Региональный оператор детского технопарка «Кванториум»
Государственное автономное образовательное учреждение дополнительного профессионального образования Владимирской области «Владимирский институт развития образования имени Л.И. Новиковой», детский технопарк «Кванториум-33»

ИНЖЕНЕРНАЯ КНИГА

Соревнования: **«РобоФест-2019» - AutoNet 18+**

Название команды: **«IMMO»**

Номер команды: **A18-06**

Разработала команда:
Пономарев В.Г.
Россолов Р.А.
Немировский Д.Ю.
Лысков Р.А.
Полиевцев В.Г.

Владимир, 2019

Содержание

Глава 1. Концепция

1.1 Концепция соревнований

Задача соревнований: робот команды-участника должен проехать от старта до парковки по игровому полю, имитирующему городскую транспортную систему.

Игровое поле:

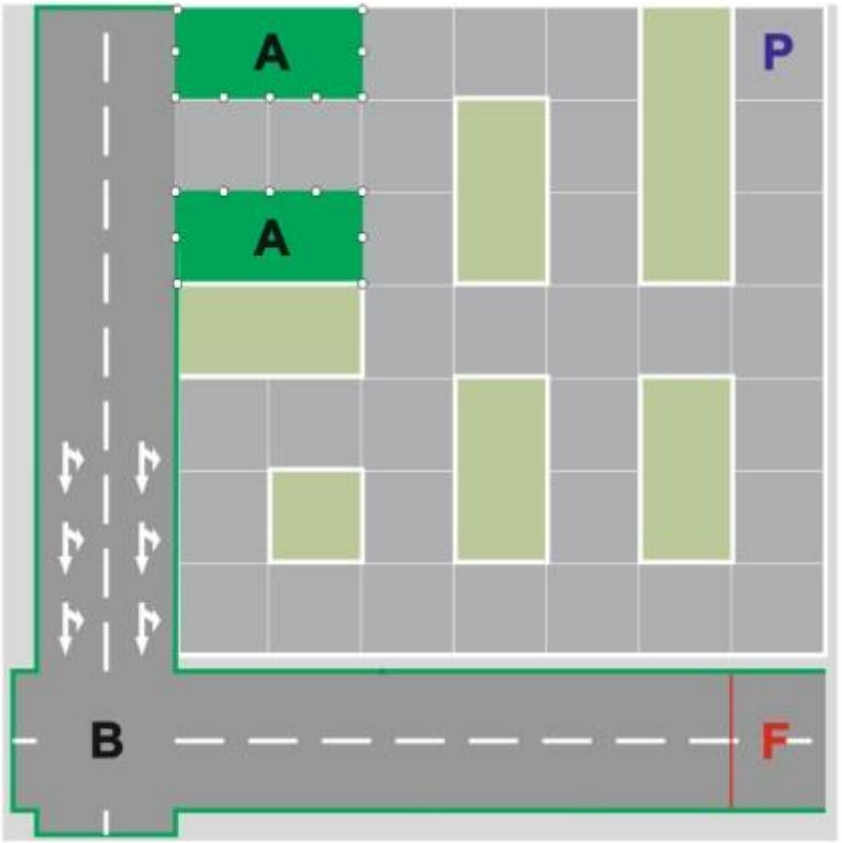

Рисунок 1. Игровое поле.

P – зона старта;
Участок P-A – зона городских кварталов. Имеет ограждение. Робот должен самостоятельно выбирать направление движения, выполняя, при необходимости, требования светофора и знаков дорожного движения.
A – площадь. Ограждения на площади выполнены в виде столбиков с цепочками. Газоны площади могут содержать различные декоративные элементы, в том числе и схожие цветом со знаками;
B – перекресток на трассе;
Участок A-F – зона скоростного движения (трасса). Не имеет ограждения. На трассе возможно появление препятствий (имитация транспортных средств);

F – финиш.

Используемые знаки дорожного движения:

- только прямо;
- только направо;
- только налево;
- прямо или направо;
- прямо или налево;
- движение запрещено (кирпич).

Рисунок 2. Используемые знаки дорожного движения.

Знаки устанавливаются справа по ходу движения робота. Нижняя граница знака находится на высоте 70 см.

Перед светофором располагается СТОП-линия, (поперечная белая линия) шириной не менее 5 см. Робот обязан остановиться перед СТОП-линией. Робот может продолжить движение только после включения зеленого сигнала светофора.

На игровом поле возможно появление препятствий, имитирующих транспортные средства. Препятствия являются статическими и имеют размер по ширине не более 50 см.

Определение победителя:

- победитель определяется по сумме баллов лучшего заезда, Инженерной книги, и подтверждения компетенций;
- оргкомитет может назначать дополнительные номинации за инженерную книгу.

Характеристики игрового поля:

Ориентировочные размеры – 18 х 18 метров.

Покрытие игрового поля – баннерная ткань или линолеум.

Кварталы представляют собой непрозрачные конструкции. Возможно наличие угловых стоек.

Ширина проезжей части в городской зоне – не менее 2,0 м.

Требования к роботу:

Максимальный размер робота для участия в матчах – 1500 мм в длину, 1000 мм в ширину и 700 мм в высоту.

Робот должен соответствовать автомобильной кинематике, «Автомобильной» считается кинематика, содержащая два управляемых (поворотных) колеса на одной оси и два колеса на другой оси (не могут быть поворотными).

1.2 Концепция проекта

Целью проекта является разработка робота с автомобильной кинематикой, способного в полностью автономном режиме перемещаться с соблюдением правил дорожного движения в условиях, приближенным к городским. Соревнования предусматривают решение роботом задач движения с учетом элементов дорожной разметки, правильного реагирования на знаки светофора и дорожного движения, планирования траектории собственного движения из зоны старта в зону финиша. При этом расположение некоторых дорожных знаков и светофора заранее неизвестно.

Для достижения указанной цели необходимо решить следующие задачи:

- разработать конструкцию робота с автомобильной кинематикой;
- разработать 3D-модели робота;
- разработать программное обеспечение, позволяющее в полностью автономном режиме перемещаться с соблюдением правил дорожного движения в условиях, приближенным к городским;
- провести отладку и испытания;
- разработать техническую документацию (инженерную книгу).

Глава 2. Конструкция

Конструкция автомобиля состоит из следующих компонентов:

- платформа для крепления основных компонентов;
- узел рулевого управления;
- узел крепления электродвигателя;
- диск оптического датчика количества оборотов колес;
- платформа крепления лазерного дальномера;
- кузов автомобиля.

Ниже на рисунке приведена сборка основных компонентов на платформе.

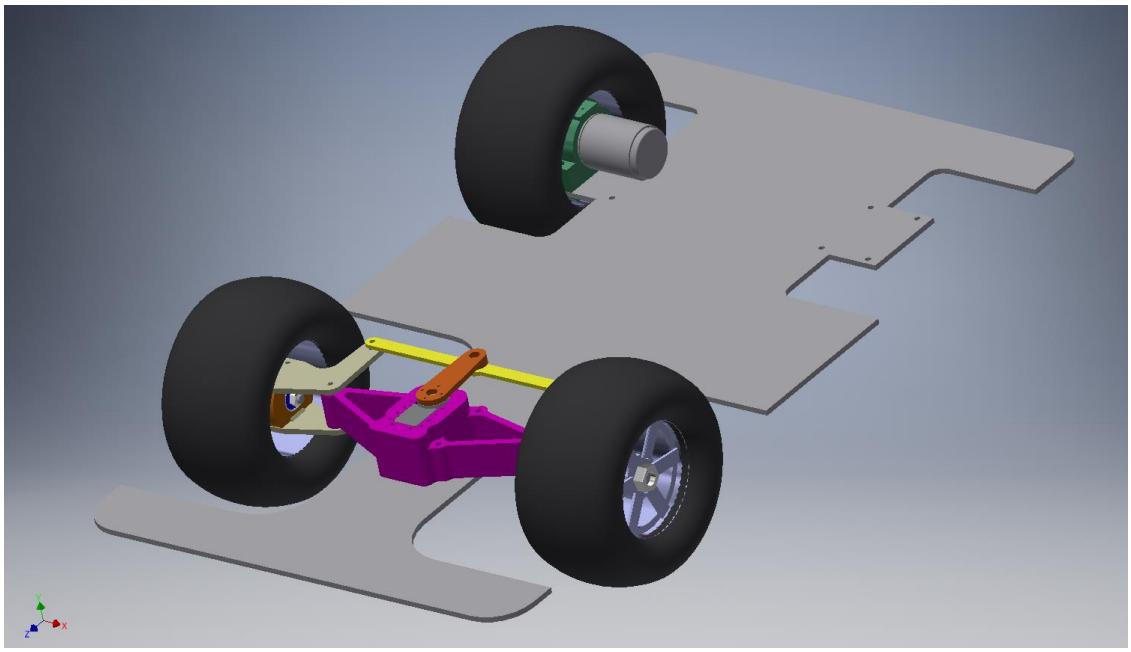

Рисунок 3. Модель основных компонентов автомобиля.

Платформа для крепления изготавливается из композитного материала или фанеры толщиной 3 мм. На передней части платформы устанавливается узел рулевого управления, закрепляемый с помощью болтового соединения. На задней части платформы закрепляются электромоторы с дисками для подсчета количества оборотов (энкодерами).

На платформе так же располагаются контроллеры управления низкого и высокого уровней, лазерный дальномер. Контроллер управление низкого уровня располагается в задней части автомобиля между электромоторами, а контроллер высокого уровня устанавливается в центральной части автомобиля. Контроллеры закрепляются на платформе с помощью алюминиевых или медных стоек посредством болтового соединения.

Аккумулятор располагается в задней части автомобильной платформы и фиксируется посредством текстильной застежки выполненной виде хомута.

Ниже на рисунке представлена задняя часть платформы с установленным контроллером низкого уровня, электромоторами и аккумулятором.

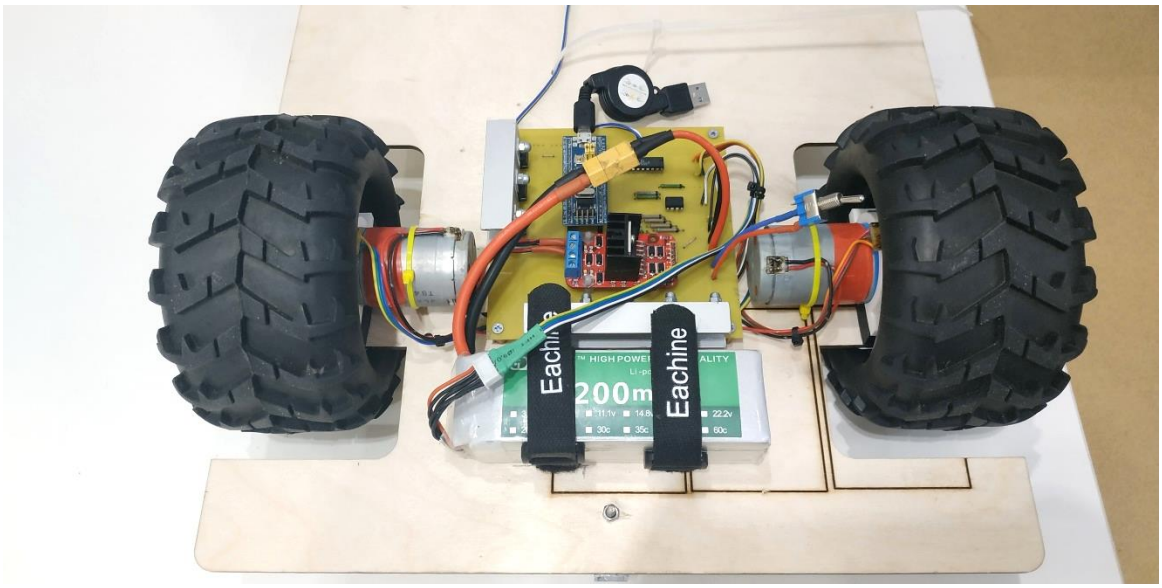

Рисунок 4. Задняя часть автомобильной платформы.

На платформе в передней и задней части автомобиля так же располагаются платы, представляющие собой фонари для световой индикации. Ниже на рисунке приведена система индикации.

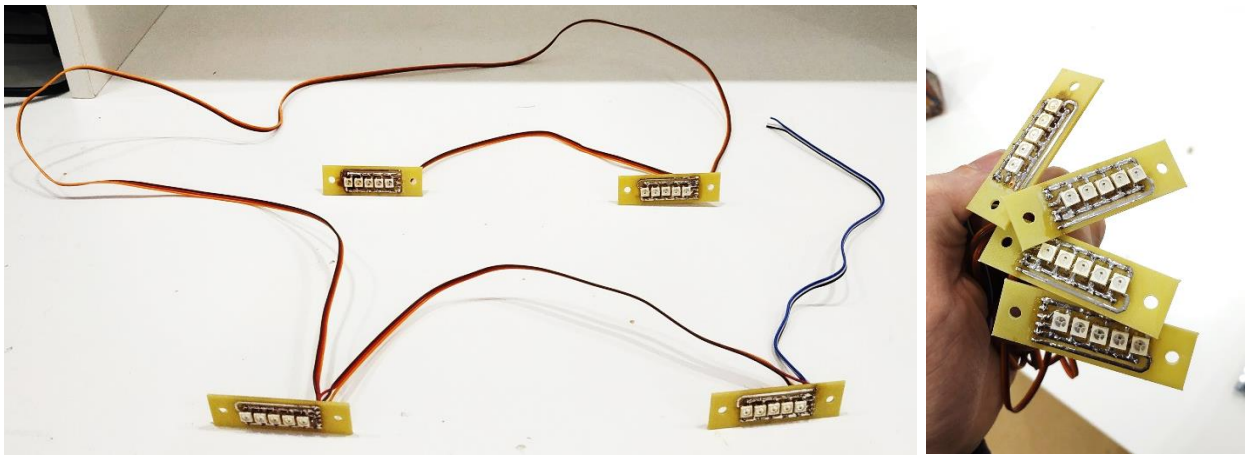

Рисунок 5. Световая индикация.

Расстояние между передней и задней осью автомобиля составляет 333 мм. Длинна автомобиля составляет 640 мм, ширина 380 мм.

Узел рулевого управления изготовлен из PLA пластика, в конструкции опоры имеется отсек для установки сервомотора. К сервомотору крепится

рулевая сошка, которая приводит в движение рулевую тягу. Рулевая тяга приводит в движение поворотные рычаги. Поворотные рычаги, рулевая тяга и рулевая сошка крепятся посредством болтового соединения с использованием подшипников скольжения. Сборка узла представлена на рисунке ниже.

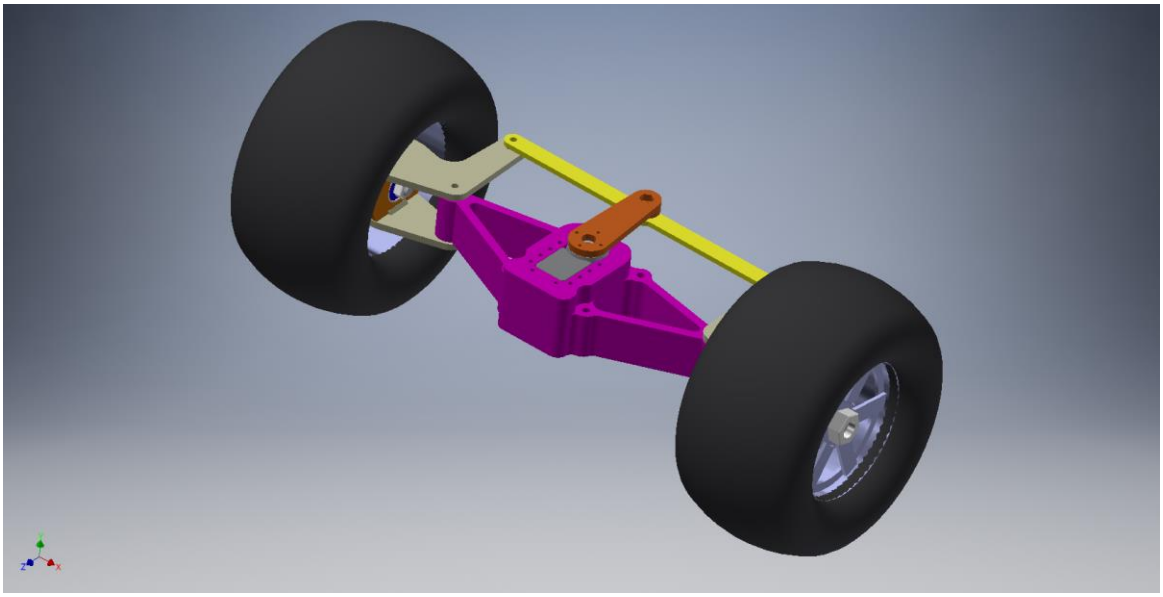

Рисунок 6. Модель узла рулевого управления.

Рисунок 7. Узел рулевого управления на платформе.

Колеса состоят из пластмассового диска, резиновой шины и поролоновой набивки. Внешний диаметр колес равен 141 мм.

Диск для оптического датчика подсчета количества оборотов устанавливается на вал электромотора внутри (непосредственно между электродвигателем и колесным диском). Диск имеет 86 отверстий позволяющие изменять состояния оптического датчика. Диаметр диска составляет 57 мм. Толщина диска в посадочном месте на вал составляет 4,5

мм. Для исключения вращения диска на валу посадочное место имеет шлиц, соответствующий лыске на валу. Ниже представлена модель диска.

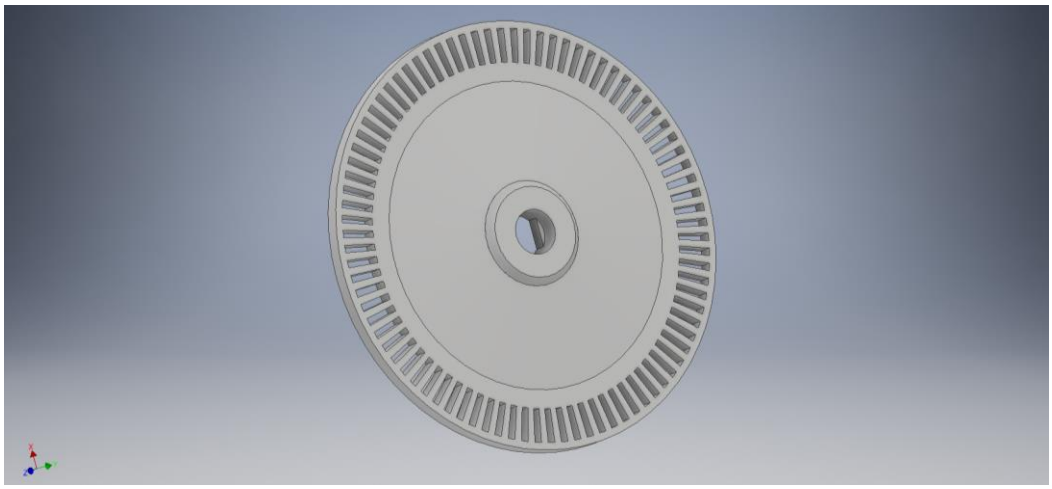

Рисунок 8. Модель диска для оптического датчика количества оборотов колес.

Крепление электродвигателей состоит из 3-х частей, две части которого устанавливаются сверху платформы являются хомутом, удерживающим электродвигатель за редукторную часть и имеющую место крепления платы оптического датчика. На нижней части располагается элемент реализующий функцию защиты диска оптического датчика количества оборотов колес. Крепление изготовлены из PLA пластика. Ниже на рисунке представлены три части кронштейна крепления электродвигателя и оптического датчика.

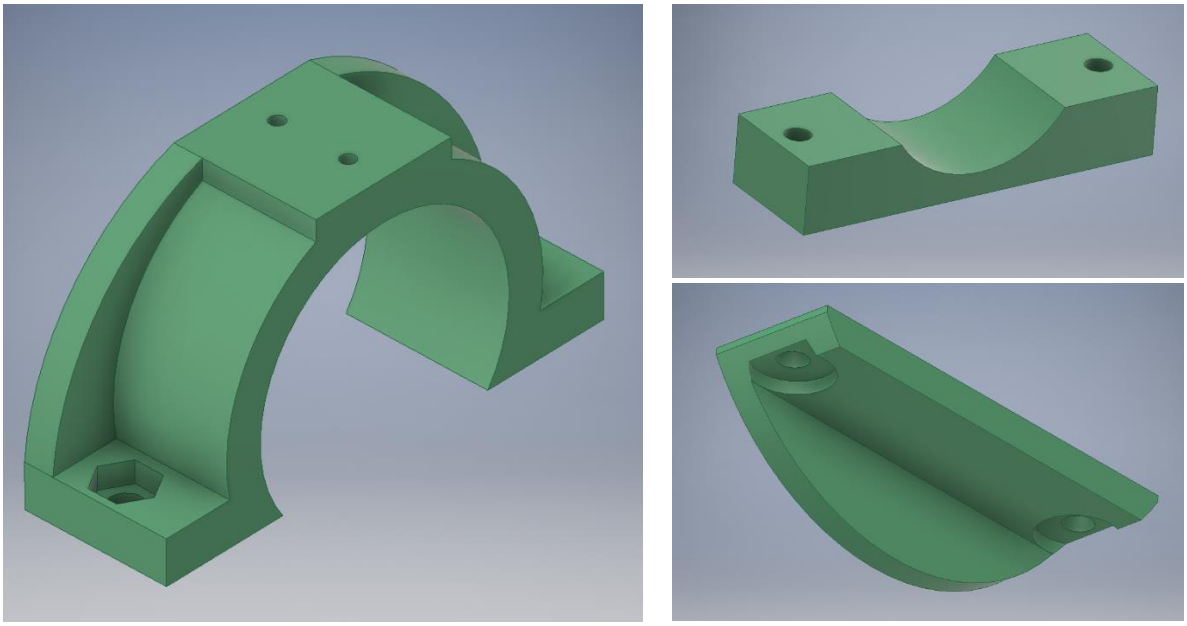

Рисунок 9. Модели кронштейна крепления электромотора и оптического датчика количества оборотов колеса.

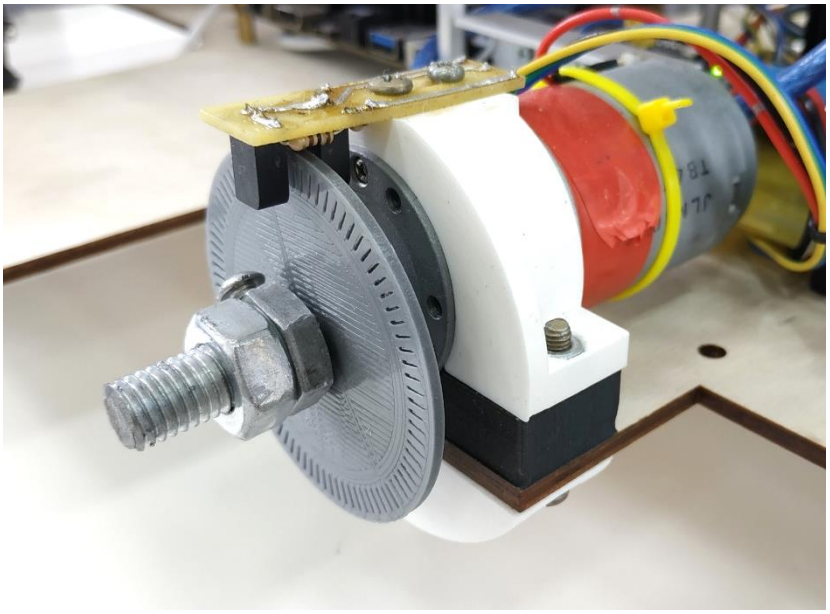

Рисунок 10. Электродвигатель с оптическим датчиком количества оборотов колеса.

Крепление лазерного дальномера представляет собой платформу, располагающуюся на алюминиевых стойках над платой контроллера управления низкого уровня. Фиксация производится посредством болтового соединения. На платформе предусмотрено крепление с использованием болтового соединения для преобразователя USB2LDS. А также отверстия для фиксирования электропроводки нейлоновыми стяжками.

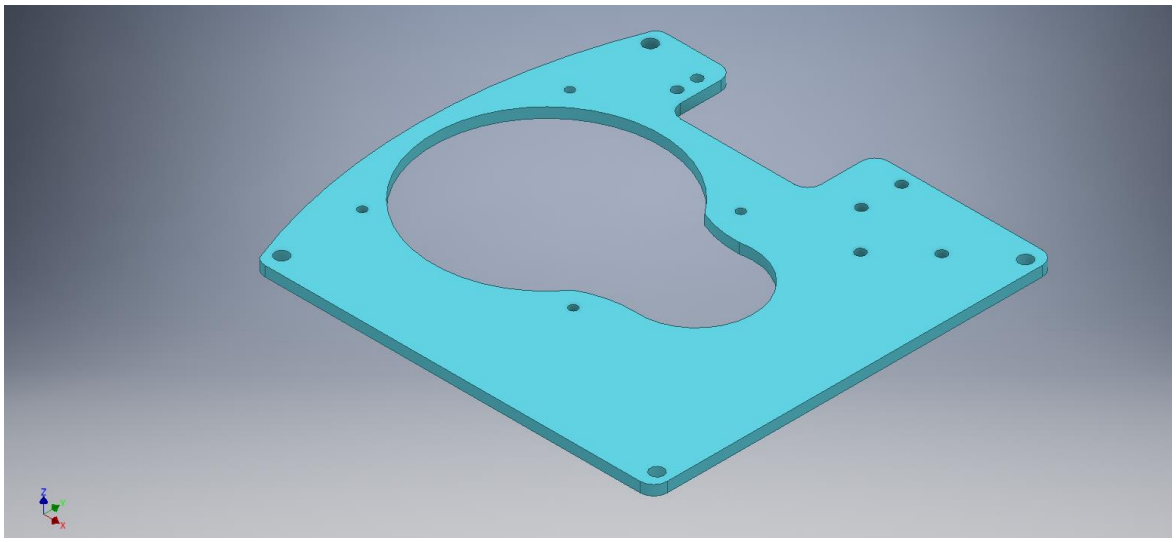

Рисунок 11. Платформа крепления лазерного дальномера.

Рама в процессе отладки навигации эксплуатировалась без кузова.

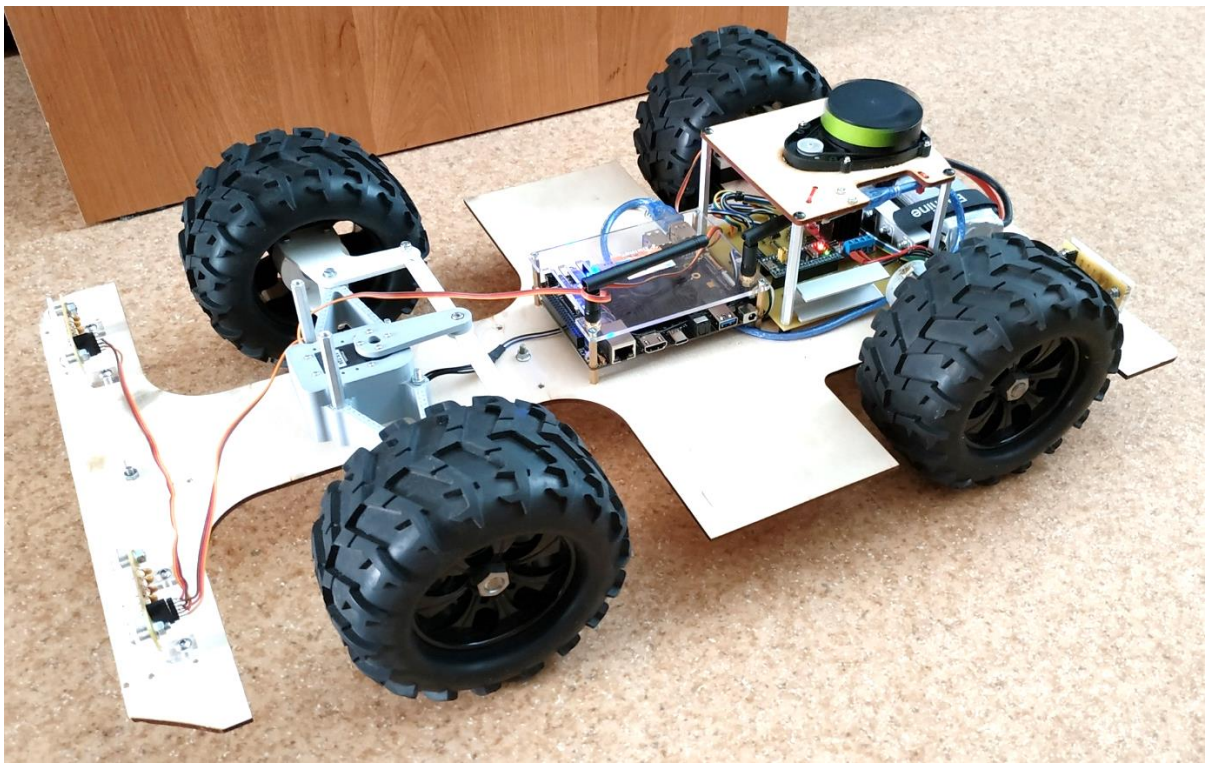

Рисунок 12. Рама на стадии процесса отладки.

Кузов автомобиля выполнен из современных материалов с применением передовых технологий конструирования. Следует отметить, что при создании экстерьера автомобиля был применен стиль Hi-Tack. В кузове предусмотрены отверстия для размещения двух камер и лазерного дальномера.

Первая камера устанавливается в передней части автомобиля и направлена по ходу движения. Вторая камера размещена на платформе, закрепленной на металлическом стержне и направленна по ходу движения.

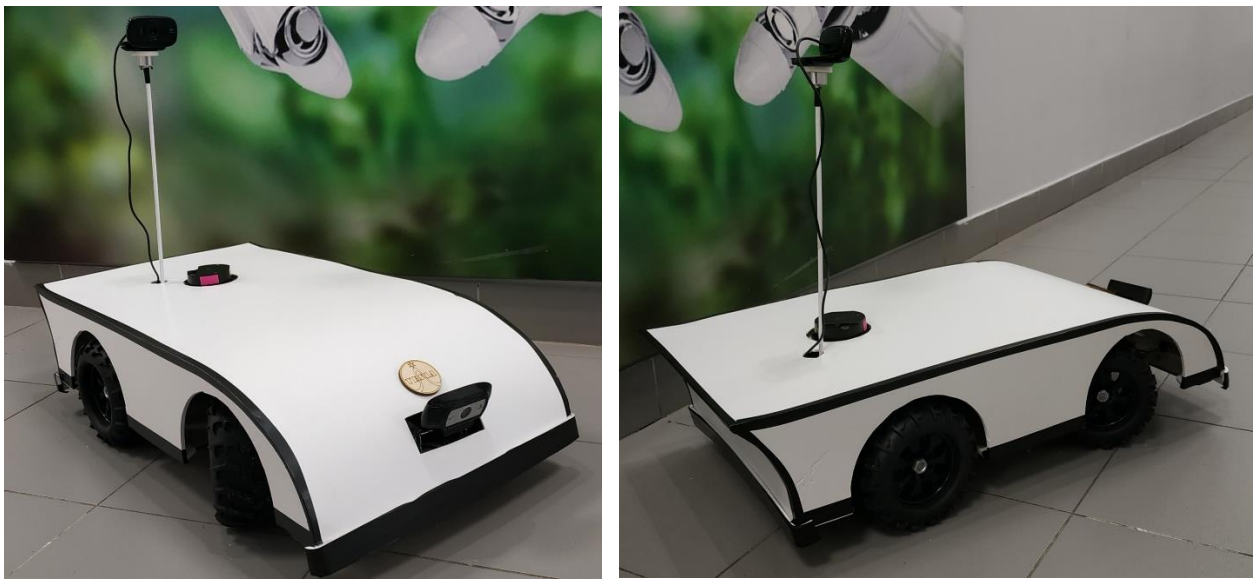

Рисунок 13. Автомобиль с кузовом.

Глава 3. Электроника

3.1 Система питания

Для автономного передвижения и питания всех компонентов транспортного средства используется аккумулятор с техническими характеристиками, приведенными в таблице ниже.

Таблица 1. Технические характеристики аккумулятора.

Наименование	Значение
Тип	LiPo
Внутренняя организация	4S
Выходное напряжение	16,8 В
Ток	30C
Ёмкость	5200 мАч
Тип разъема	XT60
Наличие балансового разъема	Есть

Получение необходимого стабилизированного напряжение 5 и 12 вольтовых линий выполнено с использованием стабилизаторов напряжения LM7805 и LM7812.

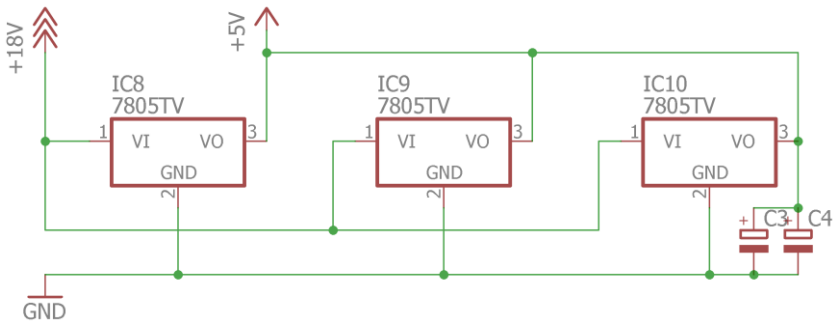

Рисунок 14. Электрическая схема системы питания 5В.

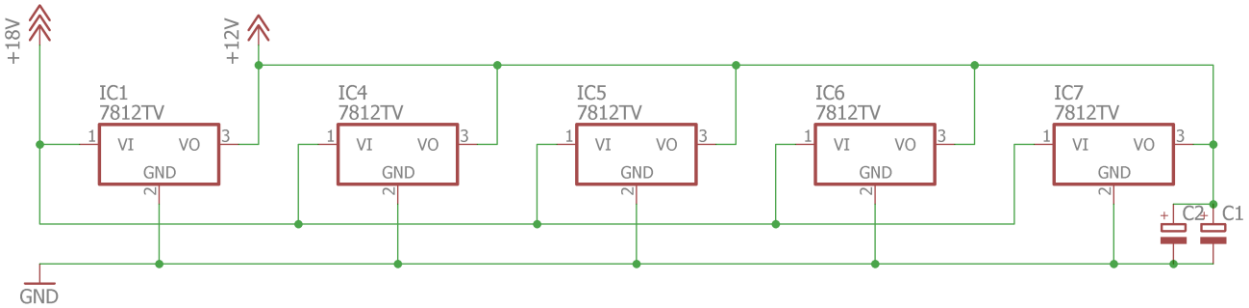

Рисунок 15. Электрическая схема системы питания 12В.

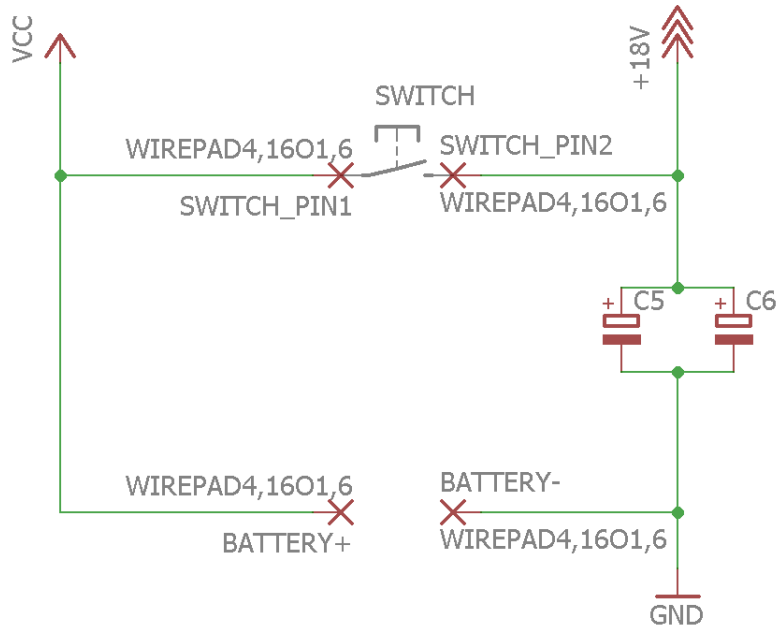

Рисунок 16. Электрическая схема подключения аккумулятора и выключателя системы питания.

3.2 Система управления низкого уровня (драйвер шасси)

Контроллером управления является Arduino STM32F103C8T6. Плата имеет выводы для подключения следующих устройств: ультразвуковые датчики, датчики числа оборотов колес, датчики направления движения, сервомотор рулевого управления, световая индикация, датчик заряда аккумулятора, драйвер управления электромоторами постоянного тока, подключение питания аккумулятора и место для подключения кнопки включения питания. Ниже на рисунке приведено обозначение выводов платы.

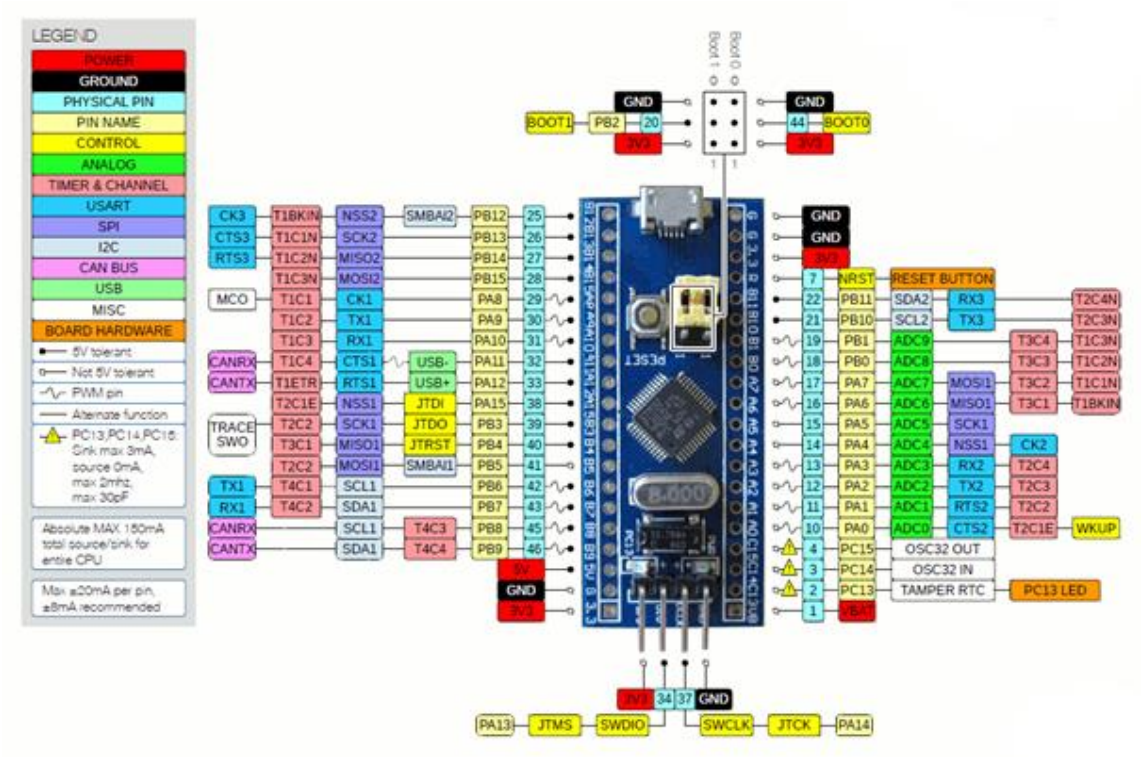

Рисунок 17. Обозначение выводов контроллера STM32F103C8T6.

Электрическая схема контроллера приведена на рисунке ниже.

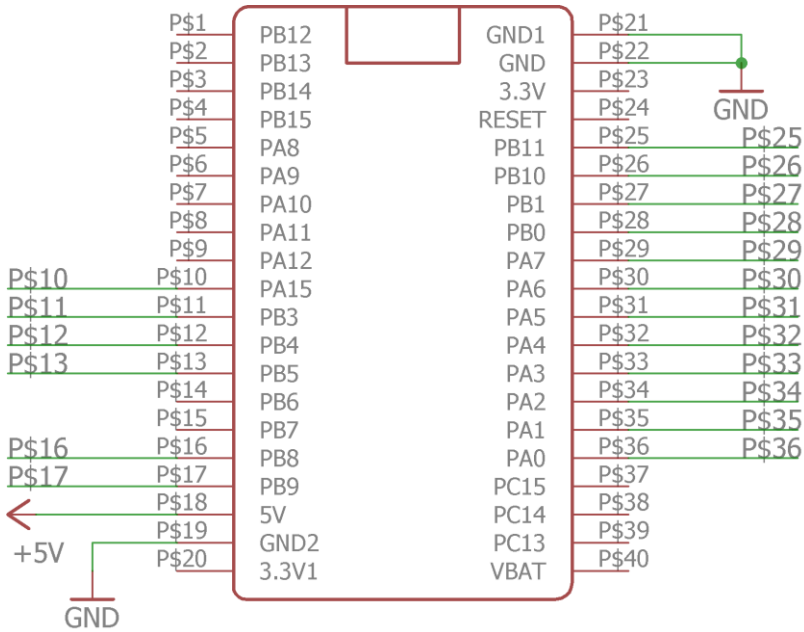

Рисунок 18. Электрическая схема контроллера.

Печатная плата имеет крепежные отверстия для крепления на шасси ТС. На рисунке ниже приведена печатная плата драйвера шасси.

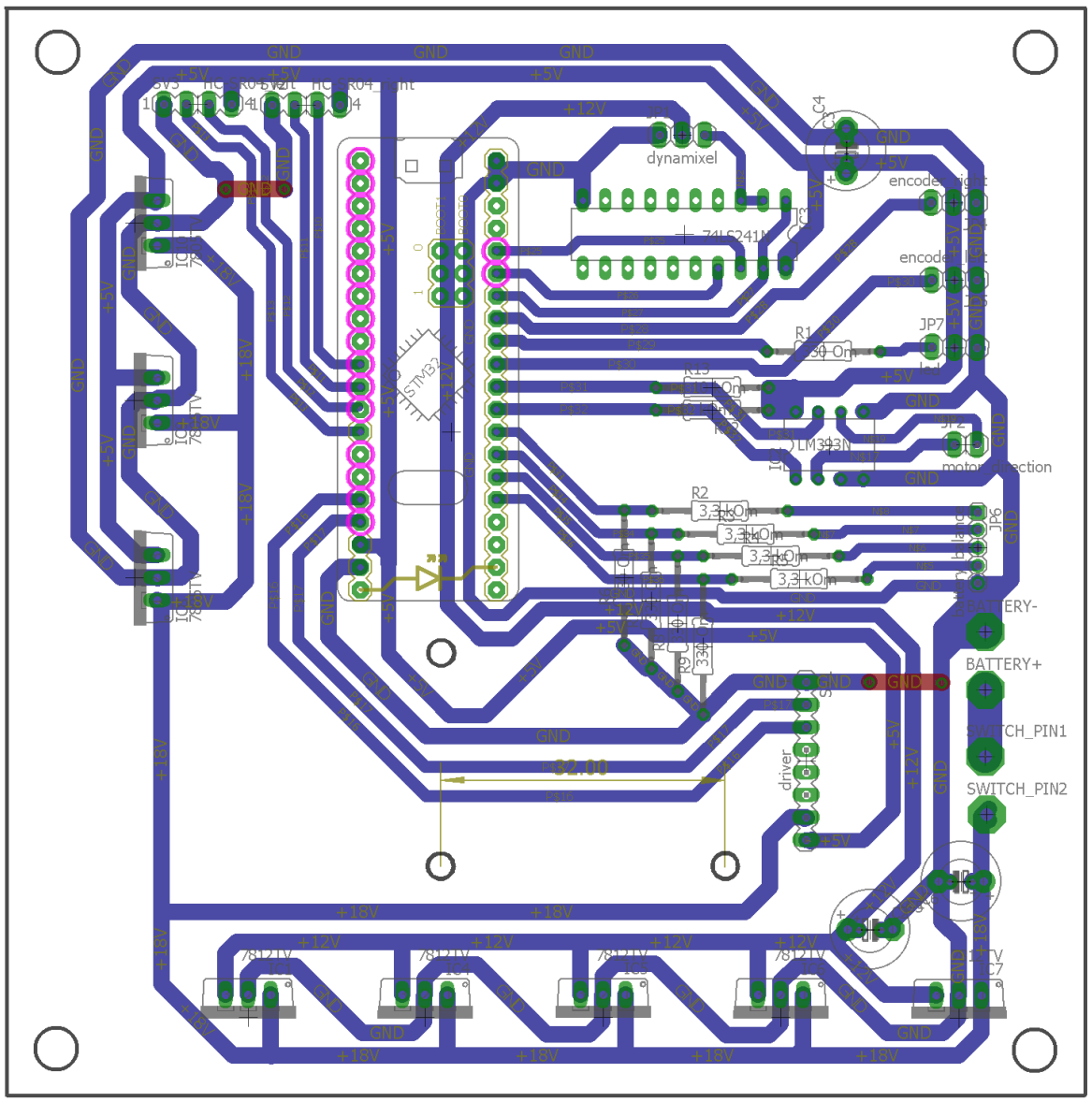

Рисунок 19. Печатная плата драйвера шасси.

3.2.1 Оптический датчик числа оборотов колеса

Для определения количества оборотов колеса используются бесконтактные оптические датчики положения. Они состоят из оптического излучателя и фотоприемника. Световой поток от излучателя попадает на фотоприемник, что вызывает определенное состояние датчика. Наличие непрозрачного объекта на пути светового луча приводит к изменению светового потока на фотоприемнике, а значит и к другому состоянию датчика.

Непрозрачным объектом является смоделированное образцовый диск с отверстиями, устанавливаемый на вал мотора (во внутренней части колеса).

Контроллер реагирует на прерывания при изменении состояния фотоприёмника с (HIGH на LOW, LOW на HIGH), что позволяет повысить точность в два раза.

Для щелевого оптрона и необходимой для него обвязки изготовлена печатная плата. Печатная плата приведена на рисунке ниже.

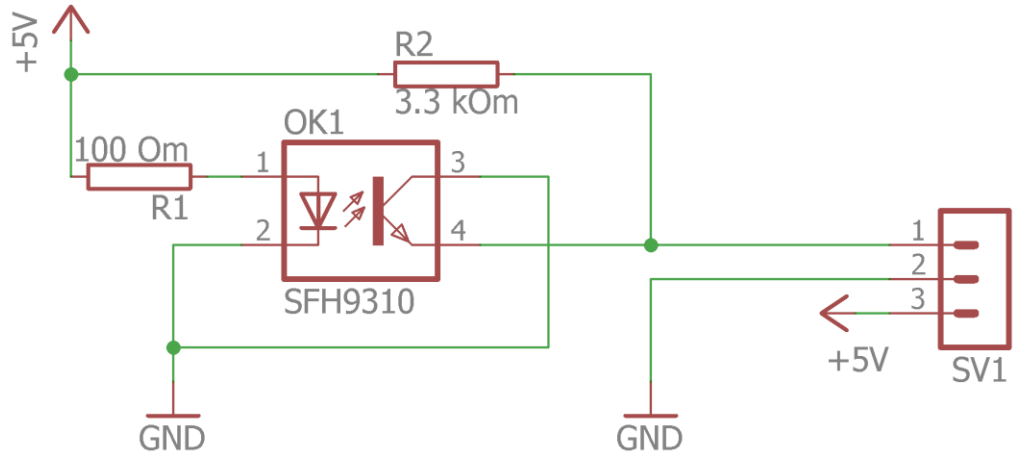

Рисунок 20. Электрическая схема оптического датчика числа оборотов.

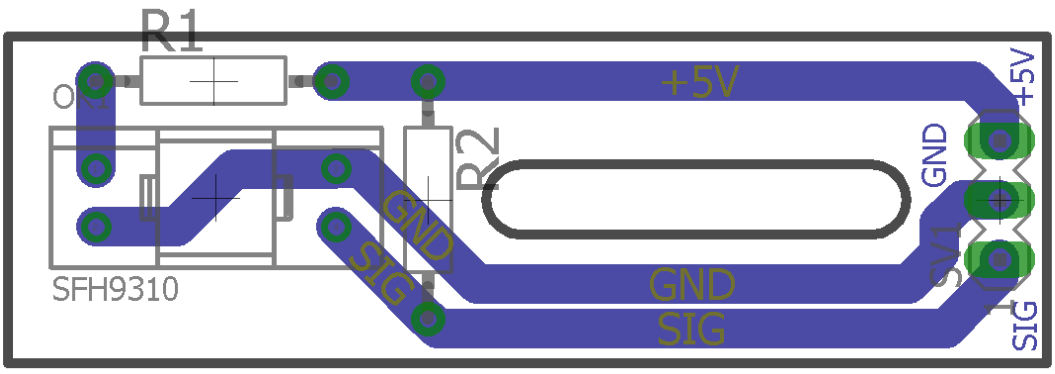

Рисунок 21. Печатная плата оптического датчика числа оборотов колеса.

Таблица 2. Описание выводов оптического датчика числа оборотов колеса.

Обозначение	Наименование
+5V	положительный контакт питания 5В
GND	отрицательный контакт питания
SIG	выход с датчика

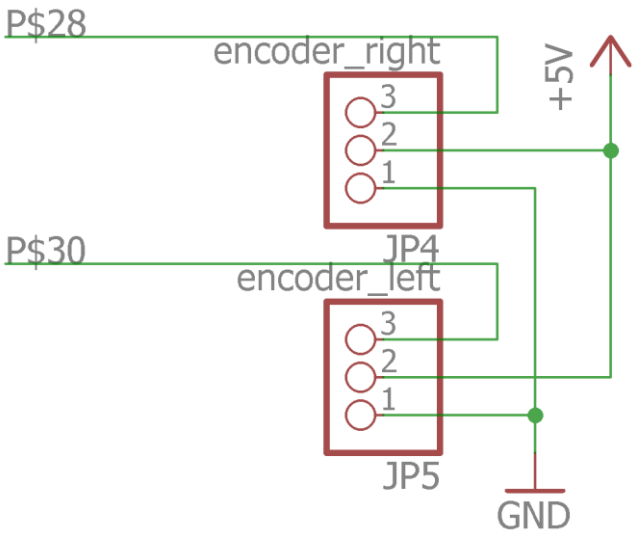

Рисунок 22. Схема подключения датчика числа оборотов колес к контроллеру.

3.2.2 Драйвер управления двигателями постоянного тока

Для управления электродвигателями постоянного тока используется драйвер L298N. Технические характеристики приведены в таблице ниже.

Таблица 3. Технические характеристики драйвера

Наименование	Значение
Размер платы	30 мм x 54 мм
Питание платы	5 В
Питание двигателей	2.5 – 46 В
Рабочий ток на канал	до 2А
Логический "0" управляющего напряжения	0...1.5 В
Логическая "1" управляющего напряжения	2,3...7 В
Максимальная частота управляющего ШИМ	до 5 кГц
Защита от перегрева	Есть

L298N содержит сразу два драйвера для управления электродвигателями (четыре независимых канала, объединенных в две пары). Имеет две пары входов для управляющих сигналов и две пары выходов для подключения электромоторов. Кроме того, у L298N есть два входа для включения каждого из драйверов. Эти входы используются для управления скоростью вращения электромоторов с помощью широтно-импульсной модуляции сигнала (ШИМ).

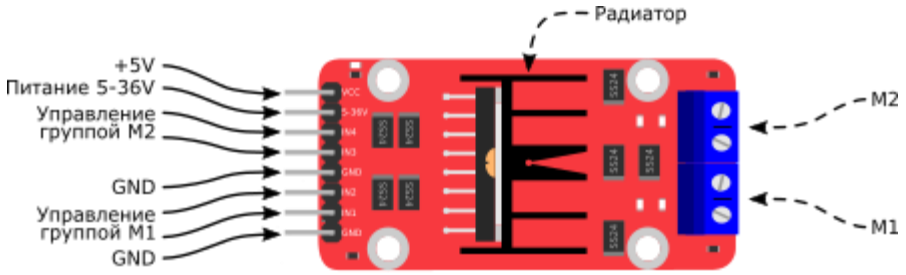

Рисунок 23. Драйвер управления электродвигателями постоянного тока L298N.

Таблица 4. Описание выводов драйвера управления электродвигателями постоянного тока

Обозначение	Наименование
VCC	положительный контакт питания платы (управляющей логики) 5В
5-36V	положительный контакт питания электродвигателей 5 – 36 В
GND	отрицательный контакт питания
IN1, IN2, IN3, IN4	управление направлением вращения и скоростью двигателей

L298N обеспечивает разделение электропитания для контроллера и для управляемых им двигателей, что позволяет подключить электродвигатели с большим напряжением питания чем у контроллера. Разделение электропитания микросхем и электродвигателей может быть также необходимо для уменьшения помех, вызванных бросками напряжения, связанными с работой моторов.

Одна микросхема L298N способна управлять двумя двигателями по 2А каждый двигатель, а если задействовать параллельное включение для одного двигателя, то можно поднять максимальный ток до 4А.

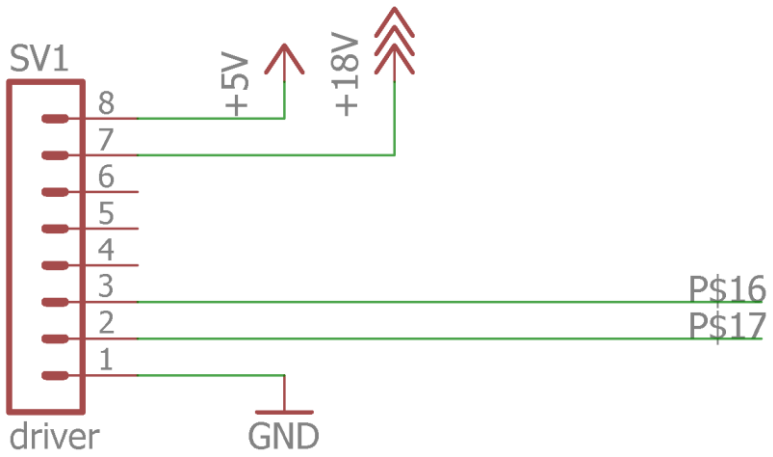

Рисунок 24. Схема подключения драйвера управления электро-двигателями постоянного тока к контроллеру.

3.2.3 Рулевое управление

Для рулевого управления используется сервомотор dynamixel AX-12a.

Таблица 5. Основные характеристики dynamixel AX-12a.

Наименование	Значение
Рабочее напряжение	9-12 В (рекомендуемое 11,1 В)
Передаточное число	254:1
Момент удержания	1,5 Н*м (12 В)
Ток удержания	1,5 А
Скорость без нагрузки	59 об/мин (12 В)
Минимальный угол поворота	0,29 градуса х 1,024
Угол поворота	в режиме актуатора - до 300 градусов, в режиме двигателя - без ограничений
Максимальный ток	900 мА
Ток покоя	50 мА
Управление	цифровое
Протокол	полудуплексный асинхронный последовательный (8N1)
Скорость интерфейса	1Мбод/с
ID	0...253 (по умолчанию ID = 1)
Обратная связь	положение, температура, нагрузка, напряжение и т.п.
Датчик положения	потенциометр
Вес	54,6 г

Размеры	32 мм x 50 мм x 40 мм

Контроллером управления сервомотором является Arduino STM32F103C8T6. Промежуточным звеном между сервомотором и контроллером является цифровая микросхема серии ТТЛ (К555АП4). Микросхемы К555АП4 представляют собой два четырехканальных формирователя с тремя состояниями на выходе. Содержат 232 интегральных элемента. Корпус типа 2140.20-1, масса не более 3,6 г.

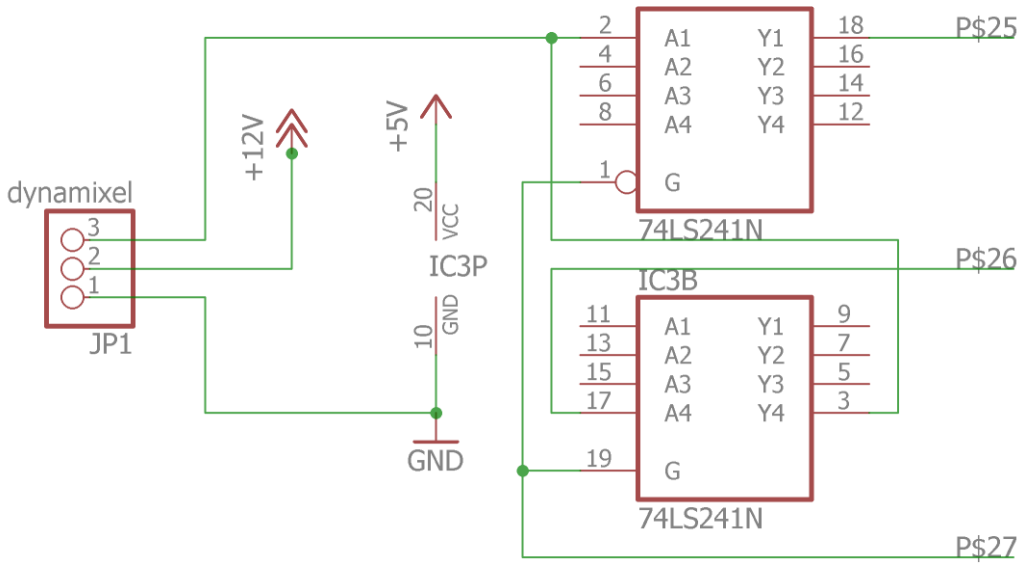

Рисунок 25. Схема подключения Микросхемы К555АП4.

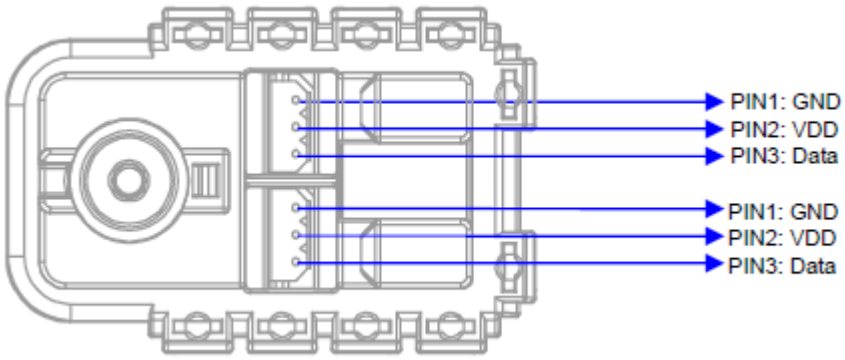

Рисунок 26. Схема распиновки сервомотора dynamixel AX-12a.

Таблица 6. Описание выводов платы световой индикации (фонаря).

Обозначение	Наименование
VDD	положительный контакт питания 9-12 В
GND	отрицательный контакт питания
Data	цифровое управление, приём и передача данных

3.2.4 Датчик напряжения аккумулятора

Датчик напряжения аккумулятора реализован путем понижения напряжения на каждом элементе балансировочного разъема (за исключением отрицательного провода) и подключению к аналоговым входам микроконтроллера. Так же в схеме присутствуют подтягтвающие резисторы к GND.

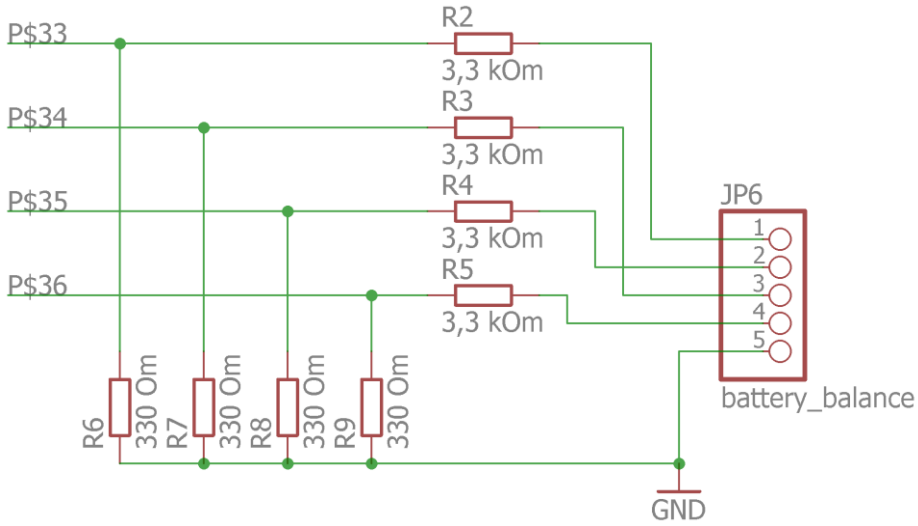

Рисунок 27. Схема датчика заряда аккумулятора к контроллеру.

3.2.5 Датчик определения направления движения

Для корректного расчёта одометрии необходимо знать направление движения транспортного средства. При движении на выходах драйвера управления электродвигателями постоянного тока L298N, предназначенных для подключения моторов появляется положительное и отрицательное питание, которое меняется при изменении направления вращения колеса.

Для преобразования сигнала к дискретному с уровнем 5В с драйвера используется LM393P, Двойной маломощный компаратор напряжения.

Таблица 7. Технические характеристики LM393P.

Наименование	Значение
Количество каналов	2
Напряжение питания	4 – 30В
Время задержки	300 нс
Ток потребления	1 мА
Температурный диапазон	0 – 70 °C
Тип корпуса	dip8
Напряжение компенсации	5 мВ

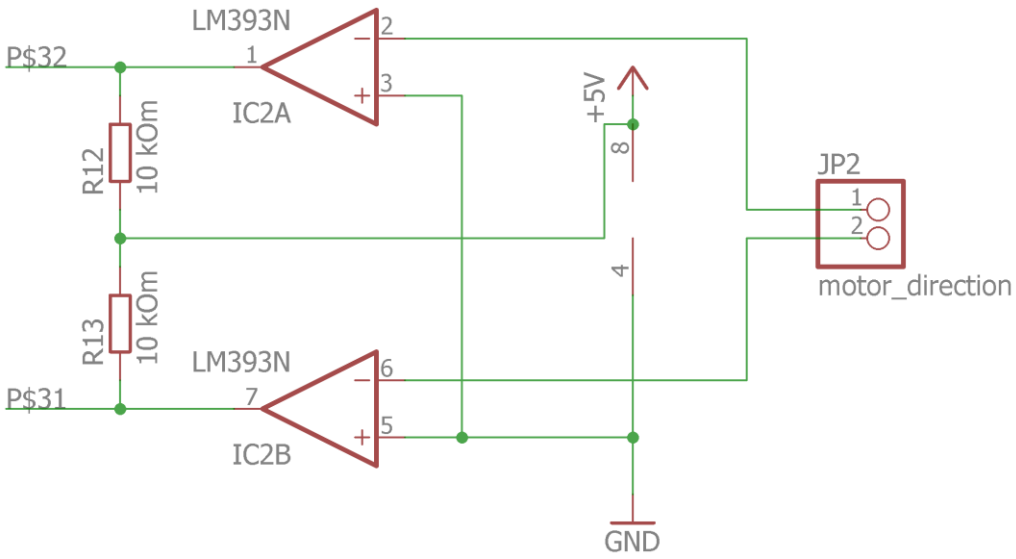

Рисунок 28. Схема подключения микросхемы LM393P к контроллеру

3.2.6 Ультразвуковой дальномер

Для определения расстояния до препятствий относительно передней части автомобиля установлены два ультразвуковых дальномера HC-SR04.

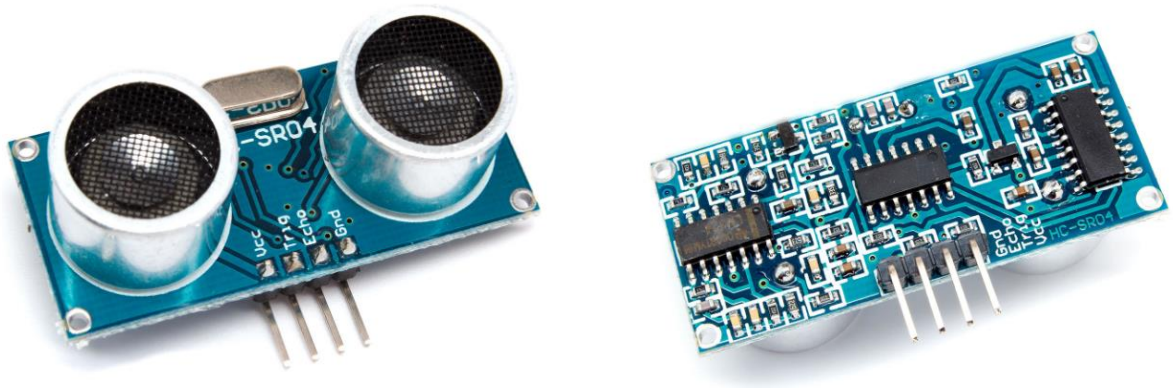

Рисунок 29. Ультразвуковой дальномер HC-SR04.

Таблица 8. Технические характеристики ультразвукового дальномера.

Наименование	Значение
Напряжение питания	5 В
Потребление в режиме тишины	2 мА
Потребление при работе	15 мА
Диапазон расстояний	2 – 400 см
Эффективный угол наблюдения	15°
Рабочий угол наблюдения	30°

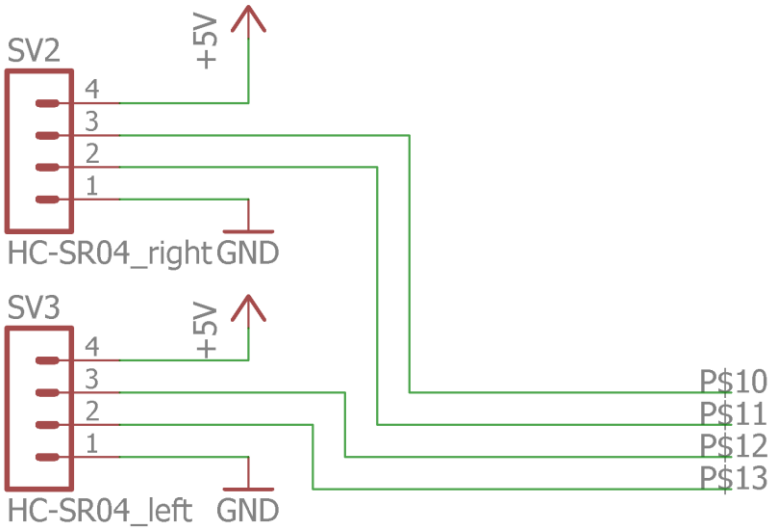

Рисунок 30. Электрическая схема подключения ультразвукового дальномера к контроллеру.

Таблица 9. Описание выводов ультразвукового дальномера.

Обозначение	Наименование
Vcc	положительный контакт питания
Trig	цифровой вход. Для запуска измерения необходимо подать на этот вход логическую единицу на 10 мкс. Следующее измерение рекомендуется выполнять не ранее чем через 50 мс.
Echo	цифровой выход. После завершения измерения, на этот выход будет подана логическая единица на время, пропорциональное расстоянию до объекта
GND	отрицательный контакт питания

3.2.7 Световая индикация

Адресуемый светодиод – это RGB-светодиод, только с интегрированным контроллером WS2801 непосредственно на кристалле. Корпус светодиода выполнен в виде SMD компонента для поверхностного монтажа. Такой подход позволяет расположить светодиоды максимально близко друг другу, делая свечение более детализированным. Схема подключения адресных диодов приведена на рисунке ниже.

Рисунок 31. Адресный светодиод WS2818B.

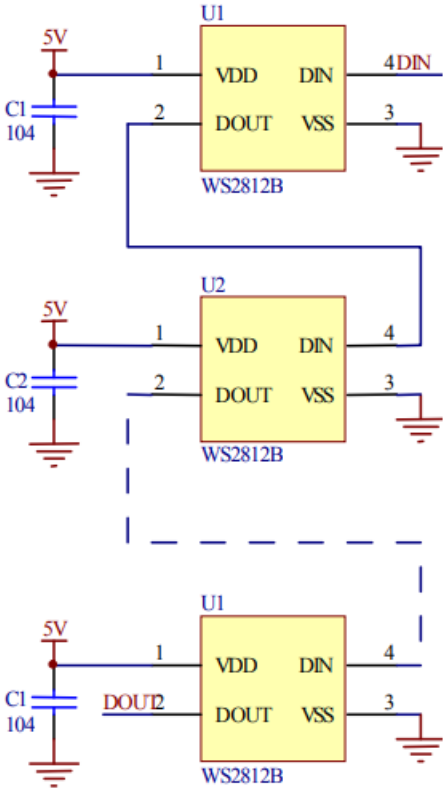

Рисунок 32. Схема подключения светодиодов WS2818B.

Таблица 10. Технические характеристики диодов WS2818B.

Наименование	Значение
Питание светодиода	3,5 ~ 5,3 В
Управление (вход)	-0,5 ~ +0,5 В
Температура эксплуатации	-25 ~ +80 ℃
Температура хранения	-55 ~ +150 ℃

Фонарь представляет собой печатную плату с пятью светодиодами и необходимой для них обвязкой. На шасси располагается четыре фонаря, общее количество светодиодов: 20. Для экономии выводов с контроллера фонари подключаются последовательно. Адресация светодиодов начинается с нулевого элемента. Разработанная электрическая схема и печатная плата приведены на рисунках ниже. Печатная плата имеет крепежные отверстия для фиксации на бампере транспортного средства.

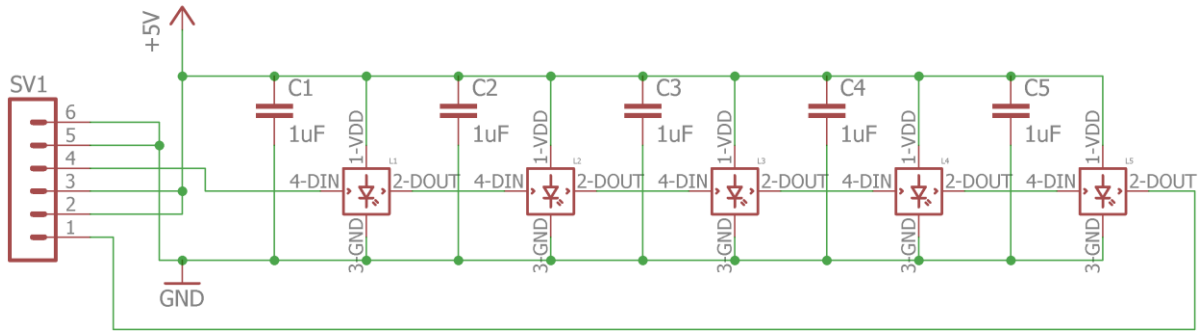

Рисунок 33. Электрическая схема световой индикации (фонарь).

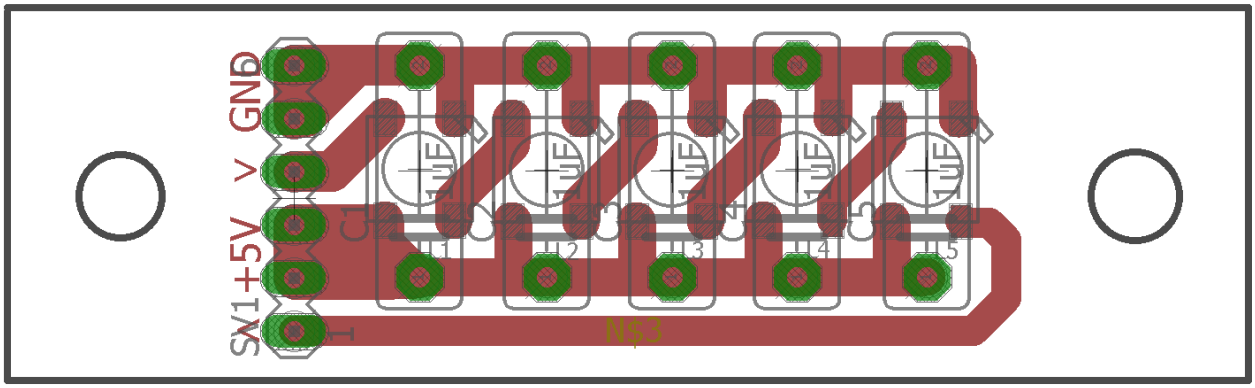

Рисунок 34. Плата световой индикации (фонарь).

Таблица 11. Описание выводов платы световой индикации (фонаря).

Обозначение	Наименование
5V	положитльный контакт питания 5В
GND	отрицательный контакт питания
>	вход управления
<	выход управления (для подключения следующего фонаря)

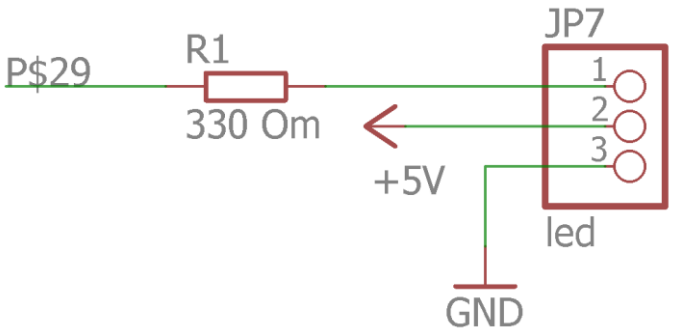

Рисунок 35. Электрическая схема подключения световой индикации к контроллеру.

3.3 Система управления высокого уровня

3.3.1 Микрокомпьютер

Для обработки большого количества данных необходим вычислительный модуль способный выполнять большинство расчётов непосредственно на борту. В качестве такого модуля выбрана аппаратная платформа Firefly RK3399 от компании T-firefly.

В таблице ниже приведены технические характеристики аппаратной платформы.

Таблица 12. Технические характеристики аппаратной платформы.

Наименование характеристики	Значение характеристики
Процессор	RockChip RK3399
Архитектура процессора	ARM 64bit
Количество процессорных ядер	2 x Cortex-A72, 4 x Cortex-A53
Частота процессора	2.0 ГГц
Видеоускоритель	Mali-T860MP4
Оперативная память	4ГБ
Постоянная память	32ГБ
Сетевые интерфейсы	Wi-Fi MIMO 2.4/5ГГц, 1000/100 Ethernet
Интерфейсы взаимодействия	USB Type-C, 4xUSB 2.0, USB 3.0,SPI,I2C,PWM, Serial, Bluetooth

Питание модуля осуществляется от бортовой сети с напряжением 12 В.

Программной частью модуля является ОС Ubuntu Linux 16.04 (версия ядра: Linux 4.4) с установленной оболочкой ROS Kinetic Kame. Ниже приведено описание компонентов микрокомпьютера.

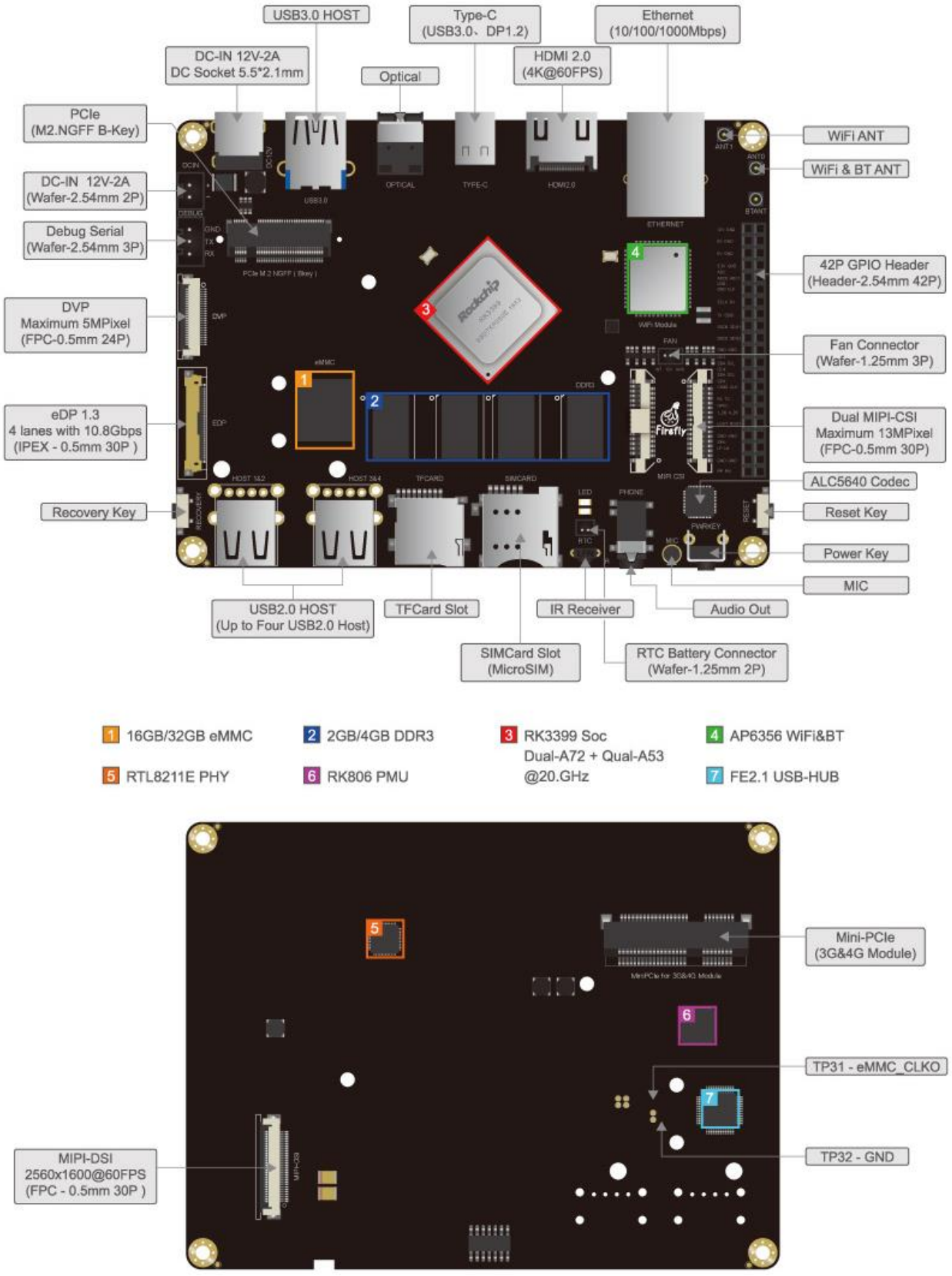

Рисунок 36. Описание компонентов микрокомпьютера.

3.3.2 Лазерный дальномер

Lidar LDS-01 – это лазерный 2D-сканер, способный измерять 360 градусов, собирая вокруг робота набор данных для использования для SLAM (одновременная локализация и картирование). Он поддерживает интерфейс USB и прост в установке на ПК. Принцип действия не имеет серьезных отличий от принципа работы радара: направленный луч источника излучения отражается от целей (непрозрачных тел), возвращается к источнику и улавливается высокочувствительным светочувствительным полупроводниковым прибором.

Рисунок 37. Лазерный дальномер LDS-01.

Таблица 13. Характеристики лазерного дальномера.

Наименование	Характеристика
Рабочее напряжение	5V DC ±5%
Источник	Полупроводниковый лазерный диод (= 785 нм)
Класс безопасности	IEC60825-1 Class 1
Потребление тока	400mA или меньше (Пиковый ток 1A)
Интерфейсы	3.3V USART (230,400 bps) 42 байта на 6 градусов, опция Full Duplex
Сопротивление окружающему освещению	10,000 люкс или меньше
Частота	1.8kHz
Габаритные размеры	69.5(В) X 95.5(Д) X 39.5(Ш)мм
Масса	Меньше 125г.

Таблица 14. Характеристики измерений лазерного дальномера.

Наименование	Характеристика
Диапазон	120мм ~ 3,500мм
Точность (120-499)мм	± 15мм
Точность (500-3500)мм	± 5%
Прецизионность измерения (120-499)мм	± 10мм
Прецизионность измерения (500-3500)мм	± 3,5%
Скорость сканирования	300 ± 10 об / мин
Угловой диапазон	360 °
Угловое разрешение	1 °

Для подключения используется интерфейс USB2LDS, это плата, которая позволяет легко использовать лазерный дальномер с микрокомпьютером или ПК. Интерфейс USB2LDS подключается к микрокомпьютеру или ПК с помощью кабеля micro-USB и имеет порты для LDS и его двигателя.

Для реализации возможностей лазерного дальномера используется драйвер для ROS: hls_lfcd_lds_driver.

3.3.3 Цифровая камера

В системе для получения информационных видеопотоков используются цифровые камеры Logitech C930e.

Рисунок 38. Цифровая камера Logitech C930e.

Основные характеристики цифровой камеры приведены в таблице ниже.

Таблица 15. Основные характеристики Logitech C930e.

Наименование	Характеристика
Макс. размер снимка	1920x1080 пикс / до 15 Мп
Фокусировка	автоматическая / 20-шаговая /
Угол обзора	90 °
Частота кадров	30 кадр/с
Интерфейсы	USB
Матрица	3 млн пикс., CMOS
Габаритные размеры	94 (В) X 43(Д) X 71(Ш)мм
Масса	162 г

Для отслеживания дорожных знаков и светофора камера установлена в верхней части корпуса. Фиксацию дорожной разметки выполняет камера, установленная на передней части корпуса. Подключение камер к микрокомпьютеру выполнено с помощью интерфейса USB.

Глава 4. Программная реализация

4.1 Система управления низкого уровня (драйвер шасси)

4.1.1 Контроллер управления

Система управления низкого уровня реализована на контроллере STM32F103C8T6 (Blue Pill).

Для использования контроллера его необходимо прошить, используя USB-UART преобразователь. Для этого нужно переключить преобразователь в режим 3,3V и подключить выводы согласно таблице ниже.

Таблица 16. Подключение контактов USB-UART преобразователя к плате Blue Pill.

USB-UART преобразователь	Blue Pill
Gnd	Gnd
Vcc	3.3V
RX	PA9
TX	PA10

Для программирования и загрузке скетча используя Arduino IDE необходимо добавить поддержку SMT32F103 (File / Preferences / Additional Boards Manager URL):

```
http://dan.drown.org/stm32duino/package_STM32duino_index.json
```

В Boards Manager (Tools / Board / Boards Manager) установить "SMT32F1xx/GD32F1xx boards by smt32duino" и активировать загрузку прошивки через Serial (Tools / Upload method / Serial). На плате нужно переключить перемычку BOOT0 в High, а BOOT1 в Low. После этого нужно произвести сброс платы и использовать полату. Для нормальной работы прошивки BOOT0 и BOOT1 должны быть в состоянии Low.

Затем в консоли требуется выполнить следующие команды:

```
$ wget -P /tmp https://github.com/rogerclarkmelbourne/STM32duino-bootloader/raw/master/binaries/generic_boot20_pc13.bin
$ cd ~/.arduino15/packages/stm32duino/tools/stm32tools/2018.12.3/linux/stm32flash
$ ./stm32flash -w /tmp/generic_boot20_pc13.bin -v -g 0x0 /dev/ttyACM0
```

После выполнения команд необходимо отключить USB-UART преобразователь, выставить перемычки BOOT0 и BOOT1 в Low и подключить плату к компьютеру через USB. Затем в Arduino IDE выбирается тип загрузки "STM32duino bootloader" (Tools / Upload method / STM32duino bootloader).

Последовательный порт платы опознается в Linux как /dev/ttyACM0 и его необходимо выбрать в Arduino IDE (Tools / Port).

Первую загрузку скетча нужно выполнять, включив режим infinite в загрузчике. Для этого подключается резистор номиналом 10k между 3,3v и PC14. После сброса светодиод на плате начинает постоянно мигать, это означает, что загрузчик будет ждать начала загрузки без ограничения по времени. После первой загрузки этот режим можно выключить и дальше пользоваться платой как обычно.

Язык программирования Arduino основан на C/C++. Для использования ROS с платой Arduino используется библиотека — ros_lib_stm32.

ROS (Robot Operating System) предоставляет библиотеки и инструменты, помогающие разработчикам программного обеспечения создавать приложения для роботов. Он обеспечивает аппаратную абстракцию, драйверы устройств, библиотеки, визуализаторы, передачу сообщений, управление пакетами и многое другое. ROS лицензируется под открытым исходным кодом, лицензия BSD.

4.1.2 Управление электродвигателями

Узел подписан на топик cmd_vel, сообщения имеют тип данных geometry_msgs::Twist. В переменной cmd_vel.linear.x содержится значение линейной скорости (м/с).

Функция update_motors() реализовывает алгоритм обновления значений в драйвере моторов с частотой 50 герц. В алгоритме производится вызов 2 функций:

- linear2driverMotor(float linear_speed);
- driverMotor(float linear).

В функции linear2driverMotor(float linear_speed) реализован ПИД-регулятор позволяющий поддерживать постоянную скорость используя среднее значение с оптических датчиков числа оборотов колес.

Исходный код позволяет изменять коэффициенты для ПИД-регулятора, значения инициализируются используя функцию update_params(). При отсутствии параметров /pid используются значения, содержащиеся в следующих константах: Kp, Ki, Kd.

```
//Расчет средней скорости движения между публикациями
float speed_actual = -(state_vel[0]+state_vel[1])/2;
//разница в скорости средней от последней публикации и желаемая (m/s)
float e = speed_actual * WHEEL_DIAMETER/2 - linear_speed;

//ПИД регулятор для расчета значения для драйвера моторов
```

```
float P = pid_constants[0] * e;
float I = I_prev + pid_constants[1] * e;
float D = pid_constants[2] * (e - e_prev);
float motor_value = round(P + I + D);
```

Полученное значение передается в функцию driverMotor(float linear), которая передает управляющее воздействие непосредственно драйверу моторов.

4.1.3 Рулевое управление

Для управления сервомотором используется библиотека DynamixelSerial3. В библиотеке реализован необходимый функционал для управления через третий последовательный порт с сервомотором.

В исходном коде реализована возможность корректировки идентификатора сервомотора (DYNAMIXEL_ID), центрального положения колес (DYNAMIXEL_CENTER), и максимального угла поворота колес (GRAD_RUDDER).

В переменной cmd_vel.angular.z содержится значение угла поворота колес в единицах угловой скорости (рад/с).

В функции angular2dynamixel(float angular) реализовано математическое преобразование угла поворота колес к значениям необходимым для управления сервомотором с программным ограничением максимального возможного угла поворота сервомотора относительно значений указанных в градусах (GRAD_RUDDER).

Перевод единиц измерений (рад/с в градусы) производим по следующей формуле:

```
angular_grad = angular * 180.0 / M_PI;
```

Расчёт максимального возможного значения поворота колес в одном направлении для управления сервомотором рассчитывается по следующей формуле:

```
angular_value_res = ((1024.0 / 300.0) * (GRAD_RUDDER));
```

Для расчета значения необходимого для управления сервомотором используется функция map и значения лимитов для сервомотора относительно центрального программируемого значения (DYNAMIXEL_CENTER):

```
map(angular_grad * 100.0, -(GRAD_RUDDER) * 100.0, (GRAD_RUDDER) * 100.0,
round(DYNAMIXEL_CENTER - angular_value_res), round(DYNAMIXEL_CENTER +
angular_value_res));
```

Значение положения сервомотора получается, используя функцию Dynamixel::Dynamixel.readPosition(DYNAMIXEL_ID). В функции dynamixel2angular(int dynamixel) реализован алгоритм преобразования значений получаемых с сервомотора в угол поворота в радианах.

4.1.4 Оптический датчик числа оборотов колеса и датчик направления движения

Для реализации расчета одометрии автомобиля необходимо получение информации о пройденном расстоянии и скорости движения. Эти значения вычисляются, анализируя изменения состояний оптических датчиков числа оборотов колес и датчика направления движения.

Расчет скорости и расстояния производится с частотой 50 герц, данные публикуются в топик joint_states.

Значения о скорости вращения левого и правого задних колес измеряется в единице измерения - радиан в секунду. А значение угла поворота передних колес в радианах.

Алгоритм получения количества оборотов колеса реализован с использованием прерываний контроллера. При каждом изменении состояний оптических датчиков происходит вызов функции getEncoderCount() и суммирование возвращаемого значения функции с значением счетчика импульсов для каждого из колес. Функция учитывает направление движения при наличии управляющего воздействия для движения, а при его отсутствии (движение по инерции) учитывается состояние датчика направления. При движении назад функция возвращает единицу с отрицательным знаком, а при движении вперед с положительным знаком.

4.1.5 Датчик напряжения аккумулятора

Аккумулятор имеет балансировочный разъем, с помощью которого можно получать данные о напряжении на каждой секции аккумулятора. Информация о напряжении получается путем считывания значения с аналогового входа и пересчитывается с помощью экспериментально полученной функции. Ниже приведен график показывающий зависимость напряжения и значений аналового входа.

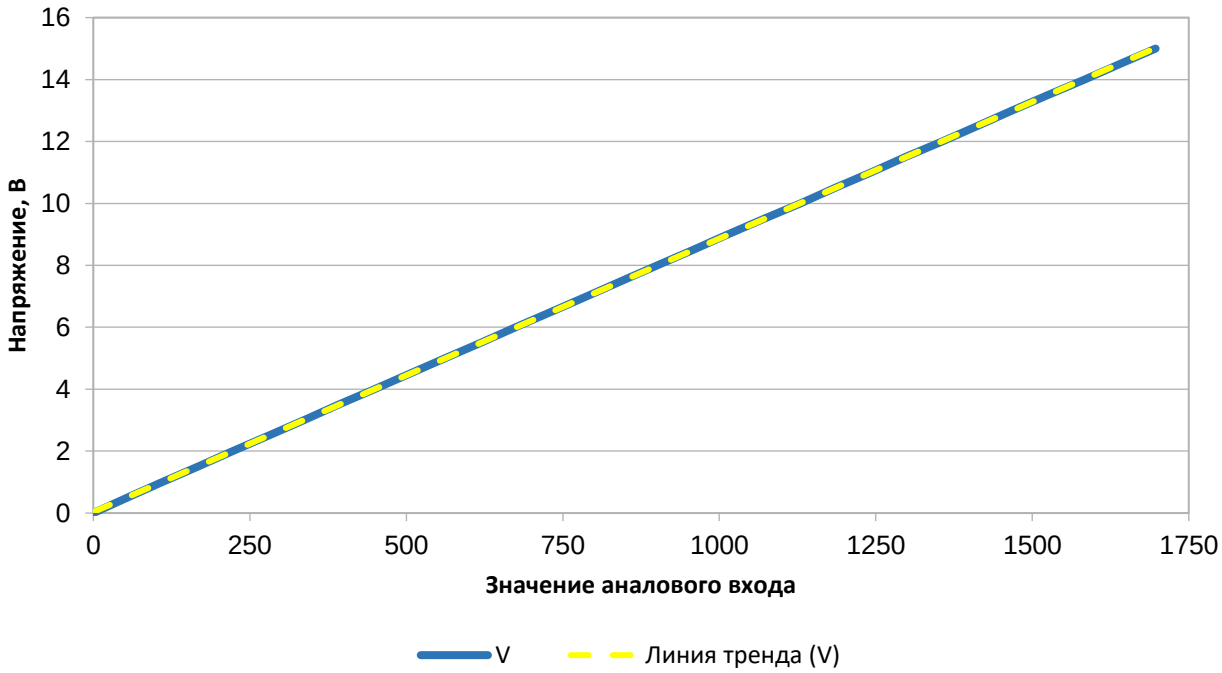

Рисунок 39. График соотношения напряжения и значений аналогового входа.

Коэффициент детерминации равный 0,9998, указывает высоком уровне достоверности аппроксимации.

Линия тренда имеет следующее уравнение:
V = 0.0088316807 * analog + 0.031247058.

Значение напряжения 2, 3 и 4 секций аккумулятора получается за счет разности значений текущей секции с предыдущей. Общее значение напряжение аккумулятора рассчитывается путем сложения всех значений напряжения по секциям.

Информация о напряжении публикуется в топик battery, типом сообщений является sensor_msgs::BatteryState.

4.1.6 Ультразвуковой дальномер

Данные о расстоянии до объектов получаемые от ультразвуковых датчиков, подвергаются фильтрации и публикуются в топики range/front/left_snr и range/front/right_snr, типом данных сообщений является sensor_msgs::Range.

Фильтрация сигналов с ультразвуковых датчиков имеет важное значение, так как сигналы с датчиков имеют большую зашумленность, чтобы использовать их непосредственно. Чтобы исправить это используется фильтрация Калмана для оценки расстояния до объектов. Фильтр Калмана

имеет еще одну особенность которая позволяет прогнозировать будущее значение.

Использовать ультразвуковые дальномеры позволяет библиотека Ultrasonic, а для фильтрации данных получаемых от дальномеров используется библиотека SimpleKalmanFilter.

4.1.7 Световая индикация

Световая индикация имеет две библиотеки:

- LedControl;
- LedWS2812BforSTM32.

Библиотека LedControl представляет собой исходный код, подписывающийся на топики light_control/head, light_control/turn, light_control/back, light_control/brake, light_control/flasher, light_control/position, для получения управляющих воздействий. Алгоритм производит заполнение массива _indication_status типа bool.

Структура элементов массива имеет вид:

1. POSITION_LAMPS;
2. DIPPED_BEAM_LAMPS;
3. HIGH_BEAM_LAMPS;
4. BRAKE_LAMPS;
5. BACK_LAMPS;
6. TURN_LEFT_LAMPS;
7. TURN_RIGHT_LAMPS;
8. FLASHER_LAMPS.

Данный массив передается в библиотеку LedWS2812BforSTM32 путем вызова функции LedWS2812BforSTM32::indication(bool *indication_status).

Данная библиотека имеет конкретную реализацию режимов работы и схемы индикации. Для изменения режимов работ или схемы необходимо заменить конечную библиотеку, реализующую непосредственно визуальную составляющую индикации.

4.2 Система управления высокого уровня

Система управления высокого уровня реализована с помощью ROS.

ROS в основном используется в связке с микроконтроллерами, которые обладают большими вычислительными возможностями и собственной операционной системой.

4.2.1 Лазерный дальномер

Использовать лазерный дальномер позволяет пакет ROS для LDS (HLS-LFCD2). LDS (Laser Distance Sensor) – это датчик, отправляющий данные на хост для одновременной локализации и картирования (SLAM). Одновременно данные об обнаружении препятствий также могут быть отправлены на хост. Компания HLDS (Hitachi-LG Data Storage) разрабатывает технологию для датчика подвижной платформы.

Информация с лазерного дальномера публикуется в топик lidar_scan, типом данных сообщений является sensor_msgs::LaserScan, представляющий собой множество точек в пространстве, для каждой точки имеется значения угла этой точки относительно начала отсчета.

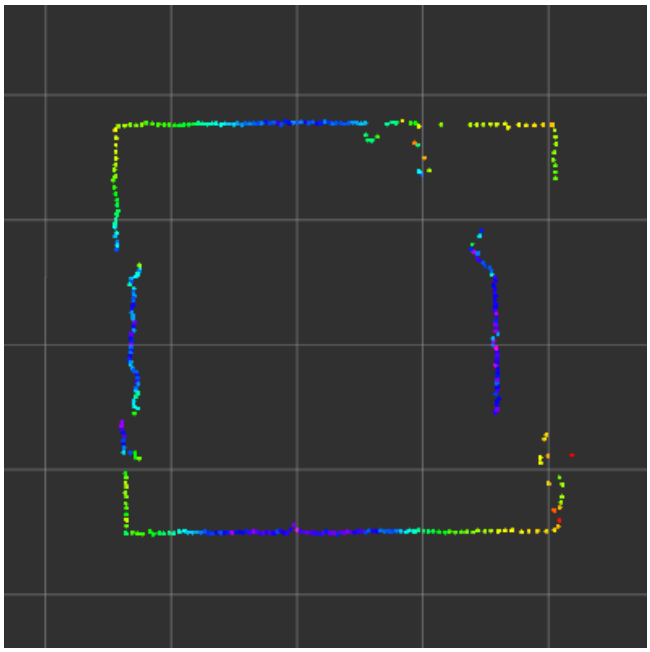

Рисунок 40. Визуальное представление данных получаемых с лазерного дальномера.

На рисунке изображено визуальное представление некоторого помещения с препятствиями. По цветовой температуре визуально видно на сколько далеко находится каждая точка относительно лазерного дальномера.

Установка:

```
$ sudo apt-get install ros-kinetic-hls-lfcd-lds-driver.
```

Предоставление прав на порт:

```
$ sudo chmod a+rw /dev/ttyUSB0.
```

Запуск узла:

```
$ roslaunch hls_lfcd_lds_driver hlds_laser_segment.launch.
```

Загрузка драйвера и исходного кода:

```
$ git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git.
```

Launch файл включает в себя следующий код:

```
<?xml version="1.0"?>
<launch>
  <node pkg="hls_lfcd_lds_driver" type="hlds_laser_segment_publisher"
name="hlds_laser_segment_publisher" output="screen">
    <param name="port" value="/dev/ttyUSB0"/>
    <param name="frame_id" value="laser"/>
  </node>
</launch>
```

4.2.2 Одометрия

Одометрия — использование данных о движении приводов, для оценки перемещения транспортного средства.

Узел имеет подписку на топик joint_states, сообщения имеют тип данных sensor_msgs::JointState. В сообщениях имеется информация о текущей скорости вращения и количестве оборотов задних колес, а также угол поворота передних колес.

После математических вычислений информация о перемещении транспортного средства публикуется в топик odom, типом данных сообщений является nav_msgs::Odometry.

Узел так же публикует сообщения о трансформации odom – base_link.

Информация в топиках обновляется с частотой 50 герц.

Запуск узла:

```
$ roslaunch odometry odometry.launch.
```

Launch файл включает в себя следующий код:

```
<?xml version="1.0"?>
<launch>
  <node pkg="odometry" type="odometry_node" name="odometry_node"
respawn="true" output="screen">
  </node>
</launch>
```

Respawn / Required являются не обязательными аргументами, эти аргументы используются по отдельности и никогда не используются вместе. Если respawn = true, то этот конкретный узел будет перезапущен, если по

какой-то причине он закрылся. Если required = true то произойдет наоборот, то есть, если этот конкретный узел выйдет из строя, он закроет все узлы, связанные с файлом запуска.

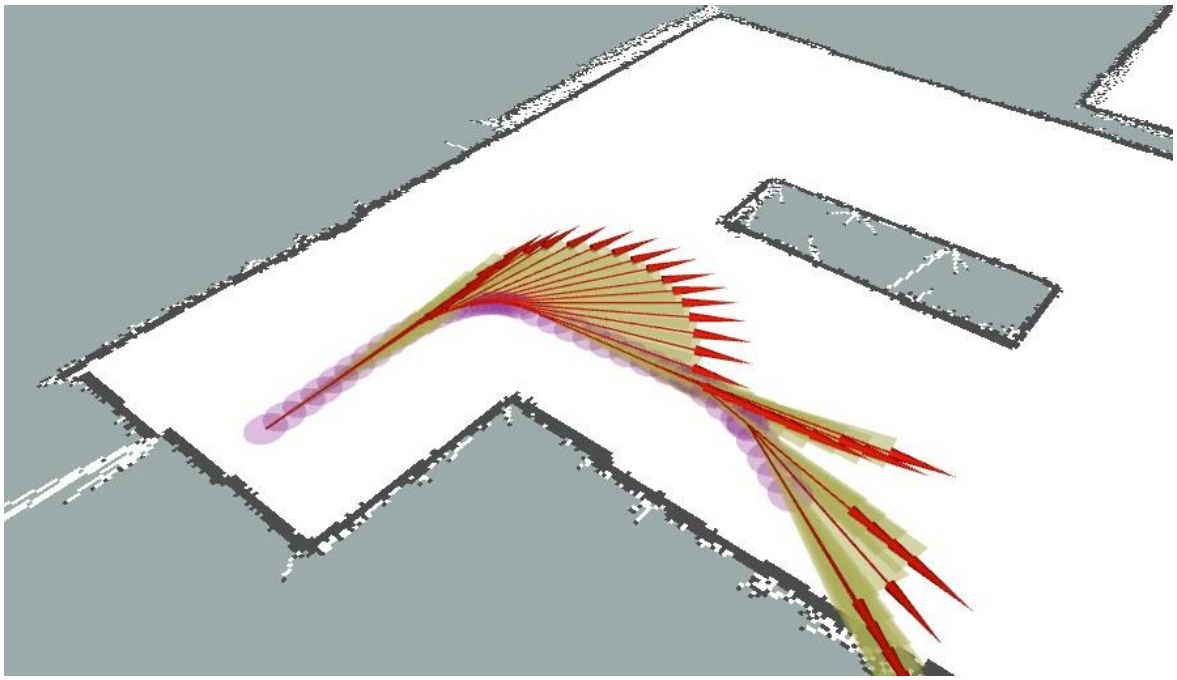

Рисунок 41. Карта с следом одометрии робота.

4.2.3 SLAM

Метод одновременной локализации и построения карты (SLAM от англ. simultaneous localization and mapping) — метод, используемый в мобильных автономных средствах для построения карты в неизвестном пространстве или для обновления карты в заранее известном пространстве с одновременным контролем текущего местоположения и пройденного пути. Популярные методы приближенного решения данной задачи включают в себя фильтр частиц и расширенный фильтр Калмана.

Узел hector_slam для работы использует данные 2D лазерного дальномера в формате sensor_msgs::LaserScan (публикуемые в топик /lidar_scan) и выполняет построение карты и локализацию на той же частоте, с которой выполняется сканирование лазерным дальномером.

На данный момент в ROS имеются несколько алгоритмов SLAM. К сожалению, большая их часть не поддерживается в последних версиях ROS.

- gmapping. Одна из самых популярных реализаций. Данный алгоритм использует данные, поступающие с лазерного дальномера и с энкодеров. Однако существуют способы заменить лазерный дальномер на другой датчик расстояния, например Kinect.

• hector_slam. Данная реализация также является одной из самых популярных, а её основной особенностью считается возможность не использовать данные, полученные с одометрии робота.

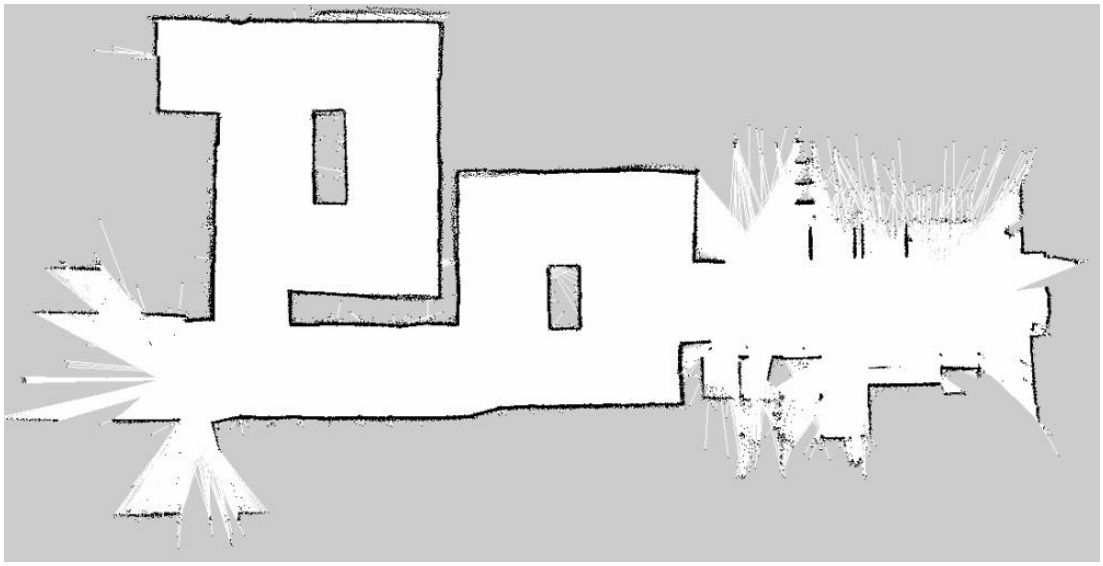

Рисунок 42. Карта отладочного полигона, построенная при помощи SLAM.

Установка ROS пакет hector_slam через apt-get:

```
$ sudo apt-get install ros-kinetic-hector-slam.
```

Загрузка драйвера и исходного кода:

```
$ git clone https://github.com/tu-darmstadt-ros-pkg/hector_slam.
```

Launch файл позволяет задать размер карты в пикселях, разрешение метров в пикселе, а так же начальную позицию автомобиля и многое другое.

```
<launch>
<arg name="odom_frame" default="base_link"/>
<arg name="scan_subscriber_queue_size" default="5"/>
<arg name="scan_topic" default="scan"/>
<arg name="map_size" default="2000"/>
<arg name="pub_map_odom_transform" default="true"/>

  <node pkg="tf" type="static_transform_publisher"
name="base_laser_link" args="-0.18 0.0 0.0 0.0 0.0 0.0 /base_link
/base_laser_link 50" />
  <node pkg="hector_mapping" type="hector_mapping"
name="hector_mapping" output="screen">
    <!-- Frame names -->
    <param name="pub_map_odom_transform" value="$(arg
pub_map_odom_transform)"/>
    <param name="map_frame"  value="map" />
```

```
    <param name="odom_frame" value="$(arg odom_frame)" />
    <param name="output_timing" value="false"/>
    <!-- Map size / start point -->
    <param name="map_resolution" value="0.02"/>
    <param name="map_size"       value="$(arg map_size)"/>
    <param name="map_start_x"    value="0.2"/>
    <param name="map_start_y"    value="0.2" />
    <param name="map_multi_res_levels" value="2" />
    <!-- Map update parameters -->
    <param name="update_factor_free"         value="0.4"/>
    <param name="update_factor_occupied"     value="0.9" />
    <param name="map_update_distance_thresh" value="0.1"/>
    <param name="map_update_angle_thresh"    value="0.04" />
    <param name="map_pub_period"             value="2" />
    <param name="laser_min_dist"             value="0.12" />
    <param name="laser_max_dist"             value="3.5" />
    <!-- Advertising config -->
    <param name="advertise_map_service"      value="true"/>
    <param name="scan_subscriber_queue_size" value="$(arg
scan_subscriber_queue_size)"/>
    <param name="scan_topic" value="$(arg scan_topic)"/>
   </node>
<node pkg="hector_trajectory_server" type="hector_trajectory_server"
name="hector_trajectory_server" output="screen">
  <param name="target_frame_name" type="string" value="/map" />
  <param name="source_frame_name" type="string"
value="/base_laser_link" />
  <param name="trajectory_update_rate" type="double" value="4" />
  <param name="trajectory_publish_rate" type="double" value="0.25" />
</node>
</launch>
```

Очистка построенной карты (очистка топика /map):

```
$ rostopic pub syscommand std_msgs/String "reset"
```

Сохранение карты в текущую директорию:

```
$ rosrun map_server map_saver
```

После выполнения команды в директории создаются 2 файла:

- map.pgm – содержащий изображение карты;
- map.yaml – содержащий информацию о размере и разрешении карты, а так же значения порогов пикселей (полностью занятых, полностью свободных).

Запуск rviz для визуализации при построении карты:

```
$ roslaunch immo/launch/rviz.launch
```

Ниже представлен граф узлов и топиков используемых SLAM.

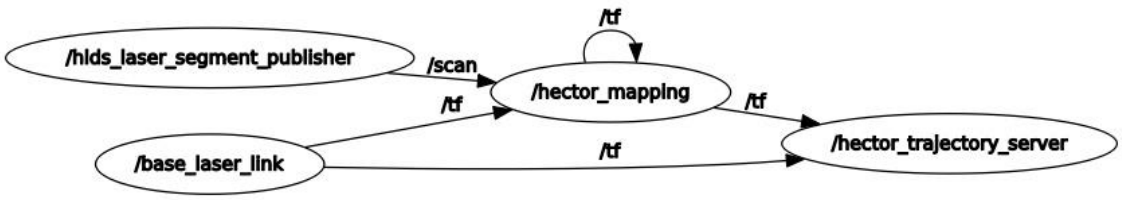

Рисунок 43. Граф узлов и топиков SLAM.

4.2.4 Дистанционное управление

Узел позволяет управлять автомобилем путем публикации сообщений имеющих тип данных geometry_msgs::Twist в топик cmd_vel.

Управляющее воздействие задается с помощью клавиатуры, а именно:

- w/x – изменение скорости на шаг;
- a/d – изменение угла поворота на шаг;
- s – обнулить скорость и угол поворота.

Константы узла позволяют задать шаг изменения значений и максимальные значения скорости вращения и угла поворота колес.

Запуск узла:

```
$ roslaunch teleop teleop_key.launch
$ roslaunch immo/launch/teleop.launch
```

Launch файл включает в себя следующий код:

```
<launch>
  <node pkg="teleop" type="teleop_key" name="teleop_keyboard"
output="screen">
  </node>
</launch>
```

Ниже представлен граф узлов и топиков используемых световой индикацией.

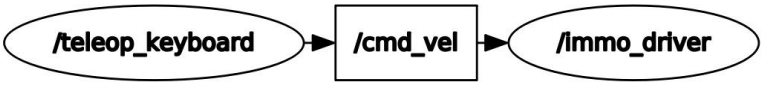

Рисунок 44. Граф узлов и топиков дистанционного управления.

4.2.5 Световая индикация

Узел имеет подписку на топик cmd_vel, сообщения имеют тип данных geometry_msgs::Twist. В переменной cmd_vel.linear.x содержится значение

линейной скорости (м/с), а в cmd_vel.angular.z содержится угол поворота колес (рад/с).

Реализованный алгоритм позволяет выполнять следующие функции:

- включать габаритные огни;
- включать ближний или дальний свет;
- включать мигалку;
- включать указатель поворота при значении поворота передних колес более чем на заданный угол;
- включать аварийный сигнал при отсутствии управляющего воздействия более чем заданное время;
- включать стоп-сигнал при понижении скорости;
- включать задний ход при движении назад.

Данный узел публикует статусы состояния индикации в следующие топики: light_control/head, light_control/turn, light_control/back, light_control/brake, light_control/flasher, light_control/position.

Ниже представлен граф узлов и топиков используемых световой индикацией.

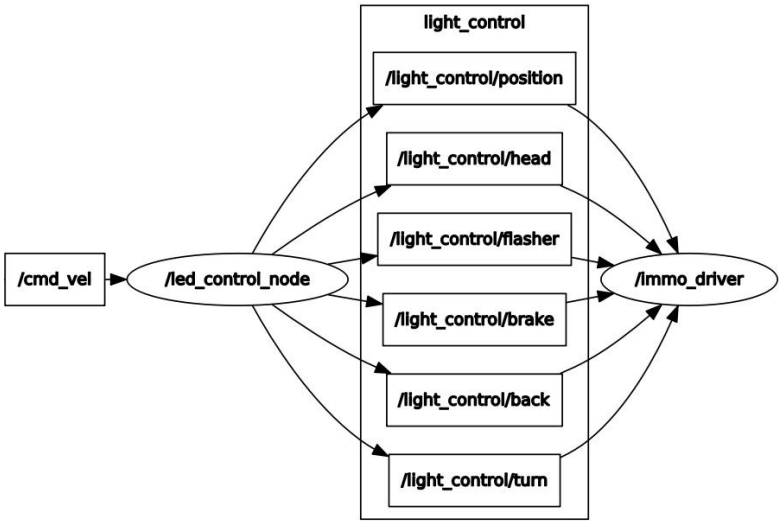

Рисунок 45. Граф узлов и топиков световой индикации.

4.2.6 Планирование и навигация

Планирование и навигация автомобиле-подобных роботов явно не предназначена для стека навигации ROS.

При разработке навигации было рассмотрено 2 варианта реализации:

1. Использование пакета teb_local_planner.
2. Разработка своей системы навигации, основанной на построении и оптимизации маршрута по графам.

Вариант 1:

Пакет teb_local_planner позволяет преодолеть ограничение стека навигации ROS путем предоставления местных планов, которые являются выполнимы для моделей аккермана.

Пакет teb_local_planner реализует плагин для base_local_planner стека 2D-навигации. Базовый метод, названный Timed Elastic Band, локально оптимизирует траекторию робота с учетом времени выполнения траектории, отделения от препятствий и соответствия кинодинамическим ограничениям во время выполнения.

Этот пакет реализует онлайн-планировщик оптимальной локальной траектории для навигации и управления автомобилями в качестве плагина для навигационного пакета ROS. Начальная траектория, сгенерированная глобальным планировщиком, оптимизируется во время выполнения, сводя к минимуму время выполнения траектории (оптимальная по времени цель), отделение от препятствий и соблюдение кинодинамических ограничений, таких как удовлетворение максимальных скоростей и ускорений.

Текущая реализация соответствует кинематике неголономных роботов (дифференциальных приводов и роботов, похожих на машины). Поддержка голономных роботов включена начиная с Kinetic.

Оптимальная траектория эффективно получается путем решения разреженной скалярной многоцелевой задачи оптимизации. Пользователь может предоставить веса для задачи оптимизации, чтобы указать поведение в случае противоречивых целей.

Местные планировщики, такие как Timed-Elastic-Band, часто застревают на локально оптимальной траектории, поскольку они не могут проходить через препятствия, реализуется расширение. Подмножество допустимых траекторий отличительных топологий оптимизируется параллельно. Местный планировщик может переключиться на текущую глобально оптимальную траекторию среди набора кандидатов. Отличительные топологии получены с использованием концепции гомологии / гомотопических классов.

Вариант 2:

Глобальную и локальную навигацию представляет граф нанесенный на карту. Вершины (узлы) графа представляют промежуточные точки на карте. Возможные связи вершин графа представлены ребрами.

Граф позволяет придерживаться нужной полосы во время движения робота.

Для определения и привязке позиции робота к графу используются данные одометрии из топика odom. Узел graph_route_node публикует несколько топиков: target_pose типа geometry_msgs/PoseStamped и graph типа visualization_msgs/MarkerArray.

В топик target_pose публикуется информация о координате следующей вершины графа. Для визуализации маршрута навигации используется топик graph.

Алгоритм реализованный узлом immo_planner использует информацию, публикуемую узлами одометрии, лазерного дальномера, графа маршрутизации движения. Так же узел получает информацию о знаках, разметке и состоянии светофора. Основываясь на перечисленном списке данных производится локальное планирование маршрута и публикация сообщения управляющего воздействия для робота в топик cmd_vel.

Корректироввка положения робота при его движении по заданному маршруту выполняется по привязке к заранее заданным точкам карты.

Этот узел является усовершенствованным аналогом базового пакета amcl. Amcl - вероятностная система локализации для робота, движущегося в 2D. Он реализует адаптивный подход локализации Монте-Карло (как описано Дитером Фоксом), который использует фильтр частиц для отслеживания положения робота на известной карте.

Ниже изображено визуальное представление графа на карте.

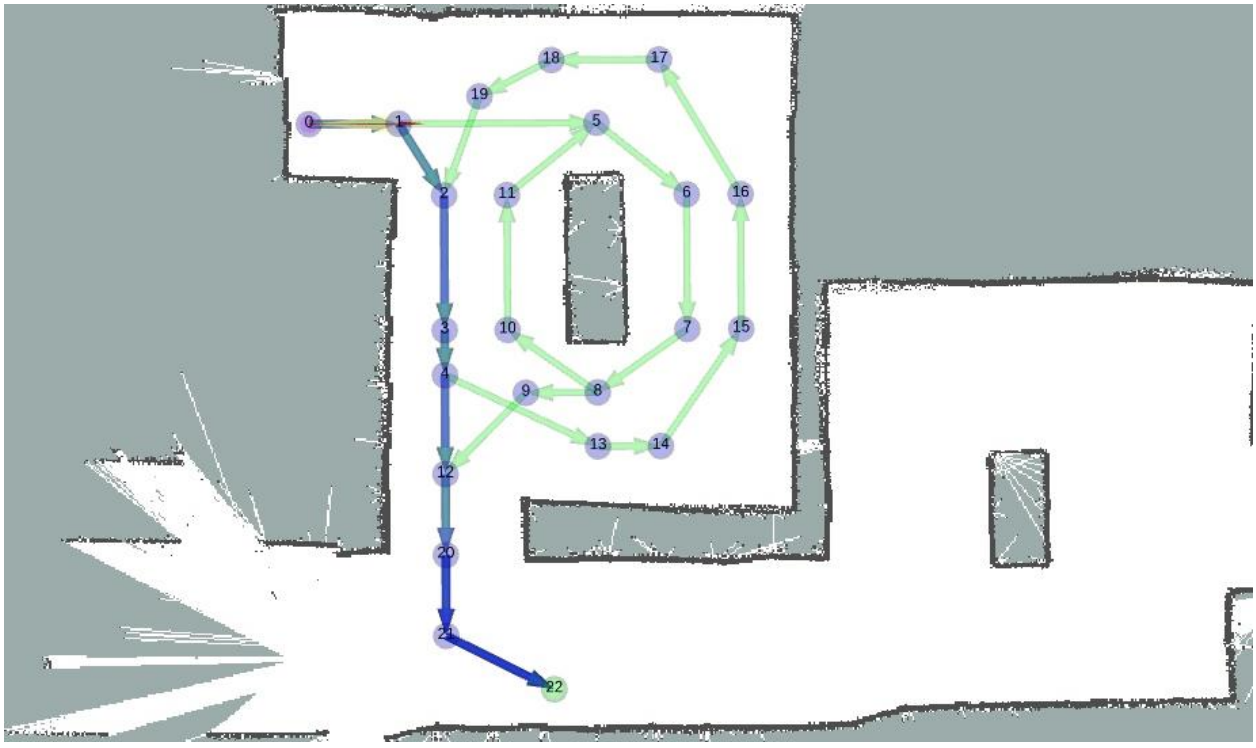

Рисунок 46. Карта с построенным графом маршрута движения.

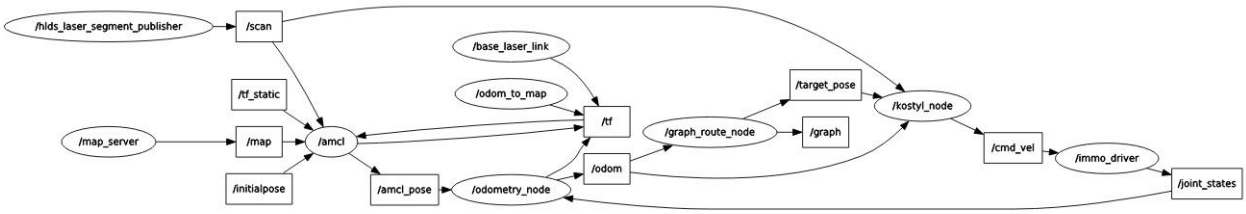

Рисунок 47. Граф узлов и топиков навигации.

4.2.7 Машинное зрение

4.2.7.1 Распознавание дорожных знаков и светофора

Программный код реализован на языке Python. Логика кода изображена на функциональной схеме, представленной на изображении ниже.

Узел camera_fwd получает изображение с камеры, направленной на дорогу и публикует это изображение в топик camera_fwd/image_raw. Узел является экземпляром стандартного компонента usb_cam.

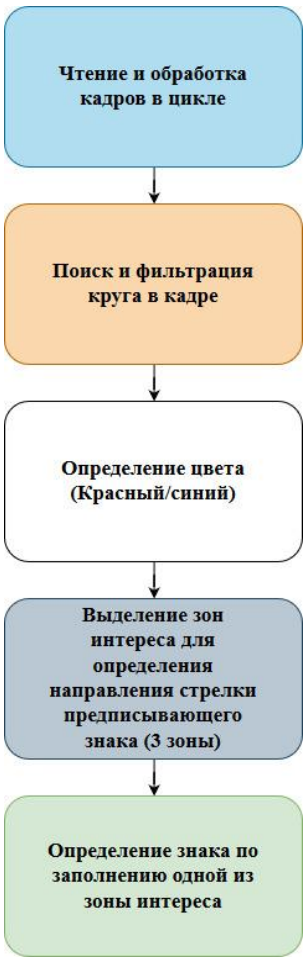

Рисунок 48. Функциональная схема кода распознавания знаков

Для запуска узла создан fwd_cam.launch. Содержимое лаунч файла приведено ниже.

```
<launch>
  <node name="fwd_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value=" fwd_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/fwd_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Узел sign_detector имеет подписку на топик camera_fwd/image_raw сообщением которого является изображение с камеры, которое подвергается алгоритмам обработки распознавания дорожных знаков. Результатом работы узла является публикация сообщения класса string в топик sign. Ниже представлен граф узлов и топиков используемых детектором знаков и результат работы детектора знаков (см. рисунок).

Рисунок 49. Граф узлов

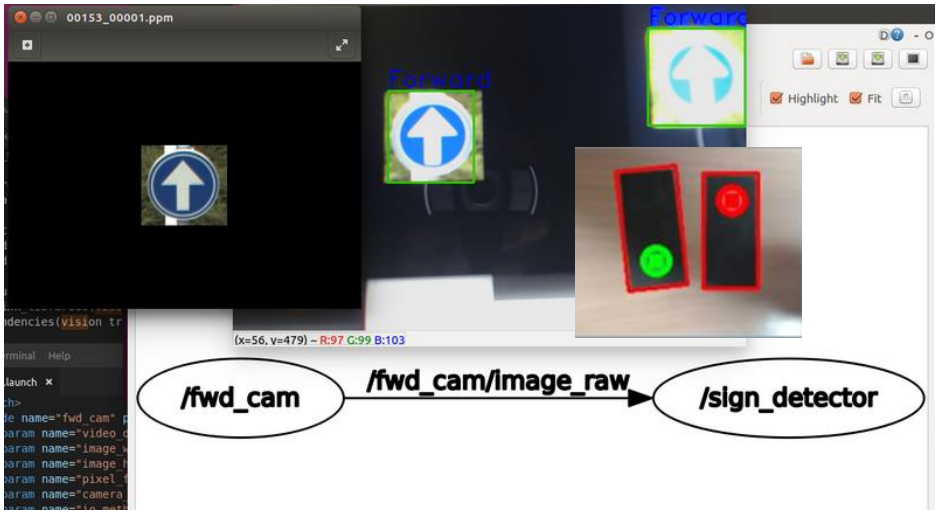

Рисунок 50. Результат работы детектора знаков и светофора

4.2.7.2 Распознавание дорожной разметки

Программный код реализован на языке Python. Логика кода определения дорожной разметки изображена на функциональной схеме, представленной на изображении ниже.

Узел camera_fwd получает изображение с камеры, направленной вперед и передает это изображение в топик /camera_fwd/image_raw. Узел является экземпляром компонента usb_cam. Для запуска узла создан тестовый файл road _cam.launch

Рисунок 51. Функциональная схема кода распознавания дорожной разметки

Для запуска узла создан road_cam.launch. Содержимое лаунч файла приведено ниже.

```
<launch>
  <node    name="road_cam"    pkg="usb_cam"    type="usb_cam_node"
output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="road_cam" />
    <param name="io_method" value="mmap"/>
```

```
        </node>
        <node   name="image_view"   pkg="image_view"   type="image_view"
respawn="false" output="screen">
          <remap from="image" to="/road_cam/image_raw"/>
          <param name="autosize" value="true" />
        </node>
      </launch>
```

Узел line_detector получает изображение из топика /camera_road/image_raw и определяет дорожную разметку. Результатом работы узла является топик /road (тип сообщений sensor_msgs::JointState).

Ниже представлен граф узлов и топиков используемых детектором знаков и результат работы детектора знаков (см. рисунок).

Рисунок 52. Граф узлов

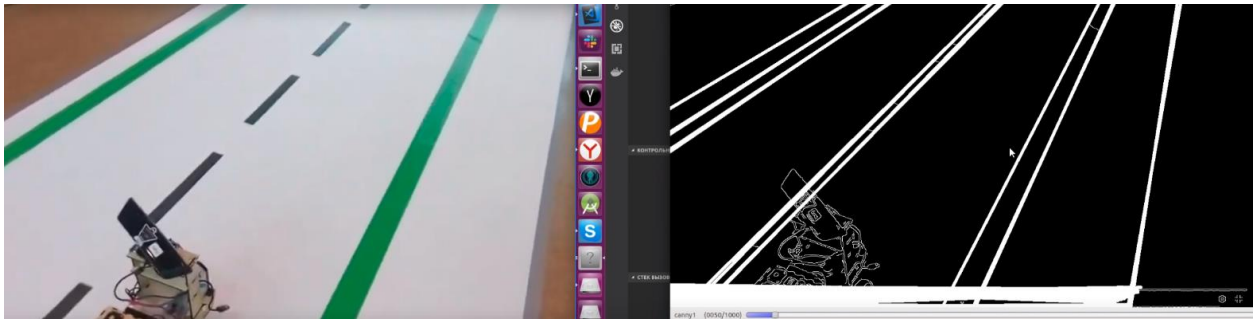

Рисунок 53. Результат работы алгоритма

4.3 Конфигурирование и настройка

4.3.1 Сервисы для автозапуска узлов

Для реализации автозапуска узлов необходимо создать две службы.

Создаем файл конфигурации для roscore:

```
$ sudo nano /lib/systemd/system/roscore.env
```

Содержимое файла roscore.env:

```
ROS_ROOT=/opt/ros/kinetic/share/ros
ROS_DISTRO=kinetic
ROS_PACKAGE_PATH=/home/firefly/immo/src:/opt/ros/kinetic/share
ROS_PORT=11311
ROS_MASTER_URI=http://localhost:11311
CMAKE_PREFIX_PATH=/home/firefly/immo/devel:/opt/ros/kinetic
```

```
PATH=/opt/ros/kinetic/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/us
r/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
LD_LIBRARY_PATH=/opt/ros/kinetic/lib
PYTHONPATH=/home/firefly/immo/devel/lib/python2.7/dist-
packages:/opt/ros/kinetic/lib/python2.7/dist-packages
ROS_IP=192.168.0.1
```

Для создания сервиса необходимо выполнить:

```
$ sudo nano /lib/systemd/system/roscore.service
```

Содержимое файла roscore.service:

```
[Unit]
Description=Launcher for the ROS master, parameter server and rosout
logging no$
After=network.target

[Service]
EnvironmentFile=/lib/systemd/system/roscore.env
ExecStart=/opt/ros/kinetic/bin/roscore
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Данный сервис будет запускаться не ранее чем будет инициализирована сеть.

Для запуска всех необходимых узлов:

```
$ sudo nano /lib/systemd/system/immo.service
```

Содержимое файла immo.service:

```
[Unit]
Description=immo ROS package
Requires=roscore.service
After=roscore.service

[Service]
EnvironmentFile=/lib/systemd/system/roscore.env
ExecStart=/opt/ros/kinetic/bin/roslaunch
/home/firefly/immo/launch/immo.launch --wait
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Данный сервис связан с сервисом roscore. Запуск сервиса immo будет произведен только после успешного старта сервиса roscore.

Для запуска сервисов выполняем:

```
$ sudo systemctl daemon-reload
```

Для включения сервиса в загрузку выполняем:

```
$ sudo systemctl enable immo.service
$ sudo systemctl enable roscore.service
```

Для отключения сервиса в загрузку выполняем:

```
$ sudo systemctl disable immo.service
$ sudo systemctl disable roscore.service
```

Вывод списка служб и их статусов:

```
$ systemctl list-units --type service -all
```

Вывод служб, которые завершились с ошибкой:

```
$ systemctl list-units --type service --state failed
```

Для запуска службы используется команда start, например:

```
$ sudo systemctl start immo.service
$ sudo service immo start
```

Причем расширение service можно опустить, оно и так подставляется по умолчанию. Если запуск прошел хорошо, программа ничего не выведет.

Остановить службу linux можно командой:

```
$ sudo systemctl stop immo.service
$ sudo service immo stop
```

Посмотреть состояние службы позволяет команда status:

```
$ sudo systemctl status immo.service
```

4.3.2 Настройка wifi точки доступа и сети

Установка hostapd:

```
$ sudo apt-get install hostapd
```

Необходимо изменить содержимое файла конфигурации /etc/default/hostapd:

```
$ sudo nano /etc/default/hostapd.conf
```

В нём раскомментировать строку вида (путь к файлу конфигурации демона hostapd):

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Настройка конфига для hostapd:

```
$ sudo nano /etc/hostapd/hostapd.conf
```

Содержание файла /etc/hostapd/hostapd.conf:

```
interface=wlan0
driver=nl80211
ssid=WIFI
hw_mode=g
#ieee80211n=1
#Раскомментировать для включения режима n
#ht_capab=[HT40-][SHORT-GI-40]
#Раскомментировать для включения режима n
channel=6
wpa=2
wpa_passphrase=11111111
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
auth_algs=1
macaddr_acl=0
```

Пробный запуск выполняется в терминале командой:

```
$ sudo hostapd -d /etc/hostapd/hostapd.conf
```

В случае если получаем ошибку в стиле:

```
"Configuration file: hostapd.conf
nl80211: Could not configure driver mode
nl80211 driver initialization failed.
hostapd_free_hapd_data: Interface wlan0 wasn't started"
```

необходимо отключить Network Manager для интерфейса, который должен запускаться с hostapd. Для этого правим конфиг NetworkManager.conf:

```
$ sudo nano /etc/NetworkManager/NetworkManager.conf
```

В файл /etc/NetworkManager/NetworkManager.conf необходимо добавить следующее содержимое:

```
[keyfile]
unmanaged-devices=mac:xx:xx:xx:xx:xx:xx
#где: xx:xx:xx:xx:xx:xx - MAC wifi устройства.
```

Конфиг автоматически проверяется перед запуском, поэтому смело можно запустить hostapd. Команды управления:

```
$ /etc/init.d/hostapd start
$ /etc/init.d/hostapd stop
$ /etc/init.d/hostapd restart
```

На подключающемся устройстве настраиваем статический ip адрес 192.168.0.2, а на автомобиле:

```
$ sudo ip addr add 192.168.0.1/24 dev wlan0
```

Для работы DHCP-сервера, необходимо настроить сетевой интерфейс (прописать статический адрес). Редактируем файл /etc/network/interfaces и добавляем абзац вида:

```
$ auto wlan0
$ iface wlan0 inet static
$ address 192.168.0.1
$ netmask 255.255.255.0
$ gateway 192.168.0.1
```

Сохранить и перезапустить сетевой интерфейс, на котором настроен DHCP:

```
$ ifdown wlan0
$ ifup wlan0
```

Проверить состояние и настройки сетевого интерфейса:

```
$ ifconfig wlan0
```

Добавляем ip-адрес сетевого интерфейса в DNS:

```
$ sudo nano /etc/resolv.conf
```

Содержимое файла /etc/resolv.conf:

```
nameserver 127.0.0.1
nameserver 192.168.0.1
```

Нужно защитить это файл от перезаписи при каждом запуске системы. Перезаписывает его dhclient.

```
$ chattr +i /etc/resolv.conf
```

Если файл не удалось защитить от перезаписи, тогда - устанавливаем resolvconf:

```
$ sudo apt-get install resolvconf
```

Единственное, что нужно для того, чтобы прописать статический адрес DNS для системы — отредактировать /etc/resolvconf/resolv.conf.d/base, вписав всё, что вписано в /etc/resolv.conf:

```
nameserver 127.0.0.1
nameserver 192.168.0.1
```

Перезапуск сервиса resolvconf:

```
$ sudo service resolvconf reload
```

Добавить группу и пользователя:

```
$ sudo groupadd -r dnsmasq
$ sudo useradd -r -g dnsmasq dnsmasq
```

Устанавливаем Dnsmasq, он запускается и готов к работе, но его необходимо отключить:

```
$ sudo apt-get install dnsmasq
$ sudo service dnsmasq stop
```

Настраиваем конфигурацию DNS:

```
$ sudo vim /etc/dnsmasq.conf
```

Содержимое файла /etc/dnsmasq.conf:

```
user=dnsmasq
group=dnsmasq
port=53
cache-size=1000
domain-needed
bogus-priv
interface=wlan0
except-interface=ppp0
dhcp-range=192.168.0.10,192.168.0.150,12h
dhcp-host=40:9f:38:36:e3:19,192.168.0.2 # KV33-ADM-N01
dhcp-host=50:B7:C3:B5:A5:67,192.168.0.3 # RR
dhcp-host=68:14:01:94:2B:4F,192.168.0.4 # Ubuntu ROS
#dhcp-host=11:22:33:44:55:66,fred,192.168.0.60,45m
#dhcp-host=11:22:33:44:55:66,ignore
#dhcp-authoritative
```

Запуск сервиса DNS:

```
$ sudo service dnsmasq start
```

4.3.3 Конфигурирование соединения

ROS - это распределенная вычислительная среда. Работающая система ROS может содержать десятки, даже сотни узлов, распределенных по нескольким машинам. В зависимости от того, как настроена система, любому узлу может потребоваться связь с любым другим узлом в любое время.

В результате ROS предъявляет определенные требования к конфигурации сети:

- Должно быть полное двунаправленное соединение между всеми парами машин на всех портах.
- Каждая машина должна объявить себя именем, которое могут разрешить все другие машины.

Подключение по SSH к роботу:

```
$ ssh robot@<IP_OF_ROBOT>
```

Необходимо заменить <IP_OF_ROBOT> именем хоста или IP-адресом робота.

На роботе необходимо выполнить следующие команды:

```
$ echo export ROS_MASTER_URI=http://localhost:11311 >> ~/.bashrc
$ echo export ROS_HOSTNAME=IP_OF_ROBOT >> ~/.bashrc
```

На ПК необходимо выполнить следующие команды:

```
$ echo export ROS_MASTER_URI=http://IP_OF_ROBOT:11311 >> ~/.bashrc
$ echo export ROS_HOSTNAME=IP_OF_PC >> ~/.bashrc
```

Или для удобства отладки необходимо добавить следующий абзац в .bashrc (при подключенной определенной сети будет переопределен ROS_MASTER):

```
#START FOR ROS
#IP адрес робота и имя wifi сети к котой мы должны быть подключенны
ip_robot1=192.168.0.1
ip_robot2=192.168.2.1
wifi_name1="IMMO"
wifi_name2="KV33-NET"

wifi=$(iwgetid -r)
if [ "$wifi" != "" ]; then
  interface=$(iwgetid | cut -d " " -f 1)
  ip_wifi=$(ifconfig "$interface" | grep "inet addr:" | cut -d " " -f
12 | cut -d : -f 2)
  if [ "$wifi" = "$wifi_name1" ] ; then
    export ROS_MASTER_URI=http://"$ip_robot1":11311
    export ROS_HOSTNAME="$ip_wifi"
    echo !!! ROS_MASTER = ROBOT !!!
  fi
  if [ "$wifi" = "$wifi_name2" ] ; then
    export ROS_MASTER_URI=http://"$ip_robot2":11311
    export ROS_HOSTNAME="$ip_wifi"
    echo !!! ROS_MASTER = ROBOT !!!
  fi
fi
#END FOR ROS
```