнительные сведения об ошибке, вызовите GetLastError.

После чтения файла необходимо конвертировать входные данные в кодировку ASCII. Данная задача была реализована при помощи функции Asc(String str), где параметром str является строка.

Функция записи конвертированных данных в COM-порт была реализована при помощи элемента SerialPort. Данный элемент добавляется на форму из списка элементов.

Далее для реализации функции записи данных в порт используется функция SerialPort.Write(String str), где параметром str является строка.

Листинг программы Converter представлен в приложении.

ЗАКЛЮЧЕНИЕ

В ходе работы был разработан виртуальный программируемый макет робота InMoov, который состоит из 3D модели и программного обеспечения реализующие взаимодействие с пользователем и физическим прототипом.

Результатом данной работы является экономия ресурсов и предупреждение возможных трудностей и ошибок при реализации реального прототипа. Были получены знания и опыт в области 3D моделирования и 3D печати, программирования интерфейсов взаимодействия между объектами и проектировании электронно-цифровых схем с использованием микроконтроллеров Arduino. Весь процесс работы был разбит на три основных этапа, а именно: исследование и описание предметной области, включающее описание структуры робота InMoov и его функционирования; проектирование информационного продукта, включающее определение и описание функциональных и обеспечивающих подсистем; разработку виртуального макета, включающую разработку 3D-модели робота InMoov, разработку интерфейса для управления контроллером, разработку графического интерфейса пользователя и реализацию конвертора для связи виртуальной и физической модели.

На заключительном этапе, был разработан программный продукт отвечающий всем требованиям необходимым для реализации поставленной цели.

Таким образом, поставленная в данной работе задача была выполнена полностью.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Прахов, А.В. 3D-моделирование и анимация. Руководство для начинающих / А.В. Прахов. - БВХ. : Изд-во БХВ-Петербург, 2009. – 266 с.

2 Джеймс Кронистер. Основы Blender. 4-е издание / Переводчики Азовцев Юрий и Корбут Юлия. Central Dauphin Hight School : Изд-во BlenderNation, 2011. – 200 с.

3 123D Circuits [Электронный ресурс] – Режим доступа: https://123dcircuits.io/ – 08.04.2016.

4 Петцольд, Ч. Программирование для Microsoft Windows на C# Введение / Ч. Петцольд. - М. : Лаборатория базовых знаний, 2010. – 278 с.

5 Hand and Forarm [Электронный ресурс] – Режим доступа: https://inmoov.fr/hand-and-foram/ – 10.04.2016.

6 Соловьев, М. М. Blender. Мир трехмерной графики / М. М. Соловьев. – М. : Солон-Пресс, 2010. – 478 с.

7 Миловская, О. А. Самоучитель Blender / О. А. Миловская. – СПб. : Питер, 2010. – 564 с.

8 Резников, Ф. А. Blender. Установка, настройка и результативная работа / Ф. А. Резников. – М. : Солон-Пресс, 2011. – 368 с.

9 Рисов, С. Анимация персонажей в Blender / С. Рисов. – СПб. : Питер, 2012. – 312 с.

10 Хокинг, Д. Unity в действии. Мультиплатформенная разработка на C# / Д. Хокинг. – СПб. : Питер, 2015. – 336 с.

11 Торн, А. Основы анимации в Unity / А. Торн. – М. : ДМК-Пресс, 2014. – 176 с.

12 Ламмерс, К. Шейдеры и эффекты в Unity. Книга рецептов / К. Ламмерс. – М. : ДМК-Пресс, 2014. – 274 с.

13 Голощапов, А. Microsoft Visual Studio 2010 / А. Голощапов. – М. : ДМК-Пресс, 2011. – 544 с.

14 Левинсон, Д. Тестирование ПО с помощью Visual Studio 2010 / Д. Левинсон. – М. : ЭКОМ Паблишерз, 2012. – 314 с.

15 Липаев В.В. Тестирование программ / В.В Липаев. – М.: Радио и связь, 1986. – 292 с.

ПРИЛОЖЕНИЕ А

Листинг программного обеспечения

```
//Листинг скрипта CameraRotateAround
using UnityEngine;
using System.Collections;

public class CameraRotateAround : MonoBehaviour
{

    public Transform target;
    public Vector3 offset;
    public float sensitivity = 3f;
    public float limit = 80f;
    private float Y;
    private float X;

    void Start()
    {
        limit = Mathf.Abs(limit);
        if (limit > 90) limit = 90;
        transform.position = target.position + offset;
    }

    void Update()
    {
        if (Input.GetMouseButton(0))
        {
            X = transform.localEulerAngles.y + Input.GetAxis("Mouse X") * sensi-
tivity;

Y += Input.GetAxis("Mouse Y") * sensitivity;
            Y = Mathf.Clamp(Y, -limit, limit);

            transform.localEulerAngles = new Vector3(0, X, 0);
        }
            Vector3 position = transform.localRotation * offset + target.position;
            transform.position = position;

    }
}
```

```
    public Transform right_index_mid;
    public Transform right_index_tip;
    public Transform right_middle_base;
    public Transform right_middle_mid;
    public Transform right_middle_tip;
    public Transform right_palm_pinky1;
    public Transform right_pinky_base;
    public Transform right_pinky_mid;
    public Transform right_pinky_tip;
    public Transform right_palm_ring1;
    public Transform right_ring_base;
    public Transform right_ring_mid;
    public Transform right_ring_tip;
    public Transform right_palm_thumb2;
    public Transform right_thumb_base;
    public Transform right_thumb_tip;
    public Transform left_palm_pinky1;
    public Transform left_pinky_base;
    public Transform left_pinky_mid;
    public Transform left_pinky_tip;
    public Transform left_palm_ring1;
public Transform left_ring_base;
    public Transform left_ring_mid;
    public Transform left_ring_tip;
    public Transform left_palm_thumb2;
    public Transform left_thumb_base;
    public Transform left_thumb_tip;
    public float speedMove = 30f;
    public float speedRotate = 60f;
    public float temp=0;
    public float temp1 = 0;
    // Use this for initialization
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        float horizontal = Input.GetAxis("Horizontal");
        float vertical = Input.GetAxis("Vertical");
        float horizontal1 = Input.GetAxis("Horizontal1");
        float vertical1 = Input.GetAxis("Vertical1");
```

```
        case 1: if (horizontal != 0 || vertical != 0)
                    {
                        Vector3 rotate1 = new Vector3(vertical, 0f, 0f);
                        Vector3 rotate2 = new Vector3(horizontal, 0f, 0f);
                        right_shoulder1.rotation = Quaterni-
on.Slerp(right_shoulder1.rotation, right_shoulder1.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
                        right_biceps0.rotation = Quaternion.Slerp(right_biceps0.rotation,
right_biceps0.rotation * Quaternion.Euler(rotate2), speedRotate * Time.deltaTime);
                    }
                        break;
                case 2: if (horizontal != 0 || vertical != 0)
                        {
                            Vector3 rotate1 = new Vector3(0f, 0f, vertical);
                            right_forearm0.rotation = Quaterni-
on.Slerp(right_forearm0.rotation, right_forearm0.rotation * Quaternion.Euler(rotate1),
speedRotate * Time.deltaTime);
                        }
                        break;
                case 3: if (vertical != 0)
                        {
                            Vector3 rotate1 = new Vector3(0f, vertical, 0f);
                            right_wrist3.rotation = Quaternion.Slerp(right_wrist3.rotation,
right_wrist3.rotation * Quaternion.Euler(-rotate1), speedRotate * Time.deltaTime);
                        }
                        Break;
        case 4: if (vertical != 0)
                        {
                            Vector3 rotate1 = new Vector3(0f, vertical, 0f);
                            if (temp <= 120 && temp >= 0)
                            {
                                right_index_base.rotation = Quaterni-
on.Slerp(right_index_base.rotation, right_index_base.rotation * Quaternion.Euler(-
rotate1), speedRotate * Time.deltaTime);
                                right_index_mid.rotation = Quaterni-
on.Slerp(right_index_mid.rotation, right_index_mid.rotation * Quaternion.Euler(-
rotate1), speedRotate * Time.deltaTime);
                                right_middle_mid.rotation = Quaterni-
on.Slerp(right_middle_mid.rotation, right_middle_mid.rotation * Quaternion.Euler(-
rotate1), speedRotate * Time.deltaTime);
        .Slerp(right_pinky_base.rotation, right_pinky_base.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
```

```
        on.Euler(rotate1), speedRotate * Time.deltaTime);
                        right_pinky_tip.rotation = Quaterni-
on.Slerp(right_pinky_tip.rotation, right_pinky_tip.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
                        right_ring_base.rotation = Quaterni-
on.Slerp(right_ring_base.rotation, right_ring_base.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
                        right_ring_mid.rotation = Quaterni-
on.Slerp(right_ring_mid.rotation, right_ring_mid.rotation * Quaternion.Euler(rotate1),
speedRotate * Time.deltaTime);
                        right_ring_tip.rotation = Quaterni-
on.Slerp(right_ring_tip.rotation, right_ring_tip.rotation * Quaternion.Euler(rotate1),
speedRotate * Time.deltaTime);
                        right_thumb_base.rotation = Quaterni-
on.Slerp(right_thumb_base.rotation, right_thumb_base.rotation * Quaternion.Euler(-
rotate1), speedRotate * Time.deltaTime);
                        right_thumb_tip.rotation = Quaterni-
on.Slerp(right_thumb_tip.rotation, right_thumb_tip.rotation * Quaternion.Euler(-
rotate1), speedRotate * Time.deltaTime);
                    }
                    if (temp<=60 && temp>=0){
                        right_palm_thumb2.rotation = Quaterni-
on.Slerp(right_palm_thumb2.rotation, right_palm_thumb2.rotation * Quaterni-
on.Euler(-rotate1), speedRotate * Time.deltaTime);
        Debug.LogError(temp);
                    }
                    if (temp <= 20 && temp >= 0)
                    {
                        right_palm_ring1.rotation = Quaterni-
on.Slerp(right_palm_ring1.rotation, right_palm_ring1.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
                    }
                    if (temp <= 60 && temp >= 0)
                    {
                        right_palm_pinky1.rotation = Quaterni-
on.Slerp(right_palm_pinky1.rotation, right_palm_pinky1.rotation * Quaterni-
on.Euler(rotate1), speedRotate * Time.deltaTime);
                    }
                    temp += 1 * rotate1.y;
                }
                break;
            }
```

```
        i01_leftArm_rotate_wrong.rotation = Quaterni-
on.Slerp(i01_leftArm_rotate_wrong.rotation, i01_leftArm_rotate_wrong.rotation *
Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                    left_shoulder3.rotation = Quaternion.Slerp(left_shoulder3.rotation,
left_shoulder3.rotation * Quaternion.Euler(rotate4), speedRotate * Time.deltaTime);
                }
                break;
            case 2: if (horizontal1 != 0 || vertical1 != 0)
                {
                    Vector3 rotate3 = new Vector3(0f, 0f, vertical1);
                    left_forearm0.rotation = Quaternion.Slerp(left_forearm0.rotation,
left_forearm0.rotation * Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                }
                break;
            case 3: if (vertical1 != 0)
                {
                    Vector3 rotate3 = new Vector3(0f, vertical1, 0f);
                    left_wrist3.rotation = Quaternion.Slerp(left_wrist3.rotation,
left_wrist3.rotation * Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                }
                break;
            case 4: if (vertical1 != 0)
                {
                    Vector3 rotate3 = new Vector3(0f, vertical1, 0f);
                    Vector3 rotate4 = new Vector3(0f, 0f, vertical1);
                    if (temp1 <= 120 && temp1 >= 0)
                    {
      left_index_base.rotation = Quaternion.Slerp(left_index_base.rotation,
left_index_base.rotation * Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_index_mid.rotation = Quaterni-
on.Slerp(left_index_mid.rotation, left_index_mid.rotation * Quaternion.Euler(rotate3),
speedRotate * Time.deltaTime);
                        left_middle_mid.rotation = Quaterni-
on.Slerp(left_middle_mid.rotation, left_middle_mid.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_middle_tip.rotation = Quaterni-
on.Slerp(left_middle_tip.rotation, left_middle_tip.rotation * Quaternion.Euler(rotate3),
speedRotate * Time.deltaTime);
                        left_pinky_base.rotation = Quaterni-
on.Slerp(left_pinky_base.rotation, left_pinky_base.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_pinky_mid.rotation = Quaterni
```

```
        on.Slerp(left_pinky_mid.rotation, left_pinky_mid.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_pinky_tip.rotation = Quaterni-
on.Slerp(left_pinky_tip.rotation, left_pinky_tip.rotation * Quaternion.Euler(rotate3),
speedRotate * Time.deltaTime);
                        left_ring_base.rotation = Quaterni
        on.Slerp(left_ring_base.rotation, left_ring_base.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_ring_mid.rotation = Quaternion.Slerp(left_ring_mid.rotation,
left_ring_mid.rotation * Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_ring_tip.rotation = Quaternion.Slerp(left_ring_tip.rotation,
left_ring_tip.rotation * Quaternion.Euler(rotate3), speedRotate * Time.deltaTime);
                        left_thumb_tip.rotation = Quaterni-
on.Slerp(left_thumb_tip.rotation, left_thumb_tip.rotation * Quaternion.Euler(-rotate4),
speedRotate * Time.deltaTime);
                    }
                    if (temp1 <= 60 && temp1 >= 0)
                    {
                        left_palm_thumb2.rotation = Quaterni-
on.Slerp(left_palm_thumb2.rotation, left_palm_thumb2.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                        Debug.LogError (temp1);
                    }
                    if (temp1 <= 20 && temp1 >= 0)
                    {
                        left_palm_ring1.rotation = Quaterni-
on.Slerp(left_palm_ring1.rotation, left_palm_ring1.rotation * Quaterni
        on.Euler(rotate3), speedRotate * Time.deltaTime);
                    }
                    if (temp1 <= 60 && temp1 >= 0)
                    {
                        left_palm_pinky1.rotation = Quaterni-
on.Slerp(left_palm_pinky1.rotation, left_palm_pinky1.rotation * Quaterni-
on.Euler(rotate3), speedRotate * Time.deltaTime);
                    }
                    temp1 -= 1 * rotate3.y;
                }
                break;
            }
        }
```