

Государственное бюджетное общеобразовательное учреждение города  
Москвы «Школа № 1557 имени Петра Леонидовича Капицы»

**ТВОРЧЕСКИЙ ПРОЕКТ**  
**РОБОТ ДЛЯ ДАЧНОГО УЧАСТКА**

Работу выполнил:

Ученик 11 «Д» класса

Козьмин Андрей Викторович

Руководитель:

Учитель технологии

Нигматулин Руслан Равильевич

Зеленоград, 2024 г.

## Содержание

Реферат.....	6
Цель работы.....	6
Методология проведения работы .....	6
Ключевые слова .....	6
Результаты работы .....	6
Область применения .....	6
Выводы.....	6
Введение .....	7
Рисунок 1 - Дачный участок, на который хозяину не хватает времени .....	7
Актуальность.....	7
Таблица 1 - Примеры бытовых роботов.....	8
Теоретическое исследование .....	9
Рисунок 2 - Пример промышленного сельскохозяйственного робота .....	9
Рисунок 3 – Диаграмма рынка сельхоз роботов.....	9
Проблематика .....	10
Таблица 2 - Некоторые типовые проблемы мобильных уличных роботов	10
Сбор и анализ информации по исследуемой проблеме.....	11
Рисунок 4 – Аспекты безопасности автономных роботов по мнению пользователей (ист. BCG, World Economic Forum) .....	11
Цель.....	12
Задачи.....	12
Поисково-исследовательский этап .....	13
Разработка идеи и концепции проекта .....	13
Рисунок 5 - Выбранный полезный дачный функционала .....	13
Рисунок 6 - Выбранная структура взаимодействия ROS1 и FreeRTOS ...	13
Выбор устройств для системы управления .....	14
Таблица 3 - Варианты систем управления роботом.....	14

Анализ систем навигации .....	15
Таблица 4 - Технологии системы навигации .....	15
Выбор системы машинного зрения.....	16
Таблица 5 - Варианты систем машинного зрения.....	16
Выбор вариантов шасси робота.....	16
Таблица 6 - Варианты шасси робота .....	16
Соответствие проекта определению “Робот” по ГОСТ.....	17
Рисунок 7 – Обоснование соответствия проекта определению “Робот” по ГОСТ .....	17
Формулировка технического задания .....	18
Таблица 7 - Техническое задание .....	18
Разработка технологического процесса .....	18
Таблица 8 – Используемые языки программирования, средства производства и комплектующие .....	18
Таблица 9 - Эксплуатационные требования к изделию.....	19
Описание процесса изготовления деталей .....	20
Таблица 10 - Материалы деталей рамы .....	20
Эстетический вид и качество робота .....	21
Конструкторско-технологический этап .....	21
Конструирование робота в САПР.....	21
Рисунок 8 - 3D вид шасси робота.....	22
Рисунок 9 - Новый вариант крепления двигателя.....	22
Собранное шасси робота.....	23
Рисунок 10 - Собранное шасси робота .....	23
Структурная схема устройства .....	24
Рисунок 11 - Компоновка .....	24
Навигация робота по GPS .....	24
Рисунок 12 – Структурная схема обработки данных GPS .....	24
Рисунок 13 - Возможная траектория по участку: от объекта 1 к 2.....	25

Процесс ввода GPS координат.....	25
Рисунок 14 - Скриншот приложения.....	26
Фрагмент программного кода .....	26
Использование машинного зрения .....	27
Рисунок 15 - Алгоритм работы машинного зрения .....	27
Алгоритм управления движением робота .....	28
Рисунок 16 - Схема управления движением робота .....	28
Таблица 11- Выбор микроконтроллера и драйвера двигателей .....	29
ROS и FreeRTOS .....	30
Таблица 12 - Топики ROS и передаваемые данные .....	30
Таблица 13 - Задачи FreeRTOS .....	31
Структурная схема робота.....	31
Рисунок 17 – Используемые компоненты.....	31
Рисунок 18 – Структурная схема.....	32
Блок-схема алгоритма.....	32
Рисунок 19 - Общий алгоритм работы.....	32
Используемые языки программирования и процесс отладки кода.....	33
Таблица 14 – Используемые языки программирования .....	33
Рисунок 20 – Средство отладки и его возможности .....	33
Собственные библиотеки .....	34
Реализация регуляторов и конечного автомата .....	34
Электроника, проектирование в САПР EDA KiCad .....	35
Рисунок 21 – Принципиальная схема.....	35
Рисунок 22 – Трассировка платы.....	36
Рисунок 23 – Используемый станок для фрезерования платы “Roland SRM-20” .....	36
Технологическая карта .....	37
Таблица 15 - Технологическая карта.....	37
Выводы.....	38

Рисунок 24 – Вид готового робота .....	38
Креативность и новизна продукта.....	39
Рисунок 25 - Элементы системы ЗОЯ.....	39
Рисунок 26 – Элементы КОЗА.....	39
Рисунок 27 – Креативность и новизна .....	40
Задачи для следующего этапа .....	40
Экологическая оценка .....	41
Экономическая оценка проекта .....	41
Приложения.....	41
Приложение: Листинг кода .....	42
Приложение: Используемое ПО .....	45
Приложение: Принципиальная и структурная схемы .....	46

## **Реферат**

Отчёт: 41 страница, 27 рисунков, 15 таблиц.

### **Цель работы**

Создать прототип дачного робота, имеющего необходимый функционал: оптимальную систему навигации (с использованием GPS, датчика компаса, машинного зрения); систему полива ядохимикатами и удобрениями.

### **Методология проведения работы**

поиск, анализ, выявление недостатков, сравнение с аналогами, моделирование, изготовление, эксперимент. В процессе работы проводилось исследование концепций построения мобильных автономных роботов, разработка, изготовление и апробация изделия.

### **Ключевые слова**

Автономный мобильный робот, систем навигации, машинное зрение, разработка мобильных приложений для ОС Android, ROS, FreeRTOS, автоматизация сельскохозяйственных работ, типы шасси, 3Д-печать, ЧПУ.

### **Результаты работы**

Изделие – полногабаритный работающий прототип, включая механическую конструкцию, электросиловую часть, электронику, алгоритмы и коды программ, мобильное приложение – интерфейс пользователя.

### **Область применения**

Использование прототипа как первого этапа, платформы для построения рабочего варианта дачного автономного мобильного робота для обработки растений.

### **Выводы**

Цели и задачи данного этапа проекта выполнены.

## **Введение**

Одна из основных причин, для чего разрабатываются роботы, это замена рутинного человеческого труда на автоматизированный. Мой проект робота как раз соответствует этой идее робототехники. Один из знакомых нашей семьи, зная, что я занимаюсь робототехникой, предложил мне такую идею – сделать робота, который выполнял бы различные хозяйственные работы на дачном участке, так как у этого знакомого как раз есть достаточно большой участок земли, для обработки которого ему не хватает времени.



**Рисунок 1 - Дачный участок, на который хозяину не хватает времени**

## **Актуальность**

Дачные участки есть у многих, но далеко не у всех достаточно времени для работы на них. Пока что сельхоз роботов с развитым функционалом можно встретить только на крупных агропроизводствах, и то только на современных. Хотя, интеллектуальные роботизированные устройства можно встретить в квартирах многих граждан. Это, например роботы-пылесосы,

роботы мойщики окон, автоматизированные системы умного дома и так далее.  
Думаю, вполне актуально сделать робота для дачи.

<p>Робот пылесос.</p> <p>Оснащён лидаром, Wi-Fi модулем, ИК датчиком нахождения базы, датчик упора в препятствие, энкодеры на колёсах, датчик уровня высоты от пола, компас и гироскоп, камера с машинным зрением и др.</p>	
<p>Робот мойщик окон.</p> <p>Оснащён механизмом удержания на стекле.</p>	
<p>Робот газонокосилка.</p> <p>Особая важность в таких роботах придаётся безопасности.</p>	

**Таблица 1 - Примеры бытовых роботов**



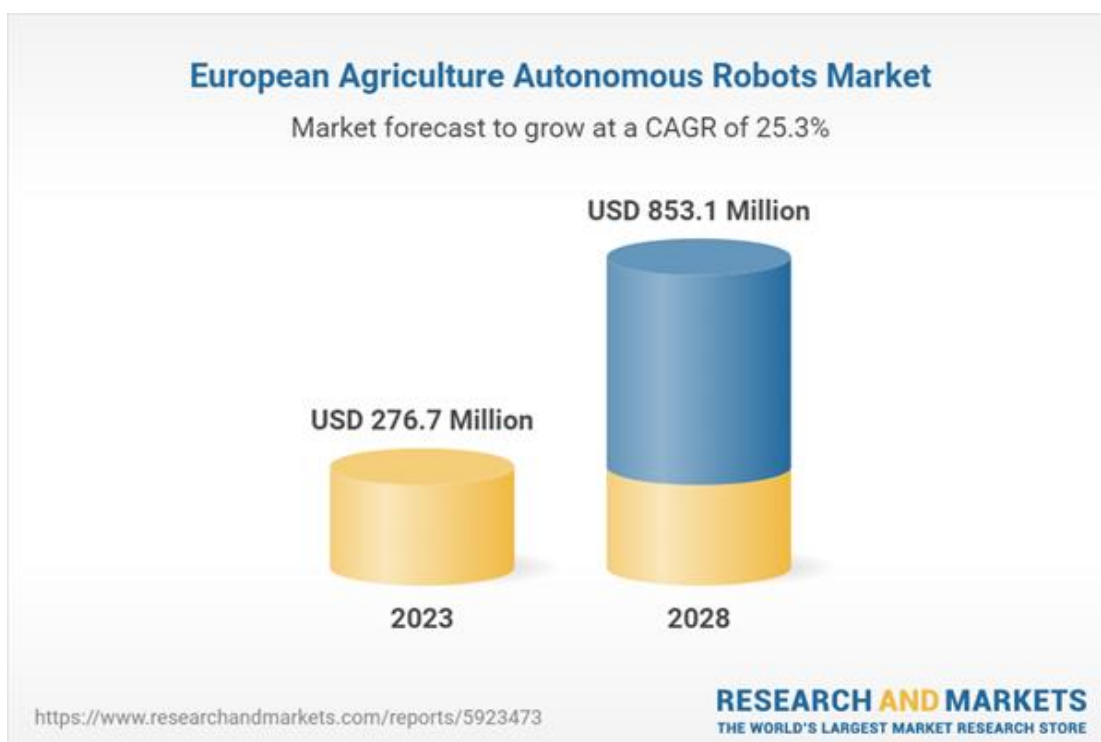
## Теоретическое исследование

Объём сельскохозяйственной продукции, производимой роботами, постоянно растёт. С другой стороны, есть тенденция постепенного перехода технологий из сферы массового производства в сферу индивидуального применения.



**Рисунок 2 - Пример промышленного сельскохозяйственного робота**

На диаграмме показаны результаты одного из исследований рынка сельхоз роботов.



**Рисунок 3 – Диаграмма рынка сельхоз роботов**

## Проблематика

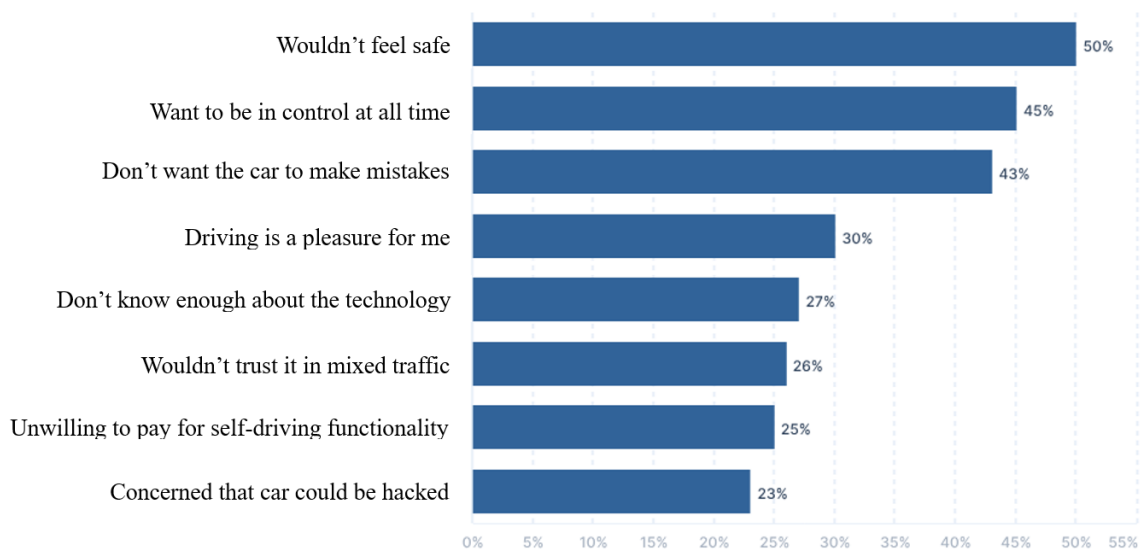
Основная проблематика использования уличных мобильных роботов для частного пользователя – это сложность конструкции и, как следствие, высокая цена и сложность обслуживания. В отличие от робота, который ездит по квартире или моет окна, робот для дачного участка будет находиться в гораздо более разнообразных условиях, то есть это полноценный уличный мобильный робот. Уличные мобильные роботы пока что в основном используются большими компаниями. В своём проекте я попытаюсь решить эту проблематику и сделать робота, приемлемого по простоте обслуживания и доступного по цене.

<p>Сложности передвижения по пересечённой местности. Пример: робот-доставщик застрял в снегу</p>	
<p>Сбой канала управления и навигации. Робот «потерял» сигналы навигации и «заблудился».</p>	
<p>Ошибки машинного зрения. Автомобиль Тесла в режиме автопилота не распознаёт препятствие и врежется.</p>	

**Таблица 2 - Некоторые типовые проблемы мобильных уличных роботов**

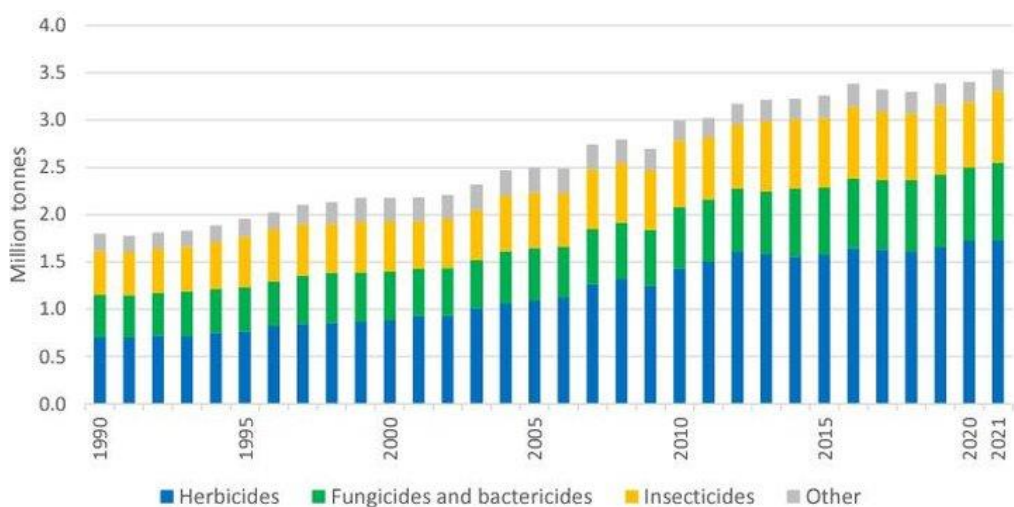
## Сбор и анализ информации по исследуемой проблеме

Рассмотрим такой аспект проблематики, как аварийность автономных мобильных роботов. Согласно исследованиям, число аварий с участием автономных автомобилей снижается. Пользователи отличают аспекты безопасности автомобилей-роботов от обычных автомобилей.



**Рисунок 4 – Аспекты безопасности автономных роботов по мнению пользователей (ист. BCG, World Economic Forum)**

Что касается сельскохозяйственных роботов, важнейшая проблематика, которую они решают – это избавление сотрудников от непосредственного контакта с ядохимикатами, точное дозирование ядохимикатов и удобрений, что однозначно положительно сказывается на здоровье сотрудников и экологии, что показано на графике одного из исследований:



Итак, на основе изучения опыта различных проектов, в которых создаются уличные мобильные роботы для различных целей, понятно, что требуется решить много задач в конструкции и в программном обеспечении.

Одна из важных задач в создании такого робота – выбрать оптимальную систему навигации, не дорогую для учебного проекта, и не сверхсложную с точки зрения алгоритмов, но при этом достаточную для задач использования робота на дачном участке. Поэтому цель проекта я сформулировал так:

## **Цель**

Создать прототип дачного робота, имеющего необходимый функционал: оптимальную систему навигации (с использованием GPS, датчика компаса, машинного зрения); систему полива ядохимикатами и удобрениями.

## **Задачи**

Проект я делаю в несколько этапов.

Главными задачами на текущем этапе были следующие:

- 1) Сконструировать шасси.
- 2) Разработать концепцию навигации робота на дачном участке.
- 3) Использовать в качестве бортового компьютера архитектуру x86\_64 с установленной ОС Linux.
- 4) Использовать фреймворк ROS (Robot Operating System) для управления всеми подсистемами робота.
- 5) Использовать операционную систему реального времени FreeRTOS, установленную на микроконтроллере ESP32.
- 6) Отладить алгоритм для навигации робота и программы машинного зрения.

## Поисково-исследовательский этап

### Разработка идеи и концепции проекта

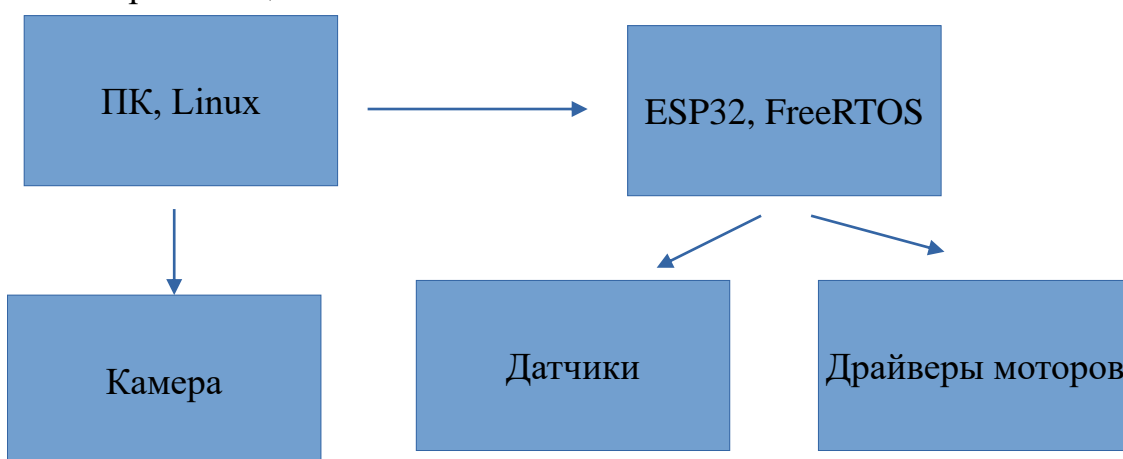
На первом этапе было важно ограничить выбор, чтобы реализовать какой-то базовый, но рабочий вариант.

На данном этапе я остановился на следующем полезном функционале: распыление ядохимикатами и полив удобрениями; так же предполагается установить звуковую сигнализацию для экстренных случаев.



**Рисунок 5 - Выбранный полезный дачный функционала**

Для мобильного робота очень важно грамотно выбрать структуру управляющего программного обеспечения. Я выбрал взаимодействие ROS и FreeRTOS в реализации ROS1 и roserial.



**Рисунок 6 - Выбранная структура взаимодействия ROS1 и FreeRTOS**

## Выбор устройств для системы управления

Система управления мобильными роботами могут строится по разной иерархии, с использованием различных структурных элементов и различных электронных устройств.

Вариант системы управления	Характеристики
Управление одним микроконтроллером (Atmega, ESP, STM32 и др.)	Относительная простота в разработке, доступность микроконтроллеров, невысокая цена. Недостатки: меньшая гибкость в алгоритмах, недостаточная производительность, сложность реализации машинного зрения
Использование компьютера или микрокомпьютера	Позволяет использовать так же и микроконтроллеры на нижестоящих ступенях, большая производительность, возможность использования разных языков программирования, полноценного машинного зрения. Недостатки: больше сложность, стоимость
Использование компьютера и ROS	Позволяет использовать общеотраслевые приёмы промышленной робототехники. Недостаток: ещё большая сложность, высокий порог входа.
Различные комбинированные и специализированные системы	Оптимальны для конкретных применений. Недостатками могут быть сложность в разработке, недоступность компонентов и др.

**Таблица 3 - Варианты систем управления роботом**



## Анализ систем навигации

Я выбрал следующие технологии навигации для своего проекта:


Вид системы навигации	Преимущества	Недостатки
<p>GPS</p> 	<p>Готовая рабочая система навигации, во многие устройства встроены соответствующие блоки приёма и обработки. Хорошо пригодна в случае требуемой точности порядка 10 метров.</p>	<p>Точность для моего сценария применения недостаточна, т.к. на дачном участке робот должен подъезжать к объектам с точностью до <math>\pm 0.5</math> метра.</p>
<p>Навигация по машинному зрению и системе датчиков</p> 	<p>Должна хорошо распознавать большинство объектов в сложной окружающей обстановке, обладает высокой точностью позиционирования.</p>	<p>Работает только в ближней зоне. Высокоэффективная система требует сложной разработки, опытной команды, и как следствие, дорогая и, в большинстве случаев коды не предоставляются для открытого доступа</p>
<p>Компас</p> 	<p>Позволяет совместно с GPS ориентировать робота по азимуту.</p>	<p>Недорогие и доступные модули, которые можно использовать в школьном проекте, не очень точные и работают нестабильно.</p>

Таблица 4 - Технологии системы навигации

## Выбор системы машинного зрения

Система машинного зрения	Характеристики
OpenCV	Распространённая, гибкая система. Может потребовать относительно большой производительности компьютера.
ML Kit (Machine Learning Kit, Google)	Предоставляет удобный интерфейс. Недостаток: недостаточно высокая распространённость, как следствие, мало документации; работает только на OS Android.
Системы собственной разработки	Оптимальны для использования, но требуют большого времени и опыта в разработке.

**Таблица 5 - Варианты систем машинного зрения**

## Выбор вариантов шасси робота

Шагающий робот. Высокая проходимость, сложность механической конструкции, сложность алгоритмов.	
Колёсный робот. Типовая распространённая конструкция, могут быть различные варианты, не всегда достаточная проходимость.	
Гусеничный робот. Типовая конструкция, высокая проходимость, несколько большая сложность по сравнению с колёсным вариантом	

**Таблица 6 - Варианты шасси робота**



**Соответствие проекта определению “Робот” по ГОСТ**

Согласно ГОСТ Р 60.0.0.10-2023 “Робот” — это устройство-агент, предназначенное действовать в физическом мире для выполнения одной или нескольких задач, имеющее в своем составе систему управления и интерфейс для взаимодействия с окружающим миром. Рассмотрим, на сколько мой проект соответствует определению робота по ГОСТ.



**Рисунок 7 – Обоснование соответствия проекта определению “Робот” по ГОСТ**

## Формулировка технического задания

В итоге, по опыту первого этапа проекта я сформулировал техническое задание в следующем виде:

Пункт технического задания	Выбранный вариант исполнения
Функционал робота на данном этапе проекта	Обработка химикатами и удобрениями отдельных растений и сегментов территории дачного участка.
Система навигации	Комбинированная система навигации с использованием GPS и машинного зрения с использованием средств OpenCV.
Система управления	Использование бортового компьютера с OS Linux. В качестве бортового компьютера использование мини ПК. Использование микроконтроллера ESP32 на нижнем уровне (управление двигателями и др.)
Шасси робота	Гусеничное с 4-мя коллекторными моторами на каждую гусеницу.

Таблица 7 - Техническое задание

## Разработка технологического процесса

На этом этапе выбирались материалы, комплектующие, средства производства, технологические операции и языки программирования.



Таблица 8 – Используемые языки программирования, средства производства и комплектующие

Кроме технического требования к изделию сформулированы эксплуатационные требования, которые приведены ниже в таблице

Требования	Разъяснения
Функциональность	Обеспечение основной выбранной задачи – передвижение по территории дачного участка и обработка растений химикатами и удобрениями.
Безопасность	Изделие должно быть безопасным с точек зрения эксплуатации и используемых материалов.
Прочность	Изделие должно обладать достаточными прочностными характеристиками.
Технологичность и ремонтпригодность	Все производственные операции и технологии должны выполняться с учетом возможностей школьных мастерских.
Управление со смартфона	Смартфон имеется у каждого человек и является удобным инструментом для управления роботом. Для смартфона можно написать мобильное приложение с самым разнообразным функционалом и интерфейсом.

**Таблица 9 - Эксплуатационные требования к изделию**

## Описание процесса изготовления деталей

Для изготовления деталей рамы и шасси были выбраны следующие материалы: фанера толщиной 8 мм, пластик PLA. Детали изготавливались на следующем оборудовании: лазерном станке, 3D принтере.

Материал	Преимущества	Недостатки
Фанера	Недорогой, доступный материал, удобен для обработки на школьном лазерном станке. Фанера выбрана для использования, т.к. на данном этапе работ является полномасштабным прототипом.	В целом фанера не самый подходящий материал для уличного робота, т.к. либо требует специальной обработки от атмосферных воздействий, либо будет недолговечным в использовании.
PLA пластик	Недорогой, доступный материал, удобен для производства деталей на 3D принтере.	Не для всех деталей робота пластик обладает достаточной прочностью, PLA пластик так же относительно неустойчив к атмосферным воздействиям.
Алюминиевые сплавы	Прочные, лёгкие, устойчивые атмосферным воздействиям.	Относительно более дороги. В школьных мастерских недостаточно оборудования для полноценного технологического процесса обработки.

**Таблица 10 - Материалы деталей рамы**

## **Эстетический вид и качество работа**

На данном этапе робот является полномасштабным прототипом, то есть по габаритам он соответствует будущему рабочему варианту, но отличается по используемым материалам.

Эстетический вид на данном этапе выбран исходя из функциональности, и в меньшей степени из-за требований дизайна.

## **Конструкторско-технологический этап**

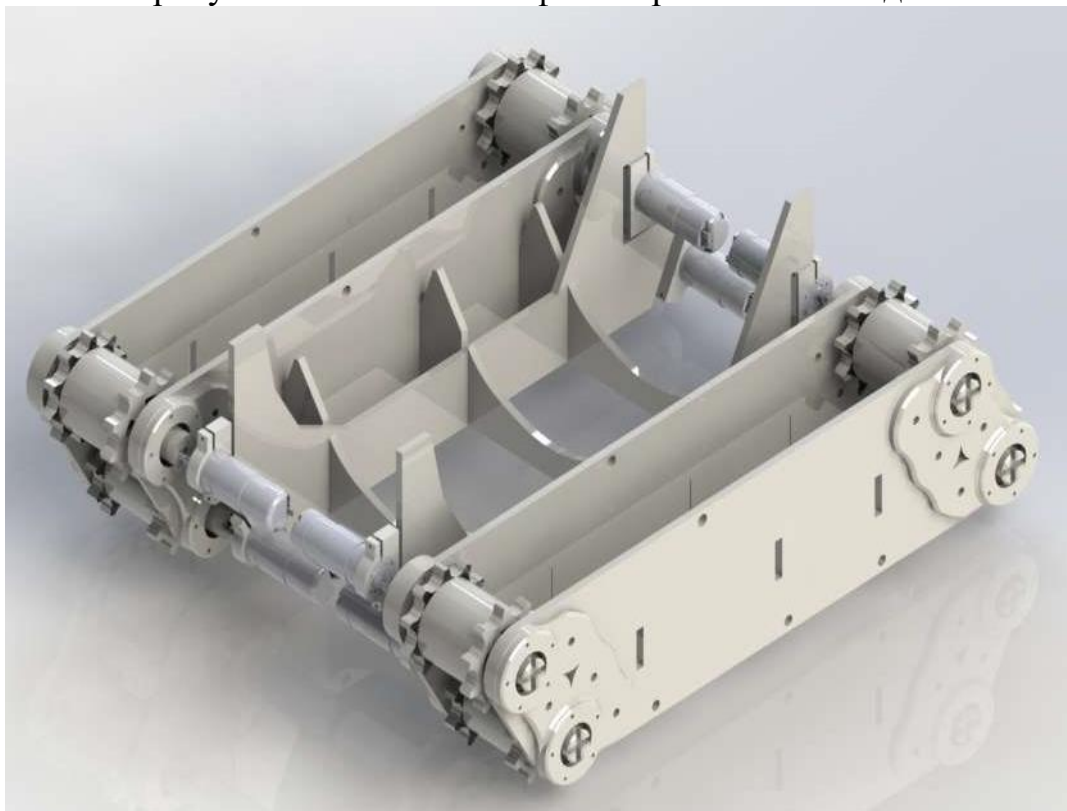
Пункты технического задания, которые я сформулировал на поисково-исследовательском этапе, мне предстояло реализовать на конструкторско-технологическом этапе. Для этого я использовал различное оборудование, инструменты, материалы, программное обеспечение: SolidWorks, Visual Studio Code, Ultimaker Cura, 3D принтер, и др.

Конструктивно было решено делать полномасштабный прототип. То есть габариты робота максимально приближены к рабочей модели, но материалы и некоторые конструктивные решения реализованы с учётом снижения стоимости на данном этапе, а также для апробирования правильности выбора.

## **Конструирование робота в САПР**

Для проектирования использовались трёхмерные системы автоматизированного проектирования (САПР): SolidWorks и Inventor. В итоге была создана 3D модель изделия, в которую вошли все детали, узлы, покупные компоненты.

На рисунке ниже показана фото отрисовка 3D модель шасси.



(разработка и фото-отрисовка выполнены в SolidWorks)

**Рисунок 8 - 3D вид шасси робота**

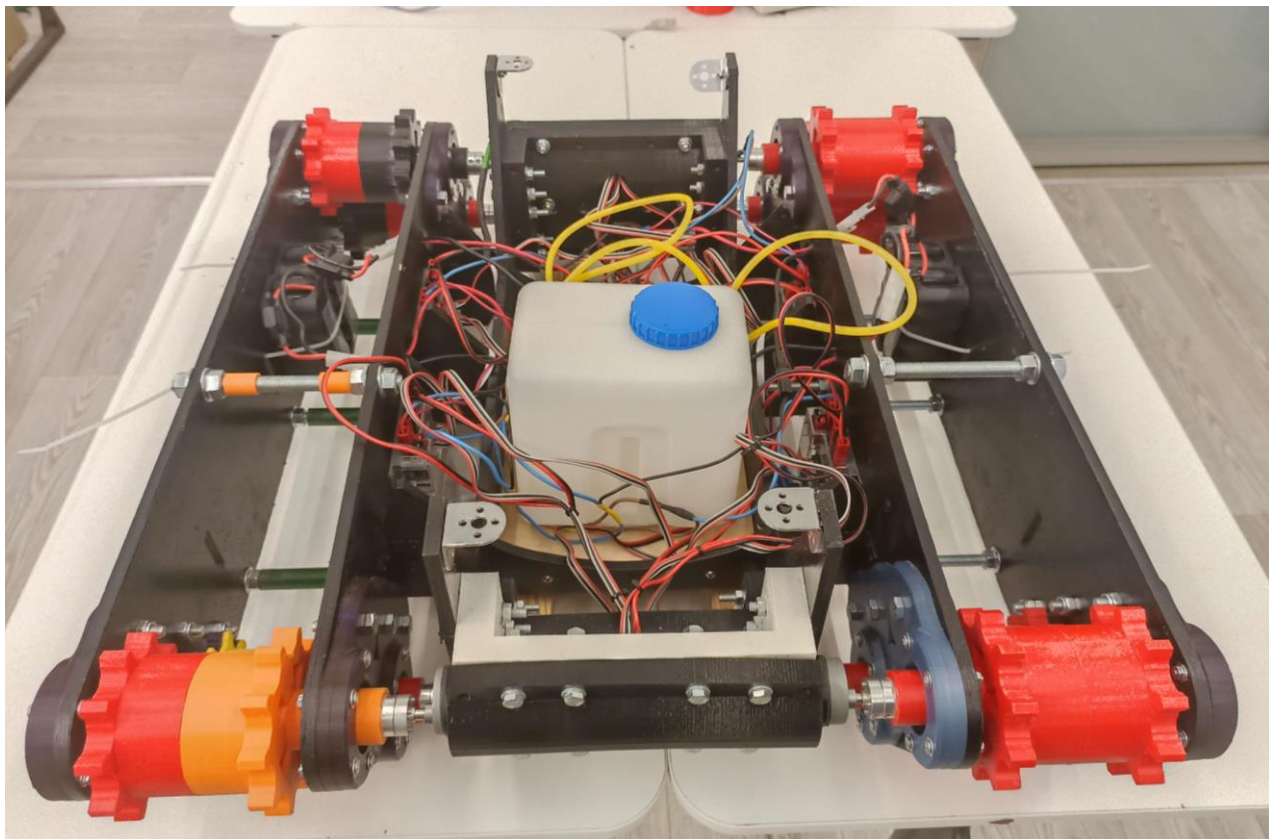
На двигатели приходится значительная нагрузка и вся конструкция робота достаточно массивна, поэтому необходимо надёжное крепление двигателей. Общее время печати кронштейнов двигателей составило более 80 часов, и израсходовано более 1,5 кг пластика.



**Рисунок 9 - Новый вариант крепления двигателя**

## Собранное шасси робота

Ниже показана фотография собранного шасси с установленным баком системы подачи химикатов и удобрений.



**Рисунок 10 - Собранное шасси робота**

Компоновка элементов на раме выполнялась исходя из следующих принципов:

- Эффективное использование пространства внутри рамы;
- Низкое расположение центра тяжести;
- Защита блоков электроники от пыли и влаги на дальнейшем этапе развития проекта;
- Наличие резервного пространства для новых узлов при добавлении функционала;
- Установка достаточного количества аккумуляторов.

## Структурная схема устройства

На рисунке показана расположение компонентов на шасси.

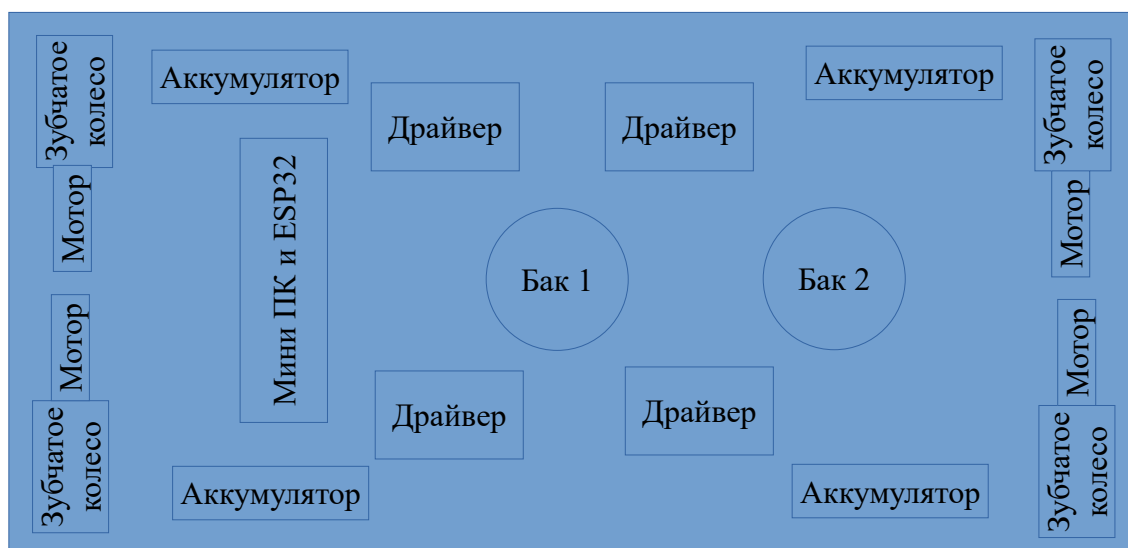


Рисунок 11 - Компоновка

## Навигация робота по GPS

Система разработана с двумя уровнями точности позиционирования. На первом уровне робот движется по GPS с точностью до 5 метров. После того как робот доехал до какого-либо объекта по GPS, он ищет QR-коды, размещённые на объекте, и далее движется более точно, ориентируясь по QR-коду.

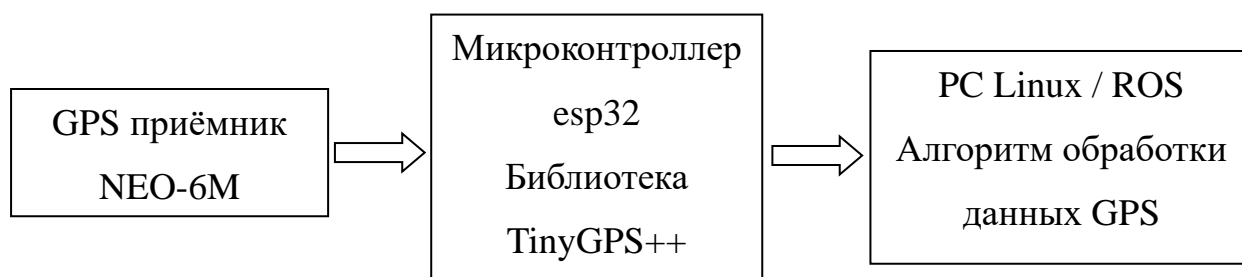


Рисунок 12 – Структурная схема обработки данных GPS



Для работы с GPS я использовал библиотеку TinyGPS++.

На первом этапе робот должен получить информацию о GPS координатах объектов на дачном участке.



**Рисунок 13 - Возможная траектория по участку: от объекта 1 к 2**

### **Процесс ввода GPS координат**

Хозяин дачного участка должен подогнать робота к объекту, используя моё программное обеспечение на смартфоне и выбрать функцию “СОХРАНИТЬ”. Эту процедуру надо повторить для всех целевых объектов дачного участка. Объектами могут быть растения, которые нужно обрабатывать, так же объектами могут быть строения, столкновения с которыми робот должен избежать (дом, гараж, забор и др.).

**Рисунок 14 - Скриншот приложения**

На рисунке показан скриншот экрана смартфона с моим приложением. Приложение называется СЛОН — Система ЛОкальной Навигации. На скриншоте видны управляющие кнопки и текущие GPS координаты робота.



### Фрагмент программного кода

Ниже, как пример, приведён фрагмент моей программы на языке C++ для обработки полученных на ESP32 GPS данных.

```
double get_dist_to_next(double lat1, double long1) {
    double lat2 = get_next_cords().first;
    double long2 = get_next_cords().second;
    double delta = (long1-long2) * MY_DEG_TO_RAD;
    double sdlong = sin(delta);
    double cdlong = cos(delta);
    lat1 = lat1 * MY_DEG_TO_RAD;
    lat2 = lat2 * MY_DEG_TO_RAD;
    double slat1 = sin(lat1);
    double clat1 = cos(lat1);
    double slat2 = sin(lat2);
    double clat2 = cos(lat2);
    delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
    delta = delta * delta;
    delta += clat2 * sdlong * clat2 * sdlong;
    delta = sqrt(delta);
    double denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
    delta = atan2(delta, denom);
    return delta * 6372795;
}
```

```

double get_angle_to_next(double lat1, double long1) {
    double lat2 = get_next_cords().first;
    double long2 = get_next_cords().second;
    double dlon = (long2-long1) * MY_DEG_TO_RAD;
    lat1 = lat1 * MY_DEG_TO_RAD;
    lat2 = lat2 * MY_DEG_TO_RAD;
    double a1 = sin(dlon) * cos(lat2);
    double a2 = sin(lat1) * cos(lat2) * cos(dlon);
    a2 = cos(lat1) * sin(lat2) - a2;
    a2 = atan2(a1, a2);
    if (a2 < 0.0) a2 += MY_TWO_PI;
    return a2 * MY_RAD_TO_DEG;
}

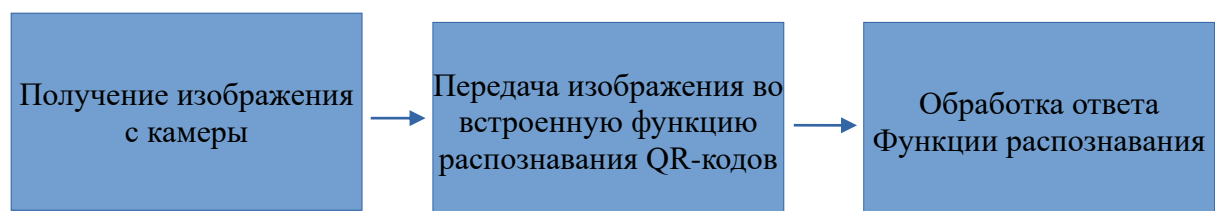
```

В данном фрагменте кода функция *get\_dist\_to\_next* высчитывает дистанцию между текущим и целевым положениями робота, а функция *get\_angle\_to\_next* определяет азимут к целевой точке, после чего эти данные обрабатываются PID регулятором и на ESP32 отправляется управляющий сигнал для моторов.

## Использование машинного зрения

После того как робот подъедет к объекту, ориентируясь по GPS, выполняется алгоритм поиска QR-кода. Для моей системы я использую QR-код 30x30см, распечатанный на плотном листе, ламинированный, и размещённый непосредственно на объекте, например, на стволе дерева.

Я использую программный инструмент OpenCV для распознавания объектов. Инструмент OpenCV подключается как библиотека в основной программе на C++ и позволяет использовать различные функции.



**Рисунок 15 - Алгоритм работы машинного зрения**

Для примера показан фрагмент кода на языке C++ с использованием функций OpenCV. Функция `qcd.detect` получает на вход изображение, и в случае успешной обработки возвращает список `points`, который хранит в себе информацию обо всех распознанных на изображении QR-кодах. Полученные данные я обрабатываю и отправляю по специальному топику в основную программу робота.

```
qcd.detect(frame, points);  
    if (points.size() == 4) {  
        polylines(frame, points, true, color, 5);  
        qrcode qr_pos;  
        qr_pos.x = (points[0].x + points[1].x + points[2].x + points[3].x) / width_to_0_1;  
        qr_pos.y = (points[0].y + points[1].y + points[2].y + points[3].y) / heigh_to_0_1;  
        qr_pos.dist = get_dist_to_qr(points[0].x, points[0].y, points[1].x, points[1].y);  
        chatter_pub.publish(qr_pos);  
    }
```

### Алгоритм управления движением робота

Для управления драйверами моторов используется микроконтроллер ESP32, в качестве драйверов используется драйвера робототехнических наборов TETRIX.

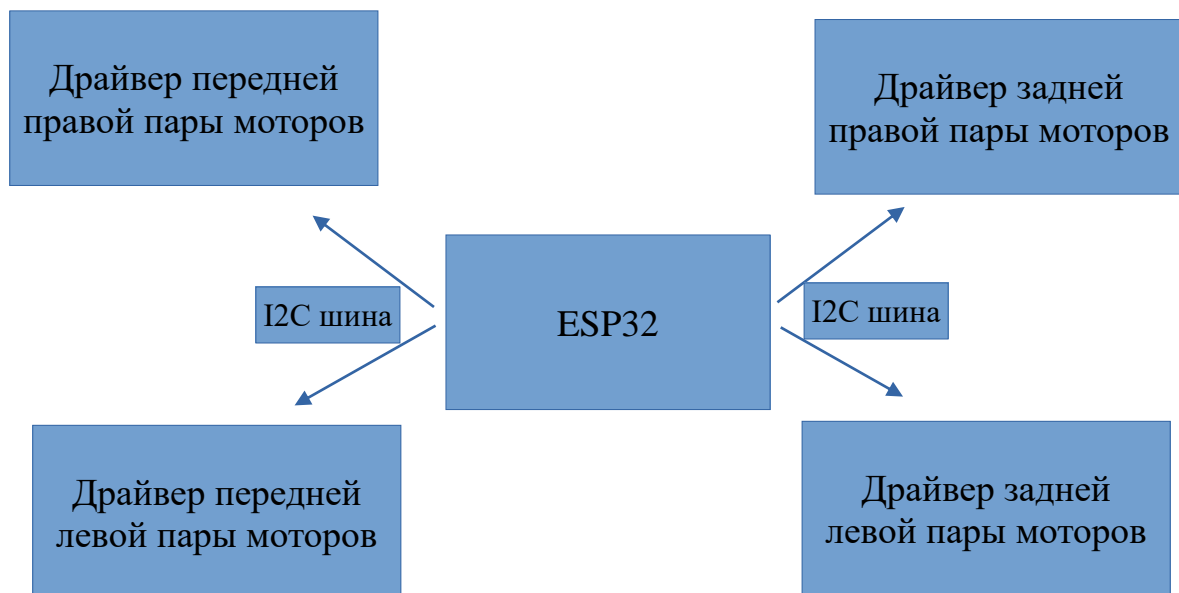


Рисунок 16 - Схема управления движением робота

Ниже в таблице приведены характеристики микроконтроллера ESP32 и драйверов TETRIX, причины их выбора для проекта.

Микроконтроллер ESP32	Преимущества: недорогой, доступный, есть Bluetooth, производительность и другие характеристики гораздо лучше по сравнению с Arduino UNO.	Минусы: возможно чуть меньшая распространённость чем платформа Arduino и, возможно, чуть большая сложность программирования в некоторых случаях.
Драйвера TETRIX	Преимущества: наличие в школе в наборах робототехники, полная документированная функциональность по командам, удобство подключения моторов TETRIX с энкодерами, работа по I2C шине с микроконтроллером.	Минусы: дороговизна в случае использования как рабочего варианта (если не брать из наличия в школе), относительно габаритный корпус.

**Таблица 11- Выбор микроконтроллера и драйвера двигателей**

#### **Примеры команд для взаимодействия энкодеров моторов**

- 0x24 - установка id мотора.
- 0x25 – включение драйвера.
- 0x45 – Установка скорости моторов.

## Пример функции для ESP32 взаимодействия с драйвером моторов по I2C шине

```
void sendCommandFourByte(  
byte id,  
int command,  
byte byte1, byte byte2, byte byte3, byte byte4  
) {  
Wire.beginTransmission(id);  
Wire.write(commnad);  
Wire.write(byte1);  
Wire.write(byte2);  
Wire.write(byte3);  
Wire.write(byte4);  
Wire.endTransmission();  
}
```

## ROS и FreeRTOS

В своём проекте я так же использую ROS и FreeRTOS, так как это позволяет намного удобнее и быстрее разрабатывать программную часть.

Используя ROS я смог разработать архитектуру, где данные параллельно и независимо друг от друга передаются между микроконтроллером ESP32 и мини-компьютером Intel NUC.

Топики	Данные
gps_data	Обработанные значения с GPS трекера
compass_data	Обработанные данные с компаса
recieve_by_bluetooth	Данные полученные по блютузу
send_by_bluetooth	Данные для отправки по блютузу
motors_control	Данные для управлением моторами
qr_code_pos	Данные о местоположении QR-кода
sprayer_control	Данные для управления распылителем

**Таблица 12 - Топики ROS и передаваемые данные**

Во время написания программы на ESP32 с использованием FreeRTOS я столкнулся с множеством проблем, такие как переполнение выделенного на задачу стека, срабатывание whatchdog таймера и др. Решив все эти проблемы получилась отличная архитектура, которая работает стабильно и параллельно.

Задачи	Пояснение
motors	Задача управления моторами
compass	Задача получения и обработки данных компаса
gps	Задача получения и обработки данных GPS
bluetooth	Задача получения, обработки и отправки данных bluetooth
sprayer	Задача управления распылителем

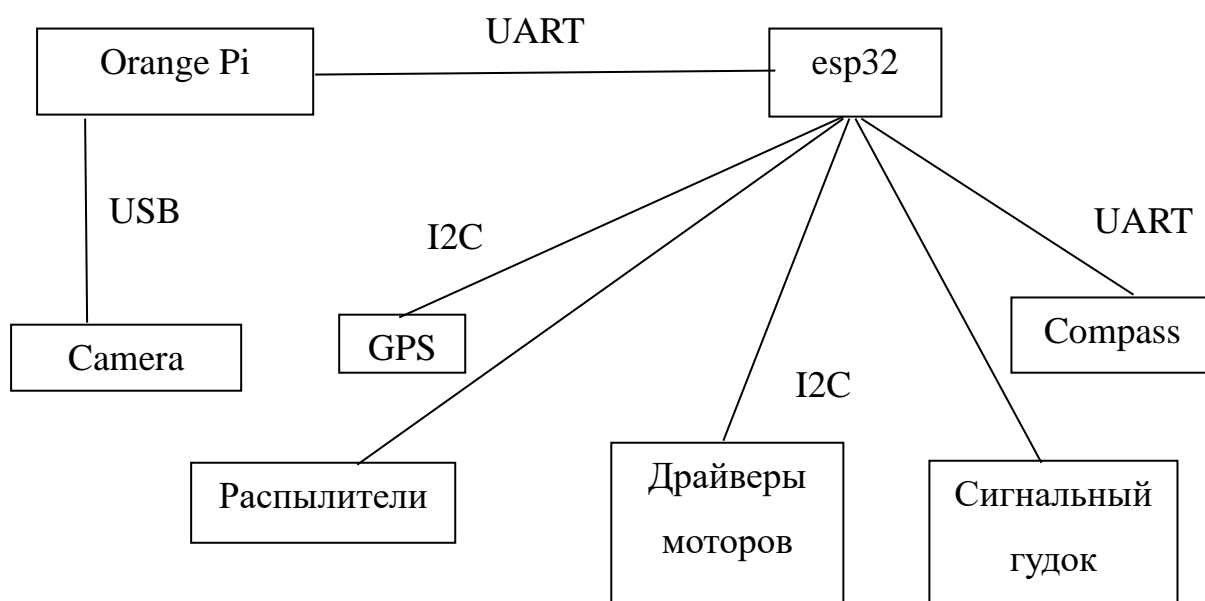
**Таблица 13 - Задачи FreeRTOS**

### **Структурная схема работа**

Выбранный элемент	Описание
Микрокомпьютер OrangePi 5 (64-bit, 2,4 GHz, 8 GB RAM)	Достаточная производительность, относительно невысокая цена
Мотор-редуктор TETRIX TorqueNADO	Питание 12V, передаточное отношение 60:1
Компас	Питание 5V, габариты 40мм x 40мм
GPS	Питание 5V, габариты 15мм x 15мм

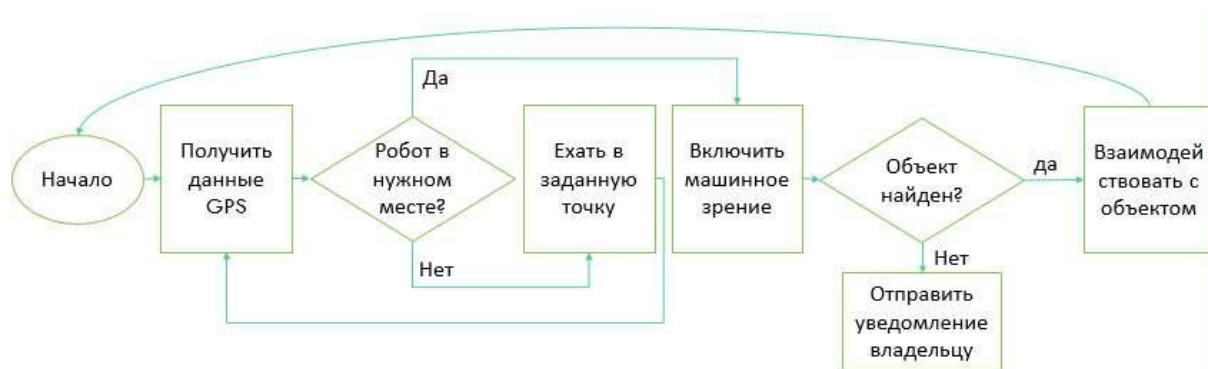
**Рисунок 17 – Используемые компоненты**

Структурная схема выглядит следующим образом (полная структурная схема в приложении) :



**Рисунок 18 – Структурная схема**

### Блок-схема алгоритма



**Рисунок 19 - Общий алгоритм работы**



## Используемые языки программирования и процесс отладки кода

В проекте использовались языки программирования C/C++ и Kotlin.



	Используется для всех алгоритмов, работающих на работе, на микроконтроллере esp32 и микрокомпьютере Orange PI.
	Используется для создания собственного пользовательского мобильного приложения для управления роботом со смартфона.

Таблица 14 – Используемые языки программирования

Для отладки использовались средства Visual Studio Code.

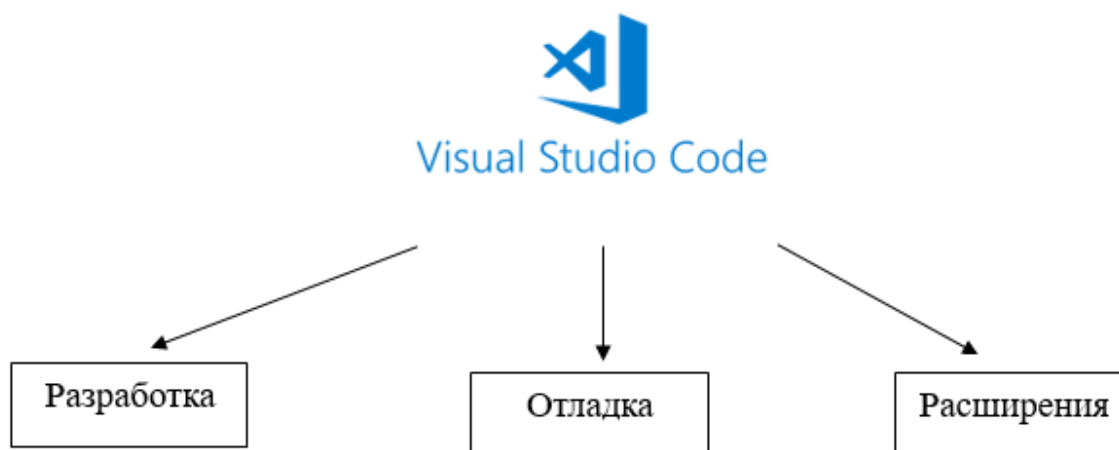


Рисунок 20 – Средство отладки и его возможности

Синтаксические ошибки достаточно легко выявляются с помощью функционала IDE, алгоритмические ошибки было выявлять гораздо сложнее.

Могу выделить следующие трудности, с которыми пришлось разбираться больше всего:

1. Получение данных компаса.
2. Распознавание объектов с помощью машинного зрения.
3. Использование команд для управления драйверами TETRIX.

## Собственные библиотеки

Созданы собственные библиотеки для оптимизации программного кода:

- Библиотека для управления драйверами TETRIX
- Библиотека для получения данных с GPS
- Библиотека для получения данных с компаса
- Библиотека для отправления / получения данных через bluetooth

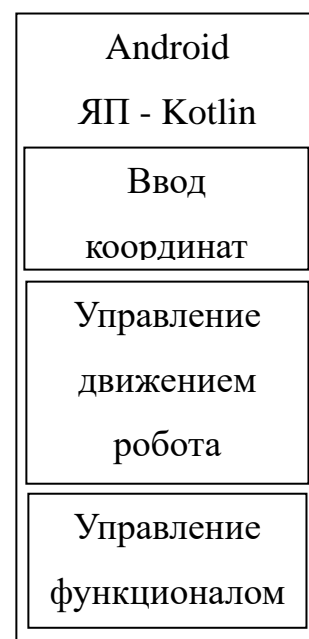
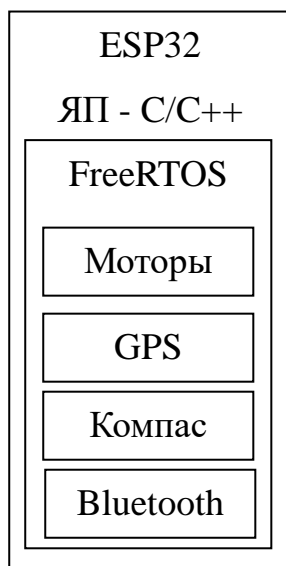
## Реализация регуляторов и конечного автомата

Регуляторы реализованы для:

- передвижения по координатам GPS
- Ориентации и движения на Aruco метку

Состояния конечного автомата:

- Управление с телефона
- Ориентация по GPS
- Ориентация по Aruco метке
- Распыление ядохимикатов



## Электроника, проектирование в САПР EDA KiCad

Для проекта выбран САПР EDA KiCad, так как он бесплатен и имеет все необходимые функции. Я разработал электронную плату для более оптимального размещения компонентов электроники. На данный момент на плате размещаются:

- Модуль esp32
- Разъёмы pbs
- Модули реле
- Винтовые зажимы
- Модуль GPS
- Магнитный датчик

В дальнейшем я планирую интегрировать на плату большее количество элементов. На рисунке ниже приведена принципиальная схема.

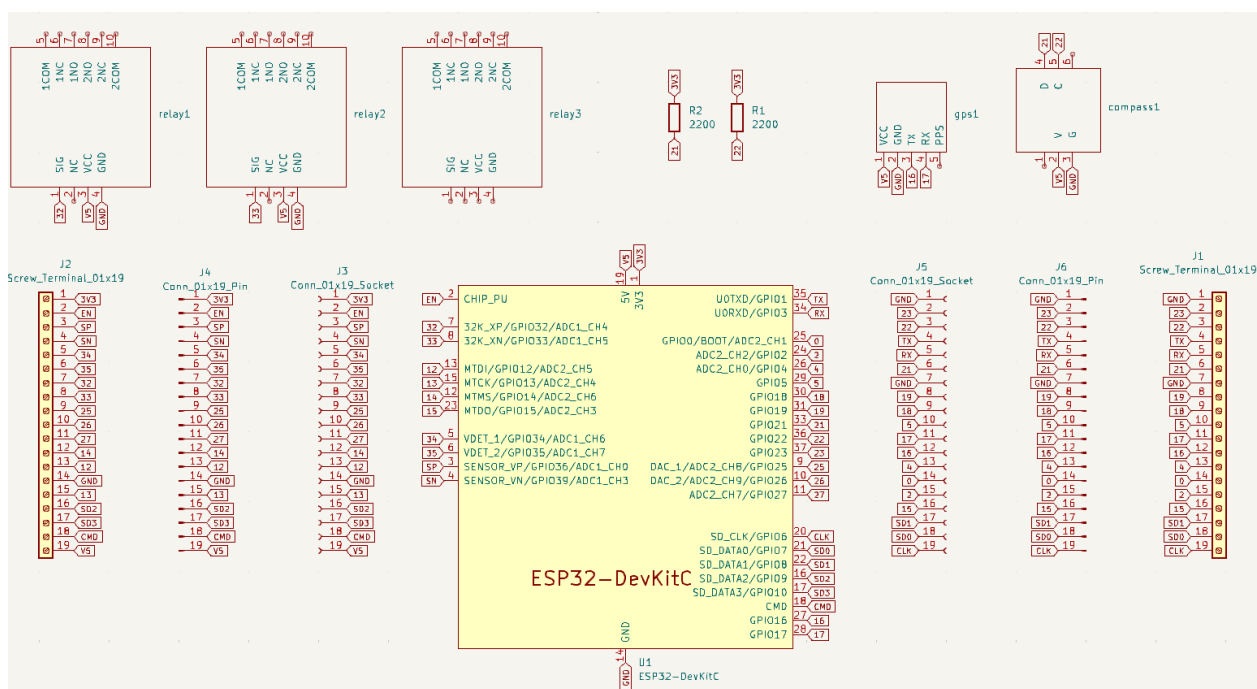
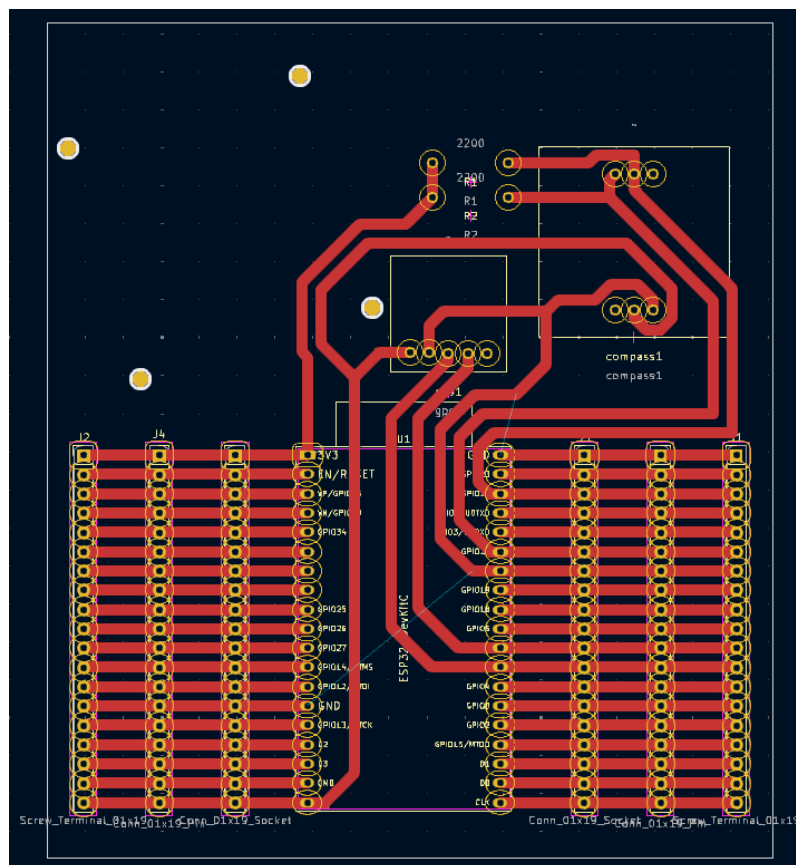


Рисунок 21 – Принципиальная схема

Примечание. На принципиальной схеме соединения элементов показаны точками подключения, а не линиями

На рисунке показана трассировка платы в программе KiCad.



**Рисунок 22 – Трассировка платы**

Плата изготавливалась фрезерованием на станке с ЧПУ.



**Рисунок 23 – Используемый станок для фрезерования платы “Roland SRM-20”**

## Технологическая карта

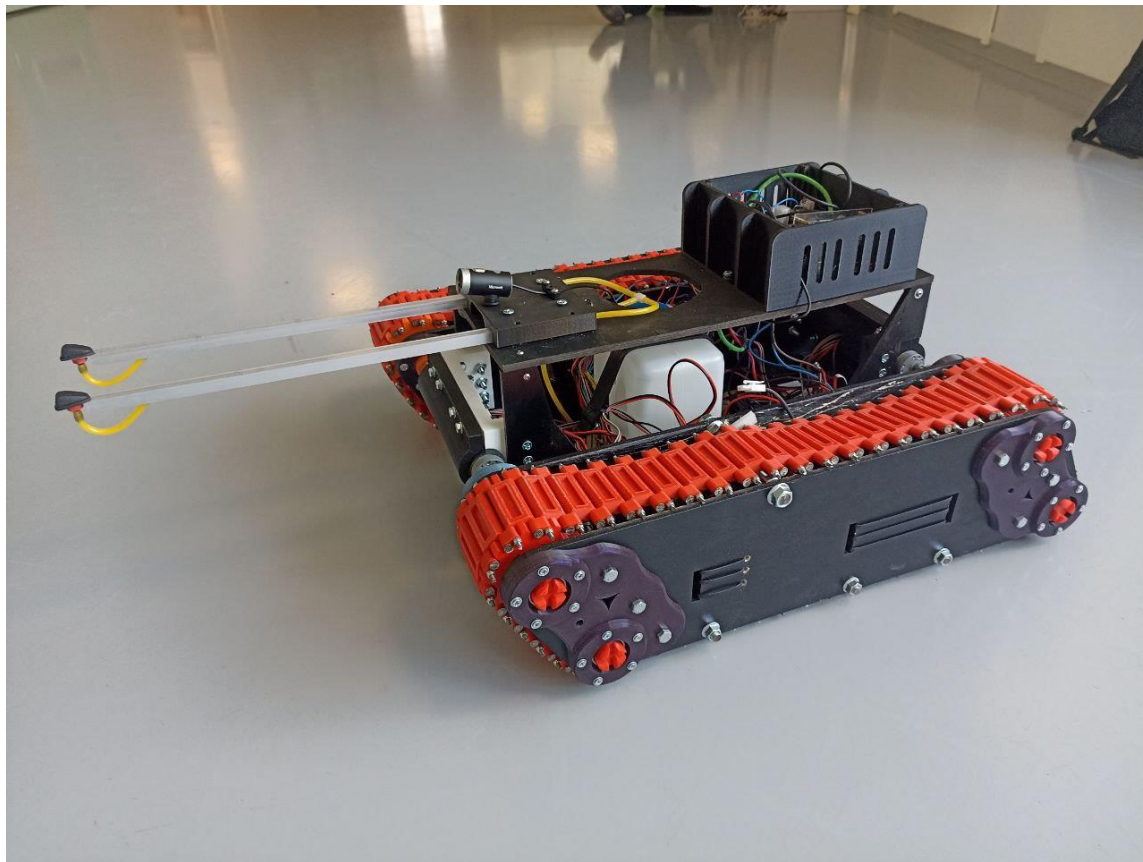
Ниже, в технологической карте указаны используемое оборудование, программное обеспечение, выполняемые технологические операции, затраченное время.

Описание технологических операций	Используемое оборудование	Время (часы)
Моделирование в САПР-программах	Solid Works, Inventor	> 25
Печать деталей	3D принтер	> 300
Сборка каркаса	Ручной инструмент, электроинструмент	20
Настройка плат управления	Компьютер, программное обеспечение	14
Разработка кода навигации	Компьютер, программное обеспечение	25
Работа с машинным зрением	Компьютер, программное обеспечение	20
Испытания	Собранная конструкция робота, компьютер, среда разработки	4
ИТОГО	-	> 408

Таблица 15 - Технологическая карта

## Выводы

На фотографии ниже показан вид готового робота на данный момент



**Рисунок 24 – Вид готового робота**

Конструкция надёжно собрана, нет явных недостатков (для прототипа).  
На роботе установлены: силовая электрическая часть, система подачи растворов химикатов.

В системе управления реализовано:

- ✓ Использован мини ПК и ОС Linux.
- ✓ Использован ROS1.
- ✓ Использован FreeRTOS.
- ✓ Отлажен алгоритм и написан программный код для передвижения по GPS координатам и компасу.
- ✓ Использован алгоритм машинного зрения для работы с Aruco метками.

## **Креативность и новизна продукта**

При разработке робота я ориентировался на собственные идеи, хотя вполне возможно, что некоторые решения могут быть реализованы в каких-либо других существующих проектах. Мои идеи представлены на рисунке ниже.

Разработана система ЗОЯ (Залповое Опрыскивание Ядохимикатами), включающая бак на 1.5 литра, веерное распыление, дальность до 1.5 метров.

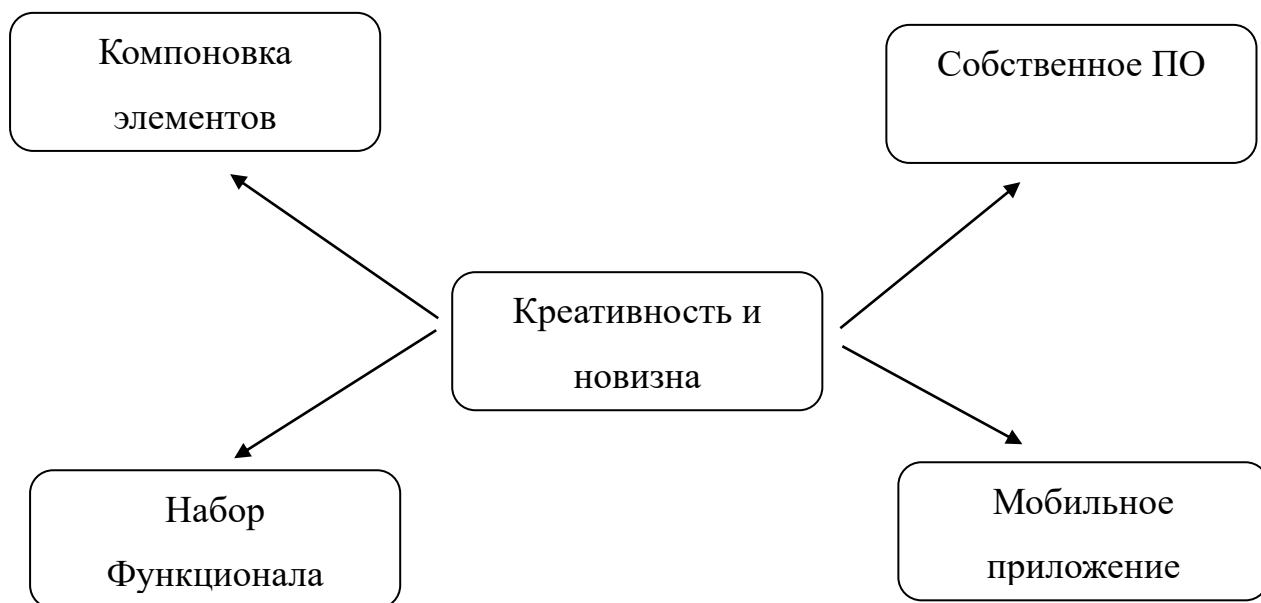


**Рисунок 25 - Элементы системы ЗОЯ**

Установлен комплект отпугивающей звуковой атаки (КОЗА), включающий реле включения, турбонасос, акустический пневмоизлучатель.



**Рисунок 26 – Элементы КОЗА.**



**Рисунок 27 – Креативность и новизна**

### **Задачи для следующего этапа**

В процессе выполнения первого этапа стали очевидными задачи, которые нужно выполнить в дальнейшем, ниже они перечислены.

- Дальнейшая оптимизация системы навигации.
- Дальнейшая оптимизация системы машинного зрения.
- Выполнение корпуса робота из конструкционных материалов (алюминиевые сплавы и др.).
- Герметизация корпуса для защиты от влаги и пыли.
- Доработка шасси с целью повышения клиренса.
- Добавление функционала автоподзарядка.
- Добавление функционала автозаполнения ёмкостей.



## Экологическая оценка

При производстве модели были использованы доступные материалы, которые имеют сертификаты качества. Не было использовано ядовитых, радиоактивных и других подобных веществ, требующих специальных экологических мероприятий. Технологические процессы при производстве также проводились с соблюдением экологических норм, без негативного влияния на окружающую среду. Электронные компоненты и аккумуляторы после завершения срока годности подлежат утилизации в соответствии с современными требованиями.

Предполагается, что автоматизация процессов удобрения и ядохимикатами позволит снизить негативный эффект на почву дачного участка за счёт точной дозировки распыления на заданную площадь.

## Экономическая оценка проекта

Робот создавался с учётом максимального использования имеющихся в школе комплектующих, материалов и оборудования.

Моторы, драйвера моторов	Из школьного набора робототехники TETRIX
3D принтер, материалы для печати	Школьное оборудование и материалы
Микрокомпьютер OrangePi и микроконтроллер esp32	Собственное оборудование
Крепёжные элементы и другие расходные материалы	Собственное приобретение

В результате собственные затраты составили около 10.000 рублей.

В дальнейшем при возможной коммерциализации проекта оцениваю продажную стоимость робота в районе 500.000 рублей.

## Приложения

В приложении приведены чертежи деталей (выборочно), сборочный чертёж, спецификация, листинг кода.

## Приложение: Листинг кода

Данный код (C++) устанавливает скорость моторов в зависимости от полученных сообщения из ROS. Функция *motors\_process* является коллбэком, и вызывается при получении нового сообщения из ROS. Функция *go* определяет направление вращения моторов для каждого драйвера. Функция *set\_motor\_speeds* разделяет скорость каждого мотора на два байта и отправляет её по I2C на драйвер.

```
void motors_process(const slon::motors& speed) {
    go(speed.l_speed, speed.r_speed);
}

void go(int lSpeed, int rSpeed) {
    set_motor_speeds(1, -lSpeed, -lSpeed);
    set_motor_speeds(2, rSpeed, rSpeed);
    set_motor_speeds(3, rSpeed, rSpeed);
    set_motor_speeds(4, -lSpeed, -lSpeed);
}

void set_motor_speeds(int address, int mSpeed1, int mSpeed2) {
    int lbyte1 = lowByte(mSpeed1);
    int hbyte1 = highByte(mSpeed1);
    int lbyte2 = lowByte(mSpeed2);
    int hbyte2 = highByte(mSpeed2);
    send_command_four_byte(address, 0x45, hbyte1, lbyte1,
hibyte2, lbyte2);
}
```

Данный код (C++) запускает задачи FreeRTOS на esp32, а также инициализирует ROS ноду. Метод *nh.initNode* инициализирует ноду esp32. Функция *xTaskCreate* создаёт новую задачу и сразу запускает её. 1-ый аргумент – функция, выполняемая этой задачей, 3-ий – размер стека, выделенного на задачу, 5-ый – приоритет задачи.

```
#include "motors.h"
#include "compass.h"
#include "gps.h"
#include "bluetooth.h"
#include "sprayer.h"
#include "AIR_HORN.h"

ros::NodeHandle nh;

void setup() {
    nh.getHardware()->setBaud(115200);
    nh.initNode();
    xTaskCreate(main_motors, "motors", 2048, NULL, 1, NULL);
    xTaskCreate(main_compass, "compass", 2048, NULL, 1, NULL);
    xTaskCreate(main_gps, "gps", 2048, NULL, 1, NULL);
    xTaskCreate(main_bluetooth, "bluetooth", 4096, NULL, 1, NULL);
    xTaskCreate(main_sprayer, "sprayer", 2048, NULL, 1, NULL);
    xTaskCreate(main_horn, "horn", 2048, NULL, 1, NULL);
}

void loop() {
    vTaskDelay(1000000);
}
```

Данный код (Kotlin) используется в Android приложении для получения списка bluetooth устройств. Сначала создаётся *BluetoothManager*, он даёт доступ к api bluetooth. Далее создаётся *BluetoothAdapter*, он является праграммным представлением bluetooth. После проверки состояния bluetooth идёт получение устройств, которые ранее были подключены к смартфону.

```
val bluetoothManager: BluetoothManager =
    getSystemService(BluetoothManager::class.java)
val bluetoothAdapter: BluetoothAdapter? = bluetoothManager.adapter
if (bluetoothAdapter == null) {
    this.finish()
}
if (bluetoothAdapter?.isEnabled == false) {
    val enableBtIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
    val startForResult = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
    ) { result: ActivityResult ->
        if (result.resultCode == Activity.RESULT_OK) { this.finish() }
        recreate()
    }
    startForResult.launch(enableBtIntent)
}
val pairedDevices: Set<BluetoothDevice>? =
    bluetoothAdapter?.bondedDevices
pairedDevices?.forEach { device ->
    val deviceName = device.name
    val deviceHardwareAddress = device.address
    Log.d("LOG_TAG", "Name: $deviceName | Addr: $deviceHardwareAddress")
}
if (pairedDevices?.isEmpty() == true) { this.finish() }
var devices: List<BluetoothDevice> = pairedDevices!!.toList()
```

## Приложение: Используемое ПО

Название ПО	Тип ПО	Причина выбора
SolidWorks	САПР	Многообразие функционала. Уже был опыт работы.
KiCAD	САПР	Наличие необходимого функционала. Распространённость. Обширная документация.
Visual Studio Code	Редактор кода	Плагины для программирования esp32 и поддержки языка программирования C++.
Cura	ПО для генерации G-кода	Наличие необходимого функционала. Уже был опыт работы.
Android Studio	ПО для создания приложений на ОС Android	Официальное ПО для разработки приложений на Android от Google. Поддержка языка программирования Kotlin.
OpenBuilds CAM	ПО для генерации G-кода	Простота использования.

Приложение: Принципиальная и структурная схемы

На рисунках ниже приведены принципиальная и структурная схемы робота. На принципиальной схеме соединения показаны точками присоединения.

