Творческий проект по профилю «Робототехника»

на тему:

«Робот-манипулятор “Добблинатор”. Исследование методов машинного обучения и технического зрения»

Иннополис

2024

Государственное автономное образовательное учреждение

“Лицей Иннополис”

ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ

Технология. Профиль «Робототехника»

Проект:

«Робот-манипулятор “Добблинатор”. Исследование методов машинного обучения и технического зрения»

Проект разработал

Ученик 8А класса

Каржавин Егор Владимирович

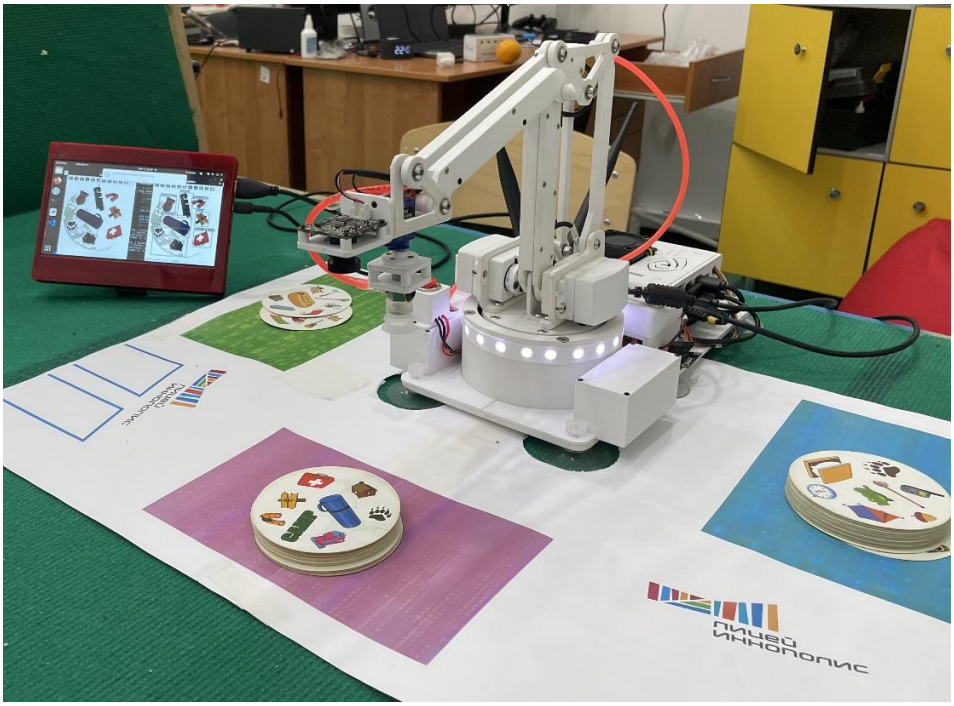
Руководитель проекта:

Михайлов Михаил Леонидович

Заместитель директора по информатизации образования

Иннополис

2024



лицей
иннополис

DOBBLINATOR

ВСОШ

ВСЕРОССИЙСКАЯ
ОЛИМПИАДА
ШКОЛЬНИКОВ

РЕФЕРАТ

Общий объем отчета составил 49 страниц, из них 46 страницы основного отчета и 5 страниц приложения с листингом кода программы. Количество иллюстраций 39, количество таблиц 4. При подготовке проекта использовано 29 источников, на которые даны ссылки по тексту отчета.

Перечень ключевых слов: робототехника, робот-манипулятор, техническое зрение, OpenCV, машинное обучение, Искусственный интеллект, Yolo, Jetson Nano, Ubuntu

Объектом исследования представленной работы является робот-манипулятор с тремя степенями свободы и камерой. Целью работы было создание действующего робота-манипулятора способного играть в настольную игру Доббль, изучение алгоритмов технического зрения OpenCV, обучения нейронной сети Yolo, практические навыки по применению библиотеки синтеза и распознавания речи Yandex Speech. В ходе работы были применены методы эмпирического уровня (тестирование, наблюдение и последующая коррекция), а также методы экспериментально-теоретического уровня.

Результатом выполнения работы стало создание робота-манипулятора который может играть с человеком в игру «Доббль» распознавая 57 различных картинок и называя их. Робот способен перемещать картонные карточки по полю, распознавать речь пользователя и отвечать в интерактивном режиме на ряд предустановленных вопросов.

Первый практический результат уже получен в виде готового робота для игры «Доббль», это робот может быть как хорошим учебным и развлекательным объектов в нашем лицее, так и, при тиражировании, в других школах страны увлекая за счет игровой динамики ребят изучать современные технологии. Дальнейшим применением результатов станет использование приобретенных знаний для процесса обучения и создания новых роботов.

Развитием данного проекта станет интеграция в робота параллельного с игрой распознавания речи игрока, а также написание полноценного графического интерфейса на базе QT библиотеки.

СОДЕРЖАНИЕ:

ВВЕДЕНИЕ

Тема творческих проектов этого года «Время созидать». Выбор робота-манипулятора в проекте "Время созидать" обоснован несколькими факторами. Во-первых, робот сделает настольные игры более интересными и увлекательными для детей, создавая атмосферу соперничества. Во-вторых, игры развивают у детей внимание, логическое мышление и командную работу, и робот будет помогать им в этом. В-третьих, использование робота в играх поможет детям понять основы робототехники и программирования. Наконец, проект способен объединить детей и взрослых, предоставив им возможность провести время весело, развивая навыки и знания.

Проблема

Проблема одиночества детей и недостатка игр, которые могли бы помочь им развиваться и проводить время с пользой, является актуальной в современном обществе. Многие дети проводят большую часть своего времени за просмотром телевизора или соц. сетей в гаджетах, что может привести к различным проблемам, таким как ухудшение зрения, нарушение осанки, снижение физической активности и развитие зависимости от электронных устройств.

Продукт (польза/выгода, которую содержит проектное решение)

Предлагаемый к разработке продукт – робот играющий в Доббль может и развлечь ребенка игрой в реальном мире и поможет детям лучше понять основы робототехники и программирования, что может стать стимулом для их дальнейшего развития в этой области.

Тема проекта

Создание действующего робота-манипулятора способного играть в Доббль и изучение алгоритмов компьютерного зрения и машинного обучения на данном примере.

Глава 1 Теоретическое исследование

1.1 Обоснование актуальности. Формулировка цели и задач, результата и выводов

Актуальность.

В рамках общей темы "Время созидать" я решил создать проект «Добблинатор» с роботом-манипулятором, играющем с ребенком в настольную игру "Доббль". Выбор такой темы обосновывается несколькими важными аспектами:

Развитие социальных навыков: Игра в настольные игры способствует развитию социальных навыков у детей. Робот-манипулятор, участвующий в игре, может стать эффективным средством для обучения детей взаимодействию, сотрудничеству и разрешению конфликтов.

Стимуляция умственного развития: Игра в "Доббль" требует от игроков быстрого распознавания схожих изображений. Робот, оснащенный искусственным интеллектом, может стимулировать умственное развитие ребенка, помогая ему развивать логику, внимание и концентрацию.

Технологический прогресс: Проект может служить примером интеграции передовых технологий в образовательный процесс. Робот-манипулятор, оснащенный сенсорами и искусственным интеллектом, может создать уникальный и интересный опыт обучения для детей.

Поддержка образовательных целей: Игры являются эффективным методом обучения. Проект "Время созидать" может быть спроектирован так, чтобы соответствовать образовательным стандартам и целям, способствуя развитию различных навыков у детей.

Поддержка инклюзивности: Робот-манипулятор может быть адаптирован с учетом потребностей детей с различными уровнями физической или умственной активности. Это создаст возможность для более широкого круга детей участвовать в образовательных играх.

Вдохновение для будущих технологий: Проект может служить вдохновением для детей, побуждая их интерес к науке, технологиям и робототехнике. Это может стать стимулом для выбора карьеры в области инженерии, программирования или исследований искусственного интеллекта.

Таким образом, проект "Добблинатор" не только способствует образовательному процессу, но и сочетает в себе социальные, технологические и инновационные аспекты, делая его уникальным и значимым для развития детей.

Цель – Создание действующего робота-манипулятора способного играть в Доббль и реализация алгоритмов компьютерного зрения и машинного обучения в данном проекте.

Задачи:

Создание робота-манипулятора "Добблинатор" с трехосевой робо-рукой и камерой технического зрения, оснащенной распознаванием объектов и синтезом речи, предполагает выполнение следующих задач:

1. Механическая конструкция:

- Разработка и создание трехосевой робо-руки с необходимой прочностью и гибкостью для эффективного выполнения задач игры "Доббль".

- Интеграция манипулятора с камерой технического зрения, обеспечивая их взаимодействие и эргономику.

2. Распознавание образов:

- Интеграция камеры технического зрения, работающей на платформе Jetson Nano, для съемки изображений игровых карточек.

- Настройка и обучение системы распознавания с использованием OpenCV и нейронной сети для точного определения объектов на картах. Выбор подходящей нейронной сети.

3. Синтез речи:

- Интеграция модуля синтеза речи, способного преобразовывать распознанные объекты в аудио-сигналы.

- Настройка синтезатора речи для четкого и понятного произнесения наименований обнаруженных объектов.

4. Управление и программирование:

- Разработка управляющей программы на языке Python для координации работы робота, взаимодействия с камерой и синтезатором речи.

- Программирование механизмов управления движением робо-руки, включая кинематику и обратную кинематику.

5. Интеграция с игрой "Доббль":

- Разработка механизмов взаимодействия с игровыми карточками, позволяющих роботу эффективно участвовать в игре.

- Тестирование и отладка работы "Добблинатора" в контексте игры "Доббль".

6. Оптимизация и улучшение производительности:

- Оптимизация программного обеспечения и алгоритмов для обеспечения быстрого и точного распознавания объектов.

- Улучшение системы управления и реакции робота для создания плавного и естественного взаимодействия с ребенком в ходе игры.

7. Безопасность:

- Разработка системы безопасности, включая датчики препятствий и механизмы аварийного останова, чтобы предотвратить возможные травмы.

8. Тестирование и доработка:

- Проведение тестовых игр с участием робота и детей для выявления возможных проблем и доработки системы на основе обратной связи.

Создание "Добблинатора" требует комплексного подхода, объединяющего механику, компьютерное зрение, технологии распознавания и искусственный интеллект для эффективной реализации поставленных задач.

Результат:

Создан Робот-манипулятор "Добблинатор" состоящий из трехосевой робо-руки и камеры технического зрения. Камера позволяет роботу распознавать объекты на игровых карточках игры Доббль и с помощью синтеза речи называть обнаруженный совпадающий объект. Распознавание осуществляется с помощью OpenCV и Yolo V8 на платформе Jetson nano. Управляющая программа написана на Python.

Выводы:

Проект реализован в жизнь полностью, поставленная цель по созданию робота-манипулятора "Добблинатор" с трехосевой робо-рукой и камерой технического зрения выполнена. В процессе работы над проектом применены различные академические и прикладные дисциплины такие как: математические основы компьютерного зрения (поиск характерных областей на изображении, который опирается внутри на конволюцию); практическое применение библиотеки OpenCV; машинное обучение в примере нейронной сети You Look Only once (Yolo); применение Yandex Speech; основы схемотехники и выбор подходящих платформ; 3d пространственное моделирование и аддитивная 3D печать.

Добавлено примечание ([VK1]): Добавить потом: конечная геометрия, проективные плоскости и плоскость Фано;

1.2 Сбор и анализ информации по исследуемой проблеме

1.2.1 Общая проблематика

Начнем с анализа правильности выбора основного мотива проекта, а именно игра в настольные игры и обоснуем это решение.

В современности наблюдается рост популярности настольных игр, сопровождаемый формированием субкультуры бордгеймеров или "настольщиков". Эта субкультура активно развивается и приобретает значение не только как способ проведения свободного времени, но и в образовательном контексте. Практическое исследование влияния настольных игр на социализацию подростков выявило (А. Л. Каткова, 2022), что для данной возрастной группы, в отличие от старшего поколения, формирование взаимоотношений среди сверстников представляет сложности.

Настольные игры стали эффективным средством, сглаживая трудности в установлении контактов и общении, а также способствуя развитию социальных навыков. Проведение серии настольных игр в рамках досуга подростков позволило им приобрести опыт в установлении взаимоотношений и формировании дружеских связей, что положительно сказалось на их социализации. Результаты опроса, проведенного по методике Сишора, подтвердили увеличение индекса групповой сплоченности среди участников настольных игр, указывая на положительное

влияние этих игр на развитие soft skills, способность к эффективному взаимодействию, и в целом на социализацию подростков и молодых людей.

Психологи утверждают (Лошинская Е.А., 2023), что правильно подобранные настольные игры обладают мощным развивающим эффектом, особенно когда они играются всей семьей. Этот возврат к традиционным формам развлечений связан с потерей навыков живого общения в эпоху социальных сетей, где важные вопросы обсуждаются виртуально. Настольные игры стали не только средством общения, но и популярным методом обучения, особенно в дошкольных учреждениях. Разработчики активно внедряют психологические тесты и тренировки в новые игры, что привлекает родителей, стремящихся развивать способности своих детей. Игры не только развивают память, внимание, мышление и речь, но и тренируют социальные навыки, такие как взаимодействие, эмоциональный контроль, поддержка и сопереживание.

Перейдем далее к выбору технической стороны в виде робота-манипулятора. Рынок роботов в целом в мире растет с темпом порядка 7% в год (2023)IFR World Robotics 2023 Report. Согласно данных изложенных в данном отчете основными отраслями применяющими роботов остаются производство электроники и автомобильная промышленность, однако доля роботов, применяемых в общей промышленности выросла за последние 10 лет с 38% до 47%. Основные задачи решаемые роботами это перемещение, сварка и сборка, что продемонстрировано на рисунке 1.

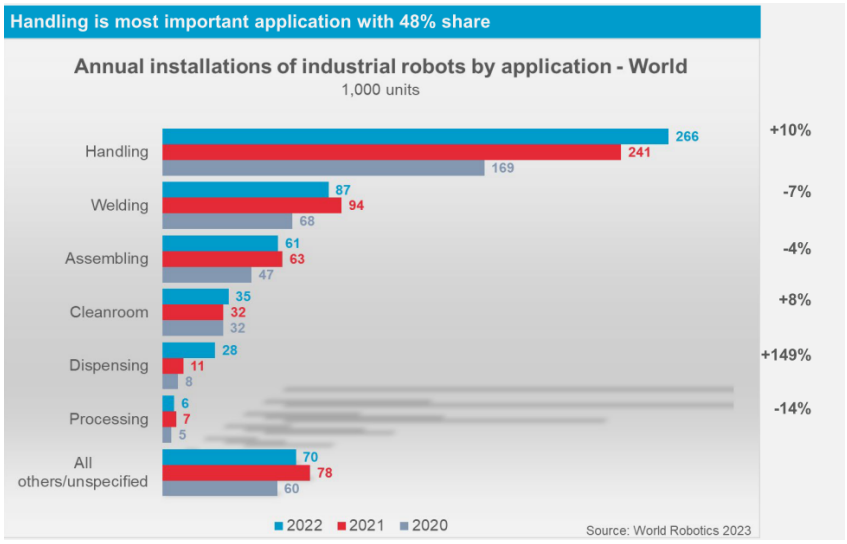


Рисунок 1. Задачи выполняемые промышленными роботами

Растет как количество так и доля коллаборативных роботов, подробнее представлено на рисунке 2.

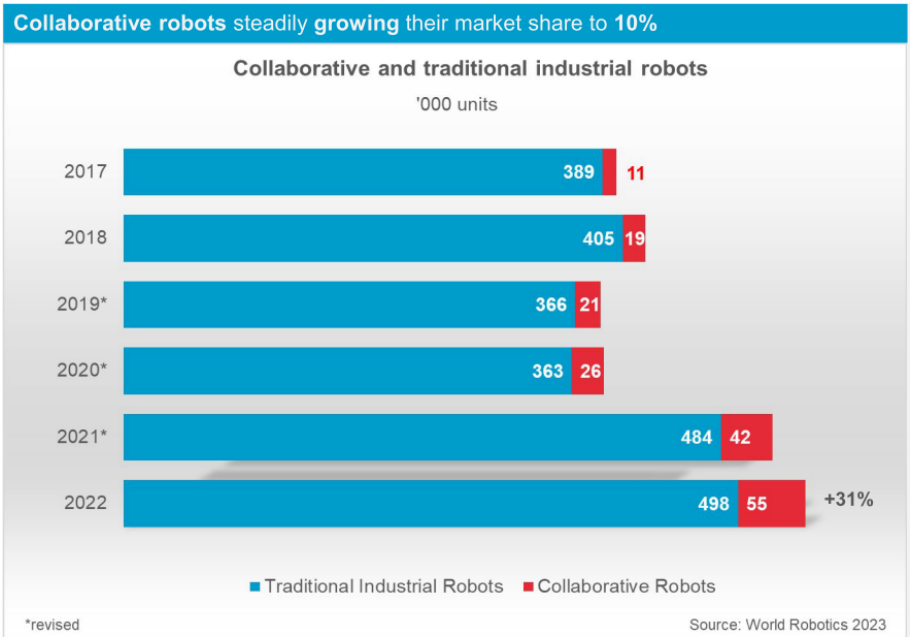


Рисунок 2. Традиционные и коллаборативные роботы в промышленности

1.2.2 Поиск прототипов и аналогов по исследуемой проблеме.

Удалось найти несколько прототипов роботов, которые также предназначены для настольных игр. Первый и наиболее интересный это Игровой робот OLEM от компании LudoTech (LudoTech, 2022), он представляет собой ультрасовременное технологическое устройство, предназначенное для того, чтобы предоставить любителям настольных игр возможность наслаждаться различными развлечениями с использованием одного универсального решения. Робот имеет компактные размеры, но при этом предлагает широкий спектр игр, которые пользователи могут выбирать через приложение на смартфоне в зависимости от числа игроков, времени и других предпочтений. Интерактивный дизайн робота позволяет ему взаимодействовать с физическими компонентами, такими как игровые доски, с применением передовых технологий для существенного улучшения опыта игрока.

Борд-игровой робот LudoTech 'OLEM' идеально подходит для игровых вечеров, позволяя пользователям ограничить количество традиционных игр, которые они берут с собой к друзьям, а также отлично подходит для использования в путешествиях.

Рисунок 3. Робот для настольных игр OLEM

Робот-компаньон для настольных игр Ludotech OLEM

- сочетает удобство виртуальности с уютным духом живых настольных игр с друзьями;
- теперь все громоздкие коробки с играми умещаются в одном роботе и приложении;
- объясняет правила, позволяет воплощать инновационные игровые механики.

С роботом OLEM, чтобы поиграть, достаточно вытащить девайс размером с шайбу, который научит играть и поможет следить за процессом. Если взять ещё и дополнительное поле, то робот будет перемещаться по нему, добавляя новые игровые механики.

К сожалению этот робот все еще стартап, исходно обещали отгрузку первых роботов покупателям по предзаказу в конце 2022 года, по состоянию на февраль 2024 отгрузки все еще не было и последние обновления от команды авторов сообщают что отгрузка планируется на апрель 2024.

Другой интересный проект в области роботизации настольных игр, это Китайский интеллектуальный робот SenseRobot.

Рисунок 4. SenseRobot для игры в Го

Китайский интеллектуальный робот SenseRobot (SenseTime, 2023) представляет собой уникального партнера для игры в Го – древнюю стратегическую настольную игру. Робот использует искусственный интеллект и способен работать как с новичками, так и с опытными гроссмейстерами. В отличие от предыдущих компьютерных программ, SenseRobot Go не только предоставляет интеллектуальное противостояние, но и физически перемещает игровые камни на доске. Робот SenseRobot Go, хотя все еще находится в процессе разработки, уже достиг высокого уровня мастерства в игре. В 2023 году он выиграл соревнование SenseTime Go Challenge, обойдя несколько опытных игроков-людей. Вот его основные характеристики:

- Зрение 3D: Robot SenseRobot Go видит игровую доску в трех измерениях, что дает ему более полное понимание ситуации на доске.
- Мощный искусственный интеллект: робот использует мощный искусственный интеллект, который позволяет ему делать продуманные и стратегические шаги.
- Самообучение: робот может учиться и становиться лучше с течением времени.
- Человекоподобная игра: робот способен играть с людьми естественным и увлекательным способом.

SenseRobot Go – это новая и перспективная технология, которая может изменить то, как люди играют в го. Старт продаж данного робота был 05 января 2024 года.

Для детей младшего возраста есть интерактивный робот Mojobot (Mojobot) для обучения программированию. Mojobot - это робот, который обучает детей программированию в форме настольной игры. Он оснащен датчиками, микрофоном и динамиками, а также двумя глазами-светодиодами. Робот может поднимать и перемещать предметы, отображать разные выражения глаз и говорить.

Программирование Mojobot осуществляется с помощью специальных тегов кодирования. Их можно использовать для выполнения различных задач, таких как перемещение жетонов, отображение различных выражений глаз и реагирование на предметы и препятствия.

Рисунок 5 Робот Mojobot

Игровой набор включает в себя самого робота, консоль для управления и программирования, карту, жетоны и теги кодирования. Игра с Mojobot помогает детям развивать логическое мышление, изучать науку, технику и математику. Вместе с тем, игра становится увлекательным времяпрепровождением для детей и взрослых, позволяя им вместе изучать программирование.

1.2.3 Анализ собранной информации

Таким образом из вышесказанного очевидно, что тема настольных игр актуальна и полезна для детей и более соответствует положительному воздействию на их развитие нежели чем онлайн игры в гаджетах. Помимо этого, для исследовательской работы робота-манипулятора является обоснованным, так как именно такого типа роботы, только более сложной механики, наиболее востребованы в мире.

Из найденных прототипов только один находится в массовой продаже длительное время, один только ожидается, второй недавно поступил в продажу и имеет довольно высокую стоимость (350USD). С одной стороны малое количество найденных прототипов может говорить о том, что

данное направление не очень востребовано, с другой стороны – два из трех найденных роботов выходят на рынок в 2024 году, что может говорить о зарождении спроса на такого рода роботов именно сейчас и подтверждает актуальность задачи.

1.3 Разработка идеи и концепции робота. Формулировка технического задания

Общая идея проекта заключается в создании робота способного перемещать объекты в виде тонких картонных карт по трем осям в пространстве. Робот должен быть достаточно компактным, чтобы помешаться на стол. При создании робота необходимо выбрать как тип механики робота, так и аппаратную и программную части.

Креативность и новизна проекта подтверждается отсутствием аналогов имеющих все существенные признаки функциональности данного проекта. При этом самих робо-манипуляторов с управлением с Arduino, Rasperry или других платформ весьма много. Как правило они имеют механические захваты для габаритных изделий и почти на всех из них отсутствует штатная возможность подключения камеры. Также известны проекты по машинному обучения контроллеров Raspberry именно игре в доббль. Есть проекты роботов для настольных игр, показанные выше в обзоре прототипов. Однако мне не удалось найти ни одного решения комбинирующего все эти признаки. Таким образом идея имеет новизну! Готовое устройство может применяться как непосредственно по прямому назначению игры в настольную игру Доббль, так и более практическое применение в качестве наглядного учебного пособия по обучению старших школьников многим аспектам современной робототехники, а именно:

- механическая система с тремя степенями свободы. Решение прямой и обратной кинематических задач
- освоение библиотеки компьютерного зрения
- освоение различных видов нейронных сетей и понимание их специализации
-освоение современных библиотек синтеза и распознавания речи
- основы схемотехники, 3d моделирования и печати.

Техническое требования к заданию на разработку робота-манипулятора "Добблинатор" можно сформулировать следующим образом

Технические требования:

1. Визуальное распознавание:
- Разработка системы визуального распознавания карт настольной игры "Доббль".
- Проведение исследования различных алгоритмов компьютерного зрения для распознавания объектов на игровых карточках.

2. Робототехническая рука:
- Выбор наиболее подходящей и простой в реализации кинематической модели манипулятора

для перемещения игровых карт

- Интеграция выбранной модели робо-руки с системой визуального распознавания.

- Разработка программного обеспечения для управления робо-рукой в соответствии с распознанными объектами.

- Выбор аппаратной платформы. Изучение какие компоненты выбрать, как их соединить и подружить друг с другом.

3. Интерактивное взаимодействие:

- Создание механизма взаимодействия робота с детьми в процессе игры.

- Реализация возможности реагирования на действия детей и создание атмосферы соперничества.

5. Интеграция технологий:

- Выбор и интеграция подходящей нейросетевой архитектуры для визуального распознавания, без привязки к конкретной модели (например, использование Yolo v8, TensorFlow, Математическая статистика и кластерный анализ).

- Исследование и выбор оптимальных технологий для программного обеспечения.

Ожидаемые результаты:

- Робот-манипулятор "Добблинатор" с реализованными функциями визуального распознавания, управления роботической рукой и интерактивного взаимодействия с детьми.

Критерии приемки:

- Функциональность визуального распознавания карт и возможность перемещения робо-рукой игральных карт Доббль.

- Эффективное взаимодействие с детьми и создание игрового опыта.

1.4 Обоснование соответствия понятию «робот»

Согласно ГОСТ Р 60.0.0.4-2019/ИСО 8373:2012 (РТК, 2019) пункта 2:

робот (robot): *Исполнительный механизм, программируемый по двум или более степеням подвижности, обладающий определенной степенью автономности и способный перемещаться во внешней среде с целью выполнения задач по назначению.* (ИСО/ТК 299 "Робототехника" в 2018 году принял новое определение: робот (robot): Программируемый исполнительный механизм с определенным уровнем автономности для выполнения перемещения, манипулирования или позиционирования.)

Самый спорный пункт данного ГОСТ это «определенная степень автономности», так как какая она должна быть именно не сказано. На эти разночтения в частности указывается в статье (agradov, 2020). В целом под автономностью в данном ГОСТ понимается следующее:

автономность (autonomy): Способность выполнять задачи по назначению на основе текущего состояния и восприятия внешней среды без вмешательства человека.

Получается для автономности у устройства в любом случае должна быть сенсорная система, чтобы он мог воспринять внешнюю среду.

Как мы видим для отнесения устройства к роботу важно проанализировать три критерия:

- количество степеней подвижности
- степень автономности (работа без вмешательства человека)
- наличие сенсорной системы для восприятия внешней среды

-способность устройства перемещаться в пространстве или (согласно более нового термина) перемещать (позиционировать, манипулировать) другие предметы.

Обсуждаемый проект «Добблинатор» имеет три степени подвижности, может самостоятельно находить, распознавать и называть совпадающие объекты на двух карта и затем перемещать карту в пространстве. «Добблинатор» выполняет свою основную задачу полностью автономно (без вмешательства оператора после старта программы).

Текущее определение Международной федерации робототехники (International Federation of Robotics, далее — IFR): «Робот — это рабочий механизм, программируемый по нескольким осям с некоторой степенью автономности и способный передвигаться в пределах определённой среды, выполняя поставленные задачи».

Таким образом «Добблинатор» безусловно является «роботом» как согласно определений ГОСТ Р 60.0.0.4-2019 так и по международной классификации IFR.

Для справки далее приведено понятие более простых робототехнических устройств:

робототехническое устройство (robotic device): Исполнительный механизм, обладающий характеристиками промышленного робота (2.9) или сервисного робота (2.10), но не имеющий либо необходимого числа программируемых степеней подвижности (4.3), либо некоторой степени автономности.

Глава 2 Разработка технологического процесса

2. Описание процесса проектирования механики робота и его изготовления

2.1. Проектирования механики робота

В интернете есть множество моделей роботов-манипуляторов, например по поисковому запросу robot-arm либо robotic arm на сайтах с архивами 3d моделей https://www.thingiverse.com/ и https://thangs.com/ можно найти несколько десятков как готовых манипуляторов, так и запчастей к ним. На рисунках далее показаны некоторые наиболее характерные варианты:

Рисунок 6. BCN3D MOVEO - A fully OpenSource 3D printed Robot Arm

Рисунок 7. EEZYbotARM MK1 & MK2

Рисунок 8.DIY Arduino Robot Arm with Smartphone Control

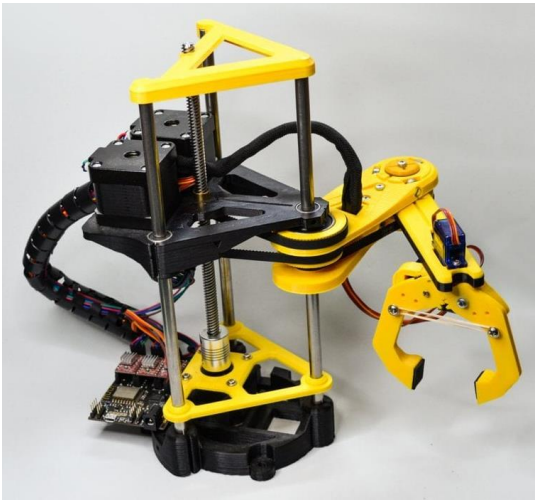


Рисунок 9. Arduino Robotic Arm (OPEN SOURCE) by JJrobots

Как мы видим представленные различные типы механики манипуляторов, от сложных 6-осных до простых трехосных типа scara и обычных. С учетом требований Технического задания проекта было решено остановится на трехосевой модели. Было изучено несколько прототипов на их основе в среде Solidworks были разработаны детали для манипулятора.

Fusion 360, SolidWorks и Компас 3D - это три различных программных продукта для компьютерного проектирования в 3D (CAD), предназначенные для создания 3D-моделей и поддержки инженерного проектирования. Давайте сравним их основные характеристики:

Fusion 360:

Лицензирование: Autodesk Fusion 360 часто предоставляется в виде облачного сервиса, что позволяет работать над проектами с разных устройств и совместно с командой.

Совместная работа: Имеет мощные функции для совместной работы, включая возможности облачного хранения и совместного доступа к проектам.

Целевая аудитория: Особенно популярен среди стартапов и небольших компаний, благодаря гибкому ценообразованию и доступу к облачным ресурсам.

Популярен у DIY-щиков и начинающих инженеров так как есть бесплатная лицензия сроком на 1 год.

SolidWorks:

Профессиональная стойкость: SolidWorks является одним из ведущих продуктов в области инженерного проектирования и используется в различных отраслях, включая крупные корпорации.

Обширные функции: Предоставляет широкий спектр инструментов для 3D-моделирования, анализа, симуляции и документации.

Сообщество и поддержка: Обширное сообщество пользователей и широкие ресурсы для обучения и поддержки. Много видео уроков на русском и английском языках.

Компас 3D:

Российский производитель: Компас 3D разрабатывается российской компанией ASCON и популярен среди пользователей в России и странах СНГ.

Поддержка стандартов: Хорошо подходит для работы с отечественными стандартами и требованиями.

Доступная цена: имеет более доступную цену по сравнению с зарубежными аналогами. Есть студенческая дешевая лицензия со стоимостью около 1000 рублей. Есть бесплатные лицензии для школ.

Общие черты, такие как возможности 3D-моделирования, анализа, симуляции и создания документации, присутствуют во всех трех продуктах. Выбор между ними зависит от конкретных потребностей пользователя, стоимости лицензии, а также предпочтений в использовании облачных или локальных решений.

В целом лучше освоить проектирование в отечественном ПО Компас 3D, однако у меня были в быстром доступе только видео уроки по SolidWorks, а также часть робота-манипулятора была найдена уже в этом формате, поэтому было решено текущий проект делать целиком в SolidWorks.

Часть модели я взял из готового проекта (сам манипулятор), самостоятельно доделывал крышки моторов, основу робота, системы вращения в плоскости x-y.

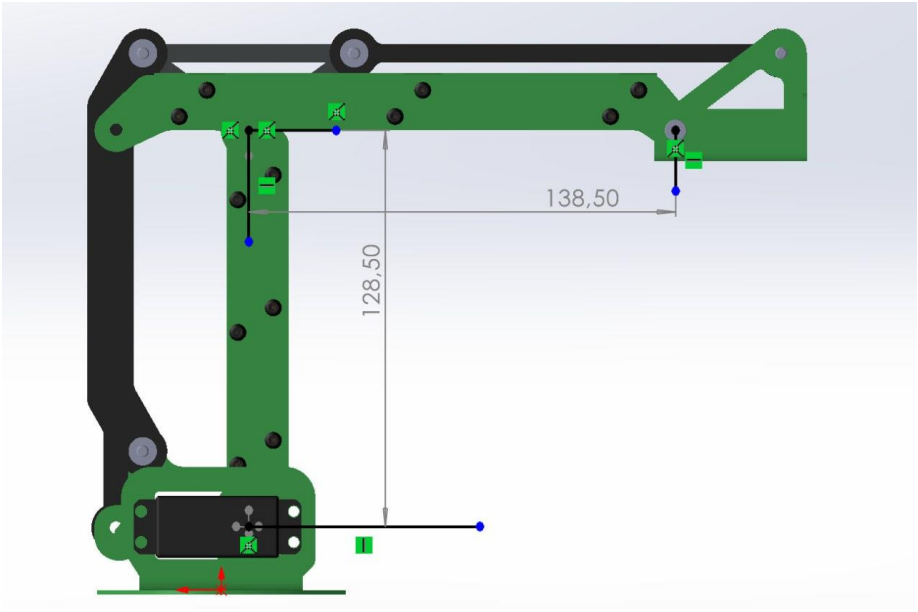


Рисунок 10. Заготовки модели роборуки., взятая за старт проекта

Важным аспектом при выборе модели и типа манипулятора было наличие площадки для крепления камеры, причем эта площадка должна всегда оставаться горизонтальной, чтобы смотреть на игровое поле четко сверху вниз. Другим аспектом было отсутствие моторов сверху руки, тем самым центр тяжести манипулятора остается внизу и он будет крепко стоять на основании.

При проектировании основания необходимо было учесть необходимость вращения на 180 градусов, разместить в основании сервопривод для этого вращения, а также разместить все остальные необходимые компоненты, количество и вид которых менялся по ходу проекта, далее представлен финальный вариант для размещения итогового набора компонент: jetson nano, raspberry pi pico, мосфеты, клапан, насос, 3 шт dc-dc, кнопка для пользователя, проводка для соединения всех этих компонент. Также планируется установить локальный экран, поэтому под него сделан вырез, на одном из элементов, но пока это еще не реализовано. Также кнопка включения будет рядом с экраном, когда появится этот экран. На следующих рисунках 11 и 12 показаны итоговые результаты

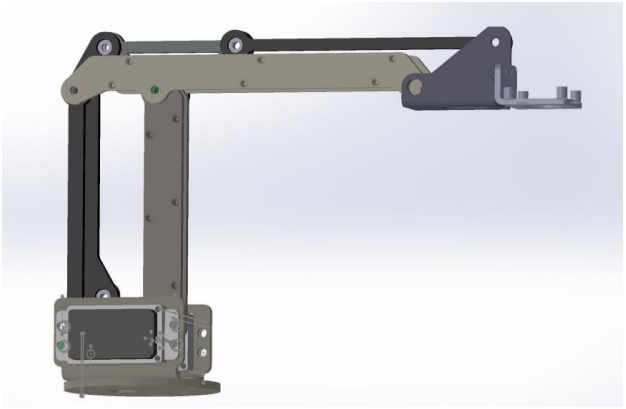


Рисунок 11. основной манипулятор с креплением камеры

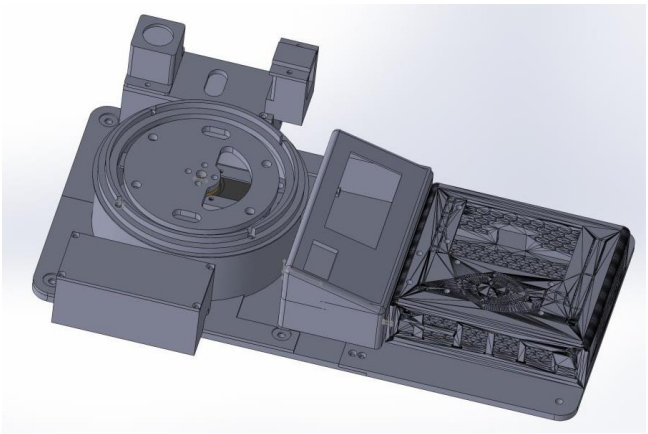


Рисунок 12. основание робота манипулятора с поворотной базой

2.2. Описание процесса изготовления механики робота

Относительное простые и доступные виду изготовления школьных проектов это 3D печать и лазерная резка фанеры/оргстекла. Рассматривать другие более промышленные способы изготовления такие как : точение, горячая штамповка, отливка в прессформы не имеет смысла в виду их недоступности в рамках школы. На мой взгляд безусловное преимущество для объектов размеро в пределах 200*200 мм имеет 3D печать, так как можно создавать не плоские объекты, а объемные. В случае лазерной резки для этого приходится проектировать много пазов, потом склеивать эти пазы. В связи с этим в качестве технологии однозначно была выбрана 3D печать. Следующим шагом стал выбор материала из которого печатать модель.

Существует несколько типов пластиков, широко используемых в технологии 3D-печати, таких как ABS (акрилонитрил-бутадиен-стирол), PLA (полимолочная кислота) и PETG (полиэтилентерефталат с модифицированным гликолем). Давайте рассмотрим их основные характеристики и обсудим, какой материал является предпочтительным для изготовления деталей манипулятора:

1. ABS (акрилонитрил-бутадиен-стирол):

- Прочность: ABS обладает высокой прочностью, что делает его подходящим для создания деталей, которые могут подвергаться механическим воздействиям или нагрузкам.

- Термостойкость: Этот материал обладает хорошей термостойкостью, что важно при работе с манипуляторами, которые могут подвергаться воздействию солнца. За счет высокой термостойкости можно использовать вплавляемые гайки.

- Разнообразие цветов: ABS доступен в широком спектре цветов, что дает больше возможностей для дизайна.

- Хорошо шлифуется и можно окрашивать

2. PLA (полимолочная кислота):

- Экологичность: PLA производится из натуральных источников и биоразлагаем, что делает его экологически более дружественным.

- Простота печати: PLA легко печатается и не требует специальных условий для 3D-печати.

- Низкая токсичность: PLA считается менее токсичным по сравнению с ABS при нагреве.

3. PETG (полиэтилентерефталат с модифицированным гликолем):

- Прочность и устойчивость к ударам PETG обладает высокой прочностью и устойчивостью к ударам, что делает его подходящим для создания прочных и надежных деталей.

- Термостойкость: также обладает хорошей термостойкостью, что важно для приложений, где может возникнуть нагрев солнечными лучами.

В итоге я выбрал ABS как предпочтительный материал для манипулятора, по следующим причинам:

- Механическая прочность: Манипуляторы могут испытывать значительные механические нагрузки, и ABS обеспечивает высокую прочность, что важно для изготовления прочных и долговечных деталей.

- Термостойкость: ABS выдерживает более высокие температуры по сравнению с PLA, что важно для приложений, где возможно воздействие солнечного нагрева. Модели напечатанные из PLA «плывут» на солнце и теряют свою форму (проверено лично на даче у родителей).

- ABS можно шлифовать, по плану в следующий ревизии руки красиво её покрасить, тем самым приблизив внешний вид к заводскому, PETG хуже шлифуется поэтому от него отказался в этом проекте, хотя по остальным характеристикам он подходит и печатать им легче.

На рисунках 13÷16 показаны этапы сборки манипулятора.

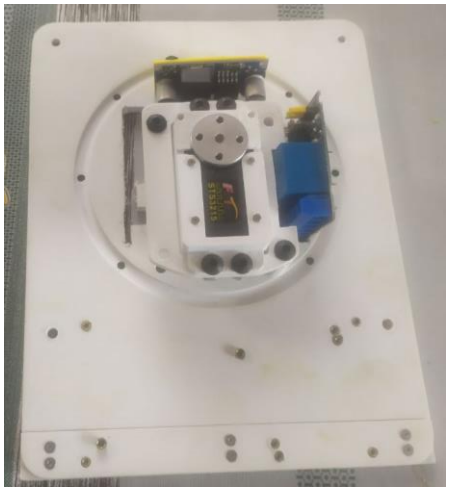


Рисунок 13. База робота с вращением

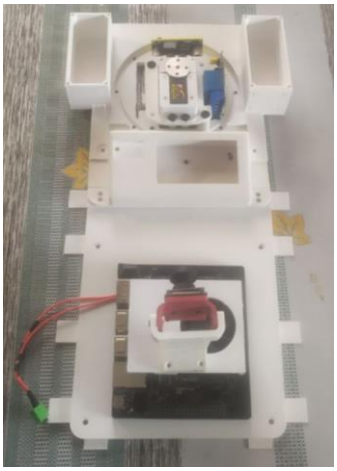


Рисунок 14. База до установки функциональных элементов

Рисунок 15. база робота со всеми корпусами

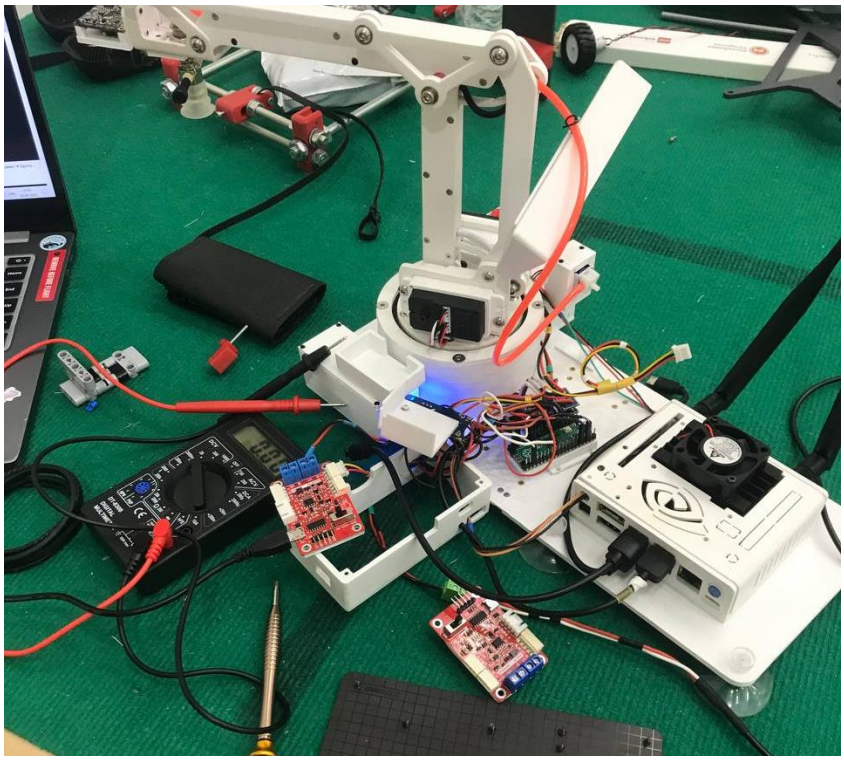


Рисунок 16. Сборка и пайка компонент манипулятора

2.3. Описание процесса проектирования электроники робота

Исходно планировалось сделать только робо-манипулятор и управлять им с ноутбука. Из программы на Python либо C++. В связи с этим использование ардуино, как в большинстве проектов не рационально, так как затруднительно будет реализовать передачу данных онлайн по usb порту с ноутбука на Arduino и далее на исполнение сервоприводам. В связи с этим был произведен поиск решений и сервоприводов которые бы позволили легко реализовать такую возможность. В результате были найдены приводы Feetech STS3215 имеющие цифровое управление по шине TTL Serial Bus Servo. Данные сервоприводы с высоким крутящим моментом и возможностью программирования по последовательной шине, оснащены 360° высокоточным магнитным энкодером, который обеспечивает управление абсолютным углом в пределах 360 градусов. С помощью специального софта с ПК можно устанавливать любые углы в качестве стартового положения, а также переключаться в режим непрерывного вращения или шагового двигателя. Встроенная функция плавного запуска и остановки делает движение более плавным. На каждом сервоприводе есть два интерфейса, которые могут использоваться последовательно. Теоретически

можно управлять одновременно 253 сервоприводами по шине, и каждый сервопривод может получать информацию о своем текущем угле, нагрузке, напряжении, режиме и так далее.

Подключение к ПК либо к контроллеру можно осуществить с помощью плату управления FE-URT1 который преобразует сигнал от serial порта USB в протокол Serial Bus Servo уровня TTL. Принципиальная схема показана на рисунке 17.

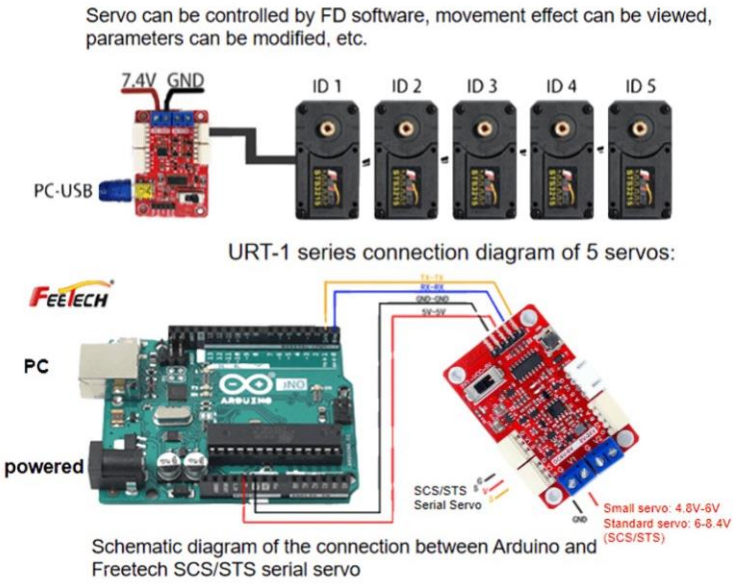


Рисунок 17. Принципиальная схема соединения цифровых сервоприводов

Аналогичное по функциональности решение, но для сервоприводов с классическим PWM/ШИМ управление было найдено у отечественного производителя в одной из статей (RyabovA) на тему сервоприводов. Однако стоимость такого контроллера в три раза выше чем (1590 против 530 рублей на начало февраля 2024) у конвертера FE1. К тому же для PWM сервоприводов придется прокладывать больше проводов (от каждой сервы к плате), а для цифровых сервоприводов все сервы соединяются последовательно на общую шину. Крому того сама плата FE-URT1 несколько меньше по своим габаритам что также облегчает дальнейший монтаж.

Таким образом сервоприводы и их управляющая плата были выбраны. Тестовые запуски и управление были написаны на ноутбуке и удалось успешно управлять манипулятором. Следующим вопросом стал захват и перемещение карт. Беглый анализ привел к тому, что механический захват никак применить не получится, так как надо брать одну верхнюю карту со стопки. С такой задачей хорошо справляются присоски с насосом. На АлиЭкспресс были найдены насос, присоска и клапан для перекрытия линии после захвата предмета и открытия при опускании.

Следующим вопросом встало как управлять этими устройствами? Выбор был между: Arduino, ESP32 и Raspberry Pi Pico

Приведем краткий анализ этих платформ:

Таблица 1. Сравнение микроконтроллеров

	Arduino	ESP32	Raspberry Pi Pico
Назначение	открытая платформа для создания прототипов электронных устройств. Основана на микроконтроллерах AVR или ARM	микроконтроллер с поддержкой беспроводных технологий, таких как Wi-Fi и Bluetooth.	микроконтроллерная плата, созданная Raspberry Pi Foundation, собранная вокруг микроконтроллера RP2040
Программирование	Используется язык программирования C/C++.	Поддерживает языки C/C++ и MicroPython. Имеется хорошая интеграция с языком Python, что обеспечивает удобство в программировании.	: Поддерживает MicroPython и C/C++. Python - один из основных языков для программирования на Pico.
Возможности	Подходит для небольших проектов, где требуется контроль над аппаратными ресурсами.	Используется для проектов с беспроводными функциями, такими как интернет вещей (IoT) и датчики	Обеспечивает низкопрофильную платформу для проектов, где не требуется полноценный мини-компьютер, но нужна мощность микроконтроллера

Выбор в зависимости от задач: Arduino хорош для простых электронных проектов. ESP32 подходит для проектов с беспроводными возможностями. Raspberry Pi Pico - для низкоуровневых микроконтроллерных задач.

Язык программирования: ESP32 и Raspberry Pi Pico лучше подходят для программирования на Python из-за поддержки MicroPython.

Поскольку базовый код для самого манипулятора уже был нписан на Python было решено весь проект реализовывать на этом языке программирования. Таким образом был выбран Raspberry Pi Pico для управления захватом-присоской, заодно к этой же плате было решено подключить адресную RGB светодиодную ленту, для цветовой подсветки манипулятора. Управление Pi Pico можно осуществлять по serial USB порты что обеспечивает универсально подключения как к ноутбук так и к мини-ПК (Raspberry, Jetson Nano и другие).

После сборки и тестирования этих компонент возникло понимание что на рабочем проекте использовать ноутбук как главный управляющий ПК не эффективно. В связи этим решено было использовать мини-ПК. Выбор стоял между Raspberry Pi 4 и Jetson Nano. В сравнение не учитывался отечественный Repka Pi так как помимо просто работы с устройствами (сервоприводы, управление помопой через pi pico) важной частью проекта является компьютерное зрения и машинное обучение, для которых важны параметры процессора и желательно наличие графического процессора помимо основного.

При выборе платформы было изучен ряд статей, в частности (DmitrySpb79, 2019), (3DGram), (Loghin, 2020). Скорости процессоров этих платформ практически одинаковые по данным тестов, что продемонстрировано на рисунке 18.

Основное отличе и преимущество Jetson Nano это наличие GPU Nvidia Maxwell с 128 ядрами CUDA. Данный видеопроцессор с поддержкой CUDA и Tensor Cores для ускорения вычислений машинного обучения дает существенные преимущества при машинном обучении.

Вывод: Если вам нужна более доступная опция для проектов машинного зрения и у вас нет высоких требований к производительности глубокого обучения, Raspberry Pi 4 - хороший выбор.

Если вам нужна максимальная производительность для машинного обучения, и вы готовы заплатить немного больше, Nvidia Jetson Nano может быть более подходящим вариантом.

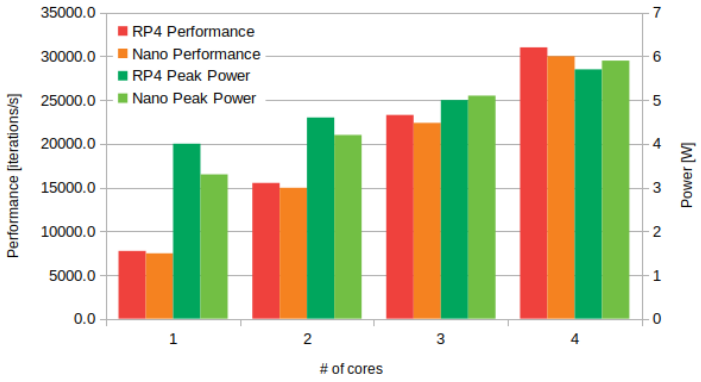


Рисунок 18. сравнение производительности процессоров Raspberry Pi 4 и Jetson Nano

Для составления структурной и принципиальной схемы необходимо было выбрать ПО причем такое, чтобы в нем были все используемые компоненты в библиотеке. Среди бесплатных программ рассматривались EasyED, Fritzing и KiCad. Fritzing отпало так как оно больше предназначено для проектов на базе Arduino и в нем не удалось найти штатно используемые контроллеры. Сравнение двух других ПО для проектирования электроники робота приведено в таблице 2.

Таблица 2. Сравнения среды проектирования электроники робота

	KiCad:	EasyEDA
Тип	Бесплатное и открытое программное обеспечение (FOSS).	Облачная платформа с возможностью использования в веб-браузере и тонком клиенте
Интерфейс	Интерфейс KiCad может быть немного более сложным для новичков из-за большего количества функций и опций.	Имеет более простой интерфейс, что делает его более привлекательным для новичков. Многие функции доступны через веб-интерфейс
Особенности	Обширные возможности для проектирования схем и печатных плат, трехмерное моделирование, поддержка множества форматов файлов	Прост в использовании, поддерживает создание схем и печатных плат, имеет встроенные библиотеки компонентов.
Сообщество	Обширное сообщество пользователей и активная поддержка.	Хотя сообщество не так обширно, как у KiCad, EasyEDA предоставляет поддержку и форум для пользователей

EasyEDA, вероятно, более подходит для начинающих благодаря своему простому интерфейсу и возможности использования в веб-браузере. KiCad предоставляет более широкий набор функций и возможностей для более продвинутых проектов.

Вывод: С учетом что я новичок в проектировании и был необходим простой способ создать принципиальные схемы для проектов, EasyEDA был признан более удобным выбором.

В результате были составлены принципиальная схема робота «добблинатор» Э1 по ГОСТ 2.701 (2.701-2008) представленная на рисунке 19 и структурная схема устройства Э3 представленная на рисунке 20. Отличия и особенности разных типов схем хорошо описаны с статье (augorelov, 2019).

Полная спецификация электронных компонент и их справочной стоимости по состоянию на 10.02.2024 (по данным AliExpress) представлена в таблице 3.

Таблица 3. Расчёт себестоимости проекта

Наименование	**Кол-во**	**Цена за 1 шт**	**Итого**
Сервопривод Feetech STS3215	3	1546	4638
Многофункциональный преобразователь сигнала Feetech URT-1	1	536	536
Микронасосы ZQ 370-02	1	156	156
Миниатюрный электромагнитный клапан ZQ	1	233	233
Вакуумная присоска L45-11, M10 с боковым подключением	1	71	71
Мосфет	2	70	140
Raspberry Pi pico	1	442	442
Jetson Nano B01	1	17 000	17 00
Катушка ABS пластика	1	2200	2200
Итого			26 294

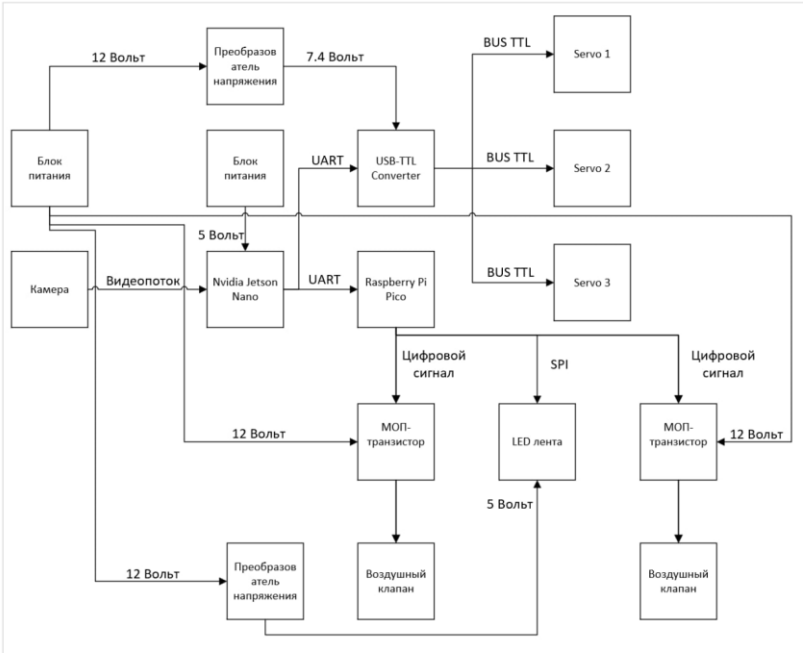


Рисунок 19. Принципиальная схема робота "Добблинатор"

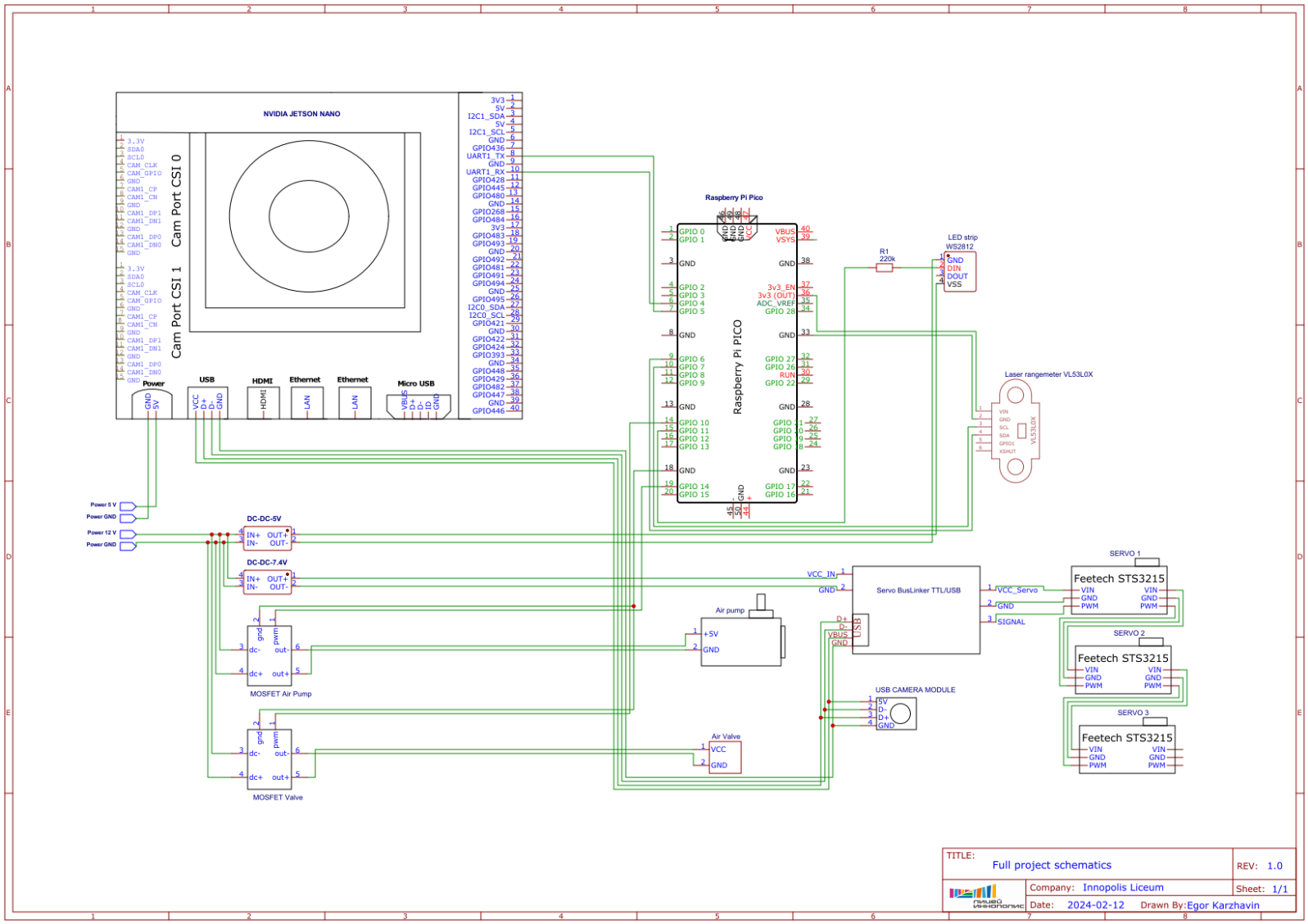


Рисунок 20. структурная схема ЭЗ робота "Добблинатор"

2.4 Описание процесса программирования, отладки и модификации робота.

В контексте описанных задач и целей проекта, использование Python представляется более удобным. Вот несколько причин:

Простота и Читаемость Кода:

Python предлагает простый и лаконичный синтаксис, что делает код более читаемым и легким для понимания. Это особенно важно для проектов, где важно быстро разрабатывать и внедрять новые функции.

Большое количество библиотек:

Python обладает обширной экосистемой библиотек и фреймворков, что может существенно ускорить разработку. Например, для компьютерного зрения (визуального распознавания) в проекте, библиотеки OpenCV и TensorFlow в Python предоставляют мощные инструменты.

Машинное Обучение и Искусственный Интеллект:

Если вам потребуется интегрировать машинное обучение или искусственный интеллект в проект, Python является одним из лидеров в этой области с богатым выбором библиотек, таких как TensorFlow, PyTorch и Scikit-learn и самое главное, легкий в использовании Yolo.

Сообщество и Поддержка:

Python имеет обширное и активное сообщество разработчиков. Это означает, что вы сможете легко находить решения для своих вопросов, а также использовать готовые решения и код из открытых источников.

Хотя C++ также может быть использован для реализации подобных проектов, Python, вероятно, предоставит более гибкое и удобное окружение для достижения поставленных целей в данном контексте. По всем этим причинам был выбран Python, хотя и пришлось его учить почти с 0, при том что уровень владения C++ на момент начала проекта был выше чем Python.

Несмотря на то, что PyCharm создан специально для Python, что означает лучшую поддержку и интеграцию с языком, в качестве среды разработки был выбран VS Code, так как VS Code более легковесен и обладает более быстрым запуском по сравнению с PyCharm, а также VS Code используется в работе с различными языками программирования, а не только в Python. Что удобно при необходимости работать с несколькими языками.

2.4.1. Программирование устройств (сервоприводы, помпа, клапан, RGB лента)

Как уже было сказано ранее, проект начался с разработки непосредственно робо-манипулятора. Его программирование было выполнено на Python довольно быстро, и была создана библиотека arm_control.py. Данная простая библиотека основана на sdk производителя сервоприводов Feetech-servo-sdk (Feetech) и работает с её использованием. Самое главное в настройке программы в этой части это ограничить возможность перехода серв вне диапазона, в

котором манипулятор может работать механически без повреждений. Такие ограничение введены на двух уровнях: на уровне программатора самих сервоприводов, таким образом даже если из управляющей программы придет команда на переход в запрещенный угол, сервопривод откажется выполнять командую и на уровне основной управляющей программы:

```
# Default setting
SERVO_1_POS_LIMIT = [970, 3020]         # вращение
SERVO_2_POS_LIMIT = [1000, 1500]         # вперед - назад
SERVO_3_POS_LIMIT = [1500, 2073]         # вниз - вверх
```

Следующим этапом работы стало написание базовой программы управления для pi pico. Идея заключалась в том, что не pico будет постоянно исполнятся программа, которая получает команды в заранее известно формате и исполняет их. Прием команд реализован по последовательному порту (USB serial). Команда передается в виде текстовой строки. По окончанию приема строки (символ конца строки). Pico разбивает принятые данные на части (используя символы-разделители команд внутри строки “,”. Всего передается три параметра означающие: что делать с насосом (включить/выключить == 1/0), что делать с клапаном (закрыть/открыть == 1/0) и какой цвет подсветки включить в виде слова на английском. Pico уже сама переводит кодовое слова в RGB коды. Основной цикл работы программы на Pico выглядит следующим образом:

```
while True:
    try:
        if uart.any():
            msg=receiveData()
            if(msg=="quit"):
                break
            elif(msg[:2]=="dp"):
                    p,c,color=map(str,msg[2:].split(","))
                    pomp.value(int(p))
                    clap.value(int(c))
                    if str(color)=="BLACK":
                        pixels_fill((0, 0, 0))
                    elif str(color)=="RED":
                        pixels_fill((255, 0, 0))
                    elif str(color)=="YELLOW":
                        pixels_fill((255, 150, 0))
                    elif str(color)=="GREEN":
                        pixels_fill((0, 255, 0))
                    elif str(color)=="CYAN":
                        pixels_fill((0, 255, 255))
                    elif str(color)=="BLUE":
                        pixels_fill((0, 0, 255))
                    elif str(color)=="PURPLE":
                        pixels_fill((180, 0, 255))
                    elif str(color)=="WHITE":
                        pixels_fill((255, 255, 255))
                    pixels_show()
```

```
#                 data=["done"]
#                 dataOut=json.dumps(data)
#                 sendData(dataOut)
    except:
        pass
```

На начально этапе команды отправлялись с ноутбука записью в последовательной порт USB.

2.4.2. Выбор подхода и инструментов для компьютерного зрения и машинного обучения

Последним и самым сложным этапом создания робота стала интеграция всех компонент в единое устройство на базе Jetson Nano и написание основного блока программы отвечающего за машинное обучение и компьютерное зрение.

Для начала надо подготовить Jetson Nano к работе, установить на него VS Code , Python с библиотекой OpenCV. Последний актульаный релиз от Nvidia для Jetson Nano B01 основан на Ubuntu 18.04 и в нем установлен Python 3.6.9 с Open CV 4.5. именно на такой платформе была начата работа по анализу видео потока с камер, нахождения игровой карты на поле и её анализа.

Предварительно были изучены две статьи с аналогичными задачи. В первой (Dinxor, 2021) автор предлагал использовать для классификации объектов соотношение длин прямоугольника и средние значения по каждому из цветовых каналов RGB. Этих данных оказалось мало для обучения простой нейросети - многослойного перцептрон (MLP). Автор добавил еще один параметр - отношение площади символа к площади описывающего его прямоугольника (автор назвал его "плотность") и сего помощью ему удалось достигнуть на ПК хороших результатов - полный проход по колоде из 55 карточек (1485 комбинаций) занял 170 секунд, ошибок 0. Недостатком данного подхода сам автор называет то, что подход работал на подготовленных для печати карточках, на которых отсутствуют шумы, блики, проблемы с реальным фоно на котором лежит карточка. Сам автор пишет : «Была идея сделать распознавание карточек по изображению с веб-камеры, но оказалось, что они сильно бликуют. Кроме того, у сети выявился недостаток - она сильно чувствительна к понижению разрешения картинки: мелкие детали (например, лучи снежинки) сливаются и это приводит к изменению параметров.» Исходя их этого был сделан вывод что такой подход не применим на практике для робота работающего с настоящими карточкам в реальном мире.

Затем была изучена статья (Harder, 2020) в переводе на русский (MaxRokatansky, 2020). В статье автор работает уже не с идеальными файлами для печати карт, а с фотографиями карт. Правда фотографии эти все были сделаны на хорошую камеру телефона, с одного расстояния и без бликов освещения. Камера в режиме «онлайн» чтения видеопотока не использовалась. Автор создала и обучить сверточную нейронную сеть с помощью библиотеки Keras, Результаты получены весьма хорошие и распознавание происходило с точностью близкой к 100%.

На основе изученных данных было решено что все получится. Копировать решение из статьи (MaxRokatansky, 2020) я решил не правильным как по причине того что сети CNN пока сложны для меня в понимании, так и из желания получить свой опыт. Тем не менее ряд идей по подготовке

изображения я позаимствовал из этих статей, хоть из и пришлось адаптировать по работу с камерой среднего качества и видео потоком с тенями и бликами, вместо «идеальных» картинок.

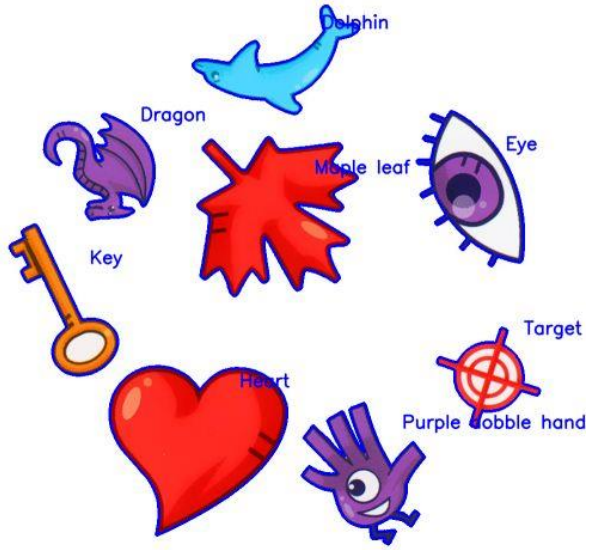


Рисунок 21. Идеальная карточка из статьи автора Dinxor

Нейронные сети Convolutional Neural Network (CNN) и YOLO (You Only Look Once) - это два разных подхода к задачам компьютерного зрения, особенно в области обнаружения объектов. Давайте рассмотрим их основные отличия:

Таблица 4 сравнение Yolo и CNN

	CNN	Yolo
Архитектура	CNN - это тип нейронных сетей, предназначенных для обработки и анализа визуальных данных. Она включает в себя слои свертки для извлечения признаков из изображений, а затем плотные слои для классификации или регрессии.	YOLO - это архитектура, предназначенная специально для обнаружения объектов в реальном времени. Она строится на сверточных слоях, но также включает слои, предсказывающие координаты и классы объектов.
Цель	Основная задача CNN - это классификация изображений. Она определяет, к какому	Основная задача YOLO - это обнаружение объектов. Она предсказывает

	классу принадлежит входное изображение.	ограничивающие рамки (bounding boxes) и классы всех объектов на изображении.
Обнаружение объектов	В случае CNN для обнаружения объектов требуется использовать дополнительные методы, такие как слои Region Proposal Network (RPN) или использование методов прогнозирования, чтобы определить, где могут находиться объекты.	YOLO решает проблему обнаружения объектов, предсказывая bounding boxes и классы одновременно. Это делает его более эффективным для реального времени, поскольку не требуется дополнительных этапов обработки.
Скорость	CNN может быть более ресурсоемким, поскольку требует дополнительных шагов для обнаружения объектов.	YOLO оптимизирована для высокой скорости обнаружения объектов в реальном времени, что делает её эффективной в приложениях, где важна скорость.

В итоге, хотя CNN и YOLO используют сверточные слои для извлечения признаков, их цели и подходы существенно различаются, с YOLO, который оптимизирован для быстрого и эффективного обнаружения объектов в реальном времени.

Данные о скорости работы, например приведены в статьях (Sojasingarayar, 2022) и (Тимошкин М.С.) в обеих из них делает вывод о большей скорости Yolo v5 по сравнение с CNN сетями, быстрее чем Yolo, однако, может работать. SSD

На момент старта проекта уже почти год была доступна версия YOLOv8 - это новейшее семейство моделей обнаружения объектов на базе YOLO от Ultralytics, обеспечивающих самые современные характеристики. По сравнению с предыдущими версиями YOLO, модель YOLOv8 работает быстрее и точнее, обеспечивая при этом единую структуру для обучения моделей для выполнения. К моменту окончания проекта уже вышла версия 9 сети Yolo, но я не успел изучить возможность её применения.

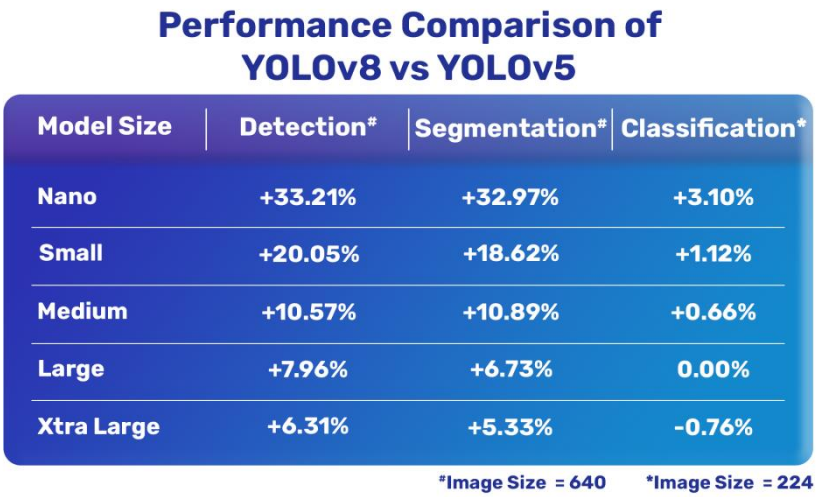


Рисунок 22. Сравнение скорости работы Yolo v8 и v5

2.4.3. Первичная обработка картинки и разметка данных

Обучение сети Yolov8 для работы с карточками было решено сделать по инструкции из следующего видео урока (engineer, 2023). Так как тут требовалось только разложить подготовленные картинки по папкам, аналогично тому, как делали в статье (Dinxor, 2021). В этих подходах нет необходимости вручную обводить каждый предмет на каждой фотографии карты (55 карт минимум по 4 фото -220 фото. На каждой карте 8 объектов, итого поставить 1760 рамок и каждую из них правильно классифицировать – нееет, очень много ручного труда!)

Порядок необходимых операций над изображениями с камеры представлен на следующей схеме:

Рисунок 23. схема подготовки изображений

Рисунок 24. исходное изображение с камеры при обучении сети

На рисунке 24 представлен исходный кадр с камеры с обведенными контурами белых цветом и степенью их «круглости». Как мы видим блики нам тоже мешают, приходилось их отгораживать стенками. Дальнейшие результаты обработки изображения представлены на рисунках 25 и 26. На рисунке 26 мы видим 8 отдельных изображений, которые затем сохранялись каждое как отдельный файл.

Рисунок 25. Игровая карта Доббль после обработки изображения

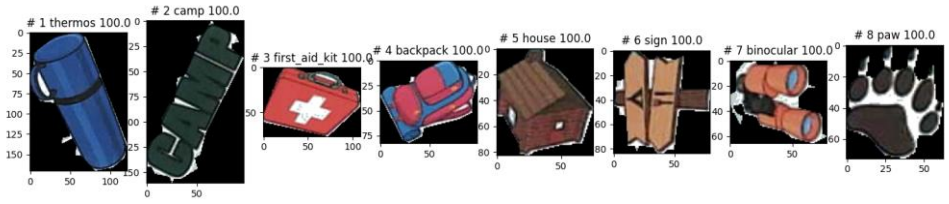


Рисунок 26. Восемь отдельных картинок после сегментации

Алгоритмы кластеризации для сохранения картинок под разными именами и/или в разные папки не применялись, хотя конечно надо было их взять из статьи для ускорения работы. С другой стороны, за 3 часа я разложил все картинки по папкам и подготовил тем самым данные для обучения сети типа «сегментация» (segmentation). Сначала картинки складывались в общую папку pictures, откуда я их разложил по подпапкам классов в dobble_dataset. Структура папок датасета для обучения показана на рисунках 27 и 28.

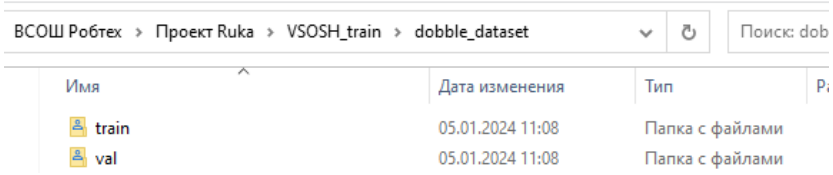


Рисунок 27. Стуруктура dataset

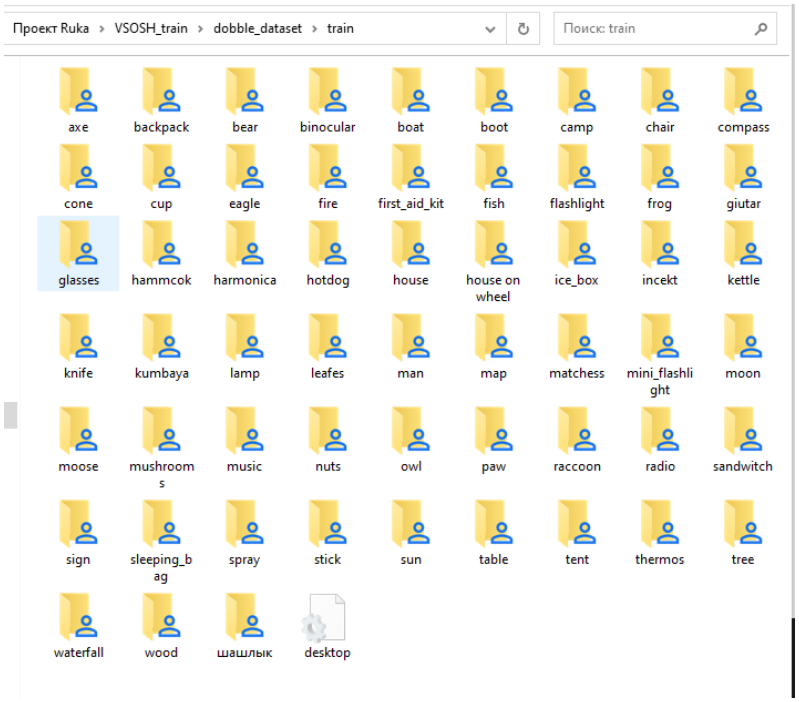


Рисунок 28. Организация картинок по классам

В папке для обучения сети 1376 файлов, в папке валидации (val) 343 файла. Рекомендации по соотношению данных 80% обучение и 20% валидация выполнены в четком соответствии

За обучение сети отвечает программа trainYolo.py представленная ниже

```
from ultralytics import YOLO
import torch

# Check for CUDA device and set it
device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f'Using device: {device}')
model=YOLO('yolov8n-cls.pt').to(device) #classify full image

model.train(data='/home/eg/VSOSH24/dobble_dataset/',
            epochs=40,
            imgsz=448,
            batch=6)
```

Здесь представлен её вариант последней версии который использует cuda ядра графического процессора Jetson GPU. Исходно я не знал как это сделать и обучение происходило на основном процессоре более 7 часов. Качество обучения сети показано на рисунке 29 и оно весьма хорошее:

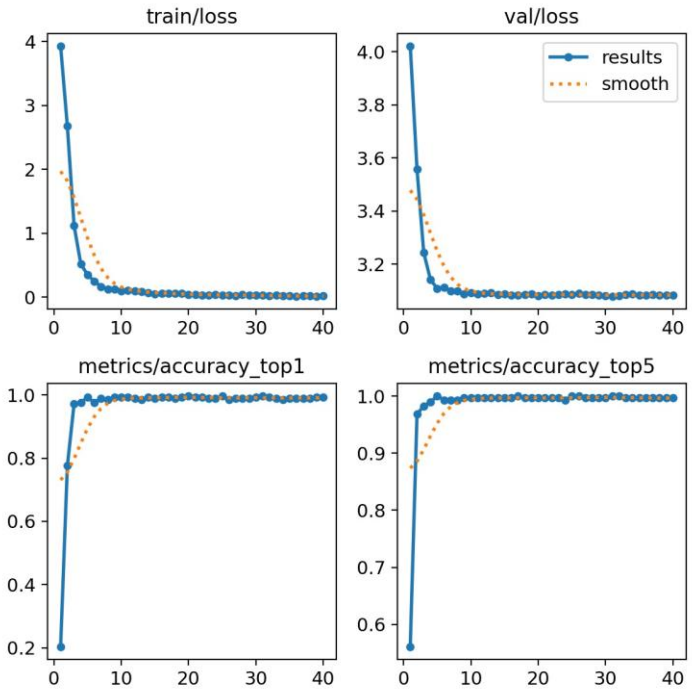


Рисунок 29. Метрики обучения сети классификации Yolo v8

Следующим этапом стала проверка того, как сеть распознает объекты. Здесь важно обратить внимание что все нейронный сети могут работать в нескольких режимах (рисунок 30) и статья (Buhl, 2023):

- Классификация (classification)
- Обнаружение объектов (object detection)
- Сегментация (segmentation)
- Определение положения тела (Pose Estimation)
- Отслеживание объекта (Object tracking)

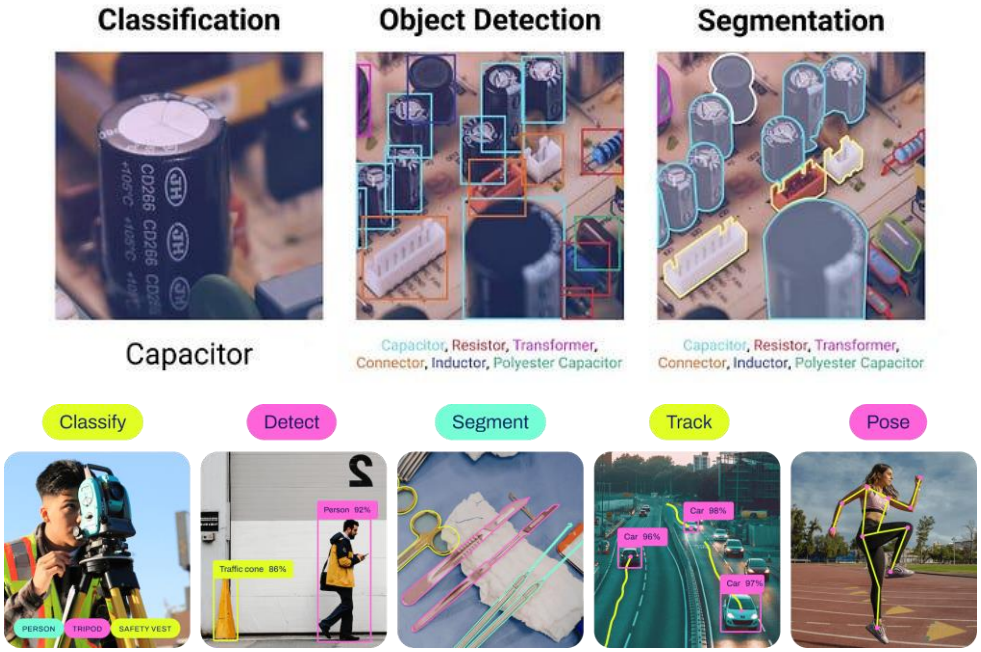


Рисунок 30. Различные виды нейронных сетей

Поскольку исходно обучение происходит не с 0, а опираясь на заранее обученную сеть, предоставленную авторами Yolo то и эту модель и способ обучения необходимо выбирать именно под тот режим работы, который будет использоваться. Нельзя обучить сеть на классификацию изображений, а потом использовать в режиме обнаружения.

Поскольку я обучил первую версию сети в режиме классификации, то для ее работы необходимо каждый раз разбирать кадр камеры на 8 отдельных картинок, в каждой из которых только 1 предмет и это предмет классифицировать сетью. Оказалось, что на такую операцию тратится на CPU Jetson Nano от 400 до 900 мс, итого на полную карту в среднем 5 секунд, что конечно же очень долго. В связи с этим пришлось сделать две вещи:

1. Переобучить сеть с режима классификации на режим обнаружения
2. Разобраться как заставить Yolo работать на cuda на Jetson

Для первой задачи была написана отдельная программа, которая сначала берет кадр, находит карточку, разбивает ее на 8 объектов каждую из них классифицирует и потом показывает на общем изображении карточки к какому классу она отнесла каждый объект. На этом этапе, чтобы не бороться с бликами освещения и не искать «черную кошку в темной комнате» (белую карту на белом столе) было решено располагать карту на цветном ярком прямоугольнике, после чего с помощью аффинного преобразования вращать это прямоугольник в двух плоскостях до правильной фигуры, что убирает эффект «рыбьего глаза камеры) и потом уже на цветном фоне выделять белый круг игровой карты. Пример детекции и аннотации карты представлен на рисунке 31.

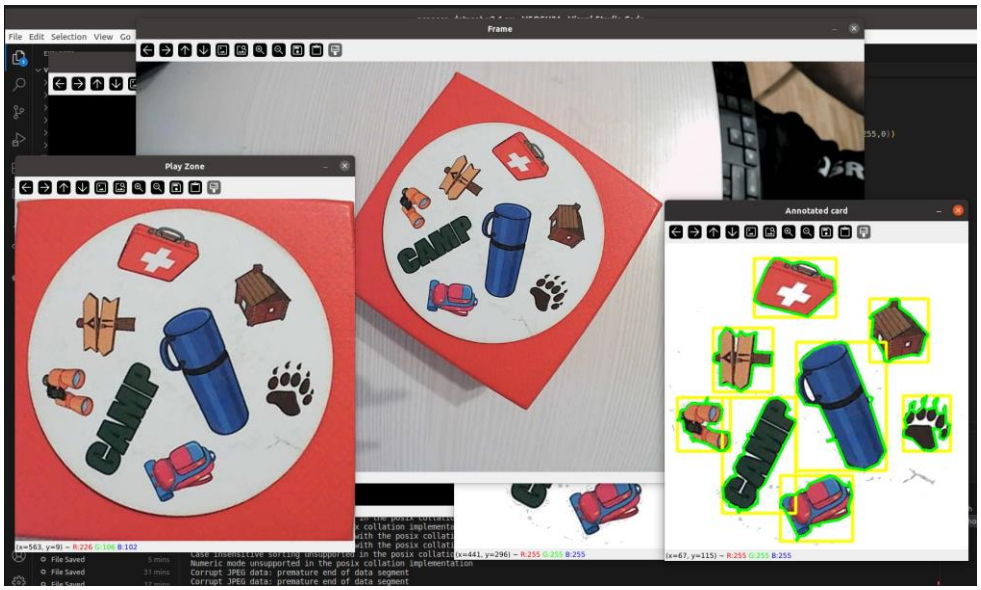


Рисунок 31. Работа по повторному обучению сети

По материалам статьи (dimanosov007, 2022) (перевод статьи (Kukil, 2022)) данные такой картинки и обнаруженных и классифицированных изображений были сохранены в формате необходимом для разметки данных Yolo. После чего уже было обучена сеть обнаружения объектов, дополнительно была изучена статья (Germanov, 2023).

Таким образом было подготовлено 283 картинки по 8 объектов на каждой. Данные разбиты в пропорции 80/20 между тренировкой и валидацией. Результаты обучения сети обнаружения объектов представлены на рисунках 32 и 33.

В контексте YOLO (You Only Look Once), Confusion Matrix (Матрица ошибок) используется для оценки производительности модели обнаружения объектов, такой как YOLO, на основе сравнения её прогнозов с истинными значениями в тестовом наборе данных. Confusion Matrix предоставляет подробную информацию о том, как модель классифицирует объекты.

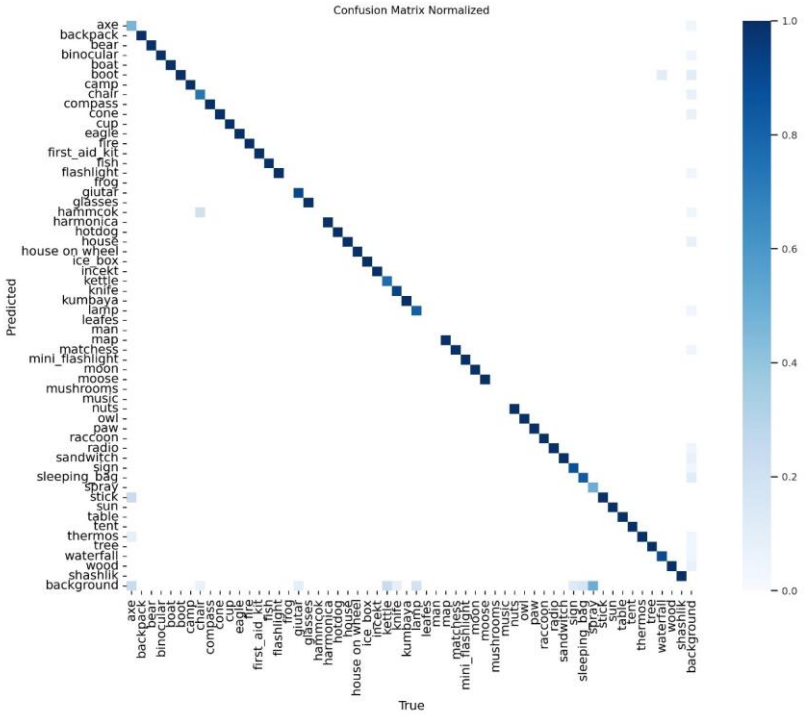


Рисунок 32. Матрица ошибок обученной модели обнаружения объектов, размер M (данных для обучения 225 картинок)

Соответственно из рисунка 32 видно, что классы объектов имена, которых отсутствуют на главной диагонали будут обнаруживаться не надежно, либо вовсе не будут. Таких классов объектов 6. Блекло голубые также требуют дополнительного обучения. Исправление этих недочетов решено было сделать на следующем этапе развития проекта, после регионального отбора.

Для более качественного обучения модели, скрипт подготовки данных с предварительной разметкой был модифицирован. Если в первом случае за источник информации брались результаты классификации отдельных объектов, которые предварительно надо было брать с помощью контуров, как описано выше на схеме рисунка 23, то на данном этапе информацией для разметки уже служили результаты работы модели с обнаружением объектов, представленной на рисунке 32. Использовались все обнаруженные объекты имеющие вероятность более 40%. Если не все из 8 объектов обнаруживались, то пропущенные объекты размечались вручную с помощью ПО Roboflow. Таким образом, всего для обучения было подготовлено и размечено 575 изображений,

при помощи аугментации инструментами Roboflow (повороты, преобразование перспективы, добавление шумов) данное количество было увеличено до 1750 изображений и разбито в пропорции 80% обучение, 10% тренировка, 10% валидация, согласно рекомендаций ресурса Roboflow.

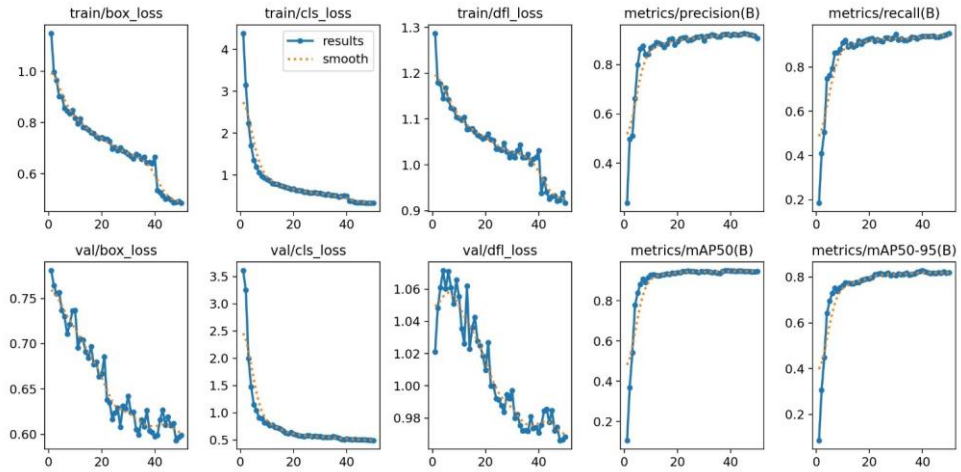


Рисунок 33. метрики качества обучения модели обнаружения объектов, размер M (данных для обучения 225 картинок)

Результаты обучения модели на итоговом объеме данных представлены на рисунке 34. Модель была уменьшена до размера S, тогда ка до этого использовалась модель размера M. Ранее модель размера S не могла пройти успешно цикл обучения, из-за малого для нее количества размеченных данных. Изменение размера модели положительно сказалось на скорости работы программы, подробнее об этом изложено в таблице 4 далее.

На рисунке 34 также видно выпадение двух классов с главной диагонали, однако в данном случае это никак не влияет на качество работы модели. Выпадающие объекты это undefined (неизвестный объект, появившийся в следствии ошибки разметки трех объектов среди всех 1750*8=14000) и background (фон картинки, технический класс используемый roboflow).

В результате тестов было модель размера S обученная на большем количестве данных показала обнаружение объектов с вероятность 80-95%, против вероятностей от 40% до 90% у модели размера M обученной на 225 изображениях. В качестве дополнительного исследования можно дополнить датасет еще 500-1500 изображениями и достичь необходимого количества для обучения модели размера nano, для получения максимальной скорости работы.

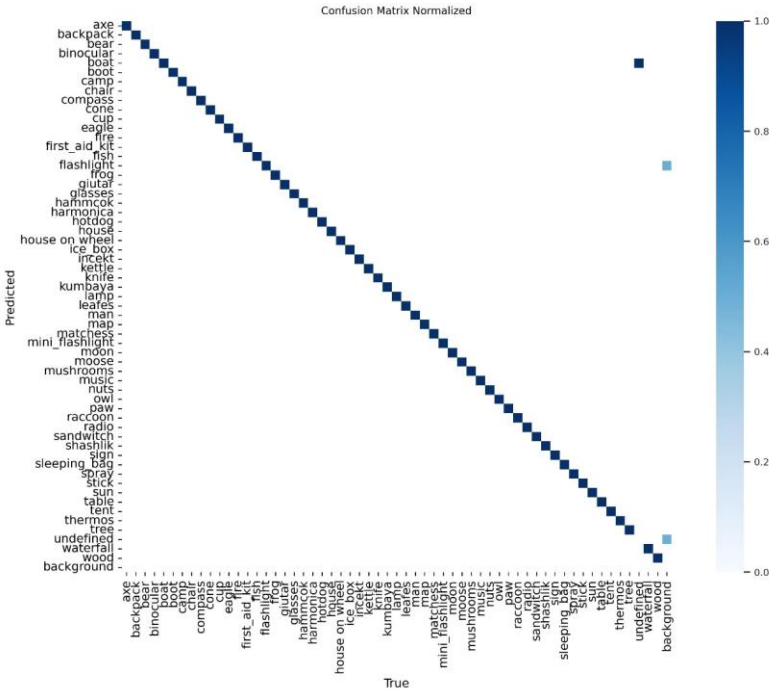


Рисунок 34. Матрица ошибок обученной модели обнаружения объектов, размер S (данных для обучения 1450 картинок)

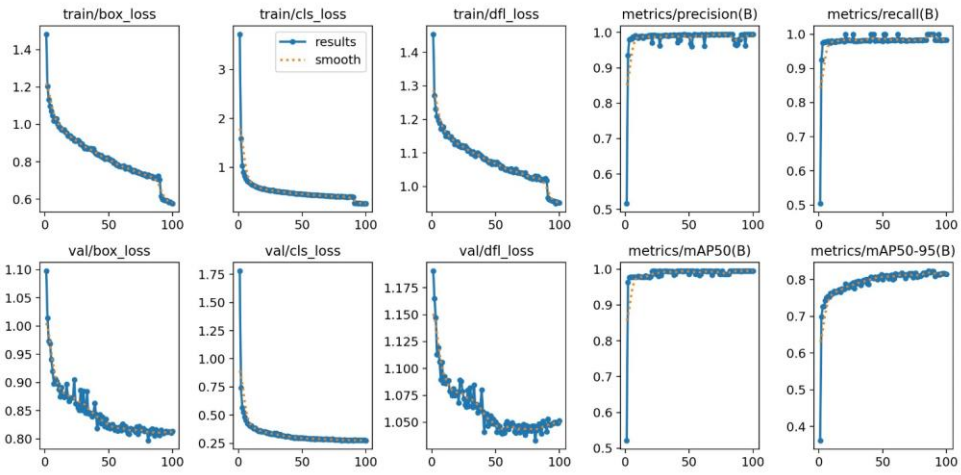


Рисунок 35. метрики качества обучения модели обнаружения объектов, размер M (данных для обучения 225 картинок)

Перейдем к сложной, важной и увлекательной части DevOps как подготовить платформу и библиотеки. Как уже было сказано последняя версия Jetson Nano предоставленная Nvidia это JetPack 4.6 на основе Ubuntu 18.04 с установленным Python 3.6.9. Однако Yolo v8 требует Python не ниже 3.8 и PyTorch>=1.8. Сначала я попробовал обновить python до версии 3.8, потом настроить чтобы не лезла включаться версия 3.6 потом пробовал установить PyTorch 1.11 (Q-engineering, 2023), однако он не захотел ставиться на старую ubuntu ни в какую из-за других зависимых библиотек.

В итоге я пришел к вывод что проще переустановить всю системы и доставить в нее необходимы библиотеки. Спасибо авторам статьи (Q-engineering, 2022) за то, что подготовили образ Ubuntu 20.4 с уже установленными важными компонентами таким как:

OpenCV (4.8.0)

PyTorch (1.13.0)

TorchVision (0.14.0)

TensorRT (8.0.1.6)

Jtop 4.2.1

На этот образ не с первого раза и не без проблем, но удалось поставить ultralytics (Yolo v8) , а благодаря pytoch и OpenCV собранными с поддержкой CUDA сразу в этом образе их даже не пришлось пересобирать чтобы заставить модель тренироваться и обучаться на процессоре GPU.

После всех этих действий у нас есть главный компонент проекта – нейросеть обнаружения объектов, она же машинное обучение!

В таблице 4 приведем анализ скорости работы нейросетей на разных этапах и с разными настройками аппаратной платформы (CUDA vs CPU)

Таблица 5. Анализ скорости работы нейросети

	Jetson Nano B01	Jetson ORIN Nano
Классификация картинок		
CPU, все работает штатно	8 раз по Speed: 25.5ms preprocess, 567.5ms inference, 7ms postprocess per image at shape (1, 3, 320, 320) Итого: 4800мс	8 раз по Speed: 9.5ms preprocess, 127.5ms inference, 0.1ms postprocess per image at shape (1, 3, 320, 320) Итого: 1120мс
Обнаружение объектов		
CPU, все работает штатно, Модель размера M		Speed: 25.0ms preprocess, 114.4ms inference, 0.1ms postprocess per image at shape (1, 3, 448, 448)

CUDA Перегружен процессор, не очищен cache, Модель размера M	0: 448x448 1 cup, 1 glasses, 1 leafes, 1 matchess, 1 mushrooms, 1 owl, 1 paw, 1 table, 3102.3ms Speed: 2361.0ms preprocess, 3102.3ms inference, 3911.7ms	
CUDA, все работает штатно, Модель размера M	0: 448x448 1 axe, 1 fish, 2 glassess, 1 hammcok, 1 house on wheel, 1 mini_flashlight, 1 raccoon, 1 sign, 270.9ms Speed: 7.7ms preprocess, 270.9ms inference, 7.1ms postprocess per image at shape (1, 3, 448, 448)	Speed: 6.9ms preprocess, 8.5ms inference, 0.1ms postprocess per image at shape (1, 3, 448, 448)
CUDA, все работает штатно, Модель размера S	0: 448x448 1 axe, 1 fish, 2 glassess, 1 hammcok, 1 house on wheel, 1 mini_flashlight, 1 raccoon, 1 sign, 270.9ms Speed: 5.7ms preprocess, 143.9ms inference, 6.1ms postprocess per image at shape (1, 3, 448, 448)	Speed: 6.1ms preprocess, 4.5ms inference, 0.1ms postprocess per image at shape (1, 3, 448, 448)

На этом этапе возникло желания добавить еще и синтез и распознавание речи, так как попалась статья (Tikhonnn, 2022) что это легко сделать на базе отечественной библиотеки Yandex SpeechKit. На ноутбуке она установилась сразу и быстро получилось сделать скрипт для генерации имен всех 57 объектов, которые робот может найти на картах, листинг скрипта приведен ниже.

```
from speechkit import Session, SpeechSynthesis
import pyaudio

oauth_token = "y0_AgAEA7qh4_wyAATuwQAAAAD5mqOdAACu7MMZKnZGorfs4oNGiNiEjY9I0g"
catalog_id = "b1g3mrejv8qqnnligebe"


# Экземпляр класса `Session` можно получать из разных данных
session = Session.from_yandex_passport_oauth_token(oauth_token, catalog_id)

sample_rate = 16000 # частота дискретизации должна
                    # совпадать при синтезе и воспроизведении

# Создаем экземляр класса `SpeechSynthesis`, передавая `session`, который уже содержит
нужный нам IAM-токен и другие необходимые для API реквизиты для входа
synthesizeAudio = SpeechSynthesis(session)
```

```
names={0: 'топор', 1: 'рюкзак', 2: 'медведь', 3: 'бинокль', 4: 'лодка', 5: 'ботинок',
56: 'жареный зефир'}

for i in range(len(names_eng)):
    #Метод `.synthesize()` позволяет синтезировать речь и сохранять ее в файл
    print(str(i)+'.wav', names[i])
    synthesizeAudio.synthesize(
        str(i)+'.wav', text='ДОББЛЬ!! '+names[i]+"!",
        voice='alena', emotion='good', format='lpcm', sampleRateHertz=sample_rate
    )
```

Теперь он сможет говорить голосом Алисы! Даже в офлайн так как все эти файлы сохранены локально. Заставить библиотеку PyAudio работать на jetson Nano с USB аудио картой оказалось существенно сложнее, но в итоге все получилось и стала готова библиотека типовых действий робота как по управлению механикой. Так и в части обнаружения объектов и голосового взаимодействия с пользователем.

2.4.4 Итоговый проект. Собираем и дружим все части.

Общая блок-схема работы программы представлена далее на рисунке 37, структура папок и файлов представлены на рисунке 36.

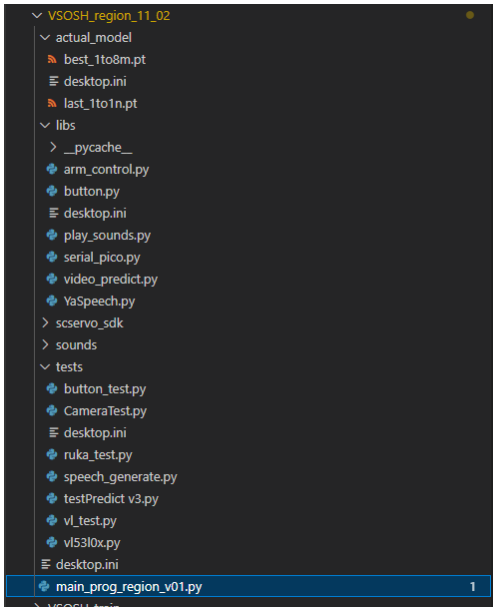


Рисунок 36. Структура папок и программ проекта "Добблинатор"

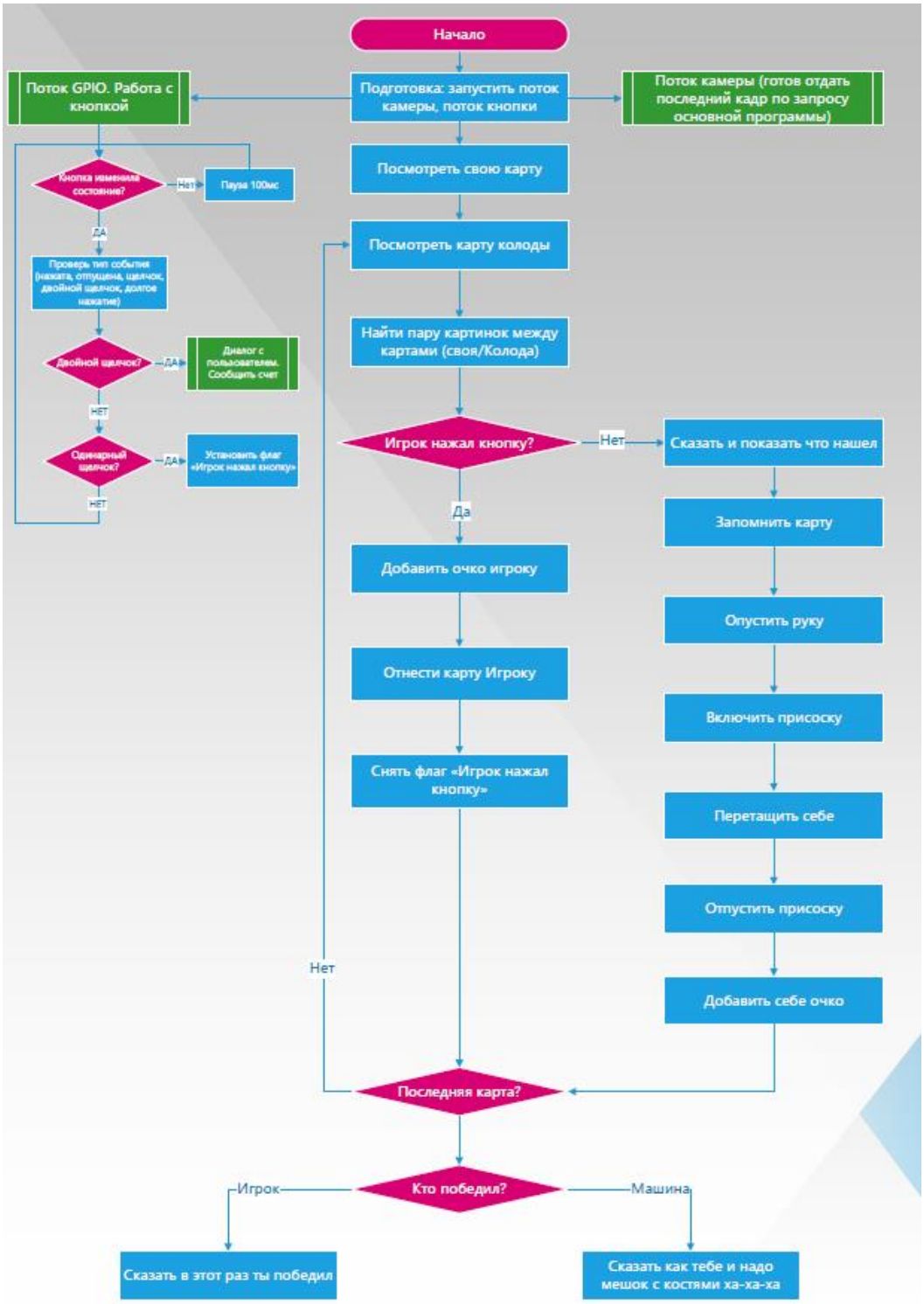


Рисунок 37. Блок схема основной программы

Для настройки yolo отображения рамок полученных объектов Yolo помогла статья (IBRAHIM, 2024).

В папке test собраны программы для тестирования отдельных функциональных блоков робота, таких как кнопка, сервоприводы, камера и распознавание объектов.

В папке libs собраны самописные части программы «библиотеки» которые затем используются основной программой. Импорт их осуществляется следующим образом:

```
from libs.arm_control import set_position
from libs.serial_pico import RPiPico_serial
pico = RPiPico_serial('/dev/ttyTHS1')
from libs.play_sounds import Say_card_name

from libs.video_predict import predict_card, compare_cards, extract_classes
from libs.button import btn_pressed, clean_btn
```

Полный список используемых внешних библиотек приведен далее

```
import sys
import os
import time
# importing GPIO to work with button and vl503
import RPi.GPIO as GPIO
# importing serial to work with serial USB port
import serial
# importing Yolo and Torch for Machine Learning
from ultralytics import YOLO
from ultralytics.utils.plotting import Annotator
import torch
# importing OpenCV and Numpy
import cv2
import numpy as np
# importing modules for Audio play, and recognize
import pyaudio
import io
import wave
```

Интерфейс пользователя во время игры представлен на следующих рисунках 38 и 39

рисунок 38. Интерфейс программы (совпадающие объекты веделены красной рамкой)

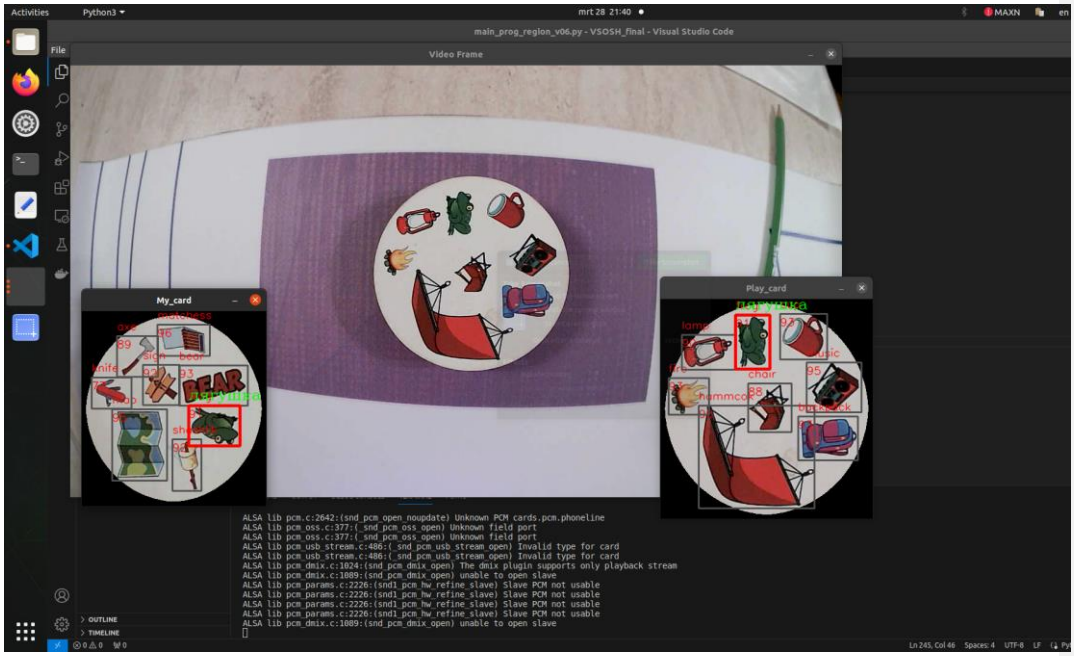


рисунок 39 Интерфейс программы (совпадающие объекты веделены красной рамкой)

ЗАКЛЮЧЕНИЕ

Достигнута цель по созданию действующего робота-манипулятора, способного играть в игру Доббль за счет применения компьютерного зрения и машинного обучения.

В ходе работы полностью успешно выполнены следующие задачи:

1. Механическая конструкция:

- Разработка и создание трехосевой робо-руки с необходимой прочностью и гибкостью для эффективного выполнения задач игры "Доббль".

- Интеграция манипулятора с камерой технического зрения, обеспечивая их взаимодействие и эргономику.

2. Распознавание образов:

- Интеграция камеры технического зрения, работающей на платформе Jetson Nano, для съемки изображений игровых карточек.

- Настройка и обучение системы распознавания с использованием OpenCV и нейронной сети для точного определения объектов на картах. Выбор подходящей нейронной сети.

3. Синтез речи:

- Интеграция модуля синтеза речи, способного преобразовывать распознанные объекты в аудио-сигналы.

- Настройка синтезатора речи для четкого и понятного произнесения наименований обнаруженных объектов.

4. Управление и программирование:

- Разработка управляющей программы на языке Python для координации работы робота, взаимодействия с камерой и синтезатором речи.

- Программирование механизмов управления движением робо-руки, включая кинематику и обратную кинематику.

5. Интеграция с игрой "Доббль":

- Разработка механизмов взаимодействия с игровыми карточками, позволяющих роботу эффективно участвовать в игре.

- Тестирование и отладка работы "Добблинатора" в контексте игры "Доббль".

6. Оптимизация и улучшение производительности:

- Оптимизация программного обеспечения и алгоритмов для обеспечения быстрого и точного распознавания объектов.

- Улучшение системы управления и реакции робота для создания плавного и естественного взаимодействия с ребенком в ходе игры.

7. Безопасность:

- Разработка системы безопасности, включая датчики препятствий и механизмы аварийного

останова, чтобы предотвратить возможные травмы.

8. Тестирование и доработка:

- Проведение тестовых игр с участием робота и детей для выявления возможных проблем и доработки системы на основе обратной связи.

ПРАКТИЧЕСКАЯ ЗНАЧИМОСТЬ проекта:

для ОБЩЕСТВА: Дополнительная возможность увлечь подростков новыми IT-технологиями через игровые технологии и вовлечение в процесс познания;

для Лицея: Собран отличный робот, который может играть в игры с учениками, а может быть использован в других проектах

для меня: освоена библиотеки OpenCV, Yolo v8; изучены основы управления Ubuntu

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. **2.701-2008 ГОСТ** Единая система конструкторской документации (ЕСКД). Схемы. Виды и типы. Общие требования к выполнению. // ГОСТ 2.701-2008.
2. **3DGram** Jetson Nano против Raspberry Pi 4: Различия [В Интернете] // 3DGram. - https://3dgram.ru/jetson-nano-protiv-raspberry-pi-4-razlichija/.
3. **agradov** Что такое робот? [В Интернете] // Хабр. - 19 06 2020 г.. - https://habr.com/ru/articles/507334/.
4. **augorelov** Схемы электрические. Типы схем [В Интернете] // Хабр. - 12 05 2019 г.. - https://habr.com/ru/articles/451158/.
5. **Buhl Nikolaj** YOLO models for Object Detection Explained [YOLOv8 Updated] [В Интернете] // encord. - 8 02 2023 г.. - https://encord.com/blog/yolo-object-detection-guide/.
6. **dimanosov007** Tutorial к автоматизации разметки изображений с использованием OpenCV Python [В Интернете] // Хабр. - 09 12 2022 г.. - https://habr.com/ru/articles/704546/.
7. **Dinxor** Доббль: практичный подход с OpenCV и NumPy [В Интернете] // Хабр. - 13 01 2021 г.. - https://habr.com/ru/articles/537222/.
8. **DmitrySpb79** NVIDIA Jetson Nano: тесты и первые впечатления [В Интернете] // Хабр. - 19 07 2019 г.. - https://habr.com/ru/articles/460723/.
9. **engineer Computer vision** IMAGE CLASSIFICATION with Yolov8 custom dataset | Computer vision tutorial [В Интернете] // youtube. - 05 04 2023 г.. - https://www.youtube.com/watch?v=ZeLg5rxLGLg&t=874s.
10. **Feetech** feetech-servo-sdk 1.0.0 [В Интернете]. - FeeTech. - https://pypi.org/project/feetech-servo-sdk/ (https://gitee.com/ftservo/SCServoSDK).
11. **Germanov Andrey** How to Detect Objects in Images Using the YOLOv8 Neural Network [В Интернете] // freeCodeCamp. - 04 05 2023 г.. - https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/.
12. **Harder Hennie de** How I taught my computer to play Spot it! using OpenCV and Deep Learning [В Интернете] // Towards Data Science. - 21 04 2020 г.. - https://towardsdatascience.com/how-i-learned-my-computer-to-play-spot-it-using-opencv-and-deep-learning-ad1f017a3ec3.
13. **IBRAHIM** Integrating OpenCV with YOLOv8: Obtaining Bounding Box Coordinates – Yolo [В Интернете] // deyCODE. - 18 01 2024 г.. - https://deycode.com/posts/integrating-opencv-with-yolov8-obtaining-bounding-box-coordinates/.
14. **Kukil** Roadmap To an Automated Image Annotation Tool Using OpenCV Python [В Интернете] // LearnOpenCV. - 06 12 2022 г.. - https://learnopencv.com/automated-image-annotation-tool-using-opencv-python/#Automated-Image-Annotation-Tool-Demo.
15. **Loghin Dumi** Raspberry Pi 4 or Jetson Nano — Which One Is Better? [В Интернете] // Medium. - 16 11 2020 г.. - https://dloghin.medium.com/raspberry-pi-4-or-jetson-nano-which-one-is-better-b0a9d185abb6.
16. **LudoTech** OLEM, the boardgame robot by LudoTech [В Интернете] // kickstarter.com. - LudoTech, 2022 г.. - https://www.kickstarter.com/projects/ludotech/olem-the-boardgame-robot.

17. **MaxRokatansky** Как я научила свой компьютер играть в Доббль с помощью OpenCV и Deep Learning [В Интернете] // Хабр. - 24 04 2020 г.. - https://habr.com/ru/companies/otus/articles/498800/.

18. **Mojobot** mojobot.io [В Интернете] // Mojobot. - Mojobot. - https://www.mojobot.io/.

19. **Q-engineering** Install PyTorch on Jetson Nano. [В Интернете] // Q-engineering. - 11 04 2023 г.. - https://qengineering.eu/install-pytorch-on-jetson-nano.html.

20. **Q-engineering** Install Ubuntu 20.04 on Jetson Nano [В Интернете] // Q-engineering. - 30 01 2022 г.. - https://qengineering.eu/install-ubuntu-20.04-on-jetson-nano.html.

21. **RyabovA** Управление сервоприводами, часть 2. Управляем сервоприводами с помощью серво-контроллера через USB любых компьютеров [В Интернете] // Хабр. - https://habr.com/ru/articles/750226/.

22. **SenseTime** SenseTime Unveils SenseRobot Go: The Second AI-Powered Board Game Robot Joins the SenseRobot Family [В Интернете] // SenseTime. - SenseTime, 14 06 2023 г.. - https://www.sensetime.com/en/news-detail/51166774.

23. **Sojasingarayar Abonia** Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms [В Интернете] // Medium. - 29 08 2022 г.. - https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc.

24. **Tikhonnn** Учимся использовать Yandex SpeechKit с помощью Python за 5 минут [В Интернете] // Хабр. - 09 08 2022 г.. - https://habr.com/ru/articles/681566/.

25. World Robotics 2023 Report [В Интернете] // https://ifr.org/ifr-press-releases/news/world-robotics-2023-report-asia-ahead-of-europe-and-the-americas. - IFR. International Federation of Robotics, 26 09 2023 г..

26. **А. Л. Каткова Е. С. Булычева, А. А. Каткова** Влияние настольных игр на социализацию подростков [Журнал] // Наука о человеке: гуманитарные исследования. - 2022 г.. - 2 : Т. 16.

27. **Лошинская Е.А. Кнышева Е.Р., Горбачева А.П., Гринцова Е.Я.** Настольные игры вместо психологов [В Интернете]. - Sciencely — Умный журнал. Естественные науки для детей., 18 04 2023 г.. - https://dzen.ru/a/ZD5q4k1vQDOCxhZA.

28. **РТК ГНЦ РФ ЦНИИ** ГОСТ Р 60.0.0.4-2019/ИСО 8373:2012 // «Роботы и робототехнические устройства. Термины и определения». - 2019 г..

29. **Тимошкин М.С. Миронов А.Н. , Леонтьев А.С.** СРАВНЕНИЕ YOLO V5 И FASTER R-CNN ДЛЯ ОБНАРУЖЕНИЯ ЛЮДЕЙ НА ИЗОБРАЖЕНИИ В ПОТОКОВОМ РЕЖИМЕ [Журнал] // Международный научно-исследовательский журнал. - 6 (120) часть 1.

ПРИЛОЖЕНИЕ 1

Листинг основной программы и чертежи робота Добблинатор

Они находятся на яндекс диске: ссылка

ЧЕК-ЛИСТ содержания пояснительной записки.

Критерии оценки проекта			Баллы	По факту	Отражено в тексте реферата
Поясни-тельная записка (10)	**1**	**Содержание и оформление документации проекта**	**10**		
	1.1	Общее оформление (ориентация на ГОСТ 7.32-2017) (баллы суммируются): 0 – оформлено без ориентации на ГОСТ; 0,5 – соблюдены общие требования ГОСТ к форматированию текста, нумерации страниц и разделов; 0,5 – соблюдены требования ГОСТ к иллюстрациям и таблицам.	**0-1**		Times New Roman, 12pt, выравнивание по ширине, 1.5 отступы абзацев
	1.2	Качество теоретического исследования	**0-3**		
		1.2.1 Обоснование актуальности. Формулировка цели и задач, результата и выводов (баллы суммируются): 0,25 – наличие обоснованной актуальности; 0,5 – корректно сформулированы цель и задачи; 0,25 – наличие описания полученного результата и выводов.	0-1		Пункт 1.1 страницы 4-5
		1.2.2. Сбор и анализ информации по исследуемой проблеме (баллы суммируются): 0,5 – представлена информация о прототипах и аналогах по исследуемой проблеме, с корректными ссылками на авторов; 0,5 – присутствует анализ и выводы по собранной информации.	0-1		Пункт 1.2. Выводы из анализа на странице 10, пункт 1.2.3
		1.2.3 Разработка идеи и концепции робота. Формулировка технического задания (баллы суммируются): 0,25 – присутствует описание идеи и концепции робототехнического устройства; 0,25 – присутствует обоснование соответствия понятию «робот» в соответствии с комплексом ГОСТ Р 60; 0,25 – присутствует обоснование креативности или новизны предложенной идеи, ее практической значимости и перспектив применения готового устройства; 0,25 – присутствует формулировка технического задания.	0-1		Пункт 1.3 Техническое задание, идея концепции и новизны Пункт 1.4 Обоснование что создан робот
	1.3	Разработка технологического процесса	**0-6**		

	1.3.1 Описание процесса проектирования, изготовления, программирования, отладки, модификации проекта (баллы суммируются): 0,25 – присутствует описание процесса проектирования в САПР конструкции робототехнического устройства или его частей; 0,25 – присутствует описание процесса проектирования в САПР электроники робототехнического устройства или его частей; 0,25 – присутствует описание процесса изготовления робототехнического устройства; 0,5 – присутствует описание процесса программирования с указанием структуры созданного ПО и описания реализованных алгоритмов управления; 0,75 – присутствует описание процесса отладки и модификации проекта со сбором и анализом промежуточных результатов.	0-2		Описание САПР – пункт 2.1 страница 16 Изготовление механики пункт 2.2. страница 19 Описание программирования пункт 2.4 Отладка на примере обучения двух типов сетей пункт 2.4.3 страницы 34-36 и страницы 39-40 про качество обучения моделей
	1.3.2 Качество схем, чертежей и другой документации (баллы суммируются): 0,5 – присутствует структурная схема устройства, 0,25 – структурная схема Э1 выполнена без грубых ошибок в соответствии с ГОСТ; 0,25 – присутствует электрическая принципиальная схема Э3 или чертеж самостоятельно спроектированной части устройства; 0,25 – эта схема или чертеж выполнен без грубых ошибок в соответствии с ГОСТ; 0,25 – присутствует блок-схема алгоритма (или UML-диаграмма); 0,25 – присутствуют фрагменты кода программы, и они отвечают требованиям читаемости и лаконичности; 0,25 – присутствуют другие виды документов, например, сборочный чертеж, спецификация, инструкция.	0-2		Схемы Э1 и Э3 представлены на страницах 27 и 28 (рисунки 18 и 19) Блок схема основной программы : рис 35, страница 44 Спецификация основных компонент таблица 3. Чертежи раздел 2.1 рисунки 9-11, презентация, доп. материалы к ПЗ Фрагменты кода программ: Раздел 2.4
	1.3.3 Обоснование выбора материалов, электронных компонентов, технологий проектирования и изготовления (баллы суммируются): 0,5 – присутствует обоснование выбора материалов, технологий и инструментов для изготовления устройства и его частей; 0,5 – присутствует обоснование выбора электронных компонентов для проекта; 0,5 – присутствует обоснование выбора технологий и инструментов проектирования конструкции и электроники робота; 0,25 – присутствует обоснование выбора технологий и средств создания программного обеспечения.	0-2		Обоснование выбора: - материалов раздел 2.2 страница 19 - электроники раздел 2.3, таблица 1, страницы 24-25 -технологий проектирования схемы таблица 2, раздел 2.3 -технологий и средств создания ПО раздел 2.4 страницы 29-31