|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_Робототехника и комплексная автоматизация (РК)\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_Теория механизмов и машин (РК2)\_\_\_\_\_\_\_\_\_\_\_

**Пояснительная записка**

**к курсовому проекту**

**по «Теории механизмов и машин»**

**на тему**

**«****Проектирование макета робота для участия в открытых соревнованиях Bauman Robotics Club 2023»**

Выполнили:

Студент   СМ7-54Б  Иванов И.С.

(Группа) (Подпись, дата) (Фамилия И.О)

Студент   СМ7-54Б  Колесников В.Е.

(Группа) (Подпись, дата) (Фамилия И.О)

Студент   СМ7-54Б  Гаденов Т.Е.

(Группа) (Подпись, дата) (Фамилия И.О.)

Студент   СМ7-54Б  Фяткуллов Д.Н.

(Группа) (Подпись, дата) (Фамилия И.О)

Студент   РК9-52Б  Давыдова К.А.

(Группа) (Подпись, дата) (Фамилия И.О.)

Руководитель проекта Сащенко Д.В.

(Подпись, дата) (Фамилия И.О.)

2023

Оглавление

[1. Введение 5](#_Toc154348664)

[2. Постановка задачи. 8](#_Toc154348665)

[3. Анализ возможных компонентов робота 9](#_Toc154348666)

[3.1 Анализ возможных видов перемещения 9](#_Toc154348667)

[3.2 Анализ электронных компонентов 16](#_Toc154348668)

[3.2.1 Микрокомпьютер высокого уровня. 17](#_Toc154348669)

[3.2.2 Микроконтроллер нижнего уровня. 17](#_Toc154348670)

[3.2.3 Двигатели с редуктором и энкодером 18](#_Toc154348671)

[3.2.4 Драйверы моторов 18](#_Toc154348672)

[3.2.5 Камера 19](#_Toc154348673)

[3.2.6 Датчик линии VEX Line Tracker 19](#_Toc154348674)

[3.2.7 Датчик расстояния 20](#_Toc154348675)

[3.2.8 Сервомотор 21](#_Toc154348676)

[3.2.9 Питающие элементы 22](#_Toc154348677)

[4. Конструирование подвижной платформы 25](#_Toc154348678)

[4.1 Упрощенный расчет кинематики 25](#_Toc154348679)

[4.2 Полный расчет кинематики 27](#_Toc154348680)

[4.3 Направление скоростей на колесах робота при различных видах движения: 31](#_Toc154348681)

[4.4 Расчет мощности двигателей 32](#_Toc154348682)

[5. Расчет схвата для захвата тел вращения 34](#_Toc154348683)

[5.1 Построение профиля поворотных губок (рабочего элемента схвата) 34](#_Toc154348684)

[5.2 Расчет механизма захватывающего устройства 36](#_Toc154348685)

[6. Реализация управления нижнего уровня робота 39](#_Toc154348686)

[6.1 Измерение угловой скорости моторов 39](#_Toc154348687)

[6.2 Регуляция угловой скорости моторов 40](#_Toc154348688)

[7. Система технического зрения для распознавания цвета и положения бочек 42](#_Toc154348689)

[7.1 Выбор программного обеспечения 42](#_Toc154348690)

[7.2 Реализация распознавания цветов и объектов 43](#_Toc154348691)

[8. Реализация езды по линии 51](#_Toc154348692)

[9. Использование RP Lidar A1 57](#_Toc154348693)

[10. Алгоритм действий робота для финального этапа соревнований 59](#_Toc154348694)

[11. Сборка и отладка робота 61](#_Toc154348695)

[Выводы 63](#_Toc154348696)

[Список литературы 64](#_Toc154348697)

[Приложение 1](#_Toc154348698)

[Приложение 2](#_Toc154348699)

[Приложение 3](#_Toc154348700)

[Приложение 4](#_Toc154348701)

[Приложение 5](#_Toc154348702)

[Приложение 6](#_Toc154348703)

[Приложение 7](#_Toc154348704)

**Аннотация**

Расчетно-пояснительная записка к курсовой работе «Проектирование макета робота для участия в открытых соревнованиях Bauman Robotics Club 2023» содержит 64 страницы машинописного текста А4, 29 рисунков, 2 таблицы и 6 приложений, 2 листа А3 и 1 лист A4 графического материала. В дополнение к расчетно-пояснительной записке составлена презентация, состоящая из 26 слайдов.

В работе рассмотрен и обоснован выбор вида перемещения макета робота, необходимой электроники для создания робота, была решена задача кинематики. Так же приведен расчет схвата для тел вращения, алгоритм для езды макета по линии, была реализована система технического зрения, так же реализовано избегание препятствий с помощью лидара и разработан алгоритм для финального этапа соревнований.

Ключевые слова: омни - колесо, мобильный робот, система технического зрения, лидар, навигация, кинематика, ПИД регуляция, датчики, робототехника, ROS, управление, механика.

# Введение

В данном проекте мы сконструируем и запрограммируем учебного робота для участия в соревнованиях Bauman Robotics Club 2023, а также принимаем участие в их организации этого мероприятия. Состязания роботов уже проводились в прошлые годы и не раз увенчались успехом, что привело к популярности данного мероприятия среди студентов.

Соревнования роботов "Bauman Robotics Club" – это ежегодное мероприятие, организованное клубом робототехники Бауманского технического университета.

Отличительные черты соревнований включают использование различных типов роботов, таких как автономные и телеуправляемые, с разными назначениями: от решения лабиринтов и соревнования на скорость до выполнения разнообразных технических задач (таких как перемещение различных предметов). В соревнованиях участвуют команды студентов и школьников, что способствует развитию технического мышления и командной работы.

Соревнования "Bauman Robotics Club" проводятся ежегодно весной (финальный этап) и осенью (отборочный этап). Варианты проведения могут включать как командные соревнования, так и индивидуальные испытания роботов. Мероприятие также включает в себя показательные выступления, лекции и демонстрации новых разработок в области робототехники.

В мире существует множество соревнований роботов, позволяющих инженерам и энтузиастами продемонстрировать свои технические навыки и креативность. Некоторые из них стали реальными явлениями, привлекая внимание миллионов людей по всему миру.

Соревнования роботов "EuroBot" известны своим фокусом на создание автономных роботов, способных выполнять различные задачи в соревновательной среде. Это ежегодное мероприятие привлекает команды из разных стран, которые соревнуются в различных категориях, таких как строительство, манипуляция, доставка и т.д.

Одной из особенностей "EuroBot" является то, что роботы должны быть полностью автономными, то есть не могут быть управляемыми или контролируемыми с помощью человека во время выполнения задач. Роботы должны основываться на предварительно программированных инструкциях и использовать различные сенсоры для управления и взаимодействия с окружающей средой.

Соревнования "EuroBot" проводятся ежегодно в различных странах, обычно весной или летом. Они привлекают не только профессиональных робототехников и инженеров, но и студентов и школьников, которые имеют возможность продемонстрировать свои навыки в создании и программировании роботов.

Одним из самых известных соревнований является FIRST Robotics Competition (FRC), основанный Дином Каменом в 1989 году. Этот международный турнир проводится ежегодно с января по апрель и позволяет старшеклассникам со всего мира конструировать, программировать и соревноваться с полноразмерными роботами. Участники FRC работают в командах и испытывают свои навыки в борьбе за выполнение назначенных заданий в реальном времени, используя различные системы управления и датчики. Этот конкурс известен своей высокой конкурентоспособностью и инновационными решениями, которые приносят команды.

Еще одной знаменитой битвой роботов является BattleBots, первоначально транслировавшийся в середине 1990-х на американском телевидении. Сегодня BattleBots - это популярное шоу и арена для инженеров-любителей, где они создают и управляют роботами, сражаясь друг с другом в грандиозных боях. Участники BattleBots соревнуются с целью уничтожения или выведения соперника из строя, используя различные виды оружия и тактики. Это соревнование привлекает множество зрителей и становится известным благодаря своей драматичности и захватывающим боям.

В дополнение к этим знаменитым соревнованиям также существуют множество других событий, в которых робототехника занимает центральное место. Некоторые из них включают DARPA Robotics Challenge, RoboCup и VEX Robotics Competition. Эти мероприятия предлагают различные задачи, от автономных миссий и спасательных операций до футбольных матчей с участием роботов, и предоставляют возможность проявить свои навыки и техническое мастерство.

Люди стремятся участвовать в этих соревнованиях по разным причинам. Некоторые видят в них возможность познакомиться с новыми технологиями и разработками, а также продемонстрировать свою способность к инновационному мышлению. Длительное участие в таких событиях может привести к улучшению навыков, созданию ценных связей и даже карьерным возможностям в робототехнике. Кроме того, некоторые люди испытывают азарт и удовольствие от участия в соревнованиях роботов и стремятся победить в них, чтобы показать свое мастерство и инженерную экспертизу.

Все эти соревнования способствуют развитию робототехники, привлекают миллионы зрителей и вдохновляют новое поколение инженеров и ученых со всего мира. Они являются отличной платформой для того, чтобы демонстрировать новейшие технологии и достижения в области робототехники, а также для того, чтобы повысить осведомленность о важности и потенциале этой области. Подобные соревнования развивают навыки разработки роботов, программирования, решения нестандартных задач и креативного мышления. Такие мероприятия–отличная возможность для инженеров продемонстрировать свои умения, помимо этого являются хорошим стимулом для развития и совершенствования технологий.

Таким образом проводимые соревнования вызывают интерес к робототехнике у студентов, помогают развитию новых технологий и инженерии, а также показывают уровень знаний среди людей по всему миру.

# Постановка задачи.

**Цель** данного курсового проекта заключается в проектировании, сборке и тестировании макета робота для участия в робототехнических соревнованиях (МРС) Bauman Robotics Club 2023.

**Задачи проекта:**

1. организовать и провести соревнования по робототехнике BRC-2023;
2. разработать структуру системы управления МРС;
3. выбрать компоненты для МРС;
4. разработать конструкцию подвижной платформы МРС;
5. разработать захватное устройство МРС;
6. разработать систему технического зрения для МРС;
7. разработать двухуровневую систему управления МРС;
8. произвести сборку МРС;
9. провести тестирование и отладку МРС в ходе соревнований

BRC-2023;

1. повысить уровень знаний учащихся в области робототехники.

**Задачи робота на отборочном этапе**

1. Основная задача: МРС необходимо преодолеть трассу по заданной траектории движения, обозначенной направляющими элементами (линиями, пунктиром, и т.д.) (см. Приложение 1).
2. Поле разделено на 4 области, обозначающие сложность преодолеваемого участка.
3. В первой области находятся: место старта; прямой и криволинейный участок траектории, обозначенный сплошной линией черного цвета.
4. Во второй зоне находятся: прямой и криволинейный участок траектории, обозначенный прерывистой линией черного цвета, а также кругами различного диаметра.
5. Третья зона черного цвета с траекторией, обозначенной линией белого цвета (инверсия).
6. В четвёртой зоне находятся: прямолинейный участок без направляющей линии; ломаный участок траектории с резкими поворотами и перегибами, обозначенный сплошной линией черного цвета.
7. Для завершения маршрута, роботу необходимо всеми проекциями оказаться в зоне финиша

**Задачи робота на заключительном этапе**

1. Основная задача: - цветные цилиндры (бочки) диаметром 50 мм и высотой 100 мм (см. Приложение 2) перенести в специально обозначенные области.
2. Поле разделено на зону старта, зону сбора бочек и зону выгрузки:

Зона сбора бочек разделена на две области: в первой области расположение бочек известно, во второй бочки расположены случайным образом

2.1 Во второй области дополнительно с цветными находятся бочки черного цвета. Эти бочки запрещено вывозить из данной области.

2.2 Зона выгрузки состоит из трёх областей, обозначенных красным, зелёным и синим цветом, в которые необходимо доставить бочки соответствующего цвета. Порядок расположения областей определяется случайным образом.

3. Для завершения попытки, роботу необходимо вернуться в зону старта и всеми проекциями оказаться в этой зоне.

# Анализ возможных компонентов робота

3.1 Анализ возможных видов перемещения

В современном мире есть довольно много подвижных платформ. Для более рационального выбора способа перемещения робота рассмотрим наиболее распространенные из них. Одними из самых часто встречающихся способов перемещения подвижных платформ являются гусеничные и колесные.

**Шагающие роботы**

Преимущества:

* маленький радиус поворота (радиус равен ширине подвижной платформы робота);

Недостатки:

* низкая скорость;
* в большинстве своем не равномерное движение;
* высокая стоимость конструкций.

**Ползающие роботы**

Преимущества:

* возможность перемещаться в узких пространствах.

Недостатки:

* высокое трение соприкасающихся поверхностей;
* высокая стоимость.

**Гусеничные роботы**

Преимущества:

* маленький радиус поворота (радиус равен ширине подвижной платформы робота);
* проходимость;

Недостатки:

* высокая стоимость;
* низкая транспортная скорость.

**Колесные роботы**

Здесь присутствуют двух, трех, четырехколесные и даже n-колёсные платформы, а также платформы на omni-колесах, колёсах Лиддиарда и колёсах Илона.

**Двухколесные роботы:**

Преимущества двухколесных платформ:

* маленький радиус поворота (радиус равен ширине подвижной платформы робота)

Недостатки:

* высокая стоимость;
* сложности балансировки.

**Трехколесные роботы:**

Трехколесные роботы бывают двух типов в зависимости от характера управляемых колес. Первый тип поставляется с колесами, которые приводятся в действие отдельно, а третий свободно вращается исключительно для баланса. Второй тип поставляется с двумя колесами, которые питаются от одного источника и другого источника для третьего колеса.

**Трёхколёсные роботы:**

Преимущества:

* оптимальное соотношение цена/возможности;
* низкая стоимость.

Недостатки:

* сложность изготовления.

**Четырехколесные и n-колёсные роботы:**

Преимущества:

* высокая эффективность из-за маленького скольжения;

Недостатки:

* сложность изготовления.

**Omni-колёсные роботы:**

****

Рис. 1 Изображение omni-колес

Некоторые колесные роботы на omni-колесах (Рис. 1) используют треугольную платформу с тремя колесами, расположенными под углом 60 градусов, Преимущества использования трех колес, а не четырех, заключаются в том, что они выходят дешевле, три точки можно использовать в одной плоскости, и только одно колесо будет вращаться в направлении движения [2].

Преимущества:

* дешевизна изготовления;
* возможность расположить omni-колеса, спрятав их внутри робота;
* возможность разворота на месте
* всенаправленное движение.

Недостатки:

* низкая эффективность, т.к. происходят потери от трения из-за того, что не все колеса крутятся в направлении движения [7].

 **Колесо Лиддиарда:**

Рис. 2 Изображение колеса Лиддиарда

Британец Вильям Лиддиард создал «всенаправленные» автомобильные колеса, позволяющие автомобилю двигаться абсолютно в любых направлениях (Рис. 2).

Преимущества:

* маленький радиус поворота;
* возможность работы в труднопроходимых местах;

Недостатки:

* высокая стоимость;
* невозможность производства.

**Колеса Илона**

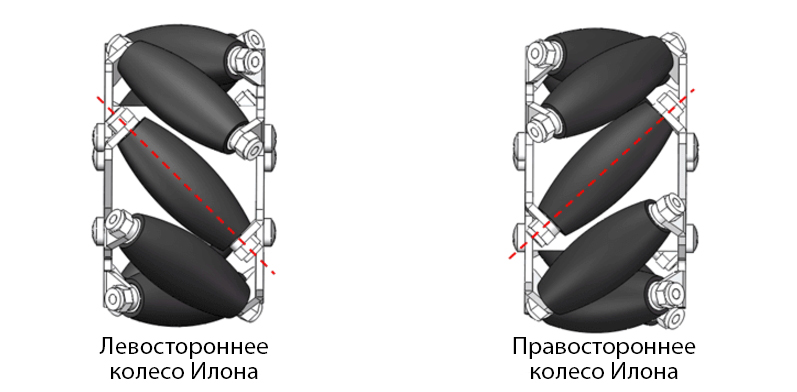


Рис. 3 Изображение колес Илона

Колесо Илона или Шведское колесо – это роликонесущее колесо которое позволяет двигаться транспорту, оснащенному четырьмя такими колесами (Рис. 3) в любом направлении.

Преимущества:

* дешевизна изготовления;
* возможность расположить их внутри робота;
* всенаправленное движение.

Недостатки:

* низкая эффективность, т.к. происходят потери от трения из-за того, что не все колеса крутятся в направлении движения.

Сравнительные характеристики всенаправленных колес представлены в таблице 1.

Таблица 1. Анализ способов перемещения роботов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Характеристики | Гусеничные  Подвижные платформы |  | Колесные платформы | | | Шагающие платформы | Ползающие платформы |
| Колеса | Omni-колеса | Колеса Лиддиарда | Колеса Илона |
| Стоимость | **-** | + | - | - | - | - | - |
| Низкий коэффициент скольжения  (трения) | + | + | - | + | - | + | - |
| Маленький радиус поворота | + | - | + | + | + | + | - |
| Скорость | + | + | - | - | + | + | - |

***Вывод:***

Проанализировав в таблице 1 способы перемещения разных видов роботов, мы приняли решение использовать 3 omni-колеса, так как они упрощают расположение компонентов на макете робота, удовлетворяют нашему бюджету. Исходя из регламента и задачи соревнований, лучше всего подойдут колеса которые дадут возможность разворачиваться в узких коридорах поля, благодаря своему строению и возможности всенаправленного движения, а так же подходящей нам скорости перемещения примерно равной 1м/с. Была составлена структурная схема (Рис.5) для реализации управления таким роботом.

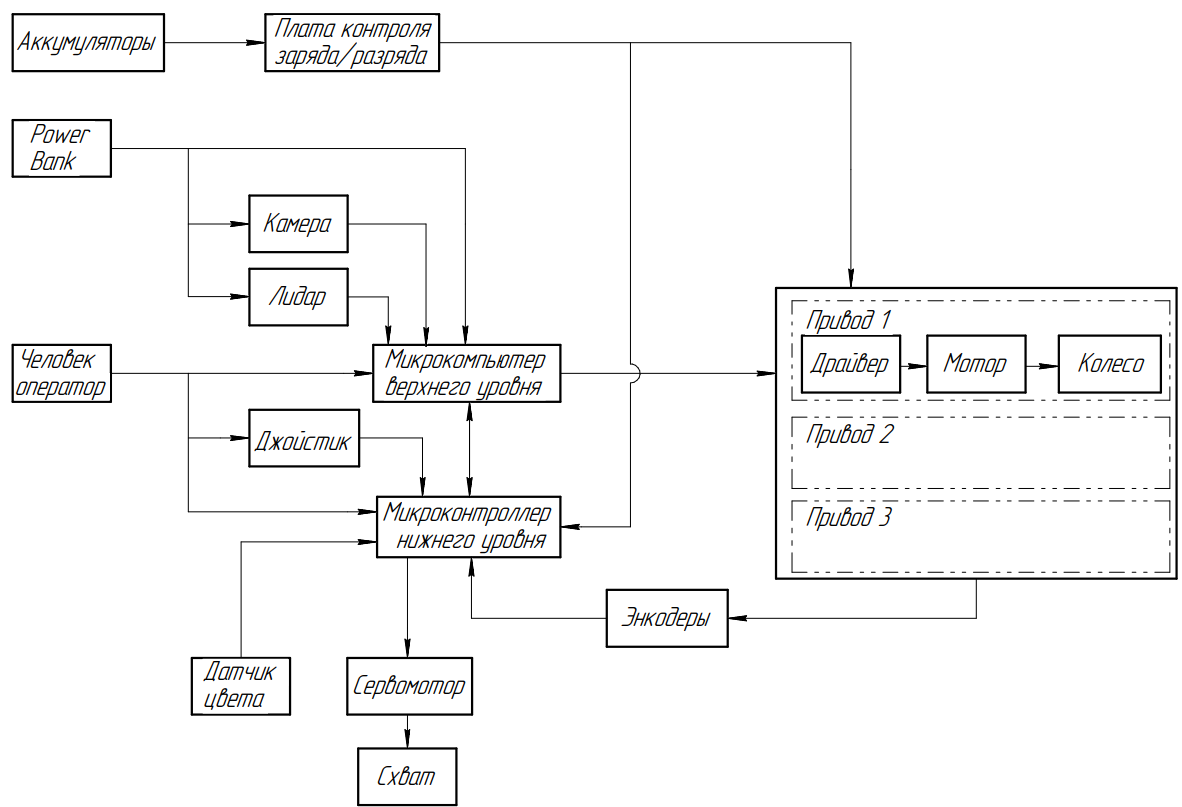


Рис. 5 Структурная схема системы управления робота

3.2 Анализ электронных компонентов

Электронная часть макета робота состоит из следующих компонентов:

* микрокомпьютер высокого уровня;
* микроконтроллер низкого уровня;
* двигатели с редукторами;
* драйверы для двигателей;
* камера;
* датчик расстояния;
* сервоприводы;
* лидар;
* питающие элементы.
* датчики линии

Далее рассмотрим выбранные устройства, оценив их преимущества и недостатки.

3.2.1 Микрокомпьютер высокого уровня.

Исходя из рассмотренных ниже характеристик в качестве микрокомпьютера выбран Raspberry Pi 4.

Преимущества Raspberry Pi:

* доступность;
* возможность подключения большого количества периферийных устройств;
* возможность установки операционной системы.
* возможность работать с ROS2.
* вычислительная мощность

Недостатки Raspberry Pi:

* большие габариты платы.
* цена

3.2.2 Микроконтроллер нижнего уровня.

В качестве управляющего микроконтроллера можно использовать Arduino, STM32. Рассмотрим их подробнее:

Преимущества Arduino:

* доступность;
* малые габариты (платы Nano и UNO);
* наибольшее количество доступных материалов в Интернете.

Недостатки Arduino:

* малая вычислительная мощность (8 битный процессор);
* несовместимость с камерами, доступными на рынке.

Преимущества STM32:

* большая вычислительная мощность;
* возможность установки операционной системы;
* низкий уровень работы с периферией.

Недостатки STM32:

* сложность программирования;
* наименьшее количество доступных материалов в Интернете.

Итог: исходя из рассмотренных выше микроконтроллеров, для нижнего уровня используем Arduino Mega, в виду достаточного количества портов и большого количества библиотек для работы с нашей периферией.

3.2.3 Двигатели с редуктором и энкодером

Данные двигатели будут использованы как основные приводные двигатели для подвижной части робота на omni-колесах. Для этой задачи подойдет мотор-редуктор постоянного тока с крутящим моментом больше 0,5 кгс/см , напряжение на питание которого будет 12В (Таблица 2).

Таблица 2. Характеристики двигателей семейства JGB37-520

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Напряжение | Без нагрузки | | Под нагрузкой | | | | В блокировке | |
| Напряжение | Скорость | Сила тока | Скорость | Сила тока | Крутящий момент | Мощность | Крутящий момент | Сила тока |
| Вольт | об/мин | мА | об/мин | мА | Н∙м | Вт | Н∙м | А |
| 12 | 1600 | 120 | 1200 | 400 | 0,010 | 3 | 0,049 | 1 |
| 12 | 1000 | 120 | 800 | 400 | 0,037 | 3 | 0,149 | 1 |
| 12 | 531 | 120 | 424 | 350 | 0,069 | 3 | 0,275 | 1 |
| 12 | 333 | 120 | 266 | 350 | 0,112 | 3 | 0,441 | 1 |
| 12 | 178 | 120 | 112 | 350 | 0,206 | 3 | 0,824 | 1 |
| 12 | 111 | 120 | 88 | 350 | 0,333 | 3 | 1,275 | 1 |
| 12 | 76 | 120 | 60 | 350 | 0,490 | 3 | 1,961 | 1 |
| 12 | 59 | 120 | 47 | 300 | 0,618 | 3 | 2,452 | 1 |

3.2.4 Драйверы моторов

Драйвера для двигателей можно выбрать по максимальному току двигателей на основе Н-мостовой транзисторной схемы, что будет довольно рациональным решением ввиду простоты управления и экономической доступности.

Нами были выбраны одни из самых распространенных драйверов L298N, поддерживающие моторы с потреблением тока до 2 А. Наши двигатели подходят.

3.2.5 Камера

В качестве камеры была выбрана веб-камера Logitech C 270. Выбор производился исходя из уже имеющихся ресурсов, совместимых с Raspberry Pi и достаточно высокого разрешения матрицы, а также подключение с помощью универсального разъема USB, которое поддерживается Hi-Speed-интерфейсом USB 2.0 и имеет пропускную способность до 480 Мбит/с, а также небольшого веса и габаритов камеры.

3.2.6 Датчик линии VEX Line Tracker

В качестве датчиков линии рассматривались датчик цвета с ИК-фильтром TCS34725, датчик линии/препятствия KY-033 и датчик линии VEX Line Tracker.

Датчик цвета TCS34725 позволяет получить значение зеленого, синего, красного цвета, а так же освещенности в видимом диапазоне в формате чисел от 0 до 1023. На датчике присутствует светодиод видимого спектра излучения, излучающий белый свет и матрица фотодиодов, распознающих разные цвета. Это может быть полезно, при разном цвете линии, по которой роботу нужно следовать. Однако при этом, датчик подключается с помощью интерфейса I2C и имеет фиксированный неизменяемый адрес 0x29, что не позволяет использовать несколько таких датчиков на одной шине. Эту проблему можно решить, используя мультиплексор, однако это существенно усложнит подключение датчиков и потребует дополнительного места. Помимо этого, датчики не калибруются с должной степенью точности производителем и разность показаний существенна для двух одинаковых датчиков.

Датчик KY-033 предназначен для обнаружения препятствий в виде белых или чёрных линий. Основным его элементом является оптопара TCRT5000, состоящая из инфракрасного светодиода и фототранзистора. При наличии питания светодиод излучает свет, который затем, отражаясь от светлой поверхности, попадает на фототранзистор. В таком случае на выходе датчика устанавливается логическая единица. В противном случае, если световой поток поглощается по той или иной причине, на выходе датчика устанавливается логический ноль. На датчике предусмотрен потенциометр для настройки порога срабатывания. Одним из недостатков является то, что сигнал с датчика принимает только два значения, что не позволяет надежно реализовать распознавание линии, частично находящейся под датчиком.

VEX Line Tracker - датчик отслеживания линейной траектории состоит из инфракрасного светового сенсора и инфракрасного светодиодного индикатора. Его принцип действия схож с KY-033. Основные отличия – калибровка датчика производителем, выходной сигнал принимает значения от 0 до 1023, что позволяет сразу использовать датчики без необходимости настройки, и отслеживать, если только часть линии находится под датчиком.

Исходя из всего вышесказанного, VEX Line Tracker – более подходящий выбор из-за своей специализации и оптимизации для этой конкретной задачи.

Оптимальные характеристики линии:

1. Оптимальное расстояние до линии: 3 мм.
2. Ширина линии: 6 мм, минимум, оптимально 13 мм или более.

3.2.7 Датчик расстояния

Датчик расстояния необходим макету робота в финальном этапе соревнований для точного позиционирования при захвате бочек. В проектируемом макете достаточно будет использован 1 датчика спереди.

По регламенту соревнований поле будет представлять собой прямоугольник 1748×2000 мм с белыми матовыми стенками высотой 250 мм. На поле будет присутствовать стандартное освещение по ГОСТ 55710-2013, температура – комнатная.

В данном случае мы можем использовать такие варианты:

* ультразвуковой дальномер HC-SR04
* лазерный дальномер VL6180X

Датчик ультразвукового дальномера HC-SR04 более доступен и имеет более низкую цену, чем лазерный дальномер VL6180X.

Также HC-SR04 обладает хорошей точностью измерения расстояния на коротких и средних дистанциях, чего будет достаточно для нашего проекта. Лазерный дальномер VL6180X обладает излишней точностью измерений.

Таким образом, выбор датчика ультразвукового дальномера HC-SR04 вместо лазерного дальномера VL6180X может быть обоснован его доступностью, низкой ценой и достаточной точностью для данных соревнований.

Принцип работы датчика:

Датчик генерирует ультразвуковой импульс (Рис. 4), который отражается от объекта и возвращается обратно к датчику. Путем измерения времени между отправкой и приемом импульса, датчик вычисляет расстояние до объекта

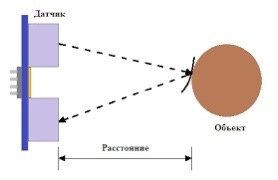
****

Рис. 4 Принцип работы датчика расстояния

3.2.8 Сервомотор

Для открытия и закрытия схвата используется сервопривод TS90A. Данный сервопривод способен установить и удерживать угол в диапазоне от и обладает крутящим моментом до 1,3 кг\*см.

Характеристики

* Модель: TianKongRC TS90A.
* Тип сервопривода: установка и удержание угла.
* Диапазон поворота вала: 0–180°.
* Диапазон ширины принимаемого импульса: 500–2500 мкс.
* Крутящий момент: 1,3 кг·см.
* Скорость вращения вала: 60° за 0,09 сек (180° за 0,27 сек).
* Аппаратный интерфейс: контакты S-V-G (PBS-3).
* Программный интерфейс: PDM (Pulse-Duration Modulation).
* Внутренний обработчик сигналов: аналоговый.
* Напряжение питания: 4,8–6 В.
* Потребляемый ток: 100 мА.
* Ток блокировки: 300 мА.
* Материал шестерней: нейлон.
* Материал корпуса: пластик.
* Длина кабеля: 25 см.
* Вес: 9 г.

3.2.9 Питающие элементы

При создании робота одним из важнейших аспектов является выбор подходящих элементов питания, так как в роботе предусмотрены различные электронные устройства (Рис. 5), требующие разного напряжения питания.

В качестве питающих элементов используются три высокотоковых литий-ионных аккумулятора 18650 с номинальным напряжением 3.7 В емкостью 3400 mAh и с максимальным продолжительным током 10 А. Аккумуляторы не обладают встроенным модулем защиты от переразряда или перезаряда и подключены последовательно, при таком подключении суммарное напряжение составляет 12,6 В при 100% заряде.

Для того, чтобы обеспечить электропитанием все устройства на роботе необходимы следующие модули:

* BMS (от англ. Battery Management System) – система управления аккумуляторами. BMS осуществляет контроль зарядки разрядки, защиту от перегрева и короткого замыкания, балансировку аккумуляторов;
* DC-DC преобразователь для преобразования выходного напряжения с BMS в напряжение питания компонентов, требующих для работы 5 В;
* портативный аккумулятор для питания одноплатного компьютера, лидара и камеры;

В качестве BMS была выбрана плата, рассчитанная на 3 аккумулятора 18650 и с максимальной силой тока 10 А.

В качестве DС-DC преобразователя был выбран понижающий преобразователь с настраиваемым выходным напряжением XL 4015. Модуль обладает защитой от короткого замыкания, и по характеристикам, заявленным производителем при преобразовании входного напряжения 12 В в выходное 5 В обеспечивает силу тока в 5 А. Однако на практике оказалось, что максимальный ток при таком преобразовании составляет 1~1,5 А, при превышении которого модуль уходит в защиту и отключает нагрузку, что не позволяет питать одноплатный компьютер Raspberry Pi 4, потребление которого составляет до 3 А.

Для данного проекта выбор портативного аккумулятора пал на REDMI Power Bank PB100. Данный аккумулятор обладает ёмкостью 10000 mAh и обеспечивает стабильное напряжение питания 5 В и максимальную продолжительную силу тока 3 А, что делает его подходящим для питания Raspberry Pi 4 с подключенными периферийными устройствами.

***Вывод:***

Таким образом, выбраны все компоненты макета робота и составлена электрическая схема (Рис. 6).

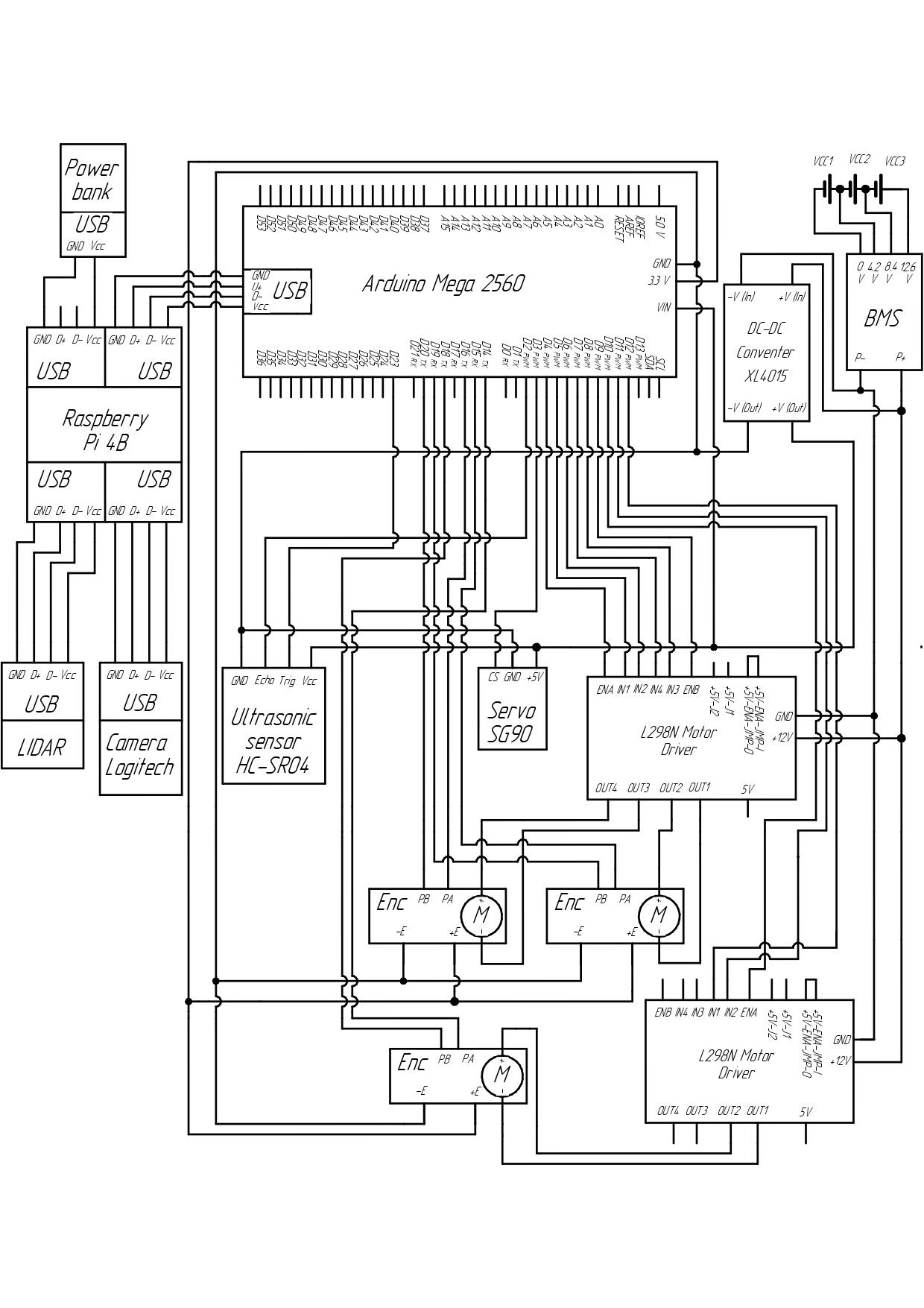


Рис. 6 Электрическая схема робота

# Конструирование подвижной платформы

Были рассмотрены различные подходы к решению задачи кинематики данного МРС [2, 6]. Наибольший интерес для данного проекта представляют упрощенный и полный расчёт кинематики.

4.1 Упрощенный расчет кинематики

Допустим, что сила трения между роликом колеса и поверхностью не зависит от положения колеса относительно направления движения робота. Следовательно, имеет смысл поступательное движение роботу сообщать колесом, ось вращения которого составляет наибольший угол с заданным направлением движения. Также колесо, ось вращения которого образует наименьший угол с заданным направлением движения, целесообразно остановить, потому что вращение этого колеса будет слабо сказываться на поступательном движении робота, но будет приводить к вращению его вокруг собственной оси. Третье колесо, занимающее промежуточное положение, необходимо использовать для подруливания робота для того, чтобы компенсировать составляющую скорости первого колеса, направленную перпендикулярно заданному направлению движения [3].

Исходя из геометрических соображений (Рис. 7), направление вращения подруливающего колеса должно быть противоположным направлению вращения ведущего. Скорость вращения подруливающего колеса будет ниже, чем ведущего, за исключением случая движения в направлении оси симметрии между ведущим и подруливающим колесом (в этом случае скорости будут равны).

Таким образом, ввиду симметрии достаточно рассмотреть 12 зон направлений поступательного движения робота. При движении робота в направлении зон 1 и 7 на рис. 5 ведущим будет являться колесо III, а неподвижным колесо I; в направлении зон 2 и 8 ведущим будет являться колесо II, а неподвижным колесо I; в направлении зон 3 и 9 ведущим будет являться колесо II, а неподвижным колесо III; в направлении зон 4 и 10 ведущим будет являться колесо I, а неподвижным колесо III; в направлении зон 5 и 11 ведущим будет являться колесо I, а неподвижным колесо II; в направлении зон 6 и 12 ведущим будет являться колесо III, а неподвижным колесо II.

**Решение кинематической задачи.** Рассмотрим в качестве примера движение робота в направлении зоны 2. Скорость вращения колеса, направленную по часовой стрелке если смотреть с внешней стороны, примем положительной. Влиянием на скорость поступательного движения робота проскальзывания подруливающего колеса пренебрежем. Тогда, зависимость касательной скорости ведущего колеса от скорости движения робота ν будет иметь вид

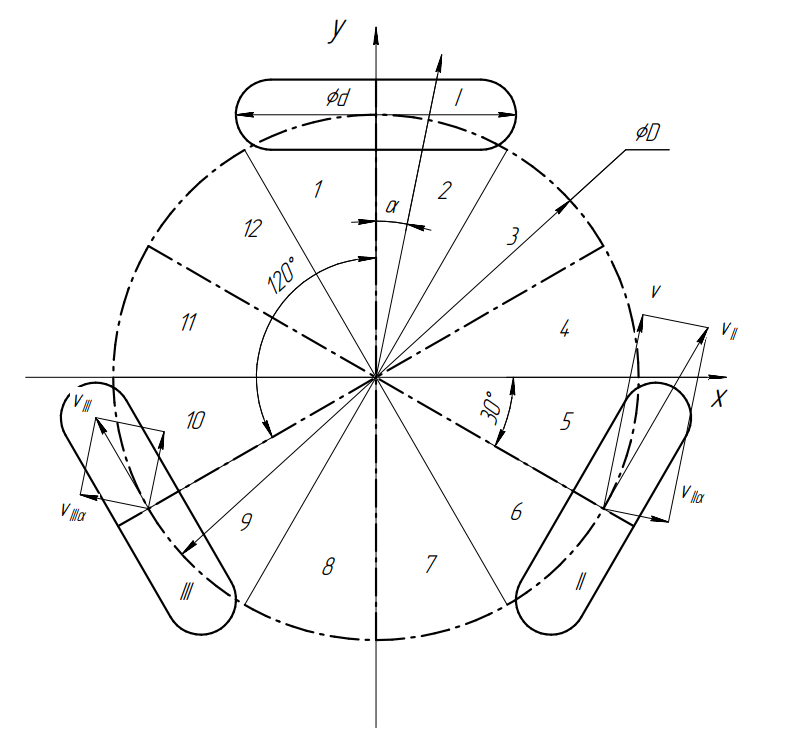


Рис. 7 Схема скоростей колес при движении робота в направлении зоны 2.

Исходя из необходимости обеспечить условие , касательная скорость подруливающего колеса определится по формуле

Скорости вращения ведущего и подруливающего колеса соответственно равны

где *d*- диаметр колес, мм.

Для других зон движения скорости вращения колес определяются аналогично.

Полученные простейшие зависимости являются базовыми для управления движением робота с тремя одинаковыми независимыми ведущими омни - колёсами и могут использоваться при создании систем управления подобными роботами. У реального робота значительное влияние на скорость и направление движения будет оказывать изменение коэффициента трения с поверхностью в зависимости от положения колес относительно направления движения робота и скорости их вращения. В ходе дальнейших практических испытаний данный способ не позволил обеспечить необходимой точности движения.

4.2 Полный расчет кинематики

Обозначим подвижную систему координат (ПСК), связанную с геометрическим центром робота как , неподвижную систему координат (НСК) как . Положение робота в НСК задается координатами . Три омни-колеса расположены относительно локальной системы координат под углом , где [5].

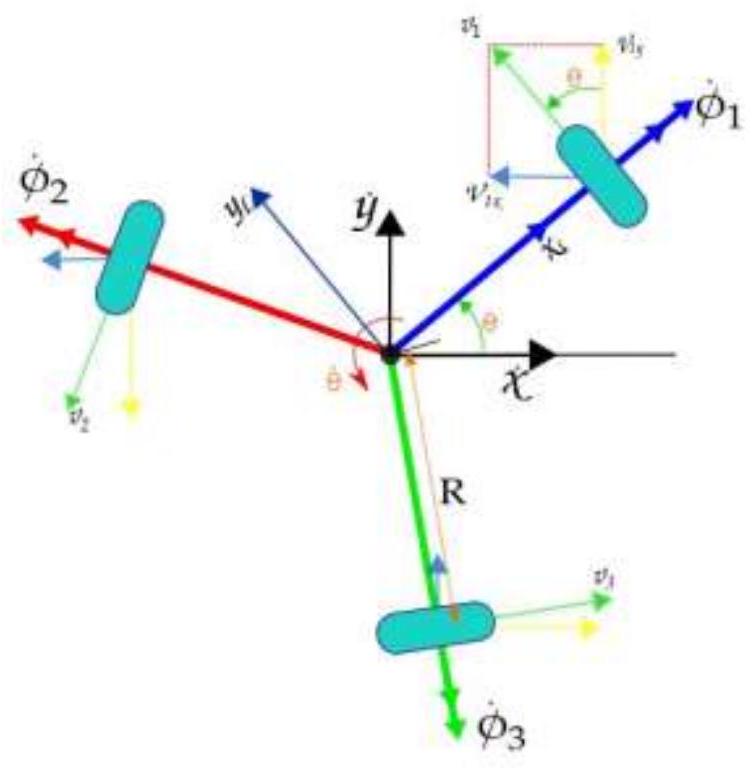


Рис. 8. Схема макета робота на трех омни-колесах.

**Решение кинематической задачи.** Абсолютная скорость геометрического центра колеса состоит из переносной () скорости и относительной скорости () робота:

где

Обобщим уравнения для всех колес, учитывая, что они расположены под углом :

R это расстояние от геометрического центра робота до центра колес.

Абсолютная скорость связана с относительной следующим отношением:

Подставим (3) и (4) подставляются в (1), получаем:

Переносная скорость связана с угловой скоростью колес через:

где - радиус омни-колеса,

Используя обратную матрицу Якоби, можно вывести угловую скорость колес, при этом известны как радиус колес, так и размеры платформы . Мы можем преобразовать (9) в матричное представление (10) и (11)

Отношение между угловой скоростью колес и абсолютной скоростью робота .

Чтобы управлять роботом в ПСК, мы можем преобразовать абсолютные координаты в местные координаты с помощью следующего уравнения:

После подстановки (10) и (11), мы получаем следующее уравнение:

Для удобной реализации в программном коде мы можем записать это матричное отношение в виде трёх отдельных уравнений для :

Данный способ расчета кинематики позволил добиться требуемой точности в ходе практических испытаний робота, и был использован в дальнейшем.

4.3 Направление скоростей на колесах робота при различных видах движения:

Рассмотрим направления векторов скорости при различных направлениях движения робота (Рис. 9).

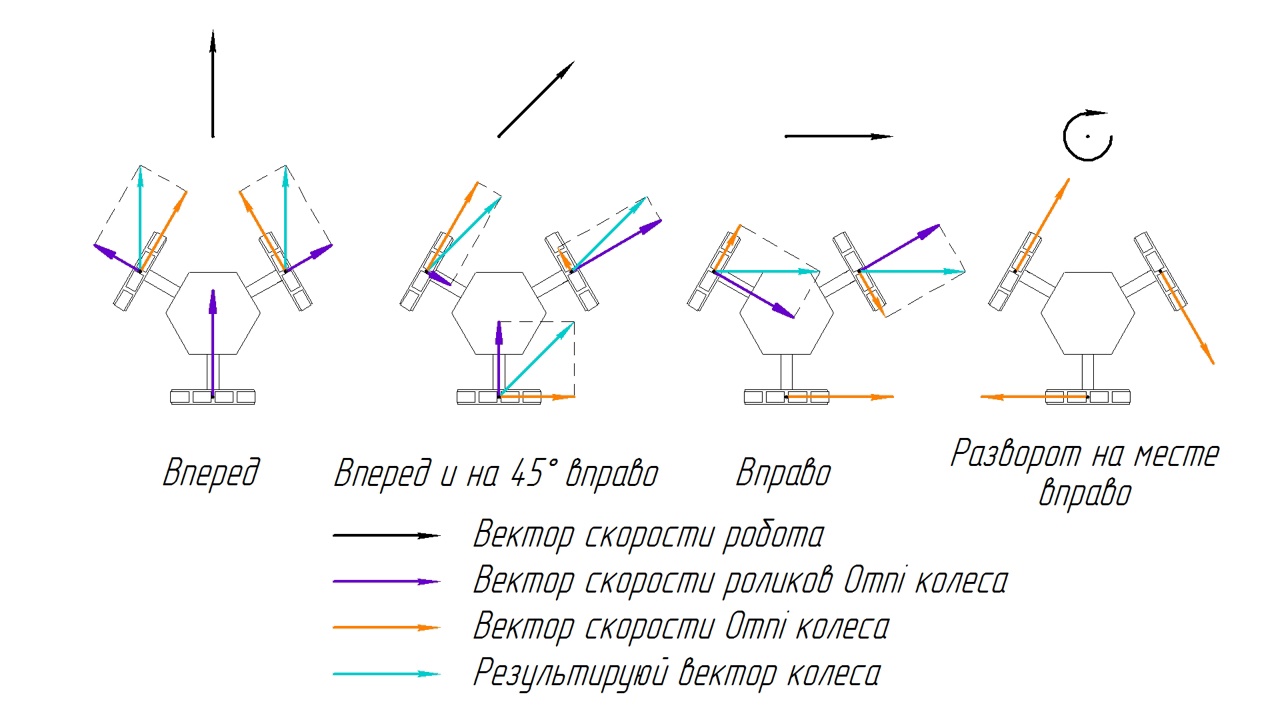


Рис. 9 Изображение векторов скоростей при различном движении

4.4 Расчет мощности двигателей

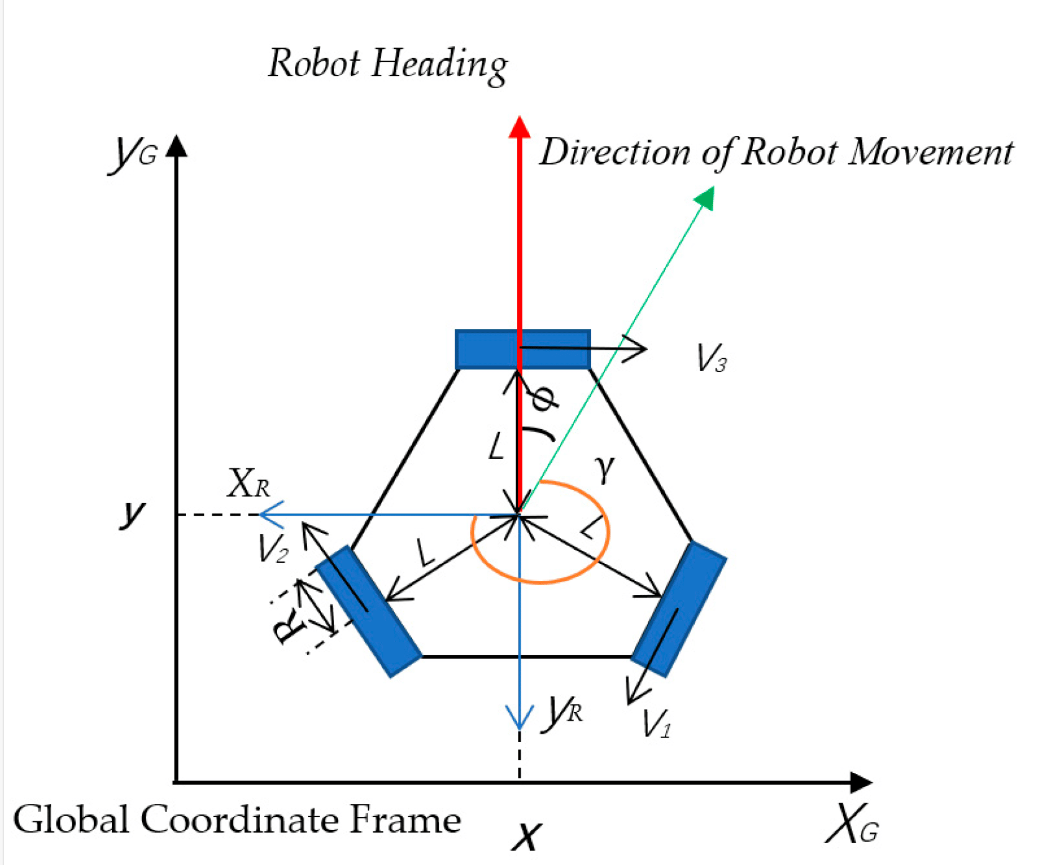
****

Рис. 10 Схема для расчета мощности робота

Наиболее затратное направление движения – под углом 30 градусов, при котором направление скорости одного колеса совпадает с направлением движения робота [7].

Определим мощность, необходимую для движения робота массой 5 кг со скоростью 1 м/с под углом 30 градусов.

– это мощность, тех сил трения, которые компенсируют друг друга во время движения робота на трех омни - колесах. – это масса робота, – коэффициент трения робота, – угол между направлением робота и его вектором скорости движения. – скорости колес робота, коэффициенты и соответственно коэффициенты сил трения качения и скольжения.

-мощность, необходимая, чтобы преодолеть силу трения при движении робота, – текущая скорость робота, -коэффициент трения.

Общая мощность, необходимая для движения

# 5. Расчет схвата для захвата тел вращения

5.1 Построение профиля поворотных губок (рабочего элемента схвата)

К механическим зубчатым устройствам, предназначенным для манипулирования цилиндрами предъявляются требования обеспечения центрирования деталей при изменении D их диаметральных размеров, изменяющихся при механической обработке. Такие зубчатые устройства оснащают поворотными губками криволинейной формы (см. Рис. 5). Губки должны быть спрофилированы таким образом, чтобы обеспечивать в определенном диапазоне центрирование цилиндров (в нашем случае бочек) различного диаметра. Верхние части губок делают одинаковой ширины, а нижние срезают так, чтобы они заходили одна за другую. Это позволяет надежно центрировать бочку даже в том случае, когда в зоне действия губок оказывается ступень с перепадами диаметров [8].

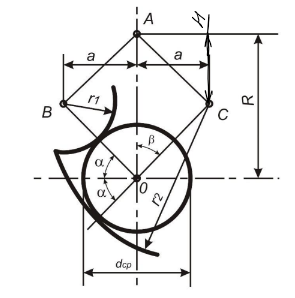


Рис. 11 Схема, поясняющая построение профиля поворотных губок дугами окружности.

Для точного центрирования цилиндра профиль губок аппроксимируют дугами окружности .

Исходные данные:

D - диаметр зажимаемых деталей, мм; R – радиус поворота губок, мм; β – угол, получаемый построением (Рис. 5) Исходными данными при построении профиля губок являются диаметр зажимаемой детали D и центральный угол 2α между точками контакта губок с деталью. Рекомендуются:

Последовательность построения профиля показана на рисунке 5. Размер R между центром зажимаемого цилиндра и осью поворота губки выбирается конструктивно и должен быть больше d. Из точек В и С строят дуги радиусами r1 и r2. Эти точки лежат на расстоянии R/2 от оси поворота губки. Горизонтальная координата точек В и С определяется по формуле

Радиусы дуг профиля губок:

Профили губок симметричны. Если обе губки поворачиваются вокруг общей оси (точка А), то точки контакта губок с деталью располагаются симметрично. Если губки имеют разные оси поворота (точки и ), то точки контакта детали с профилями радиуса r1 удаляются, а с профилями радиуса r2 – сближаются. Центральный угол φ (угол О) рекомендуется выбирать в пределах 0 ≤ φ ≤ (2α - 40°).

Далее приведен пример расчета механизма захватного устройства, одного из возможных вариантов его конструктивного исполнения.

5.2 Расчет механизма захватывающего устройства

(-передаточное отношение, , ,

Таким образом, были найдены все необходимые параметры для создания захватного устройства робота (Рис. 12), которое затем было построено в программном обеспечении КОМПАС 3Д и напечатано на 3Д принтере (Рис. 13)

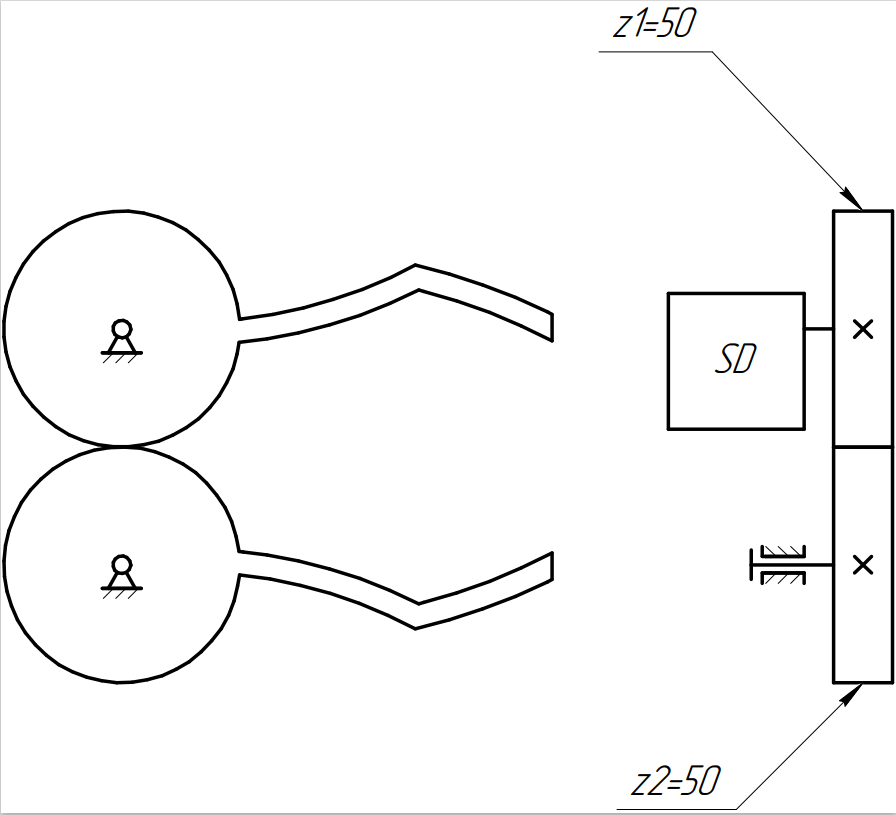


Рис. 12 Кинематическая схема схвата.

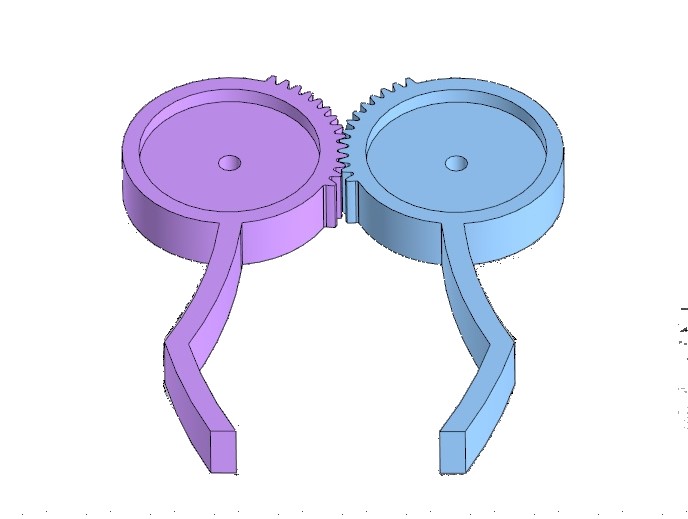


Рис. 13 3Д модель схвата.

# 6. Реализация управления нижнего уровня робота

Робот с тремя омни - колесами представляет собой сложную систему, требующую точного соответствия реальной и желаемой угловой скорости колес для достижения требуемого направления и модуля скорости.

* 1. Измерение угловой скорости моторов

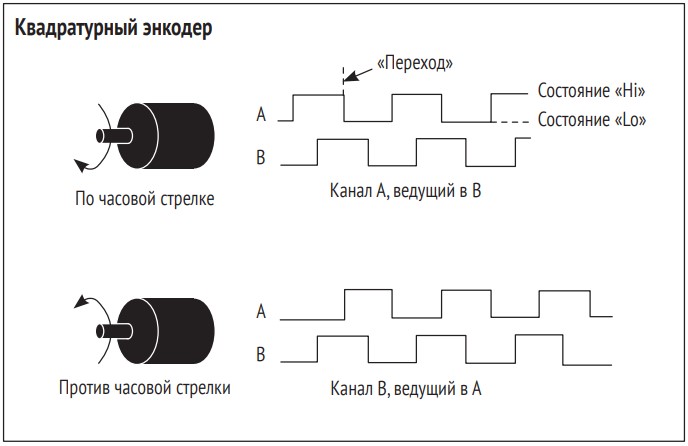
На каждом из выбранных для робота моторов установлен инкрементальный квадратурный энкодер. Этот тип энкодеров позволяет определить не только угловую скорость вала мотора, но и направление вращения. Принцип действия следующий: на валу мотора закреплён магнитный диск и неподвижная печатная плата, на которой расположены два датчика Холла. При вращении магнитного диска на выходе датчиков наблюдается либо состояние Hi, либо Low (Рис. 14). 

Рис. 14 Принцип работы квадратурных энкодеров.

В программной реализации на микроконтроллере Arduino (Приложение 5) при переходе одного из датчиков Холла (например А) из состояния Low в состояние High, вызывается функция прерывания, внутри которой считывается состояние другого датчика. При этом, если состояние другого датчика High то к значению оборотов прибавляется 1, в противном случае 1 вычитается. Таким образом, подсчитано количество оборотов вала мотора.

Чтобы найти количество оборотов выходного вала редуктора, нужно число оборотов вала мотора умножить на передаточное отношение.

6.2 Регуляция угловой скорости моторов

ПИД-регуляция будет применена к моторам робота для обеспечения точного и стабильного управления. Этот метод управления позволяет учесть ошибку, интегральную составляющую и дифференциальную составляющую для корректировки управляющего сигнала. Так же необходим отдельный ПИД регулятор на каждый мотор робота, это обосновано необходимостью точного контроля скорости и положения каждого колеса, что особенно важно для робота с омни - колесами, который должен быть способен на маневрирование в различных направлениях.

В программной реализации используем библиотеку для Arduino IDE (среды программирования для микроконтроллера) AccelMotor v1.3. Эта библиотека гибкая и позволяет настроить регуляцию для любого типа моторов. Кроме того, библиотека позволяет приближенно найти коэффициенты ПИД регулятора.

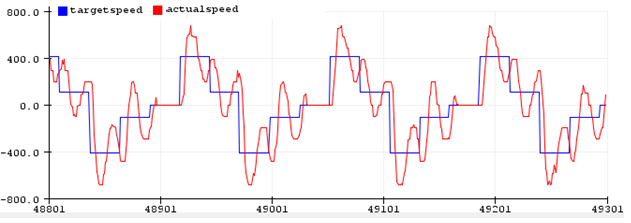
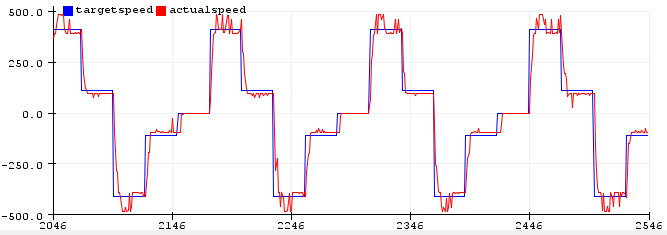
Так задача сводится только к выбору коэффициентов при пропорциональной, интегральной и дифференциальной составляющей на практике (Рис.15).

Рис. 15 Графики желаемой скорости (синий цвет) и реальной скорости (красный цвет). Сверху вниз: коэффициенты подобраны правильно, слишком большая интегральная составляющая, слишком малая интегральная составляющая

Таким образом, были подобраны оптимальные коэффициенты, использование Arduino, квадратурных энкодеров и ПИД-регуляции позволило реализовать нижний уровень управления для робота с тремя омни - колесами, обеспечивая точное и стабильное движение.

# 7. Система технического зрения для распознавания цвета и положения бочек

7.1 Выбор программного обеспечения

В ходе заключительного этапа роботу необходимо обнаруживать и определять цвет бочек, а также зон выгрузки. Решение этой задачи с помощью датчиков цвета крайне затруднительно, поэтому необходимо использовать камеру и обрабатывать изображения с неё в режиме реального времени. Для выполнения такой обработки необходимо выбрать программное обеспечение.

Python является одним из самых популярных языков программирования, который обладает простым и понятным синтаксисом, что делает его отличным выбором для разработки алгоритмов работы с машинным зрением. Кроме того, Python имеет большое разнообразие библиотек и инструментов, что упрощает разработку и ускоряет процесс создания программ.

OpenCV2, в свою очередь, - одна из широко используемых библиотек для компьютерного зрения, обработки изображений и машинного обучения. Она предоставляет набор функций и алгоритмов для работы с изображениями, включая обнаружение объектов, распознавание лиц, сегментацию изображений и многое другое. Благодаря своей мощности и гибкости, OpenCV2 отлично подходит для работы с компьютерным зрением в реальном времени.

ROS 2 (Robot Operating System 2) – фреймворк, обладающий расширяемостью, гибкостью и поддержкой различных приложений в реальном времени. ROS 2 предоставляет инструменты для разработки, тестирования и выполнения сложных систем робототехники, включая обработку изображений и компьютерное зрение.

Таким образом, выбор ROS 2, Python и OpenCV2 обоснован их широкой популярностью, богатым функционалом и удобством в использовании.

7.2 Реализация распознавания цветов и объектов

Для определения границ объектов применим функцию cv2.findContours() из библиотеки OpenCV2 – основной инструмент для анализа, распознавания и обнаружения объектов [9]. Контур представляет собой кривую, которая состоит из набора точек, образующих границу объекта или области точек схожего цвета или насыщенности на изображении.

Функция получает на вход исходное изображение, метод поиска контура и метод аппроксимации контура (Рис. 16).Для решения поставленной задачи и исходя их производительности одноплатного компьютера Raspberry Pi 4 были выбраны метод поиска контура RETR\_TREE, который позволяет получить все контуры из изображения и метод аппроксимации CHAIN\_APPROX\_SIMPLE , снижающий затраты вычислительных ресурсов, удаляя избыточные точки.



Рис. 16 Контур изображения красной лазерной указки (показан зелёным цветом)

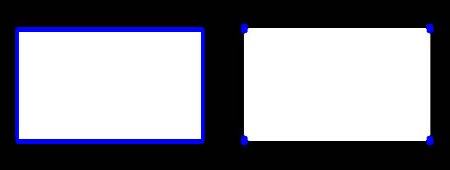


Рис. 17 Слева – метод аппроксимации CHAIN\_APPROX\_NONE (734 точки), справа - метод аппроксимации CHAIN\_APPROX\_ SIMPLE (4 точки), применённые к одному и тому же контуру

Функция возвращает два значения: список всех найденных контуров в виде массивов точек, иерархия контуров (массив, который содержит информацию о вложенности и отношениях между контурами).

После вызова функции cv2.findContours() можно использовать полученные контуры для различных целей, таких как отображение их на изображении, выделение объектов, вычисление их площади и периметра, а также для других операций обработки изображений.

Но перед тем, как искать контуры, необходимо разделить исходное изображение на несколько, содержащих только цвета интересующих нас объектов. Определение цвета усложняется тем, что на поле соревнований возможно неравномерное освещение, приводящие к существенному искажению цвета в цветовой модели RGB (Рис. 18).

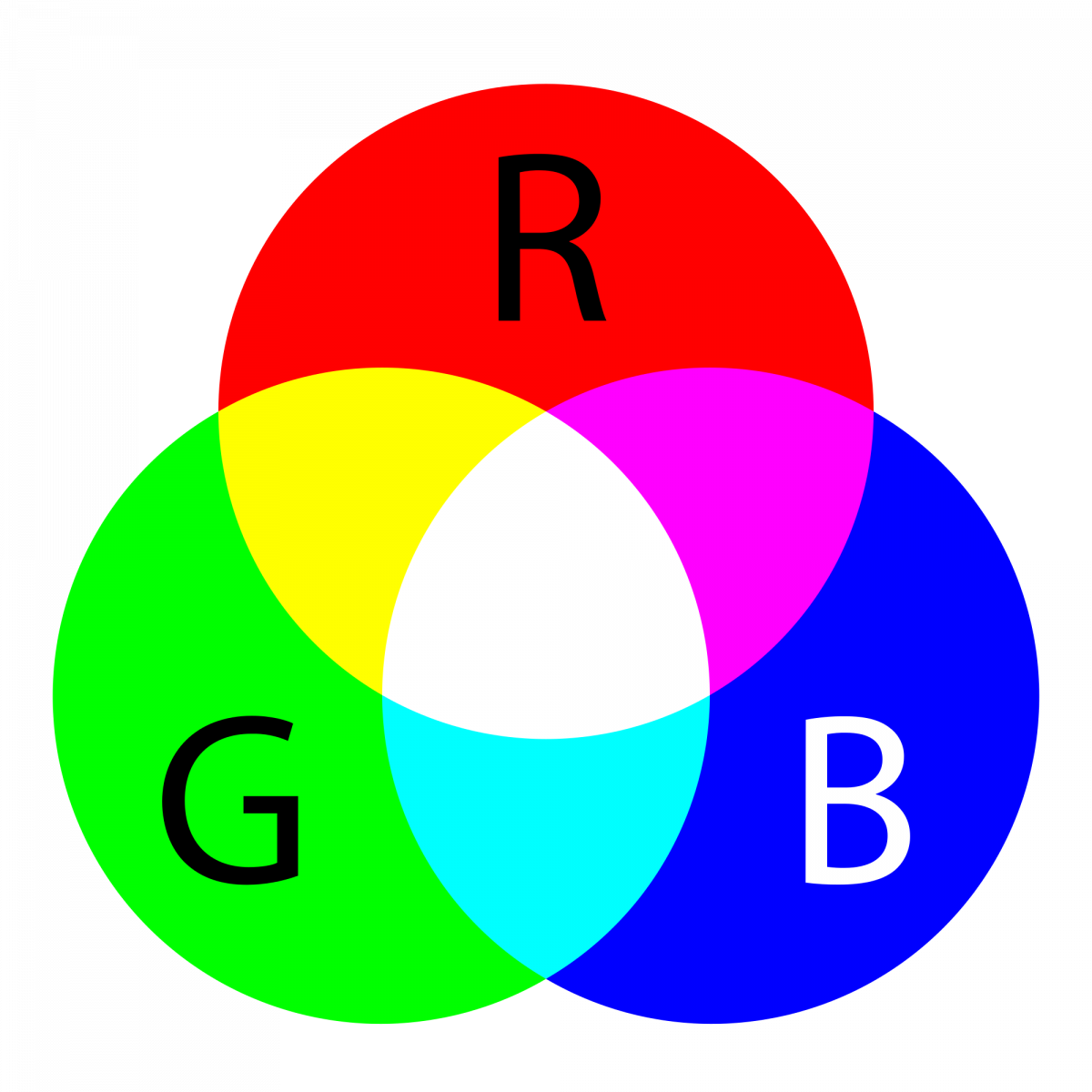


Рис. 18 RGB-цветовая модель

Для решения этой проблемы необходимо перейти в цветовую модель HSV (Рис. 19). HSV (от англ. Hue, Saturation, Value) - это цветовая модель, которая описывает цвета в терминах их оттенка (H), насыщенности (S) и яркости (V).

Компонент оттенка (H) определяет цвет в радиальной цветовой модели и измеряется от 0 до 179 в OpenCV2 и от 0 до 240 в графическом редакторе Paint.

Компонент насыщенности (S) определяет чистоту или насыщенность цвета и измеряется в процентах от 0 до 255 в OpenCV2 и от 0 до 240 в Paint. Низкая насыщенность означает, что цвет ближе к серому, а высокая насыщенность означает, что цвет яркий и насыщенный.

Компонент яркости (V) определяет яркость цвета и также изменяется от 0 до 255 в OpenCV2 и от 0 до 240 в графическом редакторе Paint. Значение 0 означает черный, а максимальное значение 255 - полная яркость.

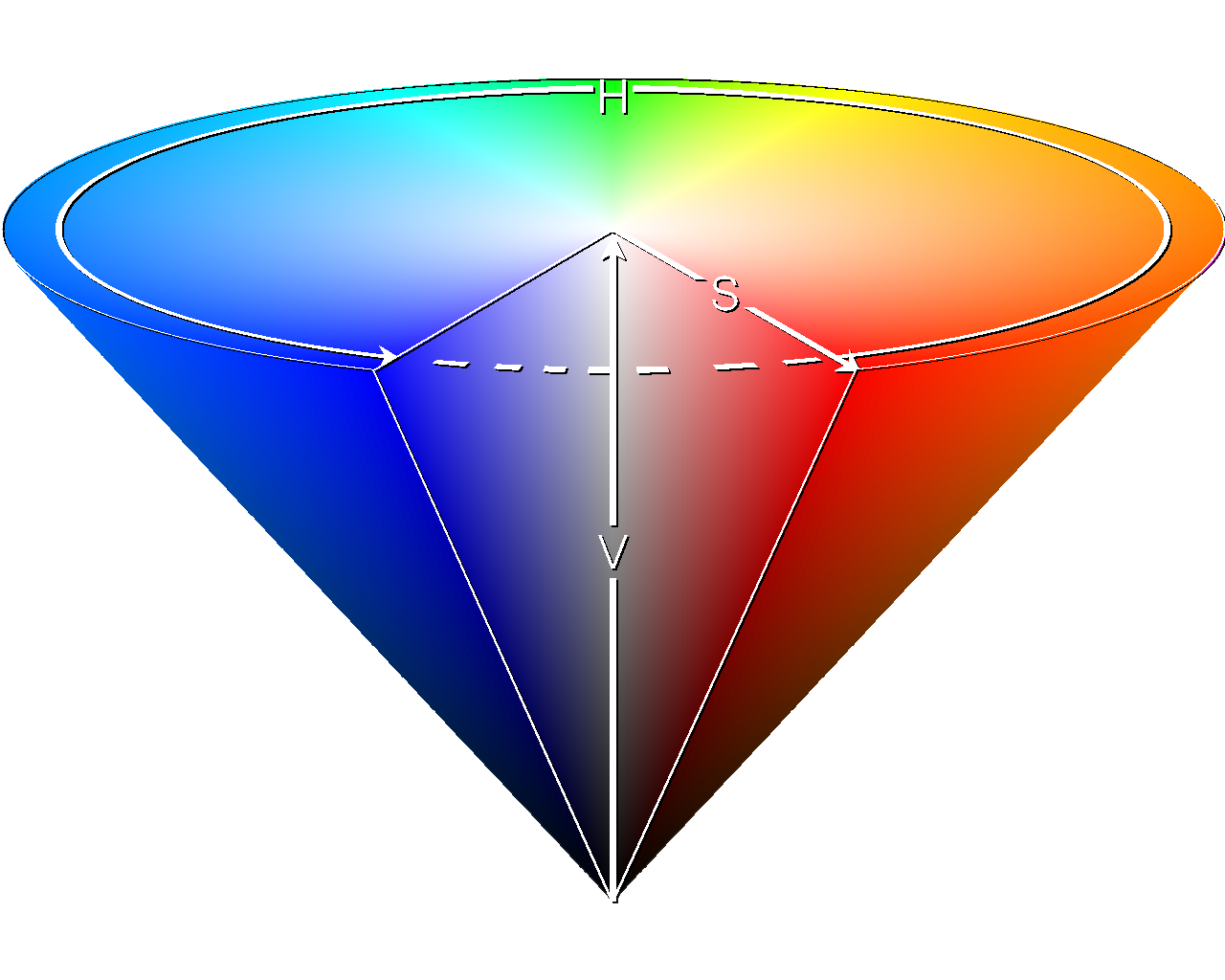


Рис. 19 Коническое представление HSV

Пространство HSV удобно для работы с цветом и насыщенностью, поскольку отделяет цветовую информацию от яркости, что делает его более устойчивым к изменениям освещения. Переход от RGB в HSV осуществляется функцией cv.cvtColor(frame, cv.COLOR\_BGR2HSV) , в которую передаем снимок (frame) и флаг (cv.COLOR\_BGR2HSV), означающий желаемый переход.

Для разделения цвета объектов можно использовать функцию cv2.inRange(hsvFrame, lower, upper), где hsvFrame - изображение в цветовой модели HSV, lower – нижняя граница маски, upper - верхняя граница маски. Маска – трафарет, который накладывается на изображение и скрывает его часть. На выходе получаем изображение, содержащее только пиксели со значениями цвета в пределах границы маски в HSV. Точные значения границ маски подбирались опытным путем.

Подбор маски для зеленого цвета был сложнее, по сравнению с подбором маски для других цветов. Это объясняется тем, что H (оттенок) у зеленого цвета находится в середине диапазона и может принимать большое число значений. Так при слишком большом значении максимального оттенка (Н) зеленый цвет ошибочно распознаётся у синих объектов (Рис. 20).

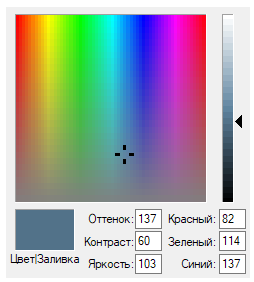
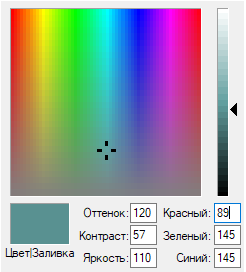
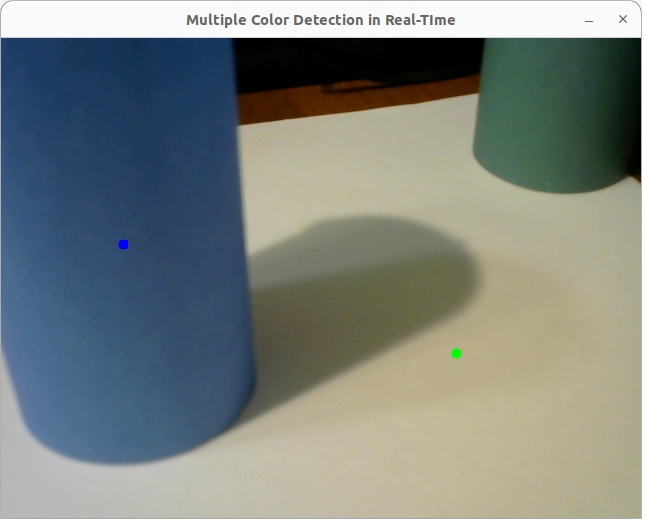


Рис. 20 Сверху – снимок синего цилиндра, точками соответствующего цвета распознаны зеленый и синий объекты, слева снизу – значение цвета возле центра распознанного зеленого объекта, справа снизу – значение цвета возле центра распознанного синего объекта, значения маски подобраны неправильно

При малом значении минимального оттенка (Н) зеленый цвет ошибочно распознаётся у серых или красных объектов (Рис. 21).



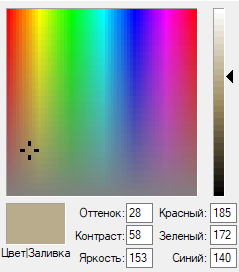
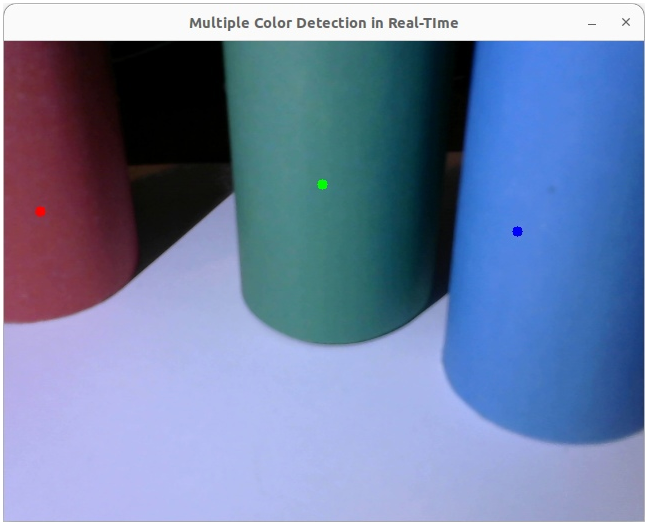
 

Рис. 21 Сверху – снимок синего и зеленого цилиндра, точками соответствующего цвета распознаны зеленый и синий объекты, слева снизу – значение цвета возле центра распознанного зеленого объекта, справа снизу – значение цвета возле центра зеленого цилиндра, значения маски подобраны неправильно

При правильно подобранных значениях коэффициентов успешно распознаются цилиндры разных цветов в одном кадре (Рис. 22).



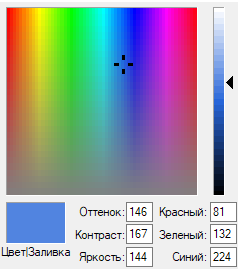
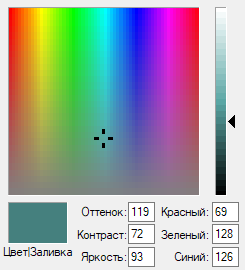
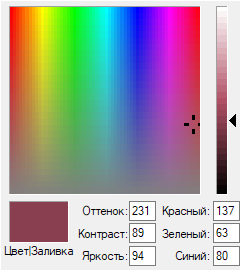


Рис. 22 Сверху – снимок красного, зеленого, синего цилиндра, точками соответствующего цвета распознаны красный, зеленый, синий объекты, снизу слева направо – значения цветов около центров распознанных объектов в соответствующем порядке, значения маски подобраны правильно

После подбора были выбраны следующие значения оттенков (значения указаны в формате, используемом OpenCV2):

-для красного цвета оттенок (H) от 136 до 179

-для синего цвета оттенок (Н) от 94 до 120

-для зеленого цвета оттенок (Н) от 30 до 93

Таким образом, после определения контуров объектов интересующих цветов можно отбирать их по соотношению высота – ширина, и использовать в дальнейшем для планирования траектории и т.д., то есть задача по распознаванию цветных бочек решена (Приложение 5).

# 8. Реализация езды по линии

Для езды по линии мы использовали аналоговые датчики фирмы VEX (Рис. 23):



Рис. 23 Датчик линии VEX Line Tracker

Нумерация датчиков приведена на чертеже ниже (Рис. 24):

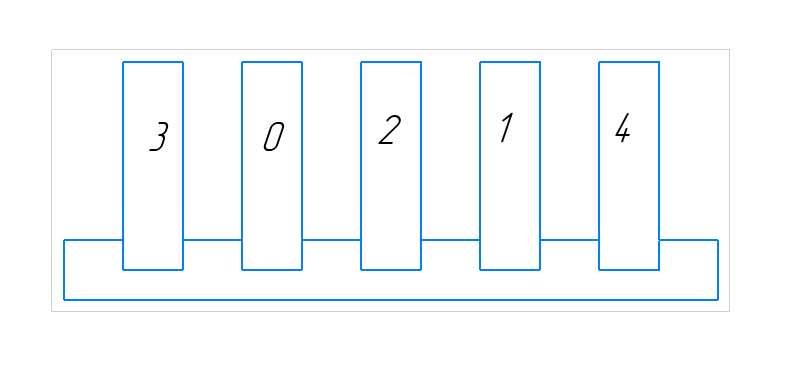


Рис. 24 Нумерация датчиков

Данные датчики выводят значения в диапазоне от 0 до 1023. Максимальное значение соответствует чёрному цвету, минимальное – белому.

Всего датчиков 5: два крайних (3-й и 4-й) отвечают за поворот с максимальной возможной угловой скоростью (иначе робот уедет с линии) (Рис. 24).

Ниже представлен код в Arduino IDE (весь код представлен в Приложении 3) для поворота при нахождении линии под одним из крайних датчиков (для них реализован релейный регулятор):

void normal\_drive(int ugl){  
 if (analogRead(3) > threshold[3] && analogRead(4) < threshold[4] ){  
 ang\_vel = ugl;  
 x\_l = 0.1;  
 }  
  
 else if (analogRead(4) > threshold[4] && analogRead(3) < threshold[3]){  
 ang\_vel = -ugl;  
 x\_l =-0.1;  
 }

Датчик посередине (ему соответствует номер 2) используется для подсчета коэффициентов ПИД регуляции. Ошибка считается как разность показаний двух ближайших ко 2-му датчиков (их номера соответственно 0 и 1).

else if (analogRead(4) > threshold[4] && analogRead(3) > threshold[3]){  
 Kp = 0.0007 \* (1023 - analogRead(2));  
 Kd = 7 \* Kp;  
 Ki = 0.0001;  
 linefollow();  
}

analogRead(i) – встроенная в ArduinoIDE функция для считывания значения с аналогового пина платы. Она выдает значения от 0 до 1023.

threshold[i] – это список из пороговых значений, соответствующих каждому датчику. С помощью данных значений решается, находится ли черная линия под датчиком и, соответственно, нужно ли роботу поворачивать.

Значения из этого списка получаем в результате калибровки робота перед началом его движения. Реализовано это с помощью функции calibration() :

void calibration(){  
 for ( int i = 0; i < 5; i++){  
 minValues[i] = analogRead(i);  
 maxValues[i] = analogRead(i);  
 }  
 long int j = millis();  
 for (j; j < 4400; j=millis()){  
 ang\_vel = 3;  
 robot\_move();  
 for ( int i = 0; i < 5; i++){  
 if (analogRead(i) < minValues[i]){  
 minValues[i] = analogRead(i);  
 }  
 if (analogRead(i) > maxValues[i]){  
 maxValues[i] = analogRead(i);  
 }  
 }  
 }  
  
 for ( int i = 0; i < 5; i++){  
 threshold[i] = (minValues[i] + maxValues[i]) / 2;  
 Serial.print(threshold[i]);  
 Serial.print(" ");  
 }  
 Serial.println();  
 ang\_vel = 0;  
}

Робот в течение 4,5 секунд движется по кругу. Спустя это время получаем два списка с минимальными и максимальными значениями для датчиков. После этого путём нахождения среднего арифметического между этими значениями, соответствующими каждому датчику, получаем список threshold[i].

Соответственно если на крайних датчиках значение меньше порогового – движемся используя 0 и 1 датчики (функция linefollow()) :

void linefollow(){  
 int error = (analogRead(0) - analogRead(1));  
 P = error;  
 I = I + error;  
 D = error - previousError;  
 PIDvalue = (Kp \* P) + (Ki \* I) + (Kd \* D);  
 previousError = error;  
 if (error >= 0){  
 sp = lfspeed + PIDvalue;  
 constrain(sp, 0, 255);  
 ang\_vel = map(sp, 0, 255, 0, 2);  
 }  
 if (error < 0){  
 sp = lfspeed - PIDvalue;  
 constrain(sp, 0, 255);  
 ang\_vel = map(sp, 0, 255, 0, -2);  
 }  
 if (count\_black == 1 or count\_black == 2){  
 x\_l = -0.2;  
 }  
 else{  
 x\_l = -0.2;  
 }  
 flag1 = 0;  
}

int error = (analogRead(0) - analogRead(1));: Считывает значения с двух аналоговых сенсоров (предположительно, подключенных к аналоговым пинам 0 и 1), вычисляет разницу между ними и присваивает результат переменной error. Это значение представляет собой отклонение от желаемого пути.

P = error; I = I + error; D = error - previousError; : Присваивает текущую ошибку пропорциональной составляющей (P), обновляет интегральную составляющую (I) путем накопления ошибок со временем, и вычисляет дифференциальную составляющую (D) как разницу между текущей ошибкой и предыдущей ошибкой.

PIDvalue = (Kp \* P) + (Ki \* I) + (Kd \* D);: Вычисляет общее значение PID-регулятора, объединяя пропорциональную, интегральную и дифференциальную составляющие с их соответствующими коэффициентами (Kp, Ki и Kd).

previousError = error; : Обновляет переменную previousError для использования в следующей итерации.

if (error >= 0) { sp = lfspeed + PIDvalue; constrain(sp, 0, 255); ang\_vel = map(sp, 0, 255, 0, 1); }: Если ошибка неотрицательна (что часто означает, что робот находится слева от желаемого пути), корректируется скорость (sp) на основе рассчитанного значения PID. Затем ограничивается sp в диапазоне [0, 255] и отображается на нормализованную угловую скорость (ang\_vel) в диапазоне [0, 1].

Далее следует участок инверсии (Рис. 25), где робот, наоборот, должен двигаться по белой линии:



Рис. 25 Участок инверсии

Для движения по данному участку нужно просто поменять все знаки сравнения в условиях на противоположные:

void inv\_drive(){  
 if (analogRead(3) < threshold[3] && analogRead(4) > threshold[4] ){  
 ang\_vel = 2;  
 x\_l =-0.2;  
 }  
 else if (analogRead(4) < threshold[4] && analogRead(3) > threshold[3]){  
 ang\_vel = -2;  
 x\_l =-0.2;  
 }  
 else if (analogRead(4) > threshold[4] && analogRead(3) > threshold[3]){  
 Kp = 0.001 \* (1023 - analogRead(2));  
 Kd = 0.00007 \* Kp;  
 Ki = 0.0001;  
 linefollow\_inv();  
 }  
 bl\_or\_white = 1;  
 count\_black = 1;  
 black\_time();  
}

В конце трассы роботу необходимо остановиться на участке в виде черного прямоугольника. Функция остановки реализована следующим образом:  
bool ifSquare(){  
 int tresholdTime = 5000;  
 for (int i = 0; i>5; i++){  
 if (analogRead(i)<threshold[i]) return 0;  
 }  
 if (flag == 0){  
 tmpTime = millis();  
 flag = 1;  
 }  
 if(millis() - tmpTime > tresholdTime) return 1;  
}

Если на всех датчиках робота чёрный цвет – функция выводит булевое значение 1 (True). В главном цикле loop() всей программы проверяется результат функции. И если он равен единице, робот останавливается.

if(ifSquare()){  
 x\_1 = 0;  
 ang\_vel = 0;

}

x\_l – это линейная скорость робота.

ang\_vel – угловая.

# 9. Использование RP Lidar A1

RP LIDAR A1 - это датчик, который использует лазерное излучение для измерения расстояний до объектов и создания точной карты окружающей среды . RP LIDAR A1 разработан китайской компанией SlamTec и является одним из популярных лидаров, доступных в мире робототехники и автоматизации.

Принцип работы RP LIDAR A1 основан на технологии лазерного сканирования. Устройство оснащено лазером, который вращается вокруг своей оси, излучая лазерные лучи вокруг себя. Когда лазерный луч сталкивается с объектом, отраженный сигнал регистрируется детектором на лидаре. Измеряя время, затраченное на возвращение лазерного луча, можно определить расстояние до объекта. Повторяя этот процесс многократно при вращении лазера, получается набор данных, который можно использовать для создания 2D-карты окружающей среды.



Рис. 26 RP LIDAR А1

Основные характеристики:

* диапазон расстояния (м) - 0,15 – 12 (белые объекты);
* угловой диапазон (°) - 0 - 360;
* поле сканирования (°) - -1,5 – +1,5;
* разрешение расстояния (мм) - < 1% от дистанции;
* угловое разрешение (°) - ≤ 1;
* продолжительность выборки (мс) - 0,125;
* частота дискретизации (Гц) - ≥ 8000;
* скорость сканирования (Гц) - 1 – 10 (5,5);
* вес (г) – 170;
* диапазон температур (°С) - 0 – 40.

Характеристики лазера:

* длина волны лазера (нм) - 775 – 795;
* мощность лазера (мВт) - 3 (норм) – 5 (макс);
* длительность импульса (мкс) - 110 (норм) – 300 (макс).

На данный момент реализован отъезд от препятствия в противоположную сторону следующим образом:

1. Обрабатываются значения с Лидара. Находится минимальное расстояние.
2. Если оно меньше порогового (15 см), передаётся значение угла, соответствующего данному расстоянию.
3. К углу прибавляется π.

Проекцию скорости на оси Х и У умножаем соответственно на косинус и синус нового значения угла. Угловую скорость не меняем.

self.speedx = linear\_speed \* math.cos(angle\_rad)  
self.speedy = linear\_speed \* math.sin(angle\_rad)  
self.ang\_vel = angular\_speed

1. Отправляем скорость по оси Х, У и угловую на ардуино.

В дальнейшем планируется реализовать построение 2D карты местности для того, чтобы робот мог определять свое местоположение относительно соревновательного поля.

# 10. Алгоритм действий робота для финального этапа соревнований

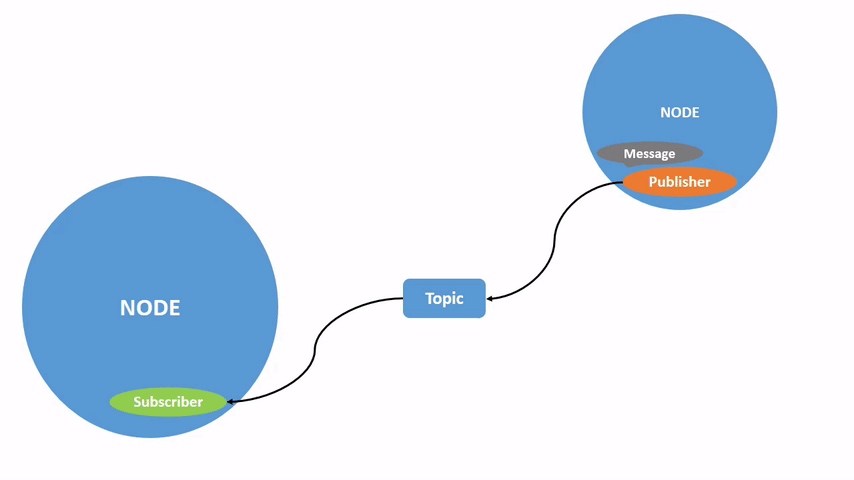
Для прохождения финального этапа соревнований макету робота помимо распознавания бочек и их цвета необходимо обрабатывать данные с датчика, лидара, изменять свою скорость, управлять схватом. Помимо этого, предусмотрена возможность отправки аудиозаписей на Bluetooth колонку для осуществления отладки без необходимости удаленного подключения к одноплатному компьютеру Raspberry Pi. Для осуществления этих задач так же подойдет фреймворк ROS2.

Рис. 27 Схема элементов publisher и subscriber в ROS2

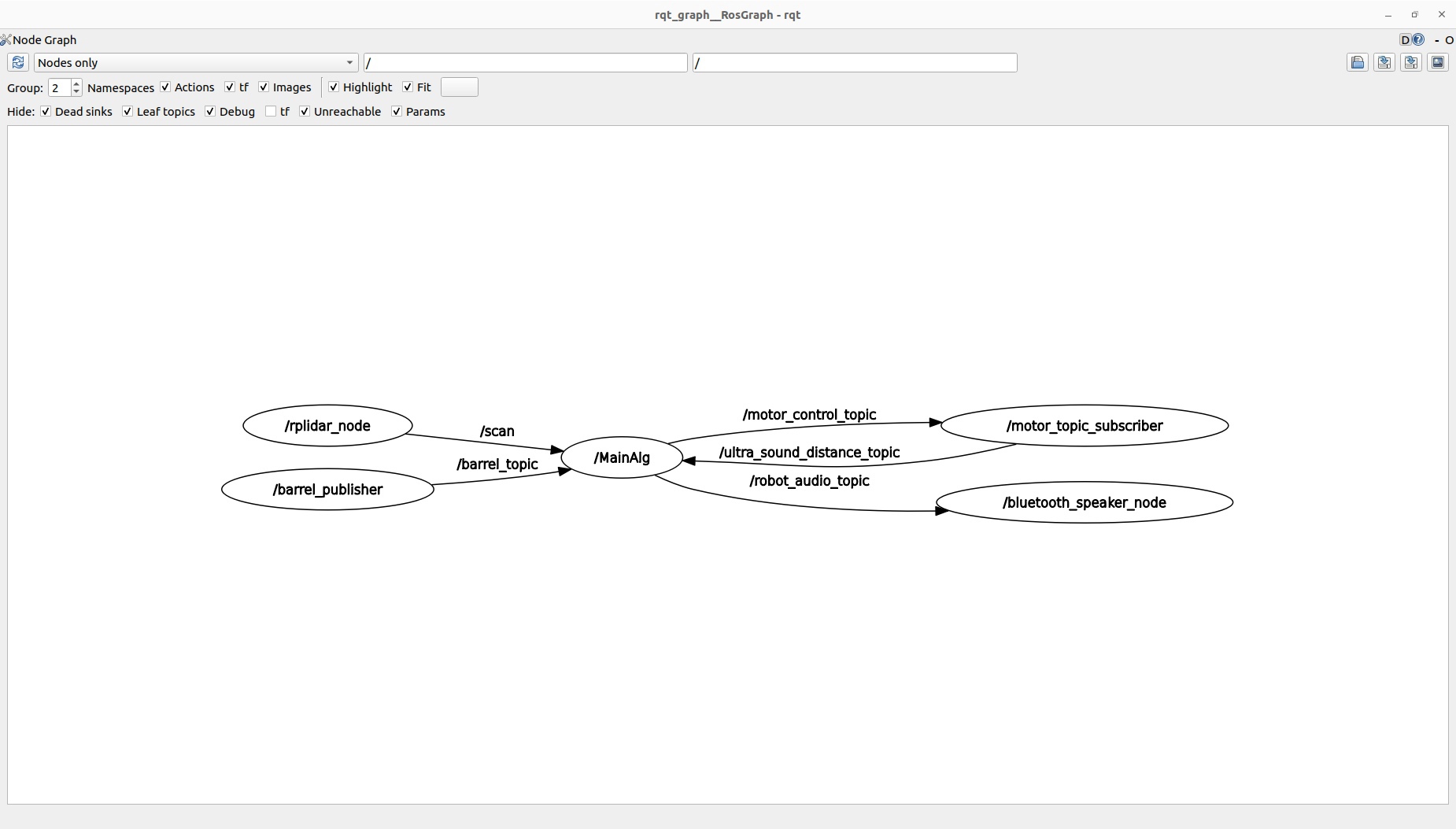
В ROS 2 (Robot Operating System 2) publisher и subscriber (Рис. 27) - это ключевые элементы для обмена сообщениями между различными узлами в системе [1]. Publisher (издатель) - это узел, который публикует (отправляет) сообщения определенного типа в определенную тему (topic). Эти сообщения могут содержать данные о состоянии робота, измерениях сенсоров или другой информации, которая должна быть доступна другим узлам в системе. Каждый publisher определяет тип сообщения, который он будет публиковать, и тему, в которую он будет публиковать. Subscriber (подписчик) - это узел, который подписывается на определенную тему и принимает сообщения, опубликованные в эту тему другими узлами. Когда сообщение публикуется в тему, все подписчики, подписанные на эту тему, получают это сообщение для обработки. Таким образом, publisher и subscriber позволяют узлам в ROS 2 обмениваться информацией, такой как данные с датчиков, команды управления и другие сообщения.

Рис. 28 Схема узлов данного проекта в ROS2

В данном проекте основной алгоритм (Main\_Alg) (Рис. 28) подписан на три темы (scan, barrel\_publisher, ultra\_sound\_distance\_topic) и публикует данные еще в две темы (motor\_control\_topic, robot\_audio\_topic). Основной алгоритм постоянно следит за минимальным расстоянием до препятствия, за наличием и цветом бочек, и на основе этого формирует желаемую скорость робота и управляет схватом и отправляет в тему (motor\_control\_topic). Так же в зависимости от состояния макета робота отправляется код аудио команды в тему robot\_audio\_topic.

Узел motor\_topic\_subscriber получает желаемые значения скорости робота и состояния сервомотора схвата и отправляет их на контроллер Arduino Mega через serial (Приложение 4), а так же получает данные с ультразвукового датчика расстояния, которые отправляет в тему ultra\_sound\_distance\_topic.

Узел rplidar\_node отправляет обработанные данные измерений с лидара в тему scan, а barrel\_publisher отправляет данные о цвете, количестве, расположении цветных цилиндров в тему barrel\_topic.

Узел bluetooth\_speaker\_node подписан на тему robot\_audio\_topic и в зависимости от полученного кода воспроизводит заранее записанную аудиозапись.

Таким образом, были решены задачи навигации, захвата бочек, технического зрения.

# Сборка и отладка робота

После составления документации была начата сборка и наладка МРС (Рис. 29), в ходе которых возникли следующие проблемы:

* дефектная пайка производителя плат DC-DC преобразователя и BMS;
* завышенные производителем характеристики DC-DC преобразователя;
* дефектные датчики цвета tcs74325;
* отсутствие подходящих переходных муфт;
* проблемы с совместимостью платы клона Arduino;
* выполнение производителем сервомотора редуктора из пластмассы привело к ее поломке при транспортировке;

Все эти проблемы были решены. Так были заново пропаяны дефектные платы, в структуру робота включен портативный аккумулятор, были заменены датчики цвета на датчики линии Vex Line Tracker, куплены и модифицированы муфты, установлены драйверы, позволяющие программировать клоны Arduino, был заменен сервомотор.

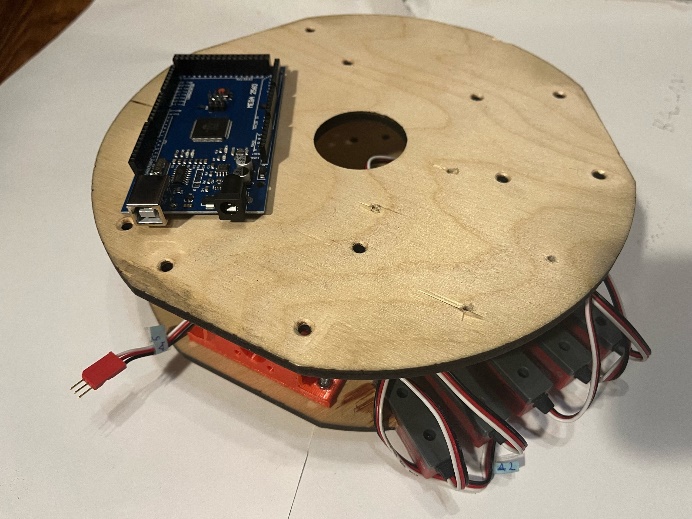
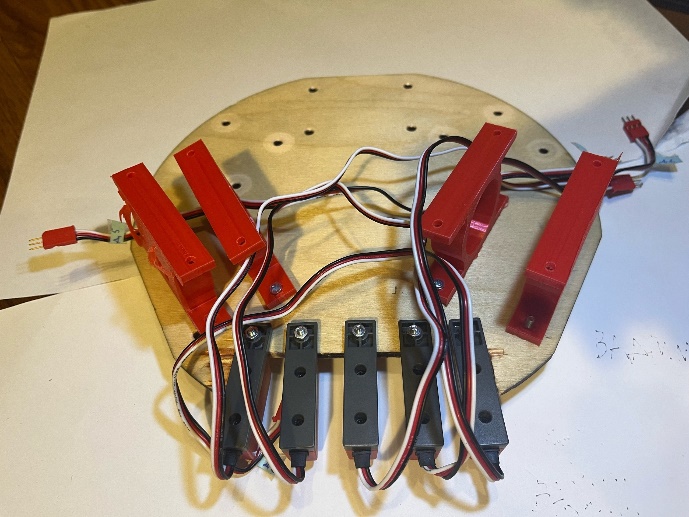




Рис. 29 Робот на разных этапах сборки.

# Выводы

В ходе данной работы был разработан робот для участия в соревнованиях BRC (Приложение 6).

Были исследованы: принципы работы различных датчиков, таких как RP Lidar A1, датчик расстояния, датчик линии; работа с сервоприводами; фреймворк для работы с робототехническими системами ROS 2, библиотека OpenCV для обработки изображений с камеры; среда разработки кода Arduino IDE для загрузки кода в микроконтроллер. Изучен алгоритм работы с 3D принтером, а также лазерным станком.

Реализованы:

* 1. Алгоритм движения с помощью омни - колёс.
  2. Система питания робота.
  3. Алгоритм движения по линии.
  4. Компьютерное зрение для определения цветов объектов с нахождением их центров.
  5. Отъезд от препятствий с помощью RP Lidar A1.

Кроме того, была составлена принципиальная схема соединения компонентов робота, техническое задание.

В ближайшем будущем планируется реализовать:

1. Способность робота создавать карту местности с помощью RP Lidar A1 для ориентирования его в полигоне.
2. Приоритетный захват бочек разных цветов роботом (в зависимости от цветов) и доставление их в зоны того же цвета.

Работая над курсовым проектом, мы смогли прочувствовать многие тонкости робототехники, столкнувшись с трудностями, такими как: проектирование робота, реализация кода для его правильного функционирования и т.д.. Мы получили незаменимый инженерный опыт, который однозначно пригодится нам в будущем.

# Список литературы

1. ROS 2 Documentation / [Электронный ресурс] // ROS2 : [сайт]. — URL: [https://docs.ros.org/en/humble/index.html](https://docs.ros.org/en/humble/index.html%20)
2. Grégoire Passault Kinematics of omni-directional wheeled robots / Grégoire Passault [Электронный ресурс] // github.io : [сайт]. — URL: <https://gregwar.github.io/omnidirectional-wheeled-robots>
3. А.Н. Афонин, А.Ю. Алейников, Е.Н. Бондарева УПРОЩЕННЫЙ РАСЧЕТ КИНЕМАТИКИ ДВИЖЕНИЯ МОБИЛЬНОГО РОБОТА С ТРЕМЯ ОМНИ - КОЛЕСАМИ // Научные ведомости. — 2014. — № 19 (190) Вып. 36.
4. Oliveira, Helder & Sousa, Armando & Moreira, A. & Costa, Paulo (2008). Dynamical Models for Omni-directional Robots with 3 and 4 Wheels. ICINCO 2008 - Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics. 1. DOI:[10.5220/0001489201890196](https://doi.org/10.5220/0001489201890196)
5. Indrazno Siradjuddin, Aang Junaidi, Ratna Ika Putri, Erfan Rohadi, Supriatna Adhisuwignjo (2019). Kinematics and Control A Three Wheeled Omni -directional Mobile Robot // SSRG International Journal of Electrical and Electronics Engineering ( SSRG - IJEEE ). — Dec 2019. — № Volume 6 Issue 12. DOI:[10.14445/23488379/IJEEE-V6I12P101](https://doi.org/10.14445/23488379/IJEEE-V6I12P101)
6. Ribeiro, Fernando & Moutinho, Ivo & Silva, Pedro & Fraga, Carlos & Pereira, Nino. (2004). Three omni-directional wheels control on a mobile robot.
7. Hou, Linfei & Zhang, Liang & Kim, Jongwon. (2019). Energy Modeling and Power Measurement for Three-Wheeled Omni-directional Mobile Robots for Path Planning. IFAC-PapersOnLine, Volume 49, Issue 15, 2016, Pages 32-37. DOI:[10.1016/j.ifacol.2016.07.610](https://doi.org/10.1016/j.ifacol.2016.07.610)
8. Лаврентьев Е. Б., Ивацевич Ю. Б., Лукьянов Е. А. Расчет и конструирование захватных устройств автоматических манипуляторов и промышленных роботов / Лаврентьев Е. Б., Ивацевич Ю. Б., Лукьянов Е. А. [Электронный ресурс] // de.donstu.ru : [сайт]. – URL:<https://de.donstu.ru/CDOCourses/structure/_new_/62239/4835/5752.pdf>
9. Open Source Computer Vision / [Электронный ресурс] // OpenCV : [сайт]. — URL: <https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html>
10. Patent US 1305535 A. Vehicle wheel / J. Grabowiecki. 1919.

# Приложение 1

**Регламент квалификационного этапа ежегодных открытых соревнований по робототехнике BRC**



**Содержание**

Общие положения 3

Требования к участникам 4

Требования к роботу4

Обучение5

Легенда соревнований5

Программа соревнований6

Элементы соревновательного поля 7

Порядок проведения соревнований 7

Баллы и пенализация 8

Дисквалификация10

Судейство10

Приложение 111

**Общие положения**

1. Данный регламент является основным документом соревнований. Регламент может дополняться и уточняться организаторами во время проведения соревнований. Решения организаторов не подлежат оспариванию и имеют большую силу, чем регламент. За несоблюдение регламента или решений организаторов участники могут быть дисквалифицированы.
2. Организатором соревнований является BRC (Bauman Robotics Club) - коллектив студентов факультета СМ (Специальное машиностроение) и факультета РК (Робототехника и комплексная автоматизация).
3. Основными задачами соревнований являются:

- совершенствование навыков в области робототехники у участников,

- популяризация робототехники среди студентов.

1. Подавая заявку на участие в робототехнических соревнованиях, участники соглашаются с данным регламентом и дают согласие на обработку своих персональных данных организаторами соревнований.
2. По желанию участников им могут быть предоставлены наборы, содержащие минимум деталей и устройств, необходимый для сборки робота.
3. Дополнительную информацию о данных соревнованиях можно получить в официальной группе ВК <https://vk.com/robot.bmstu>, а также связавшись с организаторами, контакты которых находятся в данной группе.
4. Регистрация команды производится до 1 ноября 2023 года через специальную форму, которая будет опубликована в группе.

**Требования к участникам**

1. Участники должны строго соблюдать регламент соревнований.
2. К соревнованиям допускаются студенты без ограничения на возраст или курс любых ВУЗов.
3. Студенты из сторонних вузов могут согласовать своё участие с организаторами.
4. К участию допускаются команды в составе от 1 до 5 человек включительно, заявка которых одобрена организаторами, а робот, соответствует регламенту.
5. Каждый участник соревнований может представлять лишь 1 команду.
6. Каждая команда может представить только одного автономного робота.

**Требования к роботу**

1. Робот должен быть полностью автономным. Запрещено любое взаимодействие с роботом во время выполнения задания
2. Робот может быть спроектирован на любой аппаратной платформе. Все модули должны располагаться на борту робота.
3. Габаритные размеры робота при старте не должны превышать – 300 мм в длину, 300 мм в ширину, без ограничения на высоту.
4. Робот не должен причинять ущерб соревновательному полю, другим роботам участникам и организаторам соревнований.
5. Команды могут использовать готовых роботов, при условии, что они удовлетворяют вышеуказанным требованиям.

**Обучение**

1. Для подготовки к соревнованиям организуются образовательные курсы для участников соревнований на территории МГТУ им. Н.Э. Баумана.
2. Несмотря на то, что робот может быть спроектирован на любой платформе, организаторами предлагаются курсы по проектированию робота, использующего Arduino и/или Raspberry Pi.
3. В рамках курсов будут рассказаны основы, необходимые для сборки, программирования и управления автономного мобильного робота.
4. Программа и время проведения курсов будут выложены в официальной группе в ВК. Информация о проведении того или иного обучающего мероприятия будет выложена не менее чем за неделю до его начала. Организаторы оставляют за собой право отменить или перенести мероприятие.

**Легенда соревнований**

Ваш робот был отправлен на заброшенный завод для исследования условий и возможностей его восстановления или использования в будущем.

Время, когда этот завод был процветающим предприятием, давно позади. Здания завода остались запущенными и разрушенными, а его территория превратилась в лабиринт из различных цехов и коридоров. Однако, некоторые части заброшенного завода все еще сохранились и могут быть использованы в качестве пути для робота.

В какой-то момент всё пошло не по плану, и произошёл сбой в системе связи робота. Вернуться по дистанционному управлению не представляется возможным, но, к счастью, ваш робот обладает автопилотом, который может двигаться к точке старта по ориентирам.

Задача для робота состоит в том, чтобы выбраться с завода, следуя по частично стертой и разрушенной линии, которая была нарисована на полу завода много лет назад. Эта линия является единственным ориентиром для робота, поскольку все другие указатели и ориентиры были утрачены со временем.

Робот должен быть гибким и адаптивным, чтобы справиться с частично стертыми линиями, неровностями пола и разрушенными структурами на своем пути. Каждый цех представляет собой уникальное испытание для роботов. Они должны преодолеть узкие проходы, подняться по лестницам, обойти разрушенные стены и преодолеть другие преграды.

**Программа соревнований**

1. За каждый из участков трассы предусмотрено как начисление баллов, так и их снятие. По результатам команда может набрать отрицательное количество баллов.
2. Основная задача: автономному роботу необходимо преодолеть трассу по заданной траектории движения, обозначенной направляющими элементами (линиями, пунктиром, и т.д.).
3. Поле разделено на 4 области по уровню сложности преодолеваемого участка.
4. В первой области находятся: место старта; прямой и криволинейный участок траектории, обозначенный сплошной линией черного цвета.
5. Во второй зоне находятся: прямой участок траектории, обозначенный прерывистой линией черного цвета, а также кругами различного диаметра.
6. Третья зона черного цвета с траекторией, обозначенной линией белого цвета (инверсия).
7. В четвёртой зоне находятся: прямолинейный участок без направляющей линии; ломаный участок траектории с резкими поворотами и перегибами, обозначенный сплошной линией черного цвета.
8. Для завершения маршрута, роботу необходимо всеми проекциями оказаться в зоне финиша.

**Элементы соревновательного поля**

1. Поле собрано из белых матовых листов ЛДСП толщиной 16 мм. Его размеры указаны на чертеже. Высота стенок – 250 мм.
2. Уровни сложности задаются с помощью разделения поля на зоны, выделенные определенным цветом (см. Приложение 1). В качестве направляющих элементов служит линия черного цвета шириной 15 мм (или других цветов).

**Порядок проведения соревнований**

1. Каждой команде дается 2 попытки. Вторая попытка проводится после того, как все участники совершили первую попытку и дополнительного времени на подготовку (на подготовку дается до 40 мин).
2. Команда устанавливает робота в зоне старта и запускает его по сигналу судьи, одновременно с которым начинается отсчет времени. Робот может быть запущен как дистанционно, так и непосредственно вручную, однако после этого запрещается любое взаимодействие команды с роботом.
3. Длительность одной попытки - 10 минут. Команды выступают по очередности, определенной организаторами.
4. Попытка может быть завершена досрочно:

* Организаторами, при нарушении командой Регламента.
* При поломке или бездействии робота в течении 30 секунд.
* Капитаном команды с помощью слова «Стоп!».

1. В случае бездействия робота, команде засчитывается максимальное время – 10 мин. и баллы, которые набрал робот.
2. В случае остановки робота во время заезда капитаном командой «Стоп!», для получения баллов, необходимо, чтобы робот оставался неподвижен.
3. Если элемент программы был пройден не полностью, за него баллы не начисляются и предусмотрен штраф.

**Баллы и пенализация**

|  |  |
| --- | --- |
| Выезд робота всеми проекциями со старта | +1 |
| Прохождение прямолинейного участка первой зоны | +1 |
| Прохождение криволинейного участка первой зоны | +2 |
| Прохождение первого элемента (прерывистой прямой линии) второй зоны | +6 |
| Прохождение второго элемента (круглых элементов) второй зоны | +7 |
| Прохождение третьей зоны | +6 |
| Прохождение первого элемента (участка без линии) четвертой зоны | +6 |
| Прохождение второго элемента (ломаной кривой) четвертой зоны | +11 |
| Остановка на финише всеми проекциями робота | +10 |
| Робот перешел из одной зоны в другую не по маршруту, игнорируя направляющие линии или элементы | -20 |
| Съезд робота с траектории и последующая его остановка на первой зоне | -10 |
| Съезд робота с траектории и последующая его остановка на второй зоне | -5 |
| Съезд робота с траектории и последующая его остановка на третьей зоне | -3 |
| Съезд робота с траектории и последующая его остановка на четвёртой зоне | -2 |
| Робот полностью не выполнил элемент (срезал, не доехал и т.д.) | -2 за каждое нарушение |
| В конструкции робота было использовано интересное техническое решение | До + 5 |

1. После завершения командой своей попытки организаторами выполняется подсчет баллов, набранных командой в заезде.
2. Победителем соревнований назначается команда, получившая максимальный балл за обе попытки.
3. Если несколько команд набрали одинаковое количество баллов, то выигравшей считается та, которая сделала это за наименьшее время.

**Дисквалификация**

Команда будет дисквалифицирована, если:

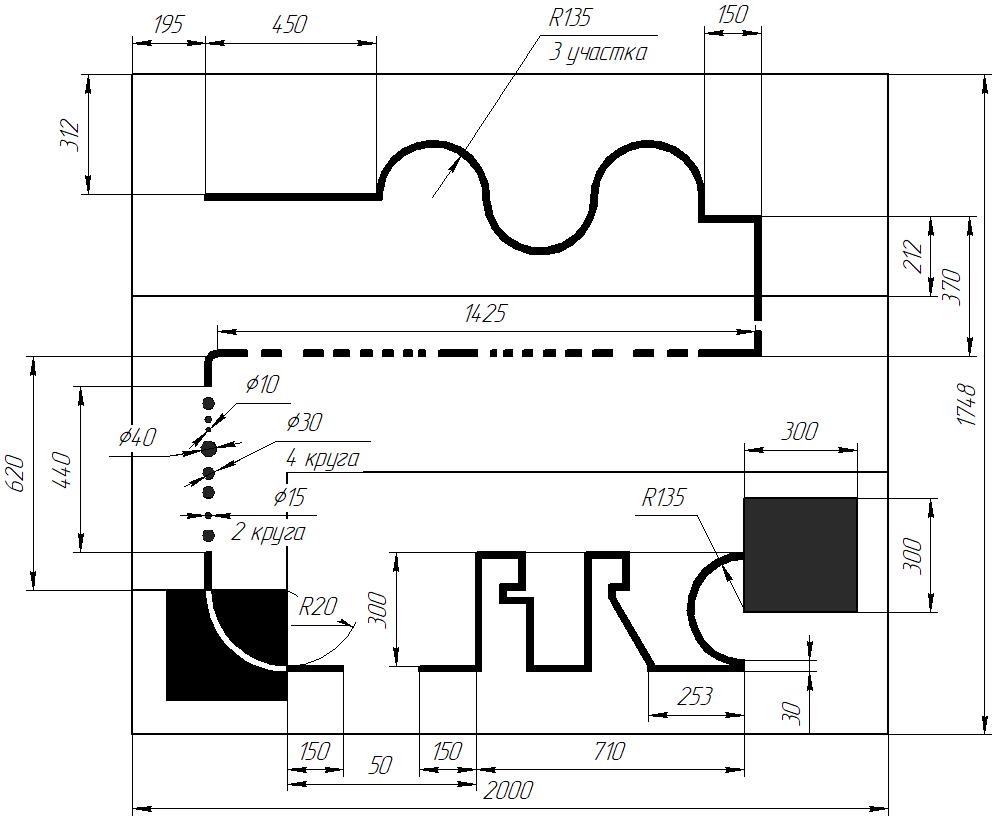
1. Команда не соблюдает требования организаторов и/ или график соревнований.
2. Команда допускает неуважительное поведение и оскорбления в адрес других участников и организаторов соревнований.
3. Робот портит покрытие пола, вредит участникам, организаторам и т.д.
4. Команда взаимодействовала с роботом, во время заезда.

**Судейство**

1. Запрещается вмешательство в действия судьи.
2. По результату каждого заезда судьями оформляется протокол заезда. Баллы начисляются команде после подписания протокола капитаном команды.
3. После подписи протокола отсутствует возможность апелляции.
4. Решение судьи окончательно и не может быть изменено.

**Приложение 1**

Соревновательное поле



# Приложение 2

**Регламент заключительного этапа ежегодных открытых соревнований по робототехнике BRC**



**Содержание**

Общие положения 3

Требования к участникам 4

Требования к роботу4

Обучение5

Легенда соревнований5

Программа соревнований6

Элементы соревновательного поля 7

Порядок проведения соревнований 8

Баллы и пенализация 9

Дисквалификация10

Судейство10

Приложение 111

Приложение 214

Приложение 316

Приложение 417

**Общие положения**

1. Данный регламент является основным документом соревнований. Регламент может дополняться и уточняться организаторами во время проведения соревнований. Решения организаторов не подлежат оспариванию и имеют большую силу, чем регламент. За несоблюдение регламента или решений организаторов участники могут быть дисквалифицированы.
2. Организатором соревнований является BRC (Bauman Robotics Club) - коллектив студентов факультета СМ (Специальное машиностроение) и факультета РК (Робототехника и комплексная автоматизация).
3. Основными задачами соревнований являются:

- совершенствование навыков в области робототехники у участников,

- популяризация робототехники среди студентов.

1. Подавая заявку на участие в робототехнических соревнованиях, участники соглашаются с данным регламентом и дают согласие на обработку своих персональных данных организаторами соревнований.
2. По желанию участников им могут быть предоставлены наборы, содержащие минимум деталей и устройств, необходимый для сборки робота.
3. Дополнительную информацию о данных соревнованиях можно получить в официальной группе ВК <https://vk.com/robot.bmstu>, а также связавшись с организаторами, контакты которых находятся в данной группе.
4. Регистрация команды производится до 1 ноября 2023 года через специальную форму, которая будет опубликована в группе.

**Требования к участникам**

1. Участники должны строго соблюдать регламент соревнований.
2. К соревнованиям допускаются студенты без ограничения на возраст или курс любых ВУЗов.
3. Студенты из сторонних вузов могут согласовать своё участие с организаторами.
4. К участию допускаются команды в составе от 1 до 5 человек включительно, заявка которых одобрена организаторами, а робот, соответствует регламенту.
5. Каждый участник соревнований может представлять лишь 1 команду.
6. Каждая команда может представить только одного автономного робота.

**Требования к роботу**

1. Робот должен быть полностью автономным. Запрещено любое взаимодействие с роботом во время выполнения задания
2. Робот может быть спроектирован на любой аппаратной платформе.
3. Габаритные размеры робота при старте не должны превышать – 300 мм в длину, 300 мм в ширину, без ограничения на высоту.
4. Робот не должен причинять ущерб соревновательному полю, другим роботам участникам и организаторам соревнований.
5. Команды могут использовать готовых роботов, при условии, что они удовлетворяют вышеуказанным требованиям.

**Обучение**

1. Для подготовки к соревнованиям организуются образовательные курсы для участников соревнований на территории МГТУ им. Н.Э. Баумана.
2. Несмотря на то, что робот может быть спроектирован на любой платформе, организаторами предлагаются курсы по проектированию робота, использующего Arduino и/или Raspberry Pi.
3. В рамках курсов будут рассказаны основы, необходимые для сборки, программирования и управления автономного мобильного робота.
4. Программа и время проведения курсов будут выложены в официальной группе в ВК. Информация о проведении того или иного обучающего мероприятия будет выложена не менее чем за неделю до его начала. Организаторы оставляют за собой право отменить или перенести мероприятие.

**Легенда соревнований**

Ваш робот успешно преодолел 1-ый этап маршрута, следуя по частично стертой и разрушенной линии до склада, в котором хранятся бочки с отходами.

Роботу необходимо расчистить путь к выходу, чтобы отходы не оказались снаружи склада.

Задача передвигать и сортировать бочки с отходами по цветам требует от робота точности, надежности и безопасности. Он должен быть способен эффективно управляться с различными бочками и обеспечивать их безопасное перемещение.

**Программа соревнований**

1. Основная задача: - цветные цилиндры (бочки) диаметром 50 мм и высотой 100 мм (см. Приложение 2) перенести в специально обозначенные области (см. Приложение 1)
2. Предусмотрено как начисление баллов, так и их снятие. По результатам команда может набрать отрицательное количество баллов.
3. Поле разделено на зону старта, зону сбора бочек и зону выгрузки.
4. Зона сбора бочек разделена на две области: в первой области расположение бочек известно, во второй бочки расположены случайным образом (см. Приложение 1).
5. Бочки любого цвета запрещено опрокидывать, за это предусмотрен штраф.
6. Во второй области дополнительно с цветными находятся бочки черного цвета (см. Приложение 2). Эти бочки запрещено вывозить из данной области.
7. Зона выгрузки состоит из трёх областей, обозначенных красным, зелёным и синим цветом, в которые необходимо доставить бочки соответствующего цвета. Каждая область дополнительно отмечена фидуциальным (AruCo) маркером (см. Приложение 4). Порядок расположения областей определяется случайным образом.
8. В левом верхнем углу поля (см. Приложение 3) расположена труба – половина цилиндра черного цвета диаметром 50 мм и высотой 250 мм. Запрещено перемещать или опрокидывать трубу.
9. Бочка считается доставленной, если по окончании попытки она всеми проекциями находится в зоне соответствующего цвета и расположена вертикально.
10. Для завершения попытки, роботу необходимо вернуться в зону старта и всеми проекциями оказаться в этой зоне.

**Элементы соревновательного поля**

1. Поле собрано из белых матовых листов ЛДСП толщиной 16 мм. Его размеры указаны на чертеже. Высота стенок – 300 мм.
2. Границы областей, зоны старта и выгрузки во второй области отмечены по периметру черной линией шириной 15 мм так, что размеры по внутренним границам ленты полностью совпадают с чертежом.
3. Бочки представляют собой цилиндры различных цветов диаметром 50 мм, высотой 100 мм, массой 50 г, напечатанные на 3D принтере. На верхней грани бочек расположен фидуциальный (AruCo) маркер, соответствующий цвету бочки (см. Приложение 2).
4. Труба - половина цилиндра черного цвета диаметром 50 мм и высотой 250 мм, напечатанная на 3D принтере (см. Приложение 3).
5. Все случайные элементы будут выбраны непосредственно перед попыткой в присутствии капитанов команд.

**Порядок проведения соревнований**

1. Каждой команде дается 2 попытки. Вторая попытка проводится после того, как все участники совершили первую попытку и дополнительного времени на подготовку (на подготовку дается до 40 мин).
2. Команда устанавливает робота в зоне старта, после чего определяется расположение случайных элементов.
3. Команда запускает робота по сигналу судьи, одновременно с которым начинается отсчет времени. Робот может быть запущен как дистанционно, так и непосредственно вручную, однако после этого запрещается любое взаимодействие команды с роботом.
4. Длительность одной попытки - 10 минут. Команды выступают по очередности, определенной организаторами.
5. Попытка может быть завершена досрочно:

* Организаторами, при нарушении командой Регламента.
* При поломке или бездействии робота в течении 30 секунд.
* Капитаном команды с помощью слова «Стоп!».

1. В случае бездействия робота, команде засчитывается максимальное время – 10 мин. и баллы, которые набрал робот.
2. В случае остановки робота во время заезда капитаном командой «Стоп!», для получения баллов, необходимо, чтобы робот оставался неподвижен.

**Баллы и пенализация**

|  |  |
| --- | --- |
| Выезд робота всеми проекциями со старта | +1 |
| Доставка бочки в соответствую зону | +20 |
| Вынос цветной бочки из зоны сбора | +5 |
| Остановка робота на финише | +5 |
| Опрокидывание цветной бочки | -10 |
| Опрокидывание черной бочки | -20 |
| Перемещение или опрокидывание трубы | -5 |
| Вывоз чёрной бочки из зоны сбора | -10 |
| Доставка цветной бочки в несоответствующую зону | -10 |
| После окончания попытки цветная бочка осталась в зоне сбора | -5 за каждую бочку |
| Остановка на финише всеми проекциями робота | +10 |
| В конструкции робота было использовано интересное техническое решение | До + 5 |

1. После завершения командой своей попытки организаторами выполняется подсчет баллов, набранных командой в заезде.
2. Победителем соревнований назначается команда, получившая максимальный балл за обе попытки.
3. Если несколько команд набрали одинаковое количество баллов, то выигравшей считается та, которая сделала это за наименьшее время.

**Дисквалификация**

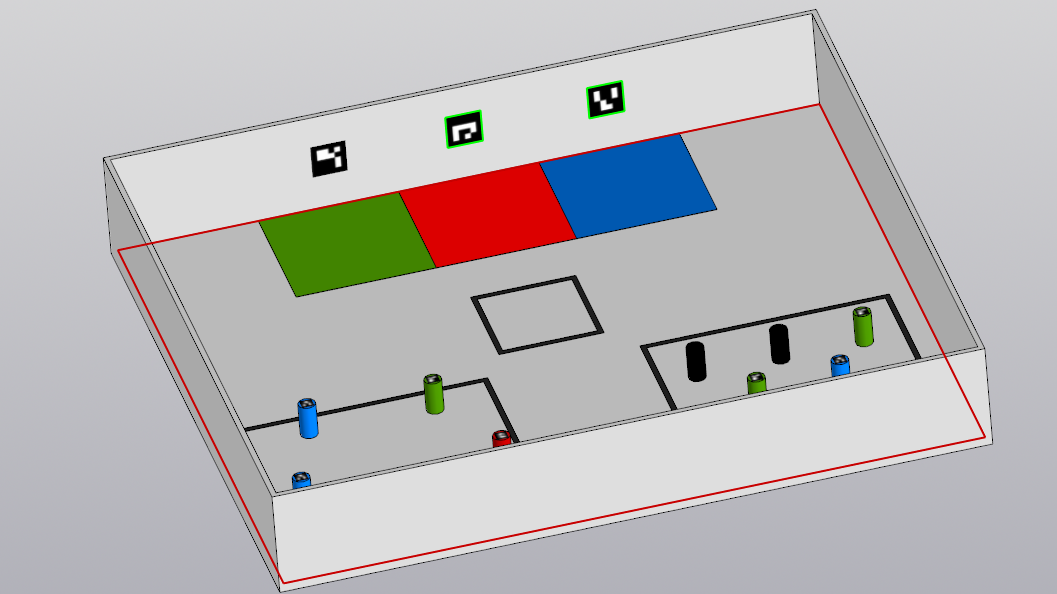
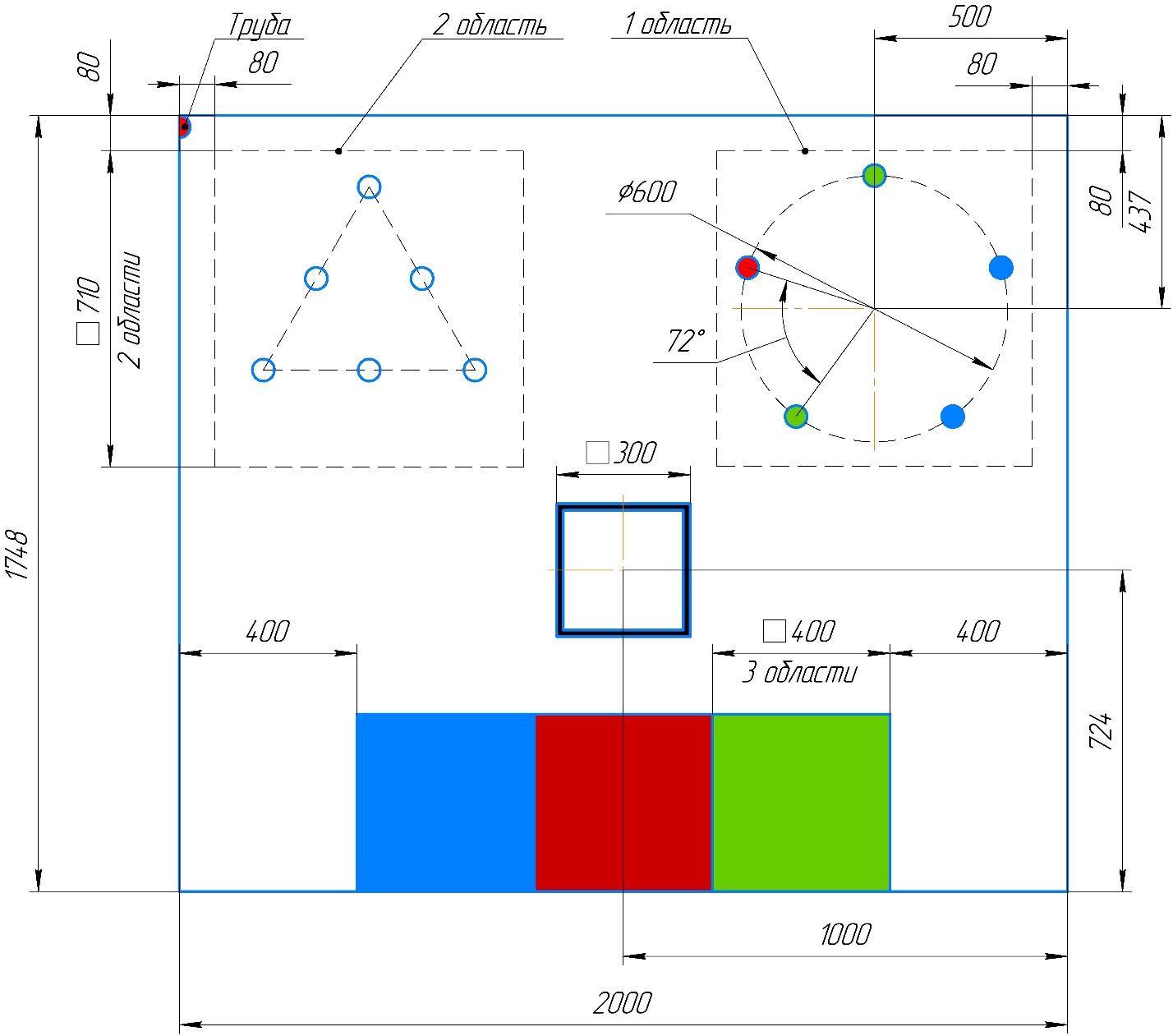
Команда будет дисквалифицирована, если:

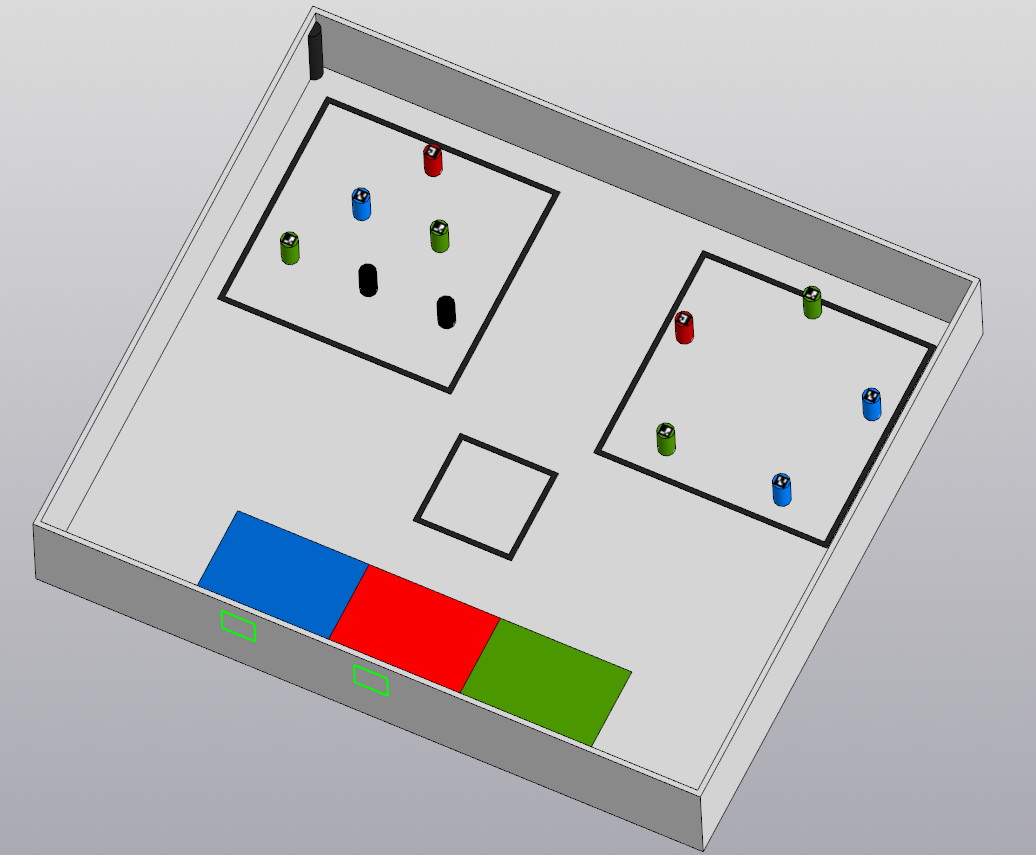
1. Команда не соблюдает требования организаторов и/ или график соревнований.
2. Команда допускает неуважительное поведение и оскорбления в адрес других участников и организаторов соревнований.
3. Робот портит покрытие пола, вредит участникам, организаторам и т.д.
4. Команда взаимодействовала с роботом, во время заезда.

**Судейство**

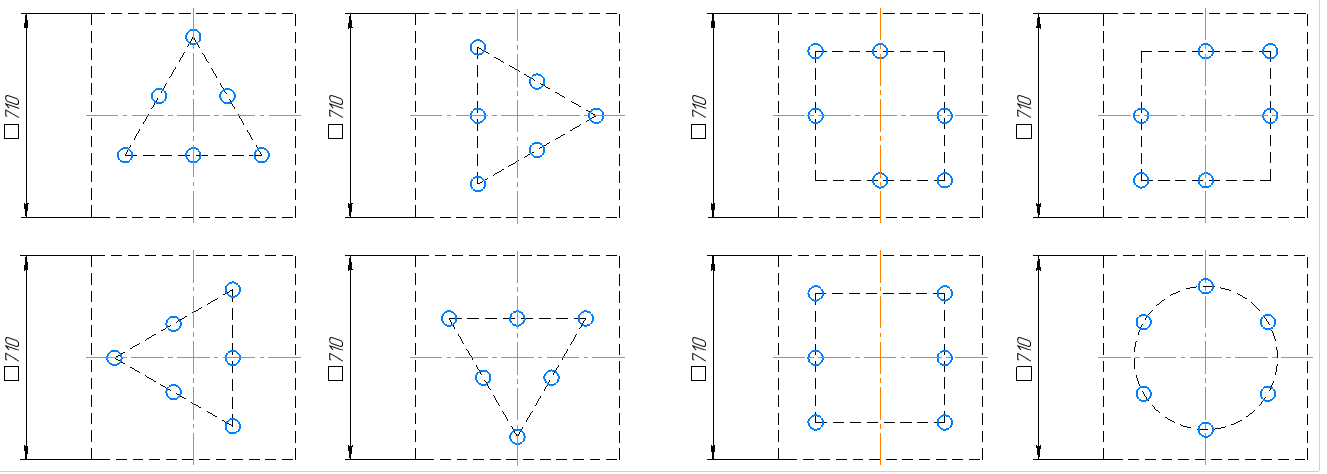
1. Запрещается вмешательство в действия судьи.
2. По результату каждого заезда судьями оформляется протокол заезда. Баллы начисляются команде после подписания протокола капитаном команды.
3. После подписи протокола отсутствует возможность апелляции.
4. Решение судьи окончательно и не может быть изменено.

**Приложение 1**

Соревновательное поле

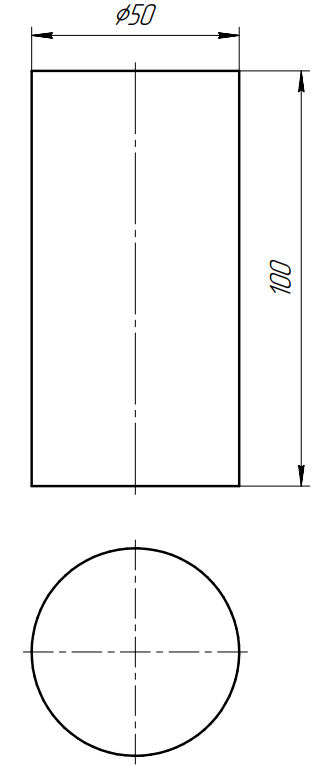


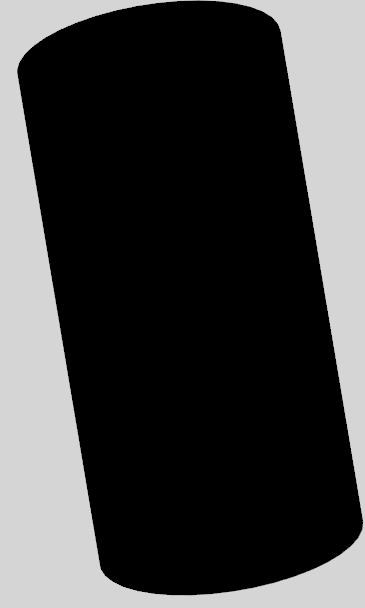
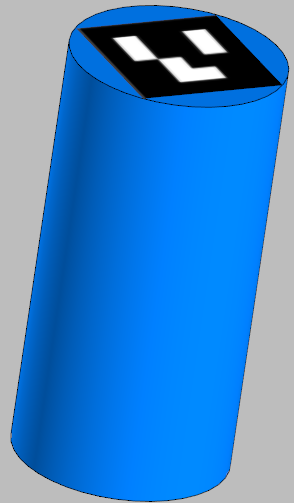
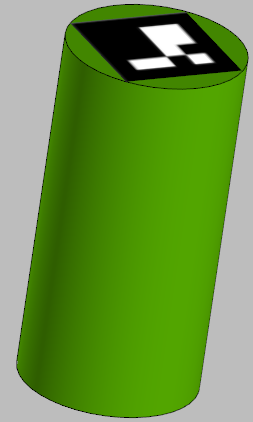
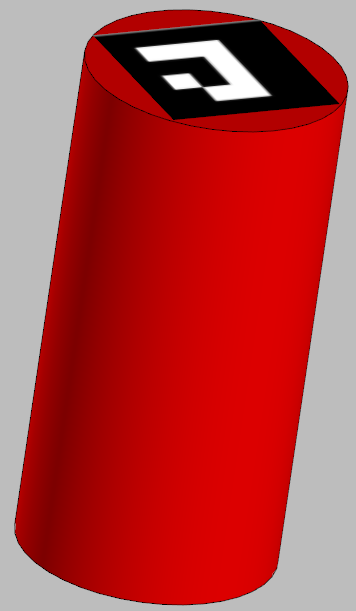
Примеры расположения бочек во второй зоне сбора

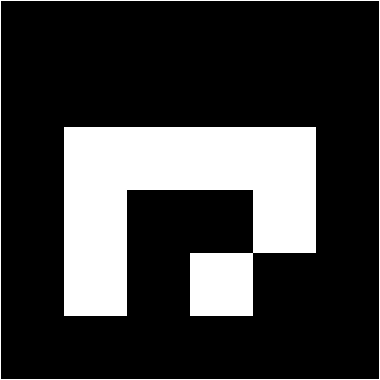
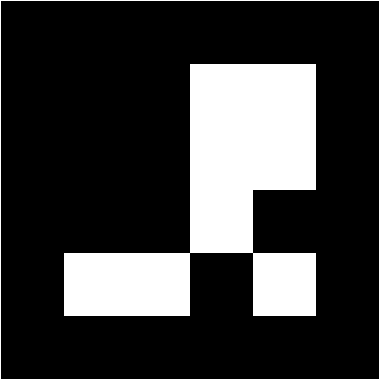
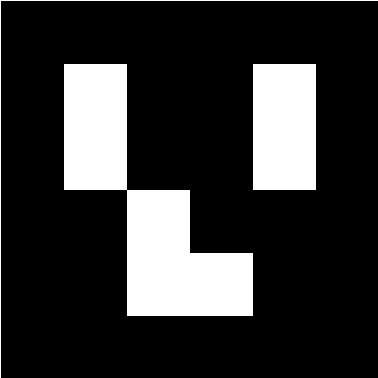


**Приложение 2**

Бочки



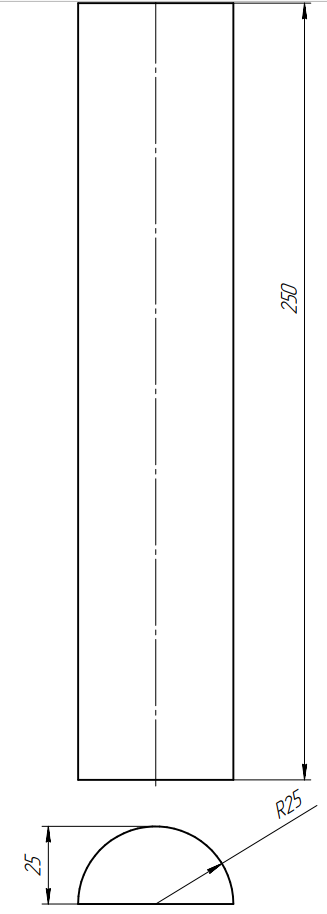
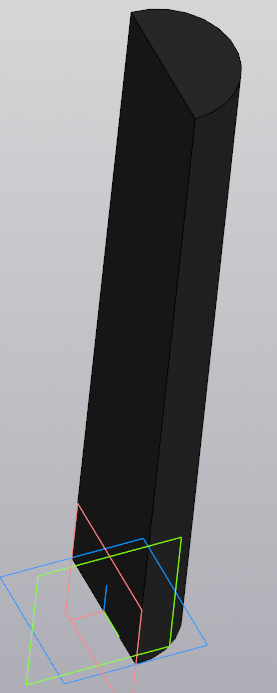


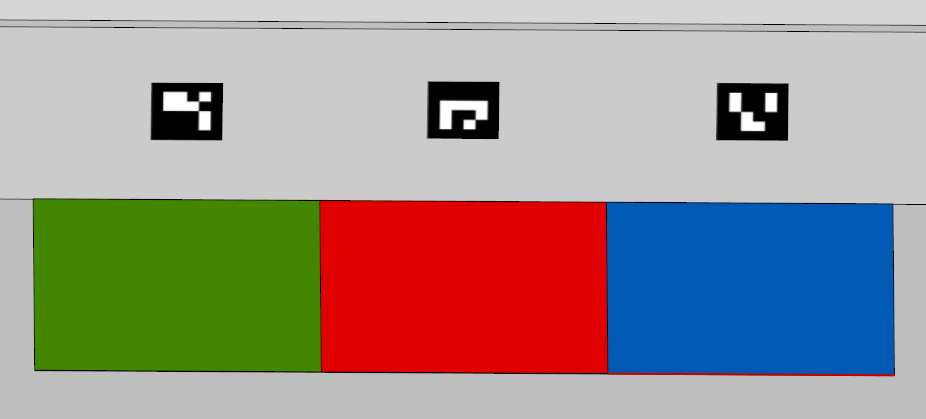
Фидуциальные (AruCo) маркеры, означающие цвет бочки, слева направо: красный, зеленый, синий

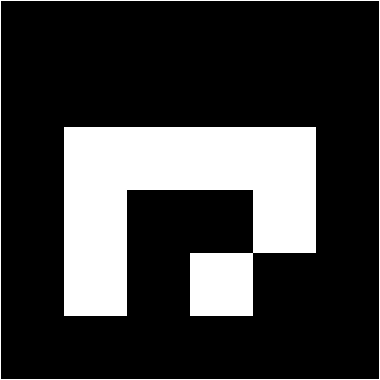
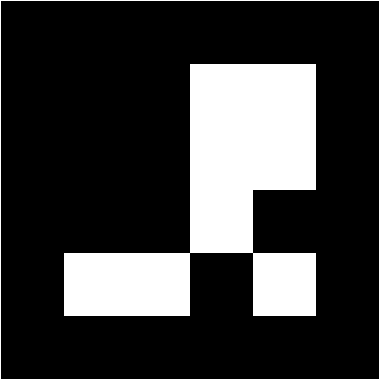
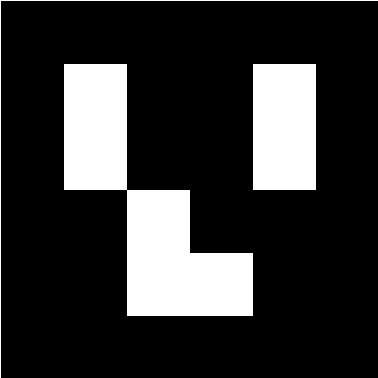
**Приложение 3**

Труба

**Приложение 4**

Зона выгрузки 

Фидуциальные (AruCo) маркеры, означающие цвет зоны выгрузки, слева направо: красный, зеленый, синий

# Приложение 3

#include <GyverPID.h>

#include <PIDtuner.h>

#include <PIDtuner2.h>

#include "AccelMotor.h"

//#include <Adafruit\_TCS34725softi2c.h>

#include <Wire.h>

//#include "Adafruit\_TCS34725.h"

#define SDApin\_1 22

#define SCLpin\_1 23

#define SDApin\_2 24

#define SCLpin\_2 25

#define PIN\_EN\_MotorA 4 // Вывод управления скоростью вращения мотора №1

#define PIN\_EN\_MotorB 9 // Вывод управления скоростью вращения мотора №2

#define PIN\_EN\_MotorC 10 // Вывод управления скоростью вращения мотора №3

#define PIN\_IN1\_MotorA 5 // Вывод управления направлением вращения мотора №1

#define PIN\_IN2\_MotorA 6 // Вывод управления направлением вращения мотора №1

#define PIN\_IN1\_MotorB 8 // Вывод управления направлением вращения мотора №2

#define PIN\_IN2\_MotorB 7 // Вывод управления направлением вращения мотора №2

#define PIN\_IN1\_MotorC 12 // Вывод управления направлением вращения мотора №3

#define PIN\_IN2\_MotorC 11 // Вывод управления направлением вращения мотора №3

#define encoder1MotorA 18 //Вход энкодера мотора №1

#define encoder1MotorB 19 //Вход энкодера мотора №2

#define encoder1MotorC 20 //Вход энкодера мотора №3

#define encoder2MotorA 14 //Вход энкодера мотора №1

#define encoder2MotorB 15 //Вход энкодера мотора №2

#define encoder2MotorC 16 //Вход энкодера мотора №3

bool flag = 0;

int cycle;

int sensor1Pin = A0;

int sensor2Pin = A1;

int sensor3Pin = A2;

int sensor1Value;

int sensor2Value;

int sensor3Value;

unsigned long timing = 0;

unsigned long tmpTime = 0;

volatile long int encoderPosA = 0;

volatile long int encoderPosB = 0;

volatile long int encoderPosC = 0;

int prev\_line = 0;

// значения параметров робота

float theta = 330/57.29578; // угол поворота робота отно глобальных координат

float x\_l = 0.0; // Скорость по Х

float y\_l = 0.0; // Скорость по

float ang\_vel = 0.0;

float R = 0.135; // Расстояние от центра робота до центра колеса

float r = 0.058; // Радиус колес

float alpha1 = 0.0; // Угол наклона омни - колеса 1 (в радианах)

float alpha2 = 2.09439510239; // Угол наклона омни - колеса 2 (в радианах)

float alpha3 = 4.18879020479; // Угол наклона омни - колеса 3 (в радианах)

float theta1, theta2, theta3;

int circle;

int minValues[5], maxValues[5], threshold[5], d\_values[5];

int P, D, I, previousError, PIDvalue, error;

int lfspeed = 200;

int sp;

int k;

float Kp = 0;

float Kd = 0;

float Ki = 0;

int arr [14] {0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0};

int prev\_line\_mid = -1;

int prev\_line\_side = -1;

float max\_sp = 1;

int count\_black;

bool bl\_or\_white;

bool flazhok = 0;

int tmpTime1;

bool flag1 = 0;

bool flag2 = 0;

float ugl=4;

int tmpTime2;

int state = 1;

int tmpTime5;

int flag5=0;

int timeout = 0;

int tmpTime6;

int flag6 = 0;

AccelMotor motorA(DRIVER3WIRE, PIN\_IN1\_MotorA, PIN\_IN2\_MotorA, PIN\_EN\_MotorA, LOW);

AccelMotor motorB(DRIVER3WIRE, PIN\_IN1\_MotorB, PIN\_IN2\_MotorB, PIN\_EN\_MotorB, LOW);

AccelMotor motorC(DRIVER3WIRE, PIN\_IN1\_MotorC, PIN\_IN2\_MotorC, PIN\_EN\_MotorC, LOW);

void setup() {

// энкодер 11 тиков на оборот

motorA.setRatio(56 \* 11);

motorB.setRatio(56 \* 11);

motorC.setRatio(56 \* 11);

// период интегрирования (по умолч. 20)

motorA.setDt(5); // миллисекунды

motorB.setDt(5);

motorC.setDt(5);

// минимальный (по модулю) ШИМ сигнал (при котором мотор трогается)

motorA.setMinDuty(5);

motorB.setMinDuty(5);

motorC.setMinDuty(5);

// коэффициенты ПИД регулятора

motorA.kp = 0.5;

motorB.kp = 0.5;

motorC.kp = 0.5; // отвечает за резкость регулирования.

// При малых значениях сигнала вообще не будет, при слишком больших – будет трясти

motorA.ki = 3.0;

motorB.ki = 3.0;

motorC.ki = 3.0;// отвечает за коррекцию ошибки в течение времени

motorA.kd = 0.009;

motorB.kd = 0.009;

motorC.kd = 0.009;// отвечает за компенсацию резких изменений

motorA.setRunMode(PID\_SPEED);

motorB.setRunMode(PID\_SPEED);

motorC.setRunMode(PID\_SPEED);

// PID\_SPEED - tick() работает в режиме поддержания скорости по ПИД регулятору

attachInterrupt(digitalPinToInterrupt(encoder1MotorA), updateEncoderA, RISING);// Прерывания для энкодеров

attachInterrupt(digitalPinToInterrupt(encoder1MotorB), updateEncoderB, RISING);

attachInterrupt(digitalPinToInterrupt(encoder1MotorC), updateEncoderC, RISING);

Serial.begin(9600);

}

void loop(){

for ( int i = 0; i < 5; i++)

{

Serial.println(analogRead(i));

}

if (k == 0){

calibration();

state = 1;

}

else {

switch(state){

case 1:

normal\_drive(2);

if (counter() > 4){

state = 2;

}

break;

case 2:

inv\_drive();

if (flazhok == 1){

state = 3;

break;

}

case 3:

tmpTime2 = millis();

normal\_drive(0);

if (millis() - tmpTime2 > 2000){

state = 4;

break;

}

case 4:

normal\_drive(0);

if (counter()>2){

state = 5;

break;

}

case 5:

BRC\_drive();

//normal\_drive(4);

// timeout = timer();

// if (timeout == 1){

// state = 6;

// break;

x\_l=0;

y\_l=0;

ang\_vel=0;

//}

// case 6:

// ang\_vel = 0;

// x\_l = 0.1;

// if (flag6 == 0){

// tmpTime6 = millis();

// flag6 = 1;

// }

// else if (millis() - tmpTime6 > 2250 ){

// flag6 = 0;

// flag5 = 0;

// state = 5;

// break;

// }

}

}

// Serial.println(ugl);

// if (k == 0){

// for (int i = 0; i < 5; i++){

// Serial.println(threshold[i]);

// }

// }

// if (count\_inv == 1){

// max\_sp = -0.1;

// }

// else{

// max\_sp = -0.5;

// }

calculateMotorSpeeds(60/57.29578, x\_l, y\_l, ang\_vel, R, r, alpha1, alpha2, alpha3, theta1, theta2, theta3);

motorA.setTargetSpeedDeg(theta1 \* 57.29578);

motorB.setTargetSpeedDeg(theta2 \* 57.29578);

motorC.setTargetSpeedDeg(theta3 \* 57.29578);

motorA.tick(encoderPosA);

motorB.tick(encoderPosB);

motorC.tick(encoderPosC);

//

// if(ifSquare()) x\_l = 0;

//Serial.println(ang\_vel);

k = 1;

}

int counter(){

int count;

for ( int i = 0; i < 5; i++)

{

if (analogRead(i) > threshold[i]){

count+=1;

}

}

return count;

}

int counter1(){

int count;

for ( int i = 0; i < 5; i++)

{

if (analogRead(i) > 500){

count+=1;

}

}

return count;

}

void BRC\_drive(){

//normal\_drive(2);

if(flag5==0){

int letter\_height=2900;

//move\_fw(5500);

move\_sleft(letter\_height+200);

move\_fw(1750);

move\_sright(letter\_height-200);

move\_fw(1600);

move\_sleft(letter\_height+200);

move\_fw(1200);

move\_sright(700);

move\_bw(400);

move\_sright(200);

move\_diag(1000);

move\_fw(2800);

//turn\_left();

move\_bw(9000);

// turn\_right();

//move\_fw(1000);

flag5=1;

//move\_diag(3500);

}

}

void move\_diag(int movtime){

long int j = millis();

while (millis() - j <movtime){

x\_l = -0.4;

y\_l= -0.72;

ang\_vel=0;

robot\_move();

}

x\_l = 0;

}

void move\_fw(int movtime){

long int j = millis();

while (millis() - j <movtime){

x\_l = -0.4;

y\_l=0;

ang\_vel=0;

robot\_move();

}

x\_l = 0;

}

void move\_bw(int movtime){

long int j = millis();

while (millis() - j <movtime){

x\_l = 0.4;

y\_l=0;

ang\_vel=0;

robot\_move();

}

x\_l = 0;

}

void move\_sright(int movtime){

long int j = millis();

while (millis() - j <movtime){

y\_l = -0.4;

robot\_move();

}

x\_l = 0;

y\_l=0;

}

void move\_sleft(int movtime){

long int j = millis();

while (millis() - j <movtime){

y\_l = 0.4;

robot\_move();

}

x\_l = 0;

y\_l=0;

}

void turn\_right(){

long int j = millis();

while (millis() - j < 4800/2){

ang\_vel = -2;

x\_l = -0.3;

robot\_move();

}

ang\_vel = 0;

}

void turn\_left(){

long int j = millis();

while (millis() - j < 3600/2){

ang\_vel = 4;

x\_l = 0;

robot\_move();

}

ang\_vel = 0;

}

void normal\_drive(int ugl){

bl\_or\_white = 0;

if (analogRead(3) > threshold[3] && analogRead(4) < threshold[4] )

{

ang\_vel = ugl;

x\_l = -0.2;

prev\_line = 0;

}

else if (analogRead(4) > threshold[4] && analogRead(3) < threshold[3])

{

ang\_vel = -ugl;

x\_l =-0.2;

prev\_line = 1;

}

else if (prev\_line == 1 && analogRead(3) > threshold[3]){

ang\_vel = ugl;

x\_l = -0.05;

}

else if (prev\_line == 0 && analogRead(4) > threshold[4]){

ang\_vel = -ugl;

x\_l = -0.05;

}

else if (analogRead(2) < 700 && (analogRead(4) < threshold[4] && analogRead(3) < threshold[3])){ //////////////////////

prev\_line = -1;

x\_l = -0.05;

Kp = 0.0007 \* (1023 - analogRead(2));

Kd = 7 \* Kp;

Ki = 0.0001;

linefollow();

}

}

void inv\_drive(){

if (analogRead(3) < threshold[3] && analogRead(4) > threshold[4] )

{

ang\_vel = 2;

x\_l =-0.2;

}

else if (analogRead(4) < threshold[4] && analogRead(3) > threshold[3])

{

ang\_vel = -2;

x\_l =-0.2;

}

else if (analogRead(4) > threshold[4] && analogRead(3) > threshold[3])

{

Kp = 0.001 \* (1023 - analogRead(2));

Kd = 0.00007 \* Kp;

Ki = 0.0001;

linefollow\_inv();

}

bl\_or\_white = 1;

count\_black = 1;

black\_time();

}

void linefollow()

{

int error = (analogRead(0) - analogRead(1));

P = error;

I = I + error;

D = error - previousError;

PIDvalue = (Kp \* P) + (Ki \* I) + (Kd \* D);

previousError = error;

if (error >= 0){

sp = lfspeed + PIDvalue;

constrain(sp, 0, 255);

ang\_vel = map(sp, 0, 255, 0, 2);

}

if (error < 0){

sp = lfspeed - PIDvalue;

constrain(sp, 0, 255);

ang\_vel = map(sp, 0, 255, 0, -2);

}

if (count\_black == 1 or count\_black == 2){

x\_l = -0.2;

}

else{

x\_l = -0.2;

}

flag1 = 0;

}

void linefollow\_inv()

{

int error = (analogRead(0) - analogRead(1));

P = error;

I = I + error;

D = error - previousError;

PIDvalue = (Kp \* P) + (Ki \* I) + (Kd \* D);

previousError = error;

if (error >= 0){

sp = lfspeed + PIDvalue;

constrain(sp, 0, 255);

ang\_vel = map(sp, 0, 255, 0, -1);

}

if (error < 0){

sp = lfspeed - PIDvalue;

constrain(sp, 0, 255);

ang\_vel = map(sp, 0, 255, 0, 1);

}

x\_l = -0.5;

}

void calculateMotorSpeeds(float theta, float x\_l, float y\_l, float dertheta, float R, float r, float alpha1, float alpha2, float alpha3,

float &theta1, float &theta2, float &theta3) {

// Вычисление угловых скоростей моторов на основе уравнений

theta1 = (-sin(theta) \* cos(theta) \* x\_l + cos(theta) \* cos(theta) \* y\_l + R \* dertheta) / r;

theta2 = (-sin(theta + alpha2) \* cos(theta) \* x\_l + cos(theta + alpha2) \* cos(theta) \* y\_l + R \* dertheta) / r;

theta3 = (-sin(theta + alpha3) \* cos(theta) \* x\_l + cos(theta + alpha3) \* cos(theta) \* y\_l + R \* dertheta) / r;

}

void updateEncoderA() {

int b = digitalRead(encoder2MotorA);

int increment = 0;

if(b>0){

// If B is high, increment forward

increment = 1;

}

else{

// Otherwise, increment backward

increment = -1;

}

encoderPosA = encoderPosA + increment;

}

void updateEncoderB() {

int b = digitalRead(encoder2MotorB);

int increment = 0;

if(b>0){

increment = 1;

}

else{

increment = -1;

}

encoderPosB = encoderPosB + increment;

}

void updateEncoderC() {

int b = digitalRead(encoder2MotorC);

int increment = 0;

if(b>0){

increment = 1;

}

else{

increment = -1;

}

encoderPosC = encoderPosC + increment;

}

void calibration(){

for ( int i = 0; i < 5; i++)

{

minValues[i] = analogRead(i);

maxValues[i] = analogRead(i);

}

long int j = millis();

for (j; j < 4400; j=millis()){

ang\_vel = 3;

robot\_move();

for ( int i = 0; i < 5; i++)

{

if (analogRead(i) < minValues[i])

{

minValues[i] = analogRead(i);

}

if (analogRead(i) > maxValues[i])

{

maxValues[i] = analogRead(i);

}

}

}

for ( int i = 0; i < 5; i++)

{

threshold[i] = (minValues[i] + maxValues[i]) / 2;

Serial.print(threshold[i]);

Serial.print(" ");

}

Serial.println();

ang\_vel = 0;

}

bool ifSquare(){

int tresholdTime = 5000;

for (int i = 0; i>5; i++){

if (analogRead(i)<threshold[i]) return 0;

}

if (flag == 0){

tmpTime = millis();

flag = 1;

}

if(millis() - tmpTime > tresholdTime) return 1;

}

void vyezd(){

if (bl\_or\_white == 0 and flag2 == 0){

tmpTime2 = millis();

ugl = 1;

flag2 = 1;

}

if (millis() - tmpTime2 > 5000 and flag2 == 1){

ugl = 4;

}

}

void black\_time(){

if (bl\_or\_white == 1 and flazhok == 0){

if (flag1 == 0){

tmpTime1 = millis();

flag1 = 1;

}

else if (millis() - tmpTime1 > 3000 and flag1 == 1) flazhok = 1;

}

}

int timer(){

if (flag5 == 0 && counter() == 0){

tmpTime5 = millis();

flag5 = 1;

return 0;

}

else if (flag5 == 1 && millis() - tmpTime5 > 2000 && counter() == 0){

flag5 = 0;

return 1;

}

else if (counter() == 1){

flag5 = 0;

return 0;

}

else {

return 0;

}

}

void robot\_move(){

calculateMotorSpeeds(60/57.29578, x\_l, y\_l, ang\_vel, R, r, alpha1, alpha2, alpha3, theta1, theta2, theta3);

motorA.setTargetSpeedDeg(theta1 \* 57.29578);

motorB.setTargetSpeedDeg(theta2 \* 57.29578);

motorC.setTargetSpeedDeg(theta3 \* 57.29578);

motorA.tick(encoderPosA);

motorB.tick(encoderPosB);

motorC.tick(encoderPosC);

}

# Приложение 4

#include "AccelMotor.h"

#include <Servo.h>

#define trigPin 23

#define echoPin 2 //Rangefinder's pins

#define GRIPPER 3 // servo's pin

#define PIN\_EN\_MotorA 4 // Вывод управления скоростью вращения мотора №1

#define PIN\_EN\_MotorB 9 // Вывод управления скоростью вращения мотора №2

#define PIN\_EN\_MotorC 10 // Вывод управления скоростью вращения мотора №3

#define PIN\_IN1\_MotorA 5 // Вывод управления направлением вращения мотора №1

#define PIN\_IN2\_MotorA 6 // Вывод управления направлением вращения мотора №1

#define PIN\_IN1\_MotorB 8 // Вывод управления направлением вращения мотора №2

#define PIN\_IN2\_MotorB 7 // Вывод управления направлением вращения мотора №2

#define PIN\_IN1\_MotorC 12 // Вывод управления направлением вращения мотора №3

#define PIN\_IN2\_MotorC 11 // Вывод управления направлением вращения мотора №3

#define encoder1MotorA 18 //Вход энкодера мотора №1

#define encoder1MotorB 19 //Вход энкодера мотора №2

#define encoder1MotorC 20 //Вход энкодера мотора №3

#define encoder2MotorA 14 //Вход энкодера мотора №1

#define encoder2MotorB 15 //Вход энкодера мотора №2

#define encoder2MotorC 16 //Вход энкодера мотора №3

unsigned long timing = 0;

volatile long int encoderPosA = 0;

volatile long int encoderPosB = 0;

volatile long int encoderPosC = 0;

// значения параметров робота

float theta = 330/57.29578; // угол поворота робота отно глобальных координат

float x\_l = 0.0; // Скорость по Х

float y\_l = 0.0; // Скорость по

float ang\_vel = 0.0;

float R = 0.135; // Расстояние от центра робота до центра колеса

float r = 0.058; // Радиус колес

float alpha1 = 0.0; // Угол наклона омни - колеса 1 (в радианах)

float alpha2 = 2.09439510239; // Угол наклона омни - колеса 2 (в радианах)

float alpha3 = 4.18879020479; // Угол наклона омни - колеса 3 (в радианах)

float theta1, theta2, theta3;

AccelMotor motorA(DRIVER3WIRE, PIN\_IN1\_MotorA, PIN\_IN2\_MotorA, PIN\_EN\_MotorA, LOW);

AccelMotor motorB(DRIVER3WIRE, PIN\_IN1\_MotorB, PIN\_IN2\_MotorB, PIN\_EN\_MotorB, LOW);

AccelMotor motorC(DRIVER3WIRE, PIN\_IN1\_MotorC, PIN\_IN2\_MotorC, PIN\_EN\_MotorC, LOW);

Servo gripper;

float servo\_ang = 0.0;

int trigState = LOW; //state of trigPin

int interval = 10; // interval in Microseconds at which trigPin turns on

int printState = LOW; //whether or not to print distance

unsigned long previousMicros = 0; //microsecond at which the pin was last writen

unsigned long print\_interval = 0;

float duration, distance; //variables

long printing = 0;

void setup() {

// энкодер 11 тиков на оборот, передаточное отношение 56

motorA.setRatio(56 \* 11);

motorB.setRatio(56 \* 11);

motorC.setRatio(56 \* 11);

// период интегрирования (по умолч. 20)

motorA.setDt(5); // миллисекунды

motorB.setDt(5);

motorC.setDt(5);

// минимальный (по модулю) ШИМ сигнал (при котором мотор трогается)

motorA.setMinDuty(5);

motorB.setMinDuty(5);

motorC.setMinDuty(5);

// коэффициенты ПИД регулятора

motorA.kp = 0.5;

motorB.kp = 0.5;

motorC.kp = 0.5; // отвечает за резкость регулирования.

// При малых значениях сигнала вообще не будет, при слишком больших – будет трясти

motorA.ki = 3.0;

motorB.ki = 3.0;

motorC.ki = 3.0;

// отвечает за коррекцию ошибки в течение времени

motorA.kd = 0.009;

motorB.kd = 0.009;

motorC.kd = 0.009;// отвечает за компенсацию резких изменений

motorA.setRunMode(PID\_SPEED);

motorB.setRunMode(PID\_SPEED);

motorC.setRunMode(PID\_SPEED);

// PID\_SPEED - tick() работает в режиме поддержания скорости по ПИД регулятору

attachInterrupt(digitalPinToInterrupt(encoder1MotorA), updateEncoderA, RISING);// Прерывания для энкодеров

attachInterrupt(digitalPinToInterrupt(encoder1MotorB), updateEncoderB, RISING);

attachInterrupt(digitalPinToInterrupt(encoder1MotorC), updateEncoderC, RISING);

gripper.attach(3);

pinMode(trigPin,OUTPUT);

pinMode(echoPin,INPUT);

Serial.begin(115200);

}

void loop() {

timing = millis();

if (Serial.available()) {

int Status = 0;

String Input = Serial.readStringUntil('\n');

Input.trim();

float values[4];

int i = 0;

char\* substr = strtok(const\_cast<char\*>(Input.c\_str()), ",");

while (substr != NULL) {

values[i++] = atof(substr);

substr = strtok(NULL, ",");

}

x\_l = values[1];

y\_l = values[0];

ang\_vel = values[2];

servo\_ang = values[3];

}

// hcsro4 stuff

unsigned long currentMicros = micros();

if (currentMicros-previousMicros >= interval) {

previousMicros = currentMicros;

if (trigState == LOW){

(trigState = HIGH);

}

else {

(trigState = LOW);

}

}

robot\_move();

gripper.write(servo\_ang);

digitalWrite(trigPin,trigState);

duration = pulseIn(echoPin,HIGH,1500);

distance = (duration/2) / 29.1;

if (millis()-printing > 100 && distance >0){

printing = millis();

Serial.println(distance);

Serial.flush();

}

}

void calculateMotorSpeeds(float theta, float x\_l, float y\_l, float dertheta, float &theta1, float &theta2, float &theta3) {

float vabs =sqrt(pow(x\_l, 2) + pow(y\_l, 2));

if(vabs > 1.0){

x\_l = x\_l \* (1.0/vabs);

y\_l = y\_l \* (1.0/vabs);

}

// Вычисление угловых скоростей моторов на основе уравнений

theta1 = (-sin(theta) \* cos(theta) \* x\_l + cos(theta) \* cos(theta) \* y\_l + R \* dertheta) / r;

theta2 = (-sin(theta + alpha2) \* cos(theta) \* x\_l + cos(theta + alpha2) \* cos(theta) \* y\_l + R \* dertheta) / r;

theta3 = (-sin(theta + alpha3) \* cos(theta) \* x\_l + cos(theta + alpha3) \* cos(theta) \* y\_l + R \* dertheta) / r;

}

void updateEncoderA() {

int b = digitalRead(encoder2MotorA);

int increment = 0;

if(b>0){

increment = 1;

}

else{

increment = -1;

}

encoderPosA += increment;

}

void updateEncoderB() {

int b = digitalRead(encoder2MotorB);

int increment = 0;

if(b>0){

increment = 1;

}

else{

increment = -1;

}

encoderPosB += increment;

}

void updateEncoderC() {

int b = digitalRead(encoder2MotorC);

int increment = 0;

if(b>0){

increment = 1;

}

else{

increment = -1;

}

encoderPosC += increment;

}

void robot\_move(){

calculateMotorSpeeds(theta, x\_l,y\_l,ang\_vel, theta1, theta2, theta3);

motorA.setTargetSpeedDeg(theta1\*57.2957795);

motorB.setTargetSpeedDeg(theta2\*57.2957795);

motorC.setTargetSpeedDeg(theta3\*57.2957795);

motorA.tick(encoderPosA);

motorB.tick(encoderPosB);

motorC.tick(encoderPosC);

}

# Приложение 5

import cv2

import numpy as np

import rclpy

from rclpy.node import Node

from std\_msgs.msg import Int32MultiArray

class Barrel\_Publisher(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_("barrel\_publisher")

self.publisher\_ = self.create\_publisher(Int32MultiArray, "barrel\_topic", 10)

def detector(self):

cap = cv2.VideoCapture(0) #check the port of the camera

cap.set(cv2.CAP\_PROP\_FPS, 30)

msg = Int32MultiArray()

ret, frame = cap.read()

frame\_width = frame.shape[1] #w:image-width and h:image-height

hsvFrame = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV) # convert to hue - saturation - value

while True:

iSee = []

ret, frame = cap.read()

# Convert the frame to HSV color space

hsvFrame = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

# Set range for red color and define mask

red\_lower = np.array([136, 87, 111], np.uint8)

red\_upper = np.array([180, 255, 255], np.uint8)

red\_mask = cv2.inRange(hsvFrame, red\_lower, red\_upper)

# Set range for green color and define mask

green\_lower = np.array([25, 52, 72], np.uint8)

green\_upper = np.array([102, 255, 255], np.uint8)

green\_mask = cv2.inRange(hsvFrame, green\_lower, green\_upper)

# Set range for blue color and define mask

blue\_lower = np.array([94, 80, 2], np.uint8)

blue\_upper = np.array([120, 255, 255], np.uint8)

blue\_mask = cv2.inRange(hsvFrame, blue\_lower, blue\_upper)

# Morphological Transform, Dilation for each color

kernal = np.ones((5, 5), "uint8")

# For red color

red\_mask = cv2.dilate(red\_mask, kernal)

# For green color

green\_mask = cv2.dilate(green\_mask, kernal)

# For blue color

blue\_mask = cv2.dilate(blue\_mask, kernal)

# Finding contours for red color

contours, \_ = cv2.findContours(red\_mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

if contours:

largest\_contour\_red = max(contours, key=cv2.contourArea)

M\_red = cv2.moments(largest\_contour\_red)

cx\_red = int(M\_red['m10'] / M\_red['m00'])

cy\_red = int(M\_red['m01'] / M\_red['m00'])

cv2.circle(frame, (cx\_red, cy\_red), 5, (0, 0, 255), -1)

x\_Diff\_red = int(cx\_red - frame\_width/2)

msg.data.insert(1, x\_Diff\_red)

msg.data.insert(2, 0)

self.publisher\_.publish(msg)

self.get\_logger().info('Distance from red"%d"' % x\_Diff\_red)

# Finding contours for green color

contours, \_ = cv2.findContours(green\_mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

if contours:

largest\_contour\_green = max(contours, key=cv2.contourArea)

M\_green = cv2.moments(largest\_contour\_green)

cx\_green = int(M\_green['m10'] / M\_green['m00'])

cy\_green = int(M\_green['m01'] / M\_green['m00'])

cv2.circle(frame, (cx\_green, cy\_green), 5, (0, 255, 0), -1)

x\_Diff\_green = int(cx\_green - frame\_width/2)

msg.data.insert(1, x\_Diff\_green)

msg.data.insert(2, 0)

self.publisher\_.publish(msg)

self.get\_logger().info('Distance from green%d"' % x\_Diff\_green)

# Finding contours for blue color

contours, \_ = cv2.findContours(blue\_mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

if contours:

largest\_contour\_blue = max(contours, key=cv2.contourArea)

M\_blue = cv2.moments(largest\_contour\_blue)

cx\_blue = int(M\_blue['m10'] / M\_blue['m00'])

cy\_blue = int(M\_blue['m01'] / M\_blue['m00'])

cv2.circle(frame, (cx\_blue, cy\_blue), 5, (255, 0, 0), -1)

x\_Diff\_blue= int(cx\_blue - frame\_width/2)

msg.data.insert(1, x\_Diff\_blue)

msg.data.insert(2, 0)

self.publisher\_.publish(msg)

self.get\_logger().info('Distance from blue%d"' % x\_Diff\_blue)

# Program Termination

#cv2.imshow("Multiple Color Detection in Real-TIme", frame)

if cv2.waitKey(10) & 0xFF == ord('q'):

cap.release()

cv2.destroyAllWindows()

break

def main(args=None):

rclpy.init(args=args)

Barrel = Barrel\_Publisher()

Barrel.detector()

Barrel.destroy\_node()

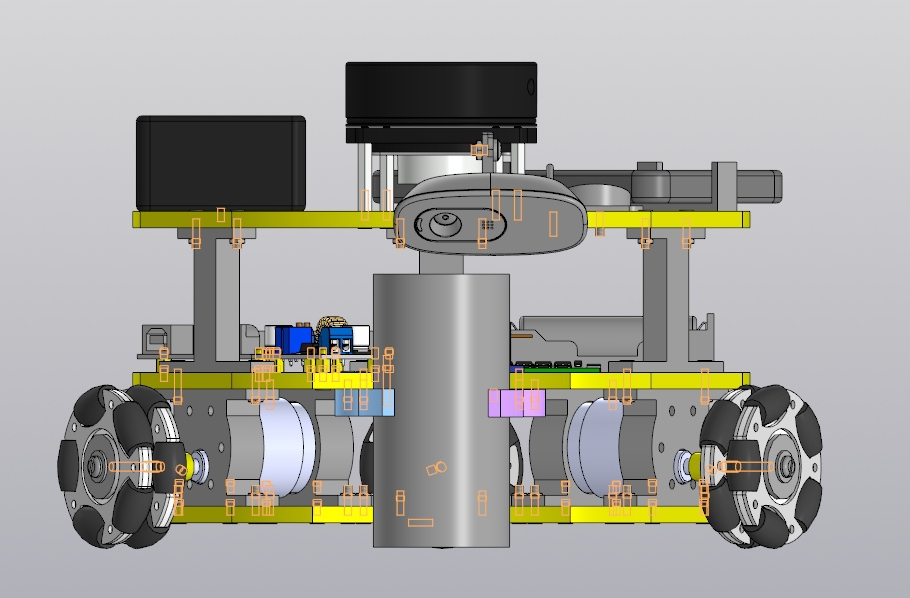
rclpy.shutdown()

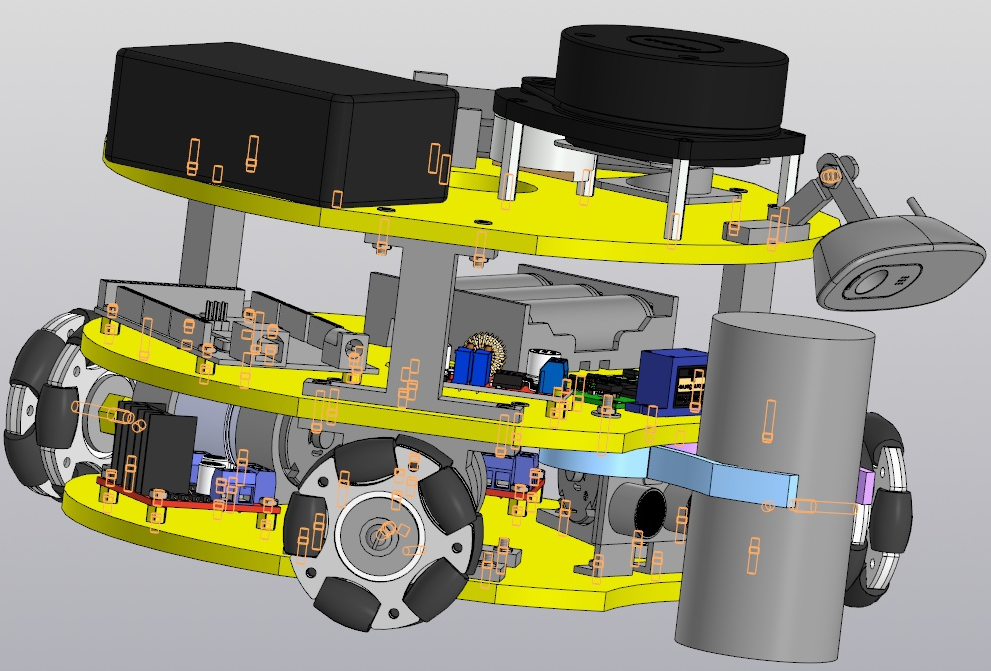
if \_\_name\_\_ == '\_\_main\_\_':

main()

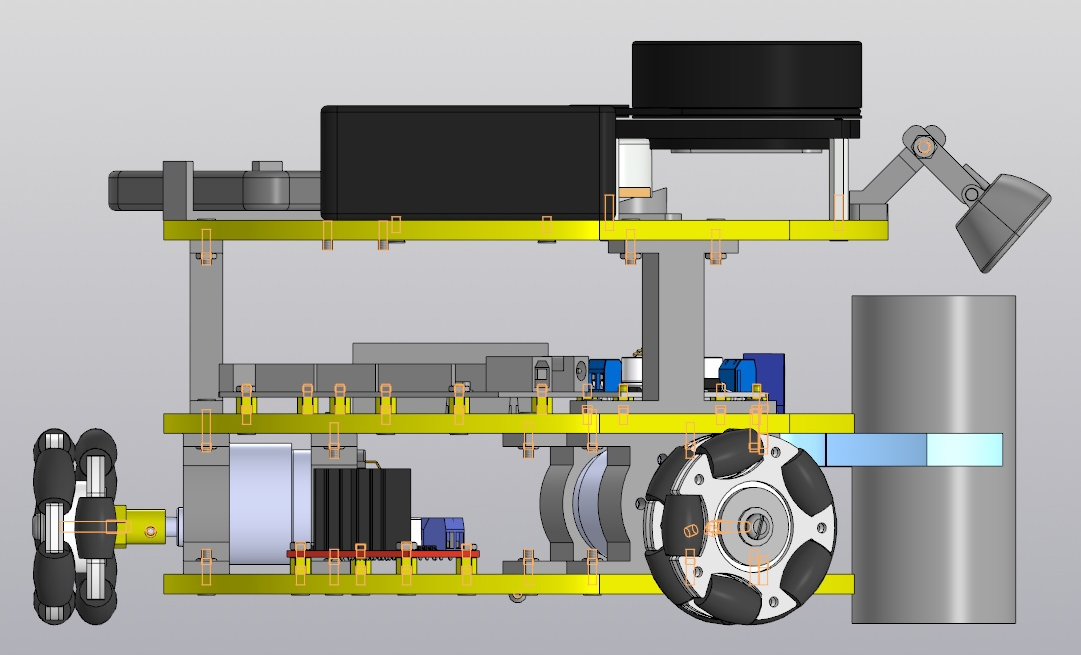
# Приложение 6

3Д модель робота









# Приложение 7

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_Специальное машиностроение (СМ)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_Робототехника и комплексная автоматизация (РК)\_\_\_

КАФЕДРА \_\_\_\_\_\_Робототехнические системы и мехатроника (СМ7)\_\_\_\_

*\_*Компьютерные системы автоматизации производства (РК9)\_

**Техническое задание**

**на курсовой проект**

**по теории механизмов и машин**

**на тему**

**«Проектирование робота для участия в открытых соревнованиях Bauman Robotics Club»**

Выполнили:

Студент \_\_\_\_СМ7-54Б\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Иванов И.С.**\_\_\_**

(Группа) (Подпись, дата) (Фамилия И.О)

Студент \_\_\_\_\_СМ7-54Б\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Колесников В.Е**.\_**

(Группа) (Подпись, дата) (Фамилия И.О)

Студент \_\_\_\_\_СМ7-54Б\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Гаденов Т.Е.**\_\_\_\_**

(Группа) (Подпись, дата) (Фамилия И.О.)

Студент \_\_\_\_\_СМ7-54Б\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Фяткуллов Д.Н.**\_\_**

(Группа) (Подпись, дата) (Фамилия И.О.)

Студент \_\_\_\_\_РК9-52Б\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Давыдова К.А.\_\_  
 (Группа) (Подпись, дата) (Фамилия И.О.)

Руководитель проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_** Сащенко Д.В.**\_\_**

(Подпись, дата) (Фамилия И.О.)

**Оглавление**

[1.](#_heading=h.gjdgxs) Введение………………………………………………………………………. 3

[2.](#_heading=h.30j0zll) Требования к проектируемому изделию……………………………………. 4

[2.1.](#_heading=h.1fob9te) Требования назначения………………………………………………….. 4

[2.2.](#_heading=h.3znysh7) Требования надежности…………………………………………………. 4

[2.3.](#_heading=h.2et92p0) Требования безопасности………………………………………………... 4

[2.5.](#_heading=h.3dy6vkm) Конструктивные требования……………………………………………. 5

[3.](#_heading=h.4d34og8) Технико-экономические показатели………………………………………... 6

[4.](#_heading=h.2s8eyo1) Стадии и этапы разработки………………………………………………….. 7

[5.](#_heading=h.17dp8vu) Ответственные за разработку………………………………………………... 8

# Введение

Робототехника – прикладная наука, занимающаяся разработкой автоматизированных технических систем и являющаяся важнейшей технической основой развития производства. Робототехника опирается на такие дисциплины, как электроника, механика, кибернетика, телемеханика, мехатроника, информатика, а также радиотехника и электротехника.

Робототехника является одним из важнейших направлений научно-технического прогресса, где проблемы механики и новых технологий контактируют с проблемами искусственного интеллекта. В настоящее время роботы используются практически во всех видах промышленности, в особенности, где человек подвергается большой опасности.

В настоящее время в школах и институтах проводят конференции, интенсивы, соревнования с целью внедрения робототехники в процесс обучения, обмена новыми научно-техническими идеями и инженерными знаниями. Примером такого рода мероприятий являются открытые робототехнические соревнования Bauman Robotics Club. Существует регламент (см. Приложение Х), в котором расписаны основные правила соревнований.

**Цель** данного курсового проекта заключается в проектировании, сборке и тестировании робота для участия в робототехнических соревнованиях Bauman Robotics Club.

**Задачи** курсового проекта:

* анализ основных задач робота, исходя из регламента;
* проектирование механизма схвата, мобильной платформы, схемы и способа питания;
* выбор по различным критериям вида схвата, датчиков, контроллера
* выбор способа перемещения робота;
* написание кода для управляющего контроллера;
* сборка робота.

# Требования к проектируемому изделию

# 2.1. Требования назначения

* 1. Основная задача: поочерёдно захватить и перенести в специально обозначенные области поля цилиндры разного цвета диаметром 50 мм и высотой 100 мм.
  2. Робот должен быть полностью автономным. Запрещено любое взаимодействие с роботом во время выполнения задания.

# 2.2. Требования надежности

1. Робот должен обладать такими критериями обеспечения работоспособности, как прочность (способность не разрушаться), жесткость (способность не деформироваться)
2. Разрабатываемый робот должен обладать такими показателями надежности, как безотказность (свойство объекта непрерывно сохранять свою работоспособность в течение заданного времени), сохраняемость (свойство объекта сохранять требуемые эксплуатационные показатели).
3. Разрабатываемое программное обеспечение управляющего устройства робота должно обладать следующими критериями: корректность (способность выполнять точные задачи так, как они определены их спецификацией), обслуживаемость (совокупность свойств системы, обеспечивающих возможность приведения ее в состояние готовности к применению и поддержание этого состояния во времени).

# 2.3. Требования безопасности

1. Робот не должен причинять ущерб соревновательному полю, участникам и организаторам соревнований.

# 2.4. Конструктивные требования

1. Все модули должны располагаться на борту робота.
2. Габаритные размеры робота не должны превышать – 300 мм в длину, 300 мм в ширину, без ограничения на высоту.

# Технико-экономические показатели

В основе проектирования робота будет лежать несколько критериев:

* + - * доступность исходных материалов;
      * качество исходных материалов;
      * минимизация типов различных материалов с учетом стоимости, взаимозаменяемости запасных частей.

# Стадии и этапы разработки

План разработки представлен в табл. 1:

Таблица 1. Этапы разработки

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № этапа | Название этапа | Содержание этапа | Планируемая дата выполнения | Фактическая дата выполнения | Подпись студента | Подпись преподавателя |
| 1 | Концептуальное проектирование | Выбор и обоснование реализации основных частей робота, разработка эскизов и моделей взаимного положения частей робота. | 30.09.2023 |  |  |  |
| 2 | Проектирование нижнего уровня робота | Проектирование подвижной платформы, механизма схвата. | 14.10.2023 |  |  |  |
| 3 | Проектирование тактического уровня робота | Разработка системы управления движением приводов. | 28.10.2023 |  |  |  |
| 4 | Проектирование высшего уровня робота | Реализация алгоритмов компьютерного зрения, планирования траектории движения | 25.11.2023 |  |  |  |
| 5 | Подготовка технической документации | Подготовка чертежей | 2.12.2023 |  |  |  |

# Ответственные за разработку

Таблица 2. Ответственные за разработку

|  |  |  |  |
| --- | --- | --- | --- |
| № этапа | Название этапа | Документация для отчета | Исполнители этапа |
| 1 | Концептуальное проектирование | Эскизы / 3-D модели / концептуальные чертежи робота, техническое задание | Иванов И.С.  Гаденов Т.Е.  Колесников В.Е.  Фяткуллов Д.Н.  Давыдова К.А. |
| 2 | Проектирование исполнительного уровня робота | Эскизы / 3-D модели / чертежи робота, чертежи схвата, подвижной платформы, | Колесников В.Е.  Фяткуллов Д.Н.  Давыдова К.А. |
| 3 | Проектирование тактического уровня робота | Программный код для микроконтроллера, чертеж электронной схемы робота | Иванов И.С.  Гаденов Т.Е.  Колесников В.Е.  Фяткуллов Д.Н.  Давыдова К.А. |
| 4 | Проектирование высшего (интеллектуального) уровня робота | Программный код | Иванов И.С.  Гаденов Т.Е. |
| 5 | Подготовка технической документации | 3-D модели / чертежи робота, пояснительная записка | Иванов И.С.  Гаденов Т.Е.  Колесников В.Е.  Фяткуллов Д.Н.  Давыдова К.А. |