МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»
Институт космической инженерии и телекоммуникаций
Кафедра электроники и робототехники

«ДОПУЩЕН К ЗАЩИТЕ»
Зав.кафедрой доц., Чигамбаев Т.О.
_____ «___» _____20__г.
(подпись)

ДИПЛОМНАЯ РАБОТА

На тему: «Разработка интеллектуального робота с применением нейронной сети»
Специальность «5В071600 – Приборостроение»
Выполнил Якубов Фархат Бахтиярович
Научный руководитель доцент Байкенов Б.С.

_____ «20» _____05_____ 2020 г.
(подпись)

Консультанты:
по экономической части: ст.преп., Боканова Г.Ш.

_____ «___»
_____20__г.
(подпись)

по безопасности жизнедеятельности: к.т.н., доц., Приходько Н.Г.

_____ «___» _____20__г.
(подпись)

Нормоконтролер: ст. преп., Фазылова А.Р.

_____ «20» _____05_____ 2020 г.
(подпись)

Рецензент: : к.т.н. Утебаев Р.М.
_____ «___» _____20__г.
(подпись)

Алматы 2020

Некоммерческое акционерное общество
«Алматинский университет энергетики и связи им. Гумарбека Даукеева»

Институт “Космическая инженерия и телекоммуникации”
Кафедра “Электроника и робототехника”
Специальность “5В071600 - Приборостроение”

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Якубову Ф.Б.

Тема проекта “Разработка интеллектуального робота с применением нейронной сети”

Утверждена приказом по университету № ___ от «___» _____ 2020 г.

Срок сдачи законченного проекта «___» _____ 2020 г.

Исходные данные к работе (требуемые параметры результатов проектирования и исходные данные объекта):

1) аппаратная часть: Raspberry Pi 4 Model B, Nucleo L031K6, The Raspberry Pi Camera Module;

2) программные компоненты: библиотеки языка программирования python tensorflow, opencv, numpy, requests, socket, pyserial, библиотека языка программирования СИ HAL.

3) электрическая часть: аккумулятор формата 18650 Sony VTC5, преобразователь напряжения XL4015E1 DC-DC, балансировочная плата HX-2S-JH20 v1.0

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломной работы:

1) обозначение основных определений и краткое введение в тему дипломной работы;

2) обзор на компоненты, используемые в проекте и обоснование выбора этих компонентов;

3) проектирование управляющего устройства;

4) сборка механической, электрической и логической части прототипа;

5) написание ряда программ реализующих связь компонентов, обработку информации и алгоритм обучения;

6) расчет безопасности жизнедеятельности (Анализ вредных факторов на производстве, расчет вредных веществ выделяющихся при пайке изделий и расчет заземления)

7) составление бизнес-плана (резюме, маркетинговый план и финансовый план).

Перечень графического материала (с точным указанием обязательных чертежей): в данной работе содержится 59 рисунков и 15 таблиц.

Основная рекомендуемая литература:

1 Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

2 Г.Ш. Боканова. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В071900 – Радиотехника, электроника и телекоммуникаций. – Алматы: АУЭС, 2020 – 26 с

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Технологическая часть	Байкенов Б.С.	15.03.2020	
Конструкторская часть	Байкенов Б.С.	20.04.2020	
Реализация проекта	Байкенов Б.С.	02.06.2020	
Бизнес-план	Боканова Г.Ш.	10.05.2020	
Безопасность жизнедеятельности	Приходько Н.Г.	20.05.2020	

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Обозначение основных определений и краткое введение в тему дипломной работы	15.03.2020	
Обзор на компоненты, используемые в проекте и обоснование выбора этих компонентов	02.04.2020	
Проектирование управляющего устройства	20.04.2020	
Сборка механической, электрической и логической части прототипа	25.05.2020	
Написание ряда программ реализующих связь компонентов, обработку информации и алгоритм обучения	02.06.2020	
Расчет безопасности жизнедеятельности	20.05.2020	
Составление бизнес-плана	10.05.2020	

Дата выдачи задания «___» _____ 2020 г.

Заведующий кафедрой _____ (Чигамбаев Т.О.)
(подпись)

Руководитель _____ (Байкенов Б.С.)
(подпись)

Задание принял к
исполнению студент _____ (Якубов Ф.Б.)
(подпись)

Аңдатпа

Дипломдық жұмыстың негізгі мақсаты - жасанды нейрондық желілер мен машиналық оқыту технологиясын қолдана отырып, мобильді роботты дамыту. Жобаның тақырыбы жұмыстың екі аспектісіне бағытталған, бұл мобильді роботтың қозғалмалы бөлігін жобалау және нейрондық желінің математикалық моделін және оқыту алгоритмін құру.

Аннотация

Главная цель дипломной работы является разработка мобильного робота, с применением технологий искусственных нейронных сетей и машинного обучения. Тематика проекта сфокусирована на двух аспектах дипломной работы, это проектирование движущейся части мобильного робота и построение математической модели нейронной сети и алгоритма обучения.

Summary

The main goal of the graduation project is the development of a mobile robot, using the technology of artificial neural networks and machine learning. The subject of the project is focused on two aspects of the work, this is the design of the moving part of the mobile robot and the construction of a mathematical model of the neural network and the learning algorithm.

Содержание

Введение

Данная дипломная работа ставит целью полноценно обучить мобильного робота ходьбе. Чтобы достичь данной цели будет использован ряд технологий и инструментов, применяемых в реальной производственной разработке электронных устройств. Будет написан комплекс программных решений задач связи и обработки.

Основной темой дипломной работы выступают технологии машинного обучения и искусственной нейронной сети. Большая часть программного кода будет написана на языке программирования python с использованием таких библиотек как numpy, tensorflow, openCV. В процессе написания программы понадобятся навыки хорошего понимания основ ООП, а именно такие понятия как класс или объект. Также необходимы знания основ теоретической части искусственных нейронных сетей и машинного обучения.

Так или иначе в работе будут задействованы элементы механики, автономного электропитания, программирования микроконтроллеров, установки связи посредством wi-fi сети.

Программирование микроконтроллера требует продвинутых знаний языка программирования Си, а именно понимания структур, понимания работы с памятью микроконтроллера, работы с регистрами разной периферии, в частности регистрами таймеров общего назначения и указателями. Также программирование микроконтроллеров требует опыта работы с микроконтроллерами STM32, а именно способами взаимодействия с ними.

Для корректной работы с электроникой будут применены навыки паяния и монтажа элементов схемотехники на монтажную плату. Каждый потребитель большого тока будет предварительно проверен лабораторным источником питания. Монтаж проводов будет производится с учетом необходимости в качественном контакте модулей между собой в условиях подвижного мобильного робота. Также так как применяемый аккумулятор способен выдавать большие токи при коротком замыкании, будут произведены все необходимые меры безопасности, такие как изоляция электродов и применение специализированных контактов.

1 Технологическая часть

1.1 Понятие “робот” и “робототехника”

В данной дипломной работе изучается вопрос разработки такого устройства как робот. Поэтому для подготовки перед самим процессом разработки, стоит составить обзор на тему робототехники в целом. В обзорной части работы будут определены такие понятия как “робот” и “робототехника”, рассмотрены процесс разработки робота, проведен обзор на основные технологии, которые применяются в ходе разработки

Понятие робототехника тесно связано с понятием робот, которое в свою очередь является устройством выполняющая механические задачи без вмешательства человека в процесс.

Роботы в зависимости от способности передвигаться делятся на мобильные и манипуляционные [1]. Манипуляционные роботы установлены в одном месте и выполняют действия манипулятором, имеющим несколько степеней подвижности (рисунок 1.1).

Рисунок 1.1 – Робот, произведенный компанией KUKA

Такие роботы применяются, к примеру, в машиностроении, где от робота не требуется способность передвигаться, но требуется очень точное и быстрое действие, несколько манипуляторов способны собрать корпус автомобиля быстрее и точнее в отличии от бригады людей. Также важной частью манипуляционного робота является перепрограммируемое устройство управления. Устройства управления бывают автоматические и полуавтоматические, управляемые оператором через пульт или копируя движения оператора.

Мобильные роботы оснащены движущимся шасси, что делает их способными перемещаться в пространстве.

В роли приводов таких роботов выступает в большинстве случаев электромотор. На колесных и гусеничных роботов устанавливаются коллекторные моторы постоянного тока, установив между колесом редуктор (рисунок 1.2). Колесные роботы просты в конструировании, поэтому являются самыми популярным видом роботов. Их минусом можно назвать неспособность передвигаться по сложным ландшафтам, тем не менее такую проблему можно частично решить, установив дополнительную степень свободы колеса перпендикулярно к валу мотора.

Рисунок 1.2 – Колесный робот спроектированные для доставки товаров

Шагающие роботы гораздо более сложные в конструкции. Такие роботы передвигаются с помощью синхронного движения опорных конструкций. Опорные конструкции называют ногами робота, а место сгиба в опорной конструкции называют суставом. Как правило количество суставов определяет количество степеней свободы. Более продвинутые модели строят двигательную часть по частям, то есть используют мощный мотор отдельно от сустава, используя приводной ремень, для передачи крутящего момента (рисунок 1.3). В упрощенных моделях применяются сервопривода в месте сустава, что сильно упрощает конструкцию, но ограничивает модель в мощности. Несмотря на это конструкция шагающего робота остается очень сложной, т.к. движение с помощью ног сложный процесс, ведь роботу приходится управлять большим количеством приводов, удерживая равновесие при ходьбе. Для способности робота ходить ровно, используются алгоритмы обучения роботов, которые методом проб и ошибок учат нейросеть робота ходить ровно, не падая. Моделированием обучающих алгоритмов занимается раздел машинного обучения, который называется обучение с подкреплением (reinforcement learning). Колесные и шагающие роботы чаще всех остальных применяются для помощи людям и в бытовых целях, тем не менее существует известный пример шагающего военного робота BigDog от компании Boston Dynamics, транспортирующего на себе до 125 килограмм груза [2].

Рисунок 1.3 – Робот Spot mini от компании Boston Dynamics

Летающих роботов называют беспилотными летательными аппаратами (БПЛА). По способу передвижения они различаются на два типа – парящие и вертолетного типа. Парящие БПЛА летают по тому же принципу что и самолеты, то есть получают разгон для взлета в начале пути и в зависимости модификации продолжают полет за счет реактивной тяги двигателей. БПЛА вертолетного типа для подъема используют бесколлекторные электродвигатели (рисунок 1.4). Такой тип больше подходит для городских условий, т.к. не требуют разгона перед взлетом. БПЛА вертолетного типа проигрывают в скорости и времени полета без подзарядки парящим БПЛА, компенсируя маневренностью и способностью летать вертикально.

Робототехника – прикладная наука, которая изучает способы проектирования, разработки и применения роботов. Для разработки роботов робототехника в свою очередь применяет множество технологии из большого количества направлений в программировании, математики, вычислительной техники, механики, телекоммуникации.

Для демонстрации обширного круга знаний и навыков, необходимых при разработке робота, приведем процесс разработки робота. К примеру, чтобы робот мог самостоятельно действовать, требуется большое количество преобразований информации из наблюдаемой реальности в сигналы, приходящие на исполнительные механизмы, что в свою очередь требует определенной вычислительной мощности и алгоритмов.

Первичными преобразователями может являться множество датчиков, видеокамера, ToF камера. Далее информация может обрабатываться на вычислительной машине самого робота или же отправляются для обработки на сервер, что свою очередь требует вычислительную технику и ПО для запуска сервера, к примеру компьютер Raspberry Pi.

Рисунок 1.4 – Дрон Mavic Pro от компании DJI

Если робот применяет камеру как выходной сигнал, для обработки сигнала применяются обученные искусственные нейронные сети, алгоритмы нахождения объектов или ускорители FPGA на входе с камеры. В простых задачах полученную информацию обработает несложный алгоритм, который передаст ее дальше на исполнительный механизм. Но если задача требует от робота более продвинутых самостоятельных решений, к примеру действия робота должны быть основаны на предыдущем опыте, то строится модель искусственной нейронной сети, которая будет обучатся по мере получения роботом опыта исследования среды, в которой он находится. Дальше обработанная информация поступает на низкий уровень к исполнительным механизмам. Средством доставки этой информации может быть, к примеру, порт GPIO на Raspberry Pi или микроконтроллер. Исполнительный механизм требуется обеспечить источником электропитания, что требует понимания силовой электроники. Также источник питания может быть автономным, а значит нужен расчет времени автономной работы и оптимальная работа с аккумулятором. Корпус проектируется с учетом подвижности робота. Многие детали печатаются на 3-ем мерном принтере, то есть обязателен навык проектирования 3-х мерных моделей и теоретические знания, для обеспечения тех или иных характеристик стойкости корпуса. Сам исполняющий механизм должен быть спроектирована с расчетом на определенные нагрузки, с учетом специфической формы робота, надежным и долговечным. К этому списку также можно добавить навыки проектирования различных систем, которые будут необходимы роботу в полевых условиях, такие как системы подзарядки аккумулятора, системы сигнализации при поломки робота и т.д.

Каждую приведенную технологию робототехника рассматривает с точки зрения применения технологии в целях создания более простых способов разработки роботов. К примеру, нейронные сети сильно упрощают обработку информации с большим количеством входных данных при этом имеющую некий шаблон. В видеопотоке, приходящему с

видеокамеры, обрабатываются данные каждого пикселя в каждом цветовом канале, а также учитывается последовательность всех пикселей, что создает шаблон. При стандартной обработке видеопотока задача найти на кадре объект любого класса невозможна без применения способа абстрагировать входные данные в некоторую модель, которая будет описывать шаблон. Поэтому популярность и последующее развитие искусственный нейронных сетей, дало возможность развитию машинного зрения, что облегчило разработку зрительных систем для роботов. Искусственные нейронные сети пригодились не только при обработке видео с камер, но и при проектировании модели поведения робота. Модель поведения описывает шаблон действий робота, в тех или иных обстоятельствах влияющих на робота из вне. Данную модель обучают, что также приводит к упрощению разработки роботов, которые способны действовать самостоятельно.

Данная дипломная работа сосредоточена на таких составных частях разработки робота как обработка приходящей с камеры информации и написание программы поведения робота, поэтому большинство описываемых технологий будут тесно связаны с программированием и математикой. В следующей подглаве будет проведен обзор основных технологий, которые будут применены в практической части проекта.

1.2 Обзор применяемых в проекте технологий

Основной рассматриваемой в дипломной работе технологией является искусственная нейронная сеть. Искусственная нейронная сеть – это математическая модель, построенная по принципам естественной нейронной сети человеческого мозга. Данная технология является продуктом технической биомимикрии, копированием принципов живой природы в технологических процессах. Изначально первая модель нейронной сети была построена исследователями Уоренном Мак-Каллокам и Уолтером Питтсом в проекте по созданию математической модели человеческого мозга [3]. Модель нейронной сети сама по себе представляла собой структуру с входным слоем, выходным слоем и скрытыми слоями между входным и выходными слоями, где каждый слой состоит из определенного количества нейронов. В дальнейшем развитие вычислительной техники и появление обучающих алгоритмов сделали возможным применение нейронных сетей в практических задачах. Главной причиной популярности применения нейронных сетей стала их структура, в которой на входной слой подается множество из данных и скрытые слои, представляющие собой часть сети, хранящие в себе шаблон, по которому сеть способна реагировать на выходные данные с определенной закономерностью. Как уже говорилось, это позволило применить нейронные сети в обработке фотографий, звука, и в любой другой информации, которую можно описать один множеством. На сегодняшний день нейронная сеть имеет много видов и применяется далеко за пределами классических задач по классификации. Выделяются такие виды нейронных сетей как нейронные сети прямого распространения, рекуррентные

нейронные сети, автокодировщики, сверточные нейронные сети.

Нейронные сети прямого распространения – это классическая нейронная сеть, прямой потомок перцептрона. Такая сеть используется не часто в отличии от глубокой нейронной сети прямого распространения. Различие этих двух сетей только в количестве скрытых слоев и нейронов, в глубоких нейронных сетях их больше (рисунок 1.5). На рисунке 5 желтые круги – это входные нейроны, зеленые – скрытые нейроны, красные – выходные нейроны.

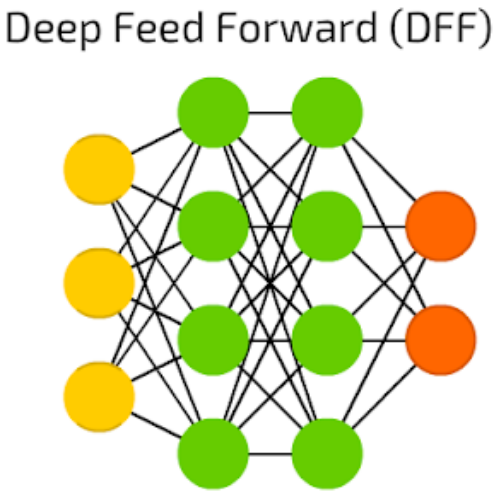

Рисунок 1.5 – Иллюстрация структуры глубокой нейронной сети

Рекуррентные нейронные сети – это нейронная сеть, в которой связь между нейронами направленная. Такая связь между нейронами дает способность нейронной сети запоминать последовательность в событиях. По данному принципу рекуррентные нейронные сети обрабатывают объекты, разбитые на части. Их используются в задачах распознавания текста и речи. На рисунке 1.6 показаны схема рекуррентной нейронной сети, где синие нейронный рекуррентные, то есть имеют способность к связи с предыдущими слоями.

Также важным подвидом рекуррентной сети является LSTM-сеть. Данная сеть также способна к памяти, при этом умеет запоминать значения на долгой и короткий промежуток времени.

Автокодировщики – это нейронные сети, у которых количество нейронов в скрытых слоях меньше, чем на входных и выходных, при этом данный вид нейронной сети симметричен, то есть количество нейронов в выходном слою равен количество нейронов в выходном.

Автокодировщики очень схожи с нейронными сетями прямого распространения (рисунок 1.7). Они применяются для сжатия данных. Также автокодировщики используются для удаления шума с изображений и аудио. Для этого на входной слой подаются зашумленные данные. После прохода данных по сети вычисляется ошибка и сравнивается с выходными данными.

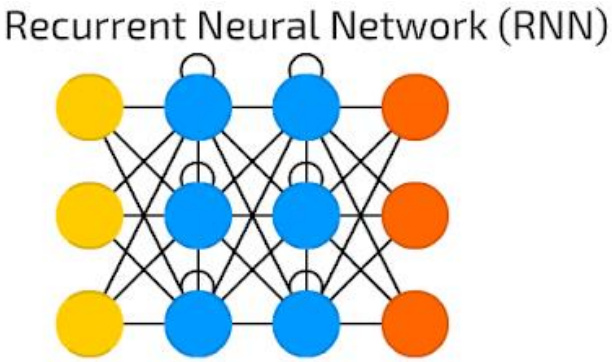

Рисунок 1.6 – Иллюстрация структуры глубокой нейронной сети

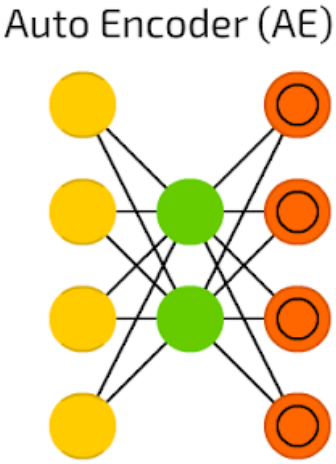

Рисунок 1.7 – Иллюстрация структуры автокодировщика

Особенным видом нейронных сетей является сверточные нейронные сети. Данные сети сильно отличаются от других видов нейронных сетей. Данный вид нейронных сетей использует не привычный метод приема данных, получая на вход всё множество, а использует так называемый “сканер”, который обработает определенный участок данных, например часть изображения размером в 20 на 20 пикселей. После обработки сканер смещается на один пиксель и продолжает то же самое. Такой принцип был скопирован из природы, а именно с зрительной коры, где были открыты простые клетки, реагирующие на прямые лучи света, и сложные клетки, реагирующие на определенную последовательность активаций простых клеток. Также следует заметить, что входные нейроны не соединены со всеми нейронами следующего слоя, что создает каналы данных. Далее в сверточной нейронной сети входные данные проходят по сверточным слоям, которые с приростом слоев сжимаются все больше. В конец сверточной нейронной сети ставится обычная нейронная сеть прямого распространения (рисунок 1.8). В основном сверточные нейронные сети применяются в распознавании образов, например в изображениях. К такому виду задачи отлично подходит задача классификации объектов на изображении.

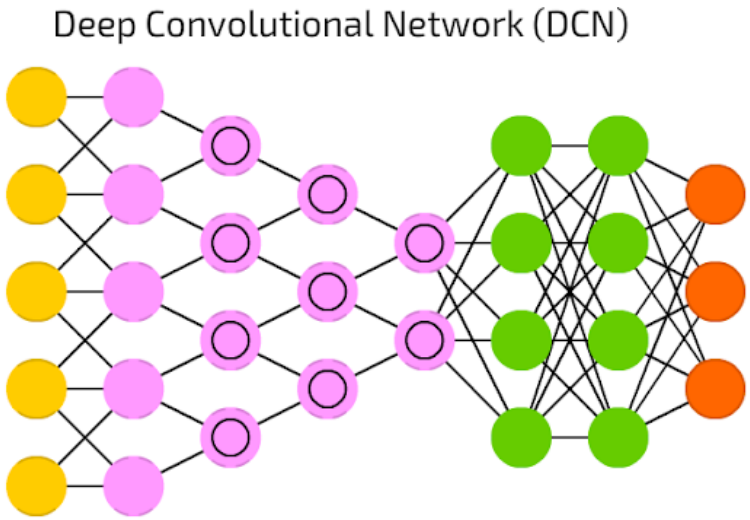

Рисунок 1.8 – Иллюстрация структуры сверточной нейронной сети

Но нейронные сети сами по себе не способны выдавать требуемый “осмысленный” результат. Подробно принцип работы нейронной сети будет рассмотрен в аналитической главе, пока достаточно знать, что нейронная сеть имеет в собственной структуре некие коэффициенты, которые и дают нейронным сетям способность реагировать на входные данные “осмысленно”. Подбором этих коэффициентов, которые называются весами, занимается технология машинного обучения.

Глобально алгоритмы по способу обучения классифицируется на три типа, а именно на обучение с учителем, обучение без учителя и обучение с подкреплением.

2 Конструкторская часть

В качестве практики и демонстрации разработки робота с применением нейронных сетей была поставлена задача, разработать четырехногого робота, обученного ходьбе с помощью обучения с подкреплением. Причиной разработки робота с ногами является тот факт, что передвижение такого робота не тривиальная задача для типичного управления одним скриптом программы, особенно в условиях неровной поверхности, так как роботу необходимо держать равновесие при этом делая шаги вперед, а каждый шаг подразумевает сложное движение нескольких суставов одновременно, контролируя угол наклона и учитывая скорость поворота каждого сустава. Решением этого является использование нейронной сети, которая будет принимать решения на основе входных данных. Нейронная сеть будет находить шаблон в приходящих сигналах с видеокамеры и датчиков, и на их основе принимать решения, постепенно обучаясь ходить. В данном случае обучение с подкреплением идеально подходит, так как принятые в этом методе абстракции такие как агент, среда, политика подходят под задачу проекта. Также в качестве дополнительной функциональности была поставлена задача, обрабатывая сигнал с видеокамеры, классифицировать выбранный пользователем объект и обозначить его как конечный пункт, к которому должен подойти робот.

Поставленную задачу можно разбить на основные две подзадачи – спроектировать шасси и собрать его, разработать управляющее устройство робота.

2.1 Шасси робота

Шасси робота будет собираться из приводов, каркаса, автономного источника питания и микроконтроллера, управляющего нагрузкой на приводе. Каждый компонент комплектующих следует тщательно подобрать под нужды проекта.

2.1.1 Привод

Вследствие того, что перед роботом стоит задача шагать, следует установить на робота в качестве приводов сервопривода. Сервопривод – это устройство, меняющее угол вала в зависимости от приходящего извне сигнала. Сервопривода отлично подходят для создания момента вращения в суставах робота, при этом их угол полностью контролируется микроконтроллером, что делает их очень удобными в задаче передвижения шагающего робота.

В качестве модели сервопривода была выбрана модель MG996R от компании TowerPro (рисунок 2.1). Данные сервоприводы оснащены металлическим механизмом, качественны и надежны. В плане показателя качество/цена данная модель фаворитом в сегменте сервоприводов средней цены. Номинальное напряжение сервоприводов 4,8-7,2 В, ток при максимальной нагрузке достигает 1000 мА. Угол вращения 180°, что

является достаточным для сгиба сустава. Скорость вращения вала в зависимости от напряжения достигает от 60°/0,17 с до 60°/0,14 с. Максимальный крутящий момент 11 кг·см [4], что необходимо учесть при моделировании механических частей робота. Эти сервопривода будут получать питание напрямую с аккумулятора и сигнал, генерируемый широко-импульсной модуляцией микроконтроллера. Было решено устанавливать по 2 сервопривода на каждую ногу робота. Данное решение обусловлено тем, что одного сервопривода недостаточно для эффективной ходьбы, при этом установка трех сервоприводов на каждую ногу может сильно усложнить и увеличить бюджет проекта. Поэтому был произведен заказа восьми сервоприводов на сайте Aliexpress с доставкой по почте.

Рисунок 2.1 – Сервопривод TowerPro MG996R

2.1.2 Микроконтроллер

Микроконтроллер в системе робота займет роль “спинного мозга”. Так как управление восемью сервоприводами требует восьми портов GPIO с поддержкой ШИМ модуляции, подключить сервоприводы напрямую к управляющему устройству не получится. Посредником между шасси и управляющим устройством будет микроконтроллер, который будет по последовательному порту получать значения углов и передавать сразу на сервоприводы путем ШИМ. Микроконтроллер будет прошит программой, реализующей общение шасси и управляющего устройства по простому протоколу, подробное описание которого будет предоставлено в практической части.

В качестве микроконтроллера был выбран микроконтроллер STM32L031K6 на платформе для разработчиков ST Nucleo (рисунок 2.2). ST Nucleo – высокопроизводительная платформа для прототипизации проектов от компании ST, производителя микроконтроллеров STM32. Данные микроконтроллеры отличаются высокими показателя производительности, гибкости и надежности. В модели использованной в проекте имеется ядро микропроцессора, который может разгоняется до частоты 32 MHz, 8 КБ RAM памяти, порт USART, интерфейсы I2C, SPI,

один 16-и разрядный таймер с четырьмя каналами ШИМ и два 16-и разрядных таймера с двумя каналами ШИМ [5]. Решение выбрать именно эту платформу в качестве микроконтроллера обусловлено несколькими факторами. Первый фактор – микроконтроллеры STM32 превосходят аналоги по показателям скорости вычислений и количества каналов ШИМ, второй фактор – в отличии от платформы Arduino на данной платформе писать оптимальный код легче, в тоже время это не так сложно в сравнении с работой с голым микроконтроллером от компании atmel, третий фактор – во время производственной практики на рабочем месте я получил большой опыт работы с микроконтроллерами STM32, в частности с моделью L073RZ, также с архитектурой микроконтроллера, таймерами, прерываниями, регистрами и памятью типа flash-памяти, EEPROM и в целом с данной платформой.

Работа с микроконтроллером будет производится из MDK-ARM IDE от компании Keil. Платформа имеет встроенный программатор ST-LINK/V2-1, поэтому для работы потребуется только подключится к порту mini-USB и запустить IDE Keil uVision. Для упрощения процесса программирования было принято решение использовать для разработки программного обеспечения микроконтроллера библиотеку HAL. Начальный код инициализации основных систем и периферии микроконтроллера будет сгенерирован генератором кода STMCubeMX.

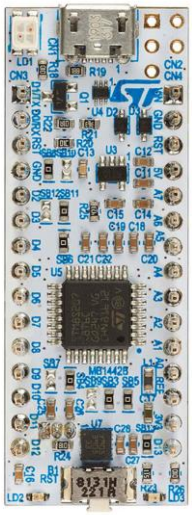

Рисунок 2.2 – Платформа Nucleo STM32L031K6

2.1.3 Аккумулятор

Аккумулятор будет собран из популярных литий-ионных аккумуляторов формата 18650, балансирующей заряд, платы и преобразователей напряжения.

Аккумуляторы 18650 будут включаться последовательно к друг другу и выдавать на выходе 8,4 В. Каждый аккумулятор имеет емкость в 2600 mAh. Максимальный разрядный ток достигает до 30 А. в качестве

аккумуляторов была выбрана модель аккумуляторов VTC5 от компании Sony (рисунок 2.3).

Рисунок 2.3 – Аккумуляторы Sony VTC5

У литий-ионных аккумуляторов много преимуществ, но одним из важных недостатков является, то, что при зарядки очень важно следить, чтобы один из аккумуляторов не был заряжен больше остальных. Поэтому в аккумуляторных сборках крайне важно устанавливать аккумуляторы с одинаковыми емкостями. Так как это не всегда возможно, были разработаны платы балансировки. Такие платы следят за емкостью каждого аккумулятора и при достижении хотя бы одного до края заряда, отключают подачу питания. В дипломном проекте будет использоваться Балансировочная плата HX-2S-JH20 v1.0 (рисунок 2.4).

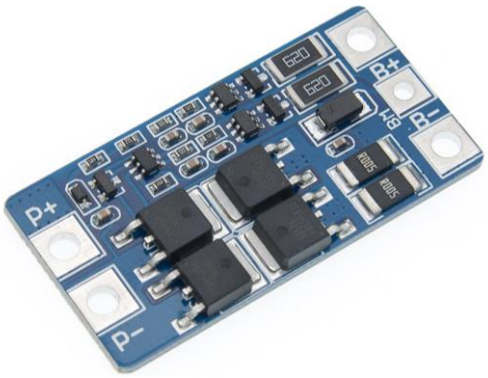

Рисунок 2.4 – Балансировочная плата HX-2S-JH20 v1.0

Для того, чтобы контролировать уровень напряжения при заряде аккумулятора, будет использоваться DC-DC понижающий модуль и блок питания, выдающий 2 А при напряжении в 12 В.

2.1.4 Каркас

Каркас будущего робота должен быть спроектирован с учетом того, что количество ног должно быть 4, они должны быть расположены симметрично относительно друг друга, в качестве суставов применяются сервоприводы модели MG996R, каждая нога имеет 2 сустава, каждый сервопривод должен крепко крепится к каркасу, а также к каркасу будет

закреплен миниатюрный компьютер Raspberry Pi, аккумуляторная батарея, платформа Nucleo и разные модули, провода и переходники.

Было принято решение составить каркас из трех основных частей, а именно из тела, верхнего звена ноги и нижнего звена ноги. Каждая составная часть будет крепится к другой через сервопривод. два звена ноги будут составлять ногу. В итоге получается одна деталь тела, четыре детали верхнего звена ноги и четыре детали нижнего звена ноги.

Для моделирования каркаса решено использовать программный комплекс SolidWorks, т.к. данное программное обеспечение имеет в инструментарии все необходимые методы моделирования и проектирования.

В качестве тела за основу было взято н-образная форма. К телу крепится 4 сервопривода, и для крепкого сцепления с корпусом было принято решение вырезать в теле специальные отверстия для установки в них сервоприводов. По бокам были сделаны отверстия для крепления ушек сервоприводов к телу с помощью винта и гайки. На рисунке 2.5 показан итоговый эскиз отверстия, к которому будет закреплен сервопривод.

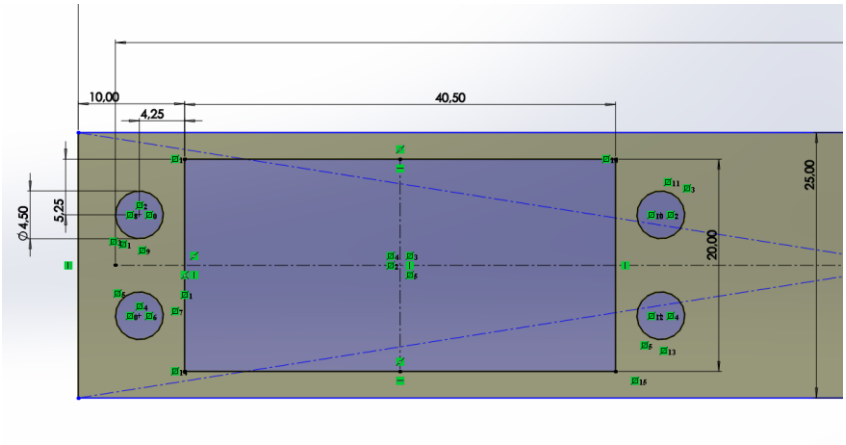

Рисунок 2.5 – Эскиз отверстия для крепления сервопривода

Далее спроектирован н-образный каркас и проделаны все четыре отверстия для сервоприводов. Тело спроектировано как две основательные бобышки, к которым крепятся ноги, соединенные тремя держателями. Они служат частями, к которым будет крепится электроника робота. Через держатели проходит платформа, на которую ставится Raspberry Pi и камера сверху, к низу крепится на термоклей аккумулятор и остальная электроника (рисунок 2.6). Далее для дополнительного пространства крепления и защиты электроники проектируется открытый корпус внизу платформы. В корпус помещается платформа Nucleo, 5,5 мм-вый вход, для зарядки аккумулятора и модуль гироскопа. Для вывода из корпуса USB-порта платформы Nucleo и вход питания аккумулятора проделаны отверстия.

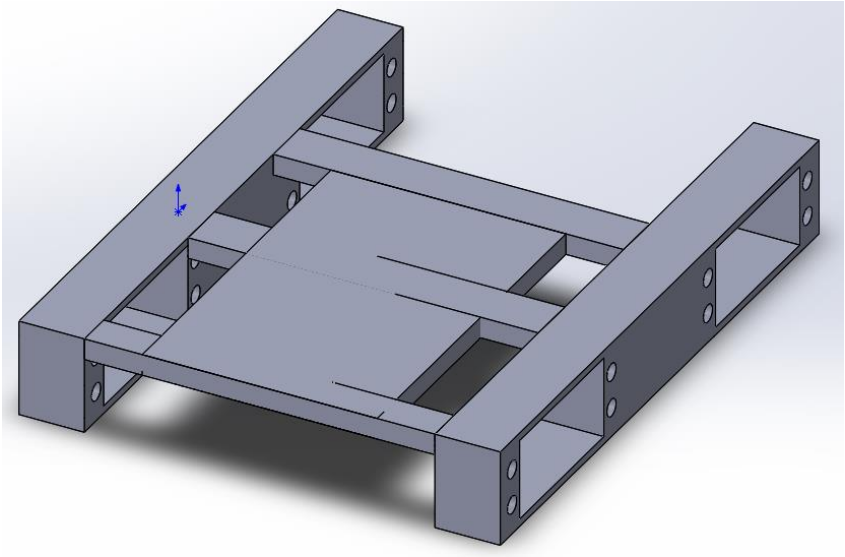

Рисунок 2.6 – Тело робота

А также для дополнительного крепления стяжками Raspberry Pi и аккумулятора к корпусу, сделаны отверстия в держателях. Для крепления сервоприводов на винт, предусмотрен зазор для вставки гайки. Итоговый вариант модели тела робота показан на рисунке 2.7.

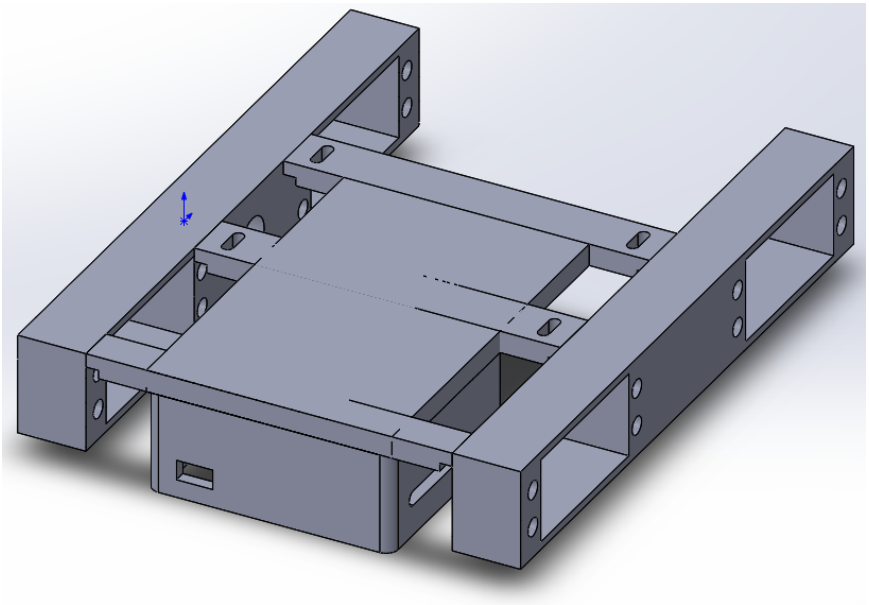

Рисунок 2.7 – Итоговая модель тела робота

Каждая нога разбита на два звена. Первое верхнее звено играет роль плеча сервопривода на теле и на нижнем звене ноги, тело передает по нему момент силы, а нижнее звено упирается об него. Верхнее звено ноги спроектировано так, чтобы иметь дополнительную прочность на местах стыка с сервоприводами. Для крепежа с сервоприводом не смоделированы отверстия под крепеж сервоприводов к верхнему звену ноги, так как отсутствуют точные чертежи места крепления сервопривода. Было принято решения проделать отверстия вручную, миниатюрным сверлом. Для

крепления было вырезано углубление глубиной 20 мм. Модель верхнего звена ноги показано на рисунке 2.8.

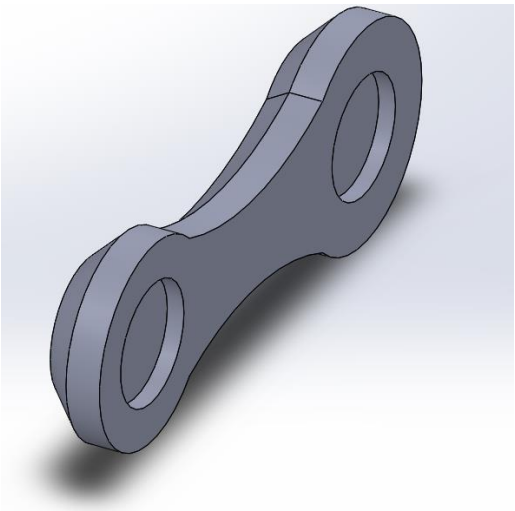

Рисунок 2.8 – Модель верхнего звена ноги

Нижнее звено ноги имеет отверстие для крепления сервопривода, такое же как в теле робота. Его основная задача отталкиваясь от земли перемещать тело по направлению вперед или назад. Часть нижнего звена, упирающаяся к земле выполнена в виде цилиндра, вытянутого горизонтально. Основание нижнего звена соединено с цилиндром двумя костями. Модель нижнего звена ноги показана на рисунке 2.9.

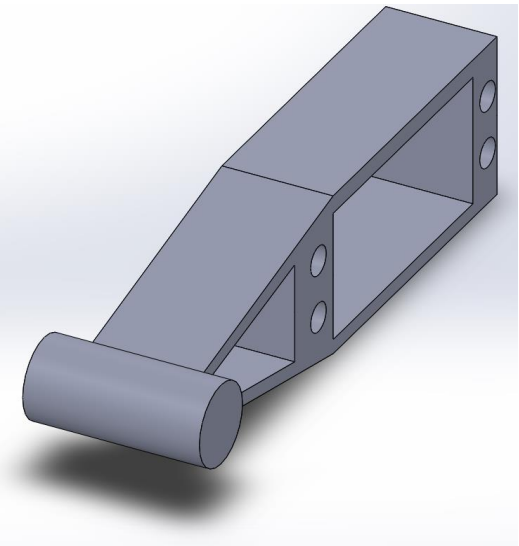

Рисунок 2.9 – Модель нижнего звена ноги

2.2. Управляющее устройство робота

Основная задача управляющего устройства является управление шасси таким образом, чтобы робот выполнял поставленную задачу. Для обработки информации была принята модель, в которой основной вычислительной машиной будет стационарный компьютер, а мобильная

часть будет передавать и принимать информацию среды по средствам беспроводной передачи информации по wi-fi.

2.2.1 Бортовой компьютер

В качестве компьютера, который будет управлять шасси и являться связующей частью между камерой и сервером, где будет выполняться обработка данных, был выбран одноплатный мини-компьютер Raspberry 4 Model B в варианте с 1 ГБ RAM-памяти. В слот памяти установлена CD-карта класса скорости class 10 и объемом 16 ГБ (рисунок 2.10).

Рисунок 2.10 – Raspberry Pi 4 Model B

На raspberry будет установлен веб-сервер mjpeg-streamer, открытое программное обеспечение, которое будет передавать видеопоток на сервер и там уже обрабатывается. Также через порт USB к Raspberry Pi подключается микроконтроллер. Таким образом через один кабель микроконтроллер получает питание и канал связи с Raspberry Pi.

На Raspberry Pi должна быть написана программа, которая будет принимать команды с сервера и передавать микроконтроллеру. Реализована она будет на языке программирования python. Использоваться будет уже установленный на raspbian интерпретатор python3.

2.2.2 Обработка потока информации

В работе с нейронными сетями и для обучения их обязательно нужен инструментарий в виде библиотек предоставляющих готовый набор классов и методов, реализующий модели нейронных сетей и обучающих алгоритмов, а также дающих удобный способ взаимодействия с данными. На основе этих библиотек и будет строится программный код робота.

В сфере машинного обучения самой популярной и распространенной библиотекой машинного обучения являются библиотека TensorFlow. Данная библиотека реализует множество алгоритмов обучения нейронных сетей через API написанную на языке Python. Она позволяет для вычислений использовать ГПУ с технологией CUDA, что заметно ускоряет

процесс обучения и прогноза. Начиная с второй версии TensorFlow использует в качестве основной API библиотеку Keras, библиотеку, которая является надстройкой над самой TensorFlow. Данная библиотека реализует TensorFlow применяя более простую для понимания систему классов, чем сам TensorFlow.

Для использования библиотеки рекомендуется работать в операционной системе из семейства unix-подобных систем Linux. Чтобы приступить к работе с библиотекой, нужно установить требуемый интерпретатор языка программирования Python или использовать уже установленный в операционную систему Linux интерпретатор. Также рекомендуется использование виртуальной среды для контроля установленных библиотек. Установка TensorFlow производится посредствам систем управления пакетами такими как pip или conda. После установки достаточно в тексте программы импортировать библиотеку строчкой “import tensorflow as tf”. В данной строчке кода указано, что библиотека TensorFlow далее в данном листинге будет обозначаться как “tf”.

Для тестирования производительности и доступности работы с библиотекой Tensor, был выбран открытый репозиторий на сайте GitHub реализующий TensorFlow и систему обнаружения Yolo третьей версии [6]. Данный репозиторий был адаптирован под проект и использован в основном программном коде проекта. Подробнее об этом будет написано в практической части дипломной работы.

TensorFlow имеет две параллельной версии – версию для вычислений на центральном процессорном устройстве (ЦПУ) и версию, где в качестве устройства вычисления используется графическое процессорное устройство (ГПУ). В компьютере, который будет использован в проекте в качестве сервера, установлен процессор модели i5-3470 с тактовой частотой 3.2 ГГц. На сервер установлена Unix-подобная операционная система Pop!_OS основанная на дистрибутиве Ubuntu 19.04. Данная операционная система также как и Ubuntu использует рабочий стол GNOME shell, но отличается заранее предустановленными проприетарными драйверами для видеокарты и готовыми инструментами для установки версии TensorFlow для работы с ГПУ. После обновления системы устанавливается программа для тестирования. Обработка каждого кадра разрешением 421 на 416 пикселей на ЦПУ составила в среднем 1 секунду. Программа выводила на экран время, потраченное на обработку одного полного цикла программы и вывод программы, можно увидеть на рисунке 2.11.

По данным теста, был сделан вывод, что производительности ЦПУ сервера недостаточно, поэтому было принято решения использовать ГПУ для обработки видеопотока, так как ГПУ лучше подходит под задачи вычислений, выполняемых в ходе работы программы. Архитектура ГПУ подразумевает вычисления с большим количеством несложных операций, что и происходит во время вычислений прогнозов нейронных сетей.

```
0.9957461357116699
1.0212767124176025
1.0380687713623047
1.0137519836425781
1.0695888996124268
1.0080769062042236
1.0051281452178955
1.0030431747436523
1.006497859954834
1.0091736316680908
```

Рисунок 2.11 – Вывод программы тестирования

TensorFlow “из коробки” поддерживает только работу с видеокартами от компании Nvidia, так как технология CUDA запатентована именно этой компанией и поставляется в их видеокартах. Поэтому для проекта уже имеющаяся видеокарта с чипом Radeon R9 380 от компании AMD не подходила. Пришлось обменять данную видеокарту на ее аналог от компании Nvidia, а именно на видеокарту с чипом GeForce 1050 ti в сборке от компании Geil (рисунок 2.12).

Рисунок 2.12 – Видеокарта GeForce 1050 ti собранная компанией Geil

Данная карта обладает характеристиками ниже среднего сегмента рынка видеокарт, что было решено будет достаточно для требуемой вычислительной мощности проекта. Видеокарта имеет количество универсальных процессоров CUDA в количестве 768, объем видеопамяти в 4 Гб. Без нагрузки датчик температуры на видеокарте показывал температуру 30 °C. По предварительному тестированию производительности один цикл работы программы, обрабатывающей видеопоток с разрешением 416 на 416 пикселей, длился в среднем 0.324 секунды, что уже лучше предыдущего теста. Вывод программы тестирования показан на рисунке 2.13. Нагрузка увеличила температуру видеокарты до 38 °C.

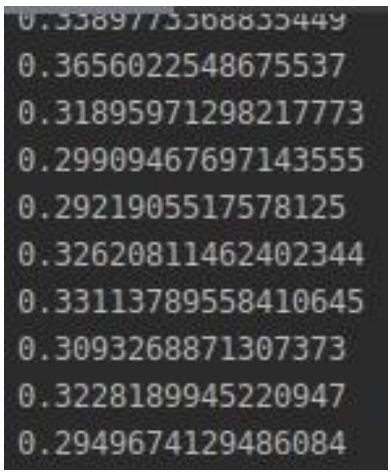

Рисунок 2.13 – Вывод программы тестирования

По данным тестирования была построена таблица 2.1 и выбран самый производительный вариант

Таблица 2.1– Результаты тестирования

№	Наименование устройства и конфигурации	Разрешение кадра	Средний период итерации программы, с
1	i5-3470, 3.2 ГГц	416 x 416	1
2	GeForce 1050 ti, 4Gb	416 x 416	0.324

2.2.3 Математическая модель

Перед тем как приступить к программированию нейронной сети, следует осознавать, что сама нейронная сеть является математической моделью, и следует ее построить.

Для построенная собственной математической модели необходимо изучить и понять основные принципы работы искусственных нейронных сетей. Для этого в качестве простейшего примера нейронной сети берется ИНС прямого распространения и обучается классифицировать рукописные цифры. Данный пример дает детальное понимание того, как строиться математическая модель и на этой математической модели далее работает программа.

Обработка начинается с фотографии рукописной цифры. На входе в программу приходит цифровая фотография рукописной цифры. Каждый пиксель фотографии мы можем представить как спектр значений от 0 до 1, где 1 это полностью белый, а 0 полностью черный пиксель. Таким образом на входе в программу приходит множество (2.1)

$$\underline{x} = \{x_1, x_2, ..., x_N\},\ x_i \in R \qquad (2.1)$$

Для удобства это множество лучше представить в виде вектора. Также нужно обусловится, что каждая фотография должна иметь

разрешение 28x28 пикселей. В таком случае N будет равняться количеству всех пикселей

$$N = 28 * 28 = 784$$

В основе классической нейронной сети лежит математическая модель перцептрона. В нашем примере у перцептрона на входе все значения вектора. В общем случае их может быть больше или меньше.

Также есть простое правило для вычисления результата, а именно веса (2.2).

$$\underline{w} = \{w_1, w, ..., w_N\},\ w_i \in R \tag{2.2}$$

Веса – это вещественные числа, выражающие важность соответствующих входных чисел для результатов. То есть у каждого входного значения есть свой критерий важности. Веса и есть тот механизм, который позволяет нейронной сети реагировать на входные значения таким образом, которым эту нейронную сеть обучат. Значения весов также определим как вектор. Выход перцептрона либо 0, либо 1. Определяется это так: если сумма меньше некоего порога b, то 0, если больше, то 1. Выходное значение такой модели можно записать как (2.3).

$$\text{Выход} = \begin{cases} 0 \text{ if } \underline{x} \times \underline{w} + b \leq 0 \\ 1 \text{ if } \underline{x} \times \underline{w} + b > 0 \end{cases} \tag{2.3}$$

Чтобы перцептрон играл роль полноценного нейрона его значение на выходе должно быть не 1 и 0, но диапазоном значений между 1 и 0. Таким образом нейронную сеть в будущем можно будет обучить, т.к. мелкие изменения весов сети влекут мелкие изменения значений на выходе. Также порог b в таком случает уже играет роль некого смещения, которое будет регулировать, то насколько легко нейрон смещаться к значению 1 или 0. Это реализуется путем определения выражения как значение z и заключения в функцию сигмоиды, где функция сигмоиды записывается как (2.4).

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.4}$$

На графике функции сигмоиды, где на оси абсцисс выходное значение, а на оси ординат значение z, можно заметить, что если значение z большое и положительное, то функция стремится к 1, а когда значение z большое, но со знаком минус, функция стремится к 0 (рисунок 2.14). Это указывает на одновременную схожесть сигмоидной функции с персептроном, при этом вывод функции не меняется скачками как у перцептрона.

Схематично нейрон можно изобразить как на рисунке 2.15. На схеме вектор показан как входное значение. Вектор и значение b как внутреннее свойство нейрона, воздействующие на входные значения. Сигмоидная функция приводить пришедшее значение в диапазон от 0 до 1.

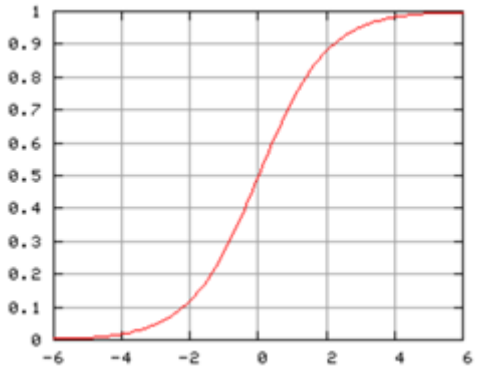

Рисунок 2.14 – График сигмоидная функции

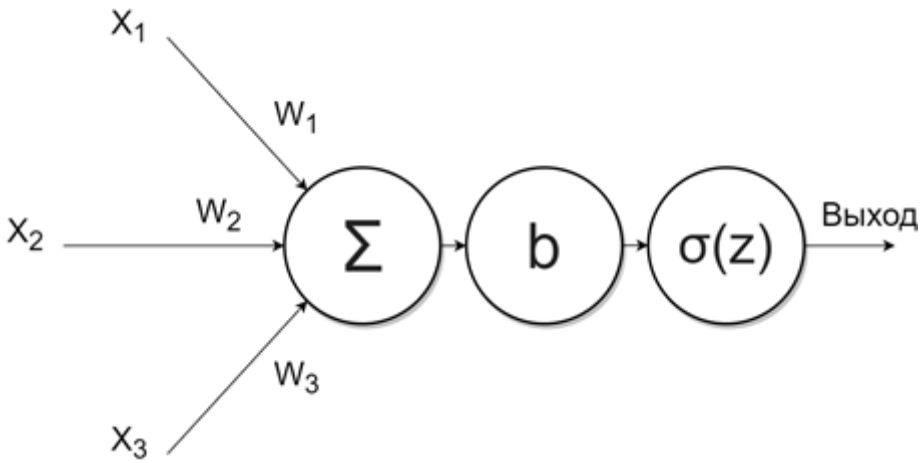

Рисунок 2.15 – Схематичное представление нейрона

В виде математического выражения это будет выглядеть как (2.5):

$$\text{Выход} = \frac{1}{1 + e^{\left(-\sum_i^N w_i x_i - b\right)}} \qquad (2.5)$$

Это только один нейрон. В нейронной сети используется множества нейронов. Схематично сеть будет выглядеть как на рисунке 2.16. Структура сети строится слоями. В каждом слое есть некоторое количество нейронов. Самый первый слой называется входным слоем. Количество нейронов этого слоя равно количеству входных значений, т.е. значению N.

Обязательное количество слоев 2. Это входной и выходной слой. В данной сети выходной слой состоит из 10 нейронов, где каждый нейрон отвечает за определённую цифру. Слои между входным и выходным слоем опциональны и называются скрытыми. Количество этих слоев и количество нейронов в этих слоях выбирается на усмотрение программиста.

Увеличивая количество слоев, можно увеличить и точность распознавания, но также и увеличивается требования и нагрузка к аппаратной составляющей вычислительного устройства. Для данного примера используется два скрытых слоя. В каждом слое по 128 нейрона.

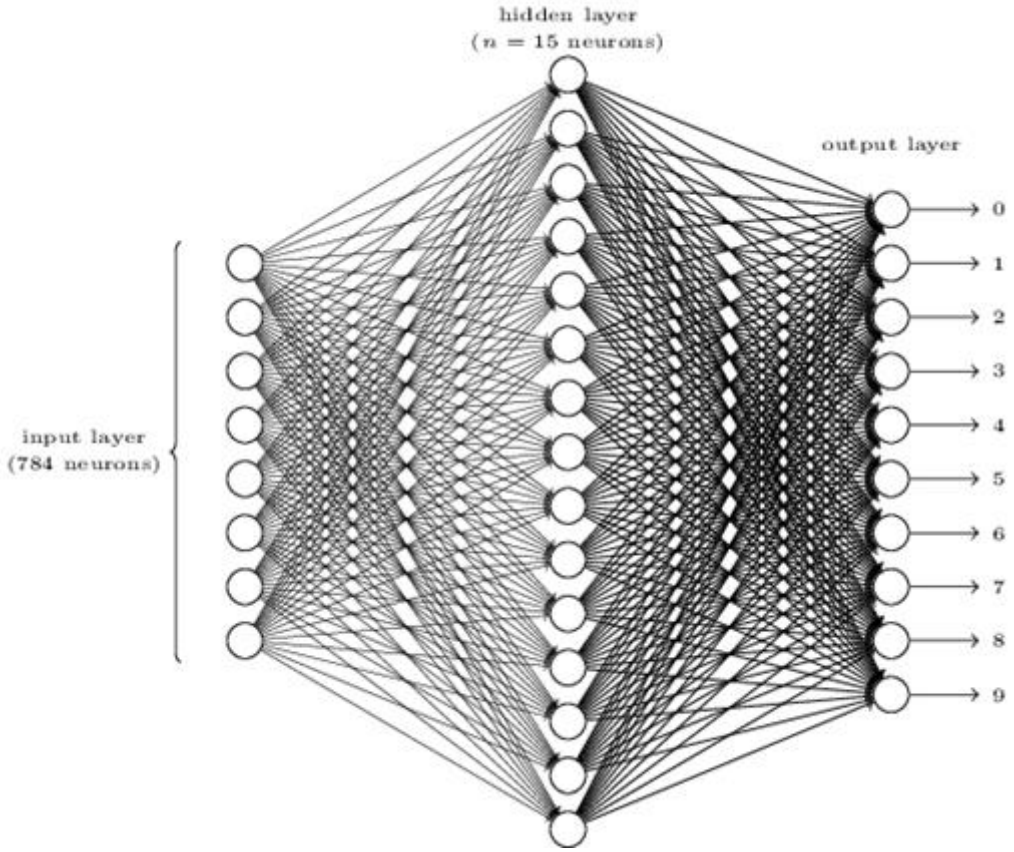

Рисунок 2.16 – Структура нейронной сети

Суть нейронной сети проявляется в поведении сети при подаче на входной слой каких-либо данных. Дело в том, что сеть будет реагировать на входные данные в зависимости от настроенных значений весов и смещений. Некоторые нейроны из второго слоя в зависимости от веса по отношению к некоторым нейронам первого слоя будут ослаблять их значение весом близким к нулю, а некоторые сохранять это значение весом близким к единице. Настройка весов и смещений называется процессом обучения. Таким образом веса и смещения во время обучения формируют абстрактные шаблоны, каждый из которых будет формировать полноценный образ.

По сути, нейронная сеть на выходе выдает 10 значений, где каждое значение это шанс того, что объектом распознавания является советующая цифра. То есть нейронная сеть измеряет шанс. Как и любое другое измерительное устройство, нейронная сеть не способна выдавать результат с абсолютной точностью. Чтобы рассчитать точность нейронной сети, рассчитывается среднеквадратичная ошибка (СКО). В процессе обучения СКО нейронной сети будет уменьшаться, пока не остановится на одном определенном значении.

Для моделирования искусственной нейронной сети используется язык программирования python и библиотека TensorFlow. В качестве среды написания кода программы используется Jupyter Notebook.

Первым делом импортируем библиотеку TensorFlow (рисунок 2.17)

```
import tensorflow as tf
tf.__version__
```

Рисунок 2.17 – Импортирование библиотеки TensorFlow

Далее нужно загрузить примеры обучения нейронной сети (рисунок 2.18). Примеры обучения – это экземпляры фотографий рукописных цифр и правильные вариант ответа. Это очень важный аспект обучения.

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Рисунок 2.18 – Загрузка примеров

Рассматриваемый вид нейронной сети относится к нейронным сетям прямого действия. Слои нейронной сети применяются последовательно от входного слоя к выходному. В TensorFlow для инициализации такого типа модели нейронной сети существует класс Sequential (рисунок 2.19).

```
model = tf.keras.models.Sequential()
```

Рисунок 2.19 – Инициализация модели

После инициализации модели в нее нужно добавить слои в том порядке, в котором они расположены на рисунке 3, т.е. входной слой, два скрытых слоя, выходной слой. В TensorFlow слои добавляются с указанием количества нейронов в слое и функцией активации на всех слоях кроме входного слоя (рисунок 2.20). Количество нейронов входного слоя будет зависит от количества входных данных. Напомню в данной сети это количества рассчитано формулой (2).

```
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
```

Рисунок 2.20 – Добавление слоев

Далее нужно произвести компиляцию модели, по которой будет производится обучение нейронной сети. В качестве расчета ошибки, применяем среднеквадратичную ошибку (рисунок 2.21).

```
model.compile(optimizer='adam',
              loss='mean_squared_error')
```

Рисунок 2.21 – Компиляция модели

Далее идет строка кода, которая запускает сам процесс обучения. В качестве параметров метод fit применяет примеры и количество “epoch” (рисунок 2.22). Epoch – это количество повторений обучения. Результат обучения может меняться в положительную сторону при повторении действий обучения несколько раз.

```
history = model.fit(x_train, y_train, epochs=10)
# print(help(tf.keras.models))

Epoch 1/10
60000/60000 [==============================] - 5s 91us/sample - loss: 27.3045
Epoch 2/10
60000/60000 [==============================] - 5s 84us/sample - loss: 27.3045
Epoch 3/10
60000/60000 [==============================] - 5s 86us/sample - loss: 27.3045
Epoch 4/10
60000/60000 [==============================] - 5s 87us/sample - loss: 27.3045
Epoch 5/10
60000/60000 [==============================] - 5s 86us/sample - loss: 27.3045
Epoch 6/10
60000/60000 [==============================] - 6s 92us/sample - loss: 27.3045
Epoch 7/10
60000/60000 [==============================] - 5s 91us/sample - loss: 27.3045
Epoch 8/10
60000/60000 [==============================] - 5s 90us/sample - loss: 27.3045
Epoch 9/10
60000/60000 [==============================] - 5s 87us/sample - loss: 27.3045
Epoch 10/10
60000/60000 [==============================] - 5s 88us/sample - loss: 27.3045
```

Рисунок 2.22 – Запуск обучения и вывод процесса обучения на монитор

В переменную history была сохранена история процесса обучения, а именно значения СКО в определенные промежутки времени. Эти данные можно вывести в график как на рисунке 2.23. На графике видно, что с количеством повторений СКО уменьшается, увеличивая шанс на успешное распознавание рукописной цифры.

Для проверки результатов, нейронной сети на вход нужно подать фотографии рукописных цифр, которые не были использованы во время тренировки. На рисунке 2.24 показана сама фотография, которая подается на вход нейронной сети. На рисунке 2.25 показан вывод программы.

Данная нейронная сеть сильно упрощена, но дает понимание того, как работаю любые нейронные сети и как работает высокоуровневое API Keras. С помощью этого инструмента будет и строится поведение мобильного робота.

Обучение с подкреплением строиться на таких терминах как агент, среда, наблюдение, награда, политика, действие, сессия и т.д. В случае данного дипломного проекта агентом будет являться мобильный робот. Средой будет окружающая агента реальность, откуда он будет получать данные, то есть наблюдения. Чтобы объяснить задачу роботу необходимо

построить систему поощрения, которая будет при правильных действиях робота поощрять его “наградой” или “наказанием”.

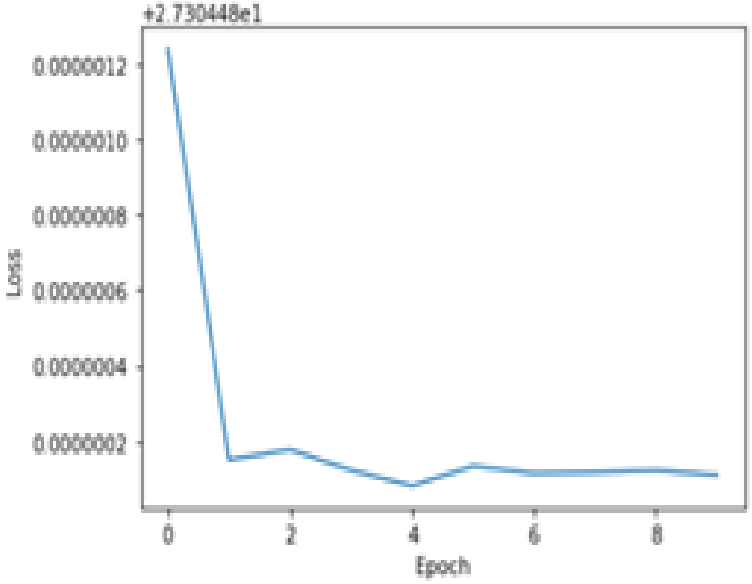

Рисунок 2.23 – График СКО от времени обучения

Чтобы достигать максимального счета награды робот будет формировать политику, которая в свою очередь будет генерировать необходимое действие для достижения цели. Любой цикл деятельности робота будет заканчиваться окончанием сессии. Конец сессии может быть возбужден либо неудачей, либо достижением цели. Каждый из этих процессов можно запрограммировать и в зависимости от этого будет формироваться поведение робота. Например, если запрограммировать систему поощрения отнимать баллы “награды” каждую секунду, то робот, для максимизации баллов будет решать задачу быстрее. Или если при удачном конце сессии давать роботу большую награду, то робот может начать игнорировать некоторые неочевидные действия, стремясь получить награду.

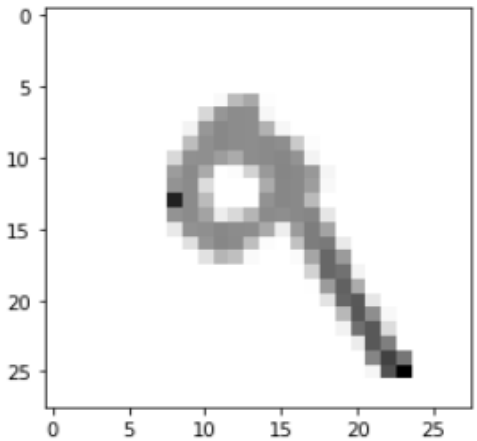

Рисунок 2.24 – Изображение переданное на входной слой

```
import numpy as np

print(np.argmax(prediction[7]))
```

```
9
```

Рисунок 2.25 – Вывод предсказания цифры

Для дипломного проекта было решено спроектировать собственную среду, которая будет поощрять агента за каждое действие приводящие к приближению к цели, в случае проекта – кружки. Среда будет обращаться к функции классификации из файла “processing.py” и получить размер рамки объекта и ее координаты. Размер рамки определяет то насколько агент близок к объекту, а координата относительно центра будет указывать насколько прямо к цели идет агент. Неудачный конец сессии будет происходит при потере объекта из видимости камеры в течении нескольких циклов получения наблюдения. В этом случае агент получит штраф. Успешная сессия будет засчитывается при достижении рамки определенного размера. Тогда агент получит весомую награду. Каждое увеличение рамки будет пропорционально увеличивать награду, а уменьшение, наоборот, уменьшать. Таким образом агент будет стремиться повышать счет “награды” приближаясь к объекту при этом не теряя эти баллы, отойдя назад.

В качестве политики была выбрана простая политика, предложенная Рональдом Уильямсом в 1992 году [7]. “Дайте возможность политике в форме нейронной сети сыграть в игру несколько раз и на каждом шаге вычисляйте градиенты, которые сделают выбранное действие даже более вероятным, но пока не применяйте градиенты. После прогона нескольких эпизодов подсчитайте оценки каждого действия (используя метод, который описан в предыдущем разделе). Если оценка действия положительная, то это означает, что действие было хорошим, и вы хотите применить вычисленные ранее градиенты, чтобы сделать выбор данного действия в будущем еще более вероятным. Если же оценка действия отрицательная, то это значит, что действие было плохим, и вы хотите применить противоположные градиенты, чтобы сделать его выбор в будущем чуть менее вероятным. Решение заключается просто в умножении каждого вектора-градиента на соответствующую оценку действия. Вычислите все результирующие векторы-градиенты и воспользуйтесь ими для выполнения шага градиентного спуска” [8].

Политика в форме нейронной сети будет принимать на входе вектор из трех значений: площадь рамки, координата относительно центра по оси x и координата относительно центра по оси y. В качестве скрытого слоя будет применяться один слой из 5-ти нейронов с функцией активации “ELU”. Выходной слой сети будет иметь 6 нейронов, которые будут значением от 0 до 1, где 0 – это крайнее заднее положение, а 1 – крайнее

переднее положение. Нейронная сеть представляет из себя стандартный перцептрон и схематично будет выглядеть как показано на рисунке 2.26.

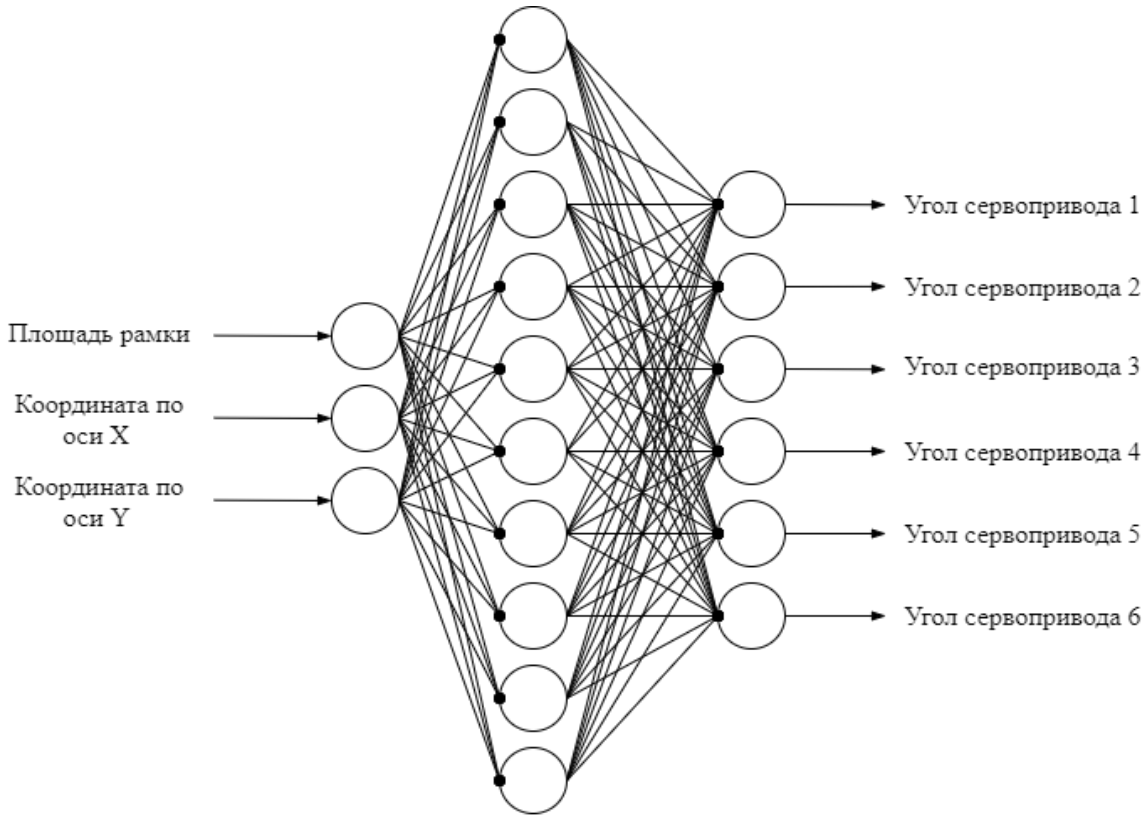

Рисунок 2.26 – Политика в виде нейронной сети

3 Реализация проекта

Первым шагом, для реализации проекта, будет полная сборка каркаса и электронных частей робота в единое автономное устройство. Для этого 3-х мерные модели каркаса распечатываются на 3D-принтере. Используется 3D-принтер модели Designer X от компании Picaso. В качестве материала устанавливается ABS пластик, внутреняя камера разогревается до 100° С, стол, на котором производится печать, нагревается на 250°. Задание для принтера подготавливается программой-слайсером Polygon X. Итоговый результат показан на рисунках 3.1 - 3.3.

Рисунок 3.1 – Распечатанное основание каркаса

Рисунок 3.2 – Распечатанные первые звенья ноги

Рисунок 3.3 – Распечатанные вторые звенья ноги

Далее в основание нужно вставить 4 сервопривода и закрепить. На этом этапе возникла проблема, так как на этапе проектирования не было

учтено, что отверстие под сервопривод следует моделировать немного больше, чем сам сервопривод. Из-за этого пришлось отверстия расширять напильником и нагревать пластик.

После этого следует подготовка двух звеньев ноги. К первому звену ноги нужно прикрепить плечо сервопривода, которое шло в комплекте с сервоприводом (рисунок 3.4).

Рисунок 3.4 – Пластиковое плечо сервопривода поставленное в комплекте

Для этого на первом звене ноги смоделировано углубление. В это отверстие плечо сервопривода приклеивается термоклеем. Сила сцепления плеча и звена ноги была проверена. В итоге тестирования, при прокрутке звена ноги на сервоприводе, сцепление выдержало до порога, при котором не выдержала нагрузки резьба плеча. Во время проектирования также не учился процесс крепления плеча к валу сервопривода с помощью винта. Для этого необходимо было спроектировать отверстие в месте крепления плеча. Из-за этого было сделано отверстие вручную, шуруповертом с напряжением на аккумуляторе 9В, сверлом диаметром в 6,5 мм.

Второе звено ноги также как и основание имеет отверстия под сервоприводы. Отверстие было увеличено и вставлены сервоприводы. Механическая часть робота была полностью собрана. Результат сборки показан в приложении А.

Провода от сервоприводов необходимо подключить к соответствующим пинам питания и подачи импульса с микроконтроллера. Для этого была разработана плата, проводящая нужные электроды питания и микроконтроллера к круглым штырям, к которым будут подключаться сервоприводы. Для этого к монтажной плате были припаяны штыревые коннекторы и клемма. К штыревым коннекторам подключается микроконтроллер, а к клемме подключаться провода питания. Так как преобразователи напряжения были рассчитаны только на 5 А, было принято

решения разделить сервоприводы на две группы и питать каждую отдельным преобразователем. Первая группа будет питаться от клеммы на монтажной плате, а для второй будет смонтирована отдельная плата, которая будет подключаться через штыревые коннекторы к первой. То, как в итоге выглядит эта плата можно посмотреть на рисунке 3.5.

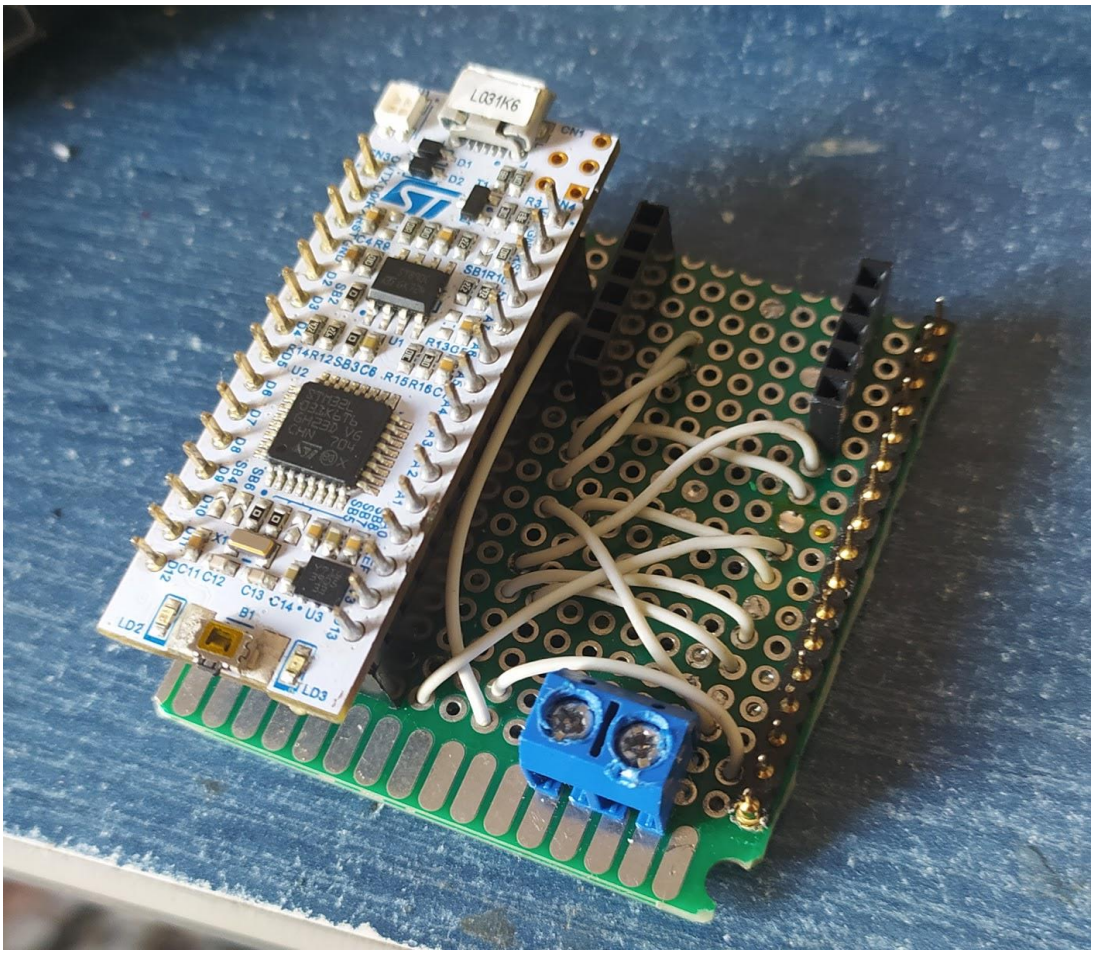

Рисунок 3.5 – Плата, соединяющая микроконтроллер, питание и сервоприводы

Аккумулятор для проекта был предоставлен руководителем дипломной работы. Из себя он представляем две подключенные последовательно литий-ионные аккумуляторные батареи формата 18650. К выходу сборки припаяны провода с коннектором XT30 “female” типа и коннекторы неизвестной модели для подключения платы балансировки, подключенные по схеме, показанной на рисунке 3.6.

Чтобы подать питание на клеммы была смонтирована плата, к которой были приклеены две преобразующие платы, припаяны провода с коннектором XT30 “male” типа, провод с коннектором DC-005/DC-022 с клеммным выходом и проводом для питания Raspberry Pi. Также к выходу преобразователей были подключены два конденсатора, чтобы сгладить импульсы питания. К выходам конденсаторов также были подключены клеммы (рисунок 3.7).

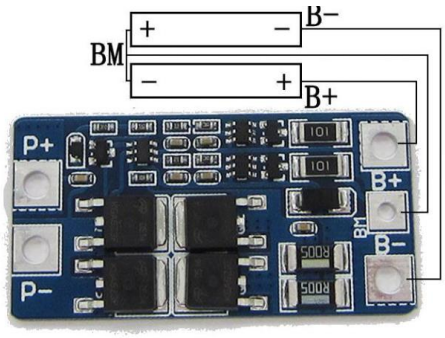

Рисунок 3.6 – Подключения балансирующей платы к аккумуляторным батареям

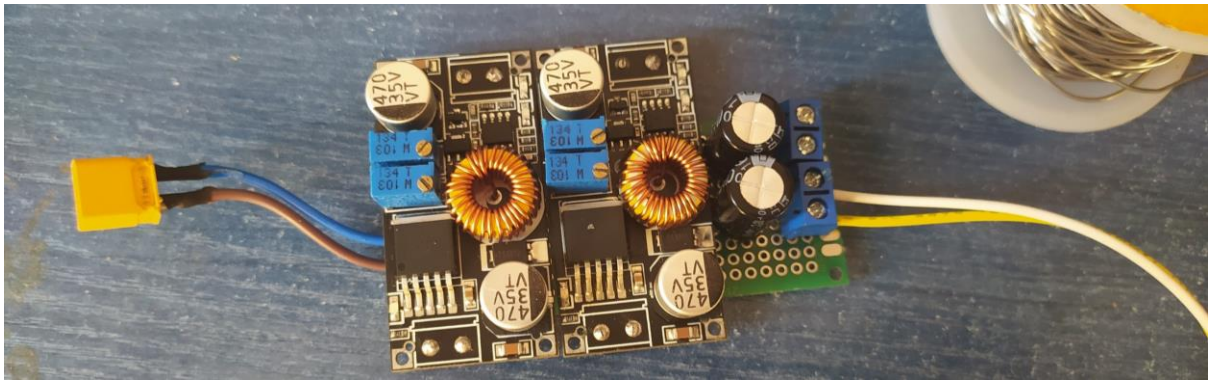

Рисунок 3.7 – Плата, преобразующая питание от аккумулятора к остальным частям робота

Далее необходимо написать программу для микроконтроллера. Для этого применяется генератор кода CubeMX. В данной программе нужно найти модель микроконтроллера, а именно STM32 L031K6. В закладке “Pinout & Configuration” нужно включить модуль USART2 в асинхронный режим, включить все три таймера TIM 2, TIM 21, TIM 22, при этом каждый канал таймера включить в режим PWM Generation CH. В процессе этого выходит, ошибка о конфликте второго канала TIM 2 и таймера TIM 21. Так как это непредвиденная ситуация принимается решения, использовать таймеры с этим конфликтом. После этого было замечено специфичная работа таймеров, при которой два первых канала таймеров TIM 21 и TIM 22 работают только при одинаковых значениях. Итоговый вид распиновки показан на рисунке 3.8.

В закладке “Clock Configuration” нужно рассчитать частоту работы внутренней архитектуры микроконтроллера. В качестве источника тактирования возьмем внутренний источник высокочастотного тактирования (HSI). Изначально он выдает частоту в 16 Мгц. Его сигнал направляется в PLL устройство, умножается на 2, в итоге частота на выходе 32 МГц. Далее сигнал идет уже на системную шину тактирования и остальные периферии, в том числе и на шину, на которой находятся все таймеры. На рисунке 3.9 показана итоговая схема тактирования.

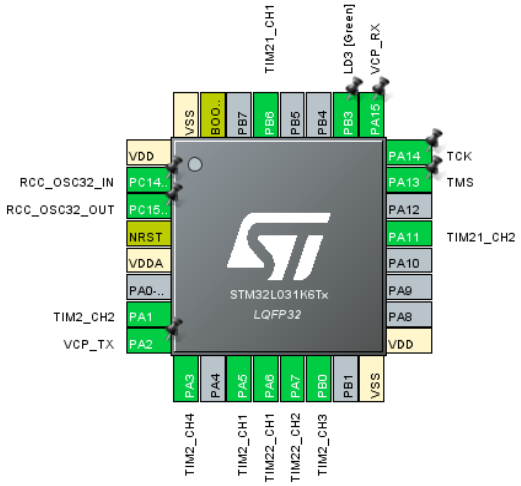

Рисунок 3.8 – Распиновка микроконтроллера

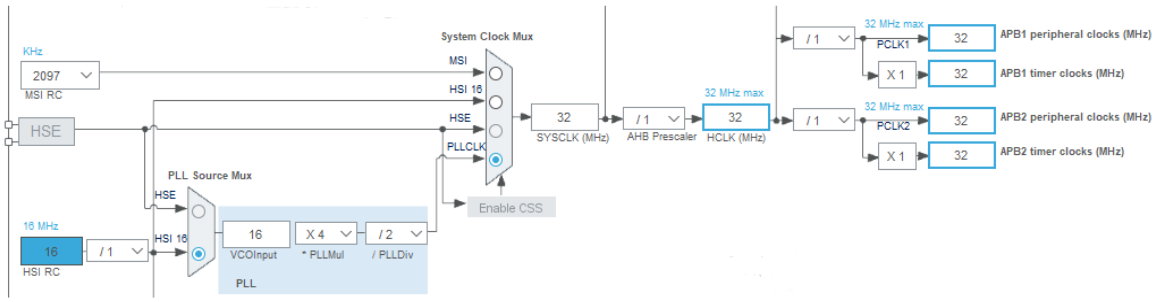

Рисунок 3.9 – Схема тактирования

После настройки тактирования генерируем код программы. Для управления и отладки код будет использоваться IDE Keil uVision5. В сгенерированном проекте CubeMX, в папке MDK-ARM нужно открыть проект с расширением “. uvprojx”. В данном проекте уже открыты файлы проекта. Каждый отдельный файл отвечает за инициализацию той или иной периферии. Открывается основновной файл “main.c”, который инициализирует всю систему микроконтроллера. В этом файле выполняется основной цикл программы. Далее создаем новые собственные файлы “servos.c” и “servos.h”. Данный файл будет отвечать за управление всеми сервоприводами. Каждый сервопривод управляется ШИМ. ШИМ генерируется таймерами общего назначения. Частота ШИМ по даташиту должна быть 50 Гц, то есть период ШИМ будет 20 мс [9]. А длиной импульса будет управляется угол вала на сервоприводе. Каждый таймер общего назначения тактируется 32 МГц-ами, как видно на рисунке 3.9. Эта частота делится на предделитель, чтобы подстроить под нужную единицу измерения. За предделитель отвечает 16-ти разрядный регистр PCL. Чтобы подстроить 32 МГц под период в 1 мкГц, нужно делить частоту на 32, поэтому поместим именно это значение в регистр PCL. За период таймера отвечает 16-ти разрядный регистр ARR. В этот регистр записываем значение 20000 мкс. Теперь переводим каждый таймер в режим PWM1. В этом режиме таймер будет генерировать ШИМ. Длину ШИМ будет

определяться значением в регистре CCRx, где x номер канала. Таких регистров у каждого таймера несколько. Таким образом угол сервоприводов регулируется регистрами CCRx. Именно таким образом файл “servos.c” будет управлять углом сервоприводов. В этом файле декларируем функцию установки угла. Экспериментальным способом получены крайние углы сервопривода. Минимальной шириной импульса является 600 мкс, а максимальной шириной 2600 мкс. Из-за разных положений сервоприводов относительно друг друга, соответствующие углы могут быть разными. Нулевая координата каждого сервопривода записана в виде константных значений в коде программы. Также инициализируется структура, содержащая углы сервоприводов. Код установки углов выглядит следующим образом (рисунок 3.10).

```
static void Set_Position(struct Servos_Position *pos)
{
  // TIM2
  htim2.Instance->CCR1 = ZERO_ANGLE_FRONT_RIGHT   - pos->front_right_feet_pos;     // front right
  htim2.Instance->CCR2 = ZERO_ANGLE_BACK_LEFT     + pos->back_left_feet_pos;       // back left
  htim2.Instance->CCR3 = ZERO_ANGLE_BACK_RIGHT    - pos->back_right_feet_pos;      // back right
  htim2.Instance->CCR4 = ZERO_ANGLE_FRONT_LEFT    + pos->front_left_feet_pos;      // front left
  // TIM21
  htim21.Instance->CCR1 = ZERO_ANGLE_SECOND_LEFT  - pos->second_left_feet_pos;     // left
  htim21.Instance->CCR2 = ZERO_ANGLE_SECOND_RIGHT + pos->second_right_feet_pos;    // right
  // TIM22
  htim22.Instance->CCR1 = ZERO_ANGLE_SECOND_LEFT  - pos->second_left_feet_pos;     // left
  htim22.Instance->CCR2 = ZERO_ANGLE_SECOND_RIGHT + pos->second_right_feet_pos;    // right
}
```

Рисунок 3.10 – Функция установки углов

Установка угла будет производится сразу же при получении команды с бортового компьютера. Для этого связным элементов будет основной цикл программы, в котором будет вызываться функция установки угла. Создается функция обработки, которая будет принимать структуру, содержащую углы сервоприводов. Полный код “servos.c” и “servos.h” предоставлен соответственно в приложениях Б и В.

Далее необходимо установить связь между бортовым компьютером и микроконтроллером. Для этого создается файлы “communication.c” и “communication.h”. Этот файл используя функцию библиотеки HAL, будет считывать буфер UART канала, парсить приходящую информацию в структуру и возвращать ее (рисунок 3.11). Полный код “communication.c” и “communication.h” предоставлен соответственно в приложениях Г и Д.

Каждый приходящий пакет будет сформирован по протоколу, описанному ниже в таблице 3.1.

Таблица 3.1 – Протокол общения

frame	data1		data2		data3		data4		data5		data6		CS
	High Byte	Low Byte	High Byte	Low Byte	High Byte	Low Byte	High Byte	Low Byte	High Byte	Low Byte	High Byte	Low Byte	
0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
14 struct Servos_Position Receive_and_Parse_Packet()
15 {
16   struct Servos_Position pos = {0};
17   if(huart2.RxXferCount == 0)
18   {
19     HAL_UART_Receive_IT(&huart2,(uint8_t*) buffer, EXPECTED_LEN);
20   }
21
22   uint8_t *buf = (uint8_t*) &buffer[1];
23
24   pos.back_left_feet_pos    =   *buf++ << 8;
25   pos.back_left_feet_pos    +=  *buf++;
26
27   pos.back_right_feet_pos   =   *buf++ << 8;
28   pos.back_right_feet_pos   +=  *buf++;
29
30   pos.front_left_feet_pos   =   *buf++ << 8;
31   pos.front_left_feet_pos   +=  *buf++;
32
33   pos.front_right_feet_pos  =   *buf++ << 8;
34   pos.front_right_feet_pos  +=  *buf++;
35
36   pos.second_left_feet_pos  =   *buf++ << 8;
37   pos.second_left_feet_pos  +=  *buf++;
38
39   pos.second_right_feet_pos =   *buf++ << 8;
40   pos.second_right_feet_pos +=  *buf++;
41   return pos;
42 }
```

Рисунок 3.11 – Функция приема пакета

Программа по данному протоколу будет в конце и в начале ограничивать сообщение рамками, выбранными произвольно и проверять целостность сообщения сверяя отправленную контрольную сумму целостности пакета и расчетную контрольную сумму. Контрольная сумма будет вычисляться путем суммирования всех байтов, содержащих информацию и применяя на них операцию побитового И с маской 0xFF. Каждый угол будет записан в 16-ти разрядное поле данных, где сначала будет стоят старший байт, а потом младший.

В основном цикле программы микроконтроллер будет крутить перманентную установку углов сервоприводов и читать буфер полученных сообщений как показано на рисунке 3.12.

Далее необходимо настроить бортовой компьютер Raspberry Pi. Для этого на карту памяти устанавливается rasbian Buster OS, перед первым запуском в корневом каталоге создается файл “ssh” без расширения, а также файл с конфигурацией для подключения к сети wi-fi, при включении Raspberry. Когда Raspberry подключится к сети wi-fi, нужно подключится через протокол ssh к терминалу Raspberry. Далее командой “sudo apt-get update -y && apt-get upgrade -y” производится обновление системы и командой “sudo raspi-config” вызывается меню настройки raspberry. В этих настройках устанавливается пароль на доступ по ssh и включается интерфейс камеры. После этого следует выключить компьютер и установить шлейф камеры в специализированный слот для камеры. После

повторного включения, проверки работоспособности камеры, следует установка веб-сервера mjpeg-streamer для передачи видеопотока на сервер через локальную сеть [10]. Таким образом Raspberry Pi будет транслировать на обрабатывающий компьютер видеопоток.

```
Servos_Init();
Comm_Init();
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  Servos_Handler(&pos);
  pos = Receive_and_Parse_Packet();
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Рисунок 3.12 – Основной цикл программы

Далее необходимо написать программу, которая будет установлена на Raspberry Pi, получать по локальной сети информацию по протоколу TCP/IP и перенаправлять ее по интерфейсу UART.

Для отправки пакета по интерфейсу UART, необходимо включить соответствующий интерфейс в настройках Raspberry Pi. Для реализации этого через язык программирования python, используется библиотека pyserial. Создается файл “serialcom.py”. Данный скрипт будет включать в себя функцию, через которую основная часть программы будет отправлять нужную команду в прот. В начале скрипта открывается порт UART, обращаясь к файлу по пути “/dev/ttyACM0”. После этого пишется функция отправки пакета, которая получает список углов сервопривода и конвертировать в пакет, составленный по протоколу, указанному в таблице 3.1. Код программы показан на рисунке 3.13.

Чтобы программа ожидала того, что обрабатывающий компьютер отправит ему команду, содержащую углы сервоприводов, необходимо написать программу, которая в качестве сервера привяжется к одному из портов компьютера и установит на нем связь. Для языка программирования python, используется библиотека socket, чтобы устанавливать связь по локальной сети.

Создается файл “server.py”. В него импортируется библиотека socket и файл “serialcom.py”. При запуске программа создает объект socket с необходимыми параметрами, указывающими тип связи. Далее socket привязывается к порту 1234 и устанавливается в режим прослушивания с помощью метода listen. В цикле программы сервер принимает сообщение в виде строки, парсит ее в список и передает в функцию отправки пакета по UART. Код программы показан на рисунке 3.14.

```python
import serial

port = serial.Serial("/dev/ttyACM0", baudrate=115200, timeout=3.0)


def push_position(data):
    if len(data) != 6:
        print('Format error')
        return

    frame = 22
    cs = 0
    command = [frame]
    for angle in data:
        HighBData = angle >> 8
        cs += HighBData
        command.append(HighBData)

        LowBData = angle & 0xFF
        cs += LowBData
        command.append(LowBData)

    cs &= 0xFF
    command.append(cs)
    command.append(frame)
    port.write(serial.to_bytes(command))


def deinit_serial():
    port.close()
```

Рисунок 3.13 – Листинг программы “serialcom.py”

```python
import socket
import serialcom

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('', 1234))
s.listen(5)


def run_server():
    while True:
        clientsocket, address = s.accept()
        print(f'Connection from {address}')
        clientsocket.send(bytes("Welcome to the server", "utf-8"))
        msg = clientsocket.recv(1024)
        print(msg)
        if msg:
            data = msg.split(b',')
            data = [int(angle) for angle in data]
            print(data)
            serialcom.push_position(data)
        clientsocket.close()


if __name__ == '__main__':
    run_server()
```

Рисунок 3.14 – Листинг программы “server.py”

Со стороны обрабатывающего компьютера будет написана программа, которая в качестве клиента будет подключатся к порту Raspberry Pi и отправлять сообщение с углами сервоприводов. Необходимо, чтобы Raspberry Pi и обрабатывающий компьютер были в одной локальной сети.

В программе на обрабатывающем компьютере будет создан файл “client.py”. В нем через ту же библиотеку sockets, установится соединение с бортовым компьютером. Этот файл будет иметь функцию отправки сообщения, которой будет пользоваться программа, принимающая решения.

Основной задачей обрабатывающего компьютера является обработка видеопотока с камеры робота и по данной информации принятие решения в какую позицию ставить робота. Для обработки видеопотока используется алгоритм классификации и нахождения объектов в кадре, а именно открытый репозиторий yolov3-tf2 [6]. По инструкции со странички проекта устанавливаются нужные библиотеки, такие как tensorflow-gpu, opencv, absl и т.д. После удачного испытания демонстрации работы программы, код частично копируется в проект дипломной работы и в основном применяется для классификации объектов в кадре видеопотока. Файл, включающий в себя все функции по обработке видеопотока, называется “processing.py”. В этом файле создается функция классификации, которая, используя встроенную в библиотеку opencv функцию захвата кадра со странички mjpeg_streamer-а “VideoCapture()”, принимает кадр и передает его в объект yolo, который в свою очередь возвращает рамки и они рисуются прямо на принятом кадре. Далее уже кадр с рамками выводится в окно opencv. Этот процесс выполняется в цикле. Листинг программы можно посмотреть в приложении Е.

Теперь необходимо написать часть программы, которая будет обучать мобильного робота ходить. В качестве политики был выбран алгоритм градиента политики, поэтому именно этот алгоритм нужно реализовать на языке программирования python. Большая часть кода, реализующая эту политику, будет списана с книги Орельен Жерона “Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow” второго издания [11].

Создается новый файл “environment.py”. Этот файл содержит класс, который будет создавать объект среды. Класс среды имеет методы инициализации, получения информации с камеры, отправки задач агенту, условия конца сессии и главное метод step, который будет вызываться политикой, каждый раз при выполнении какого-либо действия. Этот метод получает переменную action и возвращать наблюдение среды в виде списка, награду и статус сессии. Переменная action будет хранить в себе информацию, которую класс среды выполнит и произведет наблюдение.

Для реализации политики создается файл “policy.py”. В файле реализуется алгоритм градиента политики. Инициализируется нейронная сеть с помощью библиотеки Keras (рисунок 3.15).

```
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Dense(10, activation='elu', input_shape=[3]),
    keras.layers.Dense(6, activation='sigmoid'),
])
```

Рисунок 3.15 – Инициализация модели нейронной сети

Далее пишется функция, которая будет выполнять один шаг. После шага записывается на время значение потери и градиента. Блок GradientTape() позволяет запоминать векторы-градиенты (рисунок 3.15).

```
def play_one_step(env, obs, model, loss_fn):
    with tf.GradientTape() as tape:
        left_proba = model(obs[np.newaxis])
        action = (tf.random.uniform([1, 1]) > left_proba)
        y_target = tf.constant([[1.]]) - tf.cast(action, tf.float32)
        loss = tf.reduce_mean(loss_fn(y_target, left_proba))
    grads = tape.gradient(loss, model.trainable_variables)
    obs, reward, done, info = env.step(int(action[0, 0].numpy()))
    return obs, reward, done, grads
```

Рисунок 3.15 – Функция одного шага

Эту функцию по алгоритму градиента политики необходимо выполнять поэпизодно и после каждого эпизода выбирать правильные варианты градиенты действий. В конце всех эпизодов формируется список оптимальных градиентов (рисунок 3.16).

```
def play_multiple_episodes(env, n_episodes, n_max_steps, model, loss_fn):
    all_rewards = []
    all_grads = []
    for episode in range(n_episodes):
        current_rewards = []
        current_grads = []
        obs = env.reset()
        for step in range(n_max_steps):
            obs, reward, done, grads = play_one_step(env, obs, model, loss_fn)
            current_rewards.append(reward)
            current_grads.append(grads)
            if done:
                break
        all_rewards.append(current_rewards)
        all_grads.append(current_grads)
    return all_rewards, all_grads
```

Рисунок 3.16 – Функция проигрывания шагов по эпизодам

Эта функция также будет выполнятся несколько раз. В конце всех итераций программа получит все значения наград и прогонит их через функции дисконтирования и нормализации. Эти функции принимают значения нормы дисконтирования. Норма дисконтирования – это значение которые будет влиять на дальновидность политики агента. При значении 1 политика будет воспринимать долгосрочную выгоду как краткосрочную, то есть не различать их. Чем меньше это значение, тем менее политика будет дальновидной. Расчет наград с учетом дисконтной ставки производится функцией, показанной на рисунке 3.17.

```
def discount_rewards(rewards, discount_factor):
    discounted = np.array(rewards)
    for step in range(len(rewards) - 2, -1, -1):
        discounted[step] += discounted[step + 1] * discount_factor
    return discounted
```

Рисунок 3.17 – Функция расчет наград с учетом дисконтирования

Функция расчет дисконтирования будет выполняться из функции нормализации (рисунок 3.18).

```
def discount_and_normalize_rewards(all_rewards, discount_factor):
    all_discounted_rewards = [discount_rewards(rewards, discount_factor)
                              for rewards in all_rewards]
    flat_rewards = np.concatenate(all_discounted_rewards)
    reward_mean = flat_rewards.mean()
    reward_std = flat_rewards.std()
    return [(discounted_rewards - reward_mean) / reward_std
            for discounted_rewards in all_discounted_rewards]
```

Рисунок 3.19 – Функция нормализации

Теперь можно определить все нужные константы, такие как количество итерации проигрывания шагов по эпизодам, количество эпизодов, максимальное количество успешный шагов, дисконтная ставка. Также инициализируется функция потерь и функция оптимизации. Для оптимизации будет использоваться алгоритмом Adam (рисунок 3.20).

```
n_iterations = 150
n_episodes_per_update = 10
n_max_steps = 200
discount_factor = 0.95
optimizer = keras.optimizers.Adam(lr=0.01)
loss_fn = keras.losses.binary_crossentropy
```

Рисунок 3.20 – Инициализация констант и функций

Сам процесс тренировки происходит в цикле. В каждом цикле проигрываются эпизоды, награды после всех эпизодов проходят через

нормализацию и дисконтирование. Далее программа проходит по всем тренируемым переменным и вычисляются веса для этих переменных. В конце каждой итерации полученные веса передаем оптимайзеру. После этого остается только воспользоваться полученной моделью для предсказания нужных действий для движения мобильного робота к объекту (рисунок 3.21). Листинг всего скрипта приведен в приложении Ж

```
print('start training')
env = Environment()
for iteration in range(n_iterations):
    all_rewards, all_grads = play_multiple_episodes(
        env, n_episodes_per_update, n_max_steps, model, loss_fn)
    all_final_rewards = discount_and_normalize_rewards(all_rewards,
                                                       discount_factor)
    all_mean_grads = []
    for var_index in range(len(model.trainable_variables)):
        mean_grads = tf.reduce_mean(
            [final_reward * all_grads[episode_index][step][var_index]
             for episode_index, final_rewards in enumerate(all_final_rewards)
                for step, final_reward in enumerate(final_rewards)], axis=0)
        all_mean_grads.append(mean_grads)
    optimizer.apply_gradients(zip(all_mean_grads, model.trainable_variables))

env.close()
print('training ends')
frames = render_policy_net(model)
```

Рисунок 3.21 – Цикл, тренирующий модель

Данная модель при своей простоте может эффективно обучать мобильного робота ходьбе при необходимом количестве повторений тренировок. Мобильный робот был помещен в условия тестирования. Тренировки, в итоге которых робот обучался, успешно доказали свою эффективность. Фотоотчет тестирования показан на рисунке 3.22.

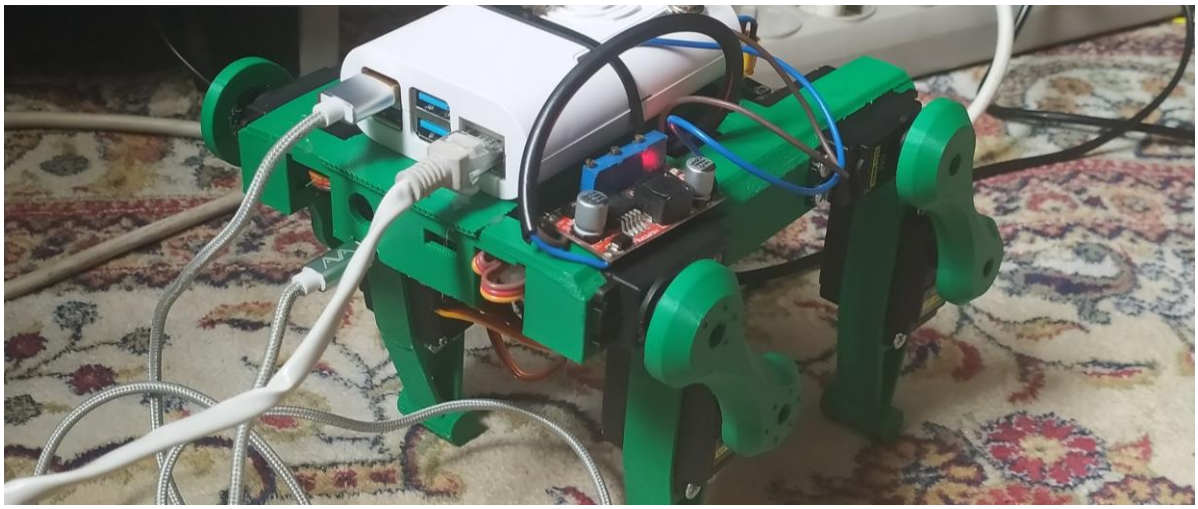

Рисунок 3.22 – Мобильный робот во время тестирования

4 Безопасность жизнедеятельности

4.1 Анализ вредных факторов на производстве

Для обеспечения безопасностью сотрудников производства, следует проанализировать условия труда и выделить особенно опасные рабочие факторы, на которые если не обратить особое внимание, они приведут к плохим последствиям, а в худшем случае и летальным. Потенциально такими факторами являются следующие:

- опасность получить ранение от станков с двигающимися частями, развивающими большую скорость;
- опасность поражения током;
- опасность отравления вредными веществами, выделяющимися в процессе пайки;
- опасность развития болезней от фонового шума, выделяемого работой охладительных систем персональных компьютеров.

Каждый пункт из данного списка подлежит тщательному анализу и промежуточному выводу о том, насколько тот или иной фактор является опасным.

4.1.1 Опасность получить ранение от станков

На предприятиях для изготовления изделий будет применятся фрезерный станок Advercut K6090T 4A со шпинделем мощностью в 2,2 кВт [12]. Данное оборудование опасно большой скоростью вращения шпинделя, и при неправильной эксплуатации станка, есть риск ранения оператора об шпиндель станка, вылет изделия из станка и попадание в сотрудника. Случаи травматизма следует регистрировать и анализировать, для получения контроля над ситуацией. Это достигается путем анализа методами статическим, групповым, монографическим и типографским [13]. Так как с объектом, представляющим опасность, будет работать один оператор в лице инженера САПР, анализ травматизма следует производить статическим методом. Для полного исключения опасности ранения сотрудника от станка, следует приобрести данную модель станка с модификацией в виде ЧПУ, что обезопасит оператора станка. Станок следует установить в отдельном помещении, чтобы не допустить случая травмы вследствие вылета изделия из станка. Также требуется систематически проводить инструктаж по применению станка. При начале работы провести вводный инструктаж, периодично раз в три месяца проводить повторный инструктаж и раз в год проводить проверку знаний правил безопасности оператору станка [13]. При соблюдении данных правил сотрудники будут обеспечены безопасностью, поэтому данный фактор следует отметить, как контролируемый и не опасный.

4.1.2 Опасность поражения током.

Так как на предприятии присутствует оборудование, питающееся высоким напряжением, следует учесть данный фактор. Опасность

поражения током может повлечь серьезные последствия. На таблице 4.1 показаны последствия [13].

Таблица 4.1 – Характер поражения в зависимости от значения и рода электрического тока, проходящего через тело человека

I, мА	При переменном токе 50 Гц	При постоянном токе
0,6-1,5	Возникновение ощущения, легкое дрожание пальцев рук	Не ощущается
5 – 7	Судороги в руках	Возникновение ощущения, нагревания кожи
8–10	Руки трудно, но еще можно оторвать от электродов; сильные боли в кистях и предплечьях	Усиление нагревания
20-25	Руки парализуются, оторвать их от электродов невозможно, дыхание затруднено	Незначительное сокращение мышц
50-80	Остановка дыхания, начало фибрилляции сердца	Сильное нагревание, сокращение мышц рук, затрудненное дыхание
90-100	Остановка дыхания и сердечной деятельности (при длительности воздействия более 3 с.)	Остановка дыхания

Основными источниками опасности на предприятии будут фрезерный станок и паяльные станции. Следует установить в электрощиток отдельный автоматический выключатель для фрезерного станка и отдельно для питания персональных компьютеров и остального оборудования. Также в электрощиток обязательно следует установить автоматический выключатель дифференциального тока, для обеспечения персонала безопасностью. Данное оборудование питается от сети напряжением в 220 В, что классифицируется как оборудование, использующее менее 1000 В, поэтому оно не требует дополнительных мер безопасности и специализированного обслуживания [14].

Тем не менее оборудование имеет металлические корпуса и не исключена ситуация попадания провода фазы на корпус оборудования. Во избежания опасных последствий в каждой модели оборудования корпуса выводятся к земле через зелено-синий провод в кабеле питания. Для того, чтобы воспользоваться этим, требуется обеспечить помещение предприятия заземлением и подключить его к каждой розетке на территории. Важным критерием заземления является его сопротивление. Данную характеристику необходимо рассчитать.

4.1.3 Опасность отравления вредными веществами

При производстве мобильных роботов инженерам-сборщикам предстоит большой объем работы, связанный с процессом пайки. При пайке электронных приборов применяется флюс для более крепкого сцепления элементов сборки. Этот флюс может выделять вредные вещества и пагубно влиять на организм человека. Особенно опасный вид флюса является паяльная кислота. Паяльная кислота имеет свойства, которые растворяют слои жировой пленки на металлических поверхностях. Такое токсичное свойство также влияет и на человека. Поэтому не допустимо попадание кислоты на кожу человека, а также после применения следует проветривать помещение. Так как процесс пайки не автоматизирован и будет осуществляться сотрудниками, фактор вредного влияния будет присутствовать на производстве. Поэтому следует рассчитать эффект вредного влияния от процесса пайки.

4.1.4 Опасность развития болезней от фонового шума

Рабочие персональные компьютеры будут работать в течении всего рабочего дня и очень важно, чтобы шум, издаваемый ими, не вызывал пагубный эффект на сотрудников. В системном блоке компьютера основным источником шума, являются кулер центрального процессора и кулеры видеопроцессора. В бизнес-плане дипломного проекта, в таблице 5.1 приведена смета компьютеров, которые будут использоваться на предприятии. Из списка 4 персональных компьютера будут обладать активными системами охлаждения только 2 компьютера, остальные будут охлаждаться пассивными системами, поэтому не будут генерировать шум. В компьютерах с активными системами охлаждения для рассеивания большой температуры процессоров будет использоваться система охлаждения NZXT Kraken M22, у которой максимальный уровень шума вне корпуса компьютера достигает максимум 36 дБ [15]. В качестве безопасного уровня шума, при котором человек способен работать на протяжении 8 часов без пагубного влияния на здоровье человека, принято 80 дБ. Поэтому фактор шума от персональных компьютеров можно считать неопасным.

4.2 Расчет вредных веществ выделяющихся при пайке изделий

При работе в производственном помещении по изготовлению роботов использование инструментов для пайки является одним из основных занятий, где необходимо соблюдать норм по обеспечению защиты сотрудников предприятия.

В создании роботов используют большое количество монтажных изделий, так как помимо корпуса разрабатываются и устанавливаются внутренние механизмы для правильной работы приборов.

В процессе создания робота будут использованы свинцово-оловянные припои (ПОС) 30 марки. Выбор марки припоя зависит от применения припоя и температуры плавления припоя. Также при расчете

необходимо учитывать вредные пары от использования канифоли. При разработке одного робота может быть использовано 2,5 грамм канифоли. Сборка робота может занять 5 рабочих дней, из которых каждый день будет выделятся по 4 часа на пайку изделий.

Количество выделения вредных веществ в процессе пайки робота находится по формуле (4.2.1).

$$G = G_{ПОС} + G_{Кан} + G_{Кис} \quad (4.2.1)$$

где $G_{ПОС}$ - количество вредных выделений от свинцово-оловянного припоя, г;

$G_{Кан}$ - количество вредных выделений от канифоли, г;

$G_{Кис}$ - количество вредных веществ от кислоты, г.

Найдем общее количество вредных выделений от припоя за один рабочий день по формуле (4.2.2).

$$G_{ПОС} = q_{ПОС} \times t \quad (4.2.2)$$

где q - удельное выделение вредного вещества, г/ч;

t - время работы с припоем, ч.

Значение удельного выделения вредного вещества зависят от марки припоя. Расчет выделения свинца и олово приводятся отдельно и в данном случае просто суммируются. Для марки припоя ПОС-30 [16]:

$$q_{ПОС} = 0{,}0028 + 0{,}0012 = 0{,}0040 \text{ г/ч}$$

Тогда:

$$G_{ПОС} = 0{,}0040 \times 4 = 0{,}0016 \text{ г}$$

Определим максимальный выброс припоя в секунду во всем производственном помещении:

$$M_{ПОС} = \frac{G_{ПОС} \times n}{t \times 3600} \quad (4.2.3)$$

$$M_{ПОС} = \frac{0.0016 \times 2}{4 \times 3600} = 2{,}23 \times 10^{-7}$$

Найдем общее количество вредных выделений от паров канифоли:

$$G_{Кан} = Q_{Кан} \times q_{Кан} \quad (4.2.4)$$

где $Q_{Кан}$ - количество использованной канифоли, г;

$q_{Кан}$ - процент количество паров канифоли при пайке, %.

Количество паров канифоли составляет примерно 20% от количества использования самой канифоли. Так как 2,5 г канифоли используется при построении одного робота (20 часов), то за один рабочий день (4 часа) используется:

$$G_{Кан} = 0,5 \times 0,20 = 0,1 \text{ г}$$

Максимальный выброс от паров канифоли определяется по формуле (4.2.5).

$$M_{Кан} = \frac{G_{Кан} \times n}{t \times 3600} \qquad (4.2.5)$$

$$M_{ПОС} = \frac{0.5 \times 2}{4 \times 3600} = 6,95 \times 10^{-5} \text{ г/с}$$

Общее количество вредных выделений кислоты при использовании канифоли вычисляется по формуле (4.2.6).

$$G_{Кис} = Q_{Кан} \times q_{Кис} \qquad (4.2.6)$$

где $q_{Кан}$ - процент количество паров кислоты, %.

Процент выделения паров кислоты, содержащийся в канифоли составляет 35%.

$$G_{Кис} = 0,5 \times 0,35 = 0,15 \text{ г}$$

Максимальный выброс паров кислоты:

$$M_{Кис} = \frac{G_{Кис} \times n}{t \times 3600} \qquad (4.2.7)$$

$$M_{Кис} = \frac{0,875 \times 2}{4 \times 3600} = 1,22 \times 10^{-4} \text{ г/с}$$

По формуле (4.2.1) найдем общее выделение вредных веществ от процесса пайки в производственном помещении:

$$G = 0,0016 + 0,1 + 0,15 = 0,2516 \text{ г}$$

При содержании в воздухе рабочей зоны вредных веществ отношение концентрации этих веществ к их предельно допустимому концентрату не должно превышать 1. [17]

$$\frac{К}{ПДК} \leq 1 \qquad (4.2.8)$$

Расчет концентрации вредных веществ в воздухе рабочего помещения:

$$К = \frac{G}{V} \qquad (4.2.9)$$

где V - объем рабочей зоны.

Параметры рабочей зоны составляют 30 м × 20 м × 5 м. Тогда:

$$К = \frac{0{,}2516}{3000} = 0{,}084 \text{ мг/м}^3$$

Предельно допустимая концентрация в рабочей зоне при работе со свинцово-оловянным припоем составляет 10 мг/м^3. [18]

$$\frac{0{,}084}{10} = 0{,}0084$$

Согласно выражению, указанному выше, норма концентрации вредных веществ от припоя в воздухе не превышает 1, что говорит о безопасной среде для работы сотрудников предприятия.

4.3 Расчет заземления

При проектировании помещения, где будут работать сотрудники, очень важно учесть фактор заземления, т.к. оборудование, питающееся от высоких напряжений при отсутствии заземления, может подать ток на человека и повлечь летальный исход. Для исключения таких ситуаций одним из решений является протягивание к помещению провода заземления и подключение к всем опасным участкам, таким как корпуса станков, бытовых приборов, паяльных станций. При попадании провода фазы на токопроводящие объекты, поверхности этих объектов остаются с высоким потенциалом, который рано или поздно будет задет сотрудником и ток пойдет через него к полу, что крайне опасно, особенно в случаях, если сотрудник заденет источник тока верхней частью тела. Производители техники, которая питается от высокого напряжения во избегание таких ситуаций специально выводят заземляющий выход, который все лишние токи тянет в землю. Такой выход помечают зелено-желтым проводом, и на вилке устройства предусмотрен специальный контакт для него. На рисунке 4.1 заземляющий контакт находится ближе к основанию вилки, между двумя штырями.

Рисунок 4.1 – Вилка электрооборудования, предусматривающая заземление

Основной целью расчет данного подраздела является вычисление таких параметров заземления как количество, размеры и порядок заземлителей. Также следует посчитать расчетный ток замыкания на землю, при условии, что в незаземленных сетях весь расчетный ток замыкается на землю [19].

Для корректной работы заземляющего устройства (ЗУ) сопротивление его должно быть достаточно низким для того, чтобы основная часть тока шла по нему, а не по человеку. Так как принято, что сопротивление человека берется как 1 кОм, а суммарная мощность всего оборудования производства не превышает 100 КВт, то ЗУ должен иметь сопротивление не более 10 Ом.

Для определения коэффициента проводимости ЗУ требуется узнать однородность почвы [20]. Поэтому используя таблицу 4.1 соотносим соответствующий условиям города Алматы тип грунта, а именно чернозем.

Таблица 4.1 – Среднее удельное сопротивление разных типов грунта

Тип грунта	Среднее удельное сопротивление, Ом×м
Известняк поверхностный	5050
Гранит	2000
Базальт	2000
Песчаник	100
Гравий однородный	800
Песчаник влажный	800
Гравий глинистый	300
Чернозём	200
Разнообразные смеси глины и песка	150
Суглинок лессовидный	100
Глина полутвердая	60
Сланцы глинистые	55
Суглинок пластичный	30
Глина пластичная	20
Подземные водоносные слои	5

По таблице видно, что удельное сопротивление черноземной почвы $\rho_{изм}$ = 200 Ом × м.

По формуле (4.3.1) считаем расчетное удельное сопротивление.

$$\rho_{расч} = \rho_{изм} \times \psi \quad (4.3.1)$$

где ψ – коэффициент сезонности.

Для определения коэффициента сезонности, прежде определим климатическую зону и проведем анализ характеристик зон на таблице 4.2. [21]

Таблица 4.2 – Признаки климатических зон для определения коэффициентов сезонности

Характеристика климатической зоны	Климатические зоны			
	I	II	III	IV
Средняя многолетняя низшая температура (январь), °С	От –20 до –15	От –14 до –10	От – 10 до 0	От 0 до +5
Средняя многолетняя высшая температура (июль), °С	От +16 до +18	От +18 до +22	От +22 до +24	От +24 до +26
Среднегодовое количество осадков, см	40	50	50	30-50
Продолжительность замерзания вод, дни	190-170	150	100	0

Учитывая данные, приведенные в таблице, можно сделать вывод, что город Алматы относиться к климатической зоне под номером III. Температура в январе опускается до -4,7 °С, а в июле повышается до +23,8 °С, период замерзания воды продолжается до 67 суток, а среднегодовое количество осадков достигает значения 547 мм. Для получения коэффициента сезонности обращаемся к таблице 4.3

Таблица 4.3 – Коэффициенты сезонности

Климатическая зона	Влажность земли во время измерений ее сопротивления		
	Повышенной	Нормальной	Малой
Вертикальный электрод длиной 3 м			
I	1,9	1,7	1,5
II	1,7	1,5	1,3
III	1,3	1,3	1,2
IV	1,3	1,1	1,0

Так как влажность земли города Алматы является повышенной, в качестве коэффициента сезонности берем значение 1.3. Используя формулу (4.3.1), вычисляется расчетное удельное сопротивление.

$$\rho_{расч} = 200 \times 1{,}3 = 260 \text{ Ом} \times \text{м}$$

Следует рассчитать сопротивление стержневого заземления, которое имеет форму трубы небольшого диаметра. Для этого применим формулу (4.3.2).

$$R_{CO} = 0{,}366 \times \frac{\rho}{l}\left(\lg\frac{2l}{d} + \frac{1}{2}\lg\frac{4h+1}{4h-1}\right) \qquad (4.3.2)$$

где ρ - расчетное удельное сопротивление почвы города Алматы, Ом×м;

l - длина трубы заземлителя, м;

d - диаметр трубы заземлителя, м;

h - глубина размещения трубы заземлителя, м.

Глубину размещения заземления рассчитывает формула (4.3.3).

$$h = 0{,}5 \times l + h_0 \qquad (4.3.3)$$

где h_0 - расстояние от начала заземлителя до поверхности почвы (длина которого может быть от 0,5 до 0,8 м).

Найдем глубину размещения, при длине стержня заземлителя 3м, с диаметром 0,06 м и расстоянием от поверхности почвы 0,7 м:

$$h = 0{,}5 \times 3 + 0{,}7 = 2{,}2 \text{ м.}$$

Тогда сопротивление одиночного стержневого заземлителя будет равно:

$$R_{CO} = 0{,}366 \times \frac{260}{3}\left(\lg\frac{2\times3}{0{,}06} + \frac{1}{2}\lg\frac{4\times2{,}2+1}{4\times2{,}2-1}\right) = 63{,}49 \text{ Ом}$$

Определим количество штук заземлителей по формуле (4.3.4).

$$n = \frac{R_{CO}}{R_H} \qquad (4.3.4)$$

$$n = \frac{63{,}49}{10} \approx 7 \text{ шт}$$

Заземлители будут устанавливаться одной вертикальной группой. Для расчета более точного значения количества ЗУ используется таблица 4.4.

Таблица 4.4 – Коэффициенты использования вертикальных электродов

Число заземлителей	Отношение расстояния между стержнями к длине заземлителей (a/l)		
	1	2	3
2	0,85	0,91	0,94
4	0,73	0,83	0,89
6	0,65	0,77	0,85
10	0,59	0,74	0,81
20	0,48	0,67	0,76

Каждое ЗУ имеет свой радиус, в котором земля воздействует на сопротивление ЗУ электрода. Этот радиус в приблизительном значении 2,2 м и будет половиной расстояния между ЗУ, то есть расстоянием между ЗУ будет 4,5 м. Поделив это расстояние на длину штыря, получим нужное отношение.

$$a/l = 4{,}5/3 = 1{,}5 \text{ м}$$

Коэффициент использования вертикальных электродов будет $\eta_{ИВ}$ = 0,70. Формула (4.2.5) определяет точное количество ЗУ.

$$n = \frac{R_{CO}}{R_H \times \eta_{ИВ}} \qquad (4.2.5)$$

$$n = \frac{63{,}49}{10 \times 0{,}70} \approx 10 \text{ шт}$$

Далее необходимо определить сопротивление полосового заземления, используя формулу (4.3.6).

$$R_{ПЗ} = 0{,}366 \times \frac{\rho}{L} \times \lg \frac{2L^2}{b \times h_0} \qquad (4.3.6)$$

где L - длина полосового заземления, м;
b - ширина полосового заземления, м.

Длина находится по формуле (4.3.7).

$$L = 1{,}05 \times n \times a \qquad (4.3.7)$$

$$L = 1{,}05 \times 10 \times 4{,}5 = 47{,}25 \text{ м}$$

Ширину полосного заземления будет равной 0,04 м. Соответственно используем формулу (4.2.6) с полученными значениями.

$$R_{ПЗ} = 0{,}366 \times \frac{260}{47{,}25} \times \lg \frac{2 \times 47{,}25^2}{0{,}04 \times 0{,}7} = 10{,}478 \text{ Ом}$$

Также рассчитываем коэффициент использования горизонтальных электродов по таблице 4.5.

Таблица 4.5 – Коэффициенты использования горизонтальных электродов

Число заземлителей	Отношение расстояния между стержнями к длине заземлителей (a/*l*)		
	1	2	3
4	0,45	0,55	0,70
5	0,40	0,48	0,64
8	0,36	0,48	0,60
10	0,34	0,40	0,56
20	0,27	0,32	0,45

По таблице видно, что при данных количества ЗУ и отношения расстояния между ЗУ и длине ЗУ коэффициент использования горизонтальных электродов будет $\eta_{ИГ}$ = 0,685. Имея эти данные рассчитываем по формуле (4.3.8) устройства заземления.

$$R_{ОБ} = \frac{R_{СО} \times R_{ПЗ}}{R_{СО} \times \eta_{ИГ} + n \times R_{ПЗ} \times \eta_{ИВ}} \qquad (4.2.8)$$

$$R_{ОБ} = \frac{63{,}49 \times 10{,}478}{63{,}49 \times 0{,}685 + 10 \times 10{,}478 \times 0{,}70} = 5{,}69 \text{ Ом}$$

Из полученного значения можно сделать вывод, что сопротивление ЗУ будет гораздо меньше человеческого, поэтому в случае аварии ток пойдет по ЗУ и окажет минимальное вредное воздействие на человека. Данные по количеству ЗУ, глубине установки и длину горизонтальной полосы нужно утвердить, как соответствующие безопасности.

5 Бизнес-план

5.1 Резюме

Современный мировой рынок сервисных роботов получает все больше инвестиций и нуждается в молодых компаниях начинающих свое производство роботов. Финансовый план бизнес-плана наглядно показывает экономическую выгоду и эффективность инвестиционных затрат на начало производства мобильных роботов. Первое время производство будет работать на нужды страны, а в перспективе начнет экспорт в ближайшие крупные страны. Приспособленные к условиям бездорожья, сложным ландшафтам, опасности в строительных локациях, горных рудников мобильные роботы, смогут переложить на себя часть опасных работ и сэкономить деньги крупным компаниям. Экспорт продукта за границу выведет Казахстанский экспорт на новый качественный уровень. Полученные деньги уйдут на привлечение молодых специалистов, тем самым замедляя процесс утечки специалистов за границу. Рынок мобильных роботов самоорганизуется формируя интеллектуальное богатство страны. Такие условия обеспечат появление новых именитых национальных it-компаний таких как российская it-компания Яндекс, китайская Huawei и т.д. Также успех производства повлечет формирование новых лабораторий, экспертизы и исследовательских учреждений, специализированных на передовых технологиях, что в перспективе откроет большие возможности для образовательной системы и системы формированиях новых компаний базирующихся на технологиях.

5.2 Маркетинговый план

5.2.1 Описание продукции

Разработанный прототип может служить основой для производства серии более совершенных моделей, что будут успешно проданы, обеспечив дальнейшею организацию предприятия по производству изделий, разработку новых моделей и формированию нового рынка робототехнической продукции. Автономные мобильные роботы могут применяться в сферах обеспечения безопасности граждан, систем охраны, военной техники, горной промышленности, нефтяным компаниям, помощи людям с ограниченными физическими возможностями, то есть выполняют задачи автоматизированных ищейки и охранника. Помимо задач патрулирования и поиска, модификация в виде усиленного каркаса и более мощных приводов, позволит моделям переносить тяжелые грузы, что будет полезно сферах военной техники, логистики, строительства, агропромышленности. На данный момент в таком роде задач применяются прирученные животные. Собаки-спасатели занимаются поиском пострадавших, охранные собаки обучены следить за проникновением посторонних лиц не охраняемую территорию. На сегодняшний день роботам сложно передвигаться по сложному ландшафту и принимать решения без участия человека, что решается методами машинного

обучения и технологиями искусственных нейронных сетей. Лишенные проблемы передвижения и несамостоятельности принятия решений, роботизированные ищейки и охранники имеют ряд преимуществ перед прирученными животными, а именно их производство не требует большой затраты времени, не требует специализированной подготовки, дополнительных затрат на воспитание животного, уход и содержание. Также роботизированные аналоги более исполнительны, имеют способности коммуникации на дальних расстояниях, записи событий и формировании отчета. Существует проблема привязки моделей к местности с доступом к электричеству, это не позволит моделям покидать станции подзарядки на большие расстояния. Тем не менее такая же проблема существует и у прирученных животных, требующих периодического корма, что привязывает их к местности облагороженную людьми. Помимо всего перечисленного, применение роботизированной техники в опасных и летальных условиях, более допустимо с точки зрения этики. В сфере помощи людям с ограниченными физическими возможностями мобильные роботы также имеют ряд преимуществ, такие как возможность сообщать локацию хозяина службам заботы, бытовая функциональность, возможность применения голосового ассистента. Но может возникнуть отторжение таких помощников старшим поколением по причине возникновения морально-этических конфликтов.

5.2.2 Анализ рынка сбыта.

В качестве покупателей продукции могут выступать юридические лица, а именно министерство обороны КР, департаменты полиции различных городов, службы спасения, поставщики систем безопасности, фонды поддержки людей с физическими ограничениями.

Рынок продажи роботов разбит на два основных рынка. Этими рынками являются рынок производственных и сервисных роботов. Данное разделение обусловлено двумя различными по природе потребителями: производством и профессиональной/бытовой сферой. Производственные роботы в большинстве случаев стационарные роботы манипуляторы. Рынок производственных роботов уже достаточно развит и уверенно увеличивает объем инвестируемых в сферу денег. На момент написания дипломной работы рынок производственных роботов в мире опережает рынок сервисных. Причина этому то, что производственным процессам в лабораторных условиях легче адаптироваться к более эффективным роботам. Тем не менее сервисная сфера масштабнее и потенциально имеет больший объем денежных средств, поэтому развитие рынка сервисной робототехники дело времени [22].

В исследованиях Международной Федерации робототехники пишется о том, что сейчас темп роста сервисной робототехники превышает рост промышленных роботов. Также в исследованиях говорится, что в 2017 году было сбыто 109 500 единиц сервисной робототехники, а в 2018 году на 50% больше. В итоге общий объем рынка составил 8,4 млрд долларов [23].

В свою очередь большей частью проданных сервисных роботов являются логистические роботы. На рисунке 5.1 показаны количества единиц проданных роботов, классифицированных по предназначению, предоставленные исследованием Международной Федерации робототехники [24].

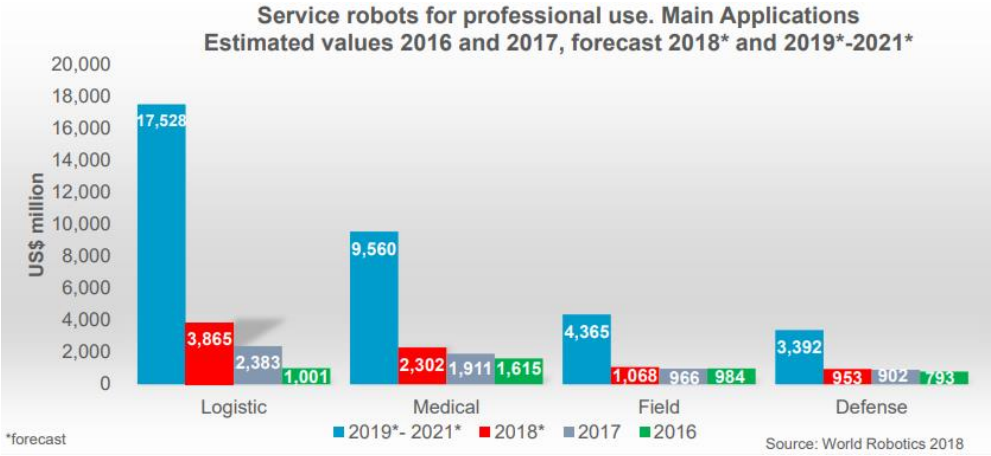

Рисунок 5.1 – Доля продаж разных классификаций робототехники

На графике видно современную и предсказанную тенденцию в популярности логистических систем.

Исходя из этих данных делается вывод, что рынок мобильных роботов имеет большую перспективу развития. Начиная работу над производством мобильных роботов сейчас, СНГ рынок и рынок Казахстана, в частности, увеличивает шанс на вступление в мировые рынки робототехники, чему сопутствует географическое приближенность к крупным странам нуждающихся в производственных и сервисных роботах. По данным все того же исследования Международной Федерации робототехники крупнейшим покупателями робототехники являются Азиатские страны [24]. График, построенный по данным исследования показан на рисунке 5.2.

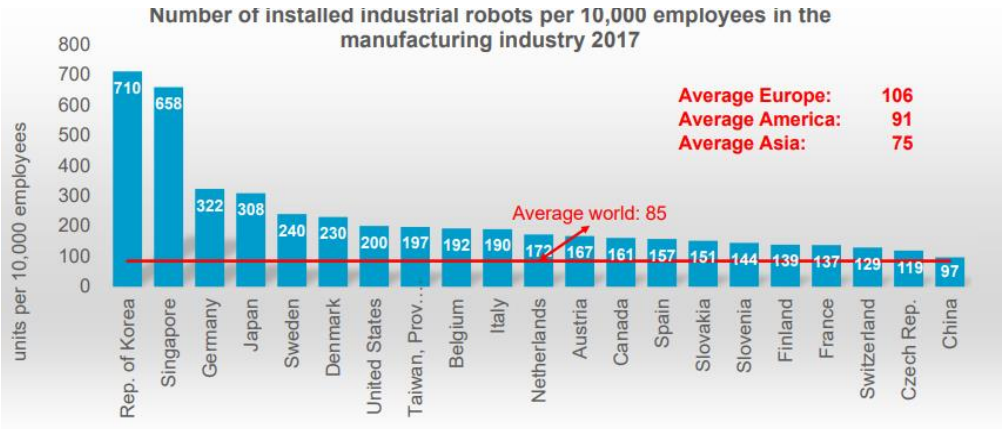

Рисунок 5.2 – Количество установленных роботизированных систем в разных странах

Сейчас известен единственный выход на коммерческий рынок мобильного робота-собаки Spot. 24 сентября 2019 года было запущены продажи робота Spot и на данный момент по словам разработчиков роботом интересуются нефтяные, строительные, электроэнергетические и горнорудные компании [25]. Ключевой особенностью робота Spot является его способность передвигаться с помощью ног. Это позволяет роботу адаптироваться к разным обстановкам и применять его в обширном списке задач.

Принимая во внимание анализируемую информацию, делается вывод, что новый быстро растущий рынок серверных роботов, состоящий в основном из мобильных роботов, имеет большую перспективу развития, что напрямую связано с экономическим обоснованием целесообразности организации производства мобильных роботов.

Маркетинговой стратегией выбирается сотрудничество с заинтересованными государственными организациями, и перспектива выходить продукта на экспорт за границу.

5.3 Финансовый план

5.3.1 Расчет инвестиционных затрат

Цель финансового плана диплома – провести расчет экономических показателей и выявить перспективы организации коммерческого производства автономных мобильных роботов. Для достижения этого требуется выполнить следующие задачи:

- расчет капитальных вложений;
- расчет расходов на производственные затраты;
- расчет доходов от реализации;
- расчет экономической эффективности.

Для организации производства мобильных роботов нужно закупить оборудование, поэтому требуется рассчитать начальные общее капиталовложения. В таблице 5.1 перечисляется нужное оборудование, стоимость и количество. Актуальные цены для формирования сметы получены из интернет-ресурса [26].

Таблица 5.1 – Смета капиталовложений для организации производства мобильный роботов

Наименование	Количество, шт	Цена за единицу в тенге	Сумма в тенге
Фрезерный станок ЧПУ Advercut K6090T (6090)	1	1913589,77	1913589,77
Монтажный стол	1	150000	150000
Персональный компьютер для разработки программного обеспечения	1	60000	60000

Продолжение таблицы 5.1

Персональный компьютер для проектирования корпусов	1	300000	300000
Персональный компьютер для разработки математических моделей интеллектуальных систем	1	500000	500000
Паяльная станция LUKEY 702	2	48400	96800
		Итого	3020389,77

Также общее общие капитальные вложения включает в себя капитальные вложения на монтажные работы и транспортные расходы, что видно в формуле (5.1).

$$\Sigma IC = Ко + Км + Ктр \quad (5.1)$$

где Ко – капитальные вложения на приобретение оборудования;

Км - капитальные вложения на монтажные работы;

Ктр- капитальные вложения на транспортные расходы (5-10% от стоимости оборудования).

Монтажные работы будут включать в себя только пусконаладку фрезерного станка с ЧПУ, который в среднем по городу Алматы обходятся в 100 тыс. тенге. В итоге общая сумма капитального вложения будет:

$$\Sigma IC = 3020389,77 + 100000 + 151000 = 3271409,22 \text{ тенге}$$

5.3.2 Расчет себестоимости товарной продукции

В сумму себестоимости изделия включаться пункты затрат, затраты на материалы и комплектующие, заработная плата работников, социальный налог, электроэнергия для производственных нужд, амортизационные отчисления, накладные затраты. В виде формулы это выглядит следующим образом (5.2):

$$\Sigma Э = ФОТ + Ос + М + Э + А + К + Н, \quad (5.2)$$

где ФОТ – фонд оплаты (основная и дополнительная заработная плата);

Ос – социальный налог (12% от ФОТ);

М – материальные затраты и запасные части (расходы на запасные части и текущий ремонт составляют 0,5% от капитальных вложений);

Э – расходы на электроэнергию;

А – амортизационные отчисления;

К – кредиты (банковские расходы);

Н – накладные расходы (косвенные расходы, сюда можно отнести все неучтенные расходы – управленческие, хозяйственные, затраты за обучение кадров, транспортные расходы). Обычно это 75 % от себестоимости.

Сумма себестоимости считается за год работы производства и является величиной годовой эксплуатационных затрат.

Чтобы рассчитать расходы на электроэнергию, нужно воспользоваться формулой (5.3)

$$Э = W \times T \times S \qquad (5.3)$$

где W – потребляемая мощность, кВт;

T – количество часов работы, ч/год;

S – стоимость киловатт-часа электроэнергии, кВт/час.

Потребляемая мощность фрезерного станка Advercut K6090T (6090) по информации поставщика [12] равна 3,3 кВт. Также для разработки математических моделей используется мощный персональный компьютер с блоком питания 0,8 кВт, персональный компьютер для проектирования корпусов имеет блок питания в 0,6 кВт, компьютер для написания программного кода не требует большой мощности и имеет блок питания 0,15 кВт. Если учесть, что компьютер для разработки математических моделей будет под нагрузкой только часть времени, потребление можно умножить на соотношение времени нагрузки и простоя. Если учитывать, что нагрузка будет занимать 40% времени разработки, то потребляемою мощностью будет:

$$0{,}8 * 0{,}4 = 0{,}32$$

Остальные два компьютеры будут под нагрузкой большую часть времени. Паяльная станция по информации от поставщика [27] потребляет 750 кВт при максимальной нагрузке. Сумма всей потребляемой мощности:

$$W = 3{,}3 + 0{,}32 + 0{,}6 + 0{,}15 + 0{,}75 * 2 = 5{,}87 \text{ кВт}$$

В итоге если учесть, что в году 1960 рабочих часов, а также что стоимость одного кВт электроэнергии в г. Алматы на момент написания дипломной работы 15,9 тенге [28], то расход на электроэнергия будет:

$$Э = 5{,}87 \times 1960 \times 15{,}9 = 182932 \text{ тенге}$$

При покупки любого дорогостоящего оборудования необходимо начать накапливать амортизационные отчисления, т.к. при использовании

любое оборудование изнашивается и тратить часть своей работоспособности на производство продукции. На момент написания дипломной работы согласно налоговому кодексу РК, 2 пункту 271 статьи, предусмотрена норма амортизации для фиксированных активов “Машины и оборудование, за исключением машин и оборудования нефтегазодобычи, а также компьютеров и оборудования для обработки информации” 25% и для “Компьютеры, программное обеспечение и оборудование для обработки информации” 40% [29], учитывая это, амортизационные отчисления нужно рассчитывают по формуле (5.4).

$$A = \frac{H_{амор т} \times C_{пер}}{100 * 12} \times \mathrm{div}\left[\frac{N}{30}\right] \times K, \qquad (5.4)$$

где $H_{амор}$ – норма амортизации;

$C_{пер}$ – изначальная стоимость оборудования;

N – количество дней эксплуатации;

К – количество единиц оборудования.

В фиксированные активы “Машины и оборудование, за исключением машин и оборудования нефтегазодобычи, а также компьютеров и оборудования для обработки информации” входят фрезерный станок ЧПУ, монтажный стол и паяльник. Их общая изначальная стоимость будет суммой всех изначальных стоимостей.

$$C_{пер.обор.}=1913589,77 + 150000 +96800 = 2160389,77 \text{ тенге}$$

Рассчитываются амортизационные отчисления (5.5):

$$A_{обор} = \frac{25 \times 2160389,77}{100 \times 12} \times \left(\frac{365}{30}\right) \times 1=547598,7959 \qquad (5.5)$$

Также рассчитываются общая сумма стоимости для приобретенных компьютеров.

$$C_{пер.комп.}=800000 + 300000 +60000=860000 \text{ тенге}$$

Также рассчитываются амортизационные отчисления.

$$A_{комп} = \frac{40 \times 860000}{100 \times 12} \times \left(\frac{365}{30}\right) \times 1=348777,7778 \text{ тенге}$$

Теперь суммируем амортизационные отчисления фиксированных активов и получаем общие.

$$A = A_{обор} + A_{комп} = 547598{,}7959 + 348777{,}7778 = 896376{,}5736 \text{ тенге}$$

Если учесть, что на начальном этапе производства мобильных роботов сборка и настройка изделий будет проводится в основном в ручную, то ожидается, что для производства одного робота потребуется 15 рабочих дня, из которых 5 дней будет потрачено на сборку шасси и 10 дней на обучение модели. Если мы разделим количество рабочих дней на количество планируемых дней производство одного изделия и округлить, то получим количество производимых изделий в год (5.6). Количество рабочих дней было взято как 246 дня [30]

$$К_и 246 / 15 \approx 16 \text{ шт} \quad (5.6)$$

Следовательно, в год можно успеть произвести 16 изделий. Так как в течение года производства планируется произвести 16 изделий, следует рассчитывать расход материалов на это количество заранее. Затраты на расходы перечислены в таблице 5.2. Актуальные цены для комплектующих взяты с интернет-ресурса [26].

Таблица 5.2 – Таблица затрат

Наименование	Количество, шт	Цена за единицу в тенге	Сумма в тенге
Лист алюминиевый 10x500x500 мм 5083H111	16	1 914,00	30 624,00
Листовой пластик 10x1000x3000 АБС	16	3 650,00	58 400,00
Пружины	64	500,00	32 000,00
Серводвигатель	192	8 400,00	1 612 800,00
Кластер из 3 Raspberry Pi 4	48	35 000,00	1 680 000,00
Ремень передачи скорости	64	840,00	53 760,00
Аккумулятор 2S10P	16	50 000,00	800 000,00
Преобразователи DC DC	32	1 000,00	32 000,00
Модуль гироскопа и акселерометра	16	3 200,00	51 200,00
Raspberrypi camera v2	16	6 341,00	101 456,00
Платформа Nucleo STM32L073RZ	16	17 350,00	277 600,00
Годовой запас паяльных материалов	1	30 000,00	30 000,00
Годовой запас крепежных материалов	1	14 000,00	14 000,00
Кабель H07VVF, 2x1.5 мм^2, метр	32	2 000,00	64 000,00

Продолжение таблицы 5.2

Кабель H07VVF, 3x1.5 мм^2, метр	32	2 400,00	76 800,00
		Ремонт	24 573,20
		Итог	4 939 213,20

В итоге годовые расходы на материалы равны сумме в 4939213,20 тенге. Также в расходы входит фонд оплаты труда. В сфере разработки очень важно иметь специалистов высокой квалификаций, поэтому в фонд оплаты будет уходить значительная сумма денег, т.к. процесс разработки сложный процесс, требующий подробного знания технологий и обширного круга навыков. Для расчета фонда оплаты нужно расписать список штата сотрудников и их заработные платы. Для разработки и сборки роботов потребуется штат разработчиков и программистов. В частности, для разработки корпуса робота будет отвечать инженер, хорошо знающий электромеханику и мехатронику. Такой специалист должен владеть ПО САПР такими как SolidWorks, Fusion 3D, AutoCAD и т.д. Также потребуется разработка высокопроизводительного ПО для шасси и управляющего устройства. Программа в условиях ограниченной вычислительной мощи должна выполнятся быстро насколько это возможно. Под такую задачу подходят языки программирования C/C++. Для эффективной разработки потребуется хотя бы два программиста. Для проектирования управляющего устройства очень важно построить математическую модель поведения, этим будет заниматься инженер анализа данных. Данный тип специалистов очень ценится, поэтому привлечение его потребует некоторых затрат. Список сотрудников приведен в таблице 5.3.

Таблица 5.3 – Список штата сотрудников и месячная заработная плата

Должность	Количество сотрудников, человек	Месячная заработная плата в тенге
Руководитель	1	20000
Data Scientist специалист	1	300000
Инженер-разработчик, программист C/C++	2	250000
Инженер САПР	1	200000
Итого	5	950000

Также при расчете фонда оплаты учитывается дополнительная заработная плата. Дополнительная заработная плата считается как 10% заработной платы (5.7).

$$ФОТ = З_{осн} + З_{доп}, \quad (5.7)$$

$$З_{доп} = З_{осн} \times 0,1 = 950000 \times 0,1 = 95000 \text{ тенге}$$

Рассчитываем годовой фонд заработной платы.

$$ФОТ = (950000 + 95000) \times 12 = 12540000 \text{ тенге}$$

Из фонда оплаты рассчитывается сумма социального налога в виде 9,5%, после вычета пенсионных отчислений (5.8).

$$О_с = 0,095 \times (ФОТ - (0,1 \times ФОТ)) \quad (5.8)$$

$$Ос = 0,095 \times (12540000 – (0,1 \times 12540000)) = 1354320 \text{ тенге}$$

Также следует учесть накладные расходы, чтобы снизить шанс оказаться в непредвиденной ситуации. По причине того, что годовые расходы является крупной суммой, в качестве суммы накладных расходов достаточно принять 5% от всей суммы. Считается вся сумма расходов за год, рассчитанная выше, и отчитывается 5% для накладных расходов.

$$С = (12540000 + 1354320 + 6154861,2 + 182932 + 896376,57) \times 0,05 = \\ = 995642,09 \text{ тенге}$$

В итоге все расходы выписаны в таблицу 5.4 и посчитана себестоимость продукта.

Таблица 5.4 – Себестоимости товарной продукции

Наименование	План, тенге	Удельный вес, %
ФОТ	12 540 000,00	60,84
Отчисления на социальные нужды	1 072 170,00	5,20
Материалы и комплектующие	4 939 213,20	23,96
Амортизационные отчисления	896 376,57	4,35
Затраты на электроэнергию	182 932,00	0,89
Накладные расходы	981 534,59	4,76
Полная себестоимость	20 612 226,36	
Планируемое количество производство изделий	16	
Себестоимость единицы продукции	1 288 264,15	

5.3.3 Расчет доходов от реализации продажи мобильных роботов

Планируемое количество проданных мобильных роботов, как показано в формуле 5.14, 16 штук в год. Для подсчета дохода от продаж,

умножаем количество планируемого количества продаж и стоимость продукта (5.9).

$$Д = N_р \times Ц, \quad (5.9)$$

где $N_р$– количество планируемых продаж изделия;

Ц – цена каждой продажи.

Так как изделие является уникальным для рынка Казахстана прибыль с изделия следует поднять до 35% от себестоимости. Таким образом цена продажи каждого изделия без учета НДС будет:

$$Ц = Э + П = 1288264{,}15 + 1288264{,}15 \times 0{,}35 = 1739156 \text{ тенге}$$

$$Д = 16 \times 1739156 = 27826505 \text{ тенге}$$

Прибыль – это разница между доходом и расходом. Если доход выше расхода, то предприятие считается прибыльным. Расчет прибыли показан ниже (5.25).

$$П = Д - \sum Э_р = 27826505 - 20612226 = 7214281 \text{ тенге}$$

Так как сумма оборота по реализации меньше 30000 МРП к цене не требуется добавлять НДС. Также прибыль облагается корпоративным налогом в виду того, что, предприятие является юридическим лицом. Корпоративный налог в РК на 2020 год составляет 20% от прибыли. После вычета налога полученная сумма называется чистой прибылью.

$$П_{чистая} = П - П \times 0{,}2 = 7214281 - 1463592 = 5771424 \text{ тенге}$$

Однако полученная сумма инвестиционного капитала не показывает важный показатель, того сколько будут стоить вложенные деньги в проект через год. Для вычисления этого показателя используется коэффициент дисконтирования a_t.

$$a_t = \frac{1}{(1 + E)^t} \quad (5.10)$$

где E – величина доходности инвестиции;

t – количество периодов времени взятые для подведения итога периода, в случае этого проекта один период равен одному году работы предприятия.

Величину доходности также называют нормой дисконта. Норма дисконта должна устанавливаться как можно надежным источником прибыли. Таким надежным источником прибыли взят банковский депозит.

В РК на момент 2020 года самый выгодный депозит, при условии высокой депозитной ставки и надежности банка, предлагает банк RBK [31]. По данному депозиту инвестиционный капитал вырастет на 14,8% при внесении депозита на 3 года [32]. Рассчитаем коэффициент дисконтирования на три года.

$$a_1 = \frac{1}{(1 + 0.148)^1} = 0,87$$

$$a_2 = \frac{1}{(1 + 0.148)^2} = 0,76$$

$$a_3 = \frac{1}{(1 + 0.148)^3} = 0,66$$

После расчета всех необходимых экономических показателей нужно занести их в одну таблицу (таблица 5.5).

Таблица 5.5 – Экономические показатели проекта

Наименование показателей	2020 г.	2021 г.	2022 г.
Доходы (Дреал), млн тг	27,826505	27,826505	27,826505
Текущие затраты (Эр.), млн тг	20,61	20,61	20,61
Амортизационные отчисления, млн.тг	0,90	0,90	0,90
Прибыль от реализаций, млн.тг	7,21	7,21	7,21
Корпоративный налог, %	20	20	20
Чистая прибыль, млн тг	5,77	5,77	5,77
Норма дисконта, (r), %	0,148	0,148	0,148
Инвестиционный капитал, млн тг	3,02038977	2,12	1,227636623

5.3.4 Расчет экономической эффективности

Экономическая эффективность рассчитывается статическими и динамическими методами. Отличие в том, что динамические методы более глубоко подходят к расчету с учетом фактора времени. Учитывается время прихода денег в проект и время оттока денег на нужды предприятия. Для дипломного проекта проведем методы расчета эффективности двумя типами методов.

Для расчета статическим методом, нужно найти такие показатели как инвестиционные затраты (К, IC), чистый поток денежных средств (CF), простая норма прибыли (ARR), срок окупаемости (PP) [33]. Простая норма прибыли – это показатель того какое количество прибыли будет получено на каждую единицу тенге. Вычисляется как отношение чистого потока прибыли к инвестиционным затратам (5.11).

$$ARR=\frac{CF}{IC} * 100\% = 191\% \quad (5.11)$$

Срок окупаемости определяется как время, за которое предприятие окупит вложенные в него инвестиции. Рассчитывается как отношение инвестиционных затрат на чистый поток прибыли в год.

$$PP = \frac{CF}{IC} = 0{,}52 \text{ года}$$

Показатели полученные статическим вносятся в таблицу 5.6.

Таблица 5.6 – Расчет показателей проекта статистическим методом

Наименование показателей	Значение
Инвестиционные затраты (К, IC), млн тг	3,02038977
Чистый поток денежных средств, (CF), млн тг	5,77
Простая норма прибыли, (ARR), %	191,0820573
Срок окупаемости (PP), лет	0,5233353745

Динамические методы расчета эффективности требуют более обширного списка показателей экономической эффективности.

Следует рассчитать чистую текущую стоимость на три года вперед умножив планируемую стоимость на коэффициент дисконтирования каждого следующего года.

$$PV_1 = CF\times a_1 = 5027371{,}87$$

$$PV_2 = CF\times a_2 = 4379243{,}79$$

$$PV_3 = CF\times a_3 = 3814672{,}29$$

Также текущую стоимость можно интерпретировать как приведенную стоимость. Расчет приведенной стоимости поможет произвести расчет чистого дисконтированного дохода. Чистый дисконтированный доход – это стоимость денежных потоков, приведенных к моменту расчета бизнес-плана. Расчет чистого дохода производится по формуле (5.12).

$$NPV = \sum_{t=1}^{n} \frac{CF_t}{(1+r)^t} - I_0 \quad (5.12)$$

Далее суммируем все приведенную стоимость и отнимаем сумму первоначальных затрат для всех трех лет.

NPV_1 = 5027371,87 - 4939213,2 = 2006982,10 тенге

NPV_2 = 5027371,87 + 4923366,32 - 4939213,2 = 6386225,89 тенге

NPV_3= 5027371,9 + 4923366,3 + 4288646,6 - 4939213,2 =
= 10200898,19 тенге

Также следует произвести расчет индекса рентабельности, который является показателем того сколько величины современного денежного потока требуется потратить для получения предлагаемой инвестиции [33]. В виде формулы это выглядит следующим образом (5.13):

$$PI = \sum_{t=1}^{n} \frac{CF_t}{(1+r)^t} / I_0 \quad (5.13)$$

По формуле 5.13 рассчитываем индекс рентабельности для трех лет.

PI = 4288646,62 / 4939213,2 = 1664477,85 тенге

PI = (4923366,32 + 4288646,62) / 4939213,2 = 3114371,45 тенге

PI = (5027371,9 + 4923366,3 + 4288646,6) / 4939213,2 = 4377345 тенге

Полученные данные внесены в таблицу 5.7

Таблица 5.7 – Расчет показателей проекта статистическим методом

Наименование показателей	2020 г.	2021 г.	2022 г.
Доходы (Дреал), млн тг	27,826505	27,826505	27,826505
Текущие затраты (Эр.), млн тг	20,61	20,61	20,61
Прибыль от реализаций, млн.тг	7,21	7,21	7,21
Корпоративный налог, %	20	20	20
Амортизационные отчисления, млн.тг	0,90	0,90	0,90
Чистый поток денежных средств, (CF), млн тг	5,77	5,77	5,77
Норма дисконта, (r), %	0,148	0,148	0,148
Коэффициент дисконтирования (at)	0,87	0,76	0,66

Продолжение таблицы 5.7

Чистая текущая стоимость (PV), млн тг	5,03	4,38	3,81
Чистый дисконтированный доход (NPV), млн тг	2,01	6,39	10,20
Индекс доходности (PI), отн. ед.	1,66	3,11	4,38
Инвестиционные затраты (K, I), млн тг	3,02038977	0,00	0,00

Заключение

В проекте дипломной работы была произведена большая работа по разработке мобильного робота, обученного ходьбе методами машинного обучения. В процессе работы были преодолены трудности со сборкой механической части, монтажом микроконтроллера в систему движения робота, проблемы с преобразованием напряжения аккумулятора на требуемые оборудованием напряжения, тестированием потребления тока сервоприводами. Также в итоге разработки была спроектирована модель обучения робота и написана программа по реализации этой модели, реализации связи обрабатывающего компьютера и таймеров общего назначения, подключенных к сервоприводам. Весь написанный код проекта полностью доступен в открытом репозитории по ссылке (https://github.com/jfultr). Каждый важный комит был сохранен, поэтому можно проследить все этапы написания программы. В ходе работы получен важный опыт разработки системы, которая способна к обучению на предоставленных средой данных. Такой подход к решению задач содержит в себе большой потенциал и способен менять многое в сфере деятельности человека, делая ее безопасной и более эффективной.

В жизни человека все больше применяются технологии интеллектуального анализа, поэтому важно понимать основы технологий, на которых базируется любая интеллектуальная система. Развитие мирового рынка мобильных роботов спровоцирует появление большого интереса у потребителей и государственных, зарубежных инвесторов. Из этого следует, что интенсивные исследования этой области в недалеком будущем приведут к появлению высокого уровня экспертизы в этом направлении, что повлечет больший контроль над ситуацией на рынке и более успешным реализациям крупных компаний в сфере интеллектуального труда и высоких технологий.

Список литературы

1 Робототехника. [Электронный ресурс]. - Режим доступа: www.indicator.ru

2 Робот BigDog. [Электронный ресурс]. - Режим доступа: www.wikipedia.org

3 Искусственная нейронная сеть. [Электронный ресурс]. - Режим доступа: www.wikipedia.org

4 Характеристика сервопривода MG996R от компании TowerPro. [Электронный ресурс]. - Режим доступа: www.smartelements.ru

5 Datasheet STM32L031x4, STM32L031x6

6 Открытый репозиторий “yolov3-tf2”. [Электронный ресурс]. - Режим доступа: www.github.com

7 Уильямс, Р. Дж. (1992). Простые статистические алгоритмы слежения за градиентом для обучения усилению коннекционизма. - 1992.

8 Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО "Альфа-книга': 2018. - 688 с.: ил. - Парал. тит. англ.

9 Datasheet MG996

10 Открытый репозиторий “mjpeg-streamer”. [Электронный ресурс]. - Режим доступа: www.github.com

11 Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472

12 Станок ЧПУ Advercut K6090T4A (6090). [Электронный ресурс]. - Режим доступа: www.3dtool.ru

13 Абдимуратов Ж.С., Дюсебаев М.К., Санатова Т.С., Хакимжанов Т.Е. Охрана труда. Конспект лекций (для студентов всех форм обучения специальности 050718- Электроэнергетика). -Алматы: АИЭС,2006. -38 с.

14 М. К. Дюсебаев; З. А. Кашкарова; Ф. Р. Жандаулетова. Конспект лекций для студентов всех форм обучения по специальности 050719 - Радиотехника, электроника и телекоммуникации. – Алматы: АИЭС, 2005.- 39 с.

15 Kraken M22. [Электронный ресурс]. - Режим доступа: www.nzxt.com

16 Гриванова С.М. Методика расчета выбросов вредных (загрязняющих) веществ в атмосферу для предприятий бытового обслуживания. – Владивосток: - 2010. – 59с.

17 ГОСТ 12.1.005-88 Система стандартов безопасности труда (ССБТ). Общие санитарно-гигиенические требования к воздуху рабочей зоны.

18 ГН 2.2.5.1313-03 Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны

19 Методические указания к выполнению раздела «Электробезопасность в электроустановках» в выпускных работах для

специальности 050718 -Электроэнергетика. Бакалавриат - Алматы: АИЭС, 2009. - 24с.

20 Типы грунтов республики Казахстан и их удельные электрические сопротивления. [Электронный ресурс]. - Режим доступа: www.zandz.kz

21 Зависимость ρ грунта от времени года. [Электронный ресурс]. - Режим доступа: www.ftemk.mpei.ac.ru

22 International Federation of Robotics – Representing the global robotics industry, 2018. [Электронный ресурс]. - Режим доступа: www.ifr.org

23 Рынок сервисов и промышленности. [Электронный ресурс]. - Режим доступа: www.cia.gov

24 Аналитический обзор мирового рынка робототехники, 2019. [Электронный ресурс]. - Режим доступа: www.adindex.ru

25 Робот spot. [Электронный ресурс]. - Режим доступа: www.bostondynamics.com

26 Ресурс с ценами. [Электронный ресурс]. - Режим доступа: www.satu.kz

27 LUKEY-702. [Электронный ресурс]. - Режим доступа: www.deltachip.kz

28 Актуальные цены стоимости электроэнергии в Алматы. [Электронный ресурс]. - Режим доступа: www.informburo.kz

29 Кодекс Республики Казахстан от 25 декабря 2017 года № 120-VI «О налогах и других обязательных платежах в бюджет (Налоговый кодекс)» (с изменениями и дополнениями по состоянию на 10.01.2020 г.)

30 Производственный календарь. [Электронный ресурс]. - Режим доступа: www.online.zakon.kz

31 Статья про депозиты. [Электронный ресурс]. - Режим доступа: www.rbs.kz

32 Депозит от РБК. [Электронный ресурс]. - Режим доступа: www.bankrbk.kz

33 Г.Ш. Боканова. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В071900 – Радиотехника, электроника и телекоммуникаций. – Алматы: АУЭС, 2020 – 26 с

Приложение А

Приложение Б

```
#include "servos.h"
#include "tim.h"
extern TIM_HandleTypeDef htim2;
extern TIM_HandleTypeDef htim21;
extern TIM_HandleTypeDef htim22;
void Servos_Init(void)
{
	MX_TIM2_Init();
	MX_TIM21_Init();
	MX_TIM22_Init();
	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
	HAL_TIM_PWM_Start(&htim21, TIM_CHANNEL_1);
	HAL_TIM_PWM_Start(&htim21, TIM_CHANNEL_2);
	HAL_TIM_PWM_Start(&htim22, TIM_CHANNEL_1);
	HAL_TIM_PWM_Start(&htim22, TIM_CHANNEL_2);
}
void Servos_Handler(struct Servos_Position *pos)
{
	Set_Position(pos);
}
static void Set_Position(struct Servos_Position *pos){
	htim2.Instance->CCR1 = ZERO_ANGLE_FRONT_RIGHT   - pos-
>front_right_feet_pos;     // front right
	htim2.Instance->CCR2 = ZERO_ANGLE_BACK_LEFT     + pos-
>back_left_feet_pos;       // back left
	htim2.Instance->CCR3 = ZERO_ANGLE_BACK_RIGHT    - pos-
>back_right_feet_pos;      // back right
	htim2.Instance->CCR4 = ZERO_ANGLE_FRONT_LEFT    + pos-
>front_left_feet_pos;      // front left
	htim21.Instance->CCR1 = ZERO_ANGLE_SECOND_LEFT  - pos-
>second_left_feet_pos;     // left
	htim21.Instance->CCR2 = ZERO_ANGLE_SECOND_RIGHT + pos-
>second_right_feet_pos;    // right
	htim22.Instance->CCR1 = ZERO_ANGLE_SECOND_LEFT  - pos-
>second_left_feet_pos;     // left
	htim22.Instance->CCR2 = ZERO_ANGLE_SECOND_RIGHT + pos-
>second_right_feet_pos;    // right
}
```

Приложение В

```
#ifndef __SERVOS_H_
#define __SERVOS_H_

#include "stm32l0xx_hal.h"

#define ZERO_ANGLE_FRONT_LEFT       800
#define ZERO_ANGLE_BACK_LEFT        800
#define ZERO_ANGLE_FRONT_RIGHT     2500
#define ZERO_ANGLE_BACK_RIGHT      2400
#define ZERO_ANGLE_SECOND_LEFT     1700
#define ZERO_ANGLE_SECOND_RIGHT  1500

struct Servos_Position
{
	uint16_t back_left_feet_pos;
	uint16_t back_right_feet_pos;
	uint16_t front_left_feet_pos;
	uint16_t front_right_feet_pos;

	uint16_t second_left_feet_pos;
	uint16_t second_right_feet_pos;
};

// public
void Servos_Init(void);
void Servos_Handler(struct Servos_Position *pos);

// static
static void Set_Position(struct Servos_Position *pos);

#endif
```

Приложение Г

```
#include "communicate.h"
#include "servos.h"
#include "usart.h"

char buffer[EXPECTED_LEN] = {0};

void Comm_Init()
{
	MX_USART2_UART_Init();
	HAL_UART_Receive_IT(&huart2,(uint8_t*) buffer, EXPECTED_LEN);
}
struct Servos_Position Receive_and_Parse_Packet()
{
	struct Servos_Position pos = {0};
	if(huart2.RxXferCount == 0)
	{
		HAL_UART_Receive_IT(&huart2,(uint8_t*) buffer,
	EXPECTED_LEN);
	}

	uint8_t *buf = (uint8_t*) &buffer[1];

	pos.back_left_feet_pos    =   *buf++ << 8;
	pos.back_left_feet_pos    +=  *buf++;

	pos.back_right_feet_pos   =   *buf++ << 8;
	pos.back_right_feet_pos   +=  *buf++;

	pos.front_left_feet_pos   =   *buf++ << 8;
	pos.front_left_feet_pos   +=  *buf++;

	pos.front_right_feet_pos  =   *buf++ << 8;
	pos.front_right_feet_pos  +=  *buf++;

	pos.second_left_feet_pos  =   *buf++ << 8;
	pos.second_left_feet_pos  +=  *buf++;

	pos.second_right_feet_pos =   *buf++ << 8;
	pos.second_right_feet_pos +=  *buf++;
	return pos;
}
```

Приложение Д

```
#ifndef __COMMUNICATE_H_
#define __COMMUNICATE_H_

#define EXPECTED_LEN          15

void Comm_Init(void);
struct Servos_Position Receive_and_Parse_Packet(void);

#endif
```

Приложение Е

```
import cv2
from absl import flags
from absl.flags import FLAGS
import tensorflow as tf
import time

from yolov3_tf2.models import (
    YoloV3, YoloV3Tiny
)

from yolov3_tf2.dataset import transform_images, load_tfrecord_dataset
from yolov3_tf2.utils import draw_outputs

flags.DEFINE_integer('num_classes', 80, 'number of classes in the model')
flags.DEFINE_string('classes', './data/coco.names', 'path to classes file')
flags.DEFINE_string('weights', './checkpoints/yolov3.tf',
                    'path to weights file')
flags.DEFINE_integer('size', 416, 'resize images to')
flags.DEFINE_string('url', 'http://169.254.49.227:8080/?action=snapshot', 'url to
stream')


def classification():
    yolo = YoloV3(classes=FLAGS.num_classes)

    yolo.load_weights(FLAGS.weights).expect_partial()
    print('weights loaded')

    class_names = [c.strip() for c in open(FLAGS.classes).readlines()]
    print('classes loaded')

    while True:
        start = time.time()
        cap = cv2.VideoCapture(FLAGS.url)
        ret, img_raw = cap.read()
        if ret:
            frame = tf.expand_dims(img_raw, 0)
            frame = transform_images(frame, FLAGS.size)
            boxes, scores, classes, nums = yolo(frame)
            frame = cv2.cvtColor(img_raw, cv2.COLOR_BGR2GRAY)
```

```
        frame = draw_outputs(frame, (boxes, scores, classes, nums),
class_names)
        cv2.imshow('gray', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    print(time.time() - start)

cap.release()
cv2.destroyAllWindows()
```

Приложение Ж

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
from environment import Environment


def render_policy_net(model, n_max_steps=200, seed=42):
    frames = []
    env = Environment()
    np.random.seed(seed)
    obs = env.reset()
    for step in range(n_max_steps):
        left_proba = model.predict(obs.reshape(1, -1))
        action = int(np.random.rand() > left_proba)
        obs, reward, done, info = env.step(action)
        if done:
            break
    env.close()
    return frames


def update_scene(num, frames, patch):
    patch.set_data(frames[num])
    return patch,


def play_one_step(env, obs, model, loss_fn):
    with tf.GradientTape() as tape:
        left_proba = model(obs[np.newaxis])
        action = (tf.random.uniform([1, 1]) > left_proba)
        y_target = tf.constant([[1.]]) - tf.cast(action, tf.float32)
        loss = tf.reduce_mean(loss_fn(y_target, left_proba))
    grads = tape.gradient(loss, model.trainable_variables)
    obs, reward, done, info = env.step(int(action[0, 0].numpy()))
    return obs, reward, done, grads


def play_multiple_episodes(env, n_episodes, n_max_steps, model, loss_fn):
    all_rewards = []
    all_grads = []
    for episode in range(n_episodes):
```

```
        current_rewards = []
        current_grads = []
        obs = env.reset()
        for step in range(n_max_steps):
            obs, reward, done, grads = play_one_step(env, obs, model, loss_fn)
            current_rewards.append(reward)
            current_grads.append(grads)
            if done:
                break
        all_rewards.append(current_rewards)
        all_grads.append(current_grads)
    return all_rewards, all_grads


def discount_rewards(rewards, discount_factor):
    discounted = np.array(rewards)
    for step in range(len(rewards) - 2, -1, -1):
        discounted[step] += discounted[step + 1] * discount_factor
    return discounted


def discount_and_normalize_rewards(all_rewards, discount_factor):
    all_discounted_rewards = [discount_rewards(rewards, discount_factor)
                              for rewards in all_rewards]
    flat_rewards = np.concatenate(all_discounted_rewards)
    reward_mean = flat_rewards.mean()
    reward_std = flat_rewards.std()
    return [(discounted_rewards - reward_mean) / reward_std
            for discounted_rewards in all_discounted_rewards]


n_iterations = 150
n_episodes_per_update = 10
n_max_steps = 200
discount_factor = 0.95
optimizer = keras.optimizers.Adam(lr=0.01)
loss_fn = keras.losses.binary_crossentropy

keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
```

```
model = keras.models.Sequential([
    keras.layers.Dense(10, activation='elu', input_shape=[3]),
    keras.layers.Dense(6, activation='sigmoid'),
])

print('start training')
env = Environment()
for iteration in range(n_iterations):
    all_rewards, all_grads = play_multiple_episodes(
        env, n_episodes_per_update, n_max_steps, model, loss_fn)
    all_final_rewards = discount_and_normalize_rewards(all_rewards,
                                                       discount_factor)
    all_mean_grads = []
    for var_index in range(len(model.trainable_variables)):
        mean_grads = tf.reduce_mean(
            [final_reward * all_grads[episode_index][step][var_index]
             for episode_index, final_rewards in enumerate(all_final_rewards)
                for step, final_reward in enumerate(final_rewards)], axis=0)
        all_mean_grads.append(mean_grads)
    optimizer.apply_gradients(zip(all_mean_grads, model.trainable_variables))

env.close()
print('training ends')
frames = render_policy_net(model)
```