МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименования института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Электроника и робототехника
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Система управления мобильным роботом

Обучающийся Н.А. Садовников
(И.О. Фамилия) (личная подпись)
Руководитель к.т.н., Е.С. Глибин
(ученая степень, звание, И.О. Фамилия)
Консультант к.п.н., доцент О.В. Лебидинская
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Название бакалаврской работы: «Система управления мобильным роботом».

Выпускная работа состоит из введения, восьми разделов, заключения, тридцати шести рисунков, двух таблиц, девяти листингов, списка литературы, включая зарубежные источники, четырёх приложений и графической части на шести листах формата А1.

В данной работе представлен процесс проектирования и реализации колесного мобильного робота. Особенностью данной разработки является использование двух вычислительных машин с системой ведущий-ведомый: ведущий для работы с техническим зрением, ведомый для управления движением.

Целью работы является разработка колёсной роботизированной платформы, ориентирующейся в пространстве с помощью технического зрения, внутри замкнутого помещения. Робот должен быть способен перемещаться по покрытиям различных типов, таких как линолеум и напольная плитка.

Бакалаврская работа может быть разделена на следующие логически взаимосвязанные части: анализ рынка и актуальности изделия; выбор способа ориентации в пространстве; разработка структурной схемы; связь между узлами мобильного робота; разработка электрической схемы соединений; разработка конструкции мобильного робота; создание программы управления; экономический расчёт данного проекта.

По окончанию разработки представлен прототип, успешно выполняющий заданные требования.

Подводя итоги, хотелось бы отметить, что в мире всё большее распространение получают мобильные робототехнические комплексы, предполагающие самостоятельное движение в замкнутых пространствах на колёсной базе, то есть без направляющих рельс и управляющих радиосигналов.

Данное решение удобно использовать в складской логистике, внутрипроизводственной логистике, сфере обслуживания.

Abstract

The topic of the given graduation work is «Control system for mobile robot».

The graduation work consists of an explanatory note on 67 pages, introduction, 8 sections, conclusion, including 36 figures, 2 tables, the list of 20 references including 6 foreign sources and 4 appendices, and the graphic part on 6 A1 sheets.

The key issue of the graduation work is the design of wheeled mobile robot using two computing machines with a master-slave system: the master to work with computer vision, the slave to control the movement.

The aim of the work is the development of a wheeled robotic platform that navigates inside an enclosed room using computer vision. The robot should be able to move stably on different types of surfaces such as linoleum and floor tiles.

The graduation work may be divided into several logically connected parts which are analysis of the market and the relevance of the product; choice of method of orientation in space; development the structural diagram, the connection between the modules of the mobile robot; development of electrical wiring diagram; development of the mobile robot design creation of two driving control programs; the economics of this project.

Finally, we present prototype that fulfills the specified requirements.

In conclusion we'd like to note that mobile robotic systems, which involve independent movement in confined spaces on a wheeled base, i.e. without guiding rails and controlling radio signals, are becoming more and more widespread in the world. This solution is convenient to use in warehouse logistics, in-process logistics, and service industries.

Содержание

Введение

Робототехническое направление одно из перспективных. Роботы всё активней применяются для автоматизации производств, ускорения темпов и оптимизации производства, замены людей в опасных средах, повышения точности и качества производства.

Так, например, высокое качество покраски деталей при массовом производстве, может быть обеспечено с помощью роботов, без рисков для человека (рисунок 1).

Рисунок 1 – Покраска кузова автомобиля с помощью робота

Высокая точность, скорость, совмещённая с безопасностью, присуща хорошо настроенным гибким производственным системам в стекольной и фарфорофаянсовой промышленности (рисунок 2).

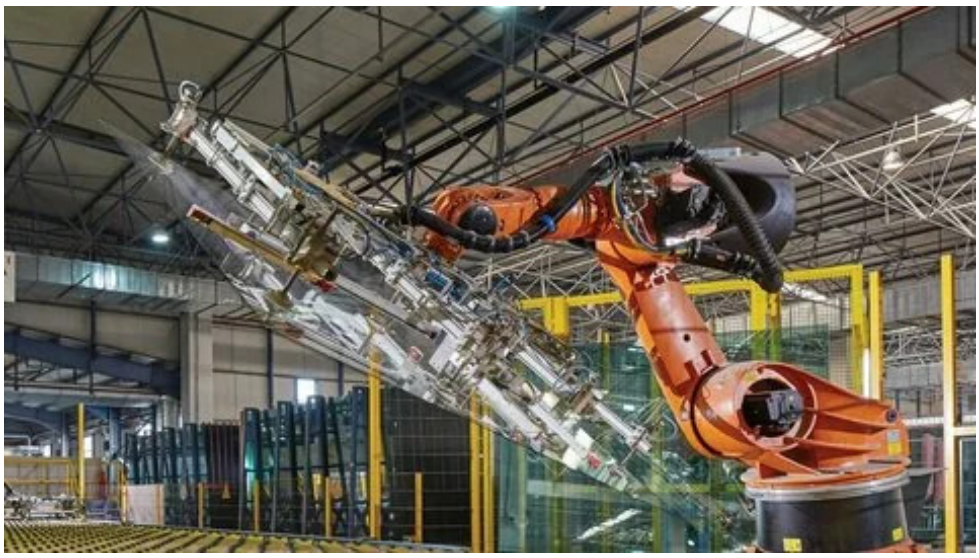

Рисунок 2 – Промышленный манипулятор со стеклом большой площади

Космическая отрасль на заре своего существования получала роботов. Со времён «космической гонки», человечество использует мобильные робототехнические комплексы для исследования планет и спутников Солнечной системы. Они способны воспроизводить сенсорные, коммуникационные, управляющие и двигательные функции человека Тенденция к расширению автоматизации сохраняется. Так, роботехнический комплекс «Канадарм2» (рисунок 3) перемещает оборудование и материалы в пределах международной космической станции, помогает космонавтам работать в открытом космосе, обслуживает полезную нагрузку на поверхности МКС.

Рисунок 3 – «Канадарм2» захватывает грузовой корабль

Подводная деятельность человека так же не обходится без роботов. Например, для обследования континентального шельфа создан робототехнических комплекс «Шельф» (рисунок 4).

Рисунок 4 – Телеуправляемый подводный робототехнический комплекс «Шельф» на выставке «Армия-2021»

Всё большее значение робототехнические комплексы получают в армии. Разведка, целеуказание, корректировка артиллерийского огня, разминирование, действия в условиях химического и биологического заражения, непосредственная поддержка войск – все эти задачи могут на себя взять роботы уже сегодня [7].

Обеспечивая безопасность для оператора, робототехнический комплекс разминирования «Уран-6» способен проделывать проходы в минно-взрывных заграждениях, обезвреживать минные заграждения. Работать урбанизированных локальных районах и малолесных горных областях (рисунок 5).

Рисунок 5 – Робототехнический комплекс разминирования «Уран-6»

Для ведения боя создаются боевые машины, оснащённые различным вооружением, от пулемётов до противотанковых управляемых ракет. Примером такой машины, что уже прошла боевые действия, является боевой многофункциональный робототехнический комплекс «Уран-9» (рисунок 6).

Рисунок 6 – Боевой робот «Уран-9»

Военный транспорт так же получает развитие. Российская компания «КАМАЗ» в экспериментальном порядке оснастила свои грузовики средствами, превращающими обычный серийный автомобиль в беспилотник (рисунок 7).

Рисунок 7 – КАМАЗ-5350, оснащённый современной системой помощи водителю ADAS

В гражданских отраслях так же получают широкое распространение наземные мобильные роботы. Не один год компании внедряют

«интеллектуальные системы помощи водителю», которые представляют собой робототехнический комплекс, анализирующий данные вокруг и внутри автомобиля, способные предотвратить столкновения, удерживать автомобиль на полосе. Активно разрабатываются автономные роботы, способные передвигаться на трассах и в городе без водителя. Например, компания Яндекс разрабатывает самоуправляемые транспортные средства на базе серийных автомобилей (рисунок 8).

Рисунок 8 – Беспилотное транспортное средство от компании Яндекс

Другим направлением автоматизации, в которой прогресс так же не стоит на месте, является складская логистика и транспортировка грузов. Крупнейшие компании всё чаще используют роботов для всевозможных операций. От перемещений внутри склада (рисунок 9 слева) до доставки товаров непосредственно потребителю (рисунок 9 справа). Это направление так же активно развивает компания Яндекс.

Рисунок 9 – Робототехнические комплексы в логистике, слева беспилотный кран, справа робот-курьер

Есть и другие отрасли, где активно применяется автоматизация. Промышленное производство, внутрицеховой транспорт, медицина, образование и т.д. Технический прогресс позволяет создавать роботов для самых различных областей, чтобы повысить эффективность, снизить затраты. Чтобы оградить человека от опасных и вредных факторов, получать физический удалённый доступ. Чтобы повысить комфорт и освободить человека от рутинных однообразных действий. И процесс автоматизации имеет положительную тенденцию.

1 Анализ рынка и актуальности изделия

С каждым годом все больше компаний прибегают к использованию роботизированных складов, поскольку они значительно увеличивают эффективность складских операций. С ростом интернет-торговли и сокращением сроков доставки к клиенту, а также в связи с необходимостью удовлетворения потребностей в доставке товаров непосредственно до дверей во время пандемии COVID-19, этот тренд только усиливается. Повышение эффективности складской логистики становится крайне важным в условиях роста нагрузки. Автоматизация склада уже была успешно внедрена в некоторых логистических компаниях. Сегодня же, согласно опыту мировых лидеров, следующим шагом должна стать роботизация склада [13].

Для этого активно разрабатываются роботизированные системы для обеспечения складской логистики, благодаря их большой гибкости и масштабируемости в сравнении с классическими моделями. Последние используют рельсовые направляющие, краны, напольные линии для навигации передвижных платформ и управляемые людьми погрузчики. Классические системы достаточно громоздкие, ограничены в применении и требовательны к эксплуатационным условиям. Например, при использовании линий для перемещения платформ, чистый пол является необходимым условием для бесперебойной работы системы, в противном случае возможны сбои в работе. Системы с направляющими имеют наибольшие размеры и наименьшую мобильность среди классических. При использовании погрузчиков, которые управляются людьми, возникает ряд проблем, включающий дополнительные затраты на оплату труда операторов, вероятность человеческой ошибки и необходимость подготовки персонала для работы с погрузчиками. Также необходимо обеспечить безопасные условия труда, включая чистоту помещений, освещение, вентиляцию и т.д., кроме того, принимать меры от кражи на складах. Роботы гораздо менее требовательны в данных вопросах. В

дополнение, автоматизированные устройства имеют меньший шанс ошибок в работе.

Роботизированные комплексы предоставляют возможность настройки и адаптации систем для конкретных задач, уменьшают участие человека в процессе работы, повышают эффективность и скорость выполнения работ, а также в некоторых случаях могут упростить требования к условиям труда.

Доступных решений для создания мобильных роботизированных платформ, которые могут ориентироваться в закрытых пространствах, пока не так много. Рассмотрим существующие системы.

Примером современной мобильной платформы может служить патент CN113686332A Европейского патентного ведомства (рисунок 10). Патент освещает конструкцию мобильного робота и способ его навигации. Данная мобильная платформа предназначена для использования в складской логистике. В передней части корпуса данный мобильный робот имеет систему для получения изображений окружающей среды. Изображение включает верхнюю часть складских полок и потолок склада, так как расположение грузов и полок нередко меняется. Для получения изображения используется структурированный подсвет. Проектор излучает световые узоры (паттерны), которые деформируются при отражении от объекта. Затем две камеры распознают геометрию объекта с помощью алгоритмов триангуляции. Для передвижения данный мобильный робот использует 4 колеса малого диаметра, приводимые в движение электромоторами[18].

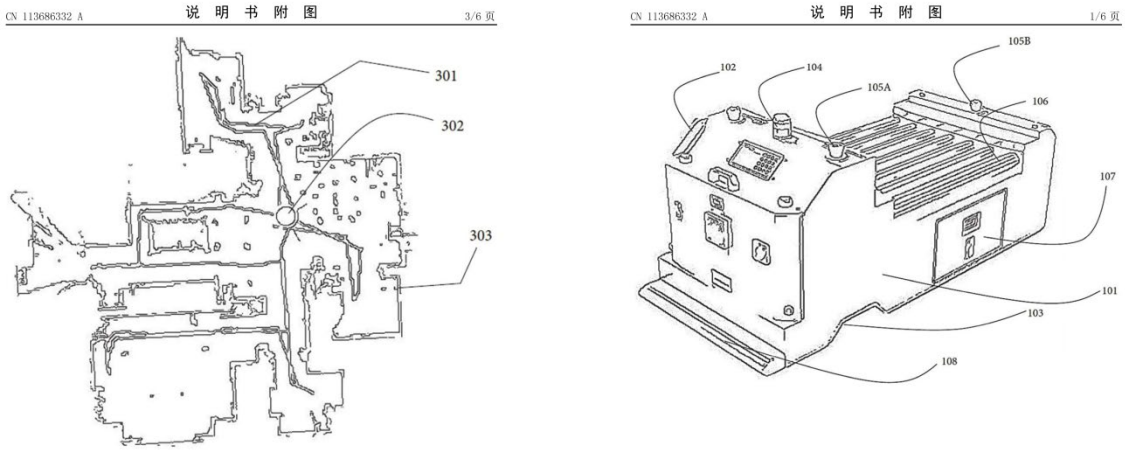

Рисунок 10 – Карта помещения, автоматически формируемая в ходе работы и внешний вид мобильного робота патента CN113686332A

Другим примером современной мобильной платформы, разработанной специально для складской логистики, может быть патент WO2019141257A1 Европейского патентного агентства (рисунок 11). Данный мобильный робот предназначен для транспортировки грузовых платформ. Для этого робот имеет относительно малые габариты. Во время работы он подъезжает под днище подвижной грузовой платформы, происходит зацепление, и перемещение груза по заданному маршруту на складе. Для ориентации используется камера, а так же датчик RFID – Radio Frequency Identification – датчик радиочастотной идентификации на днище мобильной платформы. Робот может работать в автономном режиме или в режиме радиоуправления. Для обхода препятствий используется шесть ультразвуковых датчиков. Вычисления происходят на базе микроконтроллера Arduino. Данный робот заявляется всенаправленным благодаря четырём омниколёсам – то есть способным двигаться по любому направлению. В качестве привода колёс используются электромоторы, в качестве питания – аккумулятор[18].

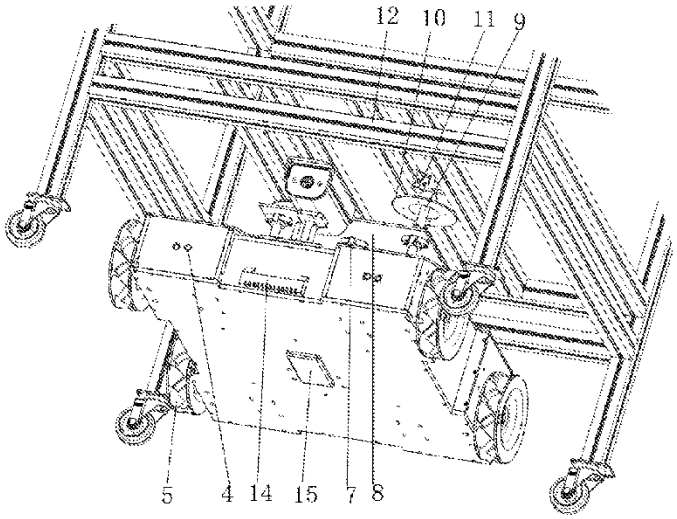

图 3

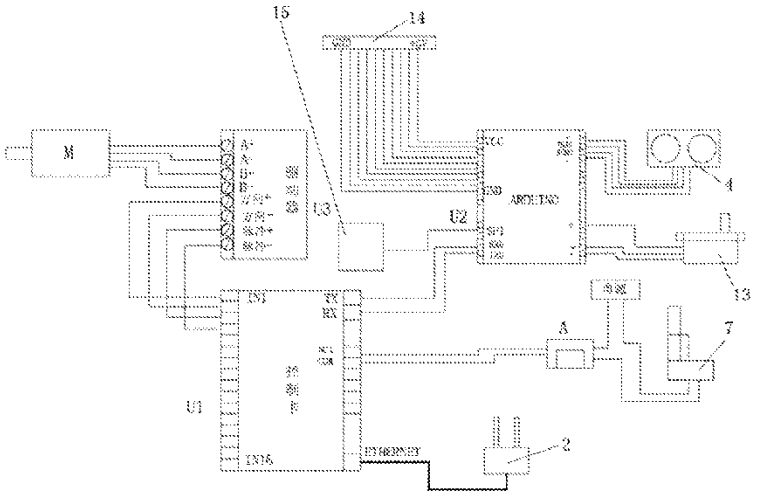

图 4

Рисунок 11 – Внешний вид мобильного робота патента WO2019141257A1 вместе с грузовой платформой и его схема электрическая соединений.

Таким образом, современные мобильные роботы, используемые в складской логистике, имеют ряд общих черт. В первую очередь, они рассчитаны на перевозки грузов, обладают колёсным движителем (состоящим из 4-х опорных колёс), используют техническое зрение для ориентации, используют электромоторы в качестве приводов колёс и собственное автономное электропитание.

2 Выбор способа ориентации в пространстве

В настоящий момент существует целый ряд способов ориентации и навигации мобильного робота в пространстве [6].

Простейшим способом может быть жёсткое механическое ориентирование с помощью направляющих. Мобильная платформа будет способна двигаться вдоль направляющих-рельс (рисунок 12). Однако, очевидны недостатки такого способа: нужна значительная подготовка к работе такого робота, создания физической системы направляющих. Кроме того, возникает проблема с переналадкой, требующая больших материальных и временных ресурсов.

Рисунок 12 – Портальный робот-манипулятор, жёстко закреплённый на направляющей

Возможно использование инерциальных датчиков (рисунок 13). Данный способ предполагает гораздо большую гибкость. Инерциальные датчики служат для измерения различных параметров, таких как угол наклона, угловая скорость, ускорение и т.д. Инерциальные датчики могут быть использованы для предоставления обратной связи для мобильного робота, что позволяет ему

контролировать свое положение и ориентацию в пространстве. При этом, навигация, основанная на инерциальных датчиках, имеет свои недостатки. Инерциальные датчики могут дрейфовать в течение времени, что может привести к ошибкам в измерении ускорения и угловой скорости. Инерциальные датчики могут иметь ограниченную точность в зависимости от их конструкции и качества изготовления. Калибровка инерциальных датчиков может быть сложной и требовать специального оборудования и навыков [11].

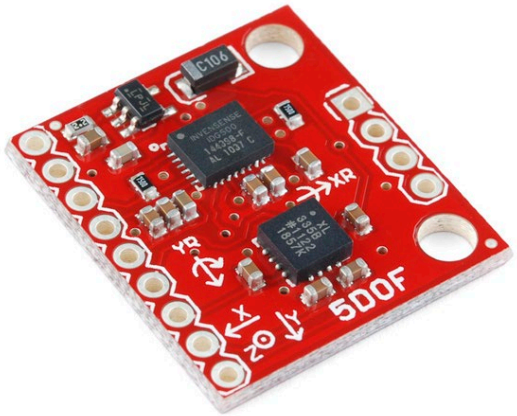

Рисунок 13 – Инерциальный датчик IMU Analog Combo Board - 5 Degrees of Freedom IDG500/ADXL335

Ещё одним способом является движение по инфракрасно-контрастной линии с помощью инфракрасных датчиков (рисунок 14). С одной стороны, способ очень дёшев, так как применяемое оборудование навигации (клейкая лента и инфракрасные датчики) имеют относительно невысокую стоимость, прост в изготовлении, настройке и программировании. С другой стороны, данный способ очень ограничен: при таком варианте исполнения системы позиционирования необходимы дополнительные датчики для обнаружения препятствий, данная система чувствительна к состоянию линии и к напольному покрытию (например, может реагировать не только на линию, вдоль которой идёт следование, но и на другие объекты, вроде затирки между плит), требует настройки для разных покрытий, по которым едет мобильный робот,

невозможно подготовиться к повороту, так как поле зрения сильно ограничено, способ зависит от освещённости окружающего пространства [5].

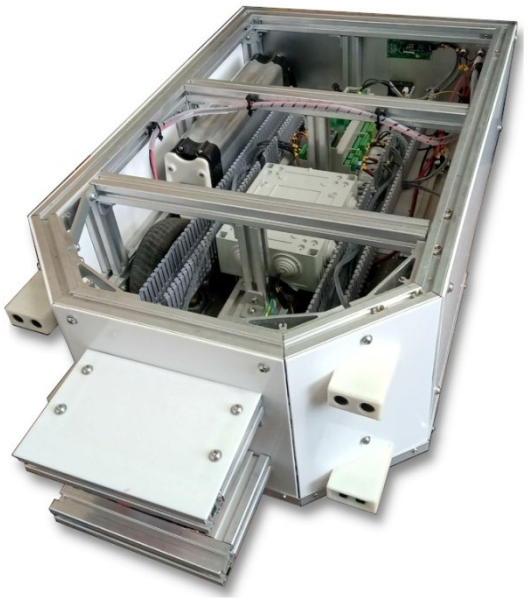

Рисунок 14 – Мобильный робот, осуществляющий навигацию с помощью инфракрасных датчиков и линии

Как правило, большинство современных систем ориентации в пространстве роботов это целый комплекс различных датчиков. Например, набирает популярность метод ориентирования SLAM. Как упоминалось ранее, Simultaneous Localisation and Mapping одновременно с помощью различных датчиков создаёт карту местности и определяет координаты робота в пространстве (рисунок 15). Данный подход увеличивает стоимость мобильной платформы, усложняет программирование, требует больших вычислительных мощностей.

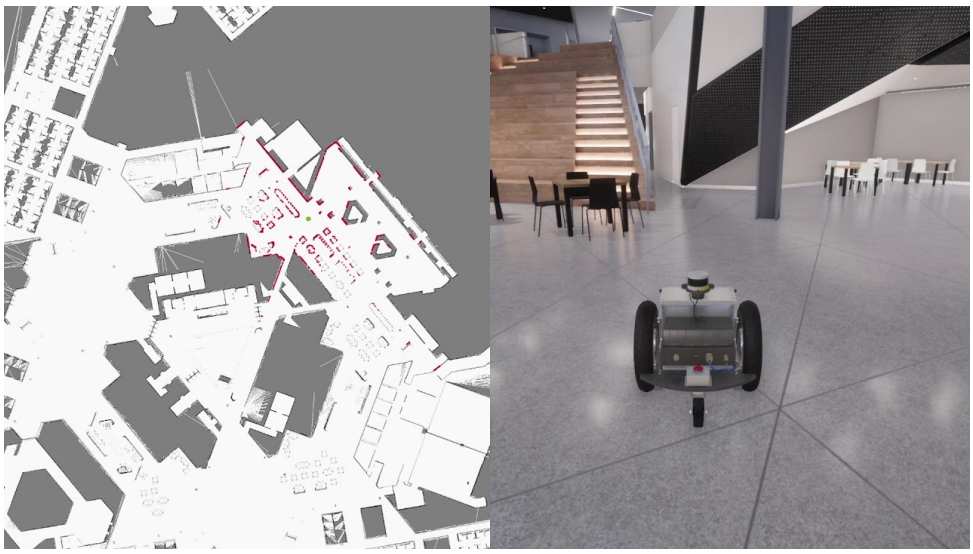

Рисунок 15 – Пример карты SLAM и робота, по ней ориентирующегося

На фоне большинства способов позиционирования, наиболее подходящим для данного проекта является движение с помощью использования камеры, компьютерного зрения и контрастной линии. Примером применения такого подхода может служить мобильный робот AGV 1811 компании INTEC (рисунок 16).

Рисунок 16 – Мобильный робот AGV INTEC 1811

Данный вид позиционирования очень прост в подготовке:

- требует минимального оборудования: фактически, только линии-направляющей;
- система работает в любых условиях освещения, что делает ее более гибкой, чем другие методы навигации;
- такая система обеспечивает высокую точность при навигации, так как контрастная линия является четким ориентиром для мобильного робота;
- не слишком требователен к качеству линии: даже некоторые повреждения не смогут нарушить движение робота;
- одним из основных преимуществ является низкий расход энергии, что позволяет использовать ее даже в условиях, когда мобильный робот имеет ограниченный источник питания;
- не очень требователен к покрытию, по которому происходит движение, главное, чтобы линия оставалась контрастной;
- возможно обнаружение препятствий без дополнительных технических средств.

Таким образом, способ ориентации в пространстве, основанный на компьютерном зрении и контрастной линии, обладает внушительным списком преимуществ, различными способами реализации, при этом достаточно прост и имеет минимум недостатков, особенно для движения в условиях закрытых помещений [1].

3 Разработка структурной схемы

Для того чтобы выполнить задачи, связанные как с компьютерным зрением, то есть с обработкой изображения, так и с функциями управляющего силовой частью устройства, необходимо использовать компьютер, обладающий достаточной вычислительной мощностью, однако, имеющий небольшие размеры, чтобы свободно уместиться в корпусе мобильного робота, и низкое энергопотребление для более продолжительной работы. В данном случае, подходящим вариантом будет одноплатный компьютер [10].

Поскольку обработка изображений требует значительного времени и затрат вычислительных ресурсов, для эффективного управления движением следует использовать отдельное устройство, которое будет принимать только команды управляющего устройства, при этом самостоятельно выполнять задачи по передвижению. Чтобы решить данную задачу, во время разработки прототипа, можно выбрать функционально законченный микроконтроллер.

Обеспечение ориентации в пространстве планируется только с помощью веб-камеры, которая будет регистрировать контрастную линию, вдоль которой и будет двигаться робот.

Передвижение мобильного робота предполагается с помощью двух ведущих колёс, приводимых в движение коллекторными электродвигателями, управление которыми ведётся с помощью драйвера. Два других колеса неуправляемые, изменяют своё направление свободно.

Во время разработки и регулирования мобильной платформы возникнет необходимость в интерфейсе общения с пользователем. Для этого необходимы экран, клавиатура и мышь.

Питание мобильного робота лучше всего разделить на две части. Для логической части, то есть микроконтроллера, одноплатного компьютера и периферии, и для силовой части, то есть электромоторов. Это решение вызвано несколькими соображениями.

Во-первых, потребление электромоторов достаточно велико, и приводит к относительно быстрой разрядке аккумулятора. Чтобы провести его замену, придётся обесточить всю систему. В схеме с двумя аккумуляторами этот недостаток отсутствует, так как за питание логики отвечает отдельный аккумулятор. Во-вторых, во время отладки работы программы управления удобно обесточить только силовую часть, при этом работая с полноценной программой, которая обсчитывает и передаёт все необходимые данные. В-третьих, использование коллекторных двигателей приводит к появлению искажений в электрической сети. Если вся сеть имеет приборы, которые требуют высокого качества электропитания, искажения могут вызвать отказ в работе этих приборов. Для исключения возможных сбоев может также потребоваться установка дополнительных фильтров, что повлечёт усложнение конструкции, увеличение затрат и объёма работ.

После определения структуры мобильного робота, приступим к выбору конкретных устройств, что будут использованы в дальнейшем.

Начнём выбор с одноплатного компьютера. Для задач компьютерного зрения, то есть непрерывной обработки графической информации, на котором основано движение мобильного робота, необходим компьютер, обладающий относительно большой вычислительной мощностью [2]. Обязательным условием для данного узла так же является наличие удобного интерфейса для работы с пользователем, например, такого как USB-порты в достаточном количестве. В таблице 1 приведены данные о доступных на рынке микрокомпьютерах.

Таблица 1 – Одноплатные компьютеры и их характеристики

Модель	Количество ядер/ частота процессора	Объём памяти оперативного запоминающего устройства	Объём памяти постоянного запоминающего устройства	Количество портов USB, шт.	Ценовой диапазон, тыс. руб.
BBC micro:bit	1 ядро/16 МГц	128 КБ	512 КБ	1	2-4
Raspberry Pi Zero 2 W	4 ядра/ 1 ГГц	512 МБ	Внешний накопитель MicroSD	1	6-8
LattePanda V1.0	4 ядра/1,92 ГГц	2 ГБ	32 Гб	3	16-18
Rock64 Media Board	4 ядра/ 1,5 ГГц	2 ГБ	128 Мб + внешний накопитель MicroSD	3	5-7
Raspberry Pi 3 Model B	4 ядра/ 1,2 ГГц	1 ГБ	внешний накопитель MicroSD	4	9-12
Orange Pi Zero	4 ядра/ 1,35 ГГц	512 МБ	внешний накопитель MicroSD	2	3-5
Banana Pi BPI-W2	4 ядра/ 1,35 ГГц	2 ГБ	8 Гб + внешний накопитель MicroSD	3	10-14

Сравнивая данные различных одноплатных компьютеров, можно сделать вывод о двух наиболее подходящих моделях для данного проекта. Raspberry Pi 3 Model B и Banana Pi BPI-W2. Оба микрокомпьютера обладают достаточно большой вычислительной мощностью, оперативной памятью подходящего объёма. При немного большой стоимости, компьютер Banana Pi BPI-W2

обладает несколько лучшими характеристиками при более высокой цене. Значительным фактором здесь становится количество USB-портов, которых потребуется не менее четырёх для удобной настройки и работы мобильного робота. Таким образом, Raspberry Pi 3 Model B наиболее удачный выбор в качестве центрального управляющего устройства.

Рассмотрим же подробнее характеристики выбранного микрокомпьютера. Этот одноплатный компьютер основан на четырехъядерном процессоре ARM Cortex-A53 с тактовой частотой 1,2 ГГц [19]. Графический процессор VideoCore IV, в свою очередь, является двухядерным, что позволяет обрабатывать HD-видео. Данный миникомпьютер имеет широкий набор выводов и портов для подключения периферийных устройств, включая четыре USB-порта, гнездо Ethernet для интернет-подключения, HDMI-выход для подключения монитора, а также стандартное гнездо 3,5 мм для подключения аудиоустройств и другое. Изображение этого одноплатного компьютера представлено на рисунке 17.

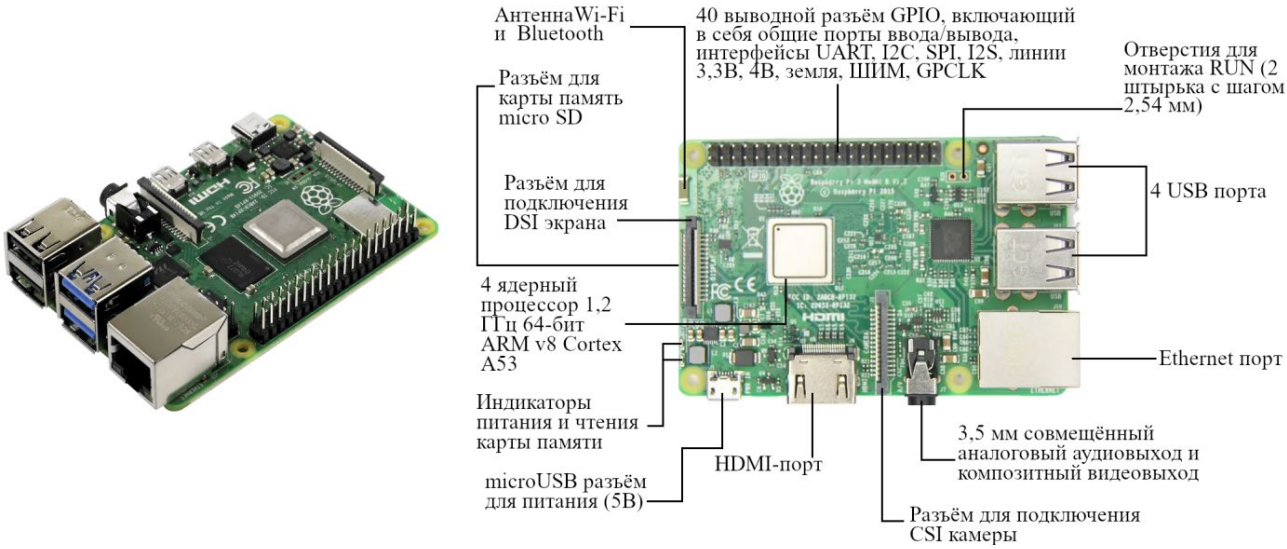

Рисунок 17 – Одноплатный компьютер Raspberry Pi 3 Model B

Теперь приступим к выбору управляющего устройства, а именно к выбору микроконтроллера. Как обозначено выше, данный блок в системе мобильного робота позволяет не только уменьшить нагрузку на ведущее

устройство, но и продолжить работу, если в нём возникла временная неисправность. Наиболее подходящим для данной роли будет семейство микроконтроллеров Arduino. Они обладают набором готовых решений: встроенным программатором, стабилизатором питания, доступных среды разработки и программных библиотек. В рамках семейства Arduino проводить сравнения практически не имеет смысла. Они отличаются только количеством портов ввода/вывода и дополнительных устройств, таких как АЦП, таймеры-счетчики и т.д. В данном проекте используется микроконтроллер Arduino MEGA, который основан на микропроцессоре Atmel ATmega2560 (рисунок 18).

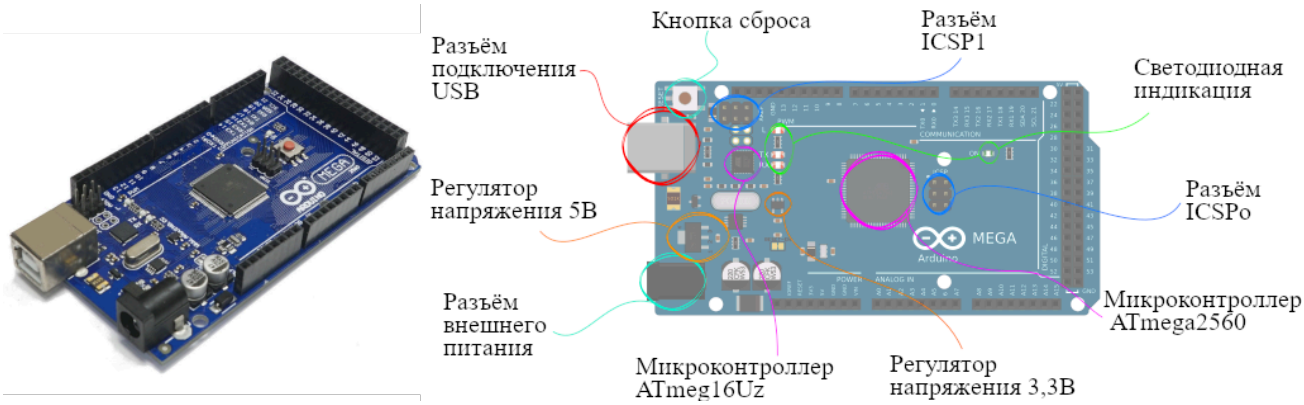

Рисунок 18 – Микроконтроллер Arduino MEGA 2560

Arduino MEGA 2560 – это расширенная модель платформы Arduino, которая представляет собой простой в использовании и гибкий микроконтроллер, подходящий для большинства проектов в области электроники и программирования. Он был разработан на основе микроконтроллера ATmega2560 от Atmel, что делает его более мощным и быстрым, чем многие другие платформы Arduino.

Микроконтроллер имеет 8-битную архитектуру и рабочую частоту до 16 МГц. Максимальный размер программы составляет 256 Кб, а объем EEPROM составляет 4 Кб. Большое количество различных портов и интерфейсов (всего 54 цифровых входа/выхода, 16 аналоговых входов и 4 серийных порта)

позволяют использовать Arduino MEGA 2560 для множества задач, включая управление датчиками и исполнительными механизмами, что и необходимо в нашем случае [17].

В свою очередь, сам Arduino MEGA 2560 имеет USB порт для подключения к компьютеру и загрузки кода программы. Кроме того, на плате имеются кнопки сброса, питания и выбора источника питания.

Выбрав оба управляющих устройства, можно перейти к выбору периферийных гаджетов: клавиатуры, дисплея, компьютерной мыши.

Особого значения они не имеют. Поэтому для ввода информации выберём клавиатуру проводную Lingshe K200**,** и мышь проводную Defender Patch MS-759 (рисунок 19).

Рисунок 19 – Клавиатура и мышь, используемые в проекте

Любое устройство, подключаемое к HDMI порту Raspberry Pi 3, может быть использовано в качестве дисплея, при условии, что разрешение экрана не менее 640x350 и не более 1920x1200. В случае подвижной платформы лучше использовать маленький экран, чтобы не превышать допустимую нагрузку. Поэтому используется совместимый с Raspberry Pi 3 дисплей – 4.3 дюймовый Display Waveshare Raspberry Pi Touch Screen Display Monitor 480x272 HDMI - LCD (рисунок 20).

Рисунок 20 – Display Waveshare Raspberry Pi Touch Screen Display Monitor 480x272 HDMI – LCD

Во время разработки в качестве камеры можно использовать практически любую компактную камеру. Выбор пал на Logitech C270 HD 720p с разрешением 1280×720 (рисунок 21).

Рисунок 21 – Веб-камера Logitech C270 HD 720p

Перейдём к выбору исполнительных механизмов. GM25-370 – это двигатель с коллектором, работающий на постоянном токе. Этот тип двигателя известен своей простотой и надежностью, а также способностью обеспечивать высокий крутящий момент на низких скоростях. Модель GM25-370 специально разработана для использования в небольших приложениях, таких как робототехника, миниатюрные насосы и другие подобные устройства.

Этот электродвигатель имеет встроенный мотор-редуктор с передаточным числом 75 к 1, а так же два датчика Холла. Номинальное напряжение двигатель составляет 9В. Модель GM25-370 имеет компактный размер, диаметр всего 25 мм и длину 37 мм, что делает ее идеальной для использования в ограниченном пространстве. Он имеет максимальное рабочее напряжение 12 В постоянного тока и может обеспечивать крутящий момент до 0,5 Нм при скорости вращения 300 об/мин. Выводы GM25-370 представлены на рисунке 22.

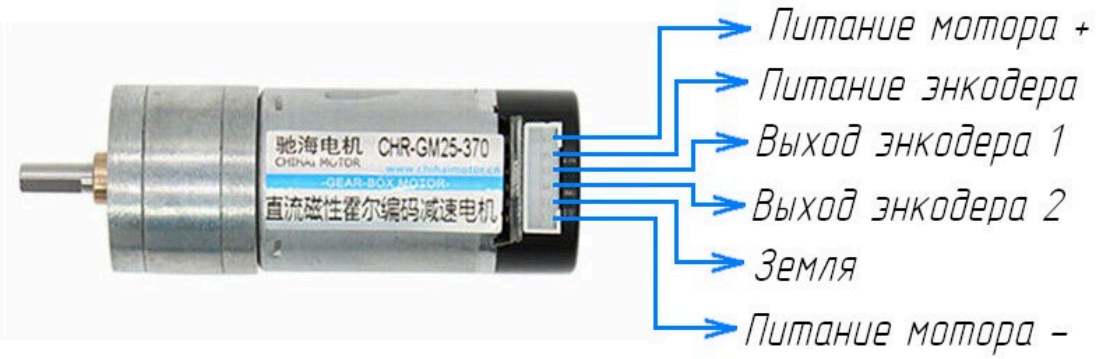

Рисунок 22 – Электромотор GM25-370

Этот мотор выбран, так как его характеристики так же удовлетворяют требованиям.

Теперь перейдём к средству управления электродвигателем. Для этого выбран драйвер L298N – широко распространённое, дешёвое и простое

решение, идеально подходящее для данного этапа разработки. На рисунке 23 представлен внешний вид драйвера и краткое описание всех его составляющих.

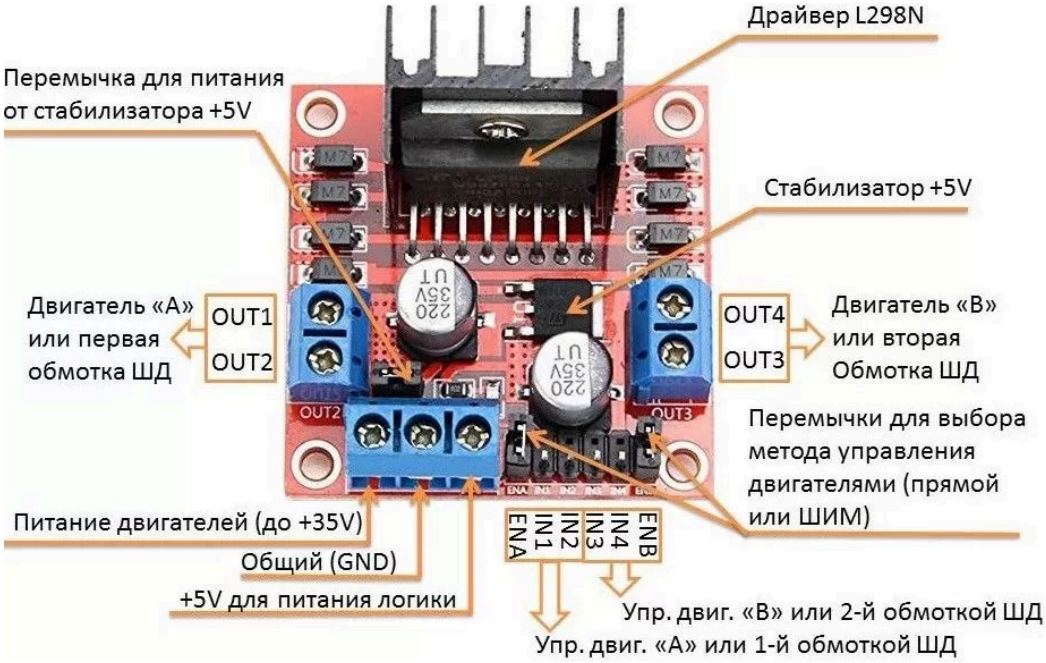

Рисунок 23 – Драйвер L298N

Драйвер двигателя L298N используется для управления двигателями постоянного тока с использованием множества функций. Модуль состоит из двух Н-мостов, что позволяет одновременно подключать один биполярный шаговый двигатель или два щеточных двигателя постоянного тока. Данная схема также предоставляет возможность изменять скорость и направление вращения двигателей. Работа со встроенным модулем L298N довольно проста, поскольку для управления моторами используется только переключение логических уровней на выводах Arduino, для контроля скорости вращения может требоваться генерация ШИМ-сигналов [14].

В качестве элементов питания используются литиевые трёхбаночные аккумуляторы с величиной выходного напряжения 11,1 В «Li-Po 11.8V 3S1P 5000 мА·ч». Их внешний вид представлен на рисунке 24.

Рисунок 24 – Внешний вид аккумуляторной батареи

Не все устройства сигнальной части имеют возможность непосредственного питания от аккумулятора напрямую. Для миникомпьютера и дисплея требуется напряжение питания 5 В, но они не имеют встроенных преобразователей напряжения. Для обеспечения питания миникомпьютера от аккумулятора используется регулируемый импульсный преобразователь напряжения LM2596S (на рисунке 25 слева), а для питания дисплея - нерегулируемый преобразователь KIS-3R33S (на рисунке 25 справа). Микроконтроллер не требует преобразователя, поскольку он уже оснащен встроенным стабилизатором напряжения с возможностью входного напряжения в диапазоне 7-12 В.

Рисунок 25 – Преобразователи напряжения LM2596S слева и KIS-3R33S справа

Микроконтроллер получает питание непосредственно от аккумулятора, благодаря наличию в нем встроенного линейного стабилизатора напряжения. Однако, миникомпьютер и дисплей подключены через импульсные преобразователи, потому что не имеют встроенной стабилизации питания. В результате, на основе выбранных компонентов и ранее описанной структуры, составлена структурная схема мобильной робототехнической платформы (рисунок 26).

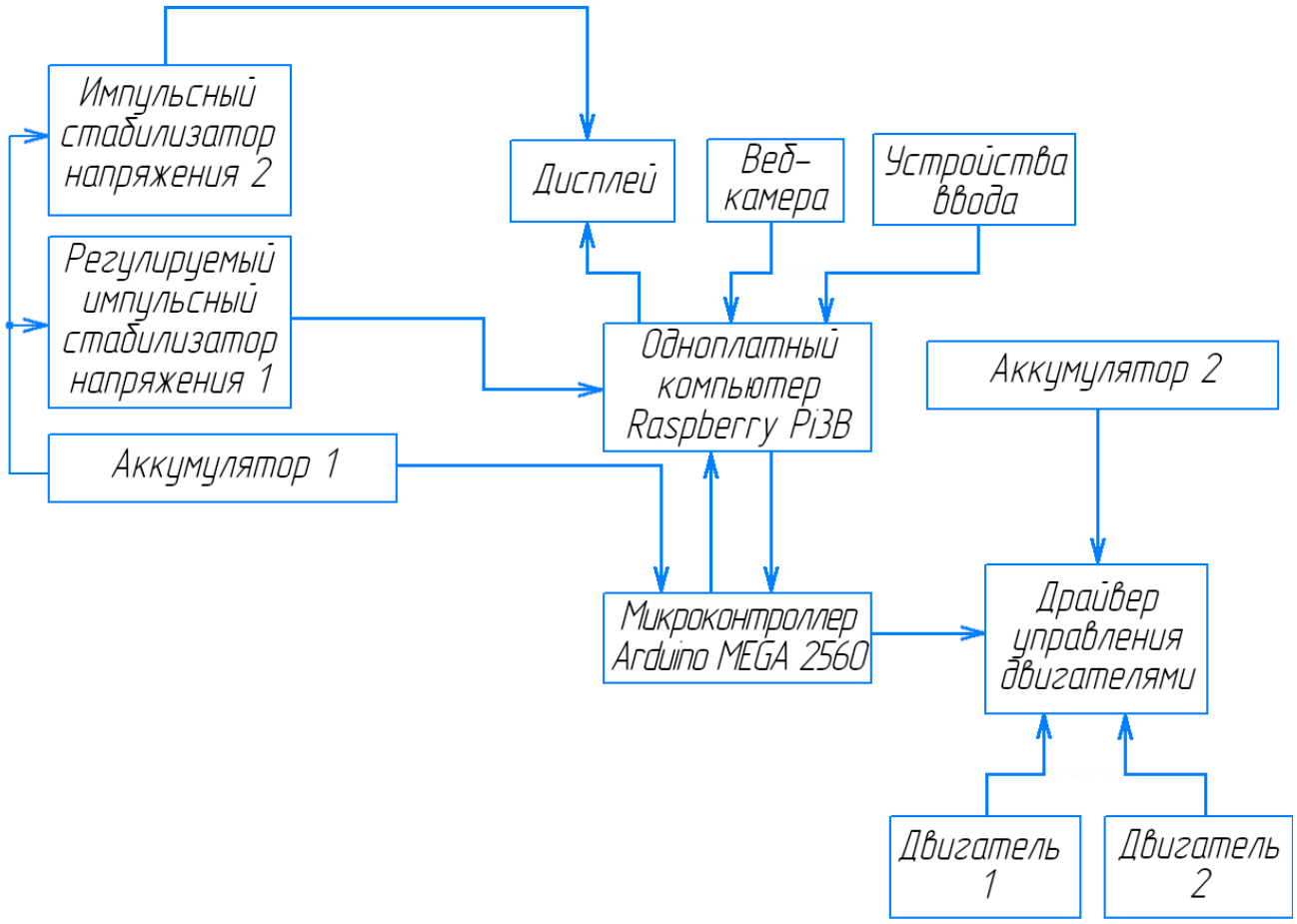

Рисунок 26 – Структурная схема мобильного робота

Таким образом, в данном разделе был произведён выбор необходимых электротехнических компонентов для мобильного робота, а так же создана его структурная схема.

4 Способ связи между узлами платформы

4.1 Выбор способа связи между микроконтроллером и микрокомпьютером

Как видно из структурной схемы на рисунке 3.14, связь микроконтроллера и одноплатного компьютера должна быть двусторонней. Это продиктовано необходимостью стабильной работы, которую можно обеспечить лишь системой с обратной связью. Это возможно обеспечить различными способами.

Один из вариантов такой связи – подключение пинов Arduino к Raspberry с помощью соединительных проводов, а затем создание программы приёма передачи, основанный строго на передаче и регистрации высокого и низкого уровня сигнала, и интерпретации его тем или иным способом. Этот способ имеет ряд недостатков. В случае соединения Arduino и Raspberry Pi через провода через пины, требуется не только подключение проводов правильно, но и настройка каждого устройства, чтобы они могли связываться между собой. Неправильная настройка может привести к тому, что данные не будут передаваться правильно, и могут происходить ошибки или сбои. Передача данных между Arduino и Raspberry Pi через провода могут иметь ограниченную пропускную способность, а значит нельзя передавать большой объем информации. Провода и пины не очень надёжны для потоковой передачи информации, так как имеют люфт при установке, что может привести к искажениям в трансляции[8].

Простым и довольно широко используемым средством передачи информации является универсальный асинхронный приёмо-передатчик, или UART – Universal Asynchronous Receiver-Transmitter [4]. Физическая реализация УАПП в случае связки Arduino-Raspberry Pi3B возможна двумя способами. Первый способ подразумевает соединение RX и TX портов у микрокомпьютера и микроконтроллера. При этом возникает проблема: у двух

устройств разные логические уровни, соответственно будет нужен переключатель уровня напряжения (рисунок 27).

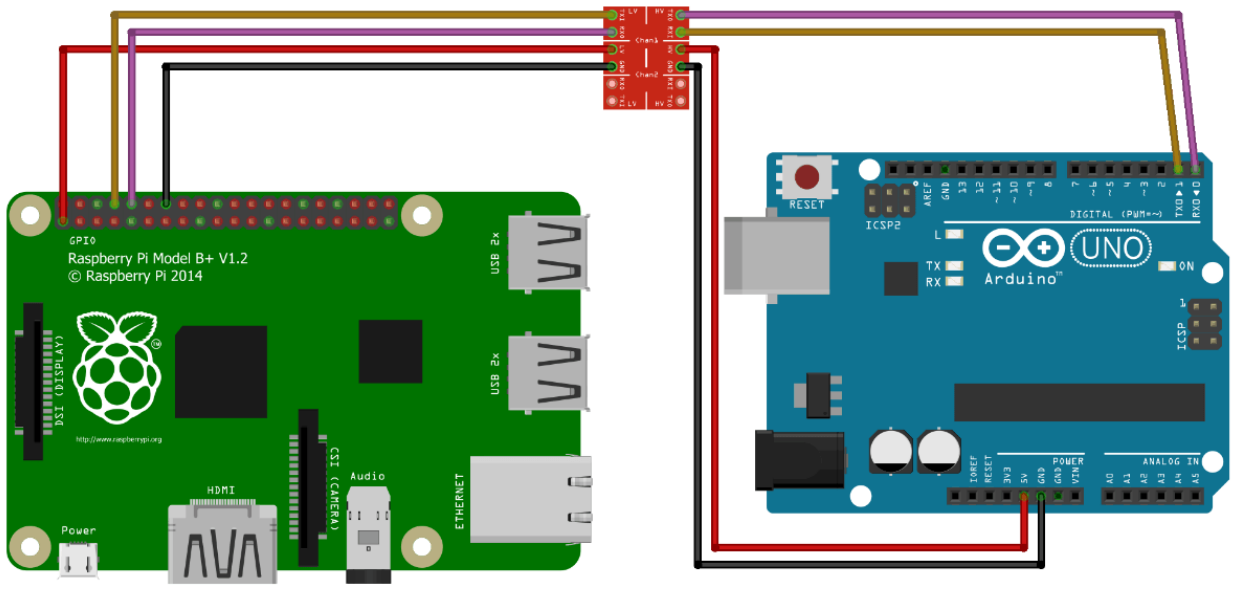

Рисунок 27 – Подключение UART с помощью переключателя уровня

Есть и ещё один вариант. UART-подключение между Arduino и Raspberry Pi 3 может быть установлено через USB-порты обеих плат.

Подключение UART между Arduino и Raspberry Pi 3 с помощью USB имеет ряд преимуществ:

– простота подключения: подключение между Arduino и Raspberry Pi 3 через USB-порты – это простой и удобный способ установить связь между двумя платами. USB-кабели очень распространены;

– низкое энергопотребление: подключение через USB может потреблять намного меньше энергии, чем другие способы связи, например, Ethernet или Wi-Fi. Это важный фактор для работы с ограниченным источником энергии, таким как аккумуляторы;

– устойчивость к помехам: подключение через USB менее подвержено помехам, чем подключение через беспроводную связь. Это уменьшает возможность ошибок при обмене данными и значительно повышает надежность связи. Кроме того, это подключение физически стабильно, в отличие от подключения проводов к пинам;

– кросс-платформенная совместимость: USB-порты являются наиболее распространенным интерфейсом в устройствах более нового поколения. Подключение Arduino и Raspberry Pi 3 через USB позволяет настроить связь между разными операционными системами, что упрощает разработку и отладку мульти-платформенных проектов.

Из всех перечисленных преимуществ можно сделать вывод, что подключение UART между Arduino и Raspberry Pi 3 через USB является одним из самых простых, надежных и быстрых способов связи между платами, что делает его лучшим выбором для данного проекта [16].

4.2 Подготовка протокола связи между микроконтроллером и микрокомпьютером

Когда определён физический способ связи между микроконтроллером Arduino и микрокомпьютером Raspberry Pi, необходимо обеспечить программную передачу значения отклонения от линии к управляющему устройству для последующей выработки управляющего воздействия.

В первую очередь, рассмотрим отклонение от линии. Отклонение формируется программой технического зрения во время работы Raspberry Pi. Вне зависимости от способа определения линии с помощью технического зрения, приводится к виду целого числа. Отклонение от линии сохраняется в целочисленную переменную delta.

Было решено создать два протокола передачи значения отклонения. Для того чтобы начать работу над непосредственно программами, необходимо определиться с их структурой.

4.2.1 Символьная передача отклонения

Простейшим способом передачи отклонения является передача заранее заготовленных команд. В таком случае заранее заготовленная команда, которая адекватно интерпретируется обеими сторонами связи, посылается в зависимости от значения, которое принимает переменная отклонения в данный момент времени.

Чтобы формировать сообщение за наименьшее время, логичнее всего предельно уменьшить объём передаваемого сообщения. Такой подход позволяет в том числе уменьшить и время, затрачиваемое на расшифровку сообщения.

Достаточно лаконичным решением данного вопроса является передача отдельных символов в качестве законченной команды. Возможности символьной передачи ограничены 127 символами, но всё же такой объём команд позволяет создать множество различных случаев или кейсов.

Каждый кейс – это отдельное событие. Он может быть связан не только с управляющим воздействием, но и, например, системой старт-стоп или алгоритмом поиска потерянной линии.

Рассмотрим кодировку команд (таблица 2).

Таблица 2 – Символьная кодировка кейсов

Номер кейса	Десятичное значение символа	Передаваемый символ char	Кейс
1	70	F	Двигаться вперёд
2	82	R	Резкий поворот направо
3	69	E	Менее резкий поворот направо
4	80	P	Плавный поворот направо
5	114	r	Медленный поворот направо
6	76	L	Резкий поворот налево
7	75	K	Менее резкий поворот налево
8	86	V	Плавный поворот налево
9	108	l	Медленный поворот налево
10	83	S	Начать движение
11	70	F	Прекратить движение

Блок-схема алгоритма передачи представлена на рисунке 28.

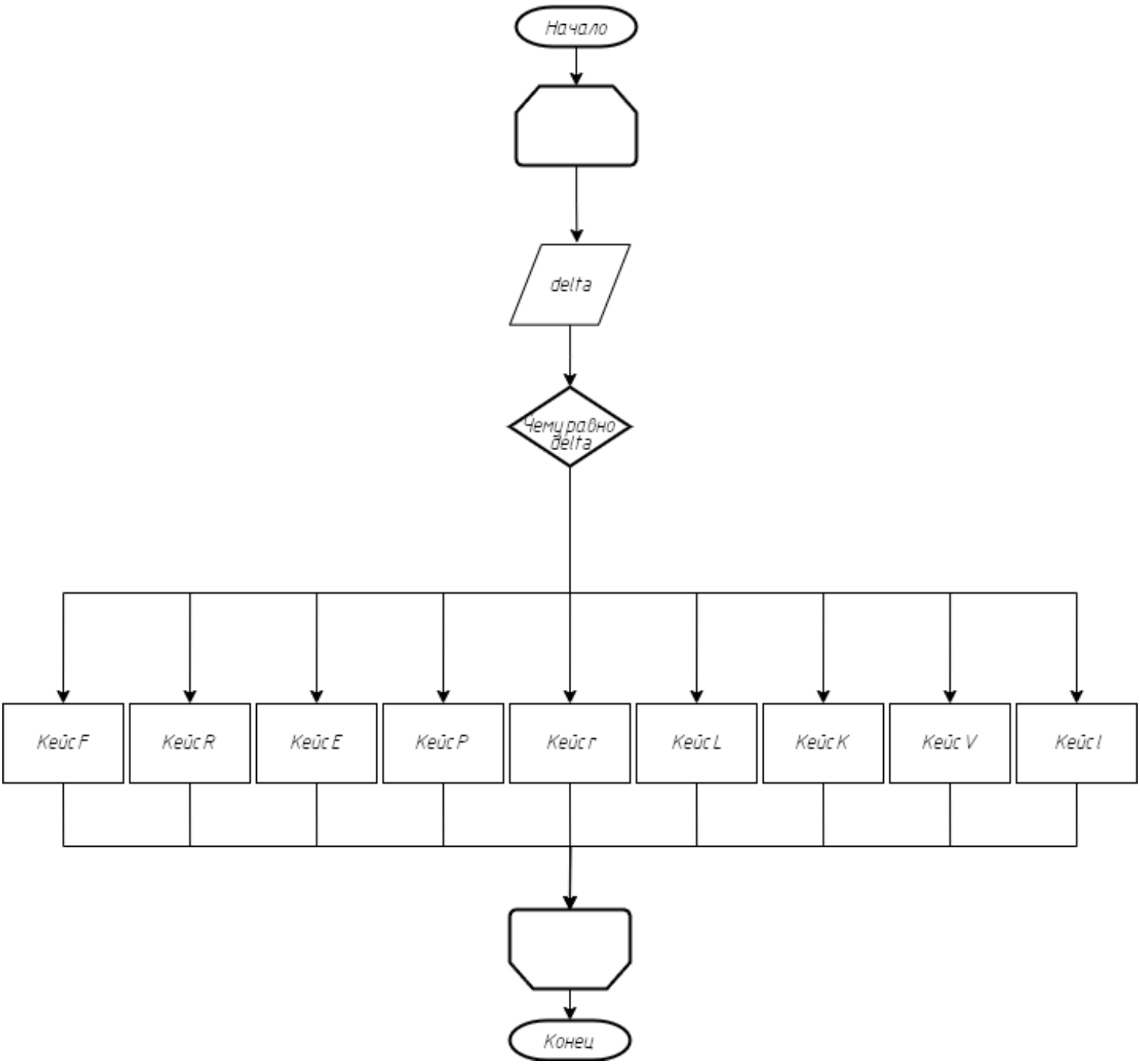

Рисунок 28 – Блок-схема алгоритма символьной передачи отклонения

Для получения данных используется небольшая и простая программа, которая, при наличии данных в последовательном порту, считывает их, и в зависимости от символа, устанавливает скорость. Данная программа небольшого объёма, проста в понимании, поэтому она не нуждается в блок-схеме алгоритма работы.

Данный протокол передачи имеет несколько преимуществ:

– он прост в реализации и понимании;
– обеспечивает малое время формирования и чтения пакета данных, одной команды;
– позволяет легко добавить команды, несвязанные с движением в обычном режим. Например, легко добавить алгоритм возврата на линию.

Вместе с тем, данный алгоритм имеет и целый ряд недостатков. Наиболее очевидным является ступенчатость регулирования, ведь при передаче кейсов невозможно в реальном времени менять напряжение на моторах, исходя прямо из отклонения. Другим минусом является нагрузка на микрокомпьютер, который и так занят обработкой изображения, так как ему требуется выбрать кейс отправки. По этой причине приготовлен и другой протокол передачи.

4.2.2 Непосредственная передача отклонения

Другим решением передачи отклонения к исполнительным механизмам является непосредственная передача значения отклонения, и последующая интерпретация микроконтроллером для управления электродвигателями колёс.

У этого решения есть несколько преимуществ. В первую очередь, возможно плавное регулирование скоростями колёс, что обеспечит плавное движение и уменьшит вероятность ошибки при движении. Кроме того, это позволит разгрузить Raspberry Pi, что расходует свою мощность на обработку изображения в реальном времени.

Перед созданием непосредственно программ, опишем их структуру. Начнём с программы передачи. Передать через UART за одну итерацию программы возможно только один символ. Соответственно, чтобы передача не состояла из сплошного потока символов, их необходимо разделять между собой. Для этого будет служить специальный символ. Так как отклонение из управляющей программы выходит в виде целого числа, его необходимо превратить в массив типа char, где каждая ячейка массива содержит одну из

цифр числа отклонения. Затем последовательно передать массив, после которого передать символ окончания передачи. Блок-схема алгоритма передачи представлена на рисунке 29.

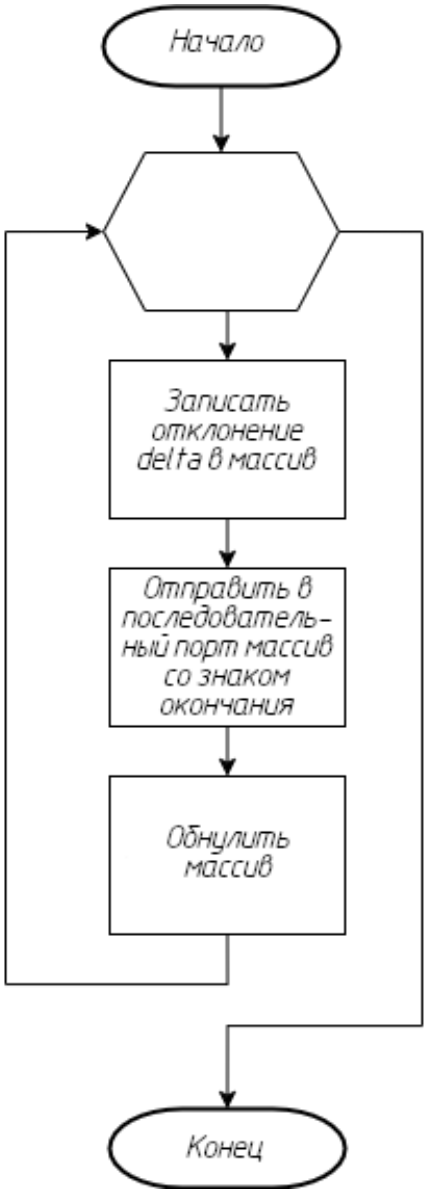

Рисунок 29 – Блок-схема алгоритма передачи непосредственно отклонения

Для Arduino стоит обратная задача. Каждый отдельный символ помещать последовательно в массив, пока не придёт символ окончания передачи. После этого символа превратить массив обратно в целое число, и уже дальше работать с ним в программе управления моторами. Блок-схема алгоритма приёма представлена на рисунке 30.

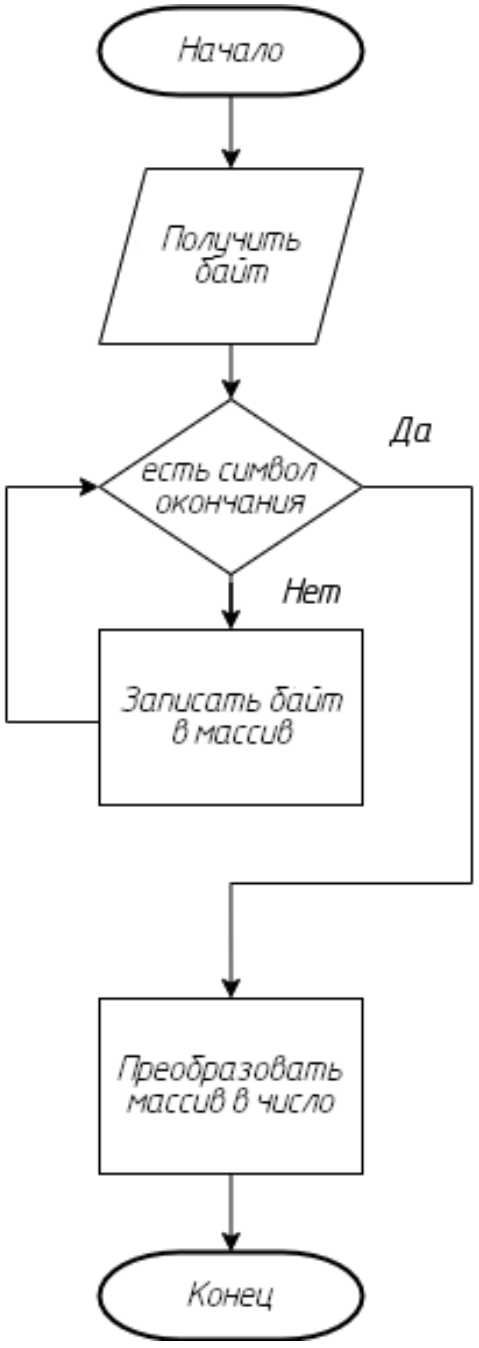

Рисунок 30 – Блок-схема алгоритма приёма непосредственно отклонения

Таким образом, в данном разделе был выбран способ связи между микроконтроллером и одноплатным компьютером. Кроме того, подготовлены два алгоритма приёма-передачи информации между узлами платформы.

5 Разработка электрической схемы соединений

Схема электрическая соединений состоит из следующих функциональных блоков:

- одноплатный компьютер Raspberry Pi 3 model B;
- микроконтроллер Arduino MEGA 2560;
- мышь проводная Logitech M90;
- клавиатура проводная Logitech 120K;
- веб-камера Logitech C270 HD 720p;
- дисплей Waveshare Raspberry Pi Touch Screen Display Monitor 480x272 HDMI – LCD;
- драйвер управления двигателями L298n;
- двигатели GM25-370 со встроенными датчиками Холла;
- литиевые трёхбаночные аккумуляторы «Li-Po 11.8V 3S1P 5000 мА·ч»;
- регулируемый DC-DC преобразователь напряжения LM2596S;
- не регулируемый DC-DC преобразователь напряжения KIS-3R33S.

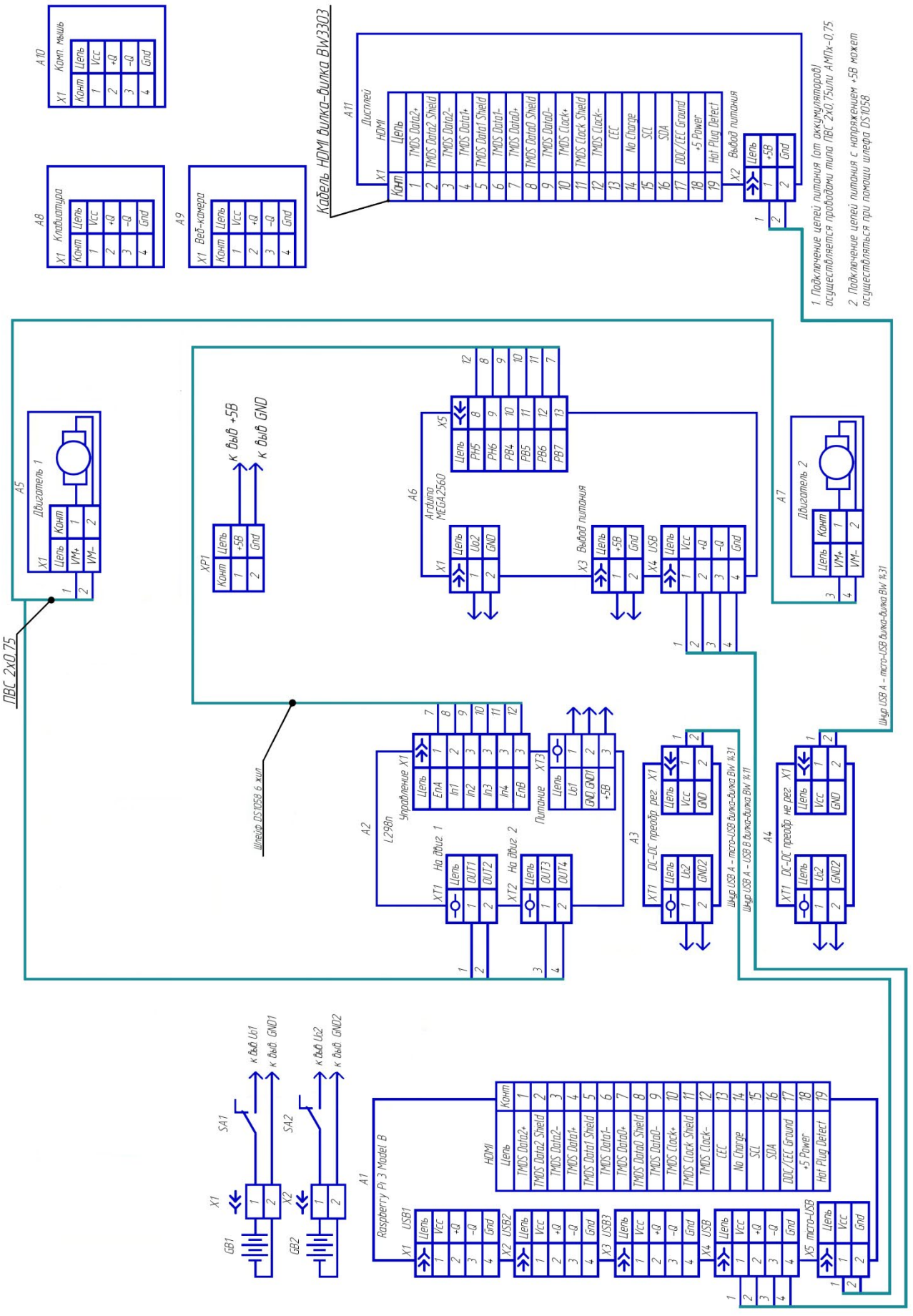

Рисунок 31 – Схема электрическая соединений

Для связи между одноплатным компьютером и микроконтроллером используется шнур со вилками USB A и USB B. Для соединения не регулируемого преобразователя и дисплея, а также регулируемого преобразователя и одноплатного компьютера используется шнур со вилками USB A и microUSB. Кабелем HDMI соединяются одноплатный компьютер и дисплей. На рисунке 31 представлена схема электрическая соединений.

Таким образом, в данном разделе была составлена схема электрическая соединений, включающая все электротехнические компоненты: логическая, силовая части, элементы управления, кабели соединения.

6 Разработка конструкции

Получив представление обо всех элементах мобильного робота, возможно разработать его конструкцию, а затем произвести сборку.

Определимся с требованиями конструкции. Устойчивость, надёжность, лёгкий доступ ко всем компонентам, низкая цена, малый вес.

Для этого будут использованы доступные материалы. Конструкция будет состоять из двух ярусов. Нижний, где будет установлено большинство оборудования: элементы логики, силовые элементы, элементы питания и верхний ярус, с камерой технического зрения, монитором отладки и при необходимости компьютерной мыши.

В результате, в качестве основы выбрана фанера, как лёгкий в обработке и дешёвый материал, для дополнительного укрепления алюминиевые уголки и оргстекло. В качестве движителя, будут использоваться обрезиненные ведущие колёса большого диаметра с хорошим сцеплением с поверхностью и ролики малого диаметра. Поворот будет обеспечен за счёт изменения разности скорости ведущих колёс, и свободному повороту роликов на 360 градусов.

При такой конструкции, где на верхнем ярусе мало элементов, возможен лёгкий доступ к большинству электронных и электрических компонентов; обеспечивается оптимальный обзор камере; достигается высокая устойчивость за счёт низкого центра тяжести; возможен разворот на месте, то есть обеспечена высокая манёвренность [20]. Сборочный чертёж мобильного робота представлен на рисунке 32.

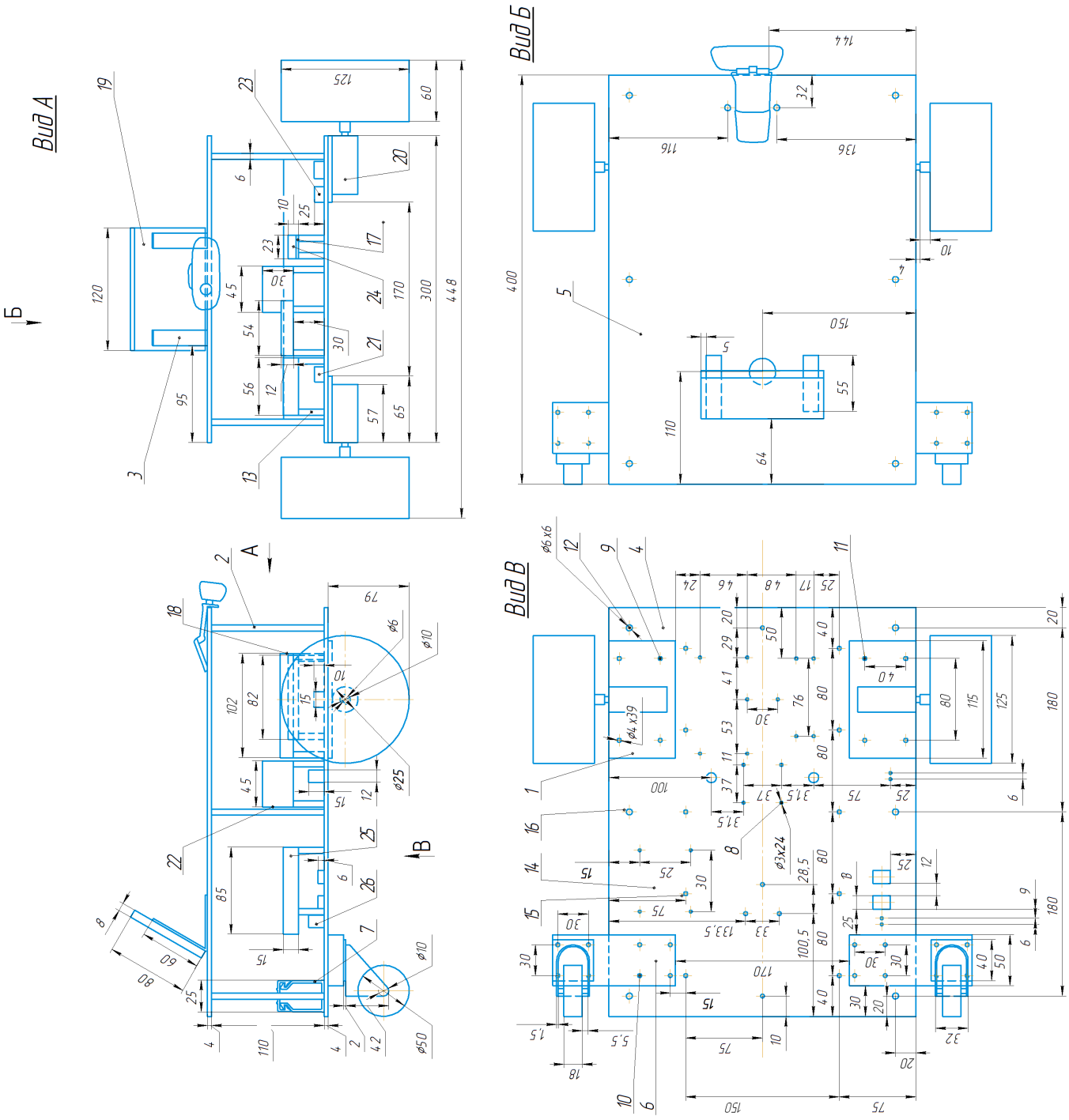

Рисунок 32 – Сборочный чертёж мобильного робота

Таким образом, в данном разделе была разработана конструкция мобильного робота. Выбран способ передвижения, разработана несущая часть и общее устройство, расположение компонентов. Разработан сборочный чертёж мобильного робота.

7 Разработка управления исполнительными механизмами

Для двух алгоритмов передачи потребуется разработать два алгоритма управления электромоторами, в силу значительной разницы между ними. Однако базовая часть для обоих алгоритмов будет одинаковой.

Так как будет подготовлена базовая часть, единая для двух алгоритмов, то для удобной работы стоит оформить её как отдельный функциональный блок при помощи объектно-ориентированного программирования [3]. Подготовим структуру программы.

Для полной работы нам понадобятся следующие методы: включение моторов, выключение моторов, установка направления вращения, установка скорости. Все они будут с общим доступом, так как требуется использовать их в программе.

Полный код класса Motor, отвечающий за управления электромоторами, входящий в состав программы управления, приведён в Приложении Б.

7.1 Разработка алгоритма управления для символьной передачи отклонения

Для начала приготовим алгоритм управляющего воздействия, написанный в среде разработки Arduino ADE. Нужно провести две проверки, приходят ли коды от Raspberry Pi, или нет, и в зависимости от этого двигаться или нет. Это будет обеспечено с помощью двух операторов условия if и функции Serial.available. Внутри условия «в последовательном порту есть данные» распознаём их с помощью функции Serial.read и записываем в переменную типа char «getCh». Для всех кейсов необходимо так же создать условия, где в зависимости от пришедшего символа, будет установлена скорость электромоторов. Более эффективным подходом в данном случае будет оператор switch. Полный код программы управления представлен в листинге 1.

Листинг 1

```
if (Serial.available()) {
    char getCh = Serial.read();

    switch (getCh) {
      case 'F':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(difMotorSpeed);
      break;
      case 'R':
      myMotor_right.set_speed(0);
      myMotor_left.set_speed(difMotorSpeed);
      break;
      case 'P':
      myMotor_right.set_speed(lowSpeed);
      myMotor_left.set_speed(difMotorSpeed);
      break;
      case 'E':
      myMotor_right.set_speed(midSpeed);
      myMotor_left.set_speed(difMotorSpeed-5);
      break;
      case 'r':
      myMotor_right.set_speed(underTopSpeed);
      myMotor_left.set_speed(difMotorSpeed-10);
      break;
      case 'L':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(0);
      break;
      case 'V':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(coef * lowSpeed);
      break;
      case 'K':
      myMotor_right.set_speed(topSpeed-5);
      myMotor_left.set_speed(coef * midSpeed);
      break;
      case 'l':
      myMotor_right.set_speed(topSpeed-10);
      myMotor_left.set_speed(coef * underTopSpeed);
      break;
    }
}
else if (Serial.available() == -1)
```

```
{
myMotor_right.set_speed(0);
myMotor_left.set_speed(0);
}
```

Переменная целочисленная topSpeed хранит в себе максимальную скорость моторов, из которой, путём вычитания вычисляются остальные скорости, такие как lowSpeed – минимальная скорость; midSpeed – средняя скорость; underTopSpeed – почти максимальная скорость.

Здесь используется так же difMotorSpeed. Дело в том, что два оба мотора не одинаковые. При одном напряжении они выдают разную частоту вращения. Экспериментально выяснено, что равная скорость вращения достигается при поправочном коэффициенте coef, равным 0,81. То есть скорость левого мотора занижается, чтобы соответствовать скорости правого мотора. Дополнительно скорость моторов регулируется для случаев резкого поворота.

Полный листинг программы управления символьного алгоритма представлен в Приложении Б.

Теперь перейдём к программированию Raspberry Pi. В случае с символьной передачей, микроконтроллер имеет очень ограниченную информацию о текущем положении мобильного робота относительно линии. Как либо регулировать движение в этом случае практически невозможно или бессмысленно. Соответственно, роль регулирования должна лечь на Raspberry Pi.

Для управления движением логичнее всего использовать пропорционально-интегрально-дифференциальный регулятор или его производные: П, ПД, ПИ регуляторы [15].

Наиболее простым, практически готовым решением для данного алгоритма является П-регулятор. Пропорциональный регулятор учитывает только текущее положение относительно линии, только исходя из этого

В нашем случае уставка, целевое значение, это 0, то есть отсутствие отклонения от линии. Переменная отклонения принимает значения от -320 до

320. Всего существует девять кейсов, поэтому для начала разобьём диапазон 640 на девять равных частей, от «резко вправо» до «резко влево» (приведено в таблице 4.2.1.1). Каждая из частей будет примерно равна 71.

Так как кейсов движения много, необходимо обеспечить максимальное быстродействие, и, желательно, удобную для восприятия и коррекции форму. Для этого создадим трёхмерный массив, который будет содержать границы для каждого из кейсов, а так же сам код команды для передачи (листинг 2).

Листинг 2

```
int cases[][3] = {
    { SHARP_LEFT, 249, 320 },
    { LESS_SHARP_LEFT, 178, 249 },
    { MID_LEFT, 122, 178 },
    { LOW_LEFT, 52, 122 },
    { FORWARD, -52, 52 },
    { LOW_RIGHT, -122, -52 },
    { MID_RIGHT, -178, -122 },
    { LESS_SHARP_RIGHT, -249, -178 },
    { SHARP_RIGHT, -320, -249 },
};
```

Теперь добавим цикл, который будет последовательно перебирать элементы массива, с нулевого по восьмой включительно. В самом цикле установим проверку переменной driver. Это переменная результат отклонения, прошедшего через П, ПД или ПИД-регуляторы. Проверка будет производиться сравнением driver и двух чисел трёхмерного массива, означающих границы каждого кейса. Программа приведена в листинге 3.

Листинг 3

```
for (int i = 0; i < 9; ++i)
{
   if(driver > cases[i][1] && driver < cases[i][2])
    {
        serialPutchar(serialPort, cases[i][0]);
```

```
                break;
            }
        }
```

Результат использования только П-регулятора оказался неудовлетворительным. Мобильная платформа двигалась вдоль линии, но её, ограниченные несовершенным алгоритмом передачи, дополнительно получали негативное воздействие от ограниченного регулирования. Здесь был оценен недостаток самостоятельного П-регулятора: мобильный робот вместо прямой траектории выполнял синусообразное движение вдоль линии. Схематично, движения мобильного робота с помощью П-регулятора представлены на рисунке 33.

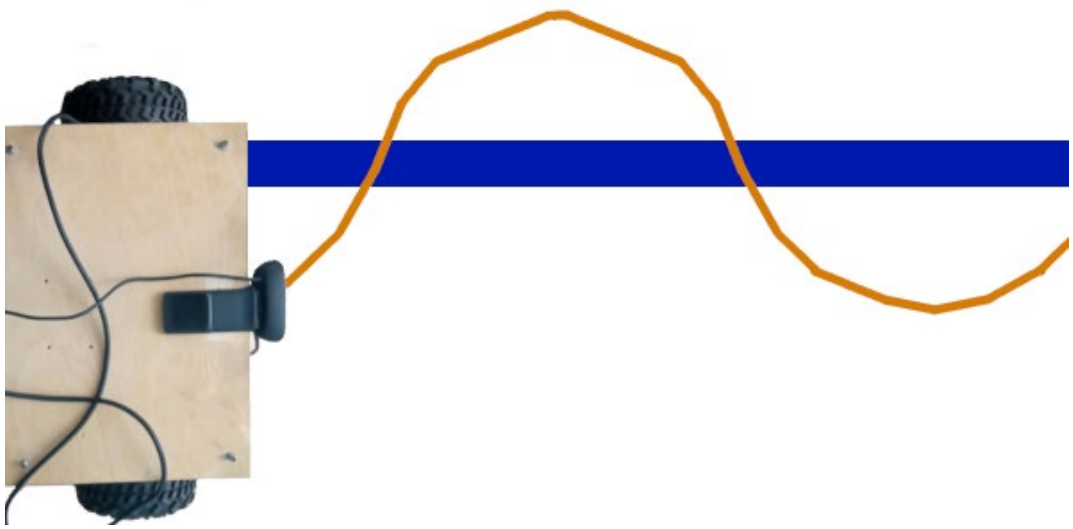

Рисунок 33 – Движение робота с помощью символьной передачи и П-регулятора

Чтобы улучшить движение мобильной платформы вдоль линии, была добавлена дифференциальная составляющая. Дифференциальная составляющая – это разность между текущей и предыдущей ошибкой.

Ошибки для дифференциального регулятора берутся через одинаковые периоды времени. Таким образом, разница между текущей и предыдущей ошибками определяет скорость изменения ошибки.

Увеличение скорости изменения ошибки повышает влияние дифференциальной составляющей. Д-регулятор выполняет две функции: снижает резкость поворотов, помогая П-регулятору, и делает крутые въезды на линию пологими.

Для работы Д регулятора нужно знать текущую и предыдущую ошибки. Запишем их в переменные currentError для текущей и previousError для предыдущей ошибок. Пропорциональный и дифференциальный коэффициенты сохранены в переменные coefP и coefD. Код ПД-регулятора представлен в листинге 4.

Листинг 4

```
currentError = delta;
proportional = currentError * coefP;
differential = (currentError - previousError) * coefD;
driver = proportional + differential;
previousError = currentError;
```

После применения и настройки Д-составляющей, движение стало более пологим. Мобильный робот после нескольких синусообразных движений на линии начинал стабилизироваться. Схематичное движение робота представлено на рисунке 34.

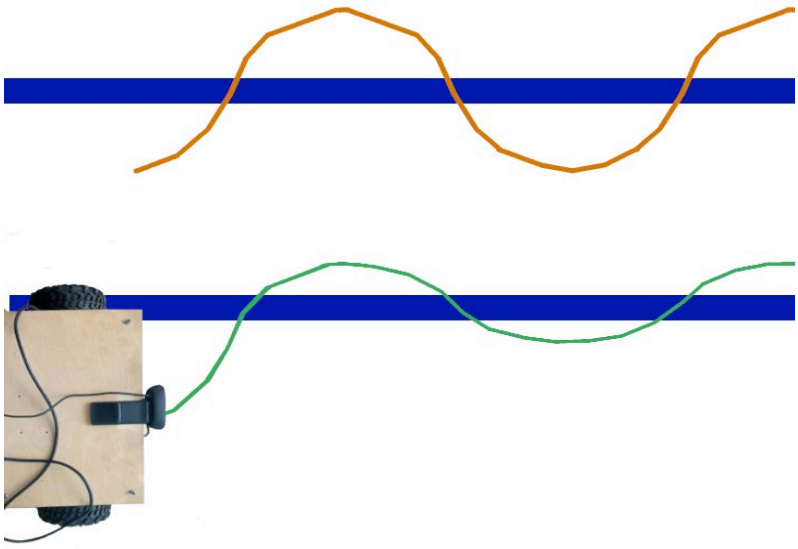

Рисунок 34 – Схематичные траектории движения при использовании П- и ПД-регуляторов

Однако любой шум (например, сильный блик линии, синие объекты в кадре) вызывали значительное отклонение в движении. Поэтому необходимо было ввести последнюю составляющую. Интегральная составляющая – сумма нескольких последних ошибок.

Скорость реакции И-регулятора зависит от количества ошибок в сумме. Если имеется малое число ошибок, то сумма будет недостаточной для надлежащего выполнения регулирования. В случае большого количества ошибок, сумма ошибок приведет к заметной инерционности воздействия.

Возьмём 10 ошибок. Роль интегрального регулятора заключается в исправлении статических и слабых ошибок, а также в устранении нежелательного воздействия шумов на Д-регулятор. С помощью И-регулятора ПД-регулятор может выровнять центр робота с центром линии как на прямых участках, так и на поворотах.

Для сохранения ошибок необходимо создать массив. Назовём его bufError. В каждую ячейку массива должна записываться ошибка, при чём должна происходить постепенная перезапись массива. Модернизируем

программу ПД-регулятора. Программа ПИД-регулятора представлена в листинге 5.

Листинг 5

```
if (i >= 10) {i = 0;}
    bufError[i] = delta;
    float summ = 0;
    for (auto j:bufError) { summ += j; }
    float proportional = bufError[i] * coefP;
    float integral = summ * coefI;
    float differential = (bufError[i] - bufError[(i + 9)
% 10]) * coefD;
    driver = proportional + integral + differential;
    i++;
}
```

Первые три строки кода отвечают за хранение значения ошибок регулирования. Первая строка сравнивает текущее значение счётчика ошибок i с 10 и при равенстве устанавливает его в 0, что позволяет перезаписывать предыдущие значения ошибок по кругу. Вторая строка сохраняет текущее значение ошибки регулирования в массиве bufError на позицию i. Третья строка вычисляет сумму всех значений в массиве bufError и сохраняет в переменной summ.

Далее вычисляются значения пропорциональной, интегральной и дифференциальной компонент. Значение пропорциональной компоненты proportional вычисляется умножением значения ошибки регулирования на коэффициент пропорциональности coefP.

Значение интегральной компоненты integral вычисляется умножением значения суммы ошибок регулирования на коэффициент интегральности coefI.

Значение дифференциальной компоненты differential вычисляется, вычитая текущее значение ошибки из значения ошибки, сохраненного в буфере на позиции (i + 9) % 10 и умножая на коэффициент D (coefD).

Затем значения всех трех компонент просуммируются и результат сохраняется в переменной driver, которая управляет автоматизированной системой.

Наконец, значение счетчика ошибок i увеличивается на единицу, что обеспечивает перезапись массива bufError по кругу.

Полный код программы символьного управления на Raspberry Pi представлен в приложении А.

Результат применения настроенного ПИД-регулятора схематично представлен на рисунке 35.

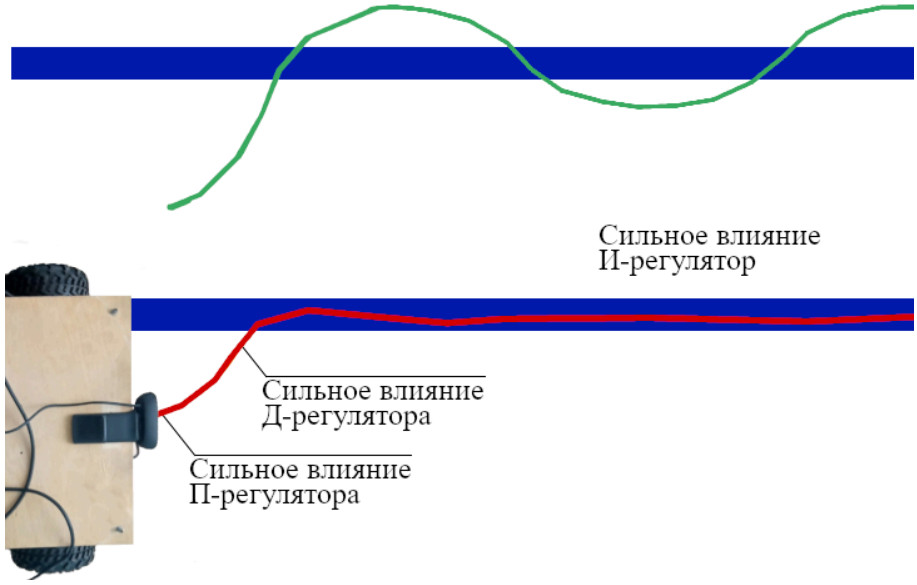

Рисунок 35 – Сравнение движения с помощью ПД и ПИД-регуляторв

Данное движение было признано удовлетворительным. Дальнейшее улучшение алгоритма может заключаться в увеличении количества кейсов для более плавного регулирования. Однако более перспективным является использование алгоритма непосредственной передачи отклонения.

7.2 Разработка алгоритма управления для непосредственной передачи отклонения

Для начала опишем код передачи числа. Как упоминалось ранее, отклонение от линии записывается в целочисленную переменную delta. Для передачи числа удобнее всего использовать функцию serialPrintf. Функция serialPrintf библиотеки WiringPi языка программирования C++ используется для отправки форматированных данных через последовательный порт. Можно сказать, что serialPrintf эмулирует системную функцию printf для последовательного порта.

Для работы требует три параметра. Последовательный порт, куда будет вестись отправка, форматируемое сообщение, и добавка к этому сообщению. Форматируемое сообщение в нашем случае будет составлять только знак окончания передачи. Добавленная я же часть – число, которое требуется передать, отклонение. Для данной функции требуется число, записанное в массив. Поэтому, перед отправкой, необходимо записать delta в массив с помощью функции sprintf.

Затем, чтобы не переписывать записанное в буфер bufToSend число, каждый из элементов массива необходимо обнулить. Тогда не возникнет проблем при переходе от однозначных к двухзначным и трёхзначным числам, и обратно. Код программы передачи представлен в листинге 6.

Листинг 6

```
sprintf(bufToSend,"%d", delta);

serialPrintf(serialPort, "%s\n", bufToSend);
bufToSend[0] = 0;
bufToSend[1] = 0;
bufToSend[2] = 0;
bufToSend[3] = 0;
```

Теперь перейдём к получению числа на платформе Arduino. Начало программы получения переданного числа аналогично алгоритму для символьной передачи. Это условие if (Serial.available), для проверки наличия информации в последовательном порту.

Для записи числа, у которого есть чёткий знак окончания передачи, в нашем случае \n, можно воспользоваться функцией Ардуино «Serial.readBytesUntil». Функция Serial.readBytesUntil в Ардуино используется для чтения байтов из последовательного порта в буфер до заданного символа. Синтаксис функции выглядит следующим образом: Serial.readBytesUntil(terminator, buffer, length), где:

- terminator – символ, который служит ограничителем для чтения байтов из Serial порта;
- buffer – массив, в который будут записаны прочитанные байты;
- length – максимальное количество байтов, которые могут быть прочитаны из последовательного порта и записаны в буфер.

В результате выполнения функции, в массив buffer будут записаны байты, прочитанные из Serial порта до символа terminator. Если terminator не был найден в прочитанных байтах до достижения максимального количества length, то чтение будет прервано, а buffer будет содержать только часть байтов.

Для работы этой функции создадим глобальную переменную bufNum, в которой будет содержаться массив, куда будет записываться информация, пришедшая в последовательный порт.

Саму запись будем производить в цикле while. Запись будет производиться, пока от Raspberry Pi приходят данные. Знак отделения чисел в передаче поставлен «\n». Создан массив bufNum, куда будут записываться числа, и установлен максимальный объём с помощью sizeof().

Затем полученный буфер необходимо преобразовать в целое число для дальнейшей работы с ним. В этом нам поможет функция atoi Ардуино. Функция atoi() (ASCII to Integer) в языке Arduino преобразует массив, содержащий числовое значение, в целое число. Синтаксис функции выглядит

так: int atoi(const char* string). Функция принимает один аргумент – указатель на строку string, которую нужно преобразовать в число. Возвращает она целое число типа int.

После записи числа в переменную getValue, обнуляем массив bufNum. Тогда не возникнет проблем при переходе от однозначных к двухзначным и трёхзначным числам, и обратно. Код программы передачи представлен в листинге 7.

Листинг 7

```
if (Serial.available() > 0) {

    while (Serial.readBytesUntil('\n', bufNum,
    sizeof(bufNum))) {
    int getValue = atoi(bufNum);

    bufNum[0] = 0;
    bufNum[1] = 0;
    bufNum[2] = 0;
    bufNum[3] = 0;
    bufNum[4] = 0;
    bufNum[5] = 0;
}
```

Полный код программы расчёта отклонения от центра линии и непосредственной передачи отклонения на Raspberry Pi представлен в Приложении В.

Теперь опишем алгоритм управляющего воздействия. В основе регулирования движения здесь вновь лежит ПИД-регулирования. Все действия по созданию ПИД-регулятора описаны ранее и ничем не отличаются, кроме выбора коэффициентов. В данной программе использован ПД-регулятор. Дело в том, что движение мобильного робота не очень быстрое, свои ограничения накладывают вычислительные мощности Raspberry Pi. Кроме того, мобильная платформа имеет относительно значительную массу. Всё это приводит к тому, что существуют инерция такого уровня, что сама является интегрирующей составляющей, не дающей совершать мгновенные, резкие передвижения. И

введение интегральной составляющей программно приводит лишь к дестабилизации движения.

Управление двигателями с помощью драйвера описано в пункте 3 «Разработка структурной схемы». Там обозначено, что управление скоростью вращения двигателя осуществляется с помощью широтно-импульсной модуляции сигнала или ШИМ-сигнала. Разрядность системных ШИМ микроконтроллера Arduino MEGA2560 составляет 8 разрядов, то максимальное значение его 2^8 или 255. Это свойство отражено в программе.

Общее значение управляющего воздействия после расчёта в ПД-регуляторе больше чем то значение, которое можно установить в качестве скорости электромотора в программе. В частности, только при пропорциональном регулировании максимальное значение составляет 320 и минус 320 соответственно. При ПД-регулировании, это число ещё увеличивается.

Кроме того, существует иная проблема. При минимальном отклонении на моторы не будет поступать управляющего воздействия, так как отклонение опосредованно поступает в качестве управляющего воздействия на электромоторы. Значит необходимо установить базовую скорость, к которой будет прибавляться управляющее воздействие. Соответственно, управляющее воздействие должно быть меньше базовой скорости, иначе число превысит границы, доступные для ШИМ. Для понимания, приведём пример. В данной программе для управления используется число delta, которое может достигать значения в 320. Базовая же скорость составляет 85. В результате сложение базовой скорости и значения delta приведут не к максимальной скорости вращения мотора, достигаемой при значении 255, а всего лишь к 149, то есть к примерно к 58% от скорости вращения моторов из-за переполнения переменной, отвечающей за ШИМ.

В связи с этой особенность, необходимо нормировать управляющее воздействие. Это возможно сделать с помощью функции map в Arduino. Функция map в Arduino позволяет изменять диапазон значений переменной на

другой диапазон. Эта функция имеет следующий синтаксис: map(value, fromLow, fromHigh, toLow, toHigh), где:

- value: значение, которое нужно преобразовать;
- fromLow: минимальное значение исходной переменной;
- fromHigh: максимальное значение исходной переменной;
- toLow: минимальное значение, которое нужно получить;
- toHigh: максимальное значение, которое нужно получить.

Однако изменение коэффициентов пропорционального и дифференциального приводят к изменению минимального и максимального значения исходной переменной, поэтому единожды установить эти границы во время разработки невозможно. Рассчитаем их, исходя из максимального значения величин пропорционального и дифференциального отклонений (листинг 8).

Листинг 8

```
float boardUp = 320 * coefP + 25 * coefD;
float boardDown = -320 * coefP + -25 * coefD;
```

Теперь можно ограничить управляющее воздействие driver следующим образом:

driver = map(driver, boardDown, boardUp, -65, 65);

Однако функция map не гарантирует, что нормированное число driver не выйдет за обозначенные границы, в данном случае минус 65 и 65. Выход за границы может случиться в том случае, когда переменная управляющего воздействия превысит границы boardDown и boardUp. Например это может произойти, когда дифференциальная ошибка превысит значение 25 при слишком резком повороте или случайном шуме. Для этого необходимо жёстко ограничить управляющее воздействие с помощью функции constrain:

driver = constrain(driver, -80, 80);

Где последние два числа – те значения, которые превысить ограничиваемое число не сможет. Оставим его границы чуть шире, в случае

если необходимость в очень резком повороте всё же будет. При этом значения этих чисел всё равно не превышают значения базовой скорости, значит ошибок в управлении моторами не произойдёт.

Последние строки кода программа – условие, при котором в последовательном порту нет данных, и устанавливается нулевая скорость электромоторов.

Таким образом, мы подготовили полную программу движения с помощью передачи непосредственно отклонения. Программа представлена в листинге 9.

Листинг 9.

```
if (Serial.available() > 0) {

      while (Serial.readBytesUntil('\n', bufNum,
sizeof(bufNum)))
      {
        int getValue = atoi(bufNum);

        bufNum[0] = 0;
        bufNum[1] = 0;
        bufNum[2] = 0;
        bufNum[3] = 0;
        bufNum[4] = 0;
        bufNum[5] = 0;

        currentError = getValue;
        proportional = currentError * coefP;
        differential = (currentError - previousError) *
coefD;
        driver = proportional + differential;
        previousError = currentError;

        float boardUp = 320 * coefP + 25 * coefD;
        float boardDown = -320 * coefP + -25 * coefD;

        driver = map(driver, boardDown, boardUp, -65, 65);
        driver = constrain(driver, -80, 80);

        myMotor_right.set_speed(speed + driver);
        myMotor_left.set_speed(weirdSpeed - (driver *
coef));
```

```
    }
}
else if (Serial.available() == -1)
{
    myMotor_right.set_speed(0);
    myMotor_left.set_speed(0);
}
```

Полный код алгоритма управления для непосредственной передачи отклонения на Arduino представлен в Приложении Г.

Таким образом, в данном разделе были полностью разработаны программы управления исполнительными механизмами. Были написаны программные коды для управления мобильным роботом с помощью двух алгоритмов пересылки данных. Произведены отладка и тестирования каждого.

8 Экономическая эффективность

Список издержек для разработки данной платформы будет включать только завершенные технические элементы, не требующие дополнительной механической обработки или трудозатрат разного рода. Экономический расчёт представлен на рисунке 36.

№	Наименование	Кол	Цена, р
1	Raspberry Pi 3 Model B	1	8400
2	L298n	1	100
3	LM2596S	1	150
4	KIS-3R33SS	1	120
5	GM25-370	2	1800
6	Arduino MEGa2560	1	1800
7	Lingshe K200	1	120
8	Defender Patch MS-759	1	100
9	Logitech C270 HD 720p	1	1600
10	Display Waveshare Raspberry Pi Touch Screen Display Monitor 480x272 HDM – LCD	1	6000
11	Li-Po 11.8V 3S1P 5000 мА×ч	2	12000
12	Rocker Switch	2	90
13	DS-1058-1	1	90
14	ПВС 2x0,75-10	1	170
15	BW3303	1	210
16	BW1411	1	230
17	BW1431	2	560
18	Винт М3	22	16,5
19	Винт М4	42	68,8
20	Гайка М3	76	57
21	Гайка М4	88	81,3
22	Гайка М6	12	18
23	Шайба М3	76	30,4
24	Шайба М4	88	44
25	Шайба М6	12	12
26	Стойка крепёжная М3	6	96
27	Blackout XB Mounted Wheel and Tyre Set Front 2pcs	1	2762
28	Колесная опора поворотная 3470U0R080P62	2	1224

Итого: 31 842 руб

Рисунок 36 – Экономический расчет

Таким образом, в данном разделе произведён полный экономический расчёт компонентов, не требующих трудозатрат разного рода.

Заключение

В рамках выполнения данной выпускной квалификационной работы был разработан автономный, мобильный, колёсный робот, осуществляющий навигацию с помощью технического зрения и контрастной линии.

В начале данной бакалаврской работы было проведено исследование областей применения робототехнических систем и выявлено наиболее перспективное направление развития.

В первом разделе, после анализа уже существующих робототехнических систем, была обоснована актуальность текущей разработки.

Во втором разделе определены методы позиционирования, широко используемые в мире, объяснён их выбор и их применимость. Осуществлён выбор способа навигации – техническое зрение и контрастная линия.

В третьем разделе была разработана структурная схема платформы, были выбраны все основные электротехнические компоненты для мобильного робота.

В четвёртом разделе выбран способ связи между микроконтроллером и одноплатным компьютером – с помощью интерфейса UART и кабеля USB. Подготовлены два алгоритма приёма-передачи.

В пятом разделе была разработана и представлена схема электрическая соединений, включающая все электротехнические компоненты и способы их соединения.

В шестом разделе была представлена конструкция мобильного робота: четырёхколёсная платформа из двух ярусов малого размера. Разработан сборочный чертёж.

В седьмом разделе созданы два алгоритма работы мобильного робота и их программная реализация, проведена их настройка, проведены экспериментальные испытания образца.

В восьмом разделе произведена оценка экономической эффективности.

Результатом выполнения данной выпускной квалификационной работы стали разработка и физическое воплощение автономного мобильного робота, осуществляющего движение по разным поверхностям с помощью технического зрения.

Список используемой литературы

1. Бакенов А.Ж. Система управления движением мобильного робота с использованием компьютерного зрения // Наука настоящего и будущего. – 2020. – Том 2. – С. 111-114.
2. Вильямс, Д. Программируемый робот, управляемый с КПК / Д. Вильямс. – М.: НТ Пресс, 2006. – 223 с.
3. Гаврилов А.В, Чусов А.В. Автономный мобильный робот. Конструирование и программирование. / Издательский Дом "Техносфера". – 2015. – 300 с.
4. Глибин Е.С., Прядилов А.В. Программирование электронных устройств. – Тольятти: ФГБОУ ВПО «Тольяттинский государственный университет», 2014. - 118 с.
5. Гусев В.Г., Гусев Ю.М. Электроника и микропроцессорная техника . – 6-е изд. – М.: Кнорус, 2016. - 798 с.
6. Кершин А.Ж., Ергалиев Д.С. Навигация и управление мобильным роботом // Труды Международного симпозиума «Надежность и качество». – 2017. – №3. – С. 4-5.
7. Лапшов В.С., Носков В.П., Рубцов И.В., Рудианов Н.А., Гурджи А.И., Рябов А.В., Хрущев В.С. Перспективы разработки автономных наземных робототехнических комплексов специального военного назначения // Известия Южного федерального университета. Технические науки. – 2016. – №10. – С. 1-13.
8. Макарова Н.В., Дмитриев В. В. Использование аппаратно-программной платформы на базе Raspberry Pi в качестве центрального контроллера роботов // Материалы ученой конференции "Интеллектуальные системы управления технологическими процессами и производством", 2018.
9. Михайлов Б.Б., Назарова А.В, Ющенко А.С. Автономные мобильные роботы - навигация и управление // Известия Южного федерального университета. Технические науки. – 2016. – С. 48-67.

10. Овсянников А.Р., Лепехина С.Ю, Сухоруков С.И. Анализ возможностей применения миникомпьютеров для построения систем управления роботизированными комплексами // Производственные технологии будущего: от создания к внедрению. - Комсомольск-на-Амуре: Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет», 2022. – С. 63-67.

11. Старовойтов Е.И. Управление мобильными роботами и робототехническими системами . – М.: Кнорус, 2021. - 264 с.

12. У бота через GPIO в Raspberry Pi 3 // Вашумныйдом : сайт. – URL: https://vashumnyidom.ru/upravlenie/ustrojstva/gpio-raspberry-pi-3.html (дата обращения: 26.04.2023).

13. Amazon. Алгоритмы работы самого крупного ритейлера в мире // Хабр : сайт. – URL: https://habr.com/ru/companies/pochtoy/articles/406783/ (дата обращения: 10.05.2023)

14. Документация на драйвер L298n // SparkFun Electronics URL: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf (дата обращения: 01.05.2023)

15. Julio E. Normey-Rico, Ismael Alcalá, Juan Gómez-Ortega, Eduardo F. Camacho Mobile robot path tracking using a robust PID controller // Control Engineering Practice. - IFAC, 2001. – С. 1209-1214.

16. Staple D. Learn Robotics Programming: Build and control AI-enabled autonomous robots using the Raspberry Pi and Python. – 2-е изд. – Packt, 2021. - 602 с.

17. Arduino : сайт. – URL: https://www.arduino.cc/ (дата обращения: 13.05.2023).

18. European Patent Office : сайт. – URL: https://www.epo.org/ (дата обращения: 26.05.2023)

19. Raspberry Pi Foundation : сайт. –. URL: https://www.raspberrypi.org/ (дата обращения: 20.05.2023)

20. TurtleBot // ROS URL: http://wiki.ros.org (дата обращения: 10.05.2023)

Приложение А

Программный код алгоритма управления для символьной передачи отклонения Raspberry Pi

```
#include <opencv2/opencv.hpp>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <vector>
#include <iostream>
#include <string>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>

using namespace std;
using namespace cv;

const float coefP = 1.0f;
const float coefI = 0.01f;
const float coefD = 6.0f;
float summOfError = 0;
int delta;
float driver;
int serialPort;

const char SHARP_RIGHT = 'R';
const char LESS_SHARP_RIGHT = 'E';
const char MID_RIGHT= 'P';
const char LOW_RIGHT = 'r';

const char SHARP_LEFT = 'L';
const char LESS_SHARP_LEFT = 'K';
const char MID_LEFT = 'V';
const char LOW_LEFT = 'l';

const char FORWARD = 'F';

float bufError[10] = {0, 0, 0, 0, 0, 0, 0 ,0 ,0 ,0};
uint8_t i = 0;
```

```
int cases[][3] = {
    { SHARP_LEFT, 249, 320 },
    { LESS_SHARP_LEFT, 178, 249 },
    { MID_LEFT, 122, 178 },
    { LOW_LEFT, 52, 122 },
    { FORWARD, -52, 52 },
    { LOW_RIGHT, -122, -52 },
    { MID_RIGHT, -178, -122 },
    { LESS_SHARP_RIGHT, -249, -178 },
    { SHARP_RIGHT, -320, -249 },
};

void regulatorPID()
{
if (i >= 10) {i = 0;}
    bufError[i] = delta;
    float summ = 0;
    for (auto j:bufError) { summ += j; }
    float proportional = bufError[i] * coefP;
    float integral = summ * coefI;
    float differential = (bufError[i] - bufError[(i + 9)
% 10]) * coefD;
    driver = proportional + integral + differential;
    i++;
}

int main(int argc, char** argv)
{

     namedWindow("Example", WINDOW_AUTOSIZE);
    moveWindow("Example", 70, 70);
     VideoCapture cap;
     cap.open(0);
     Mat frame, hsv, mask;
     Moments m_blue, m_green, m_red;
     int width, x, y, center, flag = 0, state = 0;
     vector<int>lower_blue = { 75, 60, 50 };
     vector<int>upper_blue = { 130, 255, 255 };
     vector<int>lower_green = { 40, 50, 80 };
     vector<int>upper_green = { 80, 255, 255 };
     vector<int>lower_red = { 0, 100, 100 };
     vector<int>upper_red = { 15, 255, 255 };
```

```
    char show_delta[5] = { 0 };

  delay(1000);

    for (;;)
    {
       serialPort = serialOpen("/dev/ttyUSB0",9600);
       if (serialPort == -1)
       {
           cout << "FAILED open port" << endl;
           break;
       }
       for(;;)
       {
           cap >> frame;
           if (frame.empty()) break;
           cvtColor(frame, hsv, COLOR_BGR2HSV);
           inRange(hsv, lower_green, upper_green, mask);
           m_green = moments(mask,1);

           if(m_green.m00 > 30000 && flag > 5)
           {
               state = 1;
               namedWindow("Example", WINDOW_AUTOSIZE);
               moveWindow("Example", 100, 80);
               break;
           }
           flag++;
       }

       if(state)
       {
           for(;;)
           {
               cap >> frame;
               if (frame.empty()) break;
               width = frame.size().width;
               cvtColor(frame, hsv, COLOR_BGR2HSV);
               inRange(hsv, lower_blue, upper_blue,
mask);
               m_blue = moments(mask, 1);
```

```
                if (m_blue.m00 > 600)
                {
                    x = m_blue.m10 / m_blue.m00;
                    y = m_blue.m01 / m_blue.m00;
                    circle(frame, Point(x, y), 5,
Scalar(0, 0, 255), -1);
                }

                center = width / 2;
                circle(frame, Point(center, y), 5,
Scalar(0, 255, 0), -1);
                line(frame, Point(center, y), Point(x,
y), Scalar(0, 255, 255), 2);
                delta = center - x;
                sprintf(show_delta, "%d", delta);
                putText(frame, show_delta, Point(width /
2, y - 20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255),
3);

                regulatorPID();
                for (int i = 0; i < 9; ++i)
                {
                    if(driver > cases[i][1] && driver <
cases[i][2])
                    {
                        serialPutchar(serialPort,
cases[i][0]);
                        break;
                    }
                }

                imshow("Example", frame);

                inRange(hsv, lower_red, upper_red, mask);
                m_red = moments(mask, 1);

                if(m_red.m00 > 30000)
                {
                    state = 0;
                    serialClose(serialPort);
                    destroyWindow("Example");
                    break;
```

```
                }
                if (waitKey(10) >= 0)
                {
                    state = 0;
                    serialClose(serialPort);
                    destroyWindow("Example");
                    break;
                }
            }
        }
    }
    return 0;
}
```

Приложение Б

Программный код алгоритма управления для символьной передачи отклонения Arduino

```
#define counterclockwise 0
#define clockwise 1

const float coef = 0.81;
const byte topSpeed = 110;
byte lowSpeed = topSpeed - 45;
byte midSpeed = topSpeed - 35;
byte underTopSpeed = topSpeed - 30;
byte difMotorSpeed = topSpeed * coef;

int serialTimer;

class Motor {
public:
  Motor(int INPUT1, int INPUT2, int ENABLE, int ENCODER1,
int ENCODER2);
  void start();
  void stop();
  void set_speed(int speed);
  void set_direction(bool direction);
protected:
  int _INPUT1, _INPUT2, _ENABLE, _ENCODER1, _ENCODER2,
_encoder_state;
  bool _motor_working, _motor_type, _direction;
  unsigned long _speed;
};

Motor::Motor(int INPUT1, int INPUT2, int ENABLE, int
ENCODER1, int ENCODER2) {
  _INPUT1 = INPUT1;
  _INPUT2 = INPUT2;
  _ENABLE = ENABLE;
  _ENCODER1 = ENCODER1;
  _ENCODER2 = ENCODER2;
  pinMode(_INPUT1, OUTPUT);
  pinMode(_INPUT2, OUTPUT);
  pinMode(_ENABLE, OUTPUT);
  pinMode(_ENCODER1, INPUT_PULLUP);
```

```
  pinMode(_ENCODER2, INPUT_PULLUP);
  _motor_working = false;
}

void Motor::start() {
  _motor_working = true;
}

void Motor::stop() {
  analogWrite(_ENABLE, 0);
  _motor_working = false;
}

void Motor::set_speed(int speed) {
  _speed = speed;
  if (_motor_working) { analogWrite(_ENABLE, _speed); }
}

void Motor::set_direction(bool direction) {
  _direction = direction;

  if (_direction == clockwise) {
    digitalWrite(_INPUT1, LOW);
    digitalWrite(_INPUT2, HIGH);
  } else {
    digitalWrite(_INPUT1, HIGH);
    digitalWrite(_INPUT2, LOW);
  }
}

Motor myMotor_right(11, 12, 13, 16, 17);
Motor myMotor_left(10, 9, 8, 47, 49);

void setup() {
  myMotor_right.start();
  myMotor_left.start();
  myMotor_right.set_direction(clockwise);
  myMotor_left.set_direction(clockwise);
  Serial.begin(9600);
}

void loop() {
```

```
 if (Serial.available()) {
  char getCh = Serial.read();

  switch (getCh) {
    case 'F':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(difMotorSpeed);
      break;
    case 'R':
      myMotor_right.set_speed(0);
      myMotor_left.set_speed(difMotorSpeed);
      break;
    case 'E':
      myMotor_right.set_speed(lowSpeed);
      myMotor_left.set_speed(difMotorSpeed);
      break;
    case 'P':
      myMotor_right.set_speed(midSpeed);
      myMotor_left.set_speed(difMotorSpeed - 5);
      break;
    case 'r':
      myMotor_right.set_speed(underTopSpeed);
      myMotor_left.set_speed(difMotorSpeed - 10);
      break;
    case 'L':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(0);
      break;
    case 'V':
      myMotor_right.set_speed(topSpeed);
      myMotor_left.set_speed(coef * lowSpeed);
      break;
    case 'K':
      myMotor_right.set_speed(topSpeed - 5);
      myMotor_left.set_speed(coef * midSpeed);
      break;
    case 'l':
      myMotor_right.set_speed(topSpeed - 10);
      myMotor_left.set_speed(coef * underTopSpeed);
      break;
  }
}
```

```
  else if (Serial.available() == -1)
  {
    myMotor_right.set_speed(0);
    myMotor_left.set_speed(0);
  }
}
```

Приложение В

Программный код алгоритма управления для непосредственной передачи отклонения Raspberry Pi

```
#include <opencv2/opencv.hpp>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <vector>
#include <iostream>
#include <string>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>

using namespace std;
using namespace cv;

int main(int argc, char** argv)
{

      namedWindow("Example", WINDOW_AUTOSIZE);
    moveWindow("Example", 70, 70);
      VideoCapture cap;
      cap.open(0);
      Mat frame, hsv, mask;
      Moments m,m_green,m_red;
      int width, x, y, center, delta, flag = 0, state =
0;
      vector<int>lower_blue = { 75, 60, 50 };
      vector<int>upper_blue = { 130, 255, 255 };
      vector<int>lower_green = { 40, 50, 80 };
      vector<int>upper_green = { 80, 255, 255 };
    vector<int>lower_red = { 0, 100, 100 };
      vector<int>upper_red = { 15, 255, 255 };
      char show_delta[5] = { 0 };
      int serialPort;

      char bufToSend[4];

    delay(1000);
```

```
        for (;;)
        {
           serialPort = serialOpen("/dev/ttyUSB0",9600);

           if (serialPort == -1)
           {
               cout << "FAILED open port" << endl;
               return 0;
           }

           for(;;)
           {
               cap >> frame;
               if (frame.empty()) break;
               cvtColor(frame, hsv, COLOR_BGR2HSV);
               inRange(hsv, lower_green, upper_green, mask);
               m_green = moments(mask,1);

               if(m_green.m00 > 30000 && flag > 5)
               {
                   state = 1;
                   namedWindow("Example", WINDOW_AUTOSIZE);
                   moveWindow("Example", 100, 80);
                   break;
               }
               flag++;
           }

           if(state)
           {
               for(;;)
               {
                   cap >> frame;
                   if (frame.empty()) break;
                   width = frame.size().width;
                   cvtColor(frame, hsv, COLOR_BGR2HSV);
                   inRange(hsv, lower_blue, upper_blue,
mask);
                   m = moments(mask, 1);

                   if (m.m00 > 600)
                   {
```

```
                    x = m.m10 / m.m00;
                    y = m.m01 / m.m00;
                    circle(frame, Point(x, y), 5,
Scalar(0, 0, 255), -1);
                }

                center = width / 2;
                circle(frame, Point(center, y), 5,
Scalar(0, 255, 0), -1);
                line(frame, Point(center, y), Point(x,
y), Scalar(0, 255, 255), 2);
                delta = center - x;
                sprintf(show_delta, "%d", delta);
                putText(frame, show_delta, Point(width /
2, y - 20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255),
3);

                sprintf(bufToSend,"%d", delta);

                serialPrintf(serialPort, "%s\n",
bufToSend);
                bufToSend[0] = 0;
                bufToSend[1] = 0;
                bufToSend[2] = 0;
                bufToSend[3] = 0;

                imshow("Example", frame);

                inRange(hsv, lower_red, upper_red, mask);
                m_red = moments(mask, 1);

                if(m_red.m00 > 30000)
                {
                    state = 0;
                    serialClose(serialPort);
                    destroyWindow("Example");
                    break;
                }

                if (waitKey(10) >= 0)
                {
                    state = 0;
```

```
                    serialClose(serialPort);
                    destroyWindow("Example");
                    break;
                }
            }
        }
    }
    return 0;
}
```

Приложение Г

Программный код алгоритма управления для непосредственной передачи отклонения Arduino

```
#define counterclockwise 0
#define clockwise 1

const float coef = 0.81;
const byte speed = 95;
const byte difMotorSpeed = speed * coef;
char bufNum[6];
float bufError[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
uint8_t i = 0;
const float coefP = 1.0f;
const float coefI = 0.0f;
const float coefD = 7.0f;
float driver;

class Motor {
public:
  Motor(int INPUT1, int INPUT2, int ENABLE, int ENCODER1,
int ENCODER2);
  void start();
  void stop();
  void set_speed(int speed);
  void set_direction(bool direction);
protected:
  int _INPUT1, _INPUT2, _ENABLE, _ENCODER1, _ENCODER2,
_encoder_state;
  bool _motor_working, _motor_type, _direction;
  unsigned long _speed;
};

Motor::Motor(int INPUT1, int INPUT2, int ENABLE, int
ENCODER1, int ENCODER2) {
  _INPUT1 = INPUT1;
  _INPUT2 = INPUT2;
  _ENABLE = ENABLE;
  _ENCODER1 = ENCODER1;
  _ENCODER2 = ENCODER2;
  pinMode(_INPUT1, OUTPUT);
```

```
  pinMode(_INPUT2, OUTPUT);
  pinMode(_ENABLE, OUTPUT);
  pinMode(_ENCODER1, INPUT_PULLUP);
  pinMode(_ENCODER2, INPUT_PULLUP);
  _motor_working = false;
}

void Motor::start() {
  _motor_working = true;
}

void Motor::stop() {
  analogWrite(_ENABLE, 0);
  _motor_working = false;
}

void Motor::set_speed(int speed) {
  _speed = speed;
  if (_motor_working) { analogWrite(_ENABLE, _speed); }
}

void Motor::set_direction(bool direction) {
  _direction = direction;

  if (_direction == clockwise) {
    digitalWrite(_INPUT1, LOW);
    digitalWrite(_INPUT2, HIGH);
  } else {
    digitalWrite(_INPUT1, HIGH);
    digitalWrite(_INPUT2, LOW);
  }
}

Motor myMotor_right(11, 12, 13, 16, 17);
Motor myMotor_left(10, 9, 8, 47, 49);

void setup() {
  myMotor_right.start();
  myMotor_left.start();
  myMotor_right.set_direction(clockwise);
  myMotor_left.set_direction(clockwise);
  Serial.begin(9600);
```

```
}

void loop() {
  if (Serial.available() > 0) {

        while (Serial.readBytesUntil('\n', bufNum,
sizeof(bufNum))) {
      int getValue = atoi(bufNum);

      bufNum[0] = 0;
      bufNum[1] = 0;
      bufNum[2] = 0;
      bufNum[3] = 0;
      bufNum[4] = 0;
      bufNum[5] = 0;


       currentError = getValue;
       proportional = currentError * coefP;
       differential = (currentError - previousError) *
coefD;
       driver = proportional + differential;
       previousError = currentError;

       float boardUp = 320 * coefP + 25 * coefD;
       float boardDown = -320 * coefP + -25 * coefD;

       driver = map(driver, boardDown, boardUp, -65, 65);
       driver = constrain(driver, -80, 80);

       myMotor_right.set_speed(speed + driver);
       myMotor_left.set_speed(difMotorSpeed - (driver *
coef));
      }
  }
  else if (Serial.available() == -1)
  {
     myMotor_right.set_speed(0);
     myMotor_left.set_speed(0);
  }
```