МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт энергетики и электротехники
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование кафедры)

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

Промышленная электроника
(направленность (профиль)/ специализация)

# БАКАЛАВРСКАЯ РАБОТА

на тему Мобильная робототехническая платформа

Студент С.В. Гущин _____
(И.О. Фамилия) (личная подпись)

Руководитель Е.С. Глибин _____
(И.О. Фамилия) (личная подпись)

Консультанты О.Н. Брега _____
(И.О. Фамилия) (личная подпись)

**Допустить к защите**

Заведующий кафедрой к.т.н., доцент, А.А. Шевцов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«____»_____20___г.

Тольятти 2017

Аннотация

Объём бакалаврской работы – 79 стр., таблиц – 2, рисунков – 48,

В работе разработана мобильная робототехническая платформа.

Структура работы представлена десятью разделами, заключением и списком литературы.

Определены актуальность темы, цели и задачи. В заключении сделаны выводы о проделанной работе.

Характерной чертой данной разработки является выполнение задачи позиционирования в замкнутом помещении с использованием инерциальной навигационной системы и методов компьютерного зрения.

На данный момент в Российской Федерации не распространены решения, позволяющие создавать мобильные робототехнические комплексы, способные осуществлять навигацию в замкнутых помещения без использования радиосигналов, направляющих рельс и напольных линий, в то время как в Европейских странах и США такие системы начинают применяться. Комплексы на базе разрабатываемого решения, могут иметь широкое применение в области складской логистики, а также в сфере обслуживания.

Целью работы является разработка колёсной роботизированной платформы для перемещения грузов весом до одного килограмма внутри замкнутого помещения. Робот должен быть способен перемещаться по покрытиям различных типов таких как линолеум и напольная плитка, а время автономной работы должно составлять не менее тридцати минут.

Предложена структурная схема робототехнической платформы. Обосновано выбраны отдельные конструктивные элементы. Представлена схема ориентации и движения робота, а также сборочный чертеж платформы. Разработана схема электрических соединений и алгоритм работы платформы. Оценена безопасность и экологичность проекта. Выполнен экономический расчет.

Abstract

The volume of bachelor's work is 79 p., 2 tables, 48 drawings.

A mobile robotic platform has been developed.

The structure of the work presents ten sections, conclusion and references.

The relevance of the topic, goals and objectives are determined.

A key feature of this development is the fulfillment of tasks related to the use of the inertial navigation system and computer vision methods.

At the moment, in the Russian Federation there is no solutions that allow the creation of mobile robotic complexes which are capable to navigate in closed spaces without the use of radio signals, guide rails and lines on the floor, while in European countries and the United States such systems are beginning to be used. Complexes, based on the developed solution, can have wide application in the field of warehouse logistics, as well as in the service sector.

The aim of the work is the development of a wheeled robotic platform for delivering loads of up to one kilogram weight inside a closed room. The robot should be able to move on various types of coating, such as linoleum and floor tiles. The battery life should be more than thirty minutes.

A block diagram of the robotic platform is proposed. Platform components are selected. The scheme of orientation and movement of the robot, as well as an assembly drawing of the platform are presented. The scheme of electrical connections and algorithm of the platform operation are developed. The safety and environmental friendliness of the project are described. An economic calculation was performed.

Содержание

Введение

На данный момент в мире одно из активно развивающихся направлений – робототехника. Робототехнические комплексы применяют для автоматизации, ускорения темпов производства и получения качественного результата в различных технологических процессах, а также промышленных устройствах. Например, в автомобилестроении при сварке кузовов (рисунок 1), при производстве печатных плат различной плотности монтажа, в станках с числовым программным управлением (ЧПУ) (рисунок 2), в станках для лазерной и гидроабразивной резки и многих других областях производства.

Рисунок 1 – Сварка кузовов легковых автомобилей при помощи промышленных манипуляторов


Рисунок 2 – Токарный станок с ЧПУ

Также робототехнические комплексы активно разрабатываются и внедряются в военной сфере. Они могут представлять из себя как наземную, так и

воздушную технику. Например, для разведки, нанесения точечных авиационных ударов, корректировки огня артиллерии и т.д. в современных армиях мира применяются беспилотные летательные аппараты начиная от легких, которые имеют небольшой вес, радиус действия и могут запускаться с переносных установок или даже с рук (рисунок 3 слева) и заканчивая тяжелыми авиационными системами способными на длительное нахождение в воздухе и доставку средств поражения живой силы и техники (рисунок 3 справа).

Рисунок 3 – Беспилотные летательные аппараты, слева - легкий, справа – тяжелый

Примером наземных военных робототехнических устройств являются комплексы для проведения инженерно-саперных работ или для оказания огневой поддержки войск. Примером комплексов для проведения инженерно-саперных работ является Российский комплекс «Уран-6» представленный на рисунке 4.


Рисунок 4 – Комплекс для проведения разминирования «Уран-6»

Примером роботизированного комплекса для оказания огневой поддержки войск является комплекс «Уран-9» который на данный момент поставляется в войска Российской Федерации (рисунок 5).

Рисунок 5 – Роботизированный боевой комплекс «Уран-9»

В гражданской сфере роботизированные комплексы применяются для самых различных целей. Они могут выполнять роль интерактивных рекламных устройств, для консультаций и сопровождения клиентов, в качестве систем имитации присутствия, в качестве роботов-гидов, осуществлять уборку помещений. В качестве простого примера уборщика помещений можно привести робот пылесос (рисунок 6), уже вошедший в быт человека.

Рисунок 6 – Робот пылесос производства компании iRobot

Хорошим представителем роботов-гидов, роботов для консультаций и сопровождения клиентов является продукт деятельности российской команды разработчиков робот «Promobot» изображенный на рисунке 7. Этот робот обладает широким функционалом, включающим распознавание и запоминание лиц, а также

возможность общения на различных языках. Применение таких систем позволяет исключить человеческий фактор, повысить продуктивность, снизить затраты на персонал.

Рисунок 7 – Промо-робот «Promobot»

В транспорте роботизированные системы применяются как для помощи при управлении автомобилем, так и для полностью автономного управления автомобилями. Уже на данный момент крупные автопроизводители оснащают свои автомобилями системами способными автоматически удерживать автомобиль на полосе, отслеживая дорожную разметку, выполнять роль системы защиты от столкновений, при возникновении опасных ситуаций. Также, на данный момент, существуют автомобили, способные полностью автономно управлять движением. Есть обычные автомобили, оснащаемые роботизированными комплексами для управления как, например, автомобиль «Toyota» изображенная на рисунке 8, но есть и производители уже выпускающие автомобили с интегрированной системой.

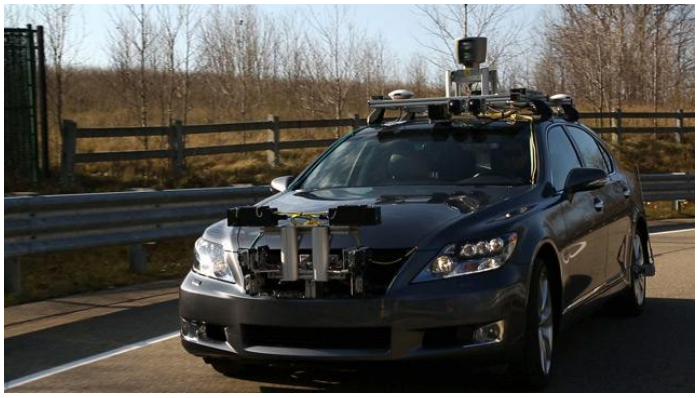


Рисунок 8 – Автомобиль, оснащенный системой автоматического управления движением

Компанией, уже наладившей производство таких автомобилей, является «Tesla». Такая система на основе информации с датчиков и камер анализирует окружающую обстановку и выполняет действия в соответствии с заложенными правилами. На рисунке 9 представлена работа такой системы. Видно, как камера распознаёт и выделяет отдельные объекты, важные с точки зрения управления движением – линию разметки, край дорожного полотна, статичные объекты и объекты, находящиеся на проезжей части.


Рисунок 9 – Работа автоматизированной системы управления в автомобиле «Tesla»

Роботизированные системы все чаще применяются в области складской логистики и для доставки различных грузов. Так, например, компания «Amazon» планирует осуществлять доставку небольших заказов при помощи специально оснащенных квадрокоптеров и гексакоптеров (рисунок 10).


Рисунок 10 – Гексакоптер-почтальон компании «Amazon»

Для доставки грузов в складских помещениях используются специальные подвижные платформы, обладающие различным функционалом в зависимости от комплектации. Для примера на рисунке 11 представлена автономная платформа для доставки грузов массой до полутора тонн разработанная компанией «OTTO Motors».


Рисунок 11 – Робот для доставки грузов «OTTO»

Также есть много других областей применения робототехники, таких как космическая отрасль, медицина, образование и т.д. Основные причины применения роботизированных систем — это защита человека от вредных воздействий на производствах, оптимизация и ускорение производственного процесса, снижение влияния человеческого фактора при работе, удешевление и повышение надежности производства. Роботизированные системы позволяют облегчить жизнь, забирая на себя некоторые бытовые обязанности, давая возможность потратить время с пользой. Роботы призваны улучшить качество жизни человека, обеспечить помощь и безопасность в различных сферах деятельности человека, а также задать более высокий темп развития человечества вцелом.

# 1 Разбор рынка и актуальность разработки

Одним из актуальных направлений развития робототехники является конструирование систем складской логистики. Причинами развития данной отрасли послужили большая гибкость и масштабируемость роботизированного решения перед классическим. Под классическими будем понимать системы использующие рельсовые направляющие и краны, напольные линии для навигации передвижных платформ или же погрузчики, управляемые людьми. Зачастую классические системы являются громоздкими, ограниченными в применении и достаточно требовательными к эксплуатационным условиям. Так, например, для платформ передвигающимся по линиям необходимо ровное и чистое покрытие на котором будет хорошо различима линия следования, иначе датчики потеряют линию и возможен сбой в работе складской системы. Системы, использующие направляющие имеют большие габариты и наименьшую мобильность из классических систем. Системы погрузчиков, управляемых людьми, имеют ряд недостатков связанных с дополнительными затратами на оплату труда водителям погрузчиков, также шанс ошибки человека, в основном, выше чем шанс ошибки автоматизированного устройства, необходимость определенной подготовки персонала для управления погрузчиками, необходимость соблюдения безопасных для здоровья человека условий труда (загрязненность помещений, освещение, вентиляция и т.д.), также шанс кражи на некоторых складах выше.

В свою очередь роботизированные комплексы позволяют подобрать и настроить систему для решения конкретной задачи, минимизировать участие человека в процессе работы, повысить скорость и качество работы, а также, в некоторых случаях, снизить требования к окружающей обстановке.

На данный момент в мире присутствует ограниченное количество решений, позволяющих создавать мобильные платформы, способные ориентироваться в замкнутых пространствах. Определенные успехи в производстве и внедрении таких систем достигли в Европе – там присутствует несколько производителей

предоставляющих широкий выбор роботизированных складских систем. Также внедрение роботизированных систем осуществляется на территории США.

Рассмотрим основные имеющиеся роботизированные системы. Одна из компаний, производящих такие роботизированные системы является «OTTO Motors». Её продукция представлена набором платформ с определенным функционалом среди которых роботы для транспортировки, подъёма грузов, платформы, оснащенные манипуляторами. Также предлагаемые устройства различаются в весовой категории – присутствуют как компактные тележки с небольшой загрузкой, так и платформы для транспортировки массивных грузов. Вид платформ представлен на рисунке 1.1.


Рисунок 1.1 – Роботизированные комплексы, предлагаемые компанией «OTTO Motors»

В основе данной системы лежит использование информации получаемой с нескольких лазерных радаров, или как их еще называют – лидаров (LIDAR - Light Identification Detection and Ranging — световое обнаружение и определение дальности) рисунок 1.2.

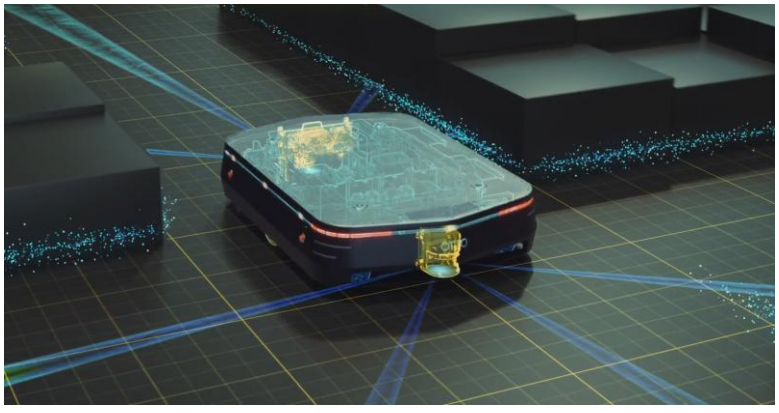


Рисунок 1.2 – Работа платформы «OTTO 1500»

Согласно информации на сайте производителя платформа полностью автономна. Заранее в неё заложена карта местности и определенные ключевые точки для коррекции, позиции которых заранее известны. При помощи лидаров платформа сканирует местность и на основе полученной информации о статичных и динамичных препятствиях осуществляет безопасное передвижение. Можно предположить, что оценка пройденного расстояния осуществляется при помощи методов одометрии, хотя такой информации производитель не предоставляет. Также для связи платформа оснащена wi-fi модулем, что обеспечивает дистанционное общение и выдачу заданий для платформы. По низкому клиренсу платформы и отсутствию существенных амортизирующих устройств можно предположить, что данная роботизированная система требовательна к качеству и загрязненности покрытия.

Другой компанией, производящей роботизированные комплексы для складской логистики, является «KUKA Roboter». В целом ряд предлагаемых платформ схож с рядом предлагаемым «OTTO» однако по методу передвижения у платформ компании «KUKA» есть некоторые преимущества. Изображение платформы представлено на рисунке 1.3.


Рисунок 1.3 – Роботизированная платформа компании «KUKA Roboter»

Для навигации в данных платформах используется схожее решение – заранее заложенная карта и набор лидаров, однако для передвижения она использует специальные колёса – колёса Илона или омни-колёса. Такие колеса оснащены дополнительными роликами, закрепленными под углом в 45 градусов по внешнему контуру колеса. Такие колеса позволяют суммировать крутящий момент и

передвигаться не только вперед-назад, но и под углом. Таким образом данная платформа благодаря таким колесам может занимать любое положение в плоскости, что добавляет ей подвижности. Однако применение таких колес требовательно к покрытию рабочей области, недопустимо наличие камней, способных заклинить ролики. Также отличительной особенностью данной платформы является модульность – несколько платформ могут объединяться в модуль с общим управлением для перемещения крупногабаритных грузов – к примеру, для перемещения на авиастроительных предприятиях фюзеляжей самолётов. На рисунке 1.4 представлена описанная сборка.


Рисунок 1.4 – Сборка из трёх платформ для транспортировки крупных грузов.

Также производством роботизированных систем со схожим функционалом занимается компания «fetch robotics». Так как продукты данной компании внешне и функционально практически не различаются с описанными ранее то подробно их рассматривать не будем. Варианты роботизированных платформ представлены на рисунке 1.5.

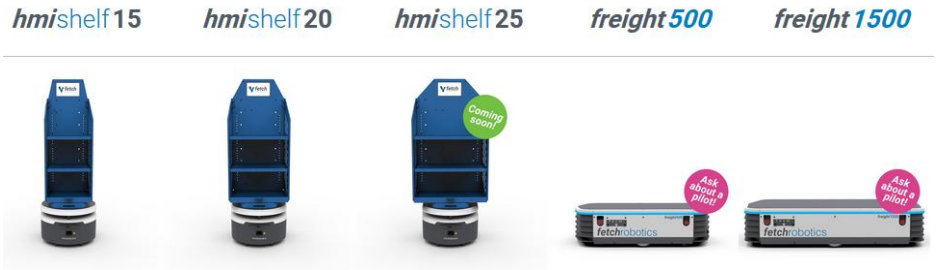


Рисунок 1.5 – Роботизированные платформы, производимые компанией «fetch robotics»

Так как большинство информации по алгоритмам работы и устройству таких систем является коммерческой тайной производителей, то у нас нет возможности провести более подробный анализ и оценку имеющихся образцов.

Общей чертой представленных систем является использование большого количества дорогостоящих датчиков, для отслеживания текущего положения в пространстве. Такие подходы к автоматизации предлагались еще в конце двадцатого века, так как вычислительные возможности ЭВМ того времени не позволяли реализовать сложные алгоритмы.

Сегодня, благодаря развитию электронной техники, возможно создавать высокопроизводительные комплексы, которые осуществляют ориентацию по небольшому набору датчиков, компенсируя малое количество регистрирующих устройств сложными, но достаточно точными алгоритмами обработки информации и последующее принятие решений, на основе рассчитанных данных. В данной работе предлагается реализация одного из таких алгоритмов.

# 2 Выбор и обоснование решения для навигации робота

Рассмотрим несколько вариантов определения местоположения существующих в мире на данный момент.

Самым распространенным методом определения текущей координаты является спутниковая навигация (GPS или ГЛОНАСС). При таком методе координата объекта, оснащенного соответствующим маяком/антенной определяется при помощи измерения времени следования электромагнитной волны от этого объекта до нескольких спутников, координаты которых известны заранее, и расчет расстояния от каждого спутника до определяемого объекта (рисунок 2.1). Данные обрабатываются системой и на GPS/ГЛОНАСС передатчик посылаются вычисленные координаты. Такой метод не подходит для точного позиционирования потому что для удовлетворительной точности определения местоположения необходимо большое количество спутников (более 3) с устойчивой передачей сигнала. Однако даже при наличии достаточного количества спутников и хорошего сигнала погрешность определения местоположения составляет десятки метров, чего явно недостаточно. Так же, зачастую, в помещениях сигнал от спутника вовсе отсутствует из-за естественных преград для радиоволн. Поэтому данный метод не подходит.

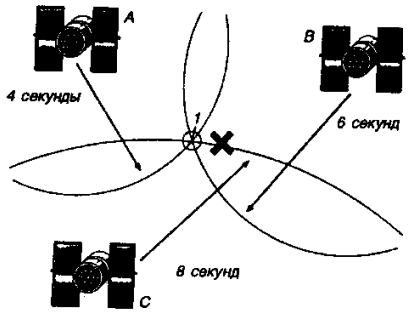



Рисунок 2.1 – Определение местоположения при помощи спутников

Другим методом является использование радио или wi-fi маяков. Принцип такого позиционирования схож со спутниковым методом, только здесь вместо спутников используются радиомаяки. При определении местоположения таким

методом накладывает особое условие, что для определения координаты нам нужно минимум три передатчика, соответственно, чем больше передатчиков, тем лучше. Каждый передатчик испускает сигнал с определенной мощностью. Чем мощность сигнала больше, тем мы ближе к источнику. Зная расположение источника сигнала и его мощность, мы можем определить расстояние от каждого источника до приемника. Учитывая, что сигнал от каждого источника распространяется в определённом радиусе, то по пересечению окружностей мы можем определить относительную координату. Такой метод достаточно сложен в техническом исполнении, так как требует установки большого количества дополнительных устройств и наладки их взаимодействия, что скажется на стоимости и мобильности комплекса, а также на сложности алгоритма работы. Также при таком методе следует учитывать, что в помещения распространение радиоволн затруднено из-за большого количества препятствий, что сказывается на точности позиционирования.

Есть и другое решение, не требующее внешних технически сложных устройств (радиомаяков или wi-fi модулей). Можно сосредоточить большую часть системы позиционирования на одной подвижной платформе, координаты которой мы хотим знать. Для этого можно использовать одновременно энкодеры, установленные на валах двигателей, для оценки перемещения платформы, и датчик акселерометра-гироскопа для отслеживания угла поворота, да и положения в пространстве в целом. Однако каждый из датчиков имеет определенные недостатки. При использовании энкодеров в течение работы накапливается ошибка, вызванная проскальзыванием колёс при передвижении по различным поверхностям, а также люфтом в деталях двигателя и редуктора. В свою очередь в связи с конструктивными особенностями датчик акселерометра-гироскопа даже находясь в статичном положении, имеет ненулевое значение угловой скорости, что приводит к постоянному смещению по всем трем осям. Компенсация с помощью акселерометра возможно только по тем осям проекции, которых по отношению к вектору гравитации являются ненулевыми. Для коррекции по оси сонаправленной с вектором гравитации может быть использован магнетометр – магнитный компас. Но

данное устройство является восприимчивым к внешним помехам, что приводит к снижению надежности такого метода коррекции.

Приведенный метод позиционирования обладает высокой точностью определения местоположения (несколько десятков сантиметров), но из-за описанных ранее проблем со временем накапливается так называемая ошибка позиционирования. Для компенсации этой ошибки было принято решение применить методы компьютерного зрения с целью распознавания визуальных меток определенной структуры (рисунок 2.2) с заранее известным положением в помещении и коррекции координат робота в зависимости от положения метки относительно камеры.

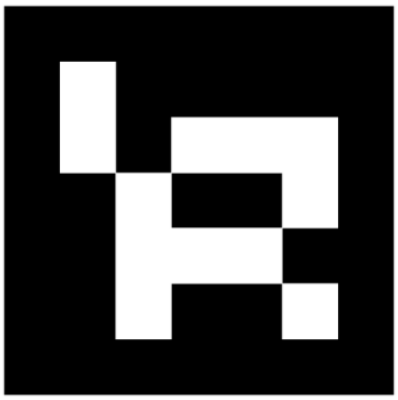


Рисунок 2.2 – Метка с идентификационным кодом

Таким образом, совместное использование энкодеров, акселерометра-гироскопа и элементов компьютерного зрения позволяет добиться достаточно высокой точности позиционирования. Также такой способ ориентации достаточно эффективен по сравнению с другими, так как не зависит от наличия внешнего сигнала, все вычисления выполняются внутри платформы. Он отличается дешевизной, так как не нужно обеспечивать большое количество внешних корректирующих устройств (радио или других технических маяков), надежностью потому, что содержит малое количество элементов, технической и программной простотой, высоким быстродействием и гибкостью системы, поэтому будем использовать именно этот метод позиционирования.

# 3 Разработка структурной схемы

Для выполнения задач, связанных как с компьютерным зрением, так и с функциями ведущего устройства, взаимодействующего с пользователем и управляющего работой всей платформы, потребуется устройство, которое обладает достаточной вычислительной мощностью, но при этом имеет небольшие габариты и энергопотребление. Для заданных условий оптимальнее всего будет подобрать одноплатный компьютер.

Так как функция обработки изображения затратна, как по времени обработки, так и по занимаемым ресурсам, то для управления движением лучше назначить отдельное устройство, которое будет только принимать команды от управляющего устройства (одноплатного компьютера), а задачи, связанные с передвижением, осуществлять самостоятельно. Для этих целей на этапе разработки прототипа можно выбрать функционально законченный микроконтроллер.

Обеспечение точного и безопасного перемещения платформы будем осуществлять при помощи набора датчиков. Для предотвращения столкновения с препятствиями оснастим платформу тремя ультразвуковыми датчиками расстояния. Для отсчета пройденного расстояния будем использовать энкодеры на основе цифровых датчиков Холла – по два на каждое колесо, а для отслеживания текущего положения в пространстве микроэлектромеханический датчик акселерометра-гироскопа.

Для движения платформы будем использовать два коллекторных двигателя, управление которыми осуществляется посредством драйвера.

Взаимодействие с пользователем будем осуществлять при помощи следующих устройств: компактной клавиатуры, беспроводной мыши и дисплея. Получение видео изображения будем осуществлять с веб-камеры.

Питание платформы разделим на две части – силовую и логическую. Так как двигатели потребляют много энергии, то и аккумулятор при работе будет разряжаться быстрее. В схеме с одним аккумулятором каждый раз при

необходимости снять аккумулятор придется обесточить всю платформу. Разделение питания позволит производить замену аккумулятора силовой части без необходимости выключения платформы. Также коллекторные двигатели при работе вносят в питающую сеть искажения. Если в общей сети есть устройства предъявляющие высокие требования к качеству питания, то такие искажения могут вызвать сбой в работе этих устройств. Для предотвращения таких сбоев пришлось бы устанавливать дополнительные фильтры.

Теперь, когда мы определились с основной структурой платформы подберем отдельные элементы.

Начнем с выбора одноплатного компьютера. Так как для коррекции ошибки позиционирования решено использовать методы компьютерного зрения необходимо чтобы центральное управляющее устройство - миникомпьютер обладал большой вычислительной мощностью. Также он будет использоваться для взаимодействия с пользователем (при задании команд, запуске и т.д.) и звуковым оповещением к примеру, при появлении на пути следования препятствий или возникновении каких-либо других событий, требующих звукового сопровождения. В таблице 3.1 приведены распространенные варианты одноплатных компьютеров.

Таблица 3.1 – Список и характеристики одноплатных компьютеров для выбора

| Наименование | Количество ядер/Частота процессора | ОЗУ | ПЗУ | Количество USB портов | Примерная стоимость |
|---|---|---|---|---|---|
| Raspberry Pi 2 Model B | 4 ядра / 900 МГц | 1 ГБ LPDDR 2 | MicroSD | 4 шт. | 2500 – 3500 руб. |
| Raspberry Pi 3 Model B | 3500 ядра / 1,2 ГГц | 1 ГБ LPDDR 2 | MicroSD | 4шт. | 3000 – 4000 руб. |
| A20-SOM- | 2 ядра / | - | MicroSD, 1ГБ | нет | 3500 – 4500 |

| 4GB | 1 ГГц | | NAND Flash | | руб. |
|---|---|---|---|---|---|
| Наименование | Количество ядер/Частота процессора | ОЗУ | ПЗУ | Количество USB портов | Средняя цена |
| A20-OLinuXIno-LIME2-4GB | 2 ядра / 1 ГГц | 1ГБ DDR3 | EEPROM 2 КБ, MicroSD, 4ГБ NAND Flash | 2 шт. | 6000 – 7000 руб. |
| BeagleBone Black Rev C | 1 ядер / 1 ГГц | 512МБ DDR3L | MicroSD, 4ГБ eMMC | 1 шт. | 7000 – 9000 руб. |
| Cubierboard 4 / CC-A80 | 8 ядер / 2 ГГц | 2 ГБ | 8 ГБ eMMC, MicroSD | 5 шт. | 11000 – 13000 руб. |

Исходя из таблицы 3.1 можно выделить два наиболее подходящих варианта – Cubierboard 4 / CC-A80 и Raspberry Pi 3 Model B. Данные модели обладают хорошей производительностью и достаточным количеством USB портов (нам необходимо минимум 3 для подключения периферийных устройств). Определенно одноплатный компьютер Cubierboard 4 / CC-A80 превосходит Raspberry Pi 3 Model B по большинству характеристик, однако его стоимость примерно в три раза выше. Для выполнения поставленной, в данной работе, задачи характеристики Raspberry Pi 3 Model B полностью удовлетворяют, значит нет необходимости выбирать более дорогой и мощный одноплатный компьютер.

Рассмотрим подробнее характеристики выбранного устройства. В основе данного одноплатного компьютера лежит четырёхядерный процессор ARM Cortex-A53, частота процессора составляет 1,2 ГГц и графический двухядерный процессора VideoCore IV способный обрабатывать HD-видео. Для подключения периферийных устройств данный миникомпьютер широким набором выводов и портов, к примеру, четыре USB порта, вывод для подключения Ethernet кабеля, HDMI выход для подключения монитора, стандартное гнездо 3,5 для подключения различных аудио устройств и др. Выбранный одноплатный компьютер изображен на рисунке 3.1.

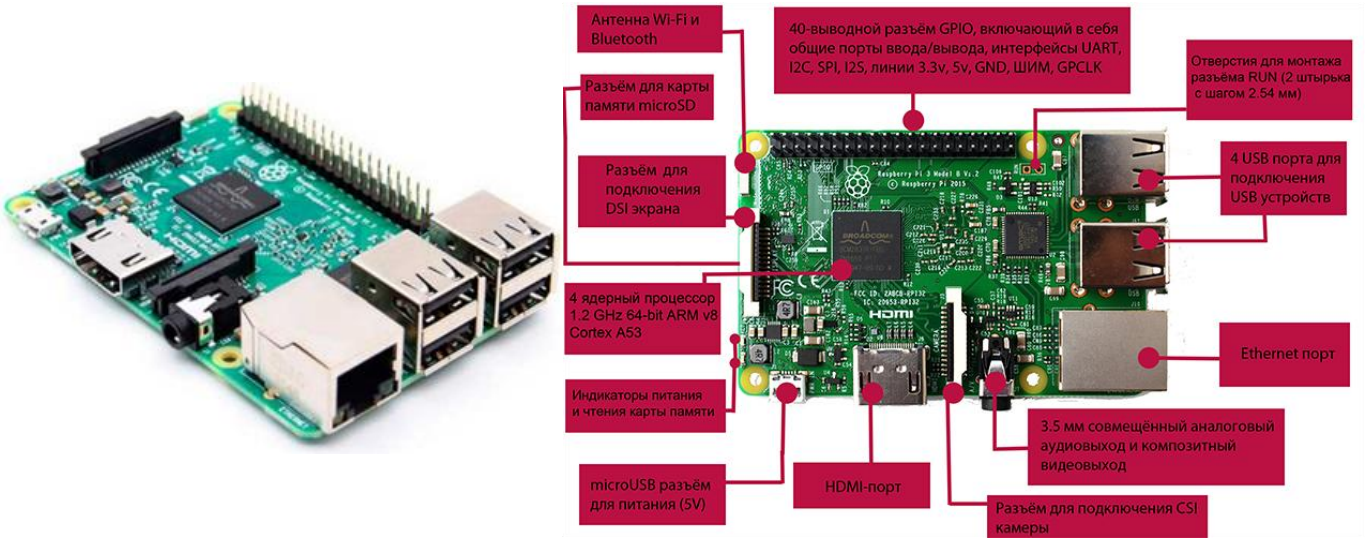



Рисунок 3.1 – Миникомпьютер Raspberry Pi 3

Теперь выберем периферийные устройства – клавиатуру, мышь, дисплей и динамик. Серьёзных требований к этим устройствам не предъявляется, однако важно, чтобы они были компактными и недорогими.

В качестве устройств ввода для экспериментального образца будет использоваться специальная небольшая клавиатура A4Tech X7 G100 и беспроводная компьютерная мышь Lenovo GX30H55791, они изображены на рисунке 3.2. Такой выбор обусловлен небольшими масса-габаритными параметрами клавиатуры и удобством управления мыши.

Рисунок 3.2 – клавиатура A4Tech X7 G100 и беспроводная компьютерная мышь GX30H55791

В качестве дисплея может быть использовано любое устройство с разрешением экрана минимум 640x350 и максимум 1920x1200 при подключении через HDMI порт Raspberry Pi 3. Однако для подвижной платформы оптимальнее всего будет использовать небольшой экран из-за ограничений по допустимой нагрузке и питанию. Поэтому будем использовать совместимый с Raspberry Pi 3 дисплей - 4.3 inch Display Waveshare Raspberry Pi Touch Screen Display Monitor 480x272 HDMI - LCD (рисунок 3.3).

Рисунок 3.3 – дисплей для вывода информации

В качестве аудио устройства выберем акустическую систему RITMIX SP-210 (рисунок 3.4) со встроенным аккумулятором. Она подключается через стандартный аудио разъем 3,5.


Рисунок 3.4 – Акустическая система RITMIX SP-210

Для экспериментальной версии платформы можно использовать любую компактную камеру. Выберем веб-камеру A4Tech PK-760E (рисунок 3.5) с разрешением 640x480.


Рисунок 3.5 – Веб камера A4Tech PK-760E.

Выбор обуславливается тем, что при данном разрешении изображения будет удовлетворительная частота обработки кадров (около 25 кадров в секунду), а максимальная дальность распознавания метки на листе формата А4 при должном освещении около 5 метров. Данная камера совместима с большинством типов устройств и не требует дополнительной установки драйверов.

Как уже было сказано ранее, для управления движением платформы будем использовать микроконтроллер, он позволяет не только разгрузить главное управляющее устройство – одноплатный компьютер, но и осуществлять независимое управление движением к примеру, когда в компьютере возникла какая-либо ошибка или неисправность. Основной задачей микроконтроллера будет опрос датчиков при движении, выдача управляющих воздействий на двигатели, и взаимодействие с одноплатным компьютером. В данной работе предлагается использовать семейство микроконтроллеров Arduino, потому что они обладают широким набором уже готовых решений – встроенный программатор, встроенный стабилизатор питания, доступная среда разработки, программные библиотеки – одним словом простота в работе с данным устройством. Приводить таблицу сравнений не имеет смысла, так как в основном все контроллеры отличаются только количеством портов ввода/вывода и количеством связанных с ними дополнительных устройств таких как АЦП, таймеры-счетчики и т.д. В данной работе будем

использовать микроконтроллер Arduino MEGA на базе микропроцессора Atmel ATmega2560 (рисунок 3.6).

Рисунок 3.6 – Arduino MEGA 2560

Данный микроконтроллер работает на частоте 16 МГц, в его составе присутствует 54 цифровых порта ввода/вывода 14 из которых могут работать в режиме широтно-импульсной модуляции, 16 аналоговых портов, встроенный программатор. Среда разработки программного обеспечения находится в свободном доступе и поддерживает программирование на языке C/C++. Так же в микропроцессор данного микроконтроллера встроены несколько таймеров счетчиков с возможностью внешнего тактирования. Для передачи данных поддерживаются интерфейсы UART, SPI и I2C. Микроконтроллер имеет встроенный линейный стабилизатор напряжения с входным напряжением 7 – 12 В и выходным стабилизированным напряжением 5 В. Основной причиной выбора данного микроконтроллера является большое количество портов ввода/вывода.

Для предотвращения столкновения с различными статическими объектами возможно использование датчиков расстояния таких как: ультразвуковые датчики, инфракрасные датчики расстояния, лазерные радары. В рамках данной работы

предлагается использовать три ультразвуковых датчика HC-SR04 (рисунок 3.7) расположив их так как показано на рисунке 3.8.

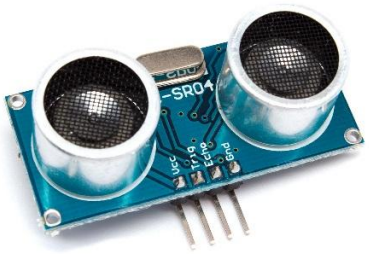



Рисунок 3.7 – Ультразвуковой датчик HC-SR04

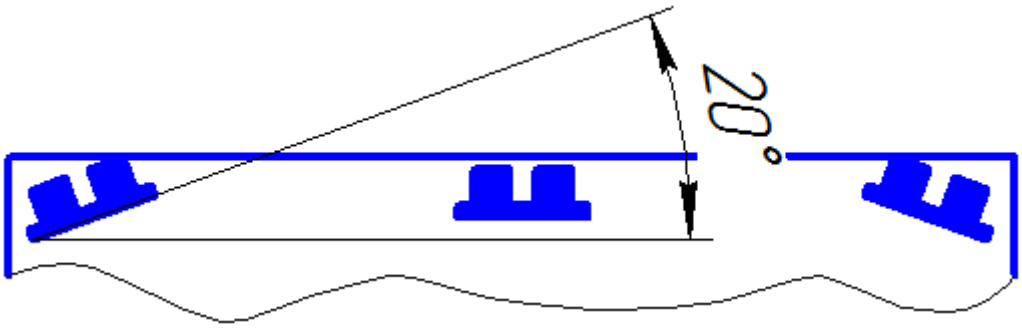



Рисунок 3.8 – Расположение ультразвуковых датчиков на платформе

Выбор обусловлен широкой доступностью и низкой ценой таких датчиков по сравнению с оптическими.

Для передвижения платформы решено использовать коллекторные двигатели постоянного тока GM25-370 (рисунок 3.9). Данный двигатель обладает встроенным мотор-редуктором с передаточным числом 75 к 1 и встроенным модулем с двумя датчиками Холла. Номинальное напряжение двигателя 9 В, номинальный ток 1,2 А, пусковой момент двигателя с редуктором 9,5 кг*см, а номинальный крутящий момент 3 кг*см.

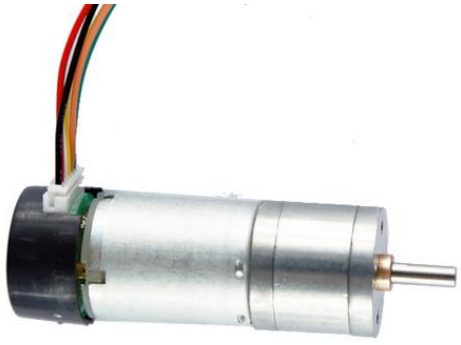


Рисунок 3.9 – Двигатель GM25-370

Выбор данного двигателя обусловлен тем, что он уже имеет встроенные датчики холла и необходимые элементы для их подключения, а также приемлемыми техническими характеристиками.

Для управления двигателями платформы будем использовать драйвер управления двигателями L298n (рисунок 3.10).

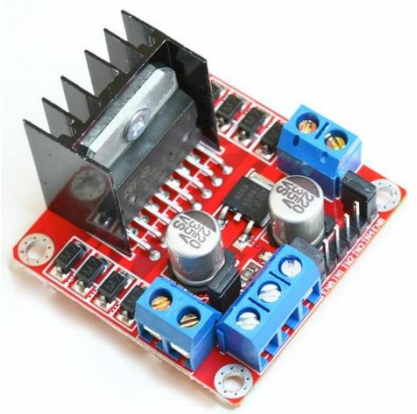


Рисунок 3.10 – Драйвер управления двигателями L298n

Для отслеживания текущего угла поворота лучше всего подойдет использование микроэлектромеханического (MEMS) датчика акселерометра-гироскопа. Данный вид датчиков имеет широкую выбор как по функциональности, так и по цене. В данной работе предлагается использовать наиболее дешевый датчик трёх осевого акселерометра-гироскопа MPU6050 (рисунок 3.11). Несмотря на низкую цену он обладает широким функционалом, таким как к примеру возможность генерирования прерываний по событиям и выполнение роли ведущего устройства. Более подробное описание этого устройства в пункте 6.1.

Рисунок 3.11 – Датчик акселерометра-гироскопа MPU6050

Питание платформы решено разделить на две части: силовую и сигнальную. Причиной этому являются повышенные требования к качеству питающего напряжения. Дело в том, что коллекторные двигатели во время работы могут отдавать в сеть различные высокочастотные помехи, а также просаживать напряжение питания из-за большой величины потребляемого тока. Для того чтобы на начальном этапе проверки работоспособности алгоритма избежать расчетов помехоподавляющих фильтров решено было разделить питание. Также разделение питания позволит, при разряде аккумулятора, заменить его без сброса питания сигнальной части.

В роли питающих элементов решено использовать литиевые трёхбаночные аккумуляторы схожие с тем что изображен на рисунке 3.12, с величиной выходного напряжения 11,1 В.


Рисунок 3.12 – Аккумулятор

Не все элементы сигнальной части способны питаться напрямую от аккумулятора. Для питания миникомпьютера и дисплея необходимо напряжения питания 5 В, однако они не оснащены встроенными преобразователями напряжения.

Поэтому, для питания от аккумулятора миникомпьютера будем использовать регулируемы импульсный преобразователь напряжения LM2596S (на рисунке 3.13 слева), а для питания дисплея будем использовать не регулируемый преобразователь KIS-3R33S (на рисунке 3.13 справа). Для микроконтроллера преобразователь не требуется, так как он уже оснащен встроенным стабилизатором напряжения с допустимым входным напряжением 7-12 В.

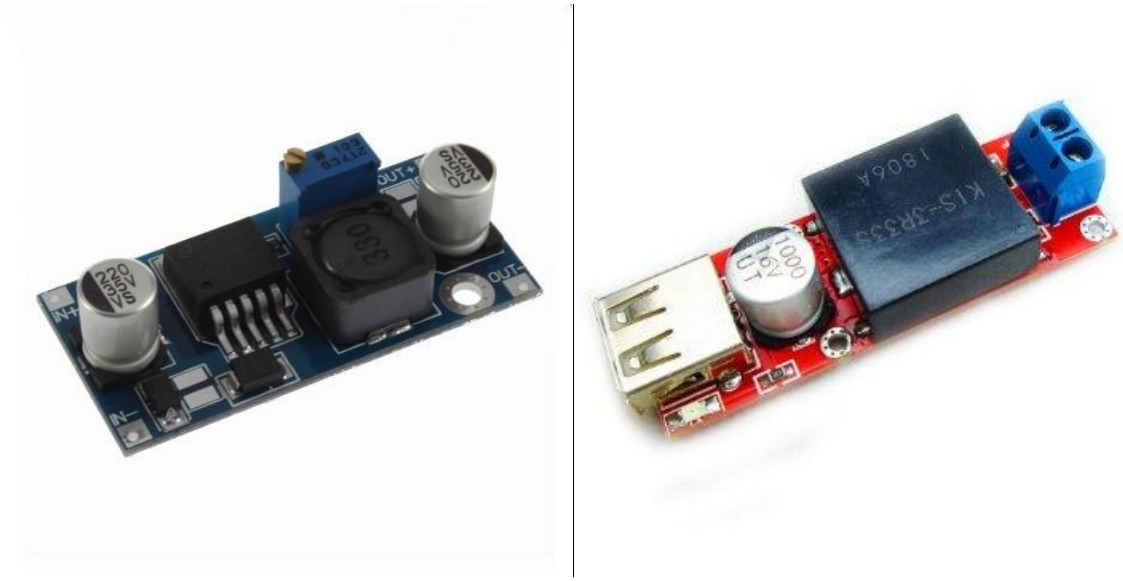



Рисунок 3.13 – Преобразователи напряжения LM2596S слева и KIS-3R33S справа

Питание микроконтроллера осуществляется напрямую от аккумулятора, так как он обладает встроенным линейным стабилизатором напряжения, а вот питание миникомпьютера и дисплея осуществляется через импульсные преобразователи, потому что данные устройства не обладают встроенной стабилизацией питания.

Таким образом, на основе выбранных элементов и описанной структуры составим структурную схему мобильной робототехнической платформы (рисунок 3.14).

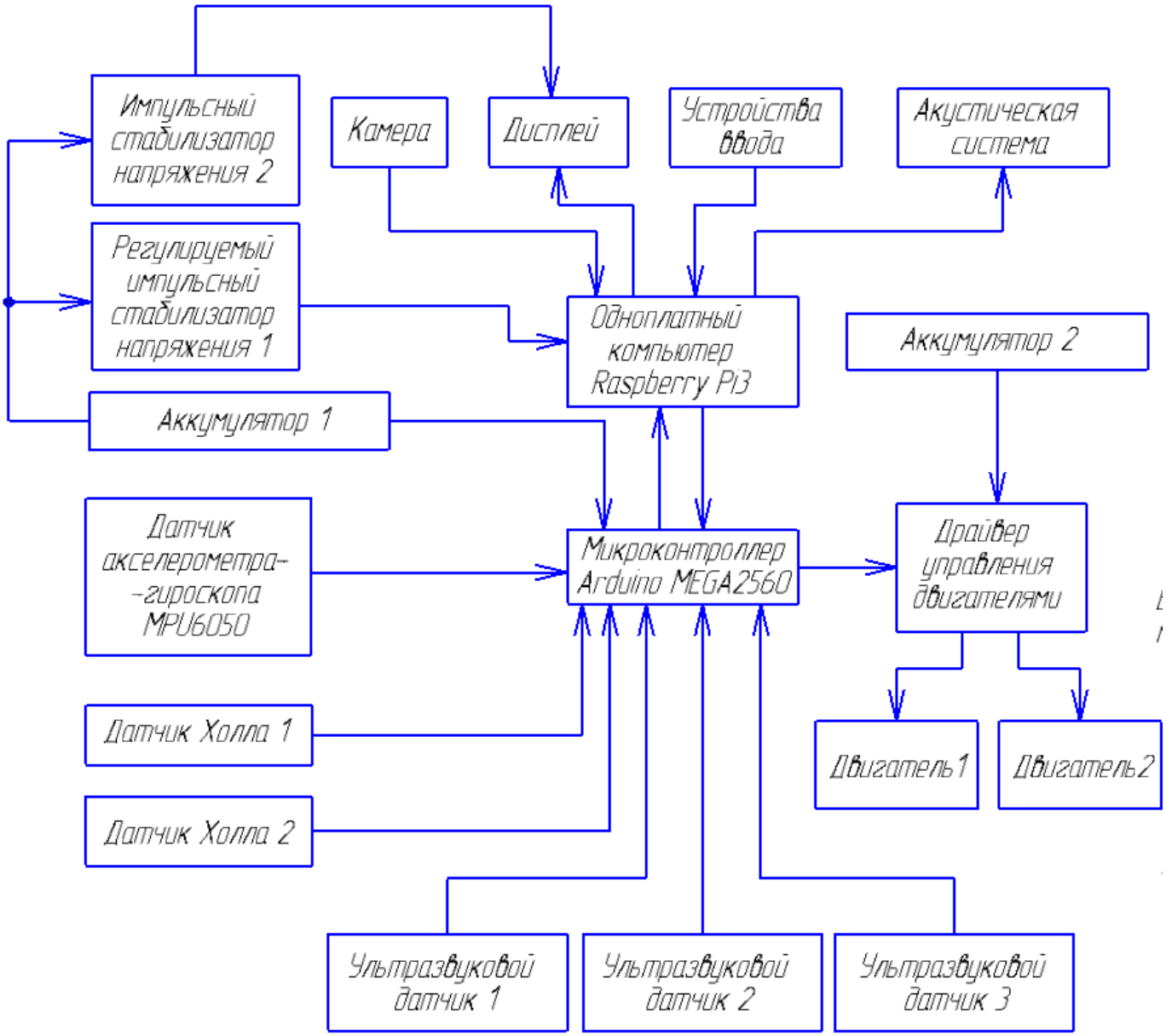



Рисунок 3.14 – Структурная схема мобильной робототехнической платформы

# 4 Разработка протокола связи между узлами платформы

Взаимодействие микроконтроллера и одноплатного миникомпьютера осуществляется по универсальному асинхронному приёмопередатчику (УАПП или UART) через преобразователь UART - USB. Микроконтроллер и миникомпьютер соединяются кабелем.

Так как прием и передача ведутся асинхронно, а присылаемые данные сохраняются в буфер ограниченного размера, то возможна ситуация, когда происходит переполнение буфера данных и часть данных будет утеряна. Для предотвращения такой ситуации необходимо привести пересылаемые сообщения к определенному формату.

Форматирование пересылаемых команд заключается в следующем – необходимо производить передачу данных в сообщениях определенной структуры: должен быть определен символ начала и конца сообщения, порядок расположения отдельных фрагментов сообщения, а также разделение фрагментов сообщения.

Определимся со структурой сообщения. Для разделения определения начала и конца сообщения будем использовать символы квадратной скобки: «[» - для обозначения начала сообщения и «]» для обозначения конца сообщения. Само сообщение будет состоять из трёх параметров:

1. Команды;
2. Координаты 1 или действия 1, или может отсутствовать;
3. Координаты 2 либо третий параметр может отсутствовать.

Разделение параметров будем осуществлять символом пробела или значение «32» в ASCII кодировке. Таким образом,формат сообщения получается следующий:

[Команда координата1/действие/отсутствует координата2/отсутствует]

В таблице 4.1 представлен набор используемых сообщений.

Таблица 4.1 – Сообщения, пересылаемые между миникомпьютером и микроконтроллером

| Общий вид сообщения | Пример | Описание |
|---|---|---|
| Сообщения, посылаемые от миникомпьютера к микроконтроллеру. | | |
| [MOV x y] | [MOV 500 100] | Движение в точку с координатами x y (в сантиметрах). |
| [LOC x y] | [LOC 505 110] | Коррекция текущих координат x y (в сантиметрах). |
| [TURN angle] | [TURN 150] | Установить угол ориентации платформы 150 градусов. |
| [TEST] | - | Подтверждение соединения между миникомпьютером и микроконтроллером. |
| Сообщения, посылаемые от микроконтроллера к миникомпьютеру. | | |
| [ONLINE] | - | Ответ на запрос о подтверждении соединения |
| [ENDMOV] | - | Ответ, посылаемый при завершении поставленной задачи |
| [OBS] | - | Obstacle – команда, отправляемая миникомпьютеру при появлении на пути следования препятствия |

Так как описанный метод является, по сути, универсальным и может использоваться для передачи данных в любых устройствах, передающих данные по асинхронным интерфейсам, то имеет смысл его программно оформить как отдельный функциональный блок при помощи объектно-ориентированного программирования.

Но прежде чем писать программу нужно сначала определиться с её структурой.

Состоять она будет из трёх функций, две из которых будут с общим доступом, а одна только для работы внутри класса.  Первая функция – чтение входящих данных, вторая – отправка данных, третья – получение из сообщения нужного параметра. Создадим заголовочный файл с названием USBRW что означает - Universal Serial Bus Read/Write. Текст с объявлением переменных и прототипов функций находится в приложении А.

Перейдем к описанию исполняемого файла и реализации функций. Первая функция – функция чтения данных из буфера входных данных. Блок-схема, описывающая работу функции чтения приведена на рисунке 4.1. Алгоритм работы достаточно прост. Если у нас есть данные в буфере, то считываем байт из буфера. Если байт соответствует символу начала сообщения устанавливаем флаг чтения. Затем проверяем установлен ли флаг чтения данных. Если не установлен, то переходим к чтению следующего байта, если установлен, то проверяем соответствует ли байт символу конца сообщения. Если байт соответствует символу конца сообщения, то сбрасываем флаг чтения и возвращаемся к чтению следующего байта, если же есть данные, то записываем их в массив. Если буфер пуст, то выходим из функции. Затем выделяем все параметры из считанного сообщения и возвращаемся к точке вызова функции в основной программе.

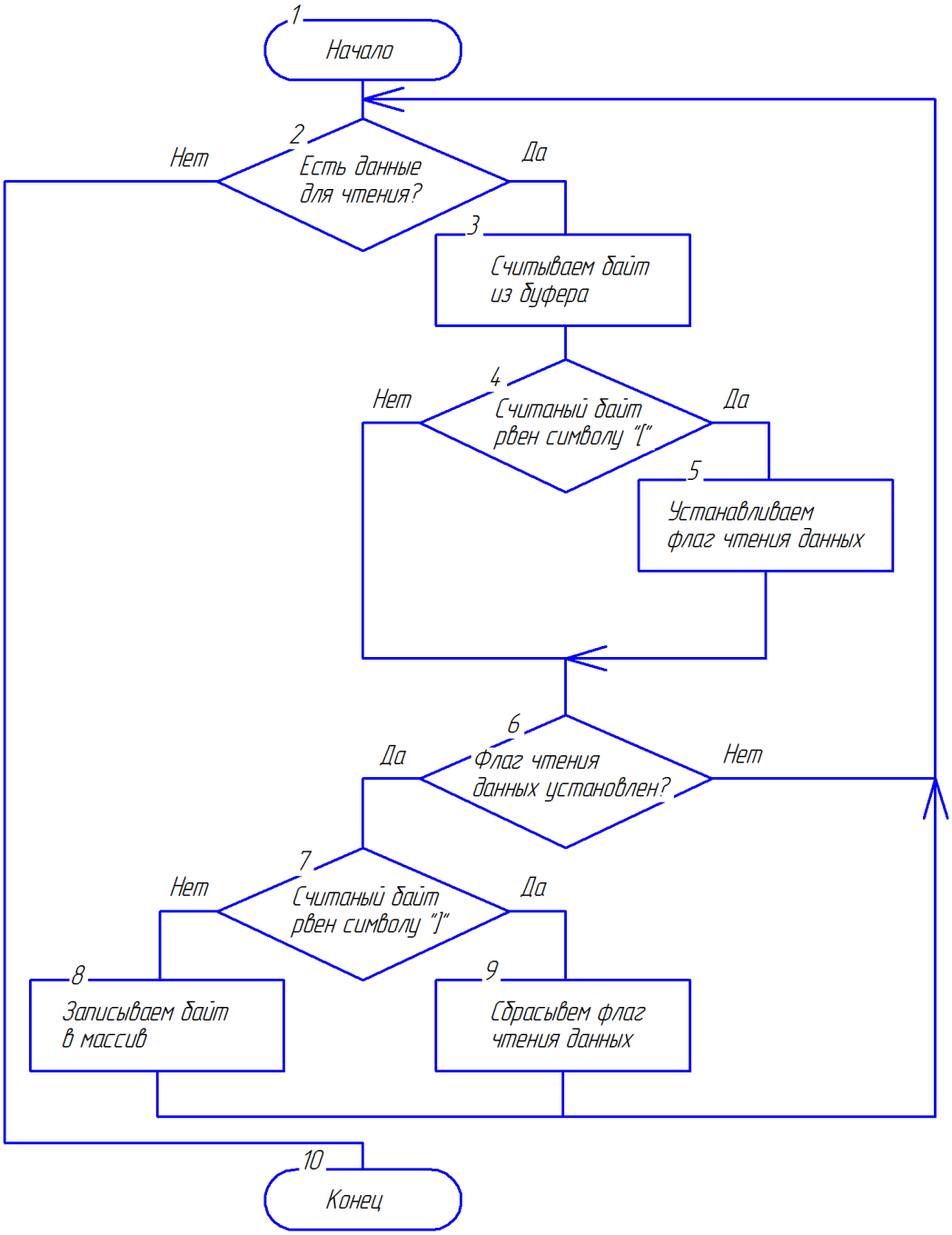



Рисунок 4.1 – Блок-схема алгоритма чтения данных

Программа, описанная блок-схемой находится в приложении А.

Для выделения данных из массива создадим отдельную функцию. Эта функция будет доступна только в пределах данного класса. Её задачей будет возвращение параметра указанного порядка (первый, второй, третий и т.д., в нашем случае параметров максимум может быть три). Блок-схема алгоритма работы функции показана на рисунке 4.2.

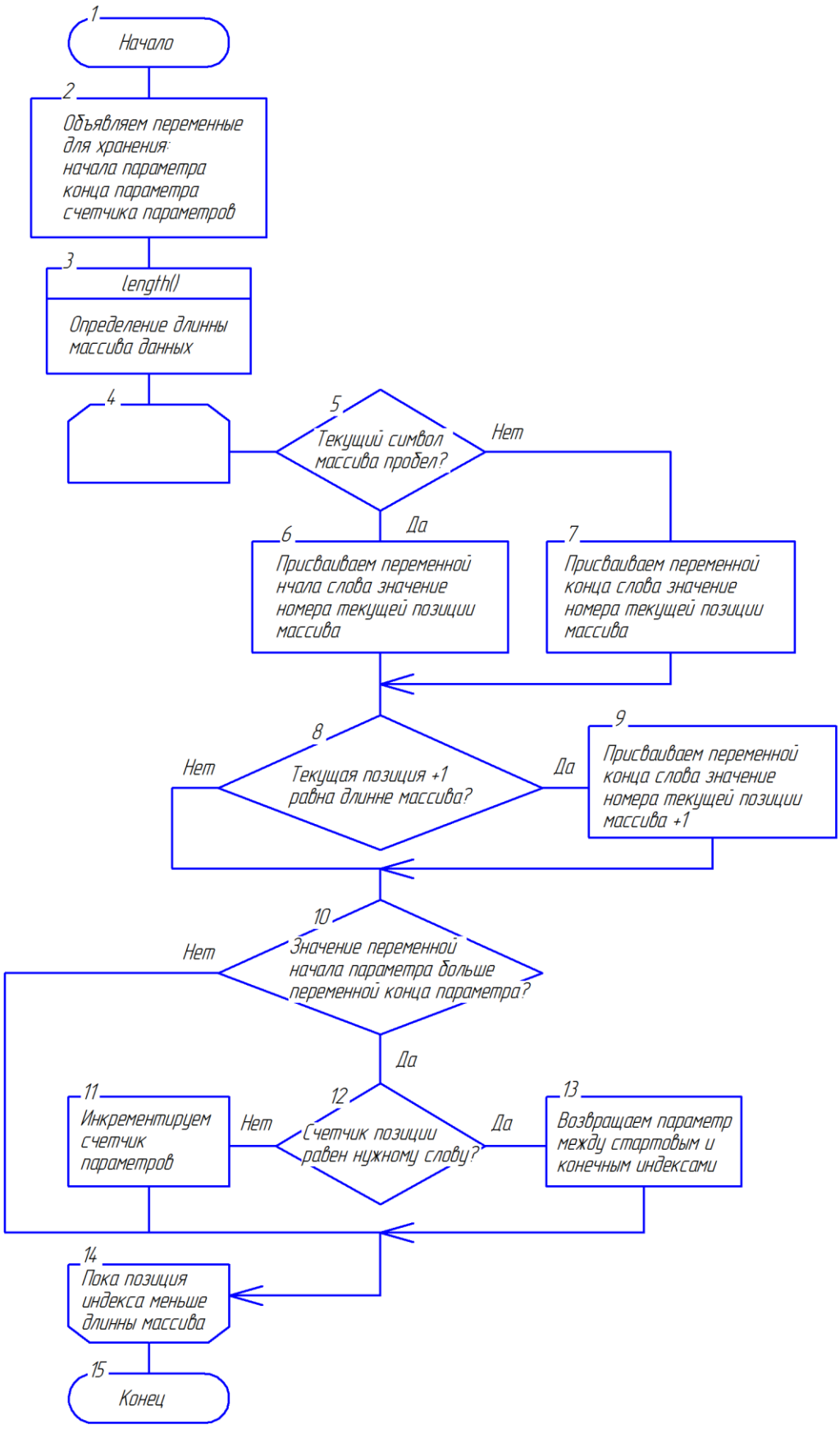



Рисунок 4.2 – Блок-схема алгоритма работы функции выделения параметров из сообщения

Начинается функция с объявления переменных для хранения начала параметра, конца параметра и счетчика параметров. Для определения длинны массива данных воспользуемся встроенной функцией length() принадлежащей классу String. Затем запускаем цикл, в котором на каждой позиции массива проверяем наличие символа разделения параметров. Если таковой присутствует, то проверяем нужный ли это по счету параметр. Если да, то возвращаем из функции параметр между стартовой и конечной позицией в массиве, если нет, то инкрементируем счетчик параметров. Проводим такое действие до тех пор, пока не найдем нужный параметр или пока не закончится массив. Программная реализация приведена в приложении А.

Для отправки данных используется небольшая и простая функция. В неё передаётся массив байт и она отправляет его с учетом описанного формата сообщений. Так как программа небольшая и простая она не нуждается в блок-схеме алгоритма работы. Программная реализация в приложении А.

# 5 Разработка электрической схемы соединений

Схема электрических соединений состоит из следующих функциональных блоков:

1. Одноплатный компьютер Raspberry Pi 3 model B;
2. Микроконтроллер Arduino MEGA 2560;
3. Мышь компьютерная Lenovo GX30H55791;
4. Клавиатура A4Tech X7 G100;
5. Веб-камера A4tech PK-760E;
6. Дисплей совместимый с Raspberry Pi 3 model B;
7. Акустическая система RITMIX SP-210;
8. Микроэлектромеханический датчик акселерометра-гироскопа MPU6050;
9. Драйвер управления двигателями L298n;
10. Двигатели GM25-370 со встроенными датчиками Холла;
11. Ультразвуковые датчики HC-SR04;
12. Аккумуляторы;
13. Регулируемый DC-DC преобразователь напряжения LM2596S;
14. Не регулируемый DC-DC преобразователь напряжения KIS-3R33S.

Соединение одноплатный компьютер – микроконтроллер выполняется при помощи шнура с контактами вилка USB A – вилка USB B.

Соединение не регулируемый преобразователь – дисплей и регулируемый преобразователь – одноплатный компьютер выполняется при помощи шнура с контактами вилка USB A – вилка microUSB.

Соединение одноплатного компьютера и дисплея выполняется кабелем HDMI.

Схема электрическая соединений представлена на рисунке 5.1. Оформление схемы производилось в системе автоматизированного проектирования КОМПАС-3D V16.

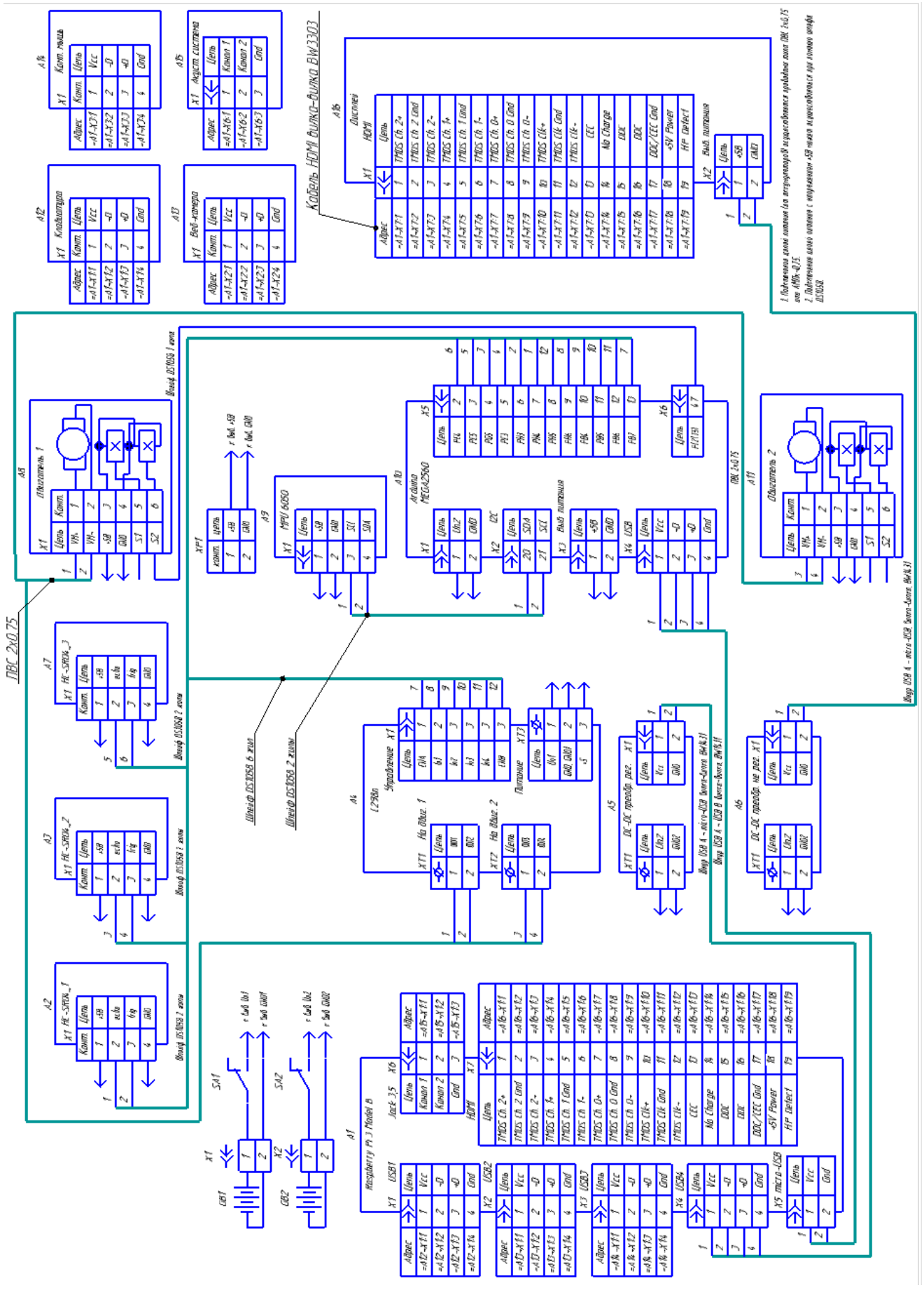


Рисунок 5.1 – Схема электрическая соединений

# 6 Разработка алгоритма движения робота

## 6.1 Описание работы с датчиком акселерометра-гироскопа

Для расчета угла поворота тележки используется датчик MPU6050. MPU6050- это микроэлектромеханический (MEMS) датчик, который имеет в своём составе трёх осевой акселерометр и гироскоп, 16 битный аналогово-цифровой преобразователь, низкочастотный цифровой фильтр, контроллер, который обрабатывает и хранит данные с акселерометра и гироскопа, буфер FIFO для удобного доступа к данным при работе с DMP, а так же сам программируемый DMP (Digital Motion Processor). Взаимодействие с датчиком осуществляется по последовательной шине связи I2C. Особенностью этого датчика является то, что на линии он может выступать как ведущее устройство, взаимодействующее, к примеру, с магнетометром, так и ведомое, выполняя команды какого-либо контроллера. Датчик обладает большим диапазоном чувствительности – у акселерометра диапазоны плюс минус 2, 4, 8 и 16 g (1g = 9.8 м/$с^2$), у гироскопа плюс минус 250, 500, 1000 и 2000 градусов в секунду. Необходимая чувствительность настраивается при помощи специально отведенных регистров управления. Устройство обладает широким функционалом, в нем присутствует возможность генерирования прерываний по наступлению определенных событий, к примеру обнаружение свободного падения, или же переполнения буфера данных FIFO, весь список возможностей находится в документации. Встроенный в датчик DMP процессор позволяет сразу, на основе полученных данных с акселерометра и гироскопа, производить расчет текущего положения в заданном виде к примеру, в углах Эйлера или кватернионах.

Для удобной работы с датчиком существуют библиотеки I2cdev и MPU6050. Библиотека I2Cdev упрощает работу с передачей и приёмом данных с датчика, а библиотека MPU6050 позволяет быстро настроить датчик, облегчает запрос данных с датчика, а также его программирование если оно необходимо (программа записывается в DMP). Написание программ производится в Arduino IDE. Отметим,

что фактически, подключаемые библиотеки являются классами C++ поэтому далее будут именоваться - класс.

Прежде чем получать данные с датчика его необходимо настроить. Для проверки работы датчика напишем тестовую программу.

Подключаем библиотеки для удобной работы с датчиком:

```
#include <I2Cdev.h>

#include <MPU6050.h>
```

Объявляем объект класса MPU6050:

```
MPU6050 mpu;
```

Далее проводим предварительную настройку необходимых элементов микроконтроллера и подключенных устройств в стандартной функции setup():

```
void setup() {
  Wire.begin();           // Запускаем интерфейс I2C
  mpu.initialize();       // Запускаем датчик MPU6050
  Serial.begin(9600);           // Запускаем последовательный порт и задаём скорость передачи данных 9600 BoD
}
```

Необходимо объявить целочисленные 16 битные переменные для хранения ускорений и угловых скоростей по осям x, y, z:

```
int16_t ax, ay, az, gx, gy, gz;
```

Теперь напишем основной текст программы который будет выполняться в бесконечном цикле – это стандартная функция loop(). Для запроса данных с датчика будем использовать метод класса MPU6050 - getMotion6( int16_t, int16_t, int16_t, int16_t, int16_t, int16_t):

```
void loop() {
```

```
// put your main code here, to run repeatedly:

mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // Запрашиваем данные с датчика

// Выводим данные в монитор порта

Serial.print("accel/gyro: \t");

Serial.print(ax); Serial.print("\t");

Serial.print(ay); Serial.print("\t");

Serial.print(az); Serial.print("\t");

Serial.print(gx); Serial.print("\t");

Serial.print(gy); Serial.print("\t");

Serial.println(gz);

delay(1000);  // Задержка опроса (не менее 20 мс)

}
```

На рисунке 6.1.1 приведены данные, получаемые с датчика MPU6050.

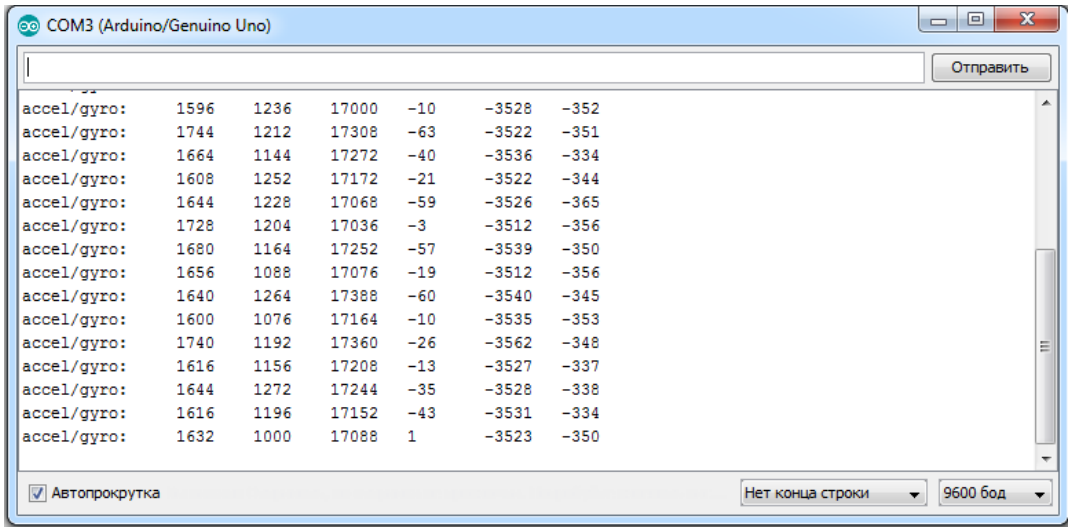



Рисунок 6.1.1 - Данные полученные с датчика MPU6050

На рисунке 6.1.1 первые три столбца это ускорение по осям X, Y и Z соответственно, вторые три столбца это угловые скорости вокруг осей X, Y и Z.

С датчика нам приходят данные которые имеют размерность разрядов АЦП. Для преобразования полученных данных в размерность g и градусов в секунду нам необходимо поделить их на коэффициент определяющий количество разрядов АЦП на единицу измерения: в случае акселерометра на 1g, в случае гироскопа количество разрядов на 1 градус в секунду. Величина коэффициента зависит от выбранного максимального диапазона измерений. Так как мы не производили настройку диапазона измерений, то используются минимальные диапазоны: для акселерометра плюс минус 2 $g$, для гироскопа плюс минус 250 град./сек.. Численные значения коэффициентов можно посмотреть в документации на датчик MPU6050 (приложение Б). Исходя из нашего диапазона чувствительности (в документации в графе conditions обозначены FS_SEL=0 и AFS_SEL=0 для гироскопа и акселерометра соответственно), определим наши коэффициенты: в строке Sensitivity Scale Factor находим обозначения FS_SEL=0 и AFS_SEL=0 и смотрим коэффициенты – для гироскопа он равен 131, а для акселерометра 16384. Однако, давайте разберемся откуда берутся такие числа. Нам известно что на датчик установлен 16 битный АЦП значит максимальное закодированное число равно $2^{16}$ или 65536. Так как 1 бит резервируется под знак то полученные данные разделим на 2 и получим 32768 - это означает, что максимальное значение, которое может выдать АЦП от плюс 32768 до минус 32767, теперь для того, чтобы найти сколько разрядов АЦП соответствуют 1g для акселерометра или 1 градусу в секунду для гироскопа, разделим максимальное возможное значение АЦП на максимальный диапазон чувствительности (пример для гироскопа) 32768 разделить на 250 равно 131,072. Это означает, что величина угловой скорости 1 градус в секунду будет представлена числом 131 в десятичном наборе разрядов АЦП. Теперь, для того чтобы найти величину углового ускорения, нам нужно приходящие данные разделить на 131.

Однако, есть некоторая особенность. Если посмотреть на данные на рисунке 6.1.1 кажется, что гироскоп находится в движении, хотя на самом деле он стоит не

месте неподвижно. Это смещение обуславливается погрешностью самого чувствительного элемента датчика, преобразующего механическое воздействие в электрический сигнал. Следует отметить, что величина смещения по-разному варьируется от датчика к датчику и соответственно настройка каждого датчика осуществляется индивидуально. Для компенсации этого смещения в датчике имеется ряд регистров. В эти регистры записывается величина смещения оси и в дальнейшем датчик сам будет отнимать заданную величину и записывать в свои регистры данных уже скомпенсированные значения. Для упрощения записи в классе MPU6050 присутствуют методы с названиями setAccelOffset и setGyroOffset.

Рассмотрим на примере компенсацию смещения данных с датчика. Модифицируем нашу программу, написанную ранее. Для этого в функции setup() для каждой из осей будем вызывать методы setAccelOffset() и setGyroOffset() для каждой из осей акселерометра и гироскопа, в которые будем передавать значения смещения используя целочисленный тип данных:

```
mpu.setXAccelOffset(0);
mpu.setYAccelOffset(0);
mpu.setZAccelOffset(0);
mpu.setXGyroOffset(0);
mpu.setYGyroOffset(0);
mpu.setZGyroOffset(0);
```

Методика следующая:

1. Смотрим данные с акселерометра и гироскопа при нулевом смещении.
2. Путем перебора подбираем коэффициенты так, чтобы значения угловых скоростей вокруг осей X, Y и Z были максимально близки к нулю.
3. Так же путем перебора подбираем коэффициенты для акселерометра, однако, для него значения ускорений должны быть нулевыми только по двум из осей, а по третьей значение должно быть близко к величине 1g.

Для хорошей точности коррекции лучше всего отображать величины ускорений и угловых скоростей в разрядах АЦП.

Для нашего датчика получили следующие коэффициенты:

mpu.setXAccelOffset(-5920); // Коррекция смещения акселерометра по оси X

mpu.setYAccelOffset(-911); // Коррекция смещения акселерометра по оси Y

mpu.setZAccelOffset(154); // Коррекция смещения акселерометра по оси Z

mpu.setXGyroOffset(12); // Коррекция смещения гироскопа по оси X

mpu.setYGyroOffset(891); // Коррекция смещения гироскопа по оси Y

mpu.setZGyroOffset(89); // Коррекция смещения гироскопа по оси Z

На рисунке 6.1.2 изображены данные, присылаемые с датчика, после коррекции встроенными средствами.

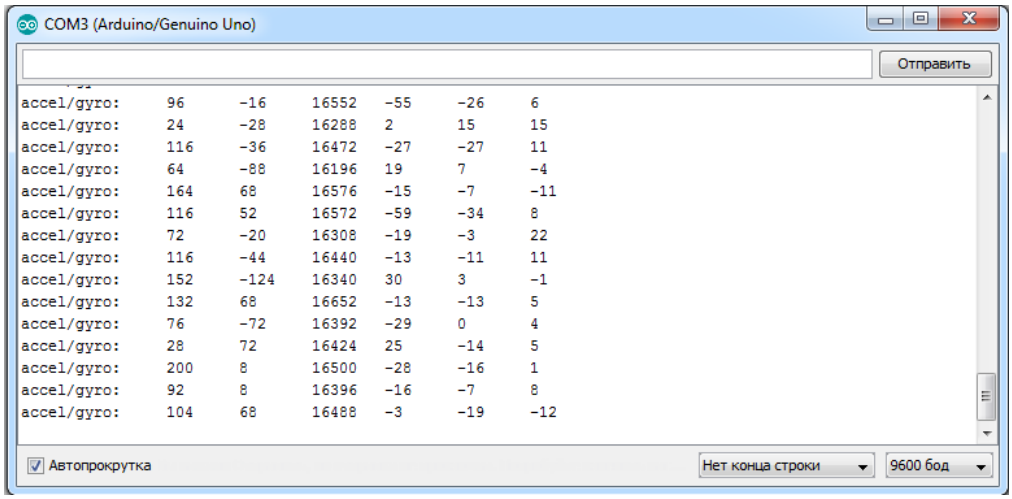



Рисунок 6.1.2 - Данные с акселерометра и гироскопа после коррекции показаний датчика

Как видно на рисунке 6.1.2 данные достаточно близки к нулевым по сравнению с данными на рисунке 6.1.1, однако всё же присутствует некоторый разброс значений. У гироскопа этот разброс меньше, у акселерометра больше, что объясняется конструкцией и принципами работы чувствительных элементов устройства.

Для устранения шума данных воспользуемся встроенными в датчик цифровым фильтром.

Настройка работы фильтра так же представлена в виде записи определенных значений в отведенный регистр. Для быстрой настройки фильтров воспользуемся методом класса MPU6050 setDLPFMode(int). Метод setDLPFMode(int) определяет работу низкочастотного фильтра, который фильтрует данные с гироскопа и акселерометра, а также определяет частоту опроса чувствительного элемента датчика. Параметр, передаваемый методу можно взять из таблицы на странице 13 документации на датчик (приведена в приложении Б).

Функция настройки фильтра заносится в функцию setup() и имеет следующий вид:

```
mpu.setDLPFMode(5);         // Задаём режим работы фильтру
```

На рисунке 6.1.3 представлены данные с датчика после подключения фильтра. Видно, что разброс данных уменьшился, поэтому можем точнее настроить смещение данных. Для настройки воспользуемся ранее описанной методикой. После коррекции получили новые коэффициенты смещения:

```
mpu.setXAccelOffset(-5930); // Коррекция смещения акселерометра по оси X
mpu.setYAccelOffset(-908);  // Коррекция смещения акселерометра по оси Y
mpu.setZAccelOffset(149);   // Коррекция смещения акселерометра по оси Z
mpu.setXGyroOffset(15);     // Коррекция смещения гироскопа по оси X
mpu.setYGyroOffset(889);    // Коррекция смещения гироскопа по оси Y
mpu.setZGyroOffset(88);     // Коррекция смещения гироскопа по оси Z
```

На рисунке 6.1.4 представлены данные после коррекции смещения данных. Исходя из полученных данных видно, что хоть разброс данных и уменьшился, они все равно имеют некоторое смещение и некоторый разброс. Так как аппаратных средств коррекции недостаточно будем использовать программные методы. Варианты коррекции и их реализацию мы рассмотрим чуть позже.

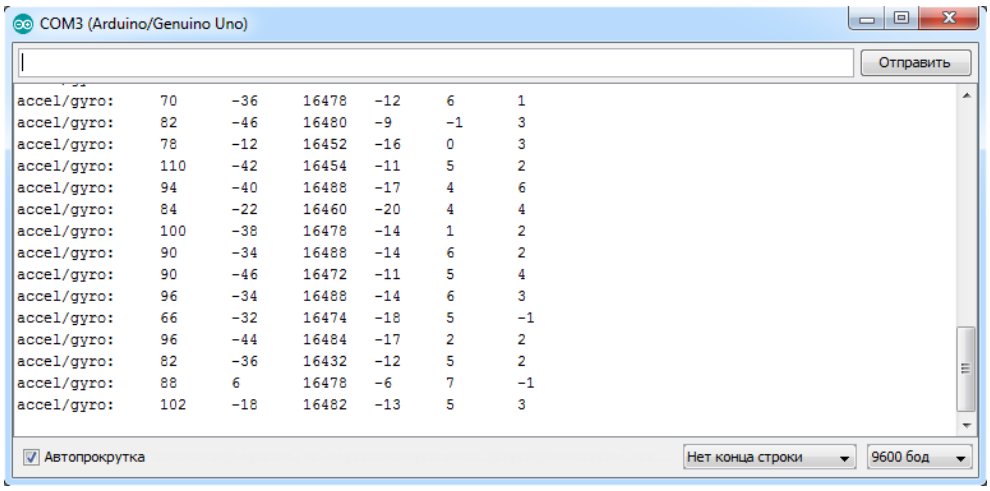



Рисунок 6.1.3 - Данные с акселерометра и гироскопа после подключения фильтров

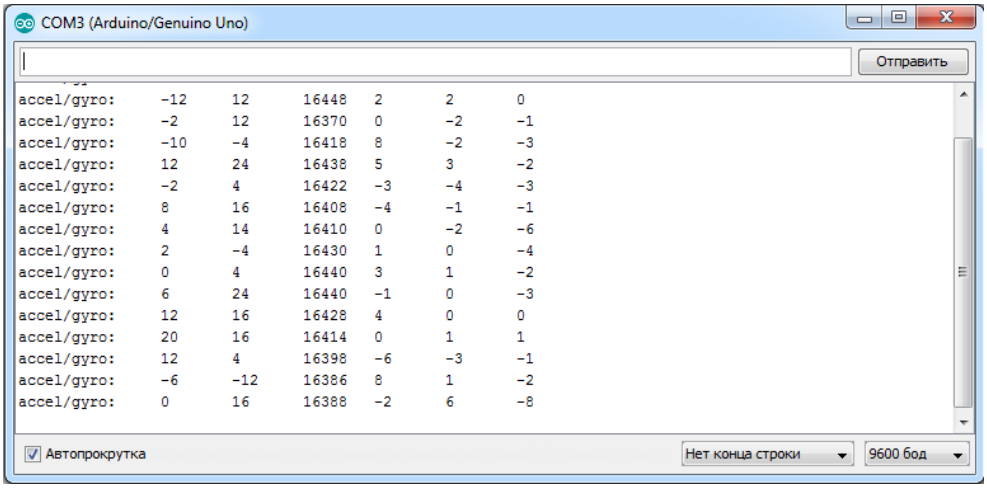



Рисунок 6.1.4 - Данные с акселерометра и гироскопа после коррекции смещений

Так как теперь мы знаем, как настраивать базовые параметры датчика, получать от него данные и преобразовывать их в общепринятые величины, теперь мы приступим к рассмотрению того, как используется выбранный датчик в нашей платформе. При расчетах текущей ориентации на прототипе платформы будем считать, что передвигается он по ровной, относительно горизонта, поверхности и для вычисления ориентации будем использовать только ось Z гироскопа, с диапазоном измерений плюс минус 2000 град./сек.

Для того, чтобы получить угол текущей ориентации относительно начального положения при помощи гироскопа, необходимо проинтегрировать угловую скорость по времени. Для интегрирования данных воспользуемся методом дискретного интегрирования – методом трапеций – формула (6.1.1):

$$\alpha = \alpha_{\text{п}} + \frac{\vartheta + \vartheta_{\text{п}} * T}{2}, \quad (6.1.1)$$

где $\alpha$- рассчитываемый угол; $\alpha_{\text{п}}$- угол в предыдущий момент времени; $\vartheta$- угловая скорость; $\vartheta_{\text{п}}$- угловая скорость в предыдущий момент времени; $T$- время между отсчетами.

Выбор данного метода обусловлен тем, что по сравнению с методом прямоугольников он имеет большую точность, и вместе с тем не требует больших вычислительных мощностей микроконтроллера в отличии от более точных, но ресурсоёмких методов таких как метод Гаусса-Кронрода или методы Рунге-Кутты и Кутты-Мерсона.

Ранее мы увидели, что даже после применения аппаратных методов устранения помех разброс данных все равно остаётся, при интегрировании это приведет к накоплению ошибки и соответственно к ошибке позиционирования платформы. Для устранения таких ошибок зачастую используют либо комплементарный фильтр, либо фильтр Калмана. Суть комплементарного фильтра заключается в том, что для вычисления текущего положения в пространстве (угла) мы используем как данные с акселерометра, так и данные с гироскопа. Так как акселерометр достаточно стабильно вычисляет текущее положение в состоянии покоя, то мы можем использовать эти значения для коррекции данных с гироскопа которые постоянно «уплывают». Однако этот фильтр корректирует данные по осям, которые не со направлены с вектором гравитации, к примеру, если с осью гравитации со направлена ось Z то при вращении вокруг этой оси не будет происходить изменение проекций векторов осей X и Y и значит коррекции не будет. Для коррекции «уплывания» данных по оси Z нужно использовать магнетометр, однако не во всех датчиках он имеется, да и само по себе это устройство сильно подвержено влиянию внешних помех. В фильтре Калмана несколько иной принцип, там мы фактически предсказываем изменение положения по текущим и предыдущим данным. Данный фильтр достаточно сложен для программной реализации и на начальном этапе его применение не имеет смысла. Так как магнетометр у нас отсутствует, то и комплементарный фильтр применить мы не

сможем. Значит пойдем другим путем. Программно ограничим чувствительность датчика. Для нашего датчика значения уровень шумов не превышают 8 разрядов АЦП что при диапазоне измерения в плюс минус 2000 град./сек. чуть менее половины градуса. Такая точность вполне нам подходит так что все значения ниже 8 разрядов АЦП будут приниматься за 0.

Рассмотрим программную реализацию вычисления текущего угла. Вся программа представлена в приложении В. Рассмотрим основные отличия от ранее описанной программы опроса.

Нужно объявить переменные для хранения текущего угла угловой скорости на текущем и предыдущем шаге – это необходимо для вычисления угла при помощи метода дискретного интегрирования:

float angularVelosity = 0.0f, prevAngularVelosity = 0.0f;

Задаём порог чувствительности, ниже которого данные будут приниматься за 0:

const int16_t MIN_GZ = 8;

Также необходимо объявить переменные для вычисления шага интегрирования:

float time1 = 0.0f, prevTime = 0.0f;

Для удобства работы объявим переменную хранящую приращение угла на текущем шаге, а также переменную для хранения абсолютного угла поворота:

float delta_angle = 0.0f;

float gyro_angle = 90.0f;

Объявим переменную для расчета изменения угла между опросамивызовами функции вычисления текущего положения:

float deltaPhi = 0.0f;

Создадим саму функцию вычисления текущего угла поворота:

void updateAngleMPU(){ // Функция опроса гироскопа

В предыдущем примере использовалась функция mpu.getMotion6(int*, int*, int*, int*, int*, int*) для запроса данных с датчика MPU6050. В данной программе используется прямой доступ к регистру данных посредством встроенной библиотеки Arduino – Wire. Такой метод получения данных работает незначительно быстрее, но в целом не имеет различий с применением mpu.getMotion6(int*, int*, int*, int*, int*, int*). Устанавливаем соединение с устройством, имеющим адрес 0x68 (как раз наш датчик MPU6050):

Wire.beginTransmission(0x68);

Назначаем регистр с которого хотим начать считывать данные:

Wire.write(0x47);  // starting with register 0x3B (ACCEL_XOUT_H)

Не закрываем линию связи:

Wire.endTransmission(false);

Так как нам необходимо только угловая скорость вокруг оси Z, то запрашиваем 2 байта (2 регистра) начиная с установленного ранее (0x47):

Wire.requestFrom(0x68, 2, true);

Записываем значения угловой скорости в размерности разрядов АЦП:

int16_t gz = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

Сразу проверяем условие не входит ли считанное значение в зону нечувствительности, которую мы определили ранее, если не входит, то рассчитываем угол, иначе обновляем время опроса и выходим из функции:

if (gz > MIN_GZ || gz < -MIN_GZ) {

Переводим значения угловой скорости из размерности разрядов АЦП в размерность градусы/секунду по ранее описанному принципу:

```
angularVelosity = gz / 16.4f;
```

Рассчитываем шаг интегрирования:

```
time1 = (micros() - prevTime) / 1000000.0f ;
```

Рассчитываем приращение угла и интегрируем данные:

```
delta_angle = ((angularVelosity + prevAngularVelosity) * time1) / 2.0f;

gyro_angle += delta_angle;

deltaPhi += delta_angle;
```

Производим запись необходимых параметров для расчетов на следующем шаге интегрирования:

```
prevAngularVelosity = angularVelosity;

prevTime = micros();
```

Переводим текущий угол в систему отсчета 0 - 360 градусов. Если угол больше 359,99 отнимаем 360. Если меньше нуля прибавляем 360:

```
gyro_angle = gyro_angle > 359.99f ? (gyro_angle - 360.0f) : gyro_angle;

gyro_angle = gyro_angle < 0 ? gyro_angle + 360.0f : gyro_angle;

}
```

## 6.2 Вычисление пройденного расстояния. Работа с энкодерами

Для расчета, пройденного платформой расстояния, будем использовать энкодеры. Энкодер – это устройство, преобразующее какой-либо вид перемещения (линейное, угловое) в последовательность электрических импульсов. Различают несколько основных видов энкодеров: оптические, резистивные и электромагнитные. В нашем устройстве будет использоваться так называемый электромагнитный поворотный энкодер, который преобразует изменение угла в последовательность электрических импульсов. Чувствительным элементом в этом

устройстве будет цифровой датчик Холла, а источником магнитного поля круговой магнит. Вся конструкция представляет из себя двигатель с закрепленным на валу круговым магнитом и двумя датчиками Холла отстоящими друг от друга на некотором расстоянии, конструкция изображена схематично на рисунке 6.2.1. При повороте вала на 360 градусов энкодер выдаёт по одному импульсу с каждого датчика. Наличие двух датчиков даёт возможность определить не только количество оборотов, но и направление вращения вала.

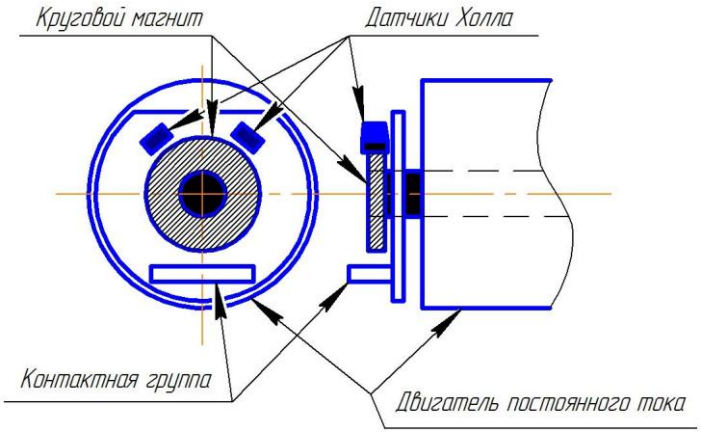



Рисунок 6.2.1 – Расположение энкодера на двигателе постоянного тока

Рассмотрим несколько методов считывания информации с энкодеров при помощи микроконтроллера. Самый простой метод — это циклический опрос выводов микроконтроллера, к которым подключены сигнальные выводы энкодеров. Такой метод применим, если частота работы микроконтроллера много больше частоты следования импульсов. В таком случае у микроконтроллера будет возможность успевать выполнять не только опрос, но и необходимые преобразования данных, а также какие-либо другие действия. Если же мы знаем, что частота работы микроконтроллера не позволяет гарантированно успевать фиксировать каждый импульс с датчика энкодера, то для этой задачи можно определить устройство, работа которого не занимает вычислительных ресурсов микроконтроллера. Таким устройством может быть таймер-счетчик, для которого источником тактовых импульсов может быть наш Энкодер. Микропроцессор ATmega 2560 имеет 5 встроенных таймер-счетчиков способных работать от внешнего тактирования. В Arduino MEGA 2560, которая установлена на нашей

платформе, используется как раз данный микропроцессор, однако, только у пятого таймера счетчика вывод для внешнего тактового сигнала объединен с выводом микроконтроллера, у остальных таймеров-счетчиков связь с выводами микроконтроллера отсутствует. Это значит, что мы сможем обрабатывать информацию только с одного энкодера. В опытном варианте платформы этого будет достаточно.

Ниже приведем последовательность настройки таймера-счетчика микропроцессора от внешнего тактирования. Для настройки работы используются конфигурационные регистры TCCR5A и TCCR5B в соответствии с документацией (приложение Г, страницы документации 157, 160, 161). Начнем с рассмотрения того что записывается в регистр TCCR5A рисунок 6.2.2.

**Timer/Counter5 Control Register A – TCCR5A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | COM5A1 | COM5A0 | COM5B1 | COM5B0 | COM5C1 | COM5C0 | WGM51 | WGM50 | TCCR5A |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Рисунок 6.2.2 – Карта регистра TCCR5A.

В данном регистре биты COM5A1, COM5A0, COM5B1, COM5B0, COM5C1, COM5C0 определяют работу выводов микропроцессора OC5(A/B/C). Они необходимы для генерирования прерываний или других сигналов при работе таймера-счетчика в режиме таймера и в нашей задаче они не нужны. Так что все перечисленные биты в данном регистре должны быть сброшены в 0. Биты WGM51, WGM50 регистра TCCR5A (рисунок 6.2.2) и WGM53, WGM52 регистра TCCR5B (рисунок 6.2.3) определяют режим работы таймера-счетчика 5.

**Timer/Counter5 Control Register B – TCCR5B**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | ICNC5 | ICES5 | – | WGM53 | WGM52 | CS52 | CS51 | CS50 | TCCR5B |
| Read/Write | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Рисунок 6.2.3 – Карта регистра TCCR5B.

Всего существует четыре режима работы: нормальный режим (normal), режим CTC, и два режима широтно-импульсной модуляции. Не будем заострять

внимание на этих режимах, отметим только, что режим работы нашего таймера счетчика – нормальный, то есть в соответствии с документацией (приложение Г, станица 160 документации, таблица 82) все биты должны быть сброшены в 0. Бит ICNC5 регистра TCCR5B (рисунок 6.2.3) включает или отключает схему подавления помех. Если бит установлен в единицу, то срабатывание происходит при наличии четырех одинаковых выборок, соответствующих фронту сигнала, если бит установлен в ноль, то срабатывание происходит при первом же детектировании фронта. В нашем случае бит устанавливается в единицу. Бит ICES5 определяет активный сигнал для схемы подавления помех. Если бит установлен в единицу, то изменение состояние счетного регистра будет происходить по фронту сигнала, если в ноль – по спаду. В нашем случае бит устанавливается в единицу. Биты CS52, CS51, CS50 определяют источник тактирования. В соответствии с документацией на микропроцессор (приложение Г, страница 161 документации, таблица 83) для внешнего тактирования по фронту сигнала все биты устанавливаются в единицу. Так как таймер-счетчик 16 битный то результат счета хранится в двух байтах регистра TCNT5: TCNT5H – старшие биты, TCNT5L – младшие биты. Чтение и запись данных в данные регистры может быть осуществлена в любой момент времени.

Стоит отметить, что при чтении и записи данных регистров TCNT5H и TCNT5L нужно соблюдать строгую последовательность действий. Сначала нужно считать данные из младшего байта, а только затем из старшего. При записи наоборот сначала записываем старший байт, а затем младший. Это обуславливается тем, что нельзя прочитать два регистра одновременно. К примеру, в регистр было записано число 05FF, сначала мы считали старший байт, то есть 05, и пока мы считывали произошла инкрементация счетчика и в младший байт записалось значение 00, теперь мы считываем значение из младшего байта и получаем окончательное число 0500, хотя на момент начала чтения там было 05FF. Для того, чтобы избежать такого в таймере счетчике присутствует специальный временной регистр. При чтении из младшего байта в этот регистр автоматически передаётся значение старшего байта и когда мы читаем старший байт нам передаётся значение

этого регистра. Для того, чтобы этот регистр работал и нужна описанная выше последовательность чтения или записи.

Настройку работы таймера-счетчика производим в функции setup():

```
// Сначала на всякий случай обнуляем регистры настройки
TCCR5A = 0;
TCCR5B = 0;
// Настраиваем таймер/счетчик на внешнее тактирование с детектированием фронта и включаем схему подавления //помех (настроена на фронт сигнала)
TCCR5B |= (1 << ICNC5) | (1 << ICES5) | (1 << CS52) | (1 << CS51) | (1 << CS50);
// На всякий случай обнуляем счетный регистр
TCNT5H = 0;
TCNT5L = 0;
```

Напишем функцию для чтения данных из регистра таймера-счетчика. Пусть данная функция сразу производит расчет пройденного платформой расстояния в сантиметрах. Так как на нашей платформе установлены колёса диаметром 9 сантиметров и редуктор с передаточным числом 75 к 1 то за 1 поворот вала двигателя колесо нашей платформы будет перемещаться на 0,52 сантиметра. Значит,умножив количество отсчетов энкодера на 0,52, мы получим пройденный путь. Программная реализация:

```
float getWay() {
  // Считываем данные из регистра
  int cntNcode = TCNT5L;
  cntNcode |= TCNT5H << 8;
  // Возвращаем пройденный путь в сантиметрах
  return (cntNcode * 0.52); //редуктор 75/1, за 1 оборот вала двигателя колесо проходит расстояние 0,52 см
}
```

## 6.3 Управление двигателями платформы

Нам уже известно, что для передвижения платформы используются два коллекторных двигателя постоянного тока с максимальным напряжением питания 12В. Так как мощности микроконтроллера недостаточно для питания двух двигателей, для этого используется специальный драйвер L298n. Он позволяет управлять работой двигателей при помощи маломощного сигнала. При помощи драйвера возможно регулирование скорости вращения двигателей при применении широтно-импульсно модуляции (ШИМ) управляющего сигнала.

Рассмотрим подробнее устройство и работу с драйвером управления двигателями. Схема драйвера управления двигателями на базе схемы L298n представлена на рисунке 6.3.1. Устройство схемы L298n представлена на рисунке 6.3.2.

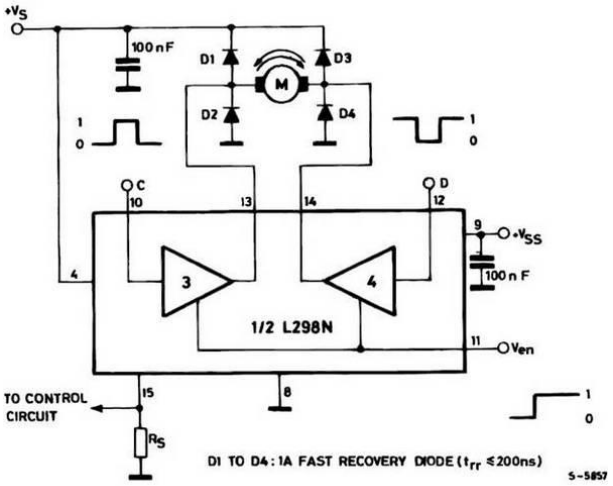



Рисунок 6.3.1 – Схема драйвера управления двигателями на базе схемы L298n, приведенная в документации на драйвер

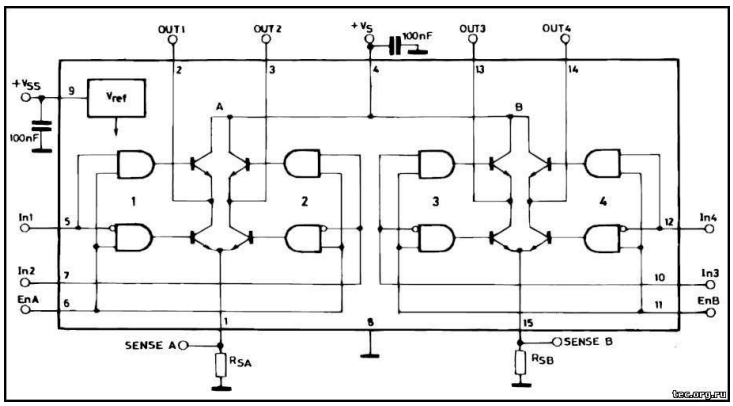



Рисунок 6.3.2 – Устройство схемы L298n, приведенная в документации на драйвер

В целом устройство схемы L298n достаточно просто. Она представляет из себя два транзисторных моста на биполярных транзисторах (рисунок 6.3.2). Каждый из мостов управляет одним двигателем, подключённым к выводам OUT1,2 (мост A) и/или OUT3,4 (мост B). Для задания направления вращения двигателя используются накрест лежащие пары транзисторов.

Для управления работой моста A используются выводы In1, In2 и EnA. Выводы In1, In2 задают направление тока в обмотке двигателя. Вывод EnA используется для управления скоростью вращения – на него подаётся ШИМ сигнал с микроконтроллера, либо, если управления дискретное, то подается логический 0 или логическая 1. Аналогичное управление осуществляется и мостом B.

На плате драйвера (рисунок 6.3.1) установлены также обратные диоды. При торможении двигатель переходит в режим генератора и сбрасывает энергию, запасенную в магнитном поле через открытые диоды в источник питания.

Логика драйвера может питаться как от одного силового источника, так и от другого внешнего источника питания. Стоит отметить, что для корректного управления работой двигателей, необходимо объединять землю источников питания. В нашей работе питание логики драйвера осуществляется от микроконтроллера напряжением 5 В.

Теперь рассмотрим управление двигателями на программном уровне. Для удобной работы с драйвером будем использовать объектно-ориентированное программирование.

Создадим класс с названием L298n – по названию самого драйвера, и объявим в нем конструктор и прототипы используемых функций (полный текст заголовка класса в приложении Д):

```
L298n(int ena, int enb, int in1, int in2, int in3, int in4);
void RotationForward(unsigned int PWM_left, unsigned int PWM_right);
void RotationBackward(unsigned int PWM_left, unsigned int PWM_right);
void RotationLR(int dir, unsigned int PWM_left, unsigned int PWM_right);
void Stop();
```

Так как функции, по сути, однотипны, то в качестве примера рассмотрим работу только одной из них. Пусть это будет функция с название RotationForward. Основной текст исполняемого файла приведем ниже (полная функция в приложении Д):

```
PWM_left = PWM_left > 206 ? 206 : PWM_left;
PWM_right = PWM_ right > 206 ? 206 : PWM_ right;
analogWrite(ENA, PWM_right);
analogWrite(ENB, PWM_left);
digitalWrite(In1, HIGH);
digitalWrite(In2, LOW);
digitalWrite(In3, LOW);
digitalWrite(In4, HIGH);
```

Первые две строки функции - это защита от перенапряжения двигателей, так как напряжение питания равно 11,1 В, а двигатель предназначен для напряжения 9В. Исходя из того, что ШИМ на выводах микроконтроллера 8-битный, то максимальное значение его $2^8$ или 255, что соответствует напряжению питания.

Чтобы определить какое значение ШИМ соответствует 9 В решим уравнение и округлим до меньшего значения (6.3.1):

$$x = \frac{9*255}{11,1} = 206,8 \approx 206 \quad (6.3.1)$$

Таким образом значение ШИМ равное примерно 9 В равно 206.

Как было описано ранее, чтобы управлять скоростью вращения двигателей нужно подавать ШИМ сигнал на выводы ENA и ENB драйвера L298n. Для этого используется функция analogWrite().

Для задания направления вращения двигателей используются цифровые порты микроконтроллера и выводы датчика In1, In2 – для первого двигателя, In3, In4 – для второго двигателя. Значение на выходе порта микроконтроллера устанавливается при помощи функции digitalWrite(). На какой вывод нужно подать логическую 1 а на кокой логический 0 зависит от полярности подключения двигателе к выводам OUT1, OUT2 и OUT3, OUT4.

## 6.4 Предотвращение столкновения с препятствиями

Как описывалось в разделе 4 для предотвращения столкновения с препятствиями используются 3 ультразвуковых датчика HC-SR04. Данный датчик является цифровым и имеет 4 вывода: Vcc, Trig, Echo и Gnd. Выводы Vcc и Gnd используются для питания датчика +5 В и земля соответственно. Вывод Trig нужен для формирования сигнала – звуковой волны. Согласно документации, на данный вывод нужно подать импульс длинною 10 мкс. После подачи сигнала на вывод Trig датчик формирует на выходе 8 импульсов частотой 40 кГц. Затем на выводе Echo выставляет высокий уровень напряжения (логическую 1) длительностью, пропорциональной времени прохождения волны от датчика до препятствия и обратно.

Для преобразования времени в расстояние до препятствия можно воспользоваться формулой (6.4.1):

$$S = \frac{\vartheta * t}{2}, \quad (6.4.1)$$

где $\vartheta$ – скорость звука в воздухе; $t$ – время прохождения звуковой волны от датчика до препятствия и обратно.

Рассмотрим программную реализацию опроса ультразвуковых датчиков. Прежде всего начнем с объявления номеров портов к которым подключены выводы Trig и Echo каждого датчика:

```
int echoPin[3] = {7, 4, 3}, // Выводы echo
    trigPin[3] = {6, 5, 2};   // Выводы trig
```

Для удобства инициализации портов объявим их в виде массива.

Далее объявим массив для хранения данных о расстоянии с каждого датчика. Пусть положение в массиве [0] - правый, [1] - средний, [2] – левый:

```
float distance[3];
```

Теперь необходимо сконфигурировать порты ввода/вывода микроконтроллера. Для этого в функции setup() в цикле настроим каждый из портов:

```
for (int i = 0; i < 3; i++) { // 3 т.к. 3 датчика
  // Порты с подключенными trig как выходные (отправляют данные/управляют чем-либо)
  pinMode(trigPin[i], OUTPUT);
  // Порты с подключенными echo как входные (принимают данные)
  pinMode(echoPin[i], INPUT);
}
```

После настройки и объявления переменных перейдем к рассмотрению функции опроса датчиков. Она достаточно проста – справа налево опрашиваем по очереди датчики. Формируем импульс длинной 10 мкс, затем при помощи встроенной функции pulseIn() получаем время прохождения волны от датчика до препятствия и обратно. Однако есть один нюанс. Функция pulseIn() забирает управление до тех пор, пока не придет звуковая волна и если возникает ситуация, когда волна не вернулась, то в функции срабатывает тайминг. Для того, чтобы

функция надолго не «зависала», установим тайминг равным 3000 мкс. Этого времени достаточно для детектирования препятствий на расстоянии до 51 см. Если в данной функции срабатывает тайминг, то она возвращает 0, что можно спутать с 0 расстоянием до препятствия, поэтому введём проверку на срабатывание тайминга: если функция вернула 0, то расстояние максимальное, т.е. 51 см. Программная реализация приведена в приложении Е:

## 6.5 Управление движением платформы

Для управления движением нашей платформы решено было использовать пропорционально интегрально дифференцирующее регулирование или сокращенно ПИД регулирование. Данный регулятор выдаёт управляющее воздействие, которое состоит из трёх составляющих: пропорциональной, интегральной и дифференциальной. Пропорциональная или П составляющая- это разность между желаемым и фактическим значением регулируемой величины, так же называемая сигналом рассогласования или ошибкой. Зачастую ошибка и необходимое значение управляющего воздействия сильно различаются, поэтому ошибку умножают на заранее подобранный коэффициент $K_p$, который или увеличивает, или уменьшает управляющее воздействие на заданную величину. В реальности, используя только пропорциональную составляющую, нельзя добиться желаемого значения регулируемой величины из-за так называемой статической ошибки. Эта ошибка обуславливается тем, что все реальные системы обладают инерционностью – это значит, что результат от нашего воздействия наступает не сразу, а в течение некоторого промежутка времени, так же при пропорциональном регулировании не учитывается воздействие на систему внешних факторов. Интегральная или И составляющая необходима как раз для компенсации влияния внешних факторов на регулируемую величину. Она накапливает это постоянное отклонение и корректирует значение управляющего воздействия так, чтобы скомпенсировать влияние внешних факторов. Скорость накопления ошибки зависит от выбранного коэффициента $K_i$. Чем он меньше, тем дольше происходит накопление и тем точнее мы можем скомпенсировать внешнее воздействие. Дифференциальная или Д

составляющая необходима для того, чтобы определить скорость изменения регулируемой величины. Она позволяет скомпенсировать инерционность реакции системы на управляющее воздействие. Величина дифференциальной составляющей так же определяется коэффициентом воздействия $K_d$. Напишем функцию ПИД регулятора исходя из выше описанного материала (6.5.1):

$$PID(t) = K_p error\ t\ + K_i \int_0^t error\ t\ dt + K_d \frac{derror(t)}{dt} = P + I + D, \quad (6.5.1)$$

где $PID\ t$ - управляющее воздействие в данный момент времени, $error\ t$ - сигнал рассогласования или ошибка, $K_p$- пропорциональный коэффициент, $K_i$- интегральный коэффициент, $K_d$-дифференциальный коэффициент, $P$- П составляющая регулятора, $I$ – И составляющая регулятора, $D$- Д составляющая регулятора.

Рассмотрим ПИД регулирование на примере нашей платформы. Здесь ошибкой будет являться разность между заданным и фактическим углом ориентации платформы (6.5.2):

$$error = targetAngle - platfAngle, \quad (6.5.2)$$

где $targetAngle$ - направление на заданную точку, $platfAngle$- текущий угол ориентации платформы. Интегральная составляющая будет компенсировать разность тяги двигателей при одинаковой ШИМ сигнала управления двигателями, а также различные физические параметры такие, как трение шестерен редукторов, температурное воздействие на драйвер управления и так далее. Дифференциальная составляющая будет регулировать скорость доворота платформы. Управляющим воздействием в нашем случае будет ШИМ сигнал, который подаётся на драйвер управления двигателями. Но прежде всего для корректной работы ПИД регулятора необходимо подобрать все его коэффициенты.

Существует несколько методов настройки ПИД регулятора. Один из методов – метод Циглера-Никольса. Сутью данного метода является то, что мы экспериментально подбираем такой пропорциональный коэффициент K, при котором система переходит в автоколебания и замеряем период этих колебаний. Затем на основании полученного коэффициента пересчитываем пропорциональный

коэффициент и рассчитываем интегральный и дифференциальный коэффициенты по известным формулам (6.5.3), (6.5.4), (6.5.5).

$$K_p = K * 0.6, \tag{6.5.3}$$

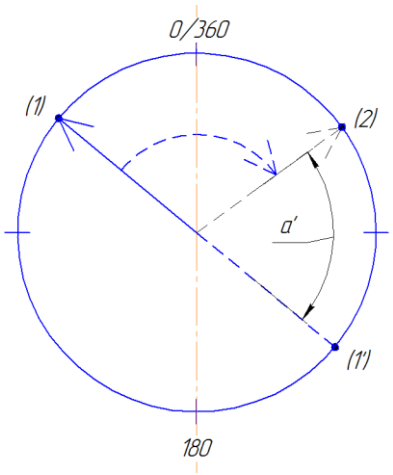
$$K_i = \frac{2K_p}{T}, \tag{6.5.4}$$

$$K_d = \frac{K_p T}{8}; \tag{6.5.5}$$

Другой метод основан на подборе каждого коэффициента вручную. Для этого обнуляют все коэффициенты и начинают увеличивать пропорциональный коэффициент. Если система выходит на заданное значение медленно этот коэффициент увеличивают, если же система нестабильна, около заданного значения коэффициент уменьшают, до тех пор, пока колебания не прекратятся. Затем настраивают дифференциальный коэффициент. При настройке данного коэффициента стоит учесть, что слишком большое его значение также может привести к колебаниям системы. Далее производят настройку интегрального коэффициента. Обычно значения данного коэффициента лежат в пределах 0,1…0,001. Настройку начинают с минимального значения и затем его постепенно увеличивают.

На начальном этапе для улучшения понимания влияния каждого из коэффициентов будем использовать второй метод подбора коэффициентов.

Теперь перейдем к рассмотрению алгоритма расчета ошибки. Для того, чтобы получить не только угол, но и кратчайшее направление разворота в



глобальной системе координат используется следующий алгоритм (рисунок 6.5.1).

Рисунок 6.5.1 – Поворот из одной полусферы в другую

Весь диапазон разбивается на две полусферы от 0 до 180 градусов и от 180 до 360. Если, к примеру, нужно осуществить поворот от направления (1) к направлению (2) (поворот из одной полусферы в другую) (рисунок 3), то сначала прибавляем к значению угла направления (1) 180 градусов и получаем угол направления (1’) формула (6.5.6):

$$1' = 1 + 180 \quad (6.5.6)$$

Затем сравниваем значение угла направления (1’) с углом направления (2). Если значения угла (2) больше, то значит угол поворота вправо (по пунктирной стрелке на рисунке 6.5.1) будет короче, чем противоположное, иначе поворот в противоположном направлении будет короче. Если фактическое (1) и ожидаемое направление (2) находятся в одной полусфере, то сравнение происходит без дополнительных вычислений. Результатом вычислений является угол, на который необходимо довернуться. Направление доворота определяется знаком, если «+», то против часовой стрелки, «-» - по часовой стрелке. Таким образом, путем простых вычислений, определяется оптимальное направление поворота.

После расчета ошибки необходимо выдать управляющее воздействие на двигатели платформы. В нашем случае управление осуществляется следующим образом: задаёмся средним значением ШИМ сигнала на драйвер двигателей. Для получения управляющего воздействия вычитаем (для левого двигателя) и суммируем (для правого двигателя) значение ошибки со средним значением ШИМ для каждого из двигателей. Отправляем сигнал на драйвер управления двигателями.

Для удобного управления воспользуемся методами объектно-ориентированного программирования. Создадим класс с названием Move, который будет осуществлять все команды движения платформы. Так как текст программы достаточно объёмный, он вынесен в приложение Ж. Описание программы в данном

разделе не осуществляется, так как текст программы подробно пояснён комментариями.

## 6.6 Расчет текущего местоположения

Местоположение платформы определяется в декартовой системе координат с координатами X и Y, а также углом поворота в системе 0 – 360 градусов, отсчет угла ведется от оси X. При запуске микроконтроллера координаты автоматически устанавливаются в ноль, а угол поворота в 90 градусов.

Порядок расчета следующий. Сперва вычисляется пройденное расстояния с момента предыдущего расчета с учетом криволинейного движения платформы. Так как данные поступают с одного энкодера, установленного на левом колесе, то для расчета перемещения середины платформы при повороте будем использовать формулу (6.6.1):

$$S = \varphi * R, \tag{6.6.1}$$

где S – путь, пройденный средней точкой; $\varphi$ – угол, на который повернулась платформа; $R$ – расстояние от колеса до центра платформы.

Для расчета суммарного пройденного расстояния суммируем данные с датчика и расчет пути при повороте по формуле (6.6.2):

$$D = D + S, \tag{6.6.2}$$

где D – расстояние, пройденное платформой.

Рассмотрим расчет координат на примере. Предположим платформе пришла команда передвигаться из точки с координатами (X1; Y1) в точку с координатами (X2; Y2) рисунок 6.6.1. В данном положении платформа рассчитает расстояние, которое нужно пройти в виде вектора (1), и угла «a» на который нужно довернуться (рисунок 6.6.1). Расчет длинны вектора производится по формуле (6.6.3):

$$L = \overline{(Y2 - Y1)^2 + (X2 - X1)^2}, \tag{6.6.3}$$

Угол на заданную точку рассчитывается через синус или косинус и длину вектора (по правилам тригонометрии). Однако в силу различных параметров платформа не сможет двигаться с места идеально по прямой в виду различных причин (инерционность двигателей, задержка управления и т.д.), её траектория движения будет представлена в виде кривой (2) рисунок 6.6.1.

Для точного расчета пройденного расстояния производится перерасчет текущей координаты. Он осуществляется через короткий точно заданный (дискретный) промежуток времени, что позволяет допускать, что платформа двигалась по прямой.

После каждого перерасчета координаты обновляются расстояние и угол, который необходимо установить, для следования к заданной точке рисунок 6.6.2. После пересечения круга радиусом 20 сантиметров будет считаться что платформа достигла точки назначения. В этот момент платформа пересчитает текущие координаты и отправит на миникомпьютер сигнал о достижении заданной точки.

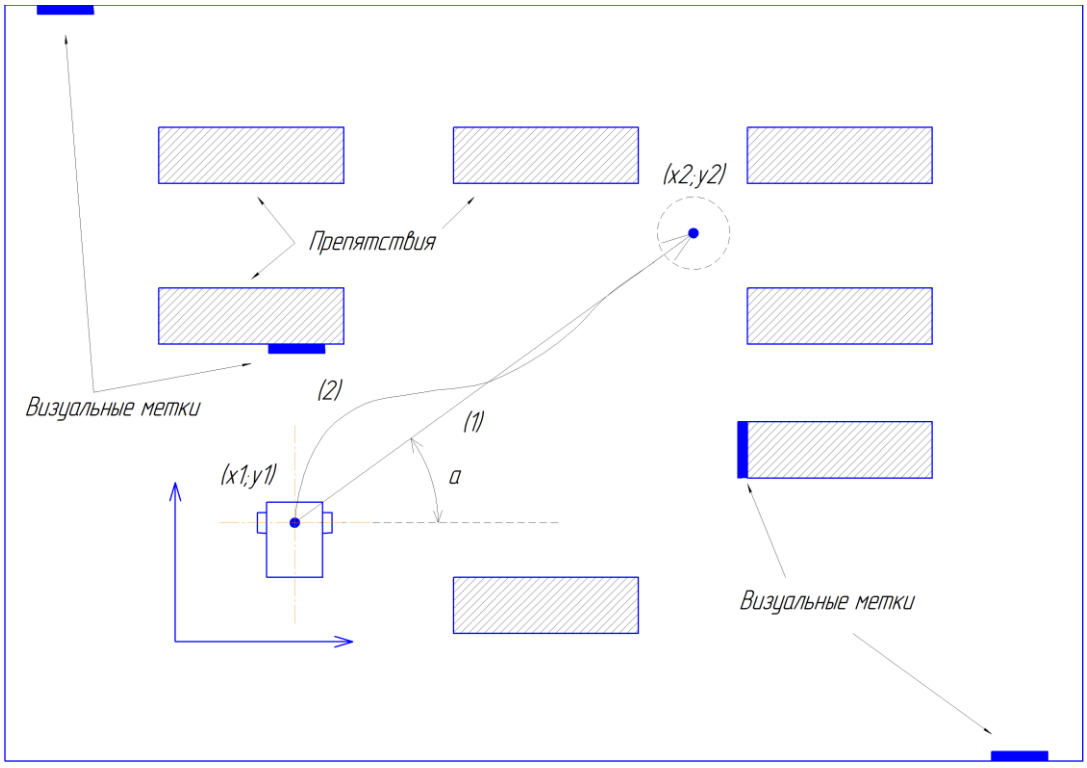



Рисунок 6.6.1 – Расчет координат

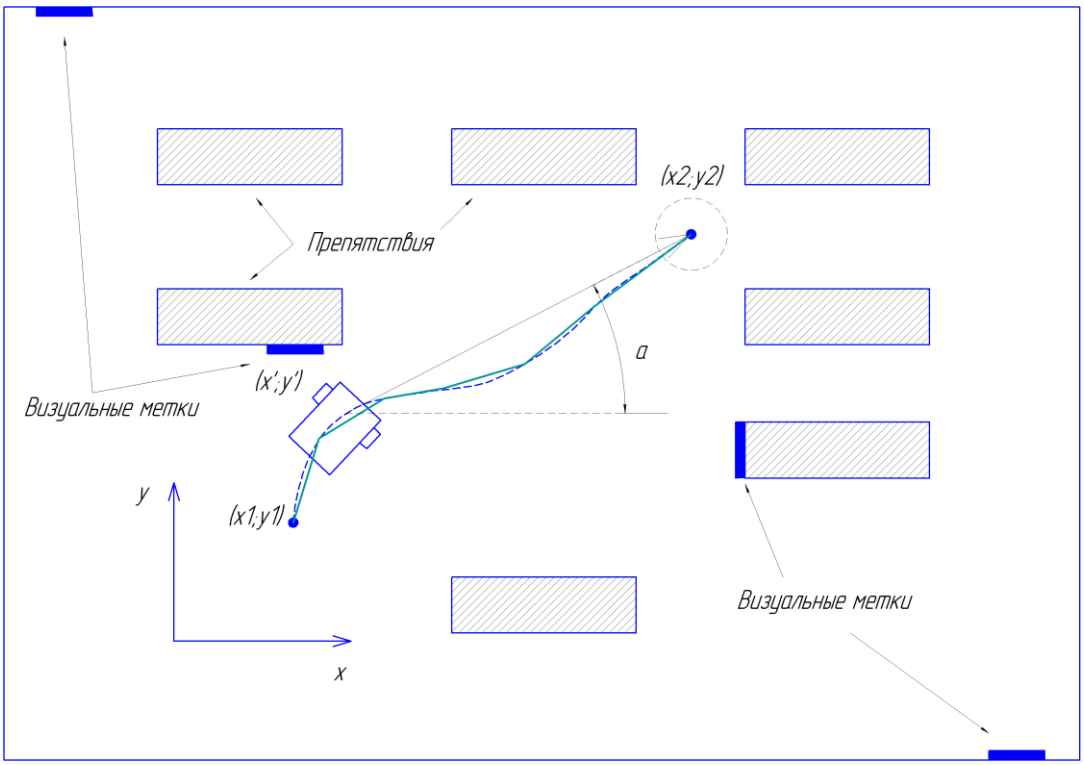



Рисунок 6.6.2 – Передвижение

Программная реализация приведенного метода расчета текущих координат будет представлена в виде класса на языке программирования C++ с названием Path. Этот класс будет содержать метод getTarget(float *, float *, float, float, float, float, float *, float *, float *), который принимает параметры необходимые для расчета (текущие координаты по осям X и Y, координаты заданной точки по осям X и Y, пройденное расстояние, текущий угол, изменение угла между расчетами, переменную для записи направления на заданную точку и переменную для записи расстояния до точки) и, соответственно, производит сам расчет всех необходимых для позиционирования данных. В данном разделе текст программы с подробными комментариями находится в приложении К из-за большого объёма.

## 6.7 Алгоритм работы платформы

Блок-схема алгоритма работы мобильной робототехнической платформы представлена тремя фрагментами, изображенными на рисунках 6.7.1, 6.7.2 и 6.7.3 соответственно. Далее представлено описание алгоритма работы.

В программе все действия выполняются циклично: на каждом шаге проверяются условия выполнения тех или иных действий. Не зависимо есть команда для действия или нет, на каждом шаге цикла опрашиваются ультразвуковые датчики и выставляется флаг наличия препятствий. Так же независимо, но каждые 15 мс опрашивается датчик акселерометра-гироскопа, для точности данных о угле поворота необходимо, чтобы заданный дискретный шаг соблюдался. Далее, на каждом шаге цикла проверяется наличие входящих данных. Данные начинают считываться, если в данный момент не выполняется никаких действий, то есть текущая команда «NAN». Если данные считались производится распознавание команды и выполнение соответствующих инструкций.

Далее происходит проверка есть ли команда к движению. Если есть команда к движению и флаг поворота сброшен, то проверяется прошло ли с предыдущего пересчета координаты 100 мс. Если прошло, то пересчитываем координаты. Затем проверяем сколько времени прошло с выдачи предыдущего управляющего воздействия. Если больше 30 мс, то выполняем проверку о том, насколько отличаются текущее направление движения и заданное. Если различие по модулю больше чем на 30 градусов, то выставляем флаг поворота. Далее происходит проверка о том, не наступило ли одно из событий для остановки. Событиями для остановки являются 4 параметра: расстояние до точки следования менее 20 см, наличие входящего сообщения, наличие флага поворота и наличие флага препятствия. Если есть одно из условий остановки, то платформа останавливается, затем проверяется по какому из событий произошла остановка. Если по флагу поворота, то дальше никаких действий не выполняем. Если по флагу препятствия, то посылаем сообщение о наличии препятствия. Если по наличию входящего сообщения, то устанавливаем команду «NAN» и идем на следующую итерацию цикла. Если по расстоянию до точки, то посылаем сообщение о завершении поставленной задачи.

Затем происходит проверка условия: была ли команда на поворот или был ли установлен флаг поворота. Если условие выполняется, то платформа выполняет поворот и на каждой итерации проверяет наличие флага завершения поворота. Если

таковой флаг выставлен, проверяется чем был вызван поворот. Если поворот был вызван флагом поворота, то сбрасываем флаг поворота и уходим на следующую итерацию цикла. Если поворот был по команде, то посылается сообщение о завершении поворота и выход из условия.

Программа с подробными комментариями приведена в приложении Л.

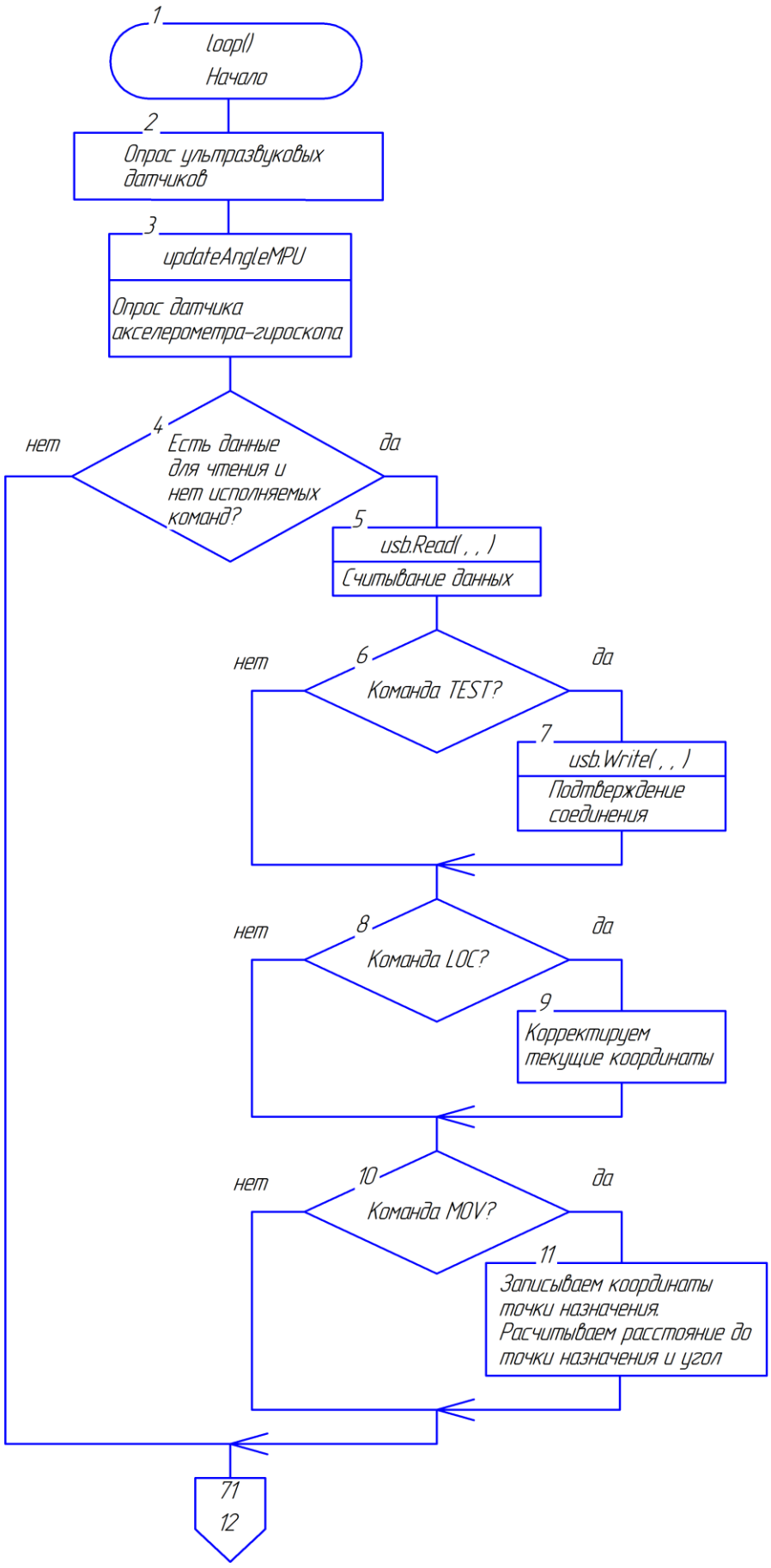



Рисунок 6.7.1 – Блок схема алгоритма работы платформы, фрагмент 1

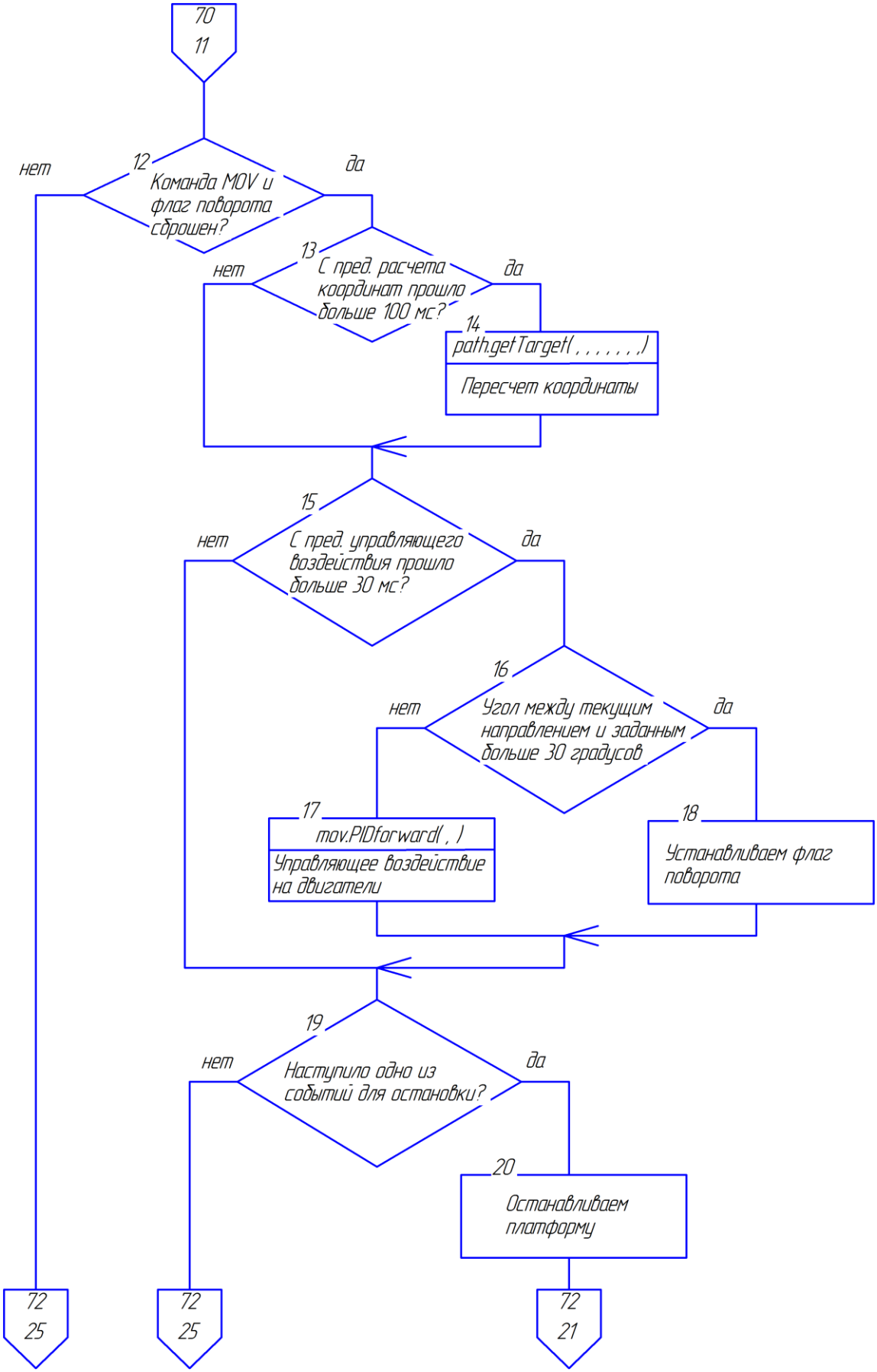



Рисунок 6.7.2 – Блок схема алгоритма работы платформы, фрагмент 2

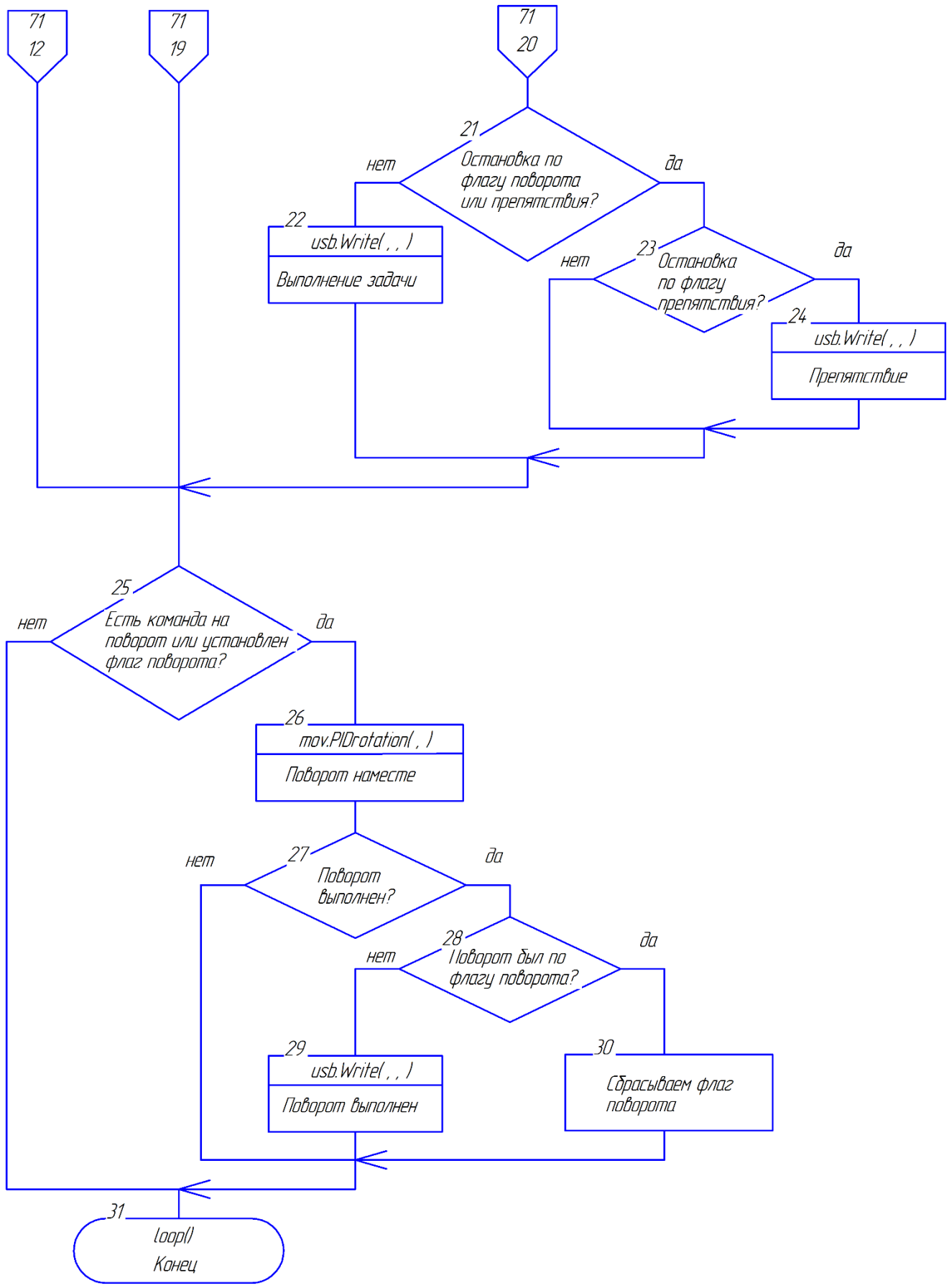



Рисунок 6.7.3 – Блок схема алгоритма работы платформы, фрагмент 3

# 7 Результаты экспериментальных испытаний

Программа испытаний состояла из следующих пунктов:

1. Проверка прямолинейного движения – необходима для отслеживания качества работы ПИД регулирования и гироскопа.
2. Проверка обнаружения препятствий – для определения работоспособности алгоритма обнаружения препятствий.
3. Проверка следования платформы по заданному маршруту с грузом.
4. Проверка работы алгоритмов визуальной ориентации – задачей являлось обнаружение метки в течение передвижения платформы.
5. Проверка работы звуковых оповещений.

Для испытания на прямолинейном участке использовалась поверхность, выложенная плиткой для выявления влияния проскальзываний и работы робота при передвижении по неровной поверхности. В результате испытаний платформа преодолела маршрут длинной в 40 метров с разворотом на 180 градусов в средней точке маршрута. Отклонения точки прибытия от точки назначения составили менее 20 см по обеим осям.

Пункты проверки работоспособности алгоритма обнаружения препятствий и следования платформы по маршруту были совмещены. Маршрут роботизированной платформы представлял из себя четыре точки описывающие неполный квадрат размером 2 на 2 метра. В конце маршрута находилась визуальная метка на листе формата А4 закрепленная на расстоянии 1,5 метра от конечной точки маршрута.

В ходе испытаний была выявлена некачественная работа ультразвуковых датчиков во время движения. Это выражалось в виде периодических пропусков динамических препятствий (устанавливаемых в различные моменты времени на пути следования). Данная проблема обусловлена тем, что, если положение, в котором датчик излучил волну, будет изменено во время следования звуковой волны до препятствия и обратно, то есть шанс того что датчик не зафиксирует вернувшуюся звуковую волну. Так же данный датчик не срабатывает на материалы

хорошо поглощающие различные колебания. К примеру, на одежду из шерсти, платки и прочие схожие материалы.

При следовании по маршруту платформа прошла через все заданные точки маршрута. Отклонение при прохождении каждой из точек не выходило за допустимый радиус равный 20 см. В конечной точке произошло обнаружение визуальной метки, что сопровождалось звуковым оповещением и выделением контура метки на дисплее платформы.

При ходовых испытаниях под нагрузкой выявлена, недостаточная мощность двигателей. Это выражалось остановкой при блокировании ведомых колес на глубоких стыках плитки при развороте на месте. При прямолинейном движении проблем с движением не возникло.

# 8 Безопасность и экологичность проекта

Корпус платформы изготавливается из фанеры и алюминиевых уголков. При обработке данных следует соблюдать все инструкции техники безопасности при ручной обработке материалов. Перед работой необходимо надеть удобную для работы одежду и защитные очки. Подготовить рабочее место и проверить наличие и исправность всех необходимых инструментов.

Во время работы необходимо использовать инструмент только по назначению. При нарезке и обработке материала необходимо соблюдать осторожность, при возникновении неисправностей немедленно прекратить использование инструмента, если инструмент электрический – обесточить. Во время выполнения технологических операций таких как резка, сверление, шлифовка, сборка и т.д. необходимо надежно закреплять детали используя различные упоры, тиски струбцины, подкладки и другие, предназначенные для той или иной операции приспособления.

При сборке электрических соединений необходимо обеспечить надежную изоляцию всех токоведущих частей – наличие оголенных проводников недопустимо. Для изоляции могут быть использованы различные материалы такие как электроизоляционная лента или термоусадочная трубка. Если при сборке необходима пайка элементов необходимо соблюдать все меры безопасности при работе с электронагревательными приборами. Помещение, в котором осуществляется пайка должно быть хорошо проветриваемо и иметь хорошее освещение.

При эксплуатации платформы необходимо соблюдать следующие условия:

1. Так как в платформе используются литиевые аккумуляторы, то необходимо соблюдать все меры безопасности при работе с такими устройствами. Заряд аккумуляторов осуществлять специализированными зарядными устройствами. Внимательно выставлять количество заряжаемых банок на зарядном устройстве для конкретного аккумулятора – недопустимо напряжение более 4,2

Вольта на банку. Также недопустим перегрев аккумулятора выше 70 градусов. Необходимо следить за герметичностью корпуса аккумулятора. Несоблюдение перечисленных требований может привести к самовоспламенению аккумуляторной батареи. Хранить аккумуляторы необходимо при заряженном на 50 – 70 % состоянии при температурах не выше 20 градусов Цельсия. При эксплуатации для продления срока службы аккумуляторов рекомендуется недозаряжать их на 5 – 10 % и недоразряжать на 15 – 20 %.

2. Перед эксплуатацией проверять надежность всех соединений. При обнаружении неподключенных проводников, произвести подключение в соответствии с документацией и зафиксировать соединение. При обнаружении не изолированных или поврежденных проводников, произвести их замену и изоляцию.

3. В целях безопасности включение питания силовой части производить только после полного запуска программного обеспечения.

4. Во время движения платформы в рабочей области в целях безопасности недопустимо присутствие людей, а также препятствий рассеивающих и поглощающих ультразвуковые волны.

5. При отклонениях в процессе работы платформы питание силовой части отключается в первую очередь при помощи перекидного выключателя.

6. При замене каких-либо элементов конструкции, необходимо сначала полностью обесточить платформу, затем снять аккумуляторы и поместить в безопасное место, и после этого производить необходимые действия.

Отказ от двигателей внутреннего сгорания и применение электрохимических накопителей энергии обеспечивает более высокий уровень экологичности. По истечении срока годности аккумуляторов их утилизируют, и они не причиняют вреда окружающей среде.

# 9 Экономическая эффективность

В список затрат на создание платформы будут входить только технически законченные элементы, не требующие дополнительной механической обработки или другого вида физических трудозатрат. Экономический расчет представлен на рисунке 9.1.

| № | Наименование | Кол | Цена, р |
|---|---|---|---|
| 1 | Raspberry Pi 3 Model B | 1 | 3800 |
| 2 | HC-SR04 | 3 | 1020 |
| 3 | L298n | 1 | 350 |
| 4 | LM2596S | 1 | 360 |
| 5 | KIS-3R33S | 1 | 200 |
| 6 | GM25-370 | 2 | 2000 |
| 7 | MPU6050 | 1 | 240 |
| 8 | Arduino MEGA 2560 | 1 | 3700 |
| 9 | A4Tech G7 X100 | 1 | 760 |
| 10 | A4Tech PK-760E | 1 | 810 |
| 11 | GX30H55761 | 1 | 900 |
| 12 | RITMIX SP-210 | 1 | 1000 |
| 13 | 4.3 inch Display Waveshare Raspberry Pi Display Monitor 480x272 HDMI - LCD | 1 | 3000 |
| 14 | DS1058-26-1 | 1 | 90 |
| 15 | BW3303 | 1 | 210 |
| 16 | ПВС 2х0,75-10 | 1 | 170 |
| 17 | BW1411 | 1 | 230 |
| 18 | BW1431 | 2 | 560 |
| 19 | Jaker 2,5 | 2 | 80 |
| 20 | Винт М3 | 22 | 16,5 |
| 21 | Винт М4 | 42 | 68,8 |
| 22 | Гайка М3 | 76 | 57 |
| 23 | Гайка М4 | 88 | 81,3 |
| 24 | Гайка М6 | 12 | 18 |
| 25 | Шайба М3 | 76 | 30,4 |
| 26 | Шайба М4 | 88 | 44 |
| 27 | Шайба М6 | 12 | 12 |
| 28 | Стойка крепежная М3 | 16 | 256 |
| 29 | Rocker Switch | 2 | 90 |
| 30 | Аккумуляторы Team Orion Batteries 11.1V 5300mAh 50C LiPo Softcase | 2 | 12980 |
| | | Итого: | 33134 р |

Рисунок 9.1 – Экономический расчет

# Заключение

В рамках бакалаврской работы была проведена работа по исследованию существующих робототехнических систем и областях их применения. Определено наиболее перспективное и доступное направление развития, рассмотрены уже существующие робототехнические системы и на основе описанной информации обоснована актуальность текущей разработки. Выделены наиболее доступные и широко применяющиеся в мире методы позиционирования, объяснена невозможность их применения для робототехнических систем складской логистики и предложено альтернативное решение. Разработана структурная схема платформы и подобраны основные элементы схемы. Для взаимодействия контроллеров разработан универсальный протокол приёмопередачи данных, описан алгоритм его работы и написана программа на языке C++. Разработана схема электрических соединений всех элементов платформы. Подробно описана работа с каждым функциональным элементом, обеспечивающим безопасное передвижение и точное позиционирование платформы. Разработан алгоритм работы платформы и его программная реализация. Представлена программа и описаны результаты экспериментальных испытаний. Оценена безопасность и экологичность проекта, а также представлены основные требования по безопасной эксплуатации робототехнической платформы. Произведена оценка экономической эффективности.

Список используемой литературы


1. Александров А.А. Электротехнические чертежи и схемы / Александров К.К., Кузьмина Е.Г.- М.:Энергоатомиздат, 1990. - 288с.
2. Invensense. [Электронный ресурс]: документация на микроэлектромеханический датчик акселерометра-гироскопа. – Электрон.текстовые дан. – режим доступа к документу: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
3. Численное интегрирование [Электронный ресурс]: электронная энциклопедия. – режим доступа: https://ru.wikipedia.org/wiki/Численное_интегрирование , свободный.
4. Энкодеры [Электронный ресурс]: электронная энциклопедия. – режим доступа: https://ru.wikipedia.org/wiki/Датчик_угла_поворота, свободный.
5. Программирование на языке С для AVR и PIC микроконтроллеров./ Сост. Ю.А. Шпак – К.: «МК-Пресс», 2006. – 400 с., ил. ISBN 966-8806-16-6
6. Колос М.В., Колос И.В. Методы оптимальной линейной фильтрации / Под ред. В.А. Морозова. – М.: Изд-во МГУ, 2000. – 102с
7. Документация на драйвер L298n [Электронный ресурс]: документация. – режим доступа: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf, свободный.
8. Одометрия [Электронный ресурс]: коллективный блог Robokraft. – режим доступа : http://robocraft.ru/blog/technology/736.html, свободный.
9. ПИД-регулятор [Электронный ресурс]: электронная энциклопедия. – режим доступа: https://ru.wikipedia.org/wiki/ПИД_регулятор, свободный.
10. Денисенко В. ПИД-регуляторы [Электронный ресурс]: Энциклопедия АСУ ТП. – режим доступа: http://www.bookasutp.ru/Chapter5_5.aspx, свободный
11. Julio E. Normey-Rico, Ismael Alcalá, Juan Gómez-Ortega, Eduardo F. Camacho, Mobile robot path tracking using a robust PID controller. // Control Engineering Practice, volume 9, issue 11, november 2001, Pages 1209-1214, 9429 symbols.

URL: http://www.sciencedirect.com/science/article/pii/S0967066101000661

12. M. Abo-Zahhad, S. Ahmed and M. Mourad, Hybrid Uplink-Time Difference of Arrival and Assisted-GPS Positioning Technique. // International Journal of Communications, Network and System Sciences, Vol. 5 No. 6, 2012, pp. 303-312, 13212 symbols. URL: http://www.scirp.org/journal/PaperInformation.aspx?PaperID=19652

13. Hongbin Wang, Jian Dong, and Yueling Wang, Discrete PID-Type Iterative Learning Control for Mobile Robot. // Journal of Control Science and Engineering, 2016, Article ID 2320746, 7 pages, 10810 symbols

URL: https://www.hindawi.com/journals/jcse/2016/2320746/

14. Nunzio Abbate, Adriano Basile, Carmen Brigante, Alessandro Faulisi, and Fabrizio La Rosa, Modern Breakthrough Technologies Enable New Applications Based on IMU Systems. // Journal of Sensors, volume 2011 (2011), Article ID 707498, 7 page, 16449 symbols.

URL: https://www.hindawi.com/journals/js/2011/707498/

15. Bing-Fei Wu, Wang-Chuan Lu, and Cheng-Lung Jen Monocular Vision-Based Robot Localization and Target Tracking. // Journal of Robotics, volume 2011 (2011), Article ID 548042, 12 pages, 35324 symbols.

URL: https://www.hindawi.com/journals/jr/2011/548042/

16. Шилдт, Герберт. Полный справочник по C++, 4-е издание. : Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 800 с. : ил. – Парал. тит. Англ. ISBN 5-8459-0489-7 (рус.)

17. Блум Джереми. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. – СПб.: БВХ-Петербург, 2015. – 336 с.: ил. ISBN 978-5-9775-3585-4

18. Усатенко С.Т., Каченюк Т.К., Терехова М.В. Выполнение электрических схем по ЕСКД: Справочник. – М.: Издательство стандартов, 1989. – 325 с.

19. Atmel. [Электронный ресурс]: документация. – режим доступа: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf, свободный.

20. Документация на ультразвуковой датчик расстояния HC-SR04 [Электронный ресурс]: документация. – режим доступа: http://www.micropik.com/PDF/HCSR04.pdf, свободный.

21. Богатырев В.В., Глибин Е.С., Гущин С.В. В сборнике: Наука. Технология. Производство - 2016: Современные методы и средства диагностики электроэнергетического и электротехнического оборудования, средств и систем автоматики материалы Всероссийской научно-технической конференции, посвященной 60-летию филиала УГНТУ в г. Салавате. Уфимский государственный нефтяной технический университет, филиал в г. Салавате. 2016. С. 42-46.