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

Институт машиностроения
(наименование института полностью)
Кафедра «Промышленная электроника»
(наименование)
11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)
Электроника и робототехника
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Аппаратно-программный комплекс мини-полигона. Система беспилотного движения автомобиля.

Студент М.В. Клеймёнов
(И.О. Фамилия) (личная подпись)
Руководитель А.К. Кудинов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Объем бакалаврской работы 93 стр., 49 рисунков, 11 таблиц, 20 источников, 2 приложений.

В данной бакалаврской работе была разработана система беспилотного движения автомобиля в рамках аппаратно-программного комплекса мини-полигона, которая управлялась автоматически при помощи устройства лидара.

Цель работы: разработка системы восприятия окружающего пространства беспилотным автомобилем, оснащенным лидаром.

Задачи работы:

1. Анализ исходных данных и известных решений

2. Выбор необходимых комплектующих;

3. Разработка структурной схема и схемы электрической соединений;

4. Моделирование и изготовление конструкции каркаса;

5. Разработка управляющей программы и программы визуализации;

6. Сборка и отладка системы при помощи экспериментальных исследований.

Работа состоит из пяти глав, в которых решены упомянутые задачи.

Для оформления чертежей воспользовался программным пакетом КОМПАС-3D V18.

Для разработки твердотельной модели конструкции использовался универсальный программный пакет Fusion 360.

Для разработки управляющей программы использовалась среда разработки Arduino IDE.

Для разработки программы визуализации использовалась среда разработки Processing.

Область применения данного комплекса, влачащегося в себя систему беспилотного управления может быть использована в любом научно-учебном

центре по развитию навыков программирования и машиностроения в сфере беспилотных транспортных средств.

Abstract

The title of the graduation work is Hardware and software complex of the testing area. Unmanned vehicle movement system.

The senior thesis consists of an explanatory note on 93 pages, introduction, including 49 figures, 11 tables, the list of 20 references including 7 foreign sources and 2 appendices.

The aim of the work is to develop a system for the perception of the surrounding space by an unmanned vehicle equipped with a LiDAR.

The graduation work may be divided into several logically connected parts which are the state of the art; hardware and software development of the device; frame design; experimental research and device assembly.

Much attention is given to the relevance of the problem concerning education of specialists in the field of unmanned vehicles.

The special part of the project gives details about the development of the control program algorithm and the writing of program code for the full operation of the system.

We also present the results of the experiments carried out to study the methods scanning the environment with LiDAR. The graphic mapping of LiDAR operation principle is displayed in the designed visual program using the Processing development environment.

In conclusion we'd like to stress the designed system can be widely used in training centers for the development of mobile robotics skills.

Содержание

Введение

Издавна человечество стремилось преодолевать большее расстояние за малый промежуток времени, затрачивая при этом минимальное количество сил. Со временем стал появляется транспорт, который облегчил обыденные потребности в передвижении, но на этом люди не остановились и пытались всячески улучшать его. Так на замену обычному колесу пришёл велосипед, а в последствии и более продвинутые виды транспортного средства.

«Современные автомобили сейчас выполняют функцию перевозки грузов или пассажиров. В России около 30 процентов населения добирается до работы на личном транспортном средстве, при этом затрачивая в среднем чуть более получаса, и это только в одну сторону. А что стоит говорить о людях, чья профессия основана на вождении - процесс который требует постоянного внимания и уйму затраченного времени.» [5]

В наше время учёные и инженеры пытаются придумать новые виды передвижения или хотя бы упростить уже имеющиеся. Одним из таких улучшений является система беспилотного движения автомобиля. По версии международного общества автомобильных инженеров (Society of Automotive Engineers) данную систему можно разделить на 6 уровней автономности и расписать следующим образом:

– уровень 0. Никаких систем помощи водителю, обычный бюджетный автомобиль. Водитель всегда должен контролировать ситуацию;

– уровень 1. Автомобиль с простой системой помощи водителю. Например, система экстренного торможения. Водитель всегда должен контролировать ситуацию;

– уровень 2. Автомобиль может одновременно управлять поддержанием скорости, торможением и процессом руления, однако водитель всегда должен быть готов принять управление;

– уровень 3. Автомобиль работает на автопилоте. Позволяет зрительно отвлекаться водителю, но может попросить принять управление на себя;

– уровень 4. Очень схож с уровнем 3, только участие водителя не требуется. В случае конфликтной ситуации, если водитель не отреагирует на просьбу автомобиля принять управление, то он припаркуется самостоятельно;

– уровень 5. Полная автоматизация процесса вождения, автомобиль решает любую задачу без участия водителя. В машине может отсутствовать руль.

Более краткое и ясное содержание можно увидеть на рисунке 1.

Рисунок 1 – Уровни автономности беспилотной системы движения

К сожалению автомобили с полной автономностью существуют только на стадии глубокой разработки. В то время как транспортные средства уровня 4 уже начинают появляться в более развитых странах.

1. Состояние вопроса

1.1. Формулировка актуальности, цели и задач проекта

«Прежде всего, система беспилотного движения – это та структура, которая позволяет транспортному средству безопасно ездить по общественным дорогам без участия человека. Такие термины, как «автопилот», «беспилотный» и «самоуправляемый автомобиль» описывают одну и ту же технологию: датчики и электронный блок управления, который может управлять транспортным средством вместо человека, по крайней мере, при некоторых условиях.» [19]

Развитие сферы беспилотных транспортных средств в России осуществляется не так быстро, как за рубежом. Поспособствовать этому может аппаратно-программный комплекс мини-полигона, задачей которого и является повышение квалификации в сферах логистики, диспетчеризации и электротехники. Комплекс включает в себя модель мобильного робота, макет городской среды и наличие программного обеспечения.

Поэтому актуальность дипломной работы заключается в развитии инженерных и новаторских навыков в сфере беспилотных транспортных средств, посредством аппаратно-программного комплекса мини полигона.

Целью выпускной квалификационной работы является разработка системы восприятия окружающего пространства беспилотным автомобилем, оснащенным лидаром.

Для достижения поставленной цели следует выполнить следующий перечень задач:

1. Поиск известных решений в литературе и на просторах интернета;
2. Разработка структурной схемы;
3. Поиск подходящих элементов для реализации;

4. Разработка схемы электрической соединений;
5. Разработка конструкции каркаса мобильной платформы;
6. Разработка алгоритма управляющей программы и написание кода;
7. Разработка программы визуализации для отладки датчика;
8. Испытания и отладка всей системы беспилотного движения;
9. Сборка мобильного робота.

1.2. Обзор существующих решений

Самая ближайшая похожая отрасль с применением систем по беспилотному передвижению — это реальные системы транспортных средства, на приобретение которых не каждый человек имеет достаточное количество финансов и возможностей.

«Усовершенствованные системы помощи водителю (ADAS) и автономные системы без водителя полагаются на беспилотные опции, чтобы помочь блоку управления транспортного средства понять, где находится транспортное средство относительно его окружения и принять решение в экстренной ситуации.» [10]

Одной из таких опций является ассистент удержания полосы движения, главной целью которой является создание корректирующего рулевого момента. Для этого в верхней области лобового стекла автомобиля устанавливают датчики полос дорожной разметки, которые отправляют сигнал при отклонении траектории вдоль полосы движения в электронный блок управления, а он в свою очередь использует электродвигатель электрического усилителя рулевого управления с возможностью корректировки. Наглядно принцип работы можно увидеть на рисунке 2.

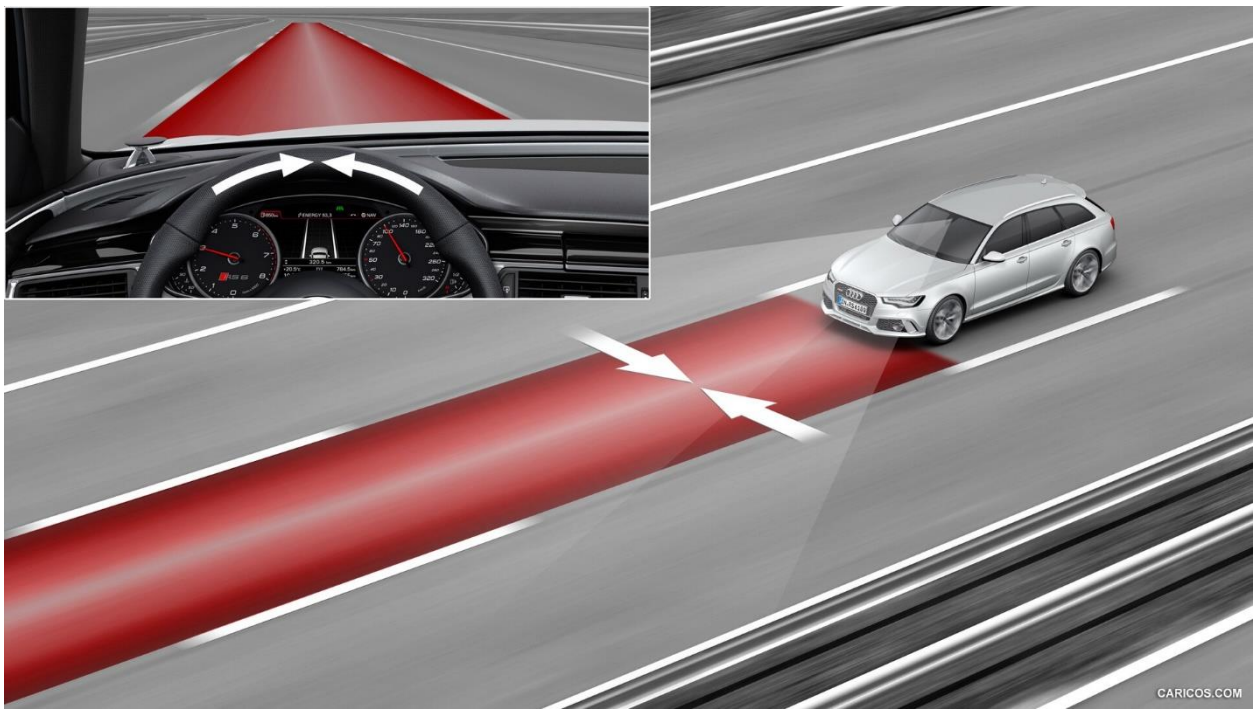


Рисунок 2 – Визуализация принципа работы ассистента удержания полосы

Ещё одной опцией является круиз-контроль, ключевой ролью которого является поддержание постоянной скорости автомобиля в определённый момент. Сама система круиз-контроля состоит из нескольких статических датчиков расстояния, установленных на фронтальной части автомобиля, а также модулем дроссельной заслонки двигателя и модулем управления тормозами. Принцип действия опции круиз-контроля продемонстрирован на рисунке 3 и подразумевает, что если автомобиль водителя имеет слишком маленькое расстояние следования, данная система посылает сигнал двигателю или тормозам, замедляя автомобиль. Как только путь будет свободен, система разгонит автомобиль до скорости, выбранной водителем. Ограничивая поток воздуха к двигателю, когда скорость автомобиля приближается к заданной скорости, и увеличивая поток воздуха, когда скорость автомобиля ниже заданной, система круиз-контроля помогает автомобилю поддерживать почти постоянную скорость.» [15] Но определять расстояние лишь в одном направлении не позволит составить целостной

картины вокруг автомобиля, и именно поэтому были разработаны технологии радар и лидар.

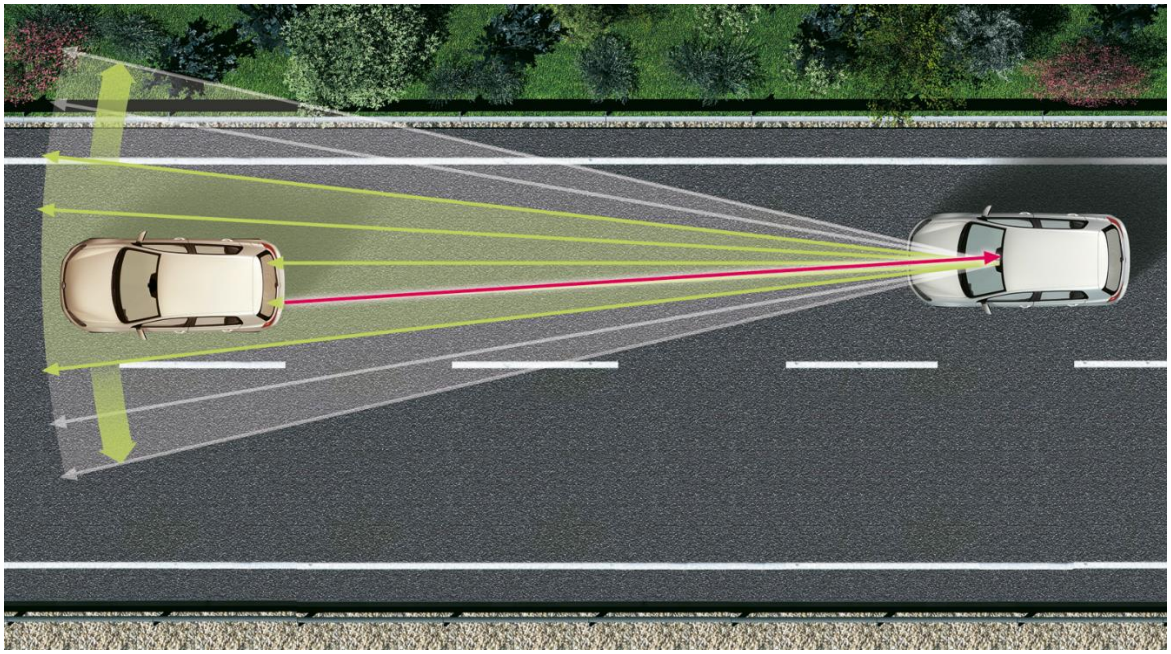


Рисунок 3 – Визуализация принципа действия круиз-контроля

«Радары излучают электромагнитные волны аналогично беспроводным компьютерным сетям и мобильным телефонам. Сигналы отправляются в виде коротких импульсов, которые могут отражаться объектами на их пути, частично отражаясь обратно на радар. Когда эти импульсы перехватывают осадки, часть энергии рассеивается обратно на радар. Эта концепция похожа на эхо.» [16] Главным отличием лидара является использование инфракрасного излучения взамен электромагнитных волн. Также стоит отметить что длина волны и точность измерения расстояния радара зависит от длины антенны, в то время как лазер с рассеивающей линзой имеет микрометрический диапазон длины волны и не зависит от другого устройства, что даёт ей большее преимущество в использовании сферы беспилотных автомобилей. Для работы лидара на 360 градусов его поставили на вращающуюся платформу, которая совершает довольно большое количество оборотов в минуту, как в принципе можно рассмотреть на

рисунке 4. Инженеры в отрасли автомобилестроения считают, что работу лидара можно дополнить камерами компьютерного зрения.

Рисунок 4 – Технология лидар

Камеры компьютерного зрения представляют из себя систему по распознаванию дорожных знаков, включающую в себя видеокамеру, блок управления и дисплей. Принцип работы заключается в отслеживании дороги перед автомобилем и выявлении дорожных знаков по пути. После выявления знака, информация в виде фотографии отправляется на блок управления, где методом сравнивания фотографии с базой дорожных знаков определяется какой именно это знак. Затем при совпадении полученной информации и информации, хранящийся в базе, знак выводится на дисплей и уведомляет водителя. Как и было сказано ранее, лидар и система машинного зрения являются взаимодополняемыми опциями, так как в ситуации дождливой или туманной погоды камера может не распознать знак в отличии от инфракрасного излучения. Хочется добавить, что современные технологии дошли и до распознавания велосипедистов, пешеходов и перегоняемого скота. Принцип работы системы машинного зрения изображён на рисунке 5.

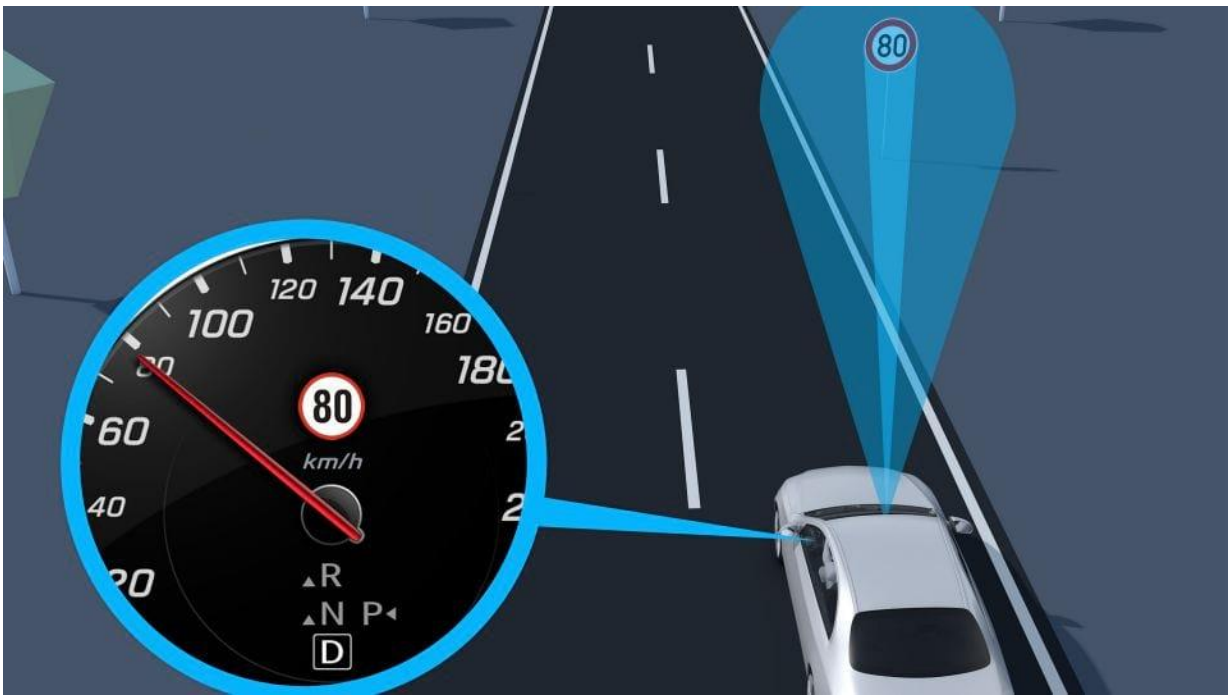


Рисунок 5 – Визуализация принципа работы системы машинного зрения

Выводы по разделу

Разработка автомобиля с системой беспилотного движения осуществляется для реализации полноценного аппаратно-программного комплекса мини-полигона с расчётом на поднятие навыков в сфере беспилотной транспортной техники. То есть основой для реализации автомобиля с данной системой будет мобильная робототехническая платформа.

Так как количество опций в транспортном средстве по мере увеличения возрастает вместе с общей стоимостью системы, то принято решение сократить количество опций до одной для реализации этой системы в рамках выпускной квалификационной работы с дальнейшим расчётом цены.

2. Аппаратная часть

2.1. Разработка структурной схемы

Для разработки аппаратной части мобильного робота необходимо составить структурную схему, на основе которой в дальнейшем будет собираться само устройство.

Во избежание столкновений на мобильную платформу устанавливается датчик, позволяющий определить расстояние до объекта. Но для ориентации в пространстве собирать информацию только в одном направлении недостаточно. Поэтому грамотным решением было принято установить датчик на сервопривод, суть которого крутить этот датчик вокруг своей оси и опрашивать состояние датчика раз в несколько миллисекунд, который в последствии будет передавать его на микроконтроллер. В зависимости от принятого состояния с датчика, в микроконтроллере формируется управляющий сигнал, подаваемый на драйвер двигателей, который, в свою очередь, распределяет нагрузку в требуемых количествах на моторчики с колёсами для движения робота в определённом направлении. Приёмник предназначен для получения сообщения с информацией для выбора маршрута в случае его развязки на макете. Датчик угла поворота, установленный на двигателях, позволит точно понять какой именно сигнал стоит подавать на колёса для корректировки движения беспилотного транспортного средства.

В разработанной структурной схеме представлены все требуемые элементы для реализации беспилотного мобильного робота:

– микроконтроллер;

– драйвер двигателей;

– электродвигатели с колёсами;

– датчики угла поворота;

– сервопривод;

– датчик расстояния;

– приёмник информации;

– электропитание.

На рисунке 6 представлена структурная схема беспилотного мобильного робота.

Рисунок 6 – Структурная схема беспилотного мобильного робота

2.2. Выбор необходимых комплектующих

Для первоначального выбора комплектующих требуется определится с электропитание. Так как робот представляет из себя мобильную платформу, то вся энергетическая часть должна быть так же мобильной. Лучшим выбором был батарейный отсек для литий-ионных аккумуляторов типоразмера 18650. Литий-ионные аккумуляторы считаются наиболее энергоэффективными из-за простоты хранения и своей многоразовостью, а также возможностью быстрой подзарядки в домашних условиях по сравнению со другими аналогами аккумуляторов. Число 18650 представляет из себя не просто взятое значение, а расшифровку из двух размерных

параметров. Первое отображает диаметр аккумулятора в 18 миллиметров, второй длину в 650 миллиметров. Данный типоразмер используется в большинстве электротехнических устройств, таких как: ноутбуки, телефоны, фотоаппараты, портативных зарядках и многих других. Также данные аккумуляторы имеют разные номиналы ёмкости, что добавляет ещё одно преимущество.

Одним из недостатков такого типа аккумулятор является востребованность постоянного частичного заряда, то есть если заряд аккумулятора упадёт в абсолютный ноль, то снова зарядить данный аккумулятор будет крайне затруднительно или даже невозможно. Компенсировать этот недостаток позволяют электронные платы по контролю заряда BMS (Battery Management System).

Ещё одним элементом электропитания являются повышающие или понижающие преобразователи напряжения, позволяющие выдавать на выходе требуемое по значению напряжение, не зависимо от входного значения. Единственным требованием является какой именно нужен преобразователь. Понижающие позволяют изменить высокое входное напряжение в наиболее низкое выходное, в то время как повышающие, наоборот, изменить низкое входное значение в наиболее высокое. Повышающие преобразователи обычно менее эффективны, чем понижающие преобразователи, но ненамного. Основная причина связана с тем, что ток индуктора течет непосредственно на землю во время работы, а не через нагрузку, как это происходит в понижающих преобразователях. Хоть разница в эффективности незначительна, существуют ещё параметры, по которым повышающие конверторы уступают понижающим, а именно: потери при низких напряжениях из-за более высокого тока и параллельное подключение аккумуляторов имеет тенденцию заряжать друг друга.

Также неотъемлемой частью мобильной платформы является двухпозиционная кнопка с фиксацией, выступающая в роле ключа электропитания. С её помощью является возможным ограничить

токопотребление элементов цепи в состоянии бездействия простым нажатием.

2.2.1. Выбор элементов энергопитания

Индивидуальным решением было принято питать цепь от 5 вольт и воспользоваться понижающим стабилизатором постоянного напряжения. Так как аккумулятор типоразмера 18650 выдаёт 3.7 вольт, то для превышения значения в 5 вольт потребуется как минимум два таких аккумулятора.

В качестве защитной платы для аккумуляторов выбор был сделан в сторону платы FDC-2S-2, характеристики которой сведены в таблицу 1. Защитная плата FDC-2S-2 показана на рисунке 7.

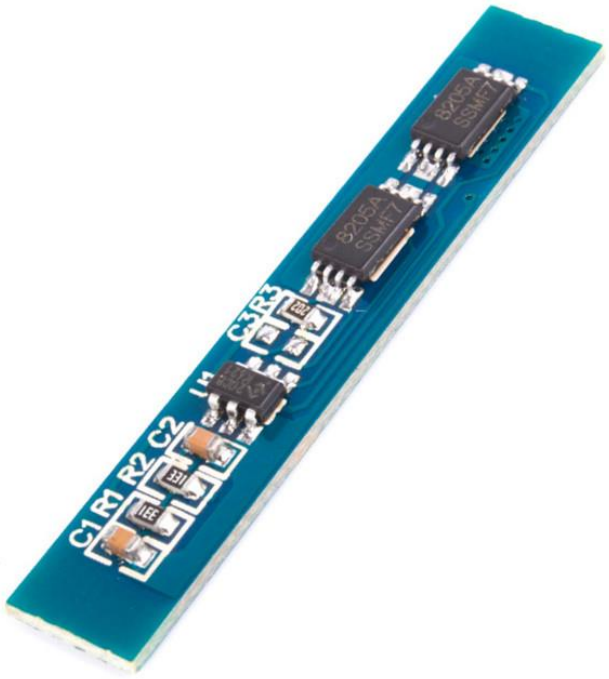


Рисунок 7 – Защитная плата FDC-2S-2

Таблица 1 – Характеристики платы FDC-2S-2.

Конфигурация	Li-Ion, 2S
Чип	20CB
Ток заряда, А	3
Ток разряда, А	3
Защита по току, А	5
Номинальное напряжение, В	7,4
Максимальное напряжение на элемент при зарядке, В	4,20 ± 0,05
Защита от короткого замыкания	Присутствует
Цена	100 рублей

Понижающий импульсный регулируемый стабилизатор напряжения должен преобразовывать напряжение из 7.4 вольт в 5 вольт, а также иметь высокий коэффициент полезного действия, свыше 70%. Ещё одним ключевым параметром должен являться выходной ток до 3 ампер без дополнительного охлаждения. Основываясь на этих требованиях, подходящим элементом является плата на базе чипа LM2596, изображённая на рисунке 8. Цена такой платы достигает 35 рублей, что даёт ему ещё одно преимущество.

Рисунок 8 – Понижающий преобразователь напряжения LM2596

2.2.2. Выбор сервопривода

Одним из требований для выбора сервопривода является напряжение питания. Так как принятым решением было запитывать цепь от напряжения в 5 вольт, то выбор существенно сократился.

Следующем требованием по выбору сервопривода был возможный угол поворота, потому что требовалось собирать информацию с 3 сторон мобильного робота, а именно: слева, спереди и справа. Следовательно, угол поворота должен быть не менее 180 градусов.

Из-за того, что робот является мобильным и должен передвигаться с постоянной скоростью, то ещё одним требованием оказалась скорость вращения сервопривода, чтобы на каждый промежуток расстояния привод смог сканировать диапазон 180 градусов хотя бы за единицу времени в секундах.

Проанализировав российский рынок по поиску сервоприводов можно было выделить два наиболее подходящих варианта. Первым являлся

сервопривод JX Servo PDI-6221MG-180 изображённый на рисунке 9. Основные параметры этого двигателя сведены в таблице 2.

Рисунок 9 – Сервопривод PDI-6221MG-180

Таблица 2 – Основные параметры сервопривода PDI-6221MG-180

Диапазон вращения	172 ± 2°
Напряжение питания	DC 4.8 - 6.6 В
Крутящий момент (4.8 Вольт)	17,25 кг ∗ см
Крутящий момент (6 Вольт)	20,32 кг ∗ см
Рабочая скорость (4.8 Вольт)	0,18 с / 60°
Рабочая скорость (6 Вольт)	0,16 с / 60°
Размер	40,5 x 20,2 x 38 мм
Вес	62 г
Цена	1920 рублей

Вторым наиболее подходящим вариантом с похожими характеристиками являлся сервопривод MTR-SERVO-FS5106B, изображённый на рисунке 10, основные параметры которого сведены в таблице 3.

Рисунок 10 – Сервопривод FS5106B

Таблица 3 – Основные параметры сервопривода FS5106B

Диапазон вращения	180°
Напряжение питания	DC 4.8 - 6.6 В
Крутящий момент (4.8 Вольт)	5 кг ∗ см
Крутящий момент (6 Вольт)	6 кг ∗ см
Рабочая скорость (4.8 Вольт)	0,18 с / 60°
Рабочая скорость (6 Вольт)	0,16 с / 60°
Размер	40,8 x 20,1 x 38 мм
Вес	51 г
Цена	900 рублей

Крутящий момент был не столь важен, так как на сервопривод крепился лишь датчик расстояния. Выбор пал именно на второй вариант из-за более низкой цены и доступности в ближайших областных магазинах.

Как правило, сервоприводы имеют уже встроенный драйвер в корпусе для управления устройством по градусам, при помощи сигнала широтно-импульсной модуляции (ШИМ).

2.2.3. Выбор датчика расстояния

Принцип работы всех датчиков расстояния одинаков. При излучении передатчик формирует сигнал в одном направлении. В последствии столкновения сигнала и объекта, одна часть сигнала поглощается, а другая отражается в приёмник, после чего датчик фиксирует сигнал. Определить расстояние S позволяет формула 1.

$$S = \frac{\upsilon * t}{2}, \quad (1)$$

где υ – скорость излучаемого сигнала, м/с;
t – время между испусканием и принятием сигнала, с.

Так как излучаемый сигнал проходит расстояние дважды, то произведение скорости и времени делится на два. В мире современных технологий существует большое количество датчиков расстояния с разным видом излучений.

Работа одного такого датчика основана на двойном преобразовании, электрического сигнала в импульсные ультразвуковые излучения и наоборот. На рисунке 11 вы можете увидеть ультразвуковой датчик расстояния HC-SR04.

Рисунок 11 – Ультразвуковой датчик расстояния HC-SR04

Датчик HC-SR04 имеет четыре вывода, два из которых отвечают за питание, и оставшиеся два непосредственно подключаются к цифровым пинам микроконтроллера. Все ультразвуковые датчики имеют огромный недостаток из-за непостоянности значения скорости звука, которая в свою очередь зависит от температуры внешней среды. Компенсировать этот недостаток позволит ещё один датчик, который будет снимать показания температуры и корректировать переменную скорости звука. Применение этих датчиков нашлось в отрасли кораблестроения в качестве сонара. Это обуславливается тем, что скорость звука в воде больше чем на суше. Как можно увидеть из сравнительной таблицы 4, главным преимуществом датчика HC-SR04 является его цена и доступность, но для поставленной цели он не подходит, потому что данный датчик обладает большей погрешностью и малым временем отклика по сравнению с другими датчиками, хоть и подходит по параметрам напряжения и тока.

Более эффективными датчиками расстояния считаются инфракрасные из-за лучшего время отклика. Так как скорость света быстрее скорости звука. Инфракрасные датчики ещё называют оптическими, потому что их диапазон измерения расстояния напрямую зависит от фокусирующих линз. От сюда выявляется и недостаток таких датчиков — это появление помех из-за прозрачных объектов. К тому же недостатку можно добавить и наличие

помех от других источников освещения. Показанный на рисунке 12 датчик расстояния Sharp 2y0a02 относится к инфракрасным.

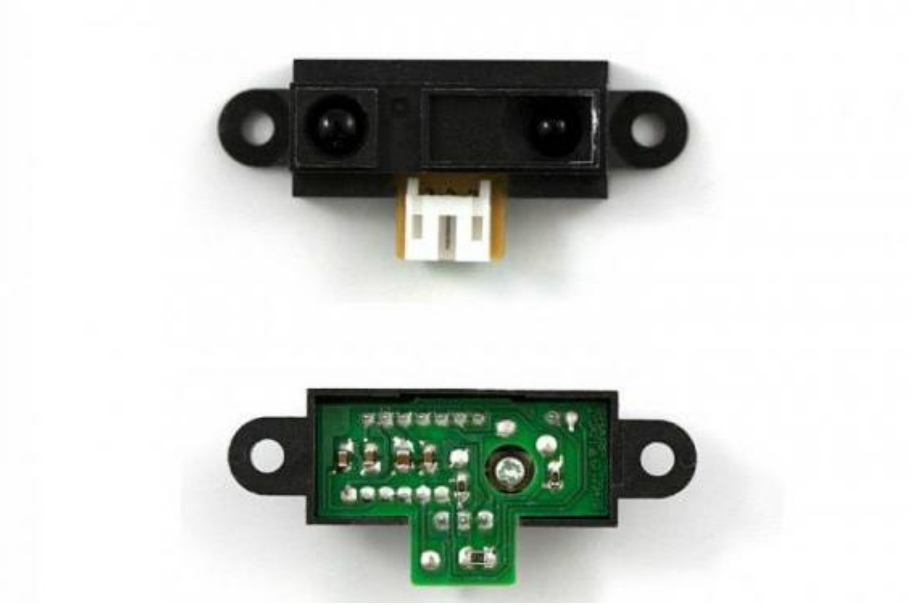


Рисунок 12 – Инфракрасный датчик расстояния Sharp 2y0a02

Данный датчик является аналоговым, что означает востребованность в аналоговом пине. Он нашёл широкое распространение в сфере робототехнике из своего рабочего напряжения и доступности в отечественных магазинах. Датчик состоит из инфракрасного светодиода и фоторезистора, которые объединены в пластиковом корпусе и закрыты тёмными линзами для предотвращения попадания света другого спектра, что сказывается на его цене (см. таблицу 4).

Ещё одним подходящим инфракрасным датчиком является GY-530 на базе чипа VL53L0X, который изображён на рисунке 13.

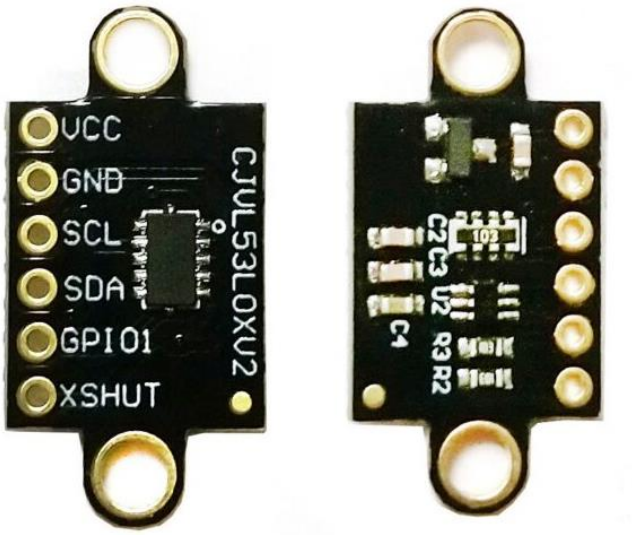


Рисунок 13 – Инфракрасный датчик расстояния GY-530 на базе VL53L0X

Работы чипа VL53L0X имеет интерфейс I^2C, поэтому для подключения требуется два аналоговых пина вычислительной микросхемы подключенных к шине I^2C.

Датчик имеет 6 контактов:

– VCC и GND (питание модуля);

– SDA (линия данных);

– SCL (линия тактирования);

– GPIO1 (программируемый выход прерывания);

– XSHUT (активный-низкий вход отключения с открытым стоком).

Данный датчик потребляет меньше всего тока, а также является наиболее точным и быстрым по сравнению с остальными датчиками расстояния, рассмотренными в таблице 4. В добавок обладает набором с парой режимов работы: режим высокой точности (РВТ) и режим высокой скорости (РВС). На просторах интернет-магазинов приобрести данный датчик можно в районе 450 рублей.

Таблица 4 – Сравнение характеристик датчиков расстояния.

Наименование	HC-SR04	Sharp 2y0a02	GY-530
Рабочее напряжение, В	5	4,5 – 5,5	2,8 – 5
Потребляемый ток, мА	15	33	10
Диапазон измерения расстояния, мм	20 - 4000	20 - 1500	30 - 2000
Точность измерения, мм	10	1	1 ± 5% (PBC) 1 ± 3% (PBT)
Время отклика, мс	100	38 ± 10	20 (PBC) 200 (PBT)
Угол обзора, град	15	25	10
Вес, г	8	7	4
Цена, руб	70	300	450

Выбор пал на дальномер GY-530 из-за своей точности, диапазона измерения расстояния, а также своей узконаправленности. Также можно отметить что потребление тока у сервопривода будет меньшим из-за массы датчика. Все эти плюсы компенсируются стоимостью самого сенсора.

2.2.4. Выбор двигателей постоянного тока и датчиков угла поворота

Двигатели постоянно тока служат для создания крутящего момента, передаваемого на колёса мобильной платформы в целях реализации движения. Для воспроизведения поворота ровно на то количество градусов, которое востребовано для манёвра, двигателям требуется энкодеры. В просторечье это датчик угла поворота, позволяющий преобразовать этот самый угол в цифровое или аналоговое значение, а в последствии передачи его на микроконтроллер. В следствии из этого, энкодер даёт больше возможностей и позволяет управлять скоростью по количеству оборотов

вала, останавливаться по заданному расстоянию, останавливаться по заданному количеству оборотов вала и получить информацию по пройденному пути. Энкодеры по принципу работы можно разделить на магнитные, ёмкостные и оптические.

«Магнитные энкодеры в своей работе используют изменения магнитного поля. В зависимости от технологии, использованной в чувствительных элементах, в магнитных энкодерах также различают несколько разновидностей. Чаще всего выделяют магнитно-резистивные, на датчиках Холла, вихретоковые и индуктивные. В магнитно-резистивных датчиках чувствительный элемент построен с использованием магниторезистивного эффекта, который заключается в изменении сопротивления тонких плёнок специально подобранных материалов в магнитном поле. Магнитное поле создаётся постоянным магнитом, установленным на вращающейся части энкодера. Энкодеры, на датчиках Холла, используют непосредственное измерение магнитного поля при помощи датчиков, действие которых основано на эффекте Холла. Магнитное поле здесь также создаётся постоянным магнитом. Индуктивные датчики используют изменение взаимной индуктивности при взаимном перемещении нескольких обмоток, или изменение собственной индуктивности (коэффициента самоиндукции) в случае использования одной обмотки и мишени из магнитного материала. Вихретоковые датчики являются одной из разновидностей индуктивных датчиков, в которых изменение магнитного поля происходит благодаря вихревым токам, наводимым в проводящем материале мишени, расположенной на вращающейся части датчика. При этом сильные магнитные поля могут вносить значительные помехи в работу магнитных датчиков.» [12]

Ёмкостной тип энкодеров основан на принципе изменения уже электрических полей. Статичные пластины обкладок конденсатора постоянно перезаряжаются под воздействием изменения электрического поля, которое вызвано вращением диэлектрического диска, установленного

на валу. Как и в случае с ранее упомянутым типом энкодеров, возникновение электрического поля может вызвать критические помехи в работе устройства.

Последний тип энкодеров основан на преобразовании электричества в инфракрасное излучение, которое при прохождении через кодовый диск попадает на фоторезистивный элемент. Кодовый диск, так называют диск с прорезями, одевающимися на вал мотора. Одетый диск осуществляет вращение вместе с валом двигателя с одинаковой скоростью, в то время как лазер либо проходит через диск, либо не проходит. Данным методом устройство энкодера подсчитывает количество попаданий лазера на фоторезистор. Главным источником помех могут послужить различные источники света из внешней среды.

Из всех рассмотренных типов энкодеров, наиболее подходящими являются оптические, так как исключить помехи информационного сигнала проще.

Рассмотрим мотор-редуктор Micro Metal Gearmotor N20 с встроенным энкодером и платой управления, изображённый на рисунке 14.

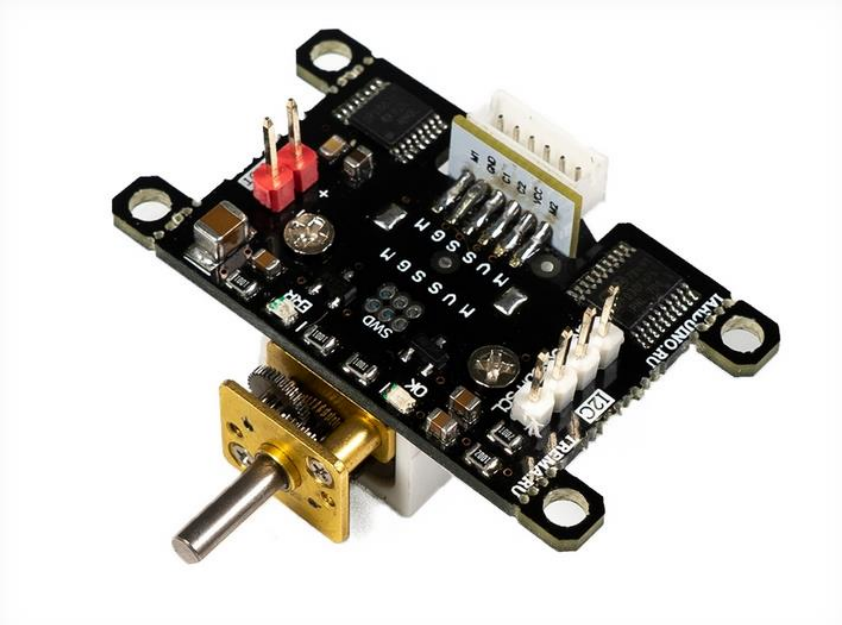


Рисунок 14 – Мотор-редуктор Micro Metal Gearmotor N20 с энкодером

Данный модуль зачастую используют для управления подвижными конструкциями. Для работы с таким типом моторов, на плате микроконтроллера обязательно присутствие шины I^2C, так как на плате модуля расположены два разъёма, один разъём из двух выводов и предназначен для питания мотора от 2,7 до 12 вольт, а другой из четырёх выводов для подключения к шине:

– SCL (вход/выход линии тактирования шины I^2C);

– SDA (вход/выход линии данных шины I^2C);

– 5V (вход питания энкодера от 5 вольт (номинально), или 3,3 вольт);

– GND (общий вывод питания).

Энкодеры является уже встраиваемым внутрь коллекторного двигателя, от чего цена такого двигателя существенно возрастает и составляет 1290 рублей.

Теперь рассмотрим точно тот же мотор-редуктор Micro Metal Gearmotor типоразмера N20, изображённого на рисунке 15, но уже без встраиваемого энкодера и с более развёрнутыми характеристиками в таблице 5.

Рисунок 15 – Мотор-редуктор Micro Metal Gearmotor N20 без энкодера

Таблица 5 – Характеристики электромотора Micro Metal Gearmotor N20.

Наименование	N20-100rpm-6V
Передаточное число	1: 298
Скорость без загрузки, об/мин	100
Скорость с нагрузкой, об/мин	20 - 4000
Номинальное напряжение, В	6
Напряжение питания, В	3-12
Максимальный ток, мА	1600
Ток без нагрузки, мА	55

Ценник такого же ревёрсно-коллекторного электромотора с металлическим редуктором без энкодера составляет 600 рублей.

Схожим аналогом ранее рассмотренных моторов является TT MotorOperating, изображённый на рисунке 16. Этот мотор имеет широкое применение в сфере мобильной робототехники из-за своих характеристик, сведённые в таблицу 6.

Рисунок 16 – Мотор-редуктор TT MotorOperating

Таблица 6 – Характеристики электромотора TT MotorOperating.

Наименование	TT MotorOperating
Передаточное число	1: 48
Скорость без загрузки, об/мин	180
Скорость с нагрузкой, об/мин	20 - 4000
Номинальное напряжение, В	6
Напряжение питания, В	3-12
Максимальный ток, мА	670
Ток без нагрузки, мА	55

Большинство робототехнических конструкторов используют именно эти моторы в связи с низкой ценой в 240 рублей. Данная цена обуславливается тем, что корпус мотора является пластиковым, а редуктор изготовлен из нейлона.

Устройство энкодера может существовать в качестве отдельно взятого датчика, а именно LM393. Данный датчик показан на рисунке 17 и представляет из себя плату с двумя ключевыми элементами. Одним из которых является микрочип компаратор напряжения LM393, благодаря которому данный датчик и получил такое название. И вторым элементом является концевой оптический датчик.

Рисунок 17 – Оптический датчик угла поворота LM393

Главным недостатком датчика является кодовый диск, который может устанавливаться лишь на моторы с выступающим валом с двух сторон. Кодовый диск, как правило, идёт в наборе с датчиком и имеет 20 прорезей для тактового считывания оптическим сенсором. Кодовые диски показаны на рисунке 18.

Рисунок 18 – Кодовые диски с 20 прорезями

Подключение данного датчика подразумевает заранее предусмотренные цифровые контакты микроконтроллера, используемые для прерываний. Средняя цена такого датчика составляет 100 рублей.

Основываясь на стоимости, можно заявить, что определённые электромоторы с встроенными энкодерами намного дороже чем моторы с отдельно дополняемыми датчиками угла поворота. Именно поэтому выбор был сделан в сторону наиболее дешёвого варианта. Также стоит подметить, что моторам с встроенным энкодером требуется дополнительная шина I^2C. Если же выбирать, основываясь на другие характеристики моторов, то можно сказать что скорость моторов играет не столько важную роль, в то время как потребляемый ток является подавляющем звеном. Выбор пал на двигатель постоянного тока TT MotorOperating, в том числе потому что ось вращения у данного типа двигателей выступает по обе стороны корпуса. Для крепления самого колеса и установки кодового диска. В данном проекте будет достаточно двух ведущих колёс и одной шариковой опоры.

2.2.5. Выбор драйвера для двигателей

Насколько известно, коллекторные двигатели подключать на прямую к микроконтроллеру не рекомендуется, так как он имеет ограничение по силе тока присоединённой к нагрузке. В момент отключения моторов в цепи создаются индуктивные выбросы, что может создать помехи на аналоговых выходах и тем самым привести к некорректной работе всего робота. Также из-за неравномерного потребления электроэнергии в цепи будут возникать просадки по питанию. Исправить эти проблемы, частично, позволяют драйверы двигателей постоянного тока, позволяющие распределить нагрузку на колёса требуемым образом. Ещё одной их задачей является регулирование скорости и направления двигателей постоянного тока, что очень важно в нашей работе.

Для плавного управления скоростью двигателей требуется процесс преобразования из цифрового значения в аналоговый. Выполнить поставленную задачу можно либо посредством модуля цифро-аналогово преобразователя (ЦАП), либо наличием шины широтно-импульсной модуляции (ШИМ) на плате микроконтроллера.

Самый распространённым модулем цифро-аналогово преобразователя является MCP4725, изображённый на рисунке 19. Перевод осуществляется в зависимости от напряжения, подаваемого на плату.

Рисунок 19 – Цифро-аналогово преобразователь MCP4725

Как видно из рисунка, на модуле присутствует 6 выводов:

– OUT (аналоговый выход напряжения);

– GND (земля для выходного аналогового напряжения);

– SCL (линия синхронизация протокола I^2C);

– SDA (линия передачи данных протокола I^2C);

– VCC (входное опорное напряжение 5V);

– GND (земля для опорного напряжения).

Почти все преобразователи подобного типа подключаются по протоколу I^2C, что добавляет надобность в ещё одной дополнительной шине I^2C.

Намного проще было бы найти микроконтроллер с шиной широтно-импульсной модуляции. В отличии от ЦАП модуля, процесс преобразования

регулируется временем. Осуществляется это постоянным переключением питания от 5 до 0 вольт и наоборот. То есть резкой подачей либо логической единицы, либо логического нуля. Подобрав время импульсов и пауз, можно регулировать скважность сигнала, на основе которого и осуществляется преобразование в аналоговое значение.

Большинство драйверов управляются ШИМ сигналом, в ходе которого скорость регулируется подачей аналогово значения от 0 до 255 на один пин. Причём для вращения колеса в обратную сторону нужно инвертировать ШИМ сигнал, посредством подачи на другой пин микроконтроллера обратного логического значения.

Рассмотрим, изображённые на рисунке 20, тройку драйверов по управлению моторами: MX1508, L9110S, TA6586.

а) б) в)

Рисунок 20 – Драйверы для двигателей: а) MX1508; б) L9110S; в) TA6586.

Различие этих драйверов заключается лишь в параметрах, сведённых в таблицу 7.

Таблица 7 – Характеристики драйверов для двигателей.

Наименование	MX1508	L9110S	TA6586
Рабочее напряжение, В	2 – 9,6	2,5 – 12	3 – 14
Ток для одного канала, А	1,5	0,8	5
Пиковый ток, А	2,5	1,5	7
Цена, руб	20	50	100

Все эти драйверы подходят по рабочему напряжению. Ключевым требованием является ток для одного канала, который должен превышать значение в 670 миллиампер для выбранных моторов TT MotorOperating. И последним по приоритету критерием является стоимость. Из таблицы можно увидеть, что наиболее подходящим драйвером является MX1508.

2.2.6. Выбор приёмника информации

Как и в случае с питанием, для связи с управляющим модулем мини-полигона требуется мобильный модуль, установленный на роботе, для получения информации. То есть контакт должен осуществляться посредством беспроводного сигнала. В наше время — это довольно легко осуществимо посредством технологии беспроводных сетей. Наибольшую популярность набрали именно две такие технологии: Wi-Fi и Bluetooth. По сравнению Bluetooth уступает Wi-Fi во многом, начиная от скорости передачи информации и заканчивая рабочим частотным диапазоном. Но всё же главными преимуществами Bluetooth является его низкая стоимость и простота программирования. Такие параметры как частота и дальность сопряжения не столь важны, так как радиопередатчики будут находиться практически рядом друг с другом.

На российском рынке радиокомпонентов довольно большой ассортимент беспроводных модулей с протоколом Bluetooth, стоимость

которых варьируется от 1000 до 3000 рублей, что является довольно затратным. Поэтому в рассмотрении был ещё и китайский рынок Bluetooth модулей. Модуль Bluetooth HC-05, показанный на рисунке 21, оказался наиболее подходящим из-за своих характеристик, которые сведены в таблицу 8.

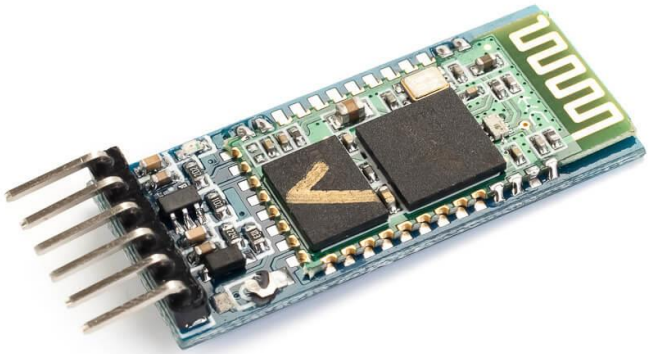


Рисунок 21 – Модуль Bluetooth HC-05

Таблица 8 – Характеристики модуля Bluetooth HC-05.

Модель	HC-05
Рабочее напряжение, В	3,3-5
Максимальный ток, мА	45
Скорость передачи данных, бод	1200 - 1382400
Рабочие частоты, ГГц	2,40 – 2,48
Версия Bluetooth	2,1
Дальность связи, м	30
Стоимость, руб	400

2.2.7. Выбор микроконтроллера

В качестве управляющей микросхемы, для реализации поставленных задач, выбор пал на Arduino, так как на данной платформе проводилось обучение в университете.

Условия, по которым осуществлялся подбор конкретной модели платы Arduino:

Тактовая частота свыше 16 мегагерц;

– питание от 5 вольт;

– наличие контактов для связи протокола I^2C (SCL, SDA);

– наличие минимум 2 контактов прерывания;

– наличие минимум 5 контактов ШИМ сигнала;

– наличие 2 контактов коммуникации (RX, TX).

«Наиболее достойными кандидатом оказалась плата Arduino Uno Rev 3, продемонстрированная на рисунке 22. Работа этой модели основана на чипе ATmega328, имеющем на борту 32 КБ флэш-памяти, 2 Кб SRAM и 1 Кбайт EEPROM памяти. На периферии имеет 14 дискретных (цифровых) каналов ввода или вывода и 6 аналоговых каналов ввода или вывода, это очень разносторонне-полезные девайсы, позволяющие перекрывать большинство любительских задач в области микроконтроллерной техники. Данная плата контроллера является одной из самых дешевых и наиболее часто используемых.» [11]

Рисунок 22 – Плата микроконтроллера Arduino Uno Rev 3

Ещё одним претендентом является плата Arduino Nano, изображённая на рисунке 23. Она является точным функциональным аналогом модели Uno Rev 3 за исключением отсутствия собственного гнезда для внешнего питания. Модель Nano имеет меньшие габариты, а соответственно, и цену, что делает её применимой в более компактных проектах.

Рисунок 23 – Плата микроконтроллера Arduino Nano

Более подробные характеристики Arduino Uno Rev 3 можно увидеть в таблице 9.

Таблица 9 – Расширенный список характеристик Arduino Uno Rev 3.

Микроконтроллер	ATmega328
Рабочее напряжение, В	5 В
Входное напряжение (рекомендуемое), В	7-12
Входное напряжение (предельное), В	6-20
Цифровые выводы	14
Аналоговые выводы	6
Количество ШИМ выводов	6
Количество выводов для передачи информации	2
Количество выводов внешнего прерывания	2
Наличие интерфейса I^2C	Имеется
Постоянный ток через выводы, мА	40
Флэш-память, Кб	32
Тактовая частота, МГц	16
Цена, руб	450

2.3. Разработка схемы электрической соединений

Опираясь на составленную структурную схему, а также выбор всех необходимых элементов была составлена схема электрическая соединений устройства. Схема электрическая соединений была разработана в комплексной системе автоматизированного проектирования КОМПАС 3D 18 версии и представлена на рисунке 24.

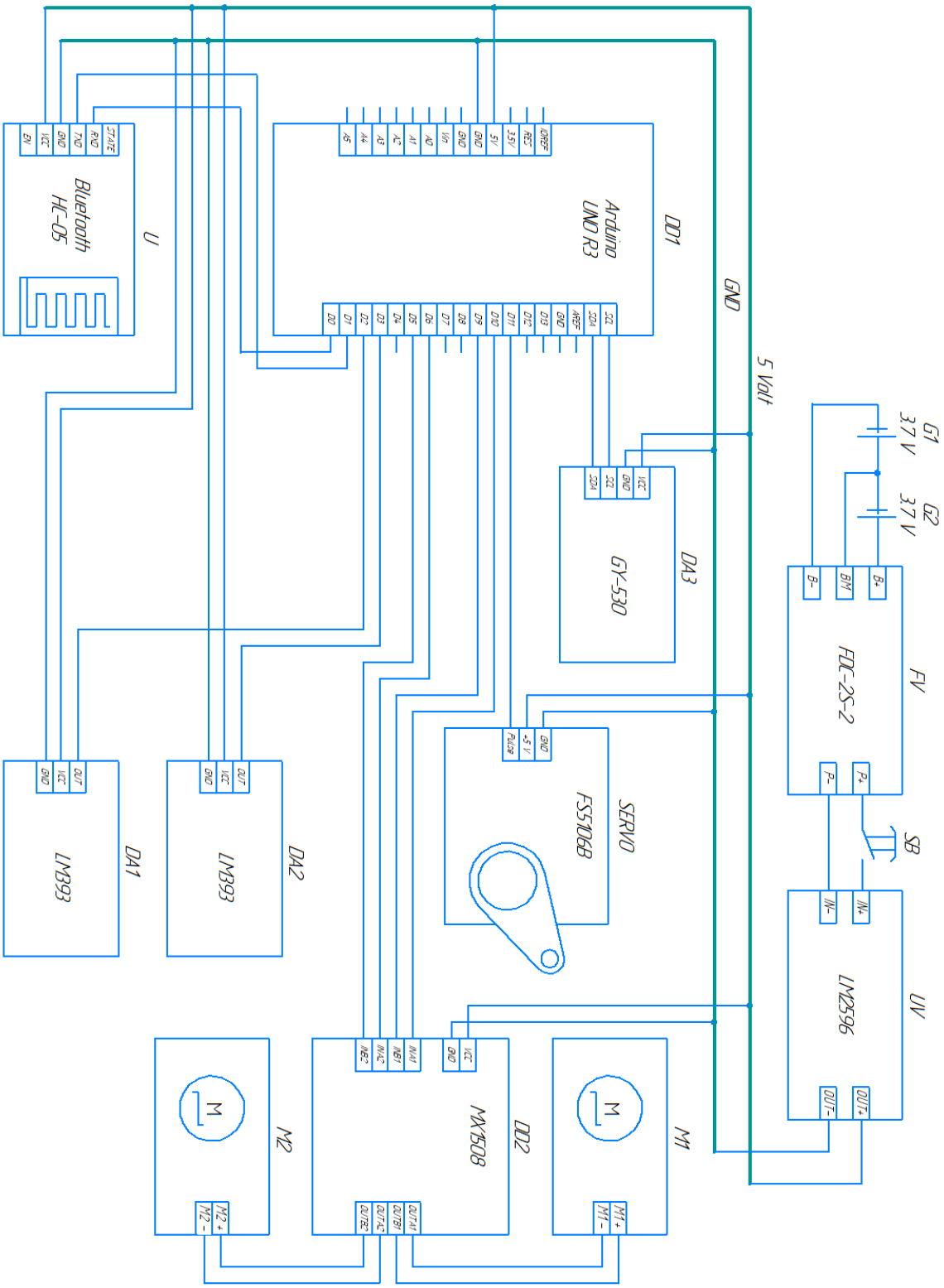


Рисунок 24 – Схема электрическая соединений

Питание на все устройства подводится через общую шину питания, которая в свою очередь питается от аккумуляторов. Подключение модуля Bluetooth осуществляется к последовательной шине контактов, то есть к цифровым выводам 0 (RX) и 1 (TX). Они используются как раз-таки для получения и передачи данных. Из всех 14 цифровых выводов, выводы 2 и 3 сконфигурированы на вызов прерывания, и именно к ним подключаются оптические энкодеры. Шина широтно-импульсной модуляции распространяется на цифровые выводы 3, 5, 6, 9, 10 и 11. Так как 3 вывод уже занят, то драйверу подходят выводы 5 и 6 под один двигатель, и выводы 9 и 10 под другой. Оставшийся 11 вывод подсоединяется к сервоприводу. На плате Arduino Uno Rev 3 предусмотрены специальные контакты SCL и SDA, которые используют для передачи данных две двунаправленные линии связи при поддержке протокола I^2C. К ним соответственно стоит подключить датчик расстояния, а остальные выводы датчика просто подводятся к любым свободным цифровым портам.

Выводы по разделу

Подводя итоги по разделу, была реализована структурная схема и схема электрическая соединений. Рассмотрен принцип работы всех необходимых устройств и возможность их синергии друг с другом. Основываясь на собственных знаниях и источниках литературы, а также доступности рынка и ценовых категорий получилось составить перечень выбранных электрокомпонентов: управляющая плата Arduino Uno Rev 3, два оптических энкодера LM393 с кодовыми дисками, два комплекта колёс с электромоторами TT MotorOperating, драйвер двигателей MX1508, защитная плата FDC-2S-2, понижающий преобразователь напряжения LM2596, два аккумулятора Li-Ion 18650, сервопривод FS5106B, датчик расстояния GY-530, двухпозиционная кнопка.

3. Разработка конструкции

Для достижения главной цели следующей поставленной задачей стояла разработка конструкции каркаса беспилотного мобильного робота с уже подготовленными площадками под крепления различных компонентов, выбранных ранее. Разработка твердотельной модели осуществлялось в программном обеспечении Fusion 360 от Autodesk, которая предоставляет учащимся бесплатную лицензированную версию со сроком в один год. Благодаря этой программе и базовым знаниям по работе в программах сфере трёхмерного моделирования, можно эффективно осуществить поставленную задачу по моделированию твердотельных объёмных объектов.

Самым распространённым методом производства является 3D печать. Работа печати осуществляется посредством 3D принтера, в основе принципа которого лежит аддитивные производство. Аддитивное производство – это процесс образования первого слоя основания заготовки и последующего наращивания следующих слоёв поверх основания с целью получения трёхмерной модели чего-либо. Аддитивные технологии применяются в различных профессиональных сферах и поэтому в качестве заправляемого материала может лежать как металл, так и дерево. Но наибольшую известность получил пластик из-за своей дешевизны и простоты производства.

Пластик, используемые в качестве заправки для печати, можно разделить на несколько видов: базовый, экзотический, профессиональный. Для поставленной задачи по разработке конструкции каркаса рассматриваться будет только базовый вид пластика.

Существуют разные типы базового пластика, которые подразделяются по своему составу:

– ПЛА (производится из молочной кислоты, также известной как полилактид);

– АБС (состоит из трёх ингредиентов: акрилонитрила, бутадиена и стирола);

– ПЭТГ (полиэтилентерефталат-гликоль);

– ТПЭ (термопластичные эластомеры);

– ПК (поликарбонат);

– Нейлон.

Каждый вид пластика обладает отличительными друг от друга характеристиками свойств, которые сведены в таблицу 10.

Таблица 10 – Свойственные характеристики различных видов пластика.

Свойства	ПЛА	АБС	ПЭТГ	ТПЭ	ПК	Нейлон
Прочность	Высокая	Высокая	Высокая	Средняя	Очень высокая	Высокая
Эластичность	Низкая	Средняя	Средняя	Очень высокая	Средняя	Высокая
Долговечность	Средняя	Высокая	Высокая	Очень высокая	Очень высокая	Высокая
Сложность производства	Низкая	Средняя	Низкая	Средняя	Средняя	Средняя
Усадка и деформация	Мин.	Знач.	Мин.	Мин.	Знач.	Знач.
Стоимость	Низкая	Средняя	Средняя	Высокая	Высокая	Высокая

Опираясь на данные из таблицы можно подчеркнуть ПЭТГ пластик как наиболее подходящий из остальных типов пластика. Так как в разработке проекта важна не малая прочность, пониженная усадка и высокая долговечность.

Для установки лидара потребуется самая высокая точка на поверхности конструкции, поэтому расположение оставшихся компонентов не должно мешать работе устройства лидара. В связи с этим конструкция будет состоять из двухуровневого каркаса, осуществлённым с помощью двух деталей: нижнего и верхней.

Нижняя часть каркаса разрабатывалась индивидуально и изображена на рисунке 25. Каждый элемент детали имеет собственное предназначение и пронумерован для дальнейшего его описания. Опоры 1 и 2 предназначены для крепления между ними мотор-редуктора, а также крепления верхней части каркаса. Цифра 3 так же обозначается как опора, но выполнена уже для сцепления передней и задней частей верхней детали при помощи отверстий под винты. Пространство 4 предусмотрено для установки ведущего колёса. Для лучшей управляемости на робота стоит установить ещё одно шариковое колесо. Одним из собственных требований являлось расположение плоской части детали параллельно поверхности макета мини-полигона. Из-за разности высоты крепления шарикового и ведущих колёс, следовало разработать поддерживающую площадку 5 для шариковой опоры. Под цифрой 6 значится отверстие для установки энкодера с кодовым диском и отверстиями для их установки при помощи винтов и гаек. Отверстия 7 сделаны специально для прокладки проводов между уровнями конструкции. Ещё одно отверстие 8 разработано исключительно для проводки энкодера для экономии длины проводов.

Верхняя деталь каркаса изображена на рисунке 26 и разрабатывалась аналогично нижней, но с некоторыми отличиями. Так опоры 9 и 10 были сдвинуты от края детали для последующего плотного соприкосновения со опорами нижней части каркаса. Сделан паз 11 для вставки опоры 2, выполненной в качестве шпонки. Без особых изменений остались отверстия проводки 12 и 13, а также пространство под колесо 14. Стоит подметить, что каждый пронумерованный элемент симметричен относительно оси и имеет схожее описание, за исключением площадки под шариковую опору.

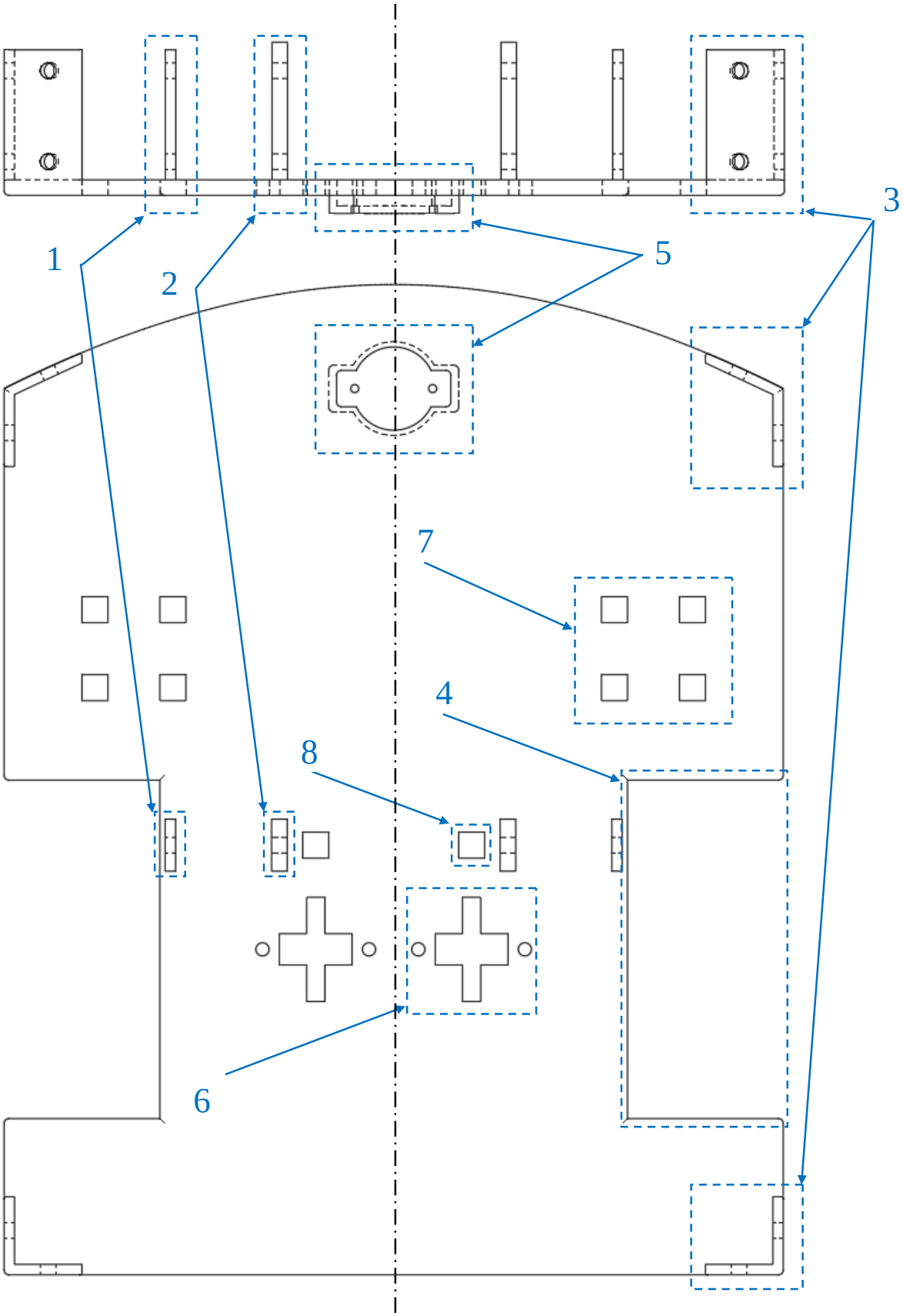


Рисунок 25 – Нижняя деталь конструкции каркаса

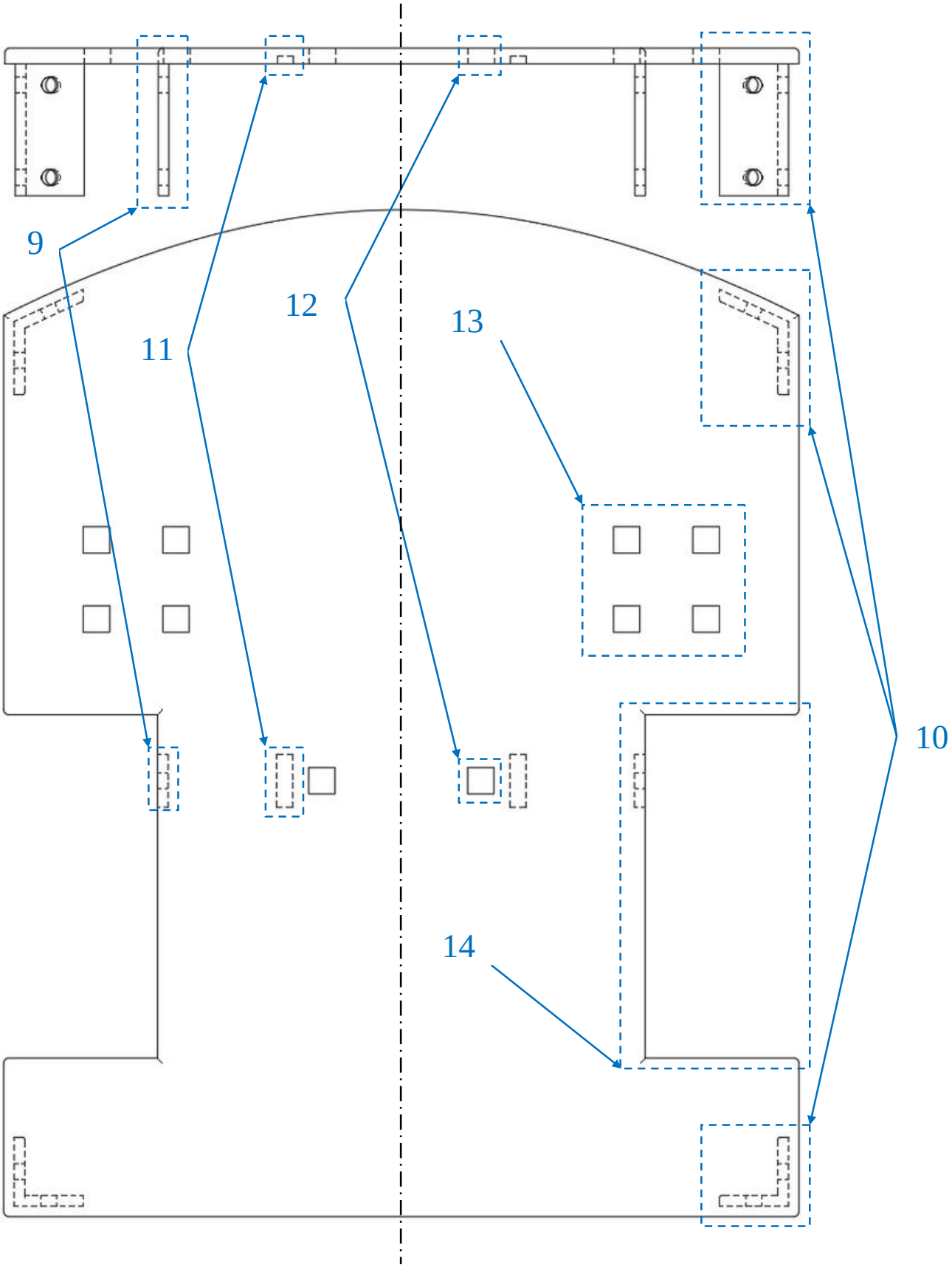


Рисунок 26 – Верхняя деталь конструкции каркаса

Выводы по разделу

Итого, в данном разделе выполнена ещё одна из поставленных задач, а именно разработка конструкции каркаса мобильной платформы. Где ключевым решением был выбор способа разработки и материла конструкции. Проведён анализ характеристик различных типов материала с дальнейшим выбором наиболее походящего из них. Также были смоделированы две детали каркаса с описанием требуемых элементов детали для установки всех необходимых комплектующих с учётом рациональной расстановки.

4. Программная часть

Как и любой человек не может существовать без смысла жизни, так и устройство не может существовать без программы. Именно благодаря программе устройство может функционировать и выполнять требуемые от неё задачи.

4.1. Алгоритмизация

Написание любой программы начинается с пошагового построения логики, в которой шаг за шагом идёт определенное действие, которое и приводит к конкретной цели. Процесс построения логики называется алгоритмом, который реализуется в виде блок-схем. Целью роботизированной мобильной платформы является движение по заданному маршруту внутри мини-полигона из точки А в точку Б. Маршрут прокладывается пользователем аппаратно-программного комплекса, после чего по беспроводной связи на платформу приходит значение, которое определяет какое именно действие ей придётся выполнить. Действия также принято называть функциями или же предопределённым процессом при построении блок-схем. Первым шагом по составлению алгоритма является подключения необходимых библиотек по работе с микроконтроллером и объявления всех переменных. После стандартной процедуры подключения следует получить то самое значение, которое решит какая функция запуститься. Из схемы видно, что если приходит значение 0, то запускается функция «ВПЕРЁД С ЛИДАРОМ», иначе рассматриваются остальные возможные варианты. После выполнения какой-либо функции или же получения неизвестного значение, которое не удовлетворяет ни одному условию, программа становиться в режим ожидании нового значения. На рисунке 27 продемонстрирована блок-схема управляющей программы.

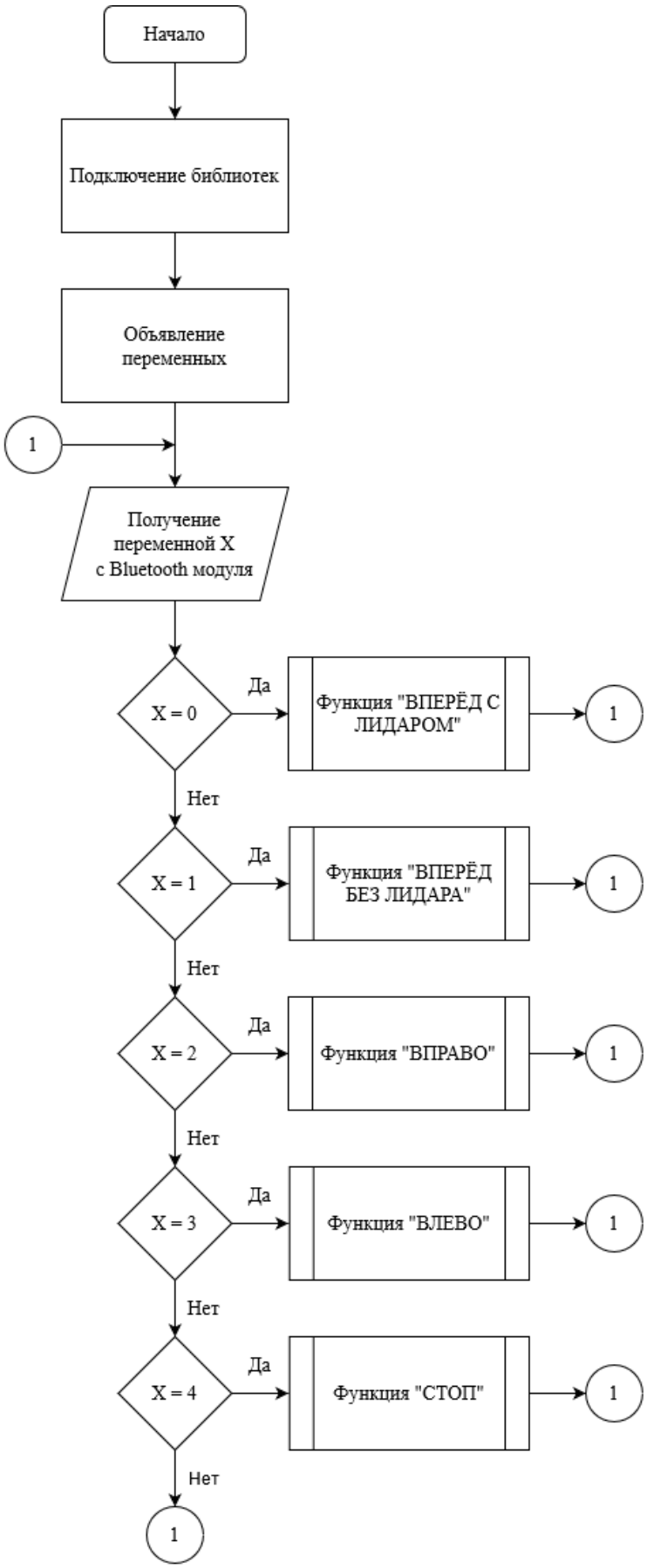


Рисунок 27 – Блок-схема управляющей программы

Внутри каждого блока предопределённого процесса происходят свои собственные операции и может иметься ещё несколько функций. Так в блоке функции «ВПЕРЁД С ЛИДАРОМ» первым шагом выполняется проверка значения. В случае прохождения проверки подаётся сигнал вперёд на оба колеса мобильной платформы, после чего запускается функция обработки данных с лидара. В случае, когда проверка не пройдена и на платформу пришла уже другое значение, то цикл функции обрывается. Далее идёт проверка наличия препятствия перед платформой, которая либо запускает функцию избегания столкновений, либо функцию удержания центра дороги. И так происходит вновь и вновь, пока значение X остаётся равным 0. Блок-схема функции «ВПЕРЁД С ЛИДАРОМ» показана на рисунке 28.

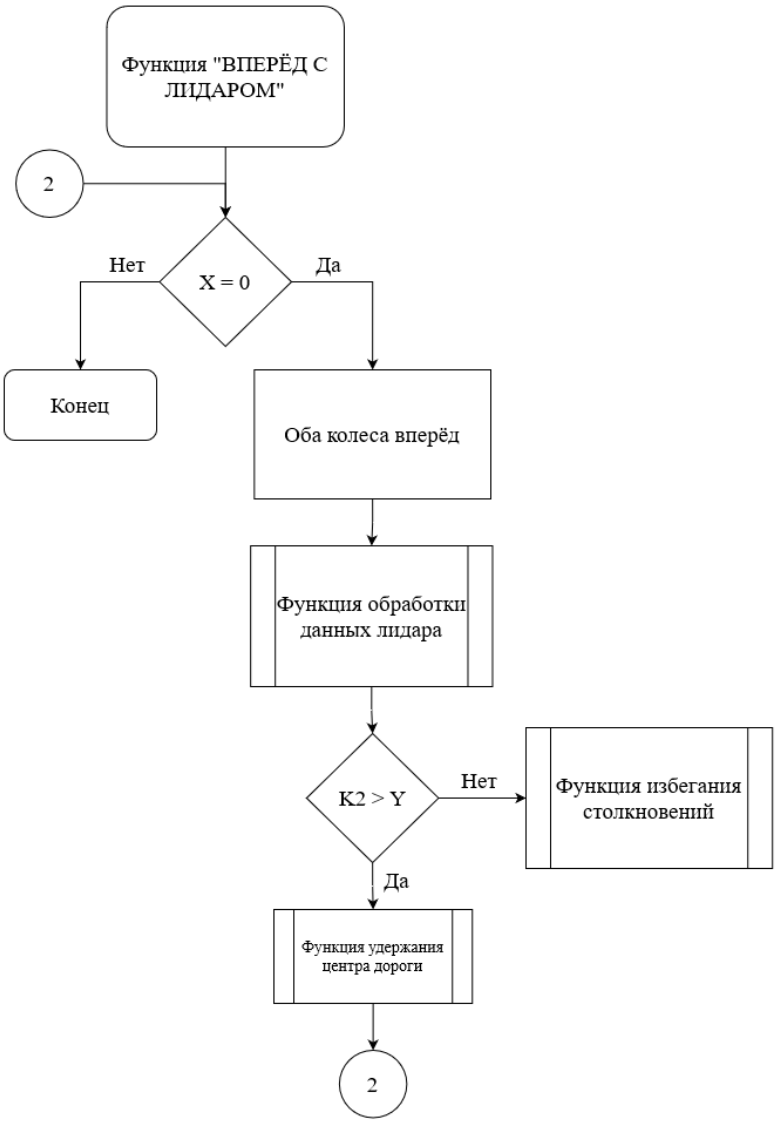


Рисунок 28 – Блок-схема функции «ВПЕРЁД С ЛИДАРОМ»

Функция «ВПЕРЁД БЕЗ ЛИДАРА» показана на рисунке 29. Данная функция будет вызываться крайне редко и предназначена лишь в целях въезда на макет мини-полигона. Суть заключается в слепом движении платформы в прямом направлении на определённое расстояние, которое будет отслеживаться оптическими энкодерами, после чего запускается функция «СТОП».

Рисунок 29 – Блок-схема функции «ВПЕРЁД БЕЗ ЛИДАРА»

Функция «ВПРАВО» предназначена для поворота направо строго на 90 градусов, осуществляется это одновременным движением колёс в разном направлении. Потом мобильной платформе требуется проехать немного вперёд, чтобы устройство лидара зафиксировало стенку коридора после поворота. Функция заканчивается на активации функции «СТОП». На рисунке 30 можно увидеть блок схему функции «ВПРАВО».

Рисунок 30 – Блок-схема функции «ВПРАВО»

Функция «ВЛЕВО» предназначена для поворота на перекрёстке в левую сторону и аналогична функции «ВПРАВО». Обе эти функции включаются лишь на секторе с разветвлением дороги. Блок-схема функции «ВЛЕВО» изображена на рисунке 31.

Рисунок 31 – Блок-схема функции «ВЛЕВО»

Функция «СТОП» вызывается в большинстве других блоков предопределённого процесса и хранит в себе команду остановки подачи сигнала на колёса и сервопривод. Блок-схема этой функции показана на рисунке 32.

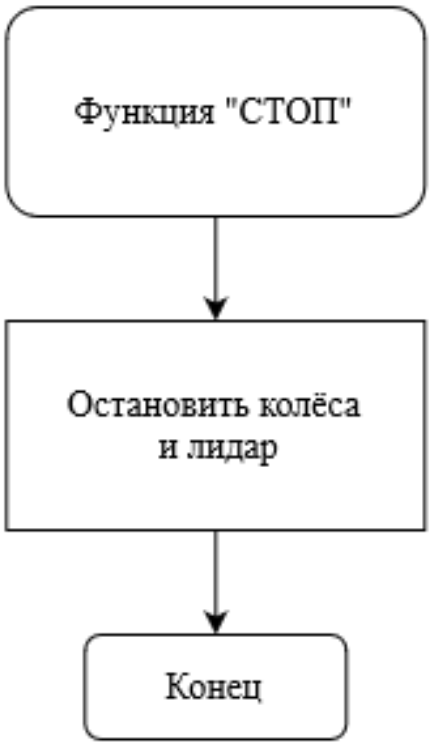


Рисунок 32 – Блок-схема функции «СТОП»

Следующей будет функция обработки данных с лидара, главной ролью которой будет подсчёт коэффициента среднего расстояния. Каждая область, состоящая из 60 градусного диапазона будет иметь собственный коэффициент среднего расстояния и заменяться каждый раз, как датчик будет собирать информацию по новой. Блок-схема этой функции можно лицезреть н рисунке 33.

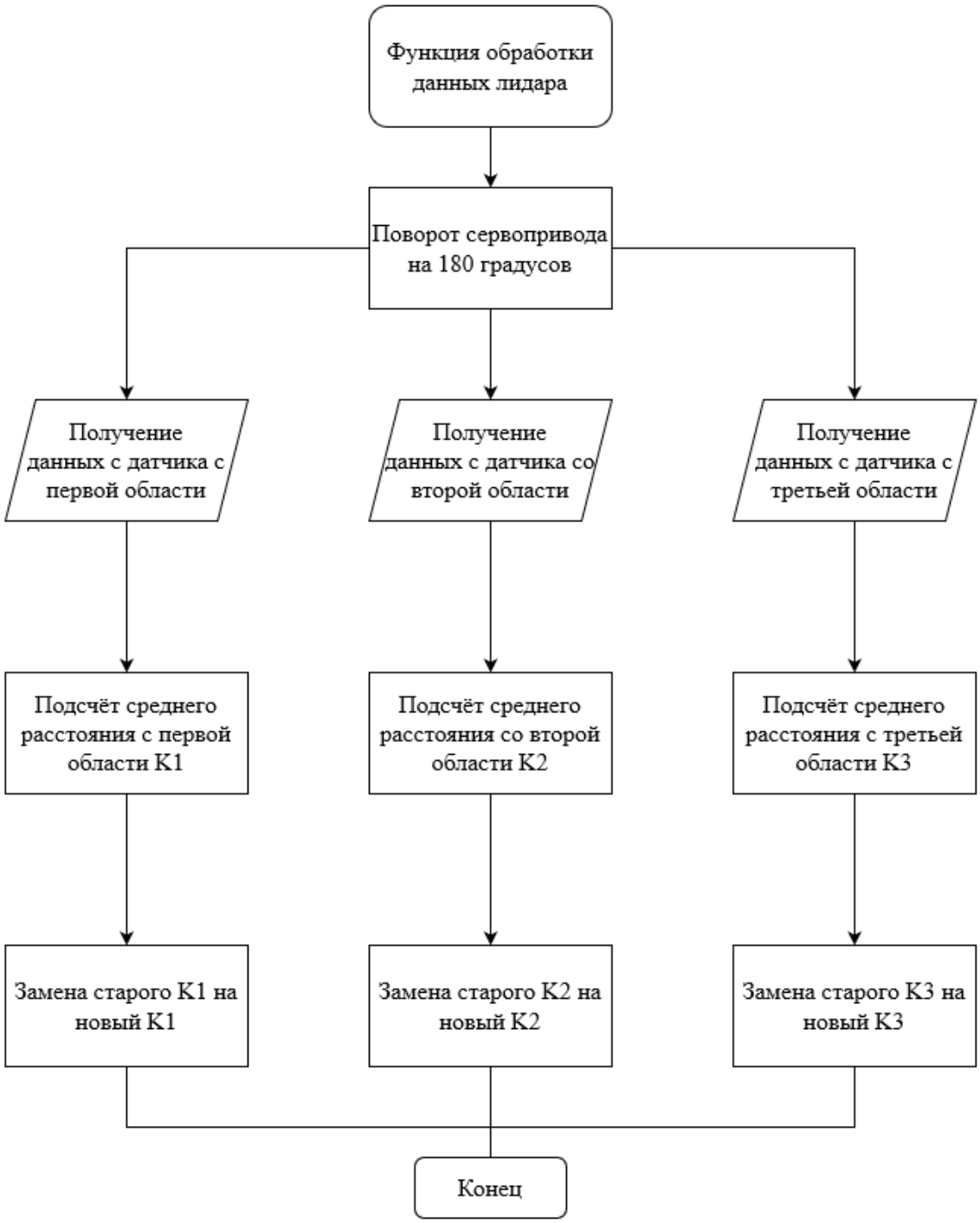


Рисунок 33 – Блок-схема функции обработки данных с лидара

Следующей не менее важной является функция избегания столкновений (см. рисунок 34). Устройство лидара помогает избежать аварий

платформы со стенками мини-полигона и является неотъемлемой частью системы беспилотного движения автомобиля. В программе прописывается постоянное установочное значение, опираясь на которое функция сравнивает показания коэффициентов среднего расстояния. Если мобильный робот движется ровно вдоль заданного направления, то запускается функция по удержанию центрального положения на дороге (см. рисунок 35). Если же каким-то образом платформа подъехала близко к стенке, то движение колёс останавливается. Затем на колёса поступает сигнал о движении назад на определённое расстояние. Удержание центра дороги позволяет корректировать движение всей платформы, за счёт встроенного ПИД-регулятора в программу, методом изменения скорости колёс. ПИД-регулятор, в свою очередь, получает данные с функции обработки данных показателей среднего расстояния. Уже после запуска цикла по удержанию положения на дороге функция «ВПЕРЁД С ЛИДАРОМ» возобновляет движение всей платформы.

Рисунок 34 – Блок-схема функции избегания столкновений

Рисунок 35 – Блок-схема функции удержания центра дороги

4.2. Разработка управляющей программы

Основываясь на разработанную блок-схему алгоритма начался процесс написания кода управляющей программы. Разрабатываемый код пишется на программном обеспечении Arduino IDE (Integrated Development Environment), основным языком которого является немного модифицированный C++.

Сама программа загружается непосредственно в микроконтроллер, который в свою очередь подключён к программатору, в нашем случае это персональный компьютер.

Как и в любой программе, всё начинается с подключения библиотек по работе устройств. Библиотека Servo ответственна за работу сервопривода и включает в себя простые команды по управлению скорости и угла поворота сервопривода. Следующая библиотека VL53L0X ответственна за работу датчика расстояния. Как правило, вместе с этой библиотекой подключается ещё и библиотеку Wire, потому что датчик расстояния осуществляется своё подключение по интерфейсу I^2C.

Следующим не менее важным этапом идёт распиновка, так называют процесс объявления каждого конкретного вывода устройства под конкретный вывод микроконтроллера, по которому будет осуществляться подключение. Из кода программы можно увидеть, что пины 5 и 6 прописываются для движения левого двигателя, а пины 9 и 10 для правого двигателя. Сервоприводу присваивается пин 11. После уже идёт создание объекта устройства в программе, то есть под каким названием оно будет подразумеваться дальше на протяжении всей программы.

Согласно алгоритму, следующим этапом идёт объявление постоянных и вспомогательных переменных, а также коэффициентов и прочих значений. Некоторые переменные ещё не имеют значения и будут прописаны только после проведения экспериментальной части.

В блоке void Setup() выполняются разовые команды, которые будут использоваться единожды и только в начале программы. К таким командам можно отнести запуск последовательного порта, по которому с программатора будет загружаться код в микроконтроллер, и задача скорости обмена данными. Также будут использоваться команды по установки режима пинов моторчиков на выход данных и установке уровня пинов на низкий параметр. Подключаются пины энкодеров для считывания прерываний.

Оставшимися командами в этом блоке являются запуск интерфейса, устройства сервопривода и датчика расстояния.

После блока void Setup() пишутся различного рода функции. Делается это ради сокращения всего кода. Чтобы каждый раз не писать перечень команд, которые выполняют одну и ту же задачу, создаётся функции с определённым названием. В момент, когда нужно вызвать функцию пишется всего лишь название той самой функции, которую требуется вызвать. Первыми являются функции прерывания для каждого колеса, чтобы вызывать их каждый раз как требуется запустить колесо и вызывать то количество раз, на сколько стоит провернуть это самое колесо.

Следующая функция отвечает за регулировку положения роботизированной платформы относительно разности расстояний между левой и правой стеной. Регулировка осуществляется по пропорционально-интегрально-дифференцирующему (ПИД) закону, согласно которому корректируется мощность, подаваемая на каждое колесо, посредством сложения трёх составляющих: пропорциональной интегральной и дифференциальной. Сложение этих составляющих позволяет воздействовать на какой-либо параметр регулирования. Если рассматривать более подробно, то регулирование осуществляется за счёт ошибки. Ошибкой называют разность установочного значения и входного. Установочное значение – это то, к чему стремится процесс регулирования. Пропорциональная составляющая исправляет ошибку полученную в текущий момент обработки. Интегральная составляющая накапливает все ошибки и на основании их воздействует на новую. Дифференциальная составляющая помогает компенсировать резкие изменения в системе регулирования.

Функция обработки данных с лидара работает по принципу постоянного снятия показаний, не в зависимости перешёл ли сигнал один раз или несколько с одного градуса. То есть, при положении сервопривода на 15 градусов датчик может снять как одно, так и три показания. Чтобы учесть этот минус, грамотным решением было посчитать количество показаний,

которые снял датчик на промежутке от 0 до 59 градусов и просуммировать каждое значения расстояния. После чего сумму всех расстояний поделить на количество этих показаний, другими словами, вычислить среднее арифметическое. Из данного определения можно вывести формулу 2, формулу коэффициента среднего расстояния *i*-той области.

$$K_i = \frac{\Sigma S}{n}, \quad (2)$$

где S – снятое показание расстояния с датчика, мм;

n – количество снятых состояний.

Также в этой функции вычисляется значение входной переменной input для ПИД регулирования, в которой коэффициент левой K1 области вычитывает коэффициент правой области K3. В конце всей функции, когда оставшаяся часть программы выполнила свои действия на основании первоначально полученных коэффициентов, все значения обнуляются, чтобы собрать их повторно.

Функция остановки движения является функцией «СТОП» из алгоритма. Она частично отличается от описания, в строчках кода этой функции подаётся LOW сигнал на колёса, что даёт им команду остановиться. В это время лидар может продолжать крутится и снимать состояние, но оно уже ни на что не повлияет. То есть программа не останавливает положения лидара на каком-либо промежутке, а даёт закончить движение и остановиться, вернуться в исходное положение.

Функция движения вперёд просто на просто подаёт сигнал на оба колеса о движении вперёд с определённой скоростью при помощи команды analogWrite(pin, value), где pin это вывод который подключается в требуемому колесу, в нашем случае это MotorRightForward. В value это значение скорости с которой колесо будет крутится, в коде указана переменная Speed, что при объявлении носила значение 100.

В коде представлена функция избегания столкновений, которая в самом начале обнуляет все накопившиеся прерывания. После чего запускается цикл с условием, что пока перед платформой не появится достаточно места, то она будет сдавать назад на одно прерывание.

Самая главная функция по движению вперёд с лидаром сделана с большим упором на функцию ПИД-регулятора. Именно в ней запускается сервопривод и движется по значения 59, 119 и 180, после чего запускается функция регулирования, где на выходе формируется переменная output. Как и было сказано ранее, в команде analogWrite(pin, value) имеется value, которое может задаваться конкретной переменной. Но также эта фраза может задаваться и сразу несколькими переменными, например, из уже имеющийся переменной Speed вычитается значение output. То есть если Speed хранит значение 100, а output имеет значение 30, то при вычитании на колесо будет подаваться значение их разности, которое будет приравниваться к 70.

Функция поворота направо обнуляет значение прерываний с левого колеса в момент вызова. Обнуляется именно левое колесо, потому что при повороте оно будет ведущим, а правое будет лишь помогать ему. После чего запускается цикл с условием i<RightCounter, данный цикл будет выполняться до тех пор пока не пройдёт определённое количество прерываний, прописанное в начале. Поворотный манёвр завершается движением платформы вперёд после приворота колёс, чтобы избежать случаев постоянного разворота. Функция поворота вправо осуществляется точно так же.

После написания всех функций, заключающим будет блок void loop(). Этот блок является основным и без него не пишется ни одна программа на языке Arduino IDE. Заключающей она является потому все используемые в ней функции должны быть закодированы ранее. Ведь void loop() является бесконечным циклом, который выполняет до тех пор, пока не перестанет подаваться электричество на плату микроконтроллера. Именно в этом блоке и расписывается блок-схема алгоритма управляющей программы, где запуск

конкретной функции запускается в зависимости от полученного состояния case, при помощи оператора switch. По сути создаётся несколько альтернативных кодов внутри программы, которые запускаются при разных условиях. Описание программы соответствует разработанному коду в приложении Б.

4.3. Разработка программы визуализации

Не менее важным этапом была разработка программы визуализации работы лидара, главная функция которой отображать всё поле зрение лидара и высчитывать расстояние до каждого объекта в диапазоне 180 градусов. Благодаря этой функции является возможным наглядно понять, как лидар видит препятствия и видит ли их вообще.

Разработка программы осуществлялась в программной среде Processing. Processing сделана специально в целях разработки визуального контента, посредством собственного языка программирования, основанного на Java. При помощи этой среды можно реализовать простенький интерфейс, анимацию или же изображение.

Основным решением было принято реализовать анимацию, которая бы отображала диапазон сканирования лидара и встречающиеся на нём объекты. Как и в случае с управляющей программой требовалось отобразить диапазон 180 градусов с разбитием его на три равные части, но для лучшей наглядности была взята цена деления в 15 градусов. Чтобы отобразить на анимации инфракрасный луч датчика расстояния, была взята динамичная линия, которая двигалась по диапазону сканирования синхронно со скоростью вращения сервопривода. Чтобы успеть визуально принимать информацию, динамическая линия будет оставлять за собой затухающий след из других линий, который будут отбрасывать «тень» после встречи с объектом. Для визуального восприятия расстояния между датчиком и объектом рисовались дуги с разными радиусами. Разность расстояния

радиусов показывала бы как близко объект находится и входит ли он в диапазон сканирования. Также стоит добавить числовое отображение значений угла поворота датчика и расстояние до объекта. Код разработанной программы показан в приложении А.

Написание кода начинается с импортирования библиотеки Serial. Это делается для связи программы микроконтроллера и программной среды Processing. Данная библиотека позволяет принимать и отправлять сообщения через порт, к которому подключен микроконтроллер. После идёт объявление переменных и написание блока void setup(). В самом блоке прописывается размер создаваемого дисплейного окна в пикселях и параметры фона этого окна, а также подключение к порту и скорость передачи данных.

Первым шагом было заполнение дисплейного окна. Окно стоило поделить на две части: под дисплей обзора лидара и под численные характеристики. Осуществляется это с помощью трёх команд, определяющие такие параметры как контур, заполнение и размер. Контур не обязателен при создании обеих частей окон, поэтому используется команда noStroke(). Команда заполнения fill() включает в себя параметры цвета и прозрачности создаваемого кона. Команда для выбора размера прямоугольного части окна rect() задаётся по координате начальной и конечной точках. В итоге получается окно, состоящее из двух разноцветных прямоугольников (см. рисунок 36).

Рисунок 36 – Создание главного окна визуализации

Вторым шагом было создание сетки лидара с ранее упомянутыми дугами и линиями. Так же, как и в случае с окнами дисплея, для создания требуется указать три уже известны параметра. В данном проекте для пущего реализма линии сетки будут иметь зелёный цвет. Дуги рисуются при помощи команды arc(a, b, c, d, start, stop), где:

– a (координата X эллипса арки);

– b (координата Y эллипса арки);

– c (ширина эллипса арки по умолчанию);

– d (длина эллипса арки по умолчанию);

– start (угол начала арки (в радианах));

– stop (угол конца арки (в радианах)).

Линии создаются схожим образом, но при помощи команды line(x1, y1, x2, y2), где:

– x1 (координата X для первой точки);

– y1 (координата Y для первой точки);

– x2 (координата X для первой точки);

– y2 (координата Y для второй точки).

Второстепенные линии с ценой деления сделаны менее яркими и более тонкими. Результат второго шага можно увидеть на изображении 37.

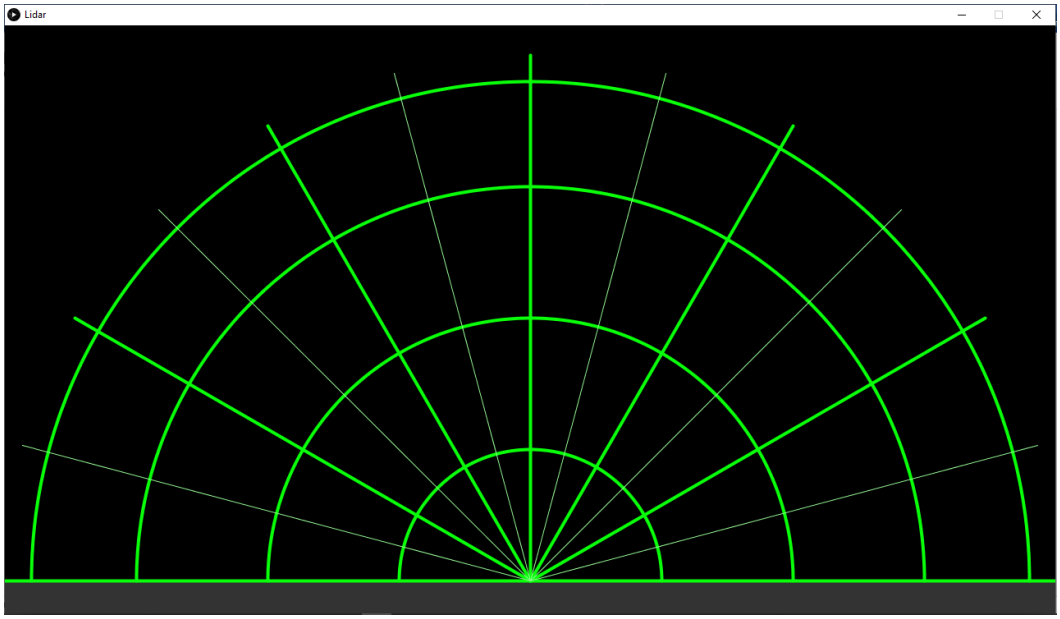


Рисунок 37 – Создание сетки дисплея лидара

Следующим шагом было создание анимации динамической линии. Для создания анимации потребуется изменять положение линии покадрово. То есть создавать новую линию с изменением конечной точки, сохраняя начальную. Ещё требовалось реализовать след после прохождения динамической линии. То есть с изменением кадра старая линия будет заменяться новой и в последствии становиться всё тусклее. Новая линия должна отображать на каком расстоянии при определённом градусе находится объект. Осуществить это можно способом падения тени, то есть линия будет иметь зелёный цвет до тех пор, пока не встретиться с препятствием, после чего будет отбрасываться тень красным цветом в точке встречи объекта. Если же объект находится за пределами последней дуги относительно начальной точки, то линия будет просто зелёной. Динамическая линия сделана более толстой и бежевого цвета для лучшего восприятия на чёрном фоне. Кадр анимации динамической линии можно увидеть на рисунке 38.

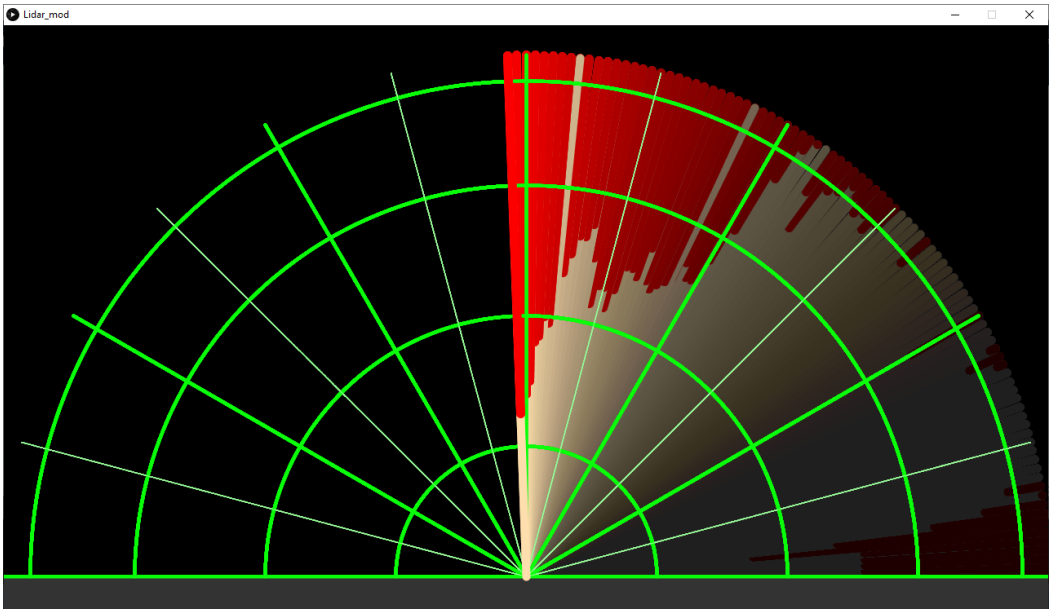


Рисунок 38 – Создание анимации динамической линии

Последним шагом по созданию программы визуализации было создание текста на дисплее. Чтобы пользователю было понятнее за какое расстояние отвечает конкретная дуга на дисплей нанесён текст, при помощи команды text(c, x, y), где:

– c (алфавитно-цифровой символ, который нужно нарисовать);

– x (координата X текста);

– y (координата Y текста).

Координаты текста можно задавать конкретными значениями или же опираясь на высотку экрана. На пример, имея размерность экрана 1280x720 можно указать числовое значение «640» для координаты X, а можно указать «width/2», что будет означать деление ширины экрана в 2 раза. Этот метод позволяет сориентировать текст относительно другого элемента на анимации. Под численные характеристики также создан статичный текст и выделено место куда будет выводиться угол обзора и расстояние. Данные будут передаваться с микроконтроллера на программу визуализации. Основываясь на этих данных можно скорректировать скорость вращения сервопривода и частоту опроса датчика расстояния. Итоговый вид программы визуализации можно увидеть на рисунке 39.

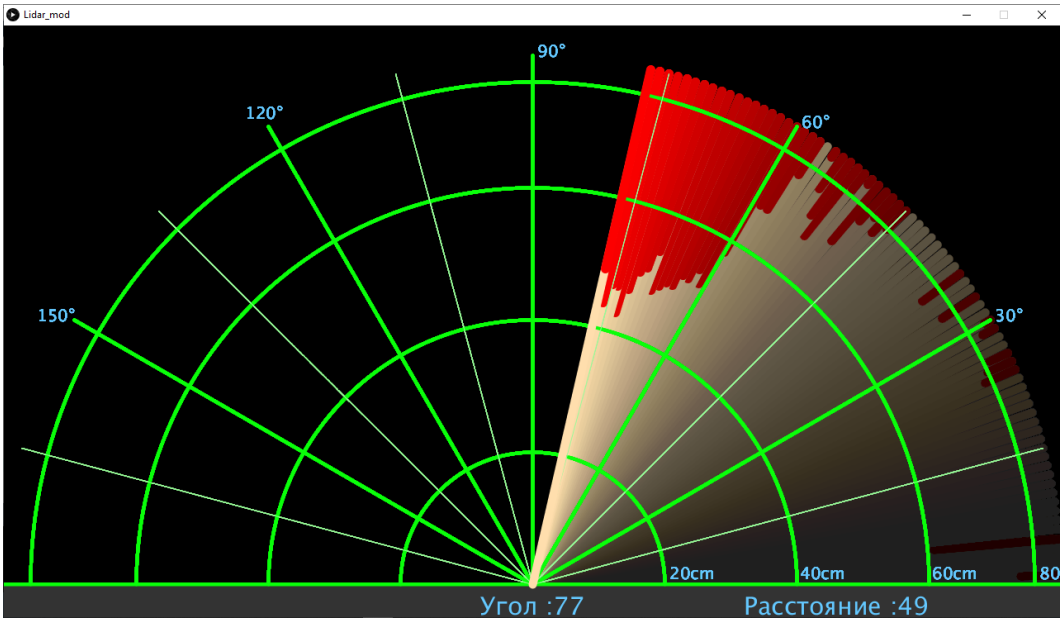


Рисунок 39 – Графический интерфейс программы визуализации

Выводы по разделу

Итого, в данном разделе выполнена ещё одна из поставленных задач, а именно разработка конструкции каркаса мобильной платформы. Где ключевым решением был выбор способа разработки и материла конструкции. Проведён анализ характеристик различных типов материала с дальнейшим выбором наиболее походящего из них. Также были смоделированы две детали каркаса с описанием требуемых элементов детали для установки всех необходимых комплектующих с учётом рациональной расстановки.

5. Конструкторско-экспериментальный раздел

5.1. Обработка результатов сканирования и отладка лидара

В прошлом разделе была разработана программа визуализации, при помощи которой наглядно получиться увидеть результаты сканирования. Лидар обладает такой характеристикой, как скорость вращения. Скорость вращения играет важную роль в быстродействии по принятию решений. Следует провести экспериментальное исследование, в ходе которого будет меняться параметр скорости вращения и по итогу выбираться наиболее подходящий.

Для полноценного проведения исследования создана частичная копия визуальной программы в виде плаката на листе формата А0, которую можно увидеть на рисунке 40.

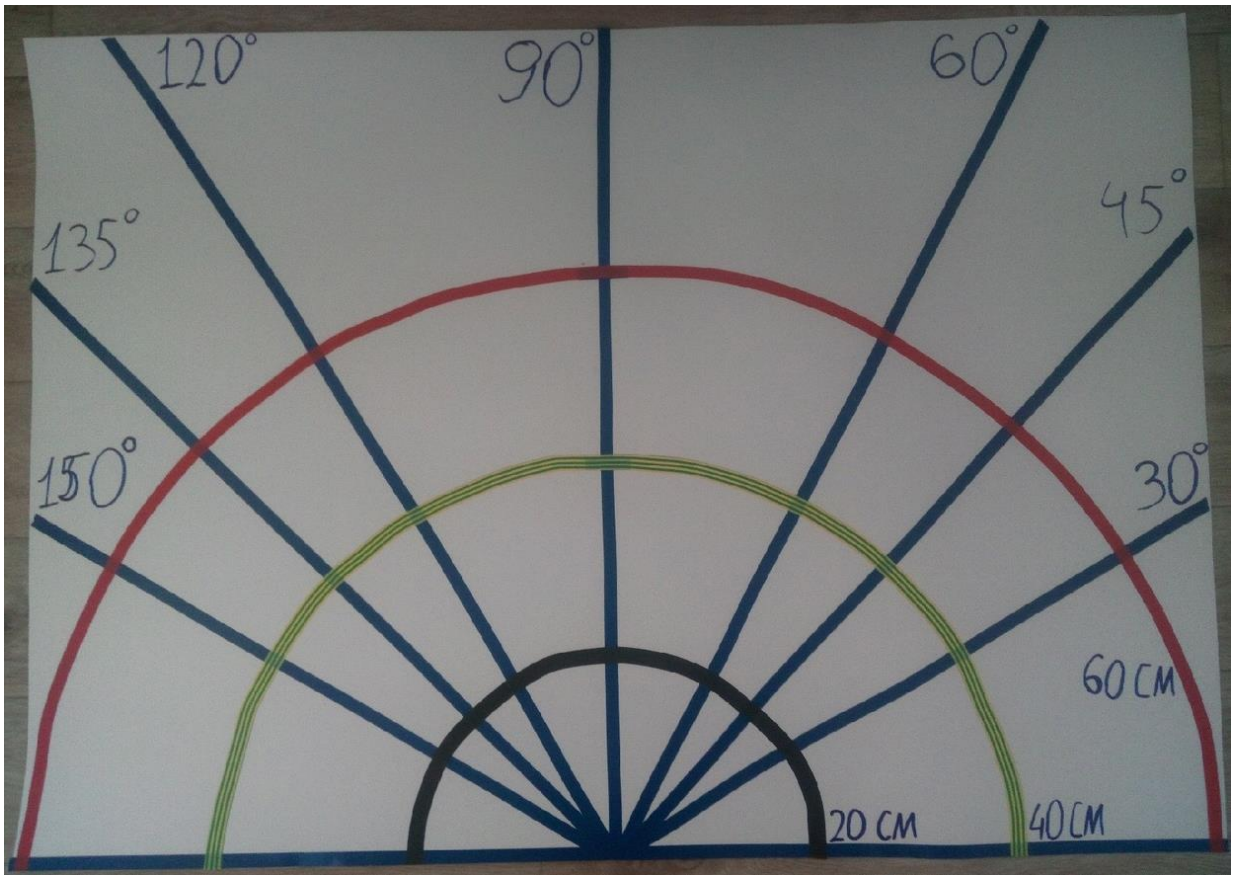


Рисунок 40 – Плакат-карта по расположению объектов

При помощи синих линий реализовано отображение угла поворота лидара, а при помощи цветных дуг дальность расположения объекта. Чёрная дуга отображает малую дистанцию вокруг лидара на расстоянии 20 сантиметров от устройства. Жёлто-зелёная дуга отображает среднюю дистанцию в 40 сантиметров. Красная дуга отображает высокую дистанцию в 60 сантиметров. Возьмём в качестве объекта зелёную четырёхугольную призму размером 80х40х40 миллиметров и расположим на средней дистанции в прямом направлении, как это показано на рисунке 41.

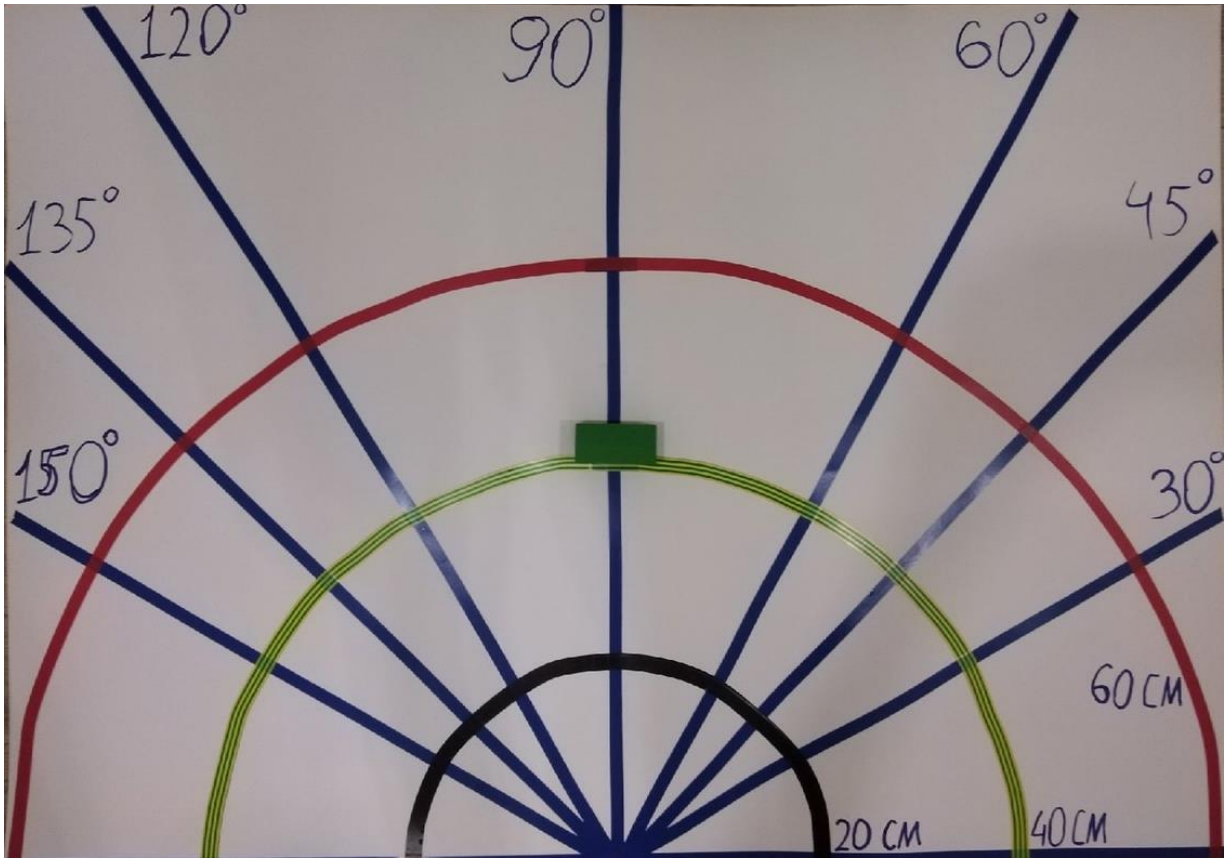


Рисунок 41 – Расположение объекта на дистанции 40 см и в направлении 90 градусов

В ходе экспериментального исследования в управляющую программу было записано три разных временных интервала, которые определяли частоту вращения лидара.

В начале подавали самый короткий интервал в 0,01 между каждым градусом. Пользуясь формулой 3, можно вычислить показатель скорости

вращения лидара ω в оборотах в минуту, для этого стоит перемножить временной интервал t и угол поворота φ. После чего получится временное значение в секундах, затраченное за один полный оборот. Чтобы перевести полученное значение в минуты требуется количество секунд в минуте поделить на вычисленное временное значение.

$$\omega = \frac{60}{t * \varphi}, \quad (3)$$

где ω – скорость вращения, об/м;

t – время прохождения от одного градуса к другому, с;

φ – угол поворота для совершения одного оборота, град.

Потом временному интервалу присваивалось значение 0,03 секунды и уже самым последним значением было 0,05 секунд. На основании заданных значений, а также известного угла поворота и использовании их в формуле по расчёту скорости вращения, можно высчитать показатель ω для каждого значения:

$$\omega_1 = \frac{60}{0{,}01 * 180} \approx 33{,}3 \text{ об/м}$$

$$\omega_2 = \frac{60}{0{,}03 * 180} \approx 11{,}1 \text{ об/м}$$

$$\omega_3 = \frac{60}{0{,}05 * 180} \approx 6{,}7 \text{ об/м}$$

На рисунках 42, 43, 44 можно видеть результат сканирования лидаром при разных скоростях вращения ω_1, ω_2, ω_3, соответственно.

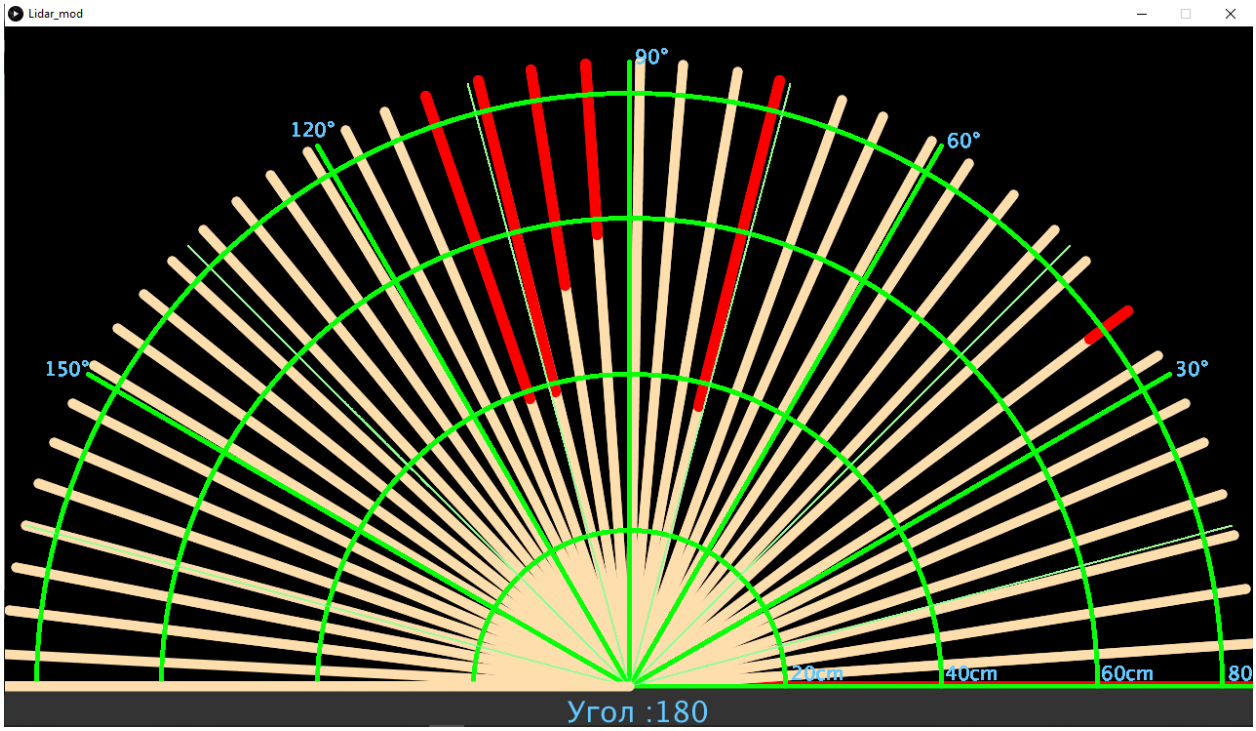


Рисунок 42 – Результаты сканирования при 33,3 оборотах в минуту

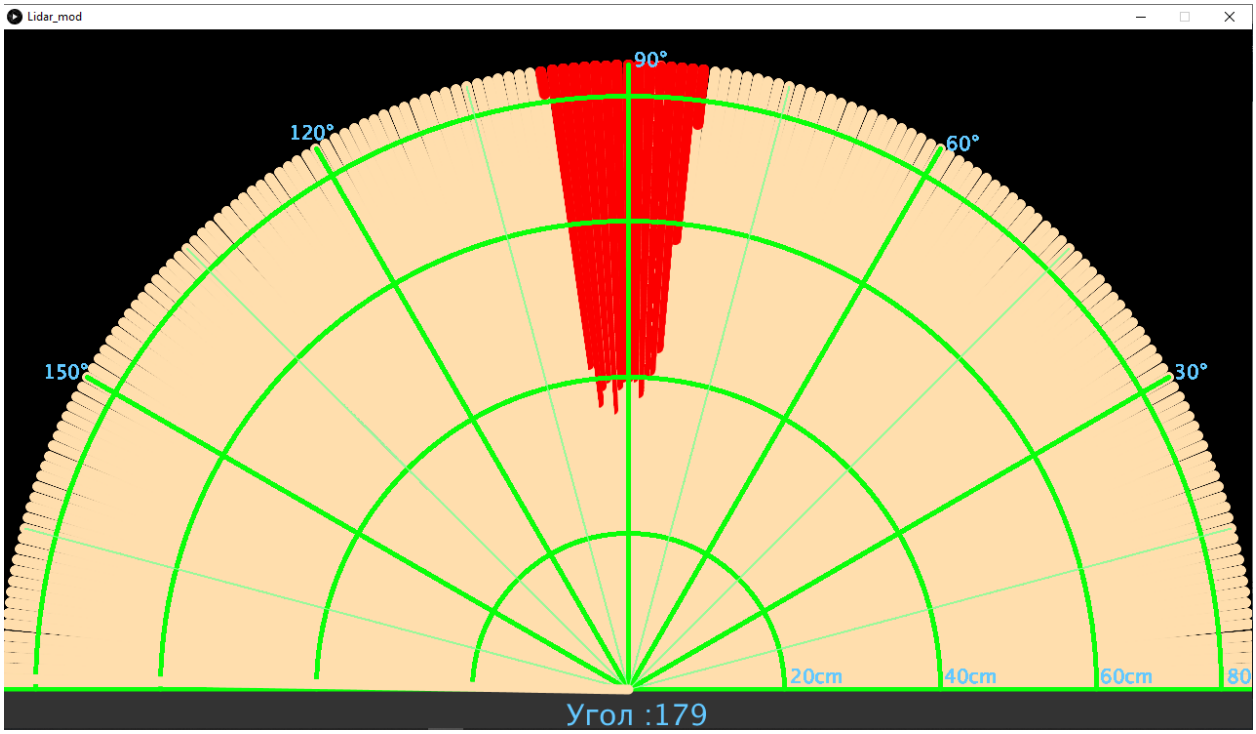


Рисунок 43 – Результаты сканирования при 11,1 оборотах в минуту

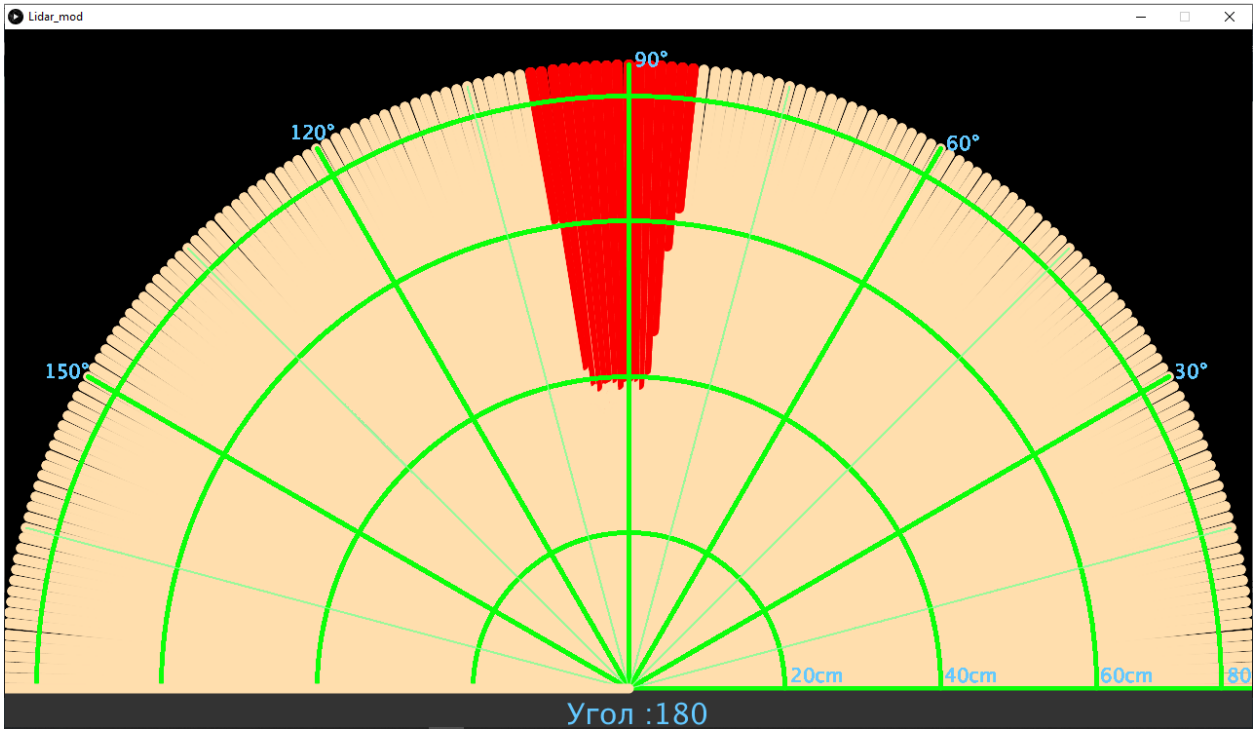


Рисунок 44 – Результаты сканирования при 6,7 оборотах в минуту

По результатам сканирования можно заявить, что если данный параметр будет слишком высоким, то датчик не будет успевать снимать показания по каждому градусу из-за чего у устройства будут возникать слепые зоны. Но чем дольше будут сниматься показания, тем точнее они будут. Так из рисунка 43 видно, что наличие слепых зон в рассматриваемом диапазоне отсутствует, но при этом снятые показания являются немного ошибочными. Глядя на рисунок 44, можно сказать что ещё большее уменьшение скорости вращения не столь сильно влияет на всю картину, хоть и ошибочные показания имеют меньшую погрешность.

Следующим экспериментом будет рассмотрение толщины объекта и их восприятие лидаром при разных расстояниях со скоростью вращения 11,1 оборотов в минуту. Для исследования взяты четыре объекта разных габаритов, расположенные в разных направлениях на всём диапазоне сканирования и расположены торцом к лидару, то есть развёрнуты шириной к сканированию. Расположение каждого объекта имеет свою собственную линию, по которой в дальнейшем будет отодвигаться на всё больше

расстояние от лидара. Размеры и расположение каждого объекта показаны в таблице 11.

Таблица 11 – Габариты и расположения объектов по разным углам обзора.

Номер объекта	Длина	Ширина	Высота	Линия расположения
1	30 мм	10 мм	70 мм	30 °
2	40 мм	15 мм	80 мм	60 °
3	40 мм	20 мм	120 мм	120 °
4	40 мм	40 мм	80 мм	150 °

Эксперимент состоит из 3 этапов. В каждом этапе все объекты будут располагаться равноудалённо от лидара. С каждым следующим этапом объекты будут отодвигаться на 20 сантиметров от лидара. Также в каждом этапе будут сниматься результаты сканирования при помощи программы визуализации. Более наглядное расположение всех объектов на разных этапах показано на рисунках 45-47.

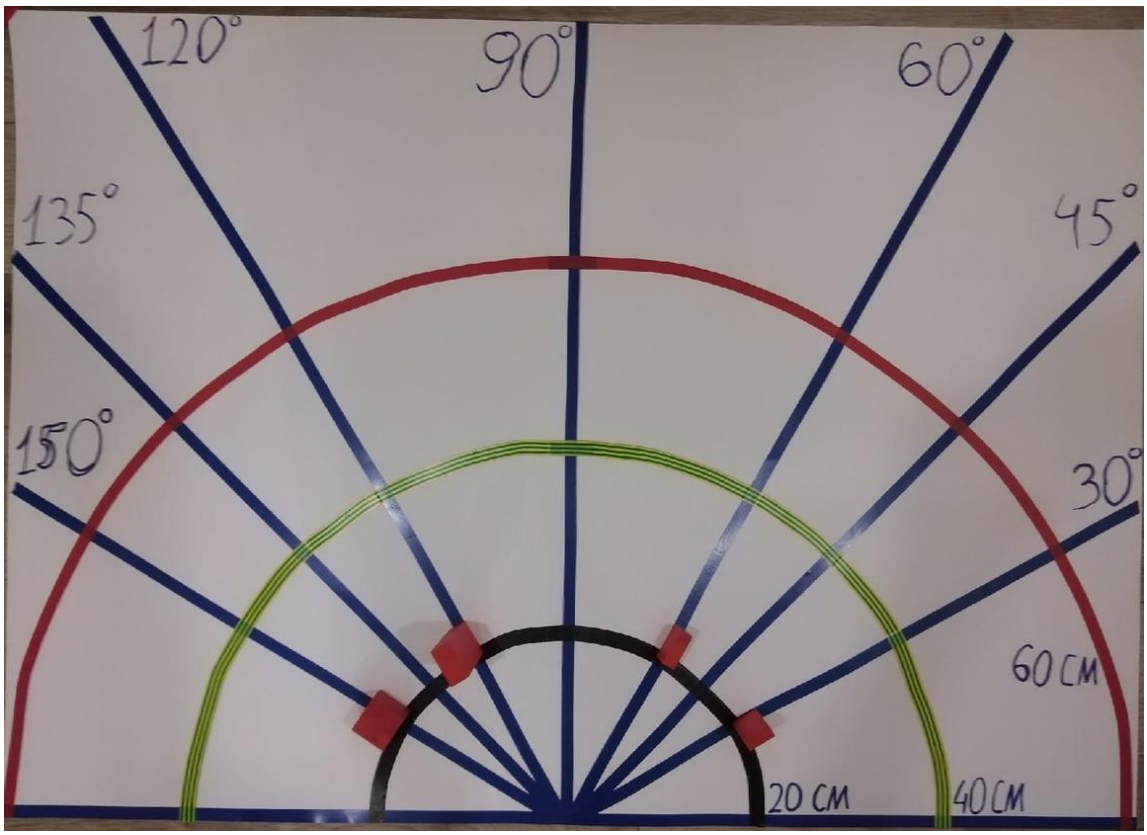


Рисунок 45 – Расположение объектов на первом этапе

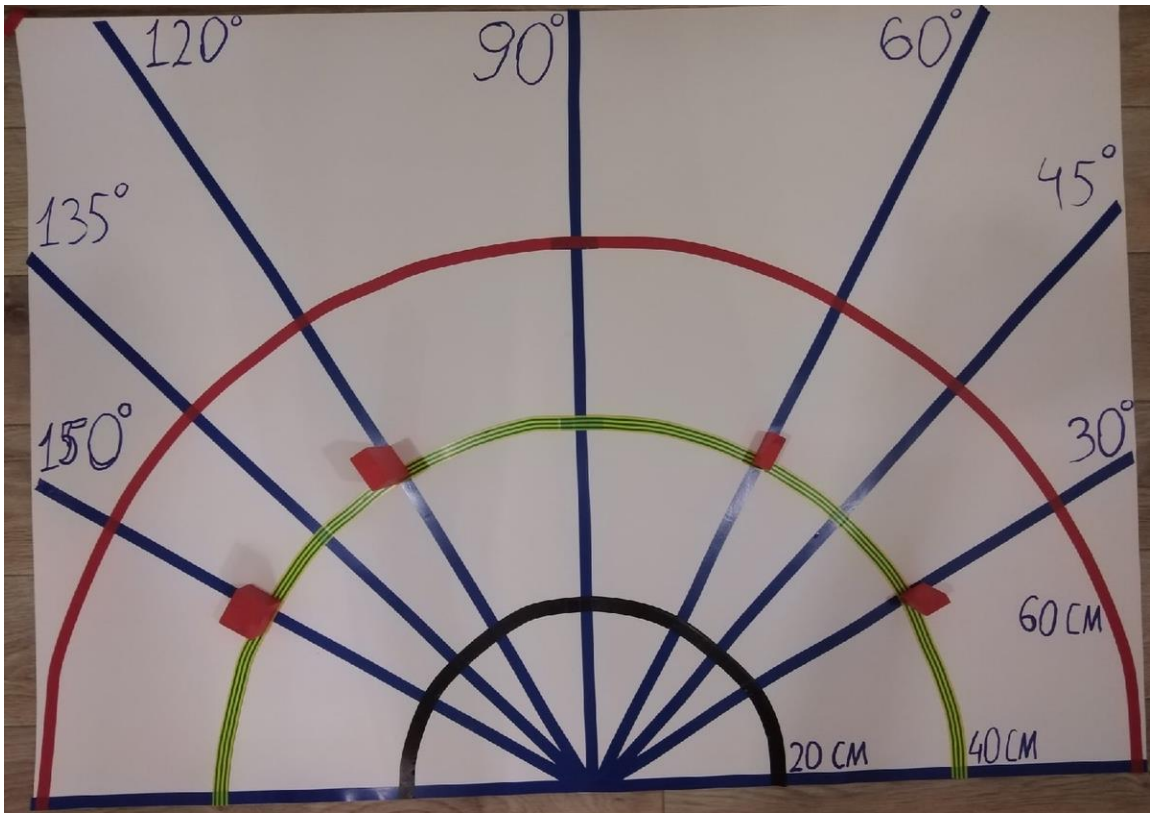


Рисунок 46 – Расположение объектов на втором этапе

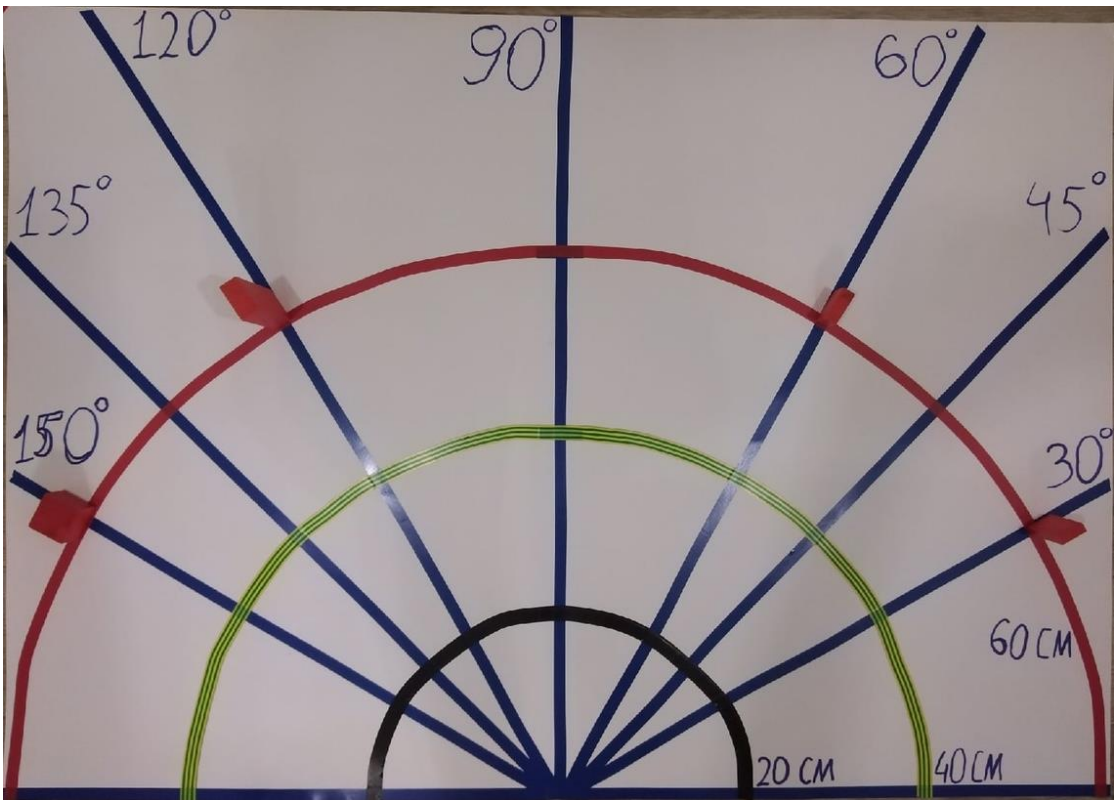
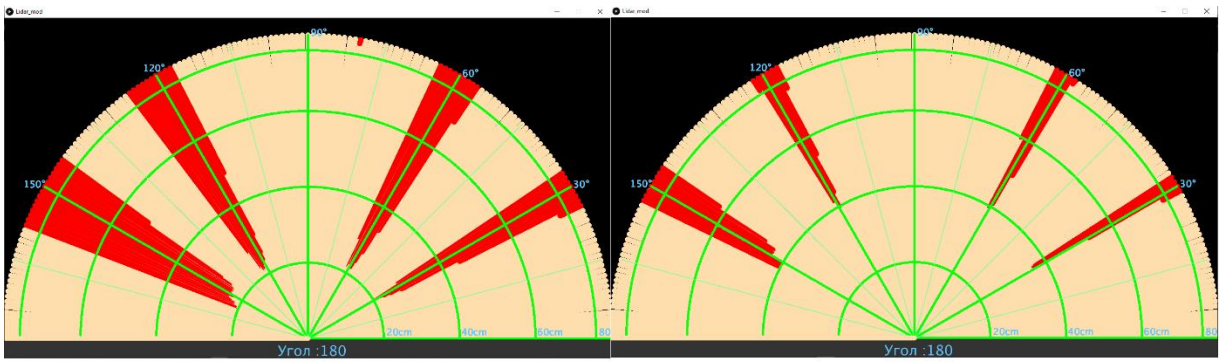
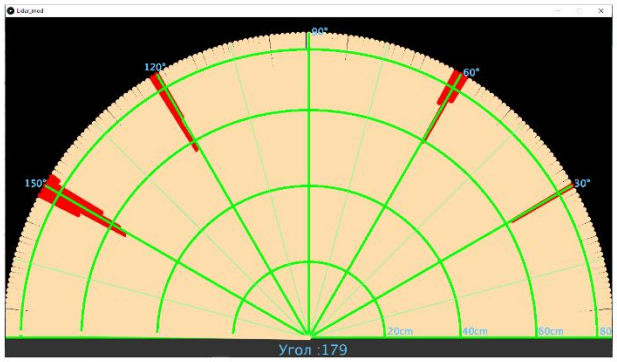


Рисунок 47 – Расположение объектов на третьем этапе

С каждого этапа были сняты результаты сканирования. На рисунке 48 можно увидеть эти самые результаты и подвести итог эксперимента. При скорости вращения равной 11,1 даже на расстоянии 60 сантиметров лидар обнаружил объект шириной в 10 миллиметров, что закрепляет выбор именно на этом значении. По результатам всех трёх этапов можно сказать, что чем дальше объект находится от лидара, тем сложнее измерить его габариты. Если на первом этапе хорошо видно увеличение каждого встречаемого объекта, то на третьем этапе уже сложно различить 2 и 3 объект. Также видно наличие появления ошибочных значений на третьем этапе



а) б)



в)

Рисунок 48 – Результаты сканирования: а) первого этапа;
б) второго этапа; в) третьего этапа.

5.2. Сборка беспилотного устройства

После всей проделанной работы по выбору комплектующих, написанию управляющей программы, разработке каркаса, а также отладки лидара, необходимо закупить все компоненты и распечатать разработанную конструкцию каркаса. Далее по составленной электрической схеме соединений подключить всё оборудование. После загрузить разработанный код и разместить все компоненты удобным для пользователя образом, при этом не нарушая принцип работы других устройств. Затем при помощи винтиков М3х10, гаек М3 и саморезов М3х10 укрепить всю конструкцию. Автоматизированная мобильная платформа с системой беспилотного движения, оснащённая лидаром, готова к использованию. Собранную платформу можно увидеть на рисунке 49.

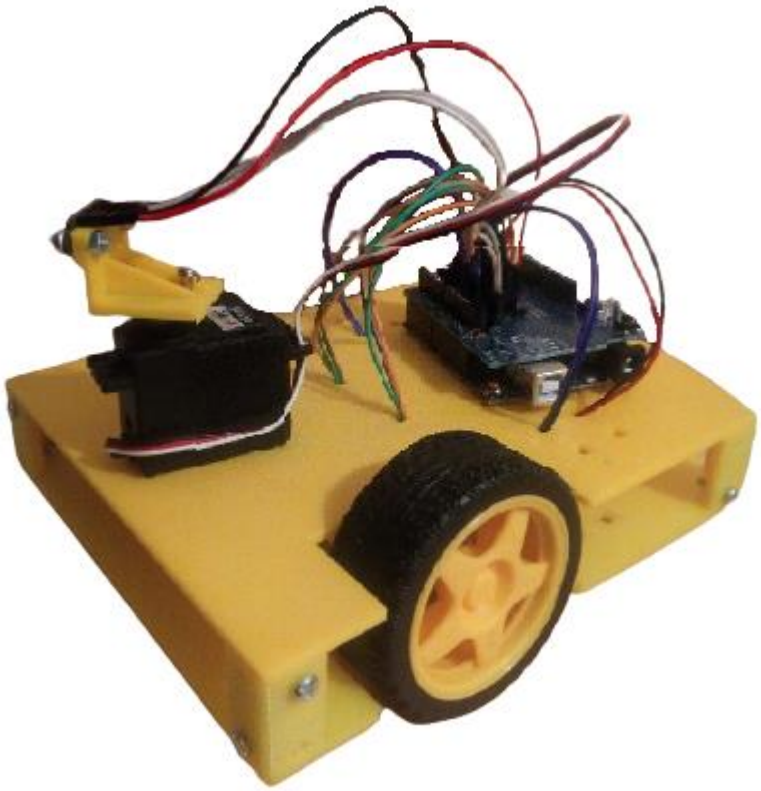


Рисунок 49 – Собранная платформа

Выводы по разделу

По окончанию данного раздела получилось на основании полученных результатов провести отладку лидара. Созданный плакат помог удостовериться в корректности работы. При помощи ранее созданной программы визуализации было проведено два экспериментальных исследования, по ходу которых принцип работы стал наглядно понятен. Также при сборке выбрано рациональное расположение лидара и прочих компонентов.

Заключение

В данной бакалаврской работе была разработана система беспилотного движения автомобиля в рамках аппаратно-программного комплекса мини-полигона. Работа данной системы рассчитана на поднятие навыков юных специалистов в сфере беспилотных транспортных средств, посредством имеющегося мобильного робота с устройством оптического сканирования области.

В процессе выполнения работы были рассмотрены все возможные аналоги комплектующих, способные заменить действующие при должных функциональных надобностях. Из рассмотренных комплектующих были выбраны наиболее подходящие с уклоном на низкую стоимость товаров. После чего на основании выбранных компонентов составлена электрическая схема соединений.

Была смоделирована и разработана конструкция каркаса, представляющая две скрепляемые детали, с выбором материала, удовлетворяющего условию свойственных характеристик. При моделировании детали учитывались все размеры комплектующих по подготовке пространства для их расположения и крепления.

В разделе программной части составлен алгоритм по работе системы в целом. После на основании алгоритма был написан программный код для загрузки его в микропроцессор. Небольшим бонусом стала программа с функцией графической модуляции принципа действия лидара.

Целью завершающего раздела являлась калибровка параметров лидара по его корректному функционированию внутри макета и за его пределами, а также разработка собственной площадки по тестированию устройства в домашних условиях. Благодаря этой площадке получилось наглядно разобраться в тонкостях работы устройства оптического сканирования и выбрать более продуктивное пространство по его установке на мобильной платформе.

В заключение хочется сказать, что все поставленные задачи по достижению главной цели, намеченные на защиту выпускной квалификационной работы, выполнены.

Список используемой литературы

1. Блок-схемы [Электронный ресурс]. URL: http://inf-w.ru/?page_id=551
2. ГОСТ 2.710-81 Единая система конструкторской документации (ЕСКД). Обозначения буквенно-цифровые в электрических схемах.
3. И.М. Чуркин. Материаловедение. Технология конструкционных материалов. ТГУ, Тольятти, 2011.
4. Каталог электронных компонентов Импульс [Электронный ресурс] URL: https://www.impulsi.ru/catalog/elektronnye_komponenty/
5. На работу за 20 минут: для москвичей — мечта, для жителей Тольятти — реальность [Электронный ресурс]. URL: https://www.superjob.ru/research/articles/111707/na-rabotu-za-20-minut/?utm_source=email&utm_medium=email&utm_campaign=mailing-list
6. Основной каталог chipdip. [Электронный ресурс] URL: https://www.chipdip.ru/
7. ПИД-регулятор. Методика настройки [Электронный ресурс]. URL: https://electric-blogger.ru/promyshlennoe/pid-regulyator-metodika-nastrojki.html
8. Пластики для 3D печати, всё что нужно знать о материалах [Электронный ресурс]. URL: https://3d-diy.ru/wiki/3d-printery/raznovidnosti-plastikov-3D-pechati/
9. Связываемся с Processing [Электронный ресурс]. URL: http://developer.alexanderklimov.ru/arduino/processing.php
10. Системы помощи водителю Audi [Электронный ресурс]. URL: https://www.audi-taganka.ru/ru/about-us/useful-information/sistemy-pomoshchi-voditelyu-audi.html
11. Сравнение плат Arduino [Электронный ресурс]. URL: https://ampermarket.kz/base/arduino_family/
12. Энкодеры: описание, технологии и виды [Электронный ресурс]. URL:

https://avi-solutions.com/library/statyi/enkodery-opisanie-tekhnologii-i-vidy/

13. Язык программирования Processing [Электронный ресурс]. URL: http://robotosha.ru/arduino/processing-programming-language.html
14. Arduino VL53L0X distance sensor [Электронный ресурс]. URL: https://electronoobs.com/eng_arduino_tut73.php
15. How Does Adaptive Cruise Control Work? [Электронный ресурс]. URL: https://theproctordealerships.com/how-does-adaptive-cruise-control-work/
16. How Radar Works [Электронный ресурс]. URL: http://www.bom.gov.au/australia/radar/about/what_is_radar.shtml
17. How to Use L298N Motor Driver [Электронный ресурс] URL: https://www.teachmemicro.com/use-l298n-motor-driver/
18. New LIDAR sensor can plot velocity as a fourth dimension [Электронный ресурс]. URL: https://www.qinetiq.com/en/blogs/new-lidar-sensor-can-plot-velocity-as-a-fourth-dimension
19. Self-Driving Cars: Understanding the 6 Autonomous Levels [Электронный ресурс]. URL: https://www.fool.com/investing/2018/09/06/self-driving-cars-understanding-6-autonomous-level.aspx
20. Self-paced learning for Fusion 360 [Электронный ресурс]. URL: https://help.autodesk.com/view/fusion360/ENU/courses/

Приложение А

Управляющая программа

```
#include <Servo.h>
#include <Wire.h>
#include <VL53L0X.h>

// Левый двигатель
#define MotorLeftForward    5  // IN4, Вперед
#define MotorLeftBackward   6  // IN3, Назад

// Правый двигатель
#define MotorRightForward   9  // IN1, Вперед
#define MotorRightBackward  10  // IN2, Назад

// Энкодеры
#define TACH_PIN_LEFT 2  // Левый энкодер
#define TACH_PIN_RIGHT 3 // Правый энкодер

// Серводвигаетель
#define ServoPin 11    // Пин подключения сервопривода

VL53L0X sensor;    // Создаем объект для сенсора
Servo ServoSensor; // Создаем объект для сервопривода

int SerialValue;   // Приходящее значение с блютуз-модуля
int WorkState = 0; // Рабочее состояние мобильной платформы
int Speed = 100;   // Начальная скорость мобильной платформы
int output = 0;    // Воздействие на колеса (хз как правильно, исправь)

int RotationTime = 270; //Время прохождения одного сектора, 90 градусов
int LeftWheel = 0;
int RightWheel = 0;
uint32_t ranges = 0;
uint32_t sectorLeft = 0;  // Переменная для левой области
uint32_t sectorFront = 0; // Переменная для передней области
uint32_t sectorRight = 0; // Переменная для правой области
uint32_t timer = millis();
int MaxSpd = 100;

// Вспомогательные переменные
const int Setpoint = -87;     // Поддерживаемая величина
const float dt = 0.1f;        // Время итерации в секундах
```

```
int LeftCounter = 0;          // Количество срабатываний энкодера для поворота налево (надо вычесть)
int RightCounter = 0;         // Количество срабатываний энкодера для поворота направо (надо вычесть)
int Counter = 0;              // Количество срабатываний энкодера
int detectState = 0;          // Переменная для чтения состояния энкодера
uint32_t input = 0;           // Переменная для входящего значения ПИД-регулятора
const float Collision = 0.0f; // Переменная для избегания столкновения

// Коэффициенты
const float Kp = 2f;  // Пропорциональный коэффициент
const float Ki = 0.5f;    // Интегральный коэффициент
const float Kd = 75f;    // Дифференциальный коэффициент

void setup()
{
  Serial.begin(9600); // Запускаем последовательный порт и задаём скорость обмена данными

  // Установление режима пинов на вывод данных
  pinMode(MotorLeftForward, OUTPUT);
  pinMode(MotorLeftBackward, OUTPUT);
  pinMode(MotorRightForward, OUTPUT);
  pinMode(MotorRightBackward, OUTPUT);

  // Установление уровня пина на низкий
  analogWrite(MotorLeftForward, LOW);
  analogWrite(MotorLeftBackward, LOW);
  analogWrite(MotorRightForward, LOW);
  analogWrite(MotorRightBackward, LOW);

  pinMode(TACH_PIN_LEFT , INPUT_PULLUP); // энкодер левый тянем к VCC
  pinMode(TACH_PIN_RIGHT, INPUT_PULLUP); // энкодер правый тянем к VCC
  attachInterrupt(digitalPinToInterrupt(TACH_PIN_LEFT), ISR_LEFT, FALLING);   // Подключаем прерывания для левого колеса
  attachInterrupt(digitalPinToInterrupt(TACH_PIN_RIGHT), ISR_RIGHT, FALLING); // Подключаем прерывания для правого колеса
```

```
 Wire.begin();
  ServoSensor.attach(servoPin);
  sensor.setTimeout(500);
  sensor.startContinuous();
}

// Функция прерывания для левого колеса
void ISR_LEFT()
{
  LeftWheel++;
}

// Функция прерывания для правого колеса
void ISR_RIGHT()
{
  RightWheel++;
}

// Функция ПИД-регулятора
void PID()
{
  float PrevError = 0.0f;    // Предыдущее значение ошибки
  static float integral = 0; // Интегральная составляющая

  float err = Setpoint - input;                              // Вычисление ошибки
  integral = constrain(integral + (float)err * dt * Ki, 0, 255); // Вычисление
интегральной составляющей
  float D = (err - PrevError) / dt;                          // Вычисление
дифференциальной составляющей
  PrevError = err;                                           // Присваиваем значение
нынешней ошибки предыдущей
  output = constrain(err * Kp + integral + D * Kd, 0, 255);      // Вычисление
входного сигнала
}


// Функция обработки данных лидара
void Lidar(int Range)
{
  signed int count = 0;
  ServoSensor.write(Range); // Поворочиваем сервопривод на прислаеное
значение
```

```
 timer = millis();
  while (!(millis() - timer > RotationTime)) {
    count++;
    ranges += sensor.readRangeContinuousMillimeters(); // Принимаем и
суммируем значение с датчика
  }
  float sector = ranges / count;
  switch (Range)
  {
    case 0: sectorLeft = sector; break;    // При получении 0 присваем значение
левой области
    case 59: sectorLeft = sector; break;   // При получении 59 присваем значение
левой области
    case 60: sectorFront = sector; break;  // При получении 60 присваем значение
передней области
    case 119: sectorFront = sector; break; // При получении 119 присваем
значение передней области
    case 120: sectorRight = sector; break;  // При получении 120 присваем
значение правой области
    case 180: sectorRight = sector; break; // При получении 180 присваем
значение правой области
  }
  input = sectorLeft - sectorRight; // Вычисляем значение входной перемнной
ПИД-регулятора
  count = ranges = 0; // Обнуляем значение переменных
}

// Функция остановки движения
void MoveStop()
{
  analogWrite(MotorRightForward, LOW);
  analogWrite(MotorRightBackward, LOW);
  analogWrite(MotorLeftForward, LOW);
  analogWrite(MotorLeftBackward, LOW);
}

// Функция движения вперед
void MoveForward()
{
  analogWrite(MotorRightForward, Speed);
  analogWrite(MotorRightBackward, LOW);
  analogWrite(MotorLeftForward, Speed);
```

```
  analogWrite(MotorLeftBackward, LOW);
}

// Функция избегания столкновений
void CollisionAvoidance()
{
  MoveStop();
  LeftWheel = RightWheel = 0; // Обнуляем значение переменных
  int Wheel = floor((LeftWheel + RightWheel) / 2); // Вычитаем среднее
значение с двух энкодеров и приводим к целому числу
  for (int i = Wheel; i < Counter; i++)
  {
    analogWrite(MotorRightForward, LOW);
    analogWrite(MotorRightBackward, Speed);
    analogWrite(MotorLeftForward, LOW);
    analogWrite(MotorLeftBackward, Speed);
  }
}

// Функция движения вперед с лидаром
void MoveForwardWithLidar()
{
  while (SerialValue == 1)
  {
    analogWrite(MotorRightForward, Speed - output);
    analogWrite(MotorRightBackward, LOW);
    analogWrite(MotorLeftForward, Speed + output);
    analogWrite(MotorLeftBackward, LOW);

    Lidar(59);  // Запуск функции лидара с передачей с проверкой левой
области
    Lidar(119);  // Запуск функции лидара с передачей с проверкой передней
области
    Lidar(180); // Запуск функции лидара с передачей с проверкой правой
области
    PID();      // Запуск функции ПИД-регулятора
    Lidar(120);  // Запуск функции лидара с передачей с проверкой правой
области
    Lidar(60);  // Запуск функции лидара с передачей с проверкой передней
области
    Lidar(0);   // Запуск функции лидара с передачей с проверкой левой области
    PID();      // Запуск функции ПИД-регулятора
```

```
    if (sectorFront < Collision)
    {
      CollisionAvoidance();
    }
  }
}

// Функция поворота направо
void MoveRight()
{
  LeftWheel = 0; // Обнуляем значение переменной
  for (int i = LeftWheel; i < RightCounter; i++)
  {
    analogWrite(MotorRightForward, LOW);
    analogWrite(MotorRightBackward, Speed);
    analogWrite(MotorLeftForward, Speed);
    analogWrite(MotorLeftBackward, LOW);
  }
  MoveForward(); // Запускаем функцию движения вперед 3 секунды
  delay(3000);
  MoveForwardWithLidar(); // Запускаем функцию движения вперед с лидаром
}

// Функция поворота налево
void MoveLeft()
{
  RightWheel = 0; // Обнуляем значение переменной
  for (int i = RightWheel; i < LeftCounter; i++)
  {
    analogWrite(MotorRightForward, Speed);
    analogWrite(MotorRightBackward, LOW);
    analogWrite(MotorLeftForward, LOW);
    analogWrite(MotorLeftBackward, Speed);
  }
  MoveForward(); // Запускаем функцию движения вперед 3 секунды
  delay(3000);
  MoveForwardWithLidar(); // Запускаем функцию движения вперед с лидаром
}

void loop()
{
  if (Serial.available()) // Проверка поданных команд
```

```
{
   SerialValue = Serial.read(); // Присваиваем переменой прочитаннного с порта значение

   switch (SerialValue)
   {
     case 0: ServoSensor.write(0); MoveForwardWithLidar();   // При получении 0 едим вперед с Лидаром
     case 1: MoveForward();           // При получении 1 едим вперед
     case 2: MoveRight();             // При получении 2 поворачиваем направо
     case 3: MoveLeft();              // При получении 3 поворачиваем налево
     case 4: MoveStop();              // При получении 4 мобильная платформа останавливается
   }
  }
}
```

Приложение Б

Программа визуализации работы лидара

```
import processing.serial.*;
Serial myPort;

String ang="";
String distance="";
String data="";

int angle, dist;

void setup() {
size (1280, 720);
myPort = new Serial(this,"COM5", 9600);
myPort.bufferUntil('.');
background(0);
}

void draw() {
fill(0,4);
noStroke();
rect(0, 0, width, height*0.94);

noStroke();
fill(51,255);
rect(0,height*0.94,width,height);

drawLidar();
drawLine();
drawObject();
drawText();
}


void serialEvent (Serial myPort) {
data = myPort.readStringUntil('.');
data = data.substring(0,data.length()-1);

int index1 = data.indexOf(",");
ang= data.substring(0, index1);
distance= data.substring(index1+1, data.length());

angle = int(ang);
```

```
dist = int(distance);
}

void drawLidar(){
pushMatrix();
noFill();
stroke(10,255,10);
strokeWeight(4);

translate(width/2,height-height*0.06);

line(-width/2,0,width/2,0);

arc(0,0,(width*0.25),(width*0.25),PI,TWO_PI);
arc(0,0,(width*0.50),(width*0.50),PI,TWO_PI);
arc(0,0,(width*0.75),(width*0.75),PI,TWO_PI);
arc(0,0,(width*0.95),(width*0.95),PI,TWO_PI);

line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));

stroke(152,251,152);
strokeWeight(1);
line(0,0,(-width/2)*cos(radians(15)),(-width/2)*sin(radians(15)));
line(0,0,(-width/2)*cos(radians(45)),(-width/2)*sin(radians(45)));
line(0,0,(-width/2)*cos(radians(75)),(-width/2)*sin(radians(75)));
line(0,0,(-width/2)*cos(radians(105)),(-width/2)*sin(radians(105)));
line(0,0,(-width/2)*cos(radians(135)),(-width/2)*sin(radians(135)));
line(0,0,(-width/2)*cos(radians(165)),(-width/2)*sin(radians(165)));

popMatrix();
}

void drawLine(){

pushMatrix();

strokeWeight(10);
stroke(255,222,173);
```

```
translate(width/2,height-height*0.06);

line(0,0,(width/2)*cos(radians(angle)),(-width/2)*sin(radians(angle)));

popMatrix();

}

void drawObject(){

pushMatrix();

strokeWeight(11);
stroke(252,0,0);
translate(width/2,height-height*0.06);

float pixleDist = (dist/80.0)*(width/2.0);
float pd=(width/2)-pixleDist;

float x=-pixleDist*cos(radians(angle));
float y=-pixleDist*sin(radians(angle));

if(dist<=80){
line(-x,y,-x+(pd*cos(radians(angle))),y-(pd*sin(radians(angle))));
}
popMatrix();
}

void drawText(){
pushMatrix();

fill(100,200,255);
textSize(20);

text("20cm",(width/2)+(width*0.130),height*0.93);
text("40cm",(width/2)+(width*0.253),height*0.93);
text("60cm",(width/2)+(width*0.378),height*0.93);
text("80cm",(width/2)+(width*0.48),height*0.93);

textSize(30);
text("Угол :"+angle,width*0.45,height*0.99);
```

```
if(dist<=80) {
text("Расстояние :"+dist,width*0.7,height*0.99);
}

translate(width/2,height-height*0.06);
textSize(20);

text(" 30°",(width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
text(" 60°",(width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
text(" 90°",(width/2)*cos(radians(90)),(-width/2)*sin(radians(91)));
text("120°",(width/2)*cos(radians(123)),(-width/2)*sin(radians(118)));
text("150°",(width/2)*cos(radians(160)),(-width/2)*sin(radians(150)));

popMatrix();

}
```