**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский

технический университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт Автоматики и электронного приборостроения

Кафедра Автоматики и управления

Направления подготовки: 27.04.04 «Управление в технических системах»

Образовательная программа: Управление и информатика в интеллектуальных технических системах

|  |
| --- |
| **К защите допустить** |
| **Зав. каф. АиУ** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Н.Н. Маливанов /** |
| **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.** |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Интеллектуальная система управления мобильным

колесным роботом»

ОБУЧАЮЩИЙСЯ Хамидуллин Адель Ильсурович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(фамилия, имя, отчество) (подпись)*

РУКОВОДИТЕЛЬ к.ф.м.н., доцент, Лазарева Полина Александровна  \_\_\_\_\_\_\_\_\_\_\_\_\_

*(ученая степень, звание, фамилия, имя, отчест (подпись)*

Казань 2023 г.

Intelligent control system for mobile wheeled robot

*By*

*Khamidullin Adel Ilsurovich*

Submitted to the Department of Automation and Control

in partial fulfillment of the Requirements for the degree of

MASTER OF SCIENCE

at the

Federal State Budgetary Educational Institution of Higher Education

«Kazan National Research Technical University named after A.N.Tupolev-KAI»

(KNRTU-KAI)

|  |  |  |  |
| --- | --- | --- | --- |
| Author |  |  | Khamidullin Adel Ilsurovich |
| *(signature)* |  |  |
| Supervisor |  |  | Polina Alexandrovna Lazareva |
| *(signature)* |  | Associate Professor, Department of Automation and Control |
| Certified by |  |  | Nikolay Nikolaevich Malivanov |
| *(signature)* |  | Associate Professor, Head of the Department of Automation and Control |
| date |  |  |  |
|  |

Kazan 2023

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Казанский национальный исследовательский технический**

**университет им. А.Н. Туполева-КАИ»**

**(КНИТУ-КАИ)**

Институт (факультет), филиал \_\_\_Автоматики и электронного приборостроения\_\_\_\_\_\_\_\_\_

Кафедра Автоматики и управления \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки/специальность\_\_\_\_\_\_Управление в технических системах\_\_\_\_\_\_

**ЗАДАНИЕ**

на выпускную квалификационную работу

обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_Хамидуллина Аделя Ильсуровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

1 Тема выпускной квалификационной работы Интеллектуальная система управления мобильным колесным роботом

утверждена приказом №\_№1360-С\_\_ от «\_14\_»\_\_\_марта\_\_\_\_\_2023г.

2 Срок сдачи обучающимся законченной ВКР «\_6\_\_»\_\_\_\_июня\_\_\_\_\_\_2023г.

3 Исходные данные к выпускной квалификационной работе: литературные и Интернет-источники по разработке алгоритмов передвижения автономного мобильного робота; патенты алгоритмов и конструкторских особенностей мобильных роботов; данные по микро-профилю дорожного покрытия; библиотека Arduino Library; библиотека ESP8266WiFi; фреймворк Django, MQTT-протокол передачи данных;

4 Перечень подлежащих разработке вопросов и исходные данные к ним:

4.1 Литературный обзор и постановка задачи

4.2 Математические модели работы двигателя постоянного тока и движения робота

4.3 Электическая схема и элементная база колесного робота

4.4 Разработка интеллектуального нечеткого алгоритма управления роботом

4.5 Моделирование системы управления роботом с нечетким регулятором

4.6 Разработка программного обеспечения управления и диспетчеризации колесного робота

5 Перечень графического материала (при наличии):

презентация, схема имитационной модели робота, электрические функиональная и принципиальная схемы, иллюстрация разработанной системы управления

6 Консультанты по ВКР (при их наличии, с указанием относящихся к ним разделов):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

Дата выдачи задания «\_8\_»\_\_\_февраля\_\_\_\_ 2023 г.

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Лазарева П.А.\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Задание к исполнению принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

**Календарный план выполнения ВКР**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов (разделов)  выпускной квалификационной работы | Срок выполнения  этапов (разделов) ВКР | Примечание |
| 1 | Литературный обзор и постановка задачи | 08.02.2023 – 17.02.2023 |  |
| 2 | Математические модели работы двигателя постоянного тока и движения робота | 20.02.2023 – 28.02.2023 |  |
| 3 | Электическая схема и элементная база колесного робота | 01.03.2023 – 21.03.2023 |  |
| 4 | Разработка интеллектуального нечеткого алгоритма управления роботом | 22.03.2023 – 10.04.2023 |  |
| 5 | Моделирование системы управления роботом с нечетким регулятором | 11.04.2023 – 21.04.2023 |  |
| 6 | Разработка программного обеспечения управления и диспетчеризации колесного робота | 22.04.2023 – 01.06.2023 |  |
| 7 | Оформление и подготовка к защите ВКР | 02.06.2023 – 05.06.2023 |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Хамидуллин А.И.\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (Фамилия, инициалы) ( дата)*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Лазарева П.А.\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (Фамилия, инициалы) (дата)*

# Аннотация

Целью выпускной квалификационной работы является разработка системы управления роботом и интеллектуального алгоритма движения робота.

В теоретическом разделе произведен литературный обзор, на основании которого был выбран способ реализации данной системы управления. Представлены математические модели двигателей постоянного тока, движения робота. Описана модель микро-профиля дорожного покрытия.

В практической части представлено описание электрических компонентов, электрические функциональная и принципиальная схема мобильного колесного робота. Разработан алгоритм автоматического управления колесным роботом с использованием нечеткой логики. Проведено имитационное моделирование системы управления робота. Выполнена алгоритмизация и программирования мобильного робота и системы управления и диспетчеризации.

Выпускная квалификационная работа содержит 60 страниц, 24 иллюстраций, 10 таблиц, 32 источника и 7 приложений.

# Abstract

The purpose of the final qualifying work is to develop a robot control system and an intelligent robot movement algorithm.

In the theoretical section, a literary review was made, on the basis of which the method of implementing this control system was considered. Mathematical models of DC motors and robot movements are presented. The model of the micro profile of the road surface is described.

The practical part presents a description of the electrical components, electrical functional and schematic diagram of a mobile wheeled robot. An algorithm for automatic control of a wheeled robot using fuzzy logic has been developed. Simulation modeling of the robot control system is carried out. Algorithmization and programming of the mobile robot and the control and dispatching system were performed.

The final qualifying work contains 60 pages, 24 illustrations, 10 tables, 25 sources, 7 appendices.

# СОДЕРЖАНИЕ

[Аннотация 6](#_Toc137682655)

[Abstract 7](#_Toc137682656)

[СОДЕРЖАНИЕ 8](#_Toc137682657)

[ВВЕДЕНИЕ 9](#_Toc137682658)

[1. Литературный обзор и постановка задачи 11](#_Toc137682659)

[1.1 Краткий обзор литературы по моделям, алгоритмам управления и методам исследования автономных транспортных средств 11](#_Toc137682660)

[1.2 Патентная проработка видов и конструкций малогабаритных автономных транспортных средств 15](#_Toc137682661)

[1.3 Постановка задачи 15](#_Toc137682662)

[2. Математические модели работы двигателя постоянного тока и движения тележки в пространстве. 17](#_Toc137682663)

[2.1 Математическая модель двигателя постоянного тока робота. 17](#_Toc137682664)

[2.2 Математическая модель управления роботом-тележкой 19](#_Toc137682665)

[2.3 Характеристика микро-профиля дороги и его математическая модель. 22](#_Toc137682666)

[3. Электрическая схема и элементная база колесного робота 23](#_Toc137682667)

[3.1 Описание электрических компонентов и соединений блока управления роботом 23](#_Toc137682668)

[3.2 Электрическая функциональная схема блока управления 29](#_Toc137682669)

[3.3 Электрическая принципиальная схема блока управления 30](#_Toc137682670)

[4. Интеллектуальный нечеткий алгоритма управления роботом 32](#_Toc137682671)

[5. Моделирование системы управления роботом с нечетким регулятором 38](#_Toc137682672)

[6. Разработка и тестирование программного обеспечения управления и диспетчеризации колесного робота 45](#_Toc137682673)

[6.1 Функциональные требования к программному обеспечению 45](#_Toc137682674)

[6.2 Перечень технологий программного обеспечения 51](#_Toc137682675)

[6.3 Разработка программного обеспечения микроконтроллерами мобильного робота 55](#_Toc137682676)

[6.4 Разработка Web-приложения управления и диспетчеризации 58](#_Toc137682677)

[ЗАКЛЮЧЕНИЕ 62](#_Toc137682678)

[CONCLUSION 63](#_Toc137682679)

[СПИСОК ЛИТЕРАТУРЫ 64](#_Toc137682680)

[Приложение А 67](#_Toc137682681)

[Приложение Б 68](#_Toc137682682)

[Приложение В 69](#_Toc137682683)

[Приложение Г 70](#_Toc137682684)

[Приложение Д 71](#_Toc137682685)

[Приложение Е 74](#_Toc137682686)

[Приложение Ж 76](#_Toc137682687)

# ВВЕДЕНИЕ

Системы автоматики и автоматизация промышленности занимают важнейшую роль в жизни современного общества и государства. Светофоры, пешеходные переходы, компьютерные сети, промышленные станки и т.д. – огромный шаг в развитии науки, социальной жизни каждого жителя планеты [1].

Одним из наиболее актуальных научных направлений является развитие технологий искусственного интеллекта, автоматизации и автономной навигации автомобилей, мобильных роботов и колесных аппаратов в целом. Однако реализация таких масштабных устройств требует больших материальных, трудовых и временных ресурсов.

Одним из решений данной проблемы может стать создания миниатюрных копий устройств. Это позволяет студентам, любителям и профессионалам углубиться в изучение робототехники, мехатроники и систем навигации, вносить вклад в развитие данной отрасли новыми алгоритмами, научными исследованиями и моделированием систем. Такие исследования позволяют использовать опыт, полученный на миниатюрных устройствах, и масштабировать полученные модели и алгоритмы на более габаритных, совершенных и сложных системах.

**Цель работы** – усовершенствование системы управления мобильным колесным роботом с использованием элементов нечеткой логики и применением Web-технологий для навигации и обхождения препятствий.

**Актуальной задачей** выпускной квалификационной работы является разработка интеллектуальной системы управления мобильным колесным роботом с обхождением препятствий и использованием современных Web-технологий.

Задачи, поставленные в ходе выполнения выпускной квалификационной работы:

1. Анализ существующих научных трудов, разработок и патентов. Постановка задачи выпускной квалификационной работы.
2. Анализ и разработка математических моделей работы двигателя и движения мобильного робота в пространстве.
3. Анализ и разработка электронной компонентной базы и электрических схем мобильного робота.
4. Разработка нечеткого алгоритма управления роботом.

5. Моделирование полученных математических моделей двигателей и алгоритма управления роботом в ПО MATLAB Simulink.

6. Разработка системы управления и диспетчеризации на базе Web-технологий для взаимодействия с роботами.

7. Анализ полученных результатов моделирования, тестирование программного обеспечения.

# 1. Литературный обзор и постановка задачи

## 1.1 Краткий обзор литературы по моделям, алгоритмам управления и методам исследования автономных транспортных средств

В течение последнего десятилетия в связи с активным развитием систем искусственного интеллекта набирает большую популярность создание более точных и адаптивных алгоритмов и систем автономного управления транспортом. Современные исследования [1] располагают широким спектром вариантов использования различных алгоритмов: от простых компенсационных схем управления до нейро-сетевых технологий с использованием элементов нечеткой логики. Общим названием такого вида алгоритмов является SLAM (System of Localization and Mapping). Исследователи в ходе моделирования нескольких алгоритмов выявили следующие критерии оценивания: наличие ошибки рассогласования, точность на различных дистанциях, сложность алгоритма и влияние внешней среды. Перечисленные характеристики позволяют выбрать наиболее оптимальный алгоритм управления мобильным колесным роботом для поставленной задачи. Прежде всего необходимо учитывать воздействие человека на управление тележкой, поскольку достичь полной автономности транспортных средств без вмешательства человека практически невозможно. Функциональная схема навигационной системы, представленная в публикации, изображена на рисунке 1.1.

Часть представленных алгоритмов базируется на использовании нечеткой логики со схемой Мамдани, которая обеспечивает низкую сложность реализации и простоту настройки. Однако гарантий полного обхода препятствий такой алгоритм не дает.

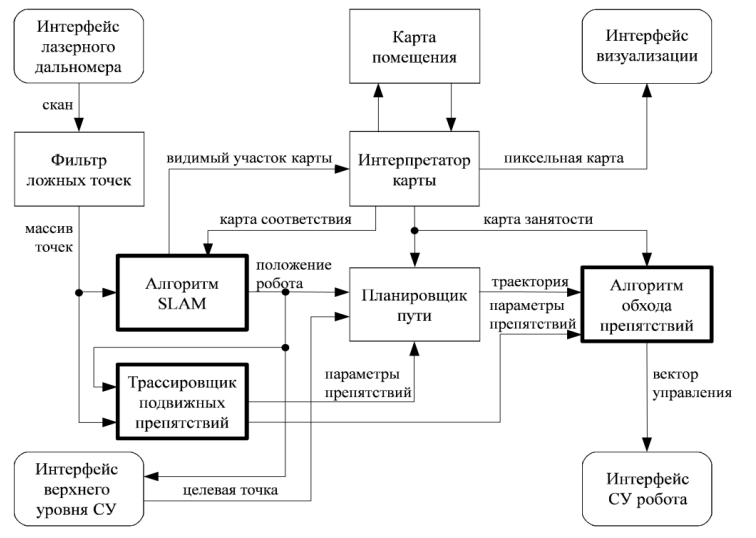


Рисунок 1.1 – Функциональная схема навигации мобильного колесного робота

Большая актуальность данного исследования заключается в описании диалогового управления и нейро-сетевых регуляторов. Системы такого типа имеют обучаемые базы знаний, которые адаптируют правила управления в динамическом режиме. Одним из наиболее значимых элементов диалогового управления является наличие аппарата распознавания речи, лингвистического анализатора.

Актуальность данных элементов управления заключается в интенсивном развитии качества распознавания речи с помощью алгоритмов искусственного интеллекта, что позволяет быстрее адаптироваться под различные интонации, языки. Однако у данной инновации есть отрицательная сторона – она требует большие вычислительные мощности и высокоэффективное оборудование для качественной обработки голосовой информации. В публикации авторы данное качестве не указали.

В статье [2] авторами представлена подробная математическая модель движения автономного робота по нелинейной и изменяемой траектории. Она содержит в себе элементы нелинейного динамического управления движением робота, адаптивных алгоритмов под изменяемую модель внешних воздействий: вода, асфальтированная дорога, грунт. Основная цель данного алгоритма – идентификация внешних воздействий на управление, адаптация под изменяемую среду и стабилизация объекта на трассе, снижая парирование отклонений от исходной траектории движения тележки.

В первую очередь авторы преобразуют нелинейные модели робота к системе задачно-ориентированных координат: выделяются матрицы связи угла вращения колес относительно центра масс, поступательное движение и расчет линейной и угловой скоростей в системе пространства состояний. Далее производится синтез алгоритма управления путем формирования матрицы состояния и управления тележкой, производится моделирование системы с влиянием шумовых характеристик и внешних воздействий. Авторы статьи выделяют желаемые параметры моделирования для достижения наилучших характеристик.

В публикации [3] авторы представляют подробную концепцию компонентно-ориентированной программной платформы для управления и работы автономного мобильного робота. На примере платформы RCE (Robot Components Engine) описаны требования к системе, функциональные возможности, а также технологии, реализованные в производстве. Одним из главных требований к разработке платформы являлись модульность, масштабируемость системы, то есть использование открытых протоколов и интерфейсов взаимодействия устройств между собой. Такая особенность, по мнению авторов, позволяет разделить функциональные элементы и структурировать взаимодействие между сенсорами, двигателями и внешней периферией.

Также авторы статьи разделяют архитектуры системы на несколько моделей: поведенческая, «очувствление-планирование-действие» и гибридную. В первом случае все компоненты системы располагаются на одном уровне и работают совместно через аппаратные или программные неразрывные соединения. Модель «очувствление-планирование-действие» предполагает декомпозированную структуру, в которой все элементы выстроены по вертикали: сбор данных сенсоров, формирование представления об окружающей среде исходя из используемой математической модели управления и передачу сигналов управления на исполнительные устройства. Таким образом, все элементы архитектуры имеют на каждом уровне свои функциональные возможности и задачи. Гибридная архитектура сочетает элементы как первой, так и второй модели.

Архитектура платформы RCE построена на базе гибридной архитектуры, в которой часть задач переносится не на элементы робота, а на компьютер-сервер. Такое размещение функций позволяет уменьшить нагрузку на сам робот и упростить как математическую модель управления, так и требования к колесному роботу. Задачи сервера заключаются в ведении журнала операций, удаленное построение маршрута для возвращения на базовую станцию в случае аварийной ситуации или снижения заряда аккумулятора, диспетчеризация и обработка сигналов для преодоления сложных препятствий с помощью нескольких сенсоров. При отправке сообщений между устройствами используются как синхронные, так и асинхронные способы передачи данных. Асинхронные задачи преимущественно используются для передачи информационных сообщений, не требующих обратного ответа от мобильного робота.

Преимущество данной платформы в том, что она построена на базе наиболее распространенного языка программирования C++, используемого в промышленных встраиваемых устройствах на базе микроконтроллеров. Язык программирования имеет компонентно-ориентированную парадигму, что позволяет разбивать задачи управления на функциональные блоки – компоненты, которые связаны по аппаратным или программным каналам связи. Исходный код программного обеспечения открыт для изучения студентами и инженерами, что позволяет дополнять или модернизировать ПО для различных аппаратных платформ.

Особую позицию занимают алгоритмы распознавания, обнаружения и объезда препятствий. Исследователи по всему миру детально и глубоко изучают данное научное направление. Так, в публикации [4] автор проводит детальный анализ существующих алгоритмов.

Наиболее распространенными считаются алгоритмы Bug 2-го поколения (далее Bug2), VFH. Алгоритм Bug 1-го поколения (далее Bug) в сравнении с 2-м имеет один существенный недостаток – для преодоления препятствия алгоритм предусматривает объезд препятствия по всему периметру, что создает дополнительные временные, ресурсные затраты.

В алгоритме Bug2 данный недостаток решается таким образом: мобильный робот объезжает препятствие по левую сторону, после чего продолжает движение по заданной прямой для достижения цели. Таким образом, если объезд по левую сторону будет иметь более длинный путь в сравнении с объездом по правую, то этот фактор не будет учитываться алгоритмом Bug2. Такое свойство является недостатком алгоритма Bug2. Для его решения был разработан алгоритм VFH на основе вероятностного подхода с использованием полярных гистограмм для построения новой траектории движения робота с объездом препятствия и возвращения на исходную.

Алгоритм VFH использует методы математической оптимизации для нахождения локального минимума при расчете расстояния, находясь рядом с препятствием. Точность и качество работы алгоритма VFH заметно повысились, особенно это заметно при передвижении робота между двумя препятствиями.

## 1.2 Патентная проработка видов и конструкций малогабаритных автономных транспортных средств

В ходе исследования зарегистрированных патентов было обнаружено более десятка тысяч исследований на территории Российской Федерации и стран СНГ с помощью сервиса Яндекс.Патенты.

С помощью сервиса Google Патенты по всему миру было найдено свыше сотни тысяч патентов. Были исследованы автономные транспортные средства, мобильные колесные роботы и их компоненты, в т.ч. алгоритмы управления движением тележки.

Наибольшее количество зарегистрированных работ находятся в КНР, Японии, США и т.д. Большинство было разработано крупными корпорациями. Среди стран СНГ наибольшее число работ зарегистрировано в ВУЗах и НИИ Российской Федерации, Казахстана и Республики Беларусь. Первые патенты были зарегистрированы в 1996 году в США, на данный момент большую популярность данная отрасль получила во второй половине десятилетия 2010-х годов.

В ходе выполнения выпускной квалификационной работы и анализа патентов были выбраны патенты из категорий алгоритмов управления колесным роботом и конструкторских изобретений [5,6]. Они наиболее близки к теме работы, что в значительной степени повлияло на ход постановки цели и задач выпускной квалификационной работы.

## 1.3 Постановка задачи

В результате проведения литературного обзора и патентного исследования в качестве объекта исследования выпускной квалификационной работы была выбрана модель гусеничного мобильного транспортного средства типа робот, управление которого производится автоматически без участия человека. Предметом исследования являются математические модели, методы и алгоритмы управления автономного транспортного средства.

Цель исследования – повышение точности и качества алгоритма управления тележкой при объезде препятствий с учетом рельефа дорожного покрытия.

Задача исследования: для указанной модели двухколесного мобильного робота необходимо разработать систему управления тележкой, которая будет манипулировать поворотом колес и скоростью движения тележки при объезде препятствий, учитывая качество дорожного полотна.

Данная задача будет решаться в два этапа. В первую очередь будет проводиться синтез управления, который обеспечивает при учете возмущений дорожного покрытия [9] стабилизацию движения робота в заданном направлении с постоянной скоростью.

Для решения поставленной задачи будет получена математическая модель движения робота с учетом изменения скорости, направления движения и заданного внешнего возмущения профиля дорожного покрытия.

Далее будет представлен синтез управления робота для обхода препятствий с помощью измерительных устройств при автономном движении.

# 2. Математические модели работы двигателя постоянного тока и движения тележки в пространстве.

Тележка представляет собой робота, собираемого руками человека, размером 20x20 см. Он содержит в себе два электрических двигателя, два ведущих колеса, а также два энкодера для вращения ведущих колес влево-вправо. В качестве измерительных приборов, а именно измерения расстояния до препятствия от робота в нем применяются 3 ультразвуковых дальномера, расположенных слева, впереди и справа от центра масс на роботе. Синтез трех датчиков препятствия обуславливается достижением наиболее точной работы алгоритма, а также движения в пространстве для достижения заданной цели.

**2.1 Математическая модель двигателя постоянного тока робота.**

Двигатель является главным звеном в работе тележки. Для его точного функционирования следует построить адекватную математическую модель [7].

Составим уравнение Лагранжа-Эйлера, а также учтем влияние противо-ЭДС в обмотке ротора двигателя постоянного тока (ДПТ).

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

где - угловая скорость вращения мотора, - скорость холостого хода, - момент, развиваемый двигателем, - пусковой момент и - момент инерции ротора двигателя.

Решением данного дифференциального уравнения является функция

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

где - электромеханическая постоянная двигателя.

Функция измерения угла во времени будет интегралом функции угловой скорости с учетом начальных условий.

|  |  |  |
| --- | --- | --- |
|  |  | (2.3) |

Данная система уравнений описывает электрические процессы, протекающие в ДПТ.

|  |  |  |
| --- | --- | --- |
|  |  | (2.4) |

где , - конструктивные постоянные, U – управляющее напряжение, L – индуктивность обмотки двигателя, I – пусковой ток. - сопротивление обмотки ротора.

Коэффициент - коэффициент передачи по току. Для его расчета применяется выражение, вычисляющее отношение момента на выходном валу на пусковой ток.

|  |  |  |
| --- | --- | --- |
|  |  | (2.5) |

Коэффициент - коэффициент передачи по току. Для его расчета применяется выражение, вычисляющее отношение поданного напряжения на выходном валу на заданную угловую скорость.

|  |  |  |
| --- | --- | --- |
|  |  | (2.6) |

Таким образом, для расчета пускового момента, выдаваемого двигателем постоянного тока (ДПТ), формула примет такой вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.7) |

**2.2 Математическая модель управления роботом-тележкой**

Проведем разработку математической модели мобильного колесного робота с использованием ранее описанной математической модели двигателя постоянного тока. В состав мобильного колесного робота для описания математической модели с дифференциальным приводом входят: два двигателя постоянного тока с установленными к ним колесами, гироскоп и 3 ультразвуковых датчика [8].

Расчет пройденного расстояния будет производиться с помощью энкодеров, расположенных на валу двигателя, для ориентации робота используются гироскоп, измеряющий угловую скорость вращения робота относительно центра масс.

На рисунке 2.1 изображена кинематическая схема мобильного колесного робота.

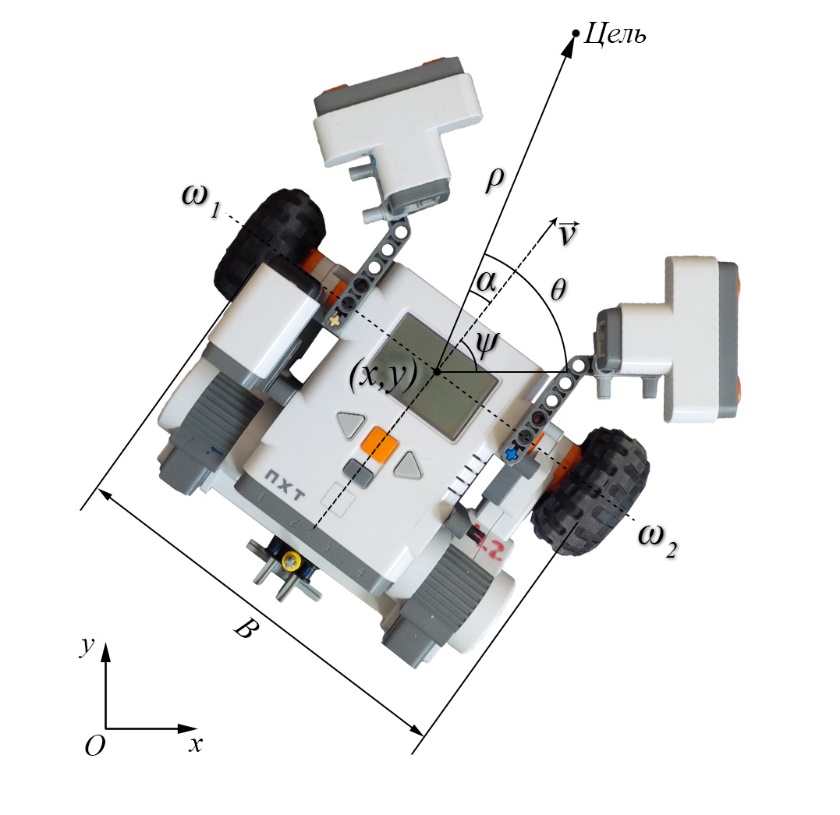


Рисунок 2.1 – Кинематическая схема мобильного колесного робота

На рисунке указаны следующие обозначения: расстояние до целевой точки, азимут, угол между осью OX и направлением на цель, – курс робота, курсовой угол, разность между курсом и азимутом, линейная скорость робота. Движение робота осуществляется за счет двух независимо управляемых колес. Математическим описанием кинематической схемы, проиллюстрированной выше, является:

|  |  |  |
| --- | --- | --- |
|  |  | (2.8) |

где , – угловые скорости вращения колес, – координаты движения робота, R – радиус колеса, B – ширина колесной базы.

Для решения задачи навигации робота нам необходимо достигать координат: , то есть расстояние до цели и курсовой угол должны достичь нуля. В рамках текущей работы используем локальный подход навигации.

Локальный подход – определение координат устройства по отношению к начальной точке. Планирование задает небольшой отрезок траектории, в конечной точке которого выбирается дальнейшая траектория движения робота [8].

Опишем математическую модель навигации робота к цели в полярных координатах:

|  |  |  |
| --- | --- | --- |
|  |  | (2.9) |

Для управления роботом необходимо найти такие значения угловой и линейной скоростей , при которых будет выполняться условие ранее поставленной задачи, а именно достижение координат: . Для этого воспользуемся функцией Ляпунова. Такая квадратичная функция, включающая в себя расстояние до цели и курсовой угол:

|  |  |  |
| --- | --- | --- |
|  |  | (2.10) |

Производная по времени отрицательна определена, поскольку расстояние до цели и курсовой угол не должны возрастать. Производная имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.11) |

Выразим производную через математическую модель управления робота:

|  |  |  |
| --- | --- | --- |
|  |  | (2.12) |

Эта производная будет отрицательно определена, если в качестве управляющего воздействия выберем следующие значения скоростей:

|  |  |  |
| --- | --- | --- |
|  |  | (2.13) |

Вышеуказанные значения позволяют достичь роботу своей цели, но без учета внешних препятствий. Для избегания препятствий вводим поправку для вычисления курсового угла:

|  |  |  |
| --- | --- | --- |
|  |  | (2.14) |

где – коэффициент пропорциональной составляющей курсового угла, – расстояние до препятствия, – минимальное расстояние до препятствия.

**2.3 Характеристика микро-профиля дороги и его математическая модель.**

В качестве внешнего возмущения, приложенного к управлению мобильного колесного робота, возьмем влияние случайного микро-профиля дороги на колебания робота во время движения. Охарактеризовать такое возмущение возможно с помощью корреляционной функции и спектральной плотности. Методы статистической динамики позволят охарактеризовать реакцию подвески и элементов управление на это воздействие такими же характеристиками корреляционной функции или спектральной плотности изменения координат системы или их производных [8].

Микро-профиль дороги описывается нормированной корреляционной функцией вида:

|  |  |  |
| --- | --- | --- |
|  |  | (2.15) |

где – время проезда неровности, скорость машины, – коэффициенты корреляционной связи, безразмерные коэффициенты (причем ).

Передаточная функция микро-профиля дороги определяется следующим выражением:

|  |  |  |
| --- | --- | --- |
|  |  | (2.16) |

Ниже в таблице 2.1 представлены рассчитанные значения микро-профиля дороги для трех видов дорожного покрытия: грунтовая, булыжная и асфальтобетонная. Каждая из них отличается качеством покрытия, устойчивостью к изменению погодных условий и колебаниями почвенного слоя [9]. Рассчитанные значения были получены экспериментальным путем при исследовании влияния дорог на подвеску и движения автомобиля [9].

Таблица 2.1 - Рассчитанные параметры микро-профиля дороги

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Вид дороги |  |  |  |  |  |  |
| Грунтовая | 0,634 | 0,366 | 0,017 | 0,153 | 0,144 | 110 |
| Булыжная | 0,8 | 0,2 | 0,32 | 0,2 | 0 | 10,7 |
| Асфальтобетонная | 1 | 0 | 0,14 | 0 | 0 | 0,25 |

# 3. Электрическая схема и элементная база колесного робота

## 3.1 Описание электрических компонентов и соединений блока управления роботом

Мобильный робот представляет собой робот~~-~~автомобиль, имеющий 2 гусеницы, 2 коллекторных мотора постоянного тока, аккумулятор, блок управления колесным аппаратом, датчики приближения [10].

Каждый компонент системы блока управления состоит из ряда базовых электронных компонентов, соединяемых между собой на печатных платах. Рассмотрим наиболее важные узлы подробнее.

1. Адаптер Bluetooth HC-06.

Данное устройство состоит из микросхемы HC-05, 4 триггеров Шмидта 74LVC2G17, буферного элемента 74LVC1G125DBV на базе RS-триггера с 3 состояниями («низкое», «высокое» и Z-состояние) и линейного стабилизатора напряжения TPS73033DBV с фиксированным выводом напряжения 3.3В. Микросхема содержит 34 вывода: первые два – RX/TX-выводы для передачи данных, 11 – вывод на сброс настроек, 12 – питание микросхемы 3.3В, 13, 22 – нулевой потенциал. 24, 31, 32 выводы – управление светодиодной оснасткой. 26, 33 – элементы подключения с внешней антенной. 34 вывод соединен с буферным элементом, который шунтируется резистором на 10 КОм. Для связи с устройствами по технологии Bluetooth в окружающей среде на плате размещается антенна.

На схеме представлены триггеры Шмидта для создания канала передачи данных по технологии Bluetooth. Первые два триггера на вход принимают внешний сигнал RX, который распределяется по двум триггерам: первый сигнализирует светодиодом и подает напряжение на вход +3.3В, второй – передает данные на 2 вывод микросхемы HC-06 для приема данных с внешнего устройства. Следующие два триггера принимают на вход сигнал TX с микросхемы HC-06 и выводят их через первый триггер на внешний канал связи, а по второму – сигнализируют светодиодом и подают напряжение на вход +5В.

Также рассмотрим линейный стабилизатор напряжения TPS73033DBV на +3.3В. Он содержит 5 выводов: входные линии – IN (напряжение +5В) и EN, подающие сигнал включения стабилизатора, ножка нулевого потенциала GND, ножка выходного напряжения +3.3В OUT. На вход подается напряжение +5В, шунтированное конденсатором на 100 нФ. Выходное напряжение выравнивается конденсаторами на 2.2 мФ и 100 нФ в пределах +3.3В.

На рисунке 3.1 представлено изображение Bluetooth-адаптера HC-06.

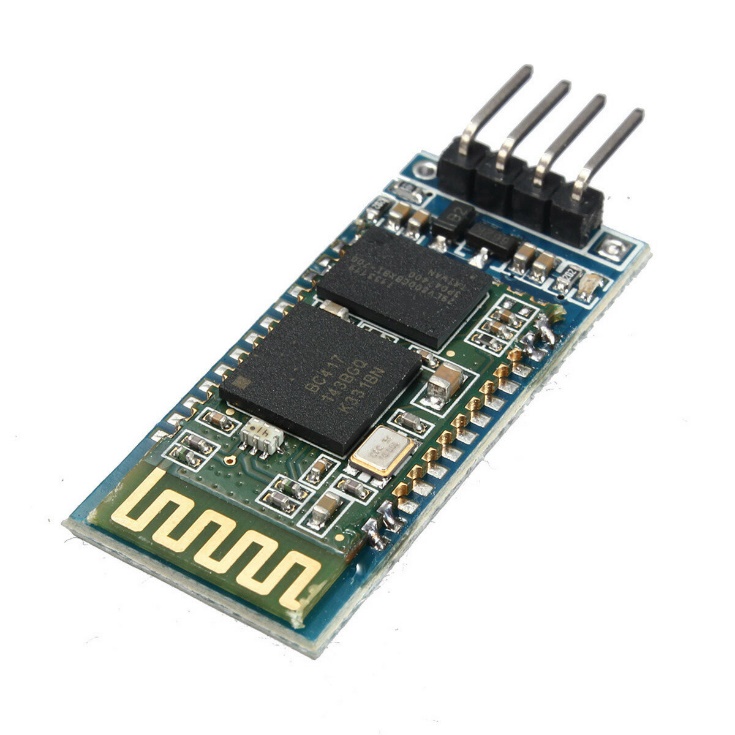


Рисунок 3.1 – Bluetooth-адаптер HC-06

2. Ультразвуковой дальномер HC-SR04.

Данное устройство состоит из набора управляющих микросхем EM78P153S, MAX232, 4-х операционных усилителей LM324, принимающего и передающего звукового устройства.

Микросхема EM78P153S является 8-битным микропроцессором с низким потреблением, изготовленным по технологии КМОП. Содержит 14 контактов, 2 из которых подключаются к линиям питания и корпуса +5В и GND. На схеме дальномера к выводам P67 и P50 подключаются контакты, синхронизирующие связь с внешним микроконтроллером (в данном случае с Arduino Uno), при этом вывод P67 при длительном отсутствии сигнала переводит дальномер в режим пониженного энергопотребления. Выводы P60, P61 принимают сигналы с операционных усилителей. По выводам P51, P52 передаются сигналы на выводы микросхемы MAX232.

Микросхема MAX232 – преобразователь сигналов порта RS-232 в цифровые сигналы (например, UART) в системах на базе КМОП-технологий. Имеет большое распространение в промышленной микроэлектронике. На схеме дальномера HC-SR04 микросхема MAX232 принимает сигналы с выходов P51, P52 микросхемы EM78P153S на входы интерфейса UART T1IN, T2IN. С помощью управляющих сигналов, поступающих на входы C2+ и VS от компаратора на базе операционного усилителя LM324, который получает сигнал с приемника, микросхема MAX232 преобразует полученный сигнал в сигнал по интерфейсу RS232 на выходной ультразвуковой передатчик, выполненный в виде звукового устройства.

Операционный усилитель LM324 4-канальный на схеме дальномера выполняет следующие функции: усилитель сигнала, фильтр полосы пропускания и компаратор. Полученный с приемника сигнал усиливается, далее с помощью последовательной RC-цепи (конденсатор на 1 нано-фарад и резистор на 6.2 кОм) и операционного усилителя проводится фильтрация низких частот. После чего сигнал вновь усиливается и сравнивается с помощью компаратора для подачи сигнала об обнаружении препятствия на микросхемы MAX232 (выводы VS и C2+) и EM78P153S (выводы P60, P61). На схеме размещается 4 последовательно соединенных операционных усилителя LM324 для обработки входных сигналов с приемника [11].

На рисунке 3.2 показано изображение аппарата – ультразвукового датчика HC-SR04.

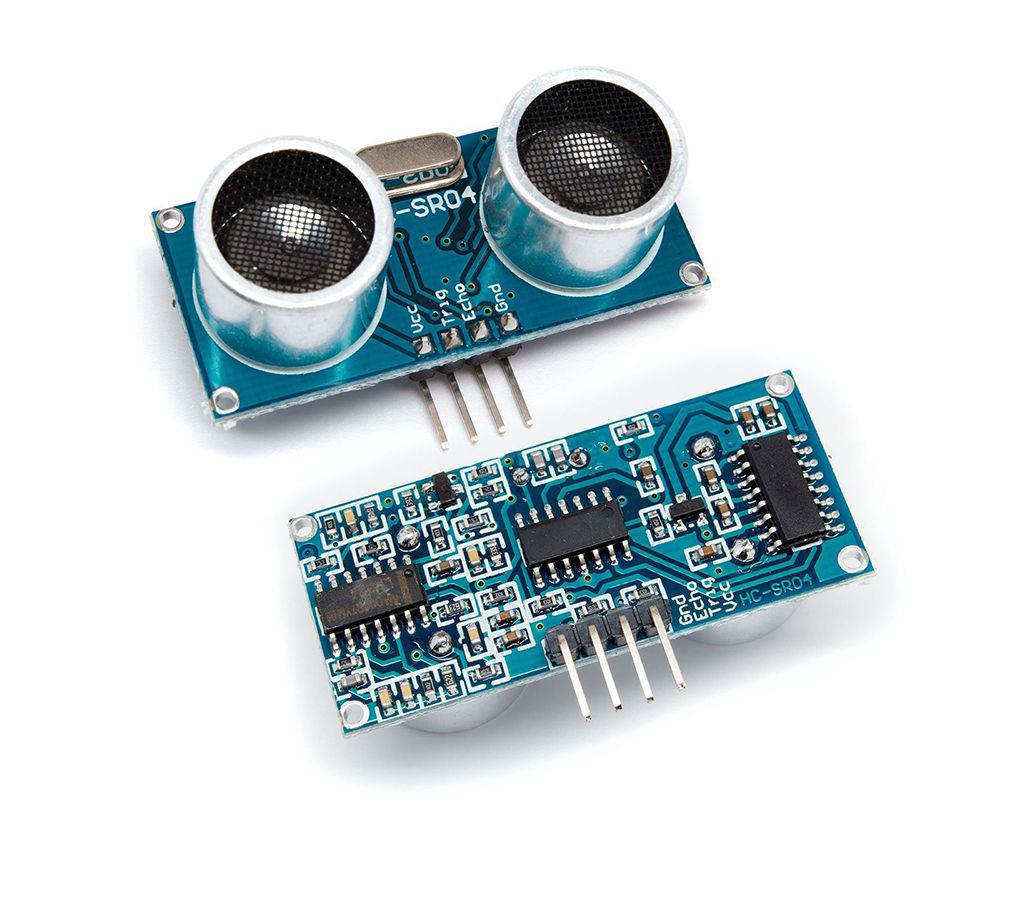


Рисунок 3.2 – Ультразвуковой дальномер HC-SR04

3. Двухканальный модуль управления моторами.

Данное устройство выполнено в виде печатной платы, повторяющей размеры микроконтроллера Arduino Uno с идентично расположенными выводами. Содержит 1 вход для питания платы Motor Shield в диапазоне от +5 до +12В с возможностью параллельного питания платы Arduino Uno, 2 выхода для подключения внешних электродвигателей с напряжением от +5 до +12В. Плата состоит из двойного полномостового драйвера L298P, 2 инвертеров HC1G04, а также 3 транзисторных усилителей и диодного моста из 8 диодов.

Двойной полномостовой драйвер двигателя L298P выполнен в виде микросхемы с 20 выводами, 2 из которых не имеют подключения, 4 вывода – вывод на корпус (GND) и два вывода питания – управляющее питание +5В и +12В для управления внешними подключаемыми устройствами. Диапазон входного напряжения для управления внешними устройствами лежит от +5В до +12В. При увеличении порогового напряжения есть вероятность выхода микросхемы из строя. К выводам Enable A, Enable B подключены соединения для управления двигателями постоянного тока, к выводам OUT1, OUT2, OUT3, OUT4 подключены по 2 контакта подключаемых электродвигателей [12].

На рисунке 3.3 изображен модуль Motor Shield, подключенный к плате Arduino Uno.



Рисунок 3.3 – Модуль управления внешними моторами Motor Shield

4. Плата-микроконтроллер Arduino Uno.

Данное устройство состоит из набора 8-битных AVR-микроконтроллеров ATMEGA328P и ATMEGA16U2, микросхемы управления питанием NCP1117ST50T3G, 2-х операционных усилителей LMV358IDGKR, стабилизатора напряжения LP2985-33.

Микросхема ATMEGA328P является 8-битным микроконтроллером. На схеме платы-микроконтроллера является основным управляющим контроллером. Содержит 28 контактов, 2 из которых подключаются к шинам питания +5В и еще 2 контакта - к GND, 2 контакта программирования через интерфейс SPI (вход MOSI и выход MISO) и 1 контакт синхронизирующего входа. К выводам TOSC1 и TOSC2 подключен тактовый генератор c частотой 16 МГц.

Микросхема ATMEGA16U2 является 8-битным микроконтроллером. На схеме платы-микроконтроллера Arduino Uno является микросхемой управления ввода, перепрограммирования и преобразователем USB -> UART. Содержит 28 контактов, 2 из которых подключаются к шинам питания +5В и еще 2 контакта - к GND, а 4 контакта используются для подключения к внешнему USB выходу. На схеме микроконтроллера к выводам INT2, INT3 подключены выводы RX и TX микроконтроллера ATMEGA328P для синхронизации между двумя управляющими микросхемами. К выводам XTAL1 и XTAL2 подключен тактовый генератор c частотой 16 МГц [8].

Операционный усилитель LMV358IDGKR 2-канальный, исполненный в виде микросхемы, на схеме платы-микроконтроллера выполняет следующие функции: индикация состояния платы после синхронизации с загруженным программным обеспечением и сравнение уровней напряжения между питанием интерфейса USB и подключенным гальваническим элементом (аккумулятор или литиевая батарея) с напряжением +5В. На схеме операционные усилителя размещаются в областях управления питанием платы и сигнализации об успешном запуске с помощью светодиода [13].

Стабилизатор напряжения NCP1117ST50T3G выравнивает напряжение на уровне +5В для подключения с помощью гальванического элемента (аккумулятор или батарея).

На рисунке 3.4 показано изображение аппарата – плата-микроконтроллер Arduino Uno.

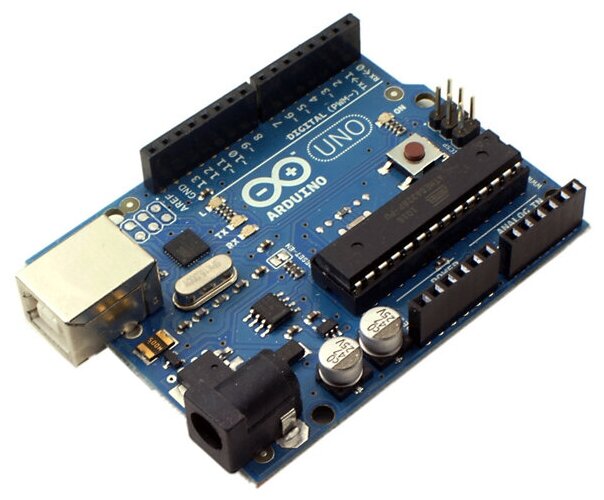


Рисунок 3.4 – Плата-микроконтроллер Arduino Uno

5. Двигатель постоянного тока JGB37-3530.

Электродвигатель с входным постоянным напряжением +12В и скоростью 166 об/мин имеет редуктор с передаточным отношением 56 и длиной 24 мм. Вес двигателя 188 г. На рисунке 3.5 представлено его изображение.



Рисунок 3.5 – Электродвигатель постоянного тока JGB37-3530

## 3.2 Электрическая функциональная схема блока управления

Электрическая функциональная схема (Э2) состоит из четырех функциональных частей: микроконтроллер Arduino Uno, Bluetooth-адаптер HC-06, ультразвуковой дальномер HC-SR04 и модуль управления электродвигателями Motor Shield. Функциональные части подключаются к микроконтроллеру по функциональной цепи [13, 14].

Внешними устройствами, подключаемыми к функциональным частям являются:

1) Батарея с номинальным напряжением +5В – питание микроконтроллера Arduino Uno;

2) Батарея с номинальным напряжением +12В – питание модуля управления электродвигателями Motor Shield;

3) Коллекторный электродвигатель с напряжением +12В и мощностью 166 об/мин. Подключается к плате Motor Shield в количестве 2 шт.

Электрическая функциональная схема представлена в Приложении А.

## 3.3 Электрическая принципиальная схема блока управления

Электрическая принципиальная схема (Э3) содержит подробное представление электрических соединений мобильного робота [12, 14]. Она, как и электрическая функциональная схема, состоит из 4 основных блоков: управление микроконтроллером (Arduino Uno), управление электродвигателями (Motor Shield), управление Bluetooth-соединением (HC-06) и измерительное устройство – ультразвуковой дальномер HC-SR04.

1. Блок управления Bluetooth-соединением.

На схеме контакты +5В и GND подключаются к общей шине электропитания с соответствующими потенциалами. Контакты RX и TX подключаются к контактам TXD и RDX блока управления микроконтроллером (Arduino Uno) для связи между устройствами.

Bluetooth-адаптер состоит из 4 триггеров Шмидта 74LVC2G17, 2 светодиодов желтого цвета с номинальным напряжением +5В, 1 светодиода красного цвета с номинальным напряжением +5В, 1 светодиода зеленого цвета с номинальным напряжением +3.3В, 1 регулятора напряжения TPS73033DBV с выходным напряжением +3.3В, 3 резисторов с сопротивлением 470 Ом, 2 резисторов с сопротивлениями 10 кОм и 1 кОм, внешней антенны и 3 конденсаторов с емкостями 100 нанофарад, 10 нанофарад и 2.2 микрофарад, соответственно.

2. Блок управления электродвигателями.

На схеме к выводам D4-D7 блока управления микроконтроллером подключены выводы платы управления электродвигателями Motor Shield для управления вращением колес (DIR12, DIR34) и поворотом по продольной оси (EN12, EN34). К контактам питания платы +12В и GND подключается внешний гальванический элемент (батарея или аккумулятор) с напряжением от +5 до +12В. Выводы платы M11, M12 и M21, M22 используются для приведения в действие внешних электрических двигателей постоянного тока.

Устройство состоит из 6 резисторов с сопротивлением 1 кОм, 5 резисторов с сопротивлением 4.7 кОм; 7 светодиодов красного цвета, 6 КМОП-транзисторов типа 2N7002, 8 диодов SS14, микросхемы управления двигателями L298P, двух инверторов HC1G04, двух конденсаторов с емкостью 100 микрофарад и одного конденсатора с емкостью 100 нанофарад.

3. Измерительное устройство HC-SR04.

Для подключения ультразвукового датчика HC-SR04 используются контакты D10 и D11 блока. Питание устройства осуществляется с помощью подключения к общей шине питания с разными потенциалами (+5В и GND) [15].

Устройство состоит из 5 резисторов с сопротивлением 10 кОм, 3 резисторов с сопротивлением 75 кОм, резисторов с сопротивлениями 1 кОм, 3.9 кОм, 4.7 кОм, 6.2 кОм, 7.5 кОм, 56 кОм и 300 кОм; 3 конденсаторов с емкостью 10 нанофарад, 2 конденсаторов с емкостью 1 нанофарад; 4 операционных 4-канальных усилителей LM324, а также микросхем управления MAX232, EM78P153S. Излучателем является пьезодинамик, а приемным устройством – пьезомикрофон. К микросхеме EM78P153S подключен кварцевый генератор с частотой 27 МГц.

4. Блок управления микроконтроллером.

Как было ранее описано, блок управления микроконтроллером состоит из двух управляющих микросхем – 8-битных микроконтроллеров ATMEGA16U2 и ATMEGA328P. Первая из них производит управление питанием и шиной USB, а вторая – управление внешними устройствами, подключенными к контактам микросхемы. Питание микросхемами осуществляется либо с помощью гальванического элемента (батарея или аккумулятор), либо за счет подключения питания к разъему USB. Переключение между режимами питания осуществляется с помощью КМОП-транзистора.

Устройство состоит из 4 резисторов с сопротивлением 1 кОм, 3 резисторов с сопротивлением 10 кОм, 2 резисторов с сопротивлением 100 кОм, 2 резисторов с сопротивлением 22 Ом, 2 варисторов CG0603MLC-05E, ферритового сердечника-фильтра BLM21, 2 кварцевых генераторов с частотой 16 МГц, 8 конденсаторов с емкостью 100 нанофарад, 2 конденсаторов с емкостью 1 микрофарад; 2 поляризованных конденсатора с емкости 47 микрофарад, КМОП-транзистор FDN340P, кнопки-переключателя, 3 диодов типа CD1206-S01575, 1 светодиод желтого цвета. На плате имеются два операционных усилителя LMV358IDGKR, микросхемы NCP1117ST50T3G, LP2985-33DBVR для управления питанием микроконтроллера. Управляющими микросхемами микроконтроллера являются ATMEGA328P и ATMEGA16U2. Для подключения к шине USB на плате имеется разъем, обозначенный как X1\_USB.

Общая электрическая принципиальная схема представлена в приложении Б в формате A0. Перечень элементов представлен в приложении В на 3-х листах в формате А4 [11, 14].

# 4. Интеллектуальный нечеткий алгоритма управления роботом

Нечеткая логика – раздел математики, использующий методы классической булевой логики, но расширенный с помощью семантических множеств, называемых лингвистическими переменными. В теории управления использование методов нечеткой логики позволяет разработать гибкий и несложный регулятор. Он широко используется в системах автоматики, робототехнике и приборостроении [16].

Суть данного подхода заключается в использовании вместо понятий «истина» (), «ложь» () из классической математической логики некие заданные множества значений (). Данное свойство алгоритмов нечеткой логики позволяет расширить диапазон возможных величин, подаваемых на вход и выход систем, и убрать возможные ограничения. На входных и выходных лингвистических переменных задаются диапазоны возможных состояний. Состояния выбираются с помощью функции принадлежности. Для формирования логических выводов разрабатывают правила связи входных и выходных величин [17,20].

Ниже в таблице 4.1 представлены состояния входных лингвистических переменных.

Таблица 4.1 – Входные функции принадлежности

|  |  |
| --- | --- |
| Наименование функции | Значения логической переменной |
| – «препятствие очень близко» | {0,…, 20} |
| *–* «препятствие близко» | {21, …, 30} |
| *–* «препятствие отсутствует» | {31, …, +∞} |

В разрабатываемом регуляторе на основе нечеткой логики будет 3 входных функции принадлежности с идентичным набором значений. Они применяются для измерительных устройств – дальномеров, установленных на корпусе колесного робота. Таким образом, количество переменных будет равно 9, они представлены в матрице входных логических переменных (4.1):

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

По горизонтали расположены логические переменные каждого дальномера, по вертикали – номер логической переменной по каждому дальномеру.

Функция принадлежности «препятствие очень близко» выполняется в тот момент, когда расстояние от датчика до препятствия находится в диапазоне от 0 до 20 см.

Функция принадлежности «препятствие близко» выполняется в тот момент, когда расстояние от датчика до препятствия находится в диапазоне от 20 до 30 см.

Функция принадлежности «препятствие отсуствует» выполняется в тот момент, когда расстояние от датчика до препятствия более 30 см. Это соответствует тому, что движению робота ничего не мешает.

На рисунке 4.1 представлены входные функции принадлежности и наборы логических переменных для дальномера в виде графика в ПО MATLAB Fuzzy Logic Designer. Значения для остальных дальномеров идентичные.

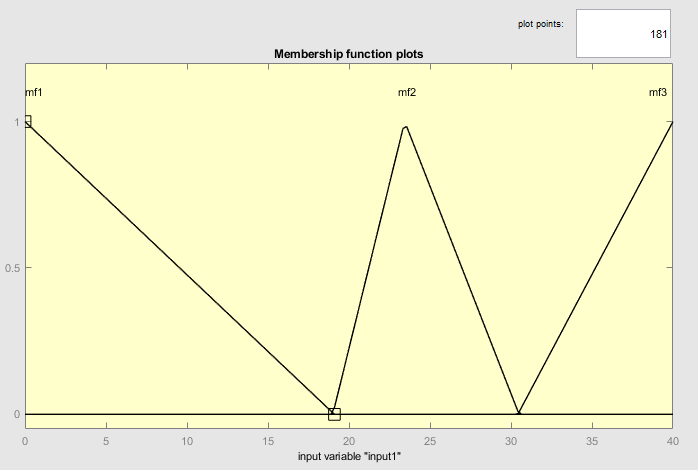


Рисунок 4.1 – Функции принадлежности входных переменных нечеткого регулятора

Далее рассчитаем возможные нечеткие логические выводы регулятора, представленные в таблице 4.2.

Таблица 4.2 – Выходные функции принадлежности.

|  |  |
| --- | --- |
| Наименование функции | Набор значений переменной |
| – «движение в обратном направлении» | {-0.6, -0.5, -0.4} |
| *–* «остановка двигателя» | {-0.05, 0, 0.05} |
| *–* «движение с малой мощностью (1/2 от полной)» | {0.4, 0.45, 0.5, 0.55, 0.6} |
| *–* «движение с полной мощностью» | {0.61, 0.7, 0.8, 0.9, 1} |

В разрабатываемом регуляторе на основе нечеткой логики будет 2 выходных множества с идентичным набором значений. Они применяются для исполнительных устройств – двигателей постоянного тока, установленных на корпусе колесного робота. Таким образом, количество переменных будет равно 8, они представлены в матрице выходных логических переменных (2.2):

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |

По горизонтали расположены логические переменные одного дальномера, по вертикали – количество дальномеров.

На рисунке 4.2 представлены значения выходных логических переменных для двигателя постоянного тока, расположенного на левой стороне шасси, в виде графика в ПО MATLAB Fuzzy Logic Designer. Значения для двигателя на правой стороне шасси идентичные.

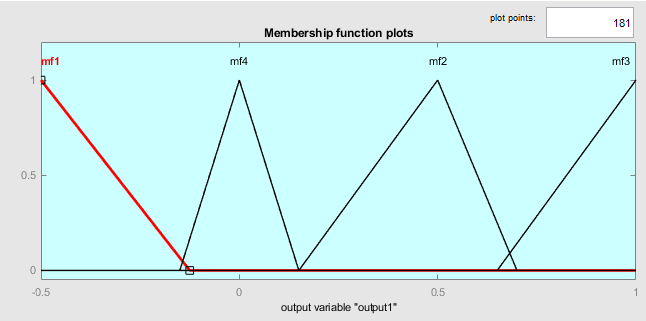


Рисунок 4.2 – Значения выходных переменных нечеткого регулятора

Для формирования связей между входными и выходными логическими терминами разрабатывается база правил нечеткого регулятора в форме «Если – То». Цель правила – создание четкого выбора значений переменных на основе наложенных на входную или выходную переменную ограничений. Между входными и выходными переменными могут использоваться логические операции «И» или «ИЛИ» [17].

У каждой выходной лингвистической переменной должно быть хотя бы одно правило. Каждое правило обозначается символом , где i – индекс правила, R – правило нечеткого регулятора. База правил представлена в таблице 4.3.

Таблица 4.3 – База правил нечеткого регулятора.

|  |  |  |
| --- | --- | --- |
| Входные лингвистические переменные | Выходные лингвистические переменные | Сформированное правило |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Ниже представлено краткое описание каждого правила в соответствии с показаниями датчиков и мощностей двигателей постоянного тока [16,18].

– правило «Задний ход». Если 3 дальномера одновременно получают сигналы о расстоянии менее 20 см, то мы можем предположить, что робот попал в тупик и ему необходимо выехать из него задним ходом.

правило «Препятствие слева». Если дальномер, расположенный на левом боку робота, показывает значение менее 20 см, то система управления блокирует поворот правого колеса двухколесного робота для совершения поворота налево.

– правило «Препятствие впереди». Если дальномер, расположенный на переднем борту робота, показывает значение менее 20 см, то система управления блокирует дальнейшее движение двух колесного робота.

правило «Препятствие справа». Если дальномер, расположенный на правом боку робота, показывает значение менее 20 см, то система управления блокирует поворот левого колеса двухколесного робота для совершения поворота направо.

правило «Препятствия отсутствуют». Каждый дальномер принимает значение более 30 см. Робот движется прямо на полной мощности.

правило «Препятствие слева приближается». Если дальномер, расположенный на левом боку робота, показывает значение менее 30 см, то система управления снижает скорость двух двигателей, чтобы обеспечить плавное вхождение в поворот при приближении к препятствию слева.

правило «Препятствие спереди приближается». Если дальномер, расположенный на передней части робота, показывает значение менее 30 см, то система управления снижает скорость двух двигателей, чтобы обеспечить плавное торможение робота.

правило «Препятствие справа приближается». Если дальномер, расположенный на правом боку робота, показывает значение менее 30 см, то система управления снижает скорость двух двигателей, чтобы обеспечить плавное вхождение в поворот при приближении к препятствию справа.

правило «Препятствие слева и спереди». Если дальномеры, расположенные на левом боку и в передней части робота, показывают значение менее 20 см, то система управления блокирует поворот правого колеса двухколесного робота для совершения поворота налево.

правило «Препятствие справа и спереди». Если дальномеры, расположенные на правом боку и в передней части робота, показывают значение менее 20 см, то система управления блокирует поворот левого колеса двухколесного робота для совершения поворота направо.

На рисунке 4.3 представлена база правил нечеткого регулятора в окне Rule Editor в ПО MATLAB Fuzzy Logic Designer. В качестве типа нечеткого вывода используется алгоритм Мамдани. Выбор данного типа обусловлен низкую сложность реализации и простоту настройки регулятора.

В верхней части данного окна представлены созданные правила нечеткого регулятора. В ниже части расположены блоки входных (первые три) и выходных (два последних) функций принадлежности. Выбрав из каждого блока функции, после нажатия Add role новое правило добавляется в базу. С помощью кнопки Change role можно изменить параметры существующего правила, а кнопка Delete role удаляет выбранное правило. Блок Connection определяет логическую операцию формирования правила. Среди операций могут быть «И» и «ИЛИ».

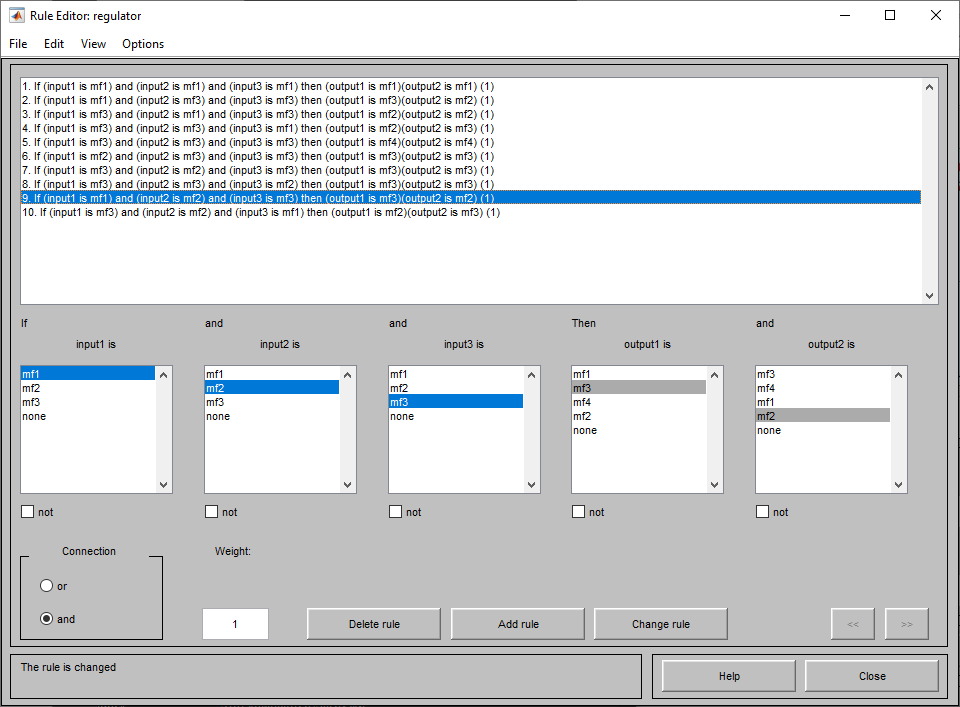


Рисунок 4.3 – База правил нечеткого регулятора

# 5. Моделирование системы управления роботом с нечетким регулятором

Произведем моделирование системы управления мобильным колесным роботом в прикладном ПО MATLAB Simulink. Для этого нам необходимо использовать элементы модели двигателя постоянного тока, представленной в п. 1.1 главы 1, а именно: коэффициент передачи противоЭДС, передаточная функция системы. Модель системы управления представлена в Приложении В.

Модель управления колесным мобильным роботом состоит из следующих элементов: 2 двигателя постоянного тока, блок вычисления линейной и угловой скоростей, вычисление координат поворота тележки, курса робота, внешнее возмущение – микро-профиль дорожного покрытия и нечеткий алгоритм управления. Микро-профиль описан в п. 2.3, нечеткий алгоритм управления представлен в главе 4.

В таблице 5.1 представлены характеристики робота, дополняющие характеристики двигателя и применяемые в ходе моделирования в среде MATLAB Simulink. Файл с параметрами робота представлен в Приложении Г.

Таблица 5.1 – Дополнительные характеристики робота.

|  |  |
| --- | --- |
| Наименование параметра (ед. измерения по СИ) | Значение параметра |
| Радиус колеса (метры) | R = 0.028 |
| Целевая координата по оси OX | X = 2 |
| Целевая координата по оси OY | Y = 2 |
| Время выполнения процесса (сек) | T = 20 |
| Минимальная погрешность ошибки рассогласования |  |
| Ширина колесной базы (метры) | B = 0.015 |

В качестве модели внешнего воздействия для дальномеров зададим следующие массивы данных в блоке повторяющихся исходных данных Repeating Sequence Stair из библиотеки элементов в ПО MATLAB Simulink. Массивы отображены в таблице 5.2. Каждый элемент массива является предположительным расстоянием до препятствия для каждой из указанных сторон [18, 20].

Таблица 5.2 – Массивы длин расстояний дальномеров.

|  |  |
| --- | --- |
| Расположение дальномера | Массивы значений расстояния |
| На левой стороне | [35 33 40 27 25 22 19 18] |
| В передней части | [35 33 40 41 42 33 22 26 39 40 45 50 47 42 36 31 24 16] |
| На правой стороне | [35 30 25 22 19 15 18 24 25 26 32].' |

Как мы можем видеть, длина массива каждой из сторон имеет разную длину, что в определенных случаях может дать более реальную картину и отработать алгоритмы движения в обратном направлении (тупик перед роботом), остановки, замедления скорости или ускорения для планомерного достижения желаемой траектории движения.

На рисунках 5.1-5.3 представлены графики движения тележки от начала координат () к целевой точке и ошибки рассогласования координат с учетом микро-профиля асфальтированного покрытия.

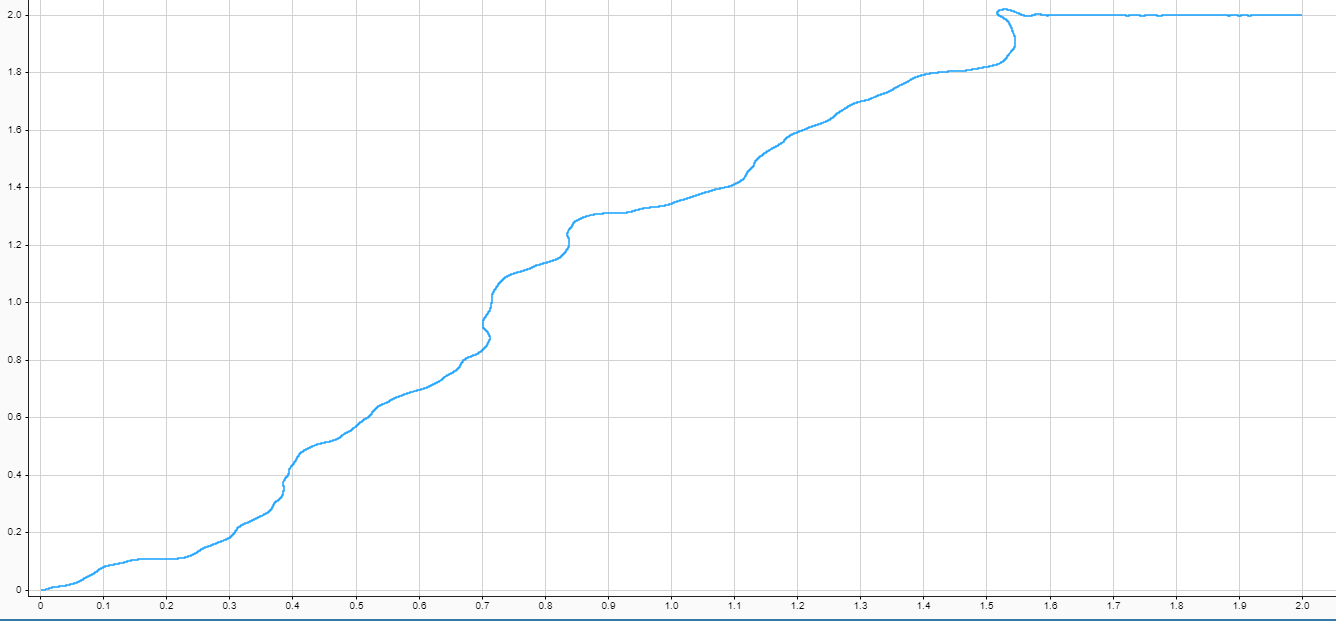


Рисунок 5.1 – Движение робота относительно координатной плоскости OXY.

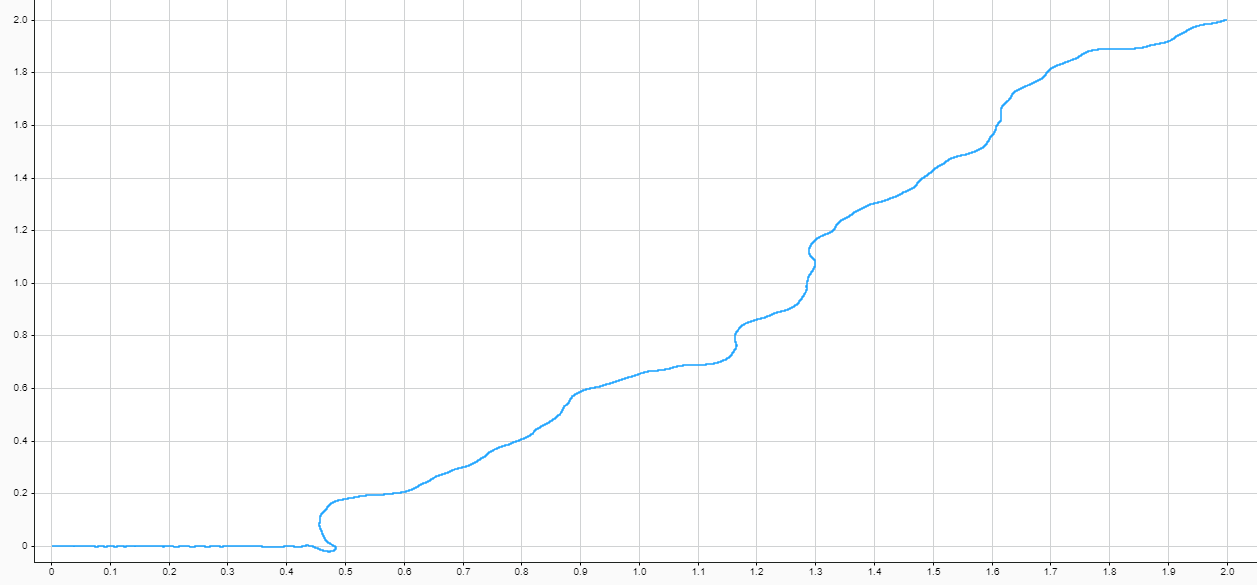


Рисунок 5.2 – Ошибка рассогласования навигации робота относительно координатной плоскости OXY.

Как мы можем видеть, первые два графика схожи и отличаются лишь проекциями. Таким образом, можем сказать, что ошибка рассогласования – величина обратная координатам движения робота.

График движения вместе с окном работы нечеткого регулятора отображены на рисунке 5.3. В окне отображены входные и выходные лингвистические переменные, а также работа 10 заданных правил ( в базе правил нечеткого регулятора.

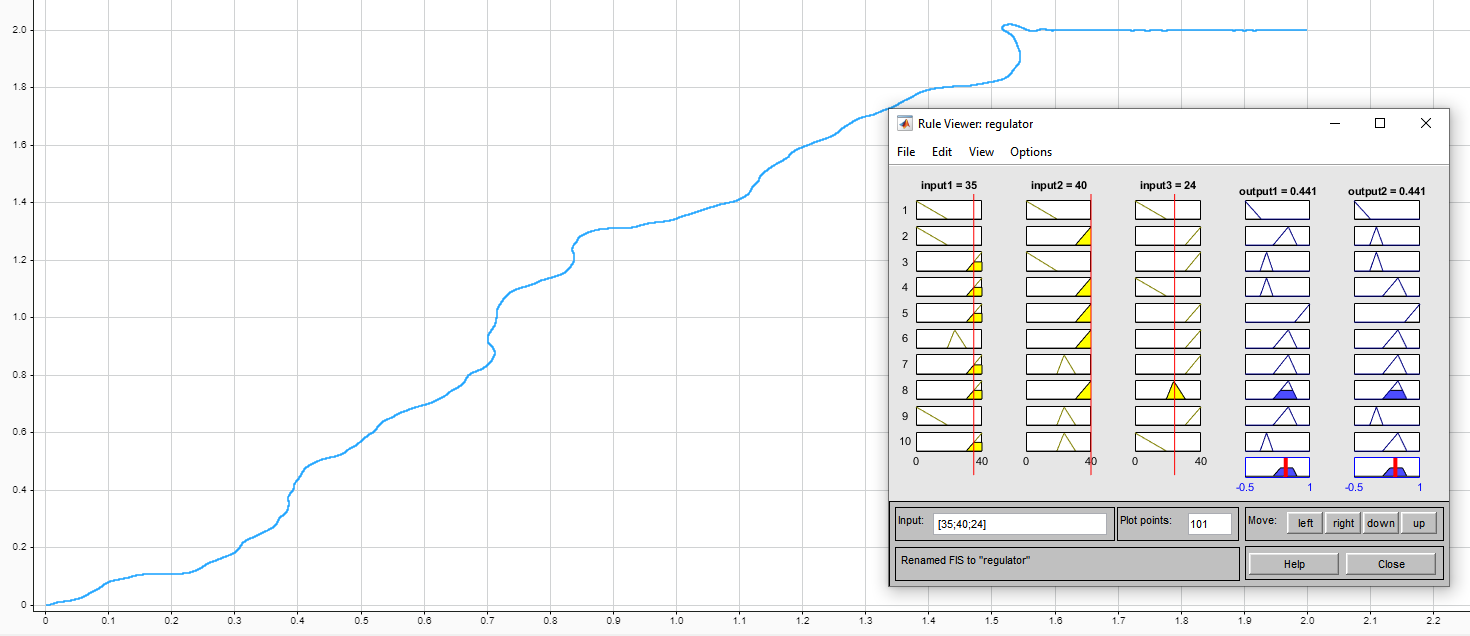


Рисунок 5.3 – Движение робота относительно координатной плоскости OXY с диалоговым окном нечеткого регулятора.

На рисунке 5.3 изображен график с регулятором. На нем для 3 входа указано значение 19, что соответствует малому расстоянию до препятствия справа.

Таким образом, робот должен совершить поворот налево с помощью остановки двигателя на левой стороне и предварительно снизить скорость. Значение коэффициента мощности левого двигателя , а мощность правого двигателя равна ½ от полной мощности для плавного вхождения в поворот. На пути робота возникало множество препятствий, однако тележка выезжала на желаемую траекторию после объезда преграды [20].

Теперь проведем моделирование системы с микро-профилем дороги с булыжным покрытием. Для этого введем коэффициенты грунтового вида дороги в файл конфигурации робота из таблицы 2.1. Полученные результаты движения мобильного робота показаны на рисунках 5.4-5.6.

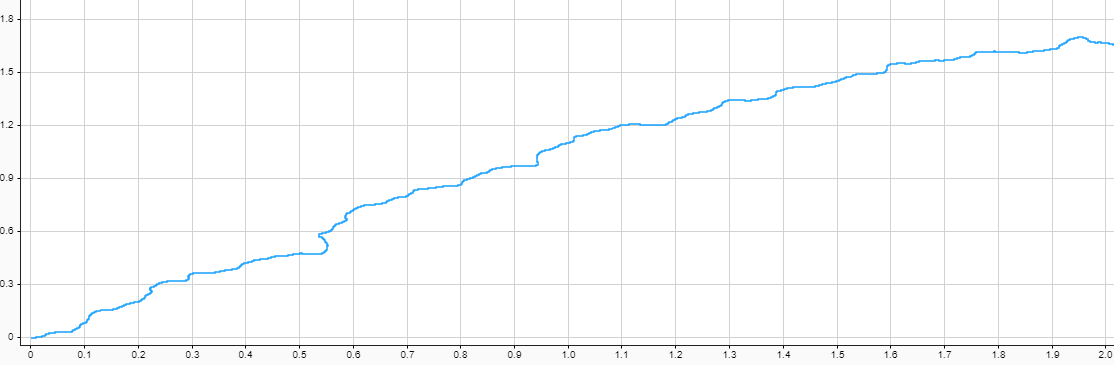


Рисунок 5.4 – Движение робота относительно координатной плоскости OXY.

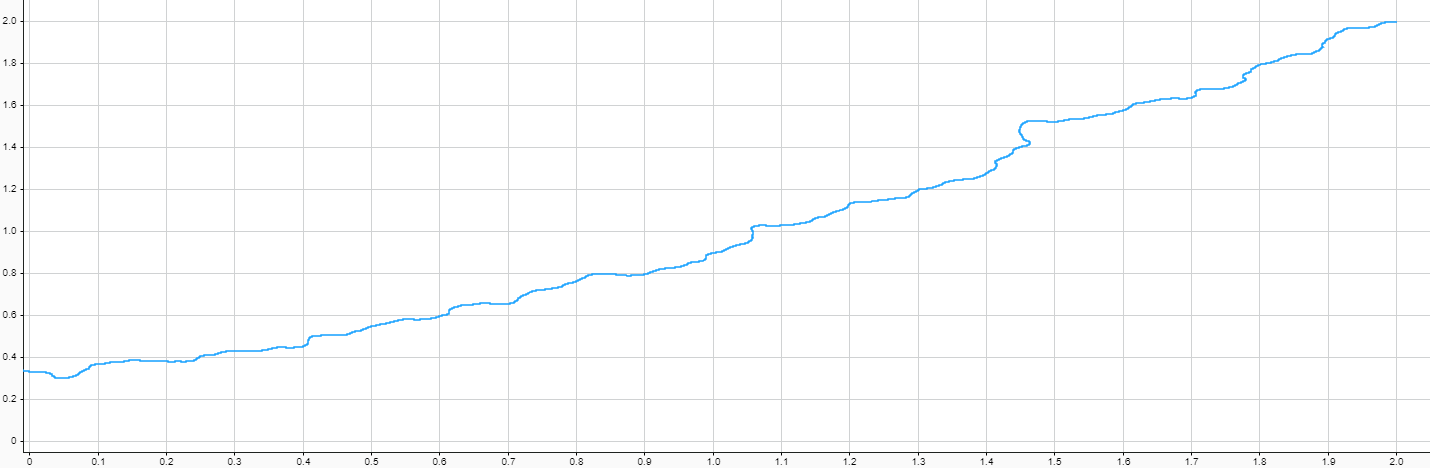


Рисунок 5.5 – Ошибка рассогласования навигации робота относительно координатной плоскости OXY.

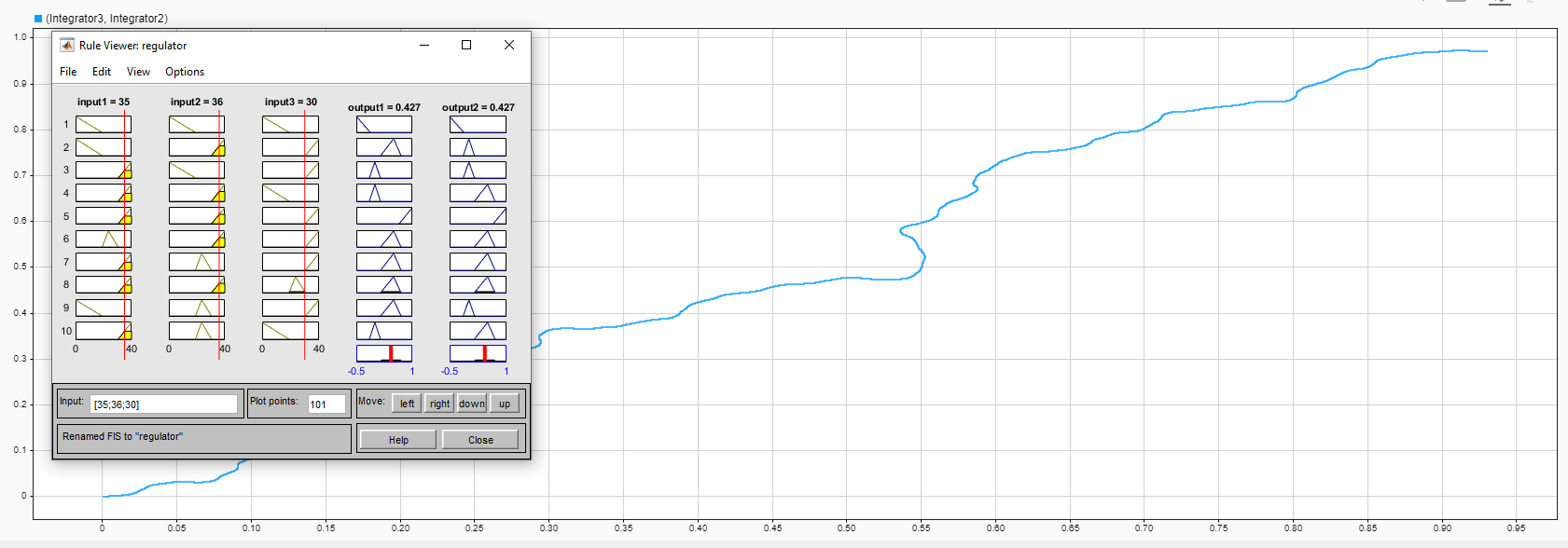


Рисунок 5.6 – Движение робота относительно координатной плоскости OXY с диалоговым окном нечеткого регулятора (булыжное покрытие).

На рисунке 5.6 изображен график с регулятором. На нем для каждого входа указано значение в диапазоне от 30 до 35, что соответствует возможному прохождению в узком пространстве с ограничением спереди.

Таким образом, робот должен снизить скорость на обоих двигателях, чтобы аккуратно проехать препятствие. Как и на асфальтированном покрытии, робот свободно объезжает возникающие на пути помехи. Однако траектория передвижения сместилась и движение усложнилось из-за проезда по булыжному покрытию [21].

Заметим, что в сравнении с асфальтированным покрытием робот на булыжном дорожном покрытии медленнее приходит к целевой точке из начала координат. Ему требуется больше усилий на преодоление возможных колебаний при движении к линии , однако он не смог ее достичь. Робот доехал к координатам .

Также исследуем влияние грунтового дорожного покрытия на управление и навигацию мобильного колесного робота. Для этого вновь введем параметры из таблицы 1 уже грунтового покрытия. Путь робота и ошибка рассогласования изображены на рисунках 5.7 – 5.9.

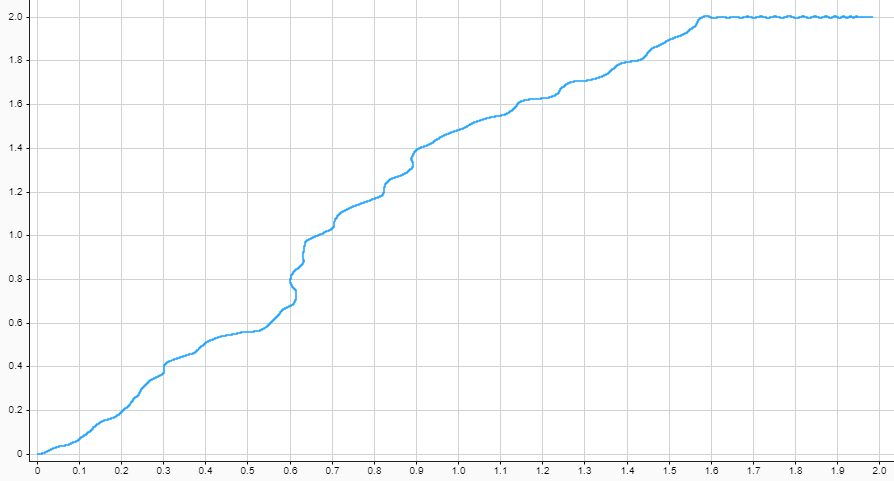


Рисунок 5.7 – Движение робота относительно координатной плоскости OXY (грунтовое покрытие).

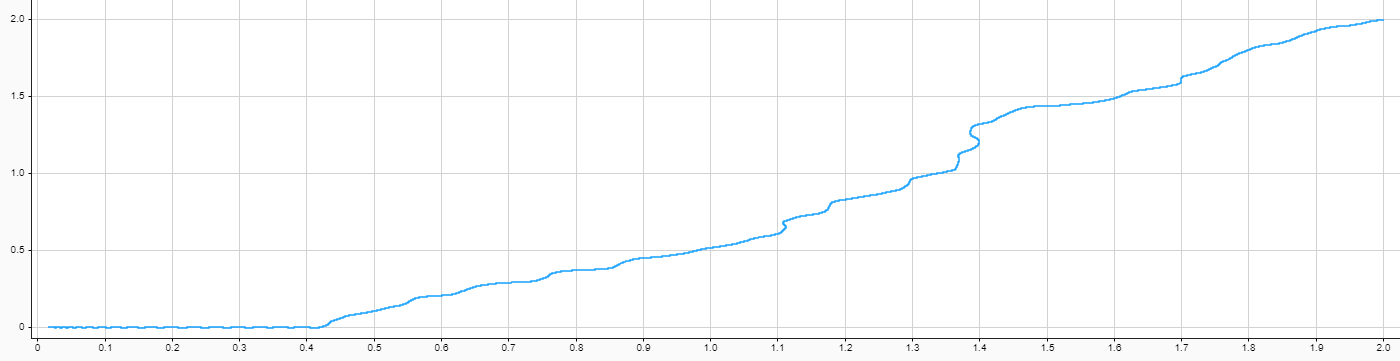


Рисунок 5.8 – Ошибка рассогласования навигации робота относительно координатной плоскости OXY (грунтовое покрытие).

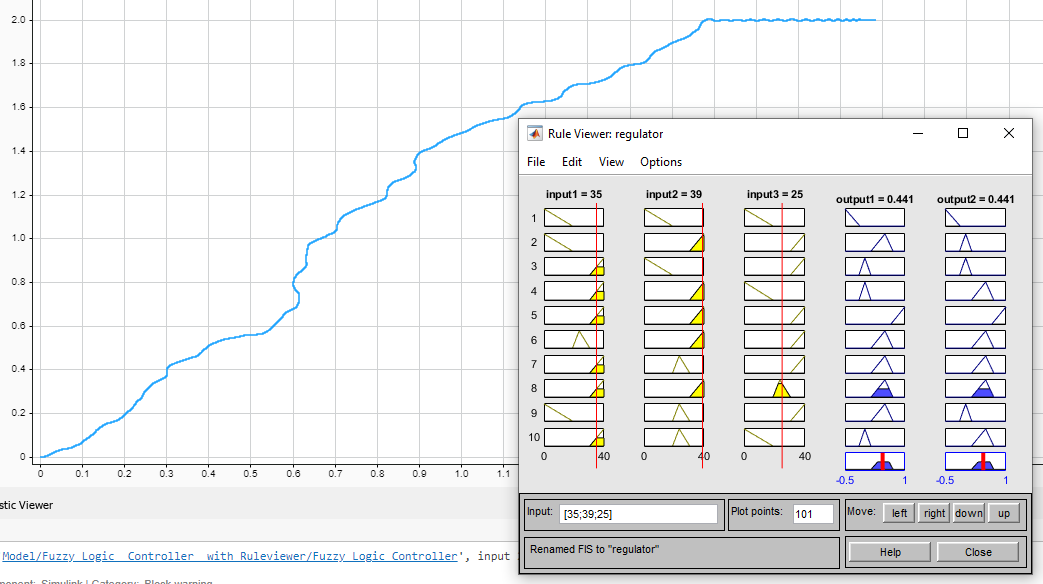


Рисунок 5.9 – Движение робота относительно координатной плоскости OXY с диалоговым окном нечеткого регулятора (грунтовое покрытие).

Сравнивая три вида дорожного покрытия, мы можем прийти к следующим выводам:

1. На асфальтовом покрытии движение робота к целевой точке происходит с наименьшими регулировкой и кратчайшей траекторией, что обеспечивает длительное прямолинейное достижение целевых координат и прямолинейный набор скорости для выхода на координату . Оставшийся путь до целевой точки наиболее короткий, т.к. робот по касательной проехал наибольшее расстояние.

2. На булыжном покрытии движение робота к целевой точке проходит с наибольшим временем регулирования и нестабильным формированием траектории движения. Это явление показывает, что покрытие значительно влияет на управляемость и координацию движения, поскольку лишь малый участок пройденного пути является прямолинейным, в отличие от асфальтового покрытия. Из-за серьезного влияния покрытия роботу сложнее по касательной выехать на желаемую траекторию, что влияет на длину остаточного пути после выхода на целевую координату по оси ОY (). Роботу не удалось достичь желаемой координаты по оси OY ().

3. На грунтовом покрытии робот движется по аналогии с асфальтированной дорогой с толь лишь разницей, что влияние чуть нестабильного покрытия незначительно повлияло на формирование траектории движения робота по касательной [22]. Таким образом время выхода на желаемую координату по оси OY () меньше лишь на 0.5 см.

# 6. Разработка и тестирование программного обеспечения управления и диспетчеризации колесного робота

## 6.1 Функциональные требования к программному обеспечению

При формировании технического задания по разработке программно-аппаратного комплекса большое значение уделяется функциональной части. Она составляет фундамент, на котором формируются основные требования к функциональным возможностям разрабатываемого прибора: методы управления исполнительными устройствами, приема сигналов с измерительных датчиков и передачи данных и сигналов с помощью исполнительных устройств, внутренние процессы, протекающие во время работы прибора и т.д [22, 29].

Разрабатываемый проект состоит из 3 основных частей:

1. Модуль управления электродвигателями

Данный модуль состоит из платы Arduino Uno, к которой подключены электродвигатели. Управление производится с помощью нечеткого регулятора, разработанного в главе 4.

1. Модуль приема-передачи данных на роботе

Блок управления приемом-передачей данных по беспроводной связи Wi-FI на роботе осуществляется с помощью микросхемы ESP8266. Она подключается к плате Arduino Uno через контакты RX-TX для обмена данными между двумя микроконтроллерами. На плате ESP-01 (базовая версия микроконтроллера ESP8266) располагается модуль беспроводной передачи Wi-Fi. С помощью него на сервер отправляется информация, собранная при обмене данными между двумя микроконтроллерами.

1. Модуль приема-передачи данных на сервере.

Сервер в данной системе является управляющим звеном. Он имеет графический интерфейс, СУБД и функциональные компоненты для взаимодействия с роботами [26]. С помощью сервера производится управление парком роботов: подключение к роботу, проверка сигналов с датчиков приближения, перевод в ручной режим управления, разработка маршрутов и формирования траектории движения. Также доступна возможность добавлять новые роботы и удалять из системы вышедшее из строя оборудование.

При формировании функциональных требования в рамках ВКР следует выделить следующие категории:

1. Управление электродвигателями:

– движение в прямом направлении;

– движение с поворотом налево;

– движение с поворотом направо;

– движение в обратном направлении;

1. Управление приемом-передачей информации на роботе:

– прием сигналов с ультразвуковых дальномеров;

– прием закодированных управляющих инструкций с сервера;

– декодирование управляющих инструкций и передача полученных данных на блок управления двигателями;

– передача сигналов с датчиков приближения на сервер;

– кодирование результатов работы с блока управления двигателями;

– передача результатов работы двигателей на сервер;

1. Управление приемом-передачей информации на сервере:

– создание записей о роботе;

– показ списка роботов, доступных к подключению;

– подключение к роботу для взаимодействия;

– прием данных с датчиков робота;

– формирование управляющих инструкций и маршрутов для робота;

Кодирование сигналов производится в модуле приема-передачи информации и на сервере. Используемым типом кодировки является стандарт кодирования ASCII символов Base64. Данная система содержит латинские символы A-Z, a-z и цифры от 0-9 и является наиболее распространенной в мире [24].

Кодирование сигналов позволяет повысить качество передачи информации между роботом и сервером за счет точного указания источника сигнала, например, от робота к серверу и в обратную сторону. Для этого необходимо определить основные характеристики кодируемого сигнала

1) Идентификатор робота;

2) Тип сигнала (датчик, направление движения, обозначение батареи);

3) Текущее значение сигнала (расстояние до препятствия, текущий заряд батареи, заданное движение робота).

4) Подтверждение выполнения операции (только для управляющих сигналов с сервера)

Идентификатор робота задается при загрузке программного обеспечения вместе с IP-адресом устройства для внесения в общую базу данных всех роботов.

В таблице 6.1 представлены все возможные обозначения типов сигналов, которые используются при передаче данных по MQTT протоколу между роботом и устройством.

Таблица 6.1 – Обозначения типов сигналов

|  |  |  |
| --- | --- | --- |
| Обозначение сигнала | Краткое описание | Источник сигнала |
| BATT | Уровень заряда батареи | Робот |
| WIFI | Режим автоматического управления | Робот |
| BLTH | Режим ручного управления | Сервер |
| SENL | Датчик препятствия на левом борту | Робот |
| SENR | Датчик препятствия на правом борту | Робот |
| SENP | Датчик препятствия на переднем борту | Робот |
| MOVP | Движение прямо | Робот |
| MOVL | Движение влево | Робот |
| MOVR | Движение вправо | Робот |
| MOVB | Движение в обратном направлении | Робот |
| MOVPS | Движение прямо | Сервер |
| MOVLS | Движение влево | Сервер |
| MOVRS | Движение вправо | Сервер |
| MOVBS | Движение в обратном направлении | Сервер |
| OK | Ответ робота на управляющий сигнал (только при задании движения с платформы) | Робот (конец сигнала) |

С помощью различных типов сигналов формируют комбинации, передаваемые по сети через протокол MQTT и программу-сервер MQTT Broker [31]. Возможные комбинации представлены в таблице 6.2. В качестве идентификатора робота используется кодовое слово ROBOT1.

Таблица 6.2 – Комбинации сигналов

|  |  |  |
| --- | --- | --- |
| Комбинация сигнала | Расшифровка сигнала | Источник сигнала |
| ROBOT1BATT95 | Текущий заряд батареи равен 95% | Робот с идентификатором ROBOT1 |
| ROBOT1WIFI1 | Включен режим автоматического управления по Wi-Fi | Робот с идентификатором ROBOT1 |
| ROBOT1BLTH1 | Включен режим ручного управления по Bluetooth | Сервер |
| ROBOT1SENL35 | Расстояние до препятствия по левому борту равно 35 см | Робот с идентификатором ROBOT1 |
| ROBOT1SENR15 | Расстояние до препятствия по правому борту равно 15 см | Робот с идентификатором ROBOT1 |
| ROBOT1SENP29 | Расстояние до препятствия по переднему борту равно 29 см | Робот с идентификатором ROBOT1 |
| ROBOT1MOVP40 | Движение прямо на расстоянии 40 см | Робот с идентификатором ROBOT1 |
| ROBOT1MOVL5 | Движение влево на расстоянии 5 см | Робот с идентификатором ROBOT1 |
| ROBOT1MOVR8 | Движение вправо на расстоянии 8 см | Робот с идентификатором ROBOT1 |
| ROBOT1MOVB3 | Движение в обратном направлении на расстоянии 3 см | Робот с идентификатором ROBOT1 |
| ROBOT1MOVPS100 | Движение прямо на расстоянии 40 см | Сервер |
| ROBOT1MOVLS5 | Движение влево на расстоянии 5 см | Сервер |
| ROBOT1MOVRS10 | Движение вправо на расстоянии 10 см | Сервер |
| ROBOT1MOVBS13 | Движение в обратном направлении на расстоянии 13 см | Сервер |
| ROBOT1MOVPS22OK | Ответ робота на выполненный управляющий сигнал «движение прямо на расстоянии 22 см» | Робот с идентификатором ROBOT1 |
| ROBOT1MOVLS4OK | Ответ робота на выполненный управляющий сигнал «движение прямо на расстоянии 22 см» | Робот с идентификатором ROBOT1 |
| ROBOT1MOVRS | Ответ робота на выполненный управляющий сигнал «движение прямо на расстоянии 22 см» | Робот с идентификатором ROBOT1 |
| ROBOT1MOVBS | Ответ робота на выполненный управляющий сигнал «движение прямо на расстоянии 22 см» | Робот с идентификатором ROBOT1 |

Одним из наиболее важных факторов является частота получения или отправки сигнала. Он может отправляться каждые несколько секунд или по окончании какого-либо действия (например, завершение движения по указанной траектории). Вариативность частот отображена в таблице 6.3.

Таблица 6.3 – Частота сигналов

|  |  |
| --- | --- |
| Обозначение сигнала | Частота отправки |
| BATT | Каждые 30 секунд |
| WIFI | При переходе в режим автоматического управления |
| BLTH | При переходе в режим ручного управления |
| SENL | Каждые 2 секунды |
| SENR | Каждые 2 секунды |
| SENP | Каждые 2 секунды |
| MOVP | По окончании движения по заданной инструкции |
| MOVL | По окончании движения по заданной инструкции |
| MOVR | По окончании движения по заданной инструкции |
| MOVB | По окончании движения по заданной инструкции |
| MOVPS | При отправке команды с сервера |
| MOVLS | При отправке команды с сервера |
| MOVRS | При отправке команды с сервера |
| MOVBS | При отправке команды с сервера |
| OK | По окончании движения по заданной инструкции |

Примером закодированного в кодировку Base64 сигнала является комбинация индикации текущего заряда батареи. Сформированный сигнал имеет длину на 1/3 больше исходного.

Исходный сигнал, сформированный роботом:

|  |  |  |
| --- | --- | --- |
|  | ROBOT1BATT95 |  |

Закодированный сигнал, полученный в ходе передачи данных:

|  |  |  |
| --- | --- | --- |
|  | Uk9CT1QxQkFUVDk1 |  |

Для каждой функциональной части формируется перечень технологий разработки программного обеспечения. С помощью него проводится алгоритмизация функциональных требований, разработка программного обеспечения и отладка взаимодействия между устройствами, физическими элементами робота [28].

## 6.2 Перечень технологий программного обеспечения

Перечень (далее стек) технологий представляет собой совокупность всех применяемых в ходе разработки языков программирования, средств управления базами данных и сред разработки и написания, анализа программного кода [25, 27]. Это является основным требованием при разработке программного продукта. Стек технологий позволяет нам подобрать необходимые дополнения, модули, облегчающие и ускоряющие разработку программного продукта. Также он подбирается исходя из параметров и требований задачи.

Данный проект состоит из трех функциональных модулей: Web-приложение для управления роботами с помощью компьютера, СУБД для хранения данных о состоянии датчиков, пройденном расстоянии и общей информации о роботах, а также программного модуля управления блоком приема-передачи информации между роботом и Web-приложением.

Выбранный стек технологий представлен в таблице 6.4.

Таблица 6.4 – Выбранный стек технологий для разработки ПО

|  |  |
| --- | --- |
| Компоненты ПО | Наименование технологий |
| Web-приложение | Python 3.10, Django 4.0 |
| Управление двигателями робота | C++, Arduino Library |
| Управление передачей данных по Wi-Fi | C++, ESP8266 Wi-Fi Library |
| Протокол взаимодействия робота и сервера | Message Queuing Telemetry Transport (далее MQTT) |
| СУБД | SQLite3 |
| Среда разработки Web-приложения | JetBrains PyCharm Professional |
| Среда разработки ПО робота | Microsoft Visual Studio Code |
| Операционная система | Ubuntu 22.04, Docker (для MQTT протокола) |

Ниже представлено более подробное описание каждой из выбранных технологий.

1) Web-приложение представляет собой монолитное приложение. Оно базируется на библиотеке Django 4.0 языка программирования Python.

Python – многопарадигмальный интерпретируемый язык программирования с полностью открытым исходным кодом, позволяющим создавать приложения для любых задач, особенно в области математического моделирования, разработки Web-приложений и робототехники (операционная система ROS).

Django – библиотека языка программирования Python для разработки универсальных Web-приложений: монолитных, двухзвенных или трехзвенных. Для отображения интерфейса используется собственная разработка библиотеки, включающая в себя языки разметки HTML, CSS и язык программирования JavaScript для управления кнопками, графиками внутри Web-страниц [32].

2) Управление двигателями робота

С++ – многопарадигмальный компилируемый язык программирования, имеющий наиболее широкое распространение и применение: от небольших встраиваемых систем, роботов до системы управления крупным промышленным оборудованием, проектирования и моделирования сложных математических, физических расчетов, а также операционные системы для компьютеров и встраиваемых систем. Имеет большое количество дополнений, библиотек по работе с оборудованием.

Arduino Library – библиотека языка программирования C++ для управления микроконтроллерами Arduino. Она позволяет управлять исполнительными элементами микроконтроллера: таймерами, прерываниями, внешними периферийными устройствами, ШИМ и тактовым генератором. Она позволяет создать сбалансированную программу для микроконтроллера, оптимальную по использованию внутренних ресурсов: флеш-памяти, ПЗУ, таймеров и т.д [30, 31].

В рамках данного проекта производится управление двигателями через выходные внешние контакты, по которым подключены устройства, и поворот колес с помощью прерываний, срабатывающие при появлении препятствий. Микроконтроллер каждые 0,5 секунд обращается к входным внешним контактам и снимает показания с датчиков, расположенных на трех бортах тележки.

3) Управление передачей данных по Wi-Fi

Библиотека ESP8266 Wi-Fi Library позволяет управлять модулем беспроводной связи Wi-Fi микроконтроллера ESP8266 средствами языка C++. Она включает в себя работу с HTTP-соединениями, передаче данных по технологиям WebSocket, MQTT [31].

В рамках ВКР микроконтроллер ESP-01 (уменьшенная версия ESP8266) подключается к линии RX-TX микроконтроллера Arduino Uno для взаимодействия между сервером и роботом. Микроконтроллер подключается к заданной Wi-Fi-сети и MQTT-серверу для обмена данными. При получении информации от микроконтроллера Arduino Uno модуль ESP-01 шифрует данные и по протоколу MQTT отправляет сообщение на сервер.

4) Протокол MQTT для передачи данных между роботом и сервером

Message Queuing Telemetry Transport (далее MQTT) – легковесный протокол передачи сообщений между IoT устройствами. Преимущества использования этого решения заключается в том, что робототехнические устройства имеют ограниченные ресурсы и пропускную способность передачи данных. Протокол MQTT использует минимальные ресурсы оборудования и требования к передаче данных (минимально 2 байта) между устройствами. Среди робототехнических устройств данная технология является стандартом де-факто по передаче данных в режиме реального времени по проводным и беспроводным соединениям Ethernet, Wi-Fi.

В рамках ВКР по протоколу MQTT передаются данные между устройствами и сервером. В качестве MQTT-сервера используется программа Eclipse Mosquitto, которое выполнено в виде Docker-контейнера.

5) СУБД SQLite3

Данная СУБД является компактной встраиваемой технологией, размещающейся на Web-сервере совместно с веб-приложением. Также данная библиотека позволяет упрощать программу, уменьшать время отклика и доступа между компонентами Web-приложения. Важной особенностью СУБД SQLite является компактность, простота развертывания и полная интеграция с языком программирования Python [26, 29].

В рамках ВКР в СУБД SQLite3 хранятся данные о роботах, пройденных маршрутах, показаний датчиков каждого из роботов в разный период времени использования. Также в базе данных хранится служебная информация библиотеки Django.

6) Доступные инструменты разработки:

– Инструменты для разработки на языке программирования Python – JetBrains PyCharm Professional, Microsoft Visual Studio 2022, Python IDLE, Microsoft Visual Code;

– Инструменты для разработки на языке программирования C++ – Arduino IDE, Microsoft Visual Studio Code, Programino Arduino IDE, PlatformIO;

– Инструменты для взаимодействия с СУБД SQLite3 – библиотека sqlite для Python, SQLite Administrator.

7) Операционные системы

В качестве операционной системы для развертывания и эксплуатации Web-приложения, MQTT-сервера и средств отладки и разработки на Arduino используется GNU/Linux Ubuntu 22.04.

## 6.3 Разработка программного обеспечения микроконтроллеров мобильного робота

Программное обеспечение микроконтроллеров, составляющие блок управления робота, состоит из двух модулей: микроконтроллера ATMEGA328P, отвечающего за управление двигателями постоянного тока, подключенных к модулю Motor Shield, Bluetooth-управлению, и микроконтроллера ESP8266 для управления передачей данных между сервером и роботом с помощью Wi-Fi соединения и MQTT-протоколу [31].

Программное обеспечение модуля ATMEGA328P разработано с помощью библиотеки Arduino Library и языка программирования C++. Библиотека включает в себя как оптимизированные для микроконтроллера основные функции языка C++, так и собственные функции, уникальные для работы микроконтроллера. Например, функция delay() для приостановки работы микроконтроллера, управление потоками данных через встроенный COM-порт с помощью класса Serial с методами begin() для инициализации потока данных, print() и println() для печати сообщения в COM-порт и read() для посимвольного считывания данных. Исходный код программного обеспечения представлен в Приложении Д.

В рамках выпускной квалификационной работы для микроконтроллера ATMEGA328P был разработан ряд собственных функций для выполнения поставленных задач:

1. byte getSensorData(const char\* type) – функция получения данных с датчика, указанного в качестве параметра. Параметр может принимать значения LEFT, RIGHT, PRIMARY, что соответствует расположению датчика. Функция обращается к контакту датчика с указанным типом и сохраняет полученное значение в локальную переменную sensor\_data. Далее функция возвращает значение переменной sensor\_data как результат выполнения функции.

2. void SendData(const char\* type, byte data) – функция отправки данных на микроконтроллер ESP8266 для дальнейшей рассылки данных по протоколу MQTT. Принимает два параметра: значение типа сигнала и текущий уровень сигнала. После получения данных функция генерирует команду и с помощью встроенной функции Serial.println() отсылает команду на микроконтроллер ESP8266.

3. void ManualDrive() – функция ручного управления. В перечень ее задач входят: переключение микроконтроллера ESP8266 для перехода в режим сна, активизации Bluetooth-соединения и передача управления на Bluetooth-контроллер HC-06. После подключения устройства, например, смартфона на ОС Google Android и запуска приложения для управления колесными роботами с помощью Bluetooth-соединения мы можем управлять роботом с помощью заданных команд.

4. char\* AutomaticDrive(char\* command) – функция автоматического управления. В ней реализован нечеткий алгоритм управления робота на базе получаемых данных с сенсоров для ориентации в пространстве. В качестве параметра принимает команду, поданную с Web-приложения. В ходе выполнения заданной команды функция отсылает данные о текущем местоположении с координатами X, Y. После окончания движения добавляет к исходной команде сигнал OK и отправляет ее на микроконтроллер ESP8266.

5. char\* getCommand() – функция получения управляющих команд с микроконтроллера ESP8266. Функция ожидает появления данных в общем потоке данных встроенного COM-порта. После получения данных она вызывает функцию AutomaticDrive(), которая задает автоматическое движение робота.

В функции setup() задаются параметры COM-порта для взаимодействия с микроконтроллером ESP8266, подключением по USB и Bluetooth-контроллера. Задаются начальные настройки переключателя ручного и автоматического режима (по умолчанию задан автоматический режим), датчиков приближения и двигателей постоянного тока.

В функции loop() производится циклическая проверка режима управления. Если режим автоматический, то устройство начинает рассылать данные датчиков и ожидает команды для выполнения. В ином случае производится переход в ручной режим управления.

Программное обеспечение модуля ESP8266 разработано с помощью библиотек ESP8266 Library, ESP8266WiFi, PubSubClient, base64 и языка программирования C++. Библиотеки включают в себя оптимизированные для микроконтроллера основные функции для взаимодействия с беспроводной сетью Wi-Fi, протоколом MQTT и кодировкой Base64 [25, 31]. Исходный код программного обеспечения представлен в Приложении Е.

В рамках выпускной квалификационной работы для микроконтроллера ESP8266WiFi был разработан ряд собственных функций для выполнения поставленных задач:

1. void connectToWiFi() – функция подключения микроконтроллера к точке доступа Wi-Fi. Данные точки доступа указываются в параметрах микроконтроллера. При попытке подключения устройство в течение 10 секунд опрашивает предполагаемый маршрутизатор. В случае положительного ответа микроконтроллер ESP8266 отсылает на Arduino Uno сигнал «Подключено» и полученный IP-адрес. В противном случае микроконтроллер через каждые 30 секунд проводит повторную попытку подключения.

2. void connectMQTT() – функция подключения к серверу MQTT. Микроконтроллер ESP8266 отпрашивает сервер по указанным данным: IP-адрес, порт. После успешного подключения микроконтроллер формирует идентификатор клиента и подключается к серверу.

3. void SendData(String command, const char\* type) – функция отправки данных по протоколу MQTT. Принимает два параметра: сгенерированную команду и значение типа сигнала. После получения данных функция кодирует информацию в кодировку Base64 и отправляет с помощью функции publish библиотеки PubSubClient на сервер MQTT.

4. String getData(char\* str) – функция получения данных с микроконтроллера Arduino Uno. После получения данных через COM-порт возвращает команду в виде строки.

5. String callback(char\* topic, byte\* payload, unsigned int length) – функция обратного вызова при получении данных из протокола MQTT.

## 6.4 Разработка Web-приложения управления и диспетчеризации

Платформа управления роботами представляет собой монолитное Web-приложение, которое взаимодействует с устройствами при помощи протокола MQTT. Исходный код Web-приложения представлен в Приложении Ж.

Главной страницей платформы является список всех роботов. Он состоит из навигационного меню по всей системе, бокового меню управления роботами. В него входят пункты списка всех устройств на складе и добавления туда нового оборудования. На рисунке 6.1 представлен интерфейс списка всех роботов на складе. Каждая единица списка выполнена в виде отдельной структуры, состоящей из элементов: пиктограмма, ID, IP-адрес устройства и идентификатор робота. В нижней части отображаются текущий заряд батареи робота и кнопка «Подробнее» для перехода к странице управления роботом [32].

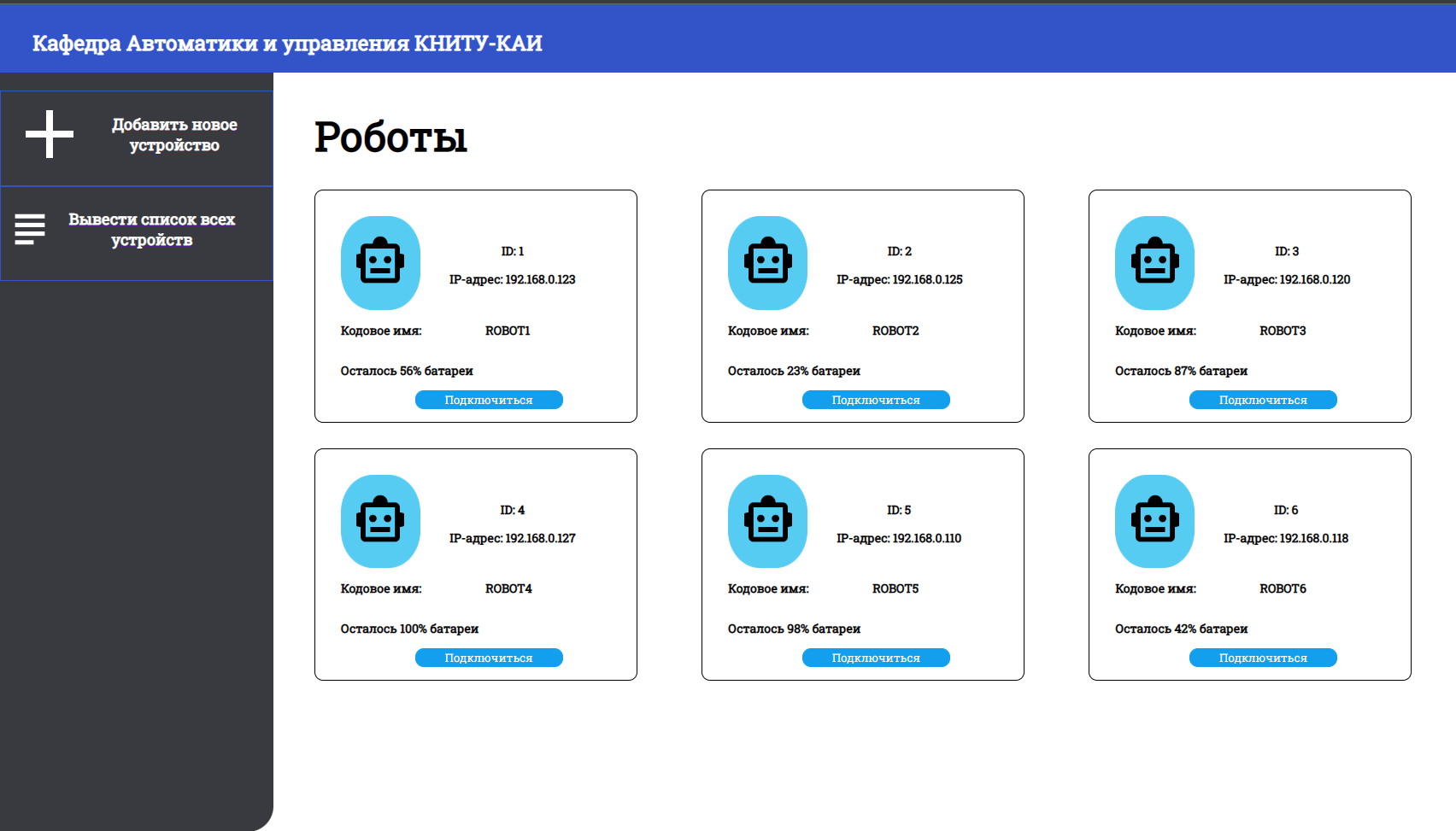


Рисунок 6.1 – Главная страница платформы

Добавление робота в систему проиллюстрировано на рисунке 6.2. Оно предлагает пользователю ввести основные данные: IP-адрес устройства и идентификатор робота. ID устройству присваивается автоматически при добавлении объекта в список всех роботов.

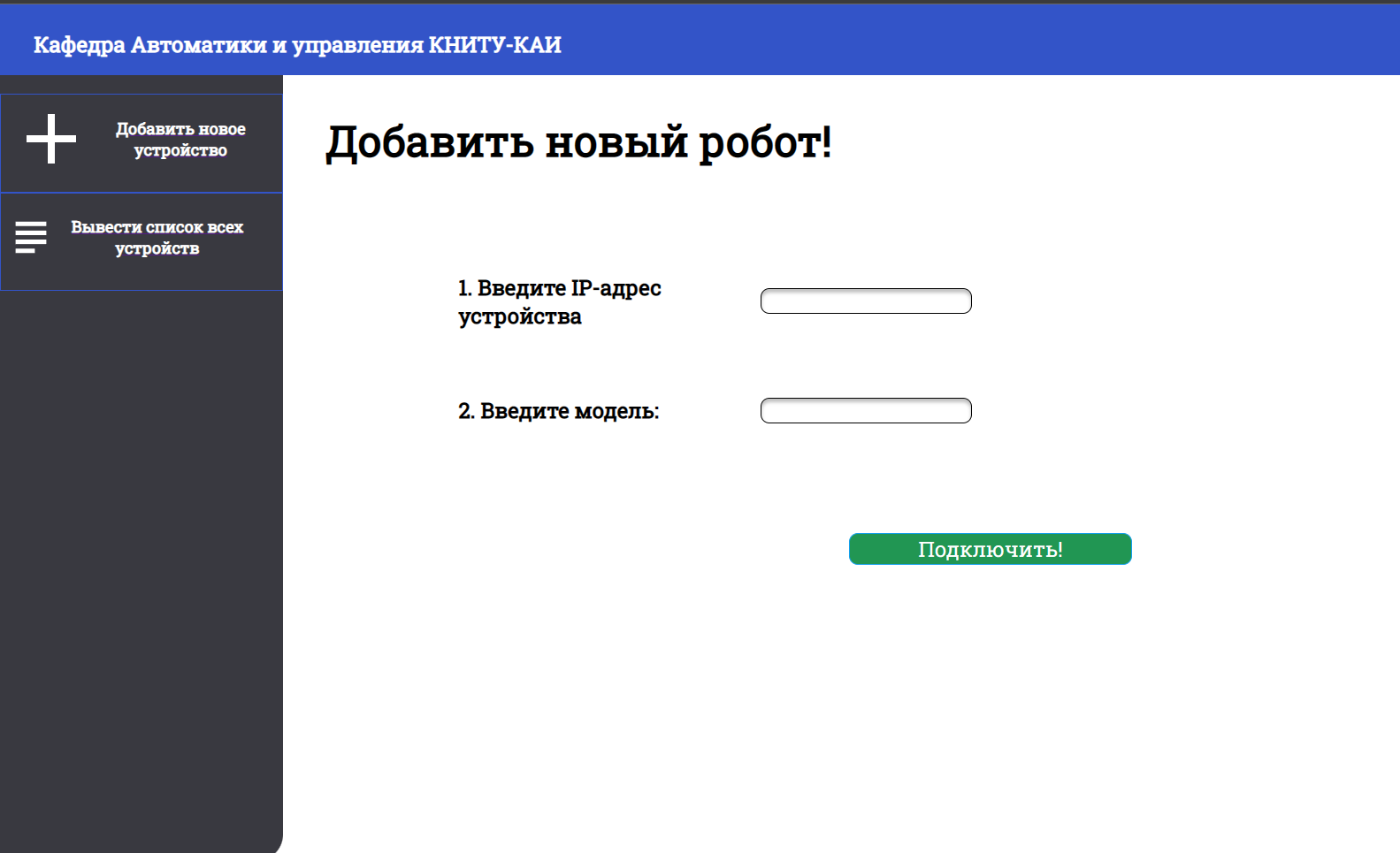


Рисунок 6.2 – Страница добавления робота

На рисунке 6.3 представлена страница выбранного робота, к которому пользователь подключается. Страница поделена на три блока: показания датчиков расстояния, основная информация о роботе и кнопки взаимодействия с роботом.

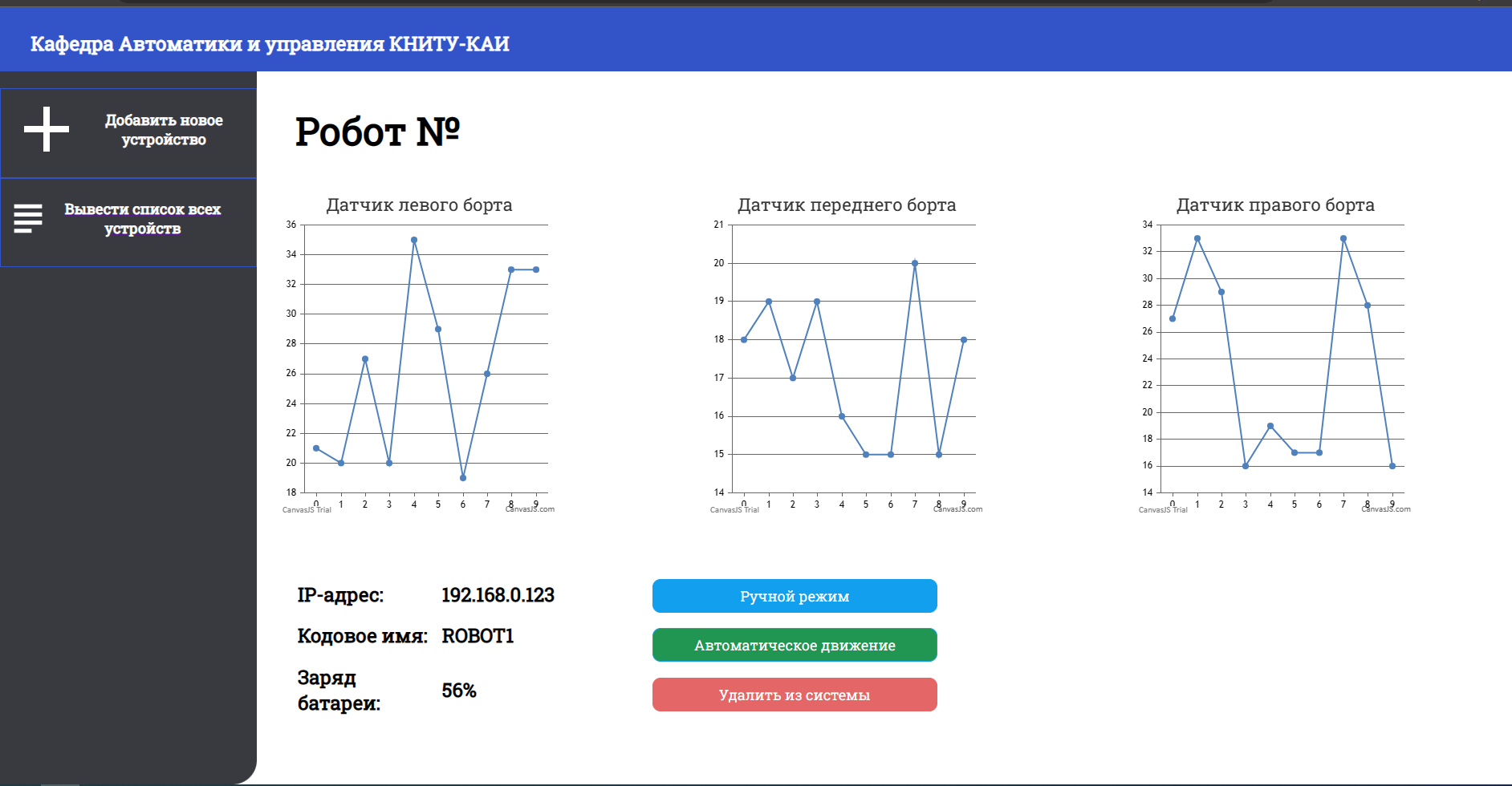


Рисунок 6.3 – Подробная информация о роботе

В блоке показаний датчиков расстояния пользователь видит графики по каждому из датчиков, установленных на роботе. Графики обновляются каждые 0.5 секунд по мере получения новых показаний с сенсоров робота.

Раздел с основной информацией о роботе позволяет пользователю ознакомиться с характеристиками робота: идентификатор, IP-адрес и уровень заряда. В правом блоке расположены 3 кнопки: ручной режим, автоматическое движение и удаление робота из системы. Рассмотрим каждую из них подробнее [24].

Ручной режим управления позволяет перевести робота в режим управления с помощью Bluetooth-контроллера, расположенного на шасси робота. Робот автоматически переводит Wi-Fi-контроллер в режим пониженного напряжения для снижения потребления энергии. Возврат к управлению с помощью Wi-Fi производится нажатием на тактовую кнопку на корпусе робота. При переключении в данный режим работы у пользователя появляется страница, представленная на рисунке 6.4.



Рисунок 6.4 – Страница ручного режима работы

Автоматическое движение – режим работы робота, в котором маршруты и движения робота задаются с помощью платформы. Перед пользователем появляется график и таблица с координатами X, Y, действиями и кнопками «Добавить» и «Удалить».

Кнопка «Добавить» увеличивает количество строк в таблице в зависимости от количества нажатий. Кнопка «Удалить» удаляет все созданные строки таблицы, кроме первой. Среди действий можно выбрать движение в прямом направлении, поворот налево, поворот направо и движение задним ходом.

Под таблицей расположена кнопка «Начать движение». Она сохраняет заданных маршрут в базу данных, после чего поочередно отправляет каждую управляющую инструкцию на приемник робота. Как только платформа получает положительный результат выполнения команды на роботе, она высылает следующую команду. На графике отображаются заданные пользователем и полученные с робота координаты. Их траектория может отличаться, поскольку робот объезжает препятствия в ходе прохождения заданного маршрута [7].

После нажатия на кнопку «Начать движение» перед пользователем появляется форма внесения данных о траектории движения и график, на котором представлен маршрут движения робота. Интерфейс данной формы представлен на рисунке 6.5.

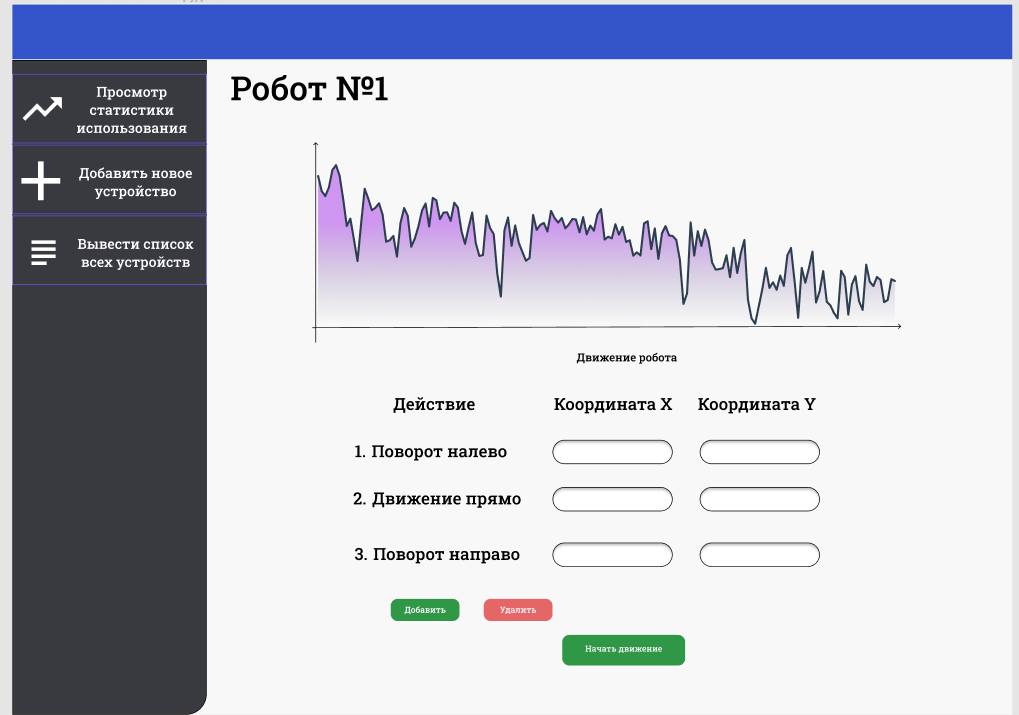


Рисунок 6.5 – Страница автоматического режима работы

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы, целью которой являлась разработка программно-аппаратного комплекса, позволяющего вести автоматическое и ручное управление роботами. Решены поставленные задачи и получены практические результаты.

В специальных разделах выпускной квалификационной работы был проведен литературный и патентный обзоры для детального анализа существующих технологий и решений, представлены математические модели двигателя постоянного тока, автоматической системы управления роботом и корреляционная функция формирующего фильтра для моделирования микро-профиля дорожного покрытия. Также представлены элементы электрических компонентов базы комплектующих, электрическая принципиальная и функциональные схемы блока управления роботом.

В ходе разработки интеллектуального алгоритма автоматического управления мобильным роботом были использованы элементы нечеткой логики: функции принадлежности, база правил нечеткого регулятора. Проведено моделирование системы с учетом различных профилей дорожного покрытия и применением нечеткого регулятора для обхода препятствий.

Была разработана системы управления мобильным роботом с использованием Web-технологий на базе Python, Django, для разработки программного обеспечения микроконтроллеров – C++, Arduino Library, MQTT Library. В качестве СУБД выбрана SQLite3. Представленная система включает в себя функции получения данных с датчиков робота, навигации робота с помощью координат, задаваемых в интерфейсе системы, добавления и удаления роботов.

# CONCLUSION

During the completion of the final qualification work, the purpose of which was to develop a software and hardware complex that allows automatic and manual control of robots. The tasks have been solved and practical results have been obtained.

In special sections of the final qualifying work, literary and patent reviews of other authors were conducted for a detailed analysis of existing technologies and solutions, mathematical models of a DC motor, an automatic robot control system and a correlation function for evaluating the micro-profile of the pavement were presented. The elements of the electrical components of the component base, the electrical schematic and functional diagrams of the robot control unit are also presented.

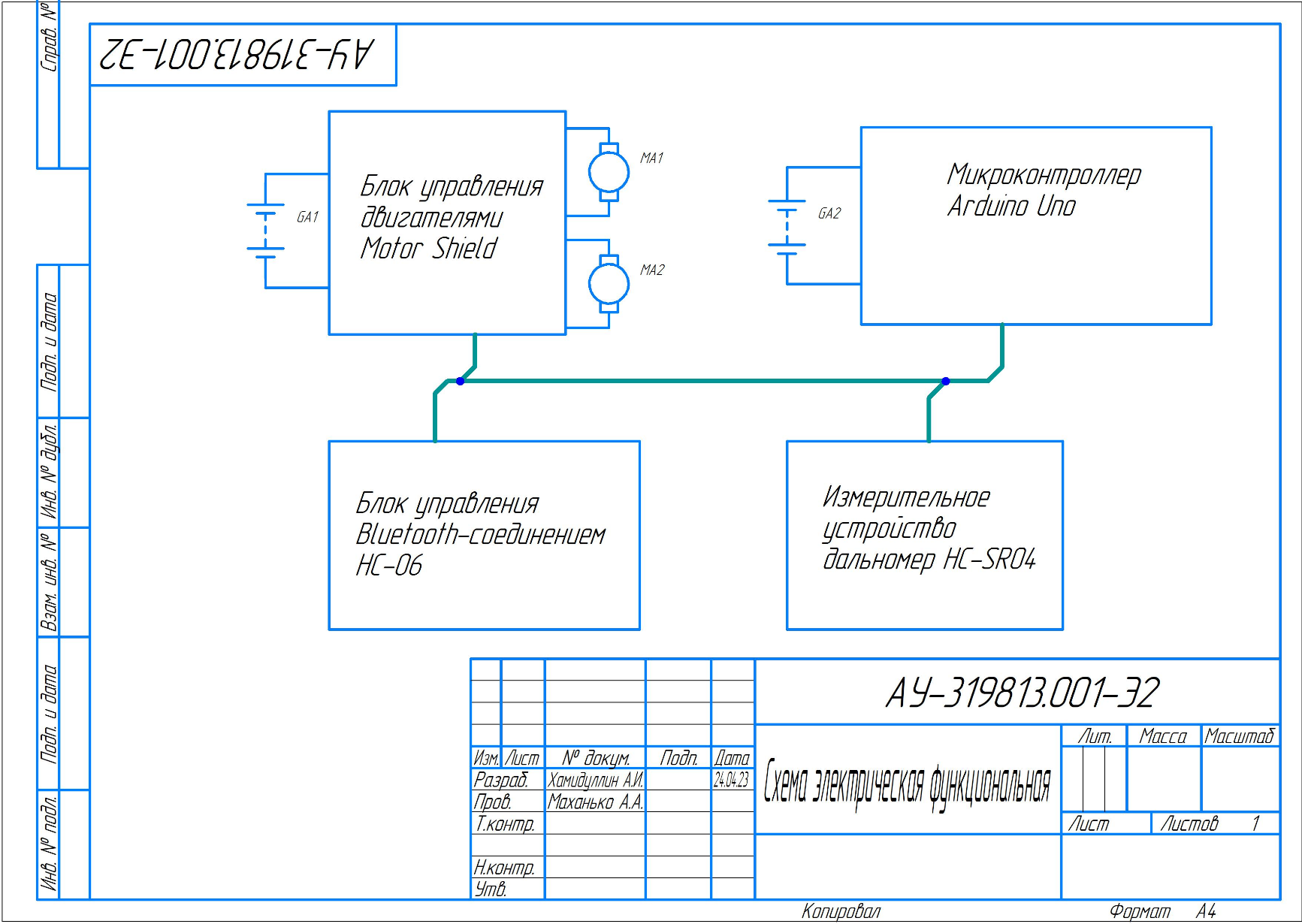
During the development of an intelligent algorithm for automatic control of a mobile robot, elements of fuzzy logic were used: membership functions, a base of fuzzy controller rules. The simulation of the system is carried out taking into account various profiles of the road surface and the use of a fuzzy controller.

JetBrains PyCharm Professional, Microsoft Visual Studio Code were chosen as control system development environments, C#, HTML, CSS as programming and markup languages for Web application development, C++, Arduino Library, MQTT Library for microcontroller software development. SQLite3 is selected as the DBMS. The developed control system of a mobile robot in the form of a Web application is presented. The software was tested together with the developed mobile robot.

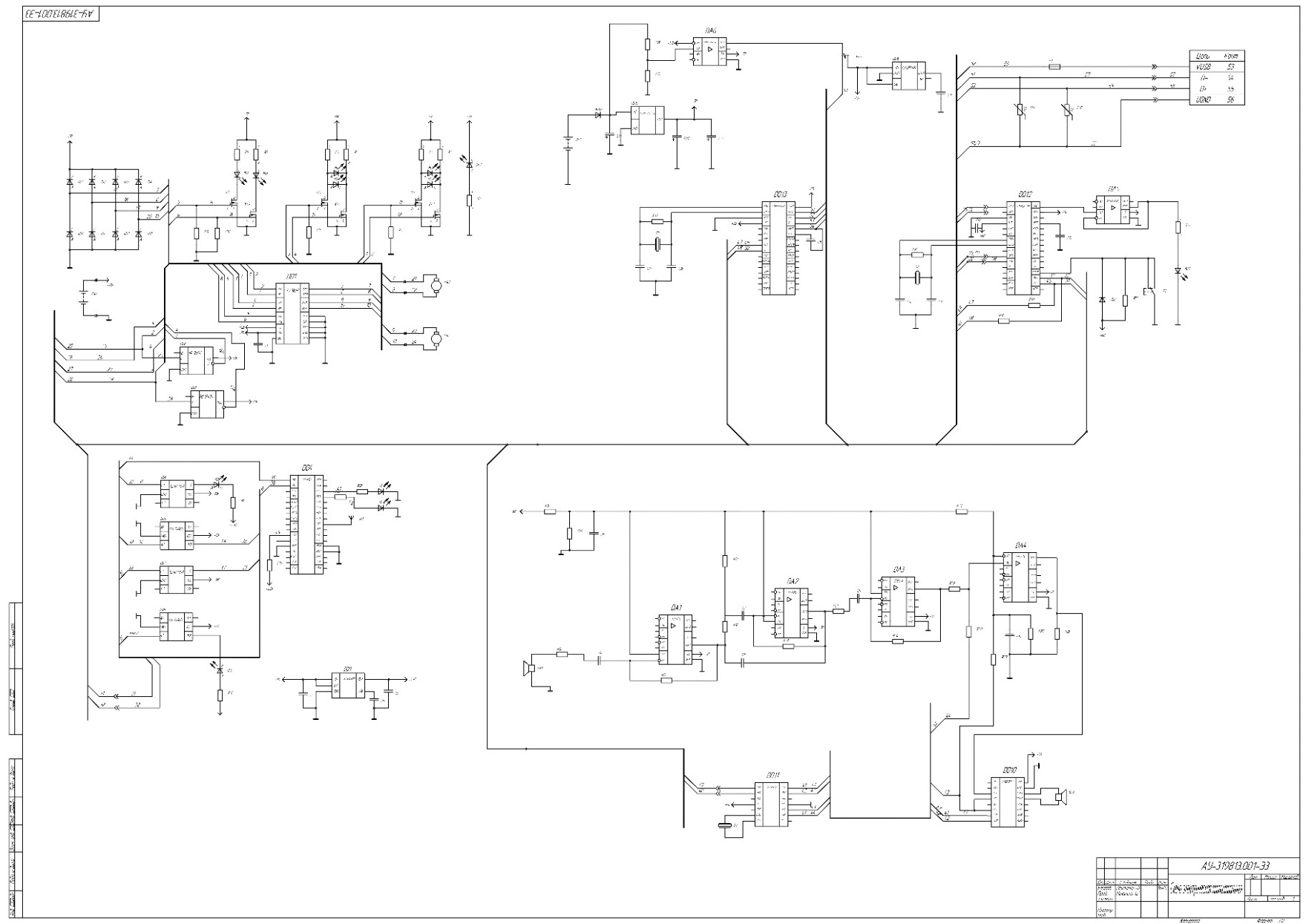
# СПИСОК ЛИТЕРАТУРЫ

1. Михайлов Б.Б., Назарова А.В., Ющенко А.С. Автономные мобильные роботы – навигация и управление // Известия ЮФУ. Технические науки. – 2016. – №2. – 48 с.
2. Мирошник И.В., Шалаев А.Н. Управление траекторным движением автономных роботов // Научно-технический вестник СПБ ГИТМО. Информационные, вычислительные и управляющие системы. – 2002. – №6. – 237 с.
3. Боровик А.И., Наумов Л.А. Компонентно-ориентированная программная платформа для автономных мобильных роботов // Известия ЮФУ. Технические науки. – 2013. – №23. – 39 с.
4. Wang J., Kumar A. Obstacle avoidance of mobile robots using modified artificial potential field algorithm // EURASIP Journal on Wireless Communications and Networking. – 2019. – V. 70. P. 1-19
5. Пат. 83858 Российская Федерация, МПК G 05 B 19/19. Система управления мобильным роботом / А. Г. Булгаков ; заявитель и патентообладатель ЮРГТУ. - № 2008151685/22 ; заявл. 25.12.2008 ; опубл. 20.06.09, Бюл. № 17.
6. Пат. EP3525992B1 Нидерланды, МПК B 25 J 9/16. Мобильный робот и роботизированная система на базе сервера и робота / С. Ю. Парк ; заявитель и правообладатель Samsung Electronics Ltd. - № 20160153535 ; заявл. 17.11.2016 ; опубл. 21.08.2019, Бюл. №3 4.
7. Мобильные роботы на базе Arduino. / М. Момот — 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2018. – 336 с. — (Электроника). – ISBN 978-5-9775-3861-9. – Текст: электронный // ЭБС iBooks [сайт]. – URL: https://ibooks.ru/products/353585 (дата обращения: 07.04.2023).
8. Математическая модель робота: [сайт]. [2013]. URL: https://habr.com/ru/post/178103/ (дата обращения: 12.04.2023).
9. Теория движения колесных машин: Учебник для студентов машиностроит. спец. вузов. – 2-е изд., доп. и перераб. – М.: Машиностроение, 1990. – 352 с.: ил.
10. Аполлонский, С.М. Теоретические основы электротехники. Электромагнитное поле: Учебное пособие / С.М. Аполлонский. – СПб.: Лань, 2018. – 592 c.
11. Курс лекций по дисциплине «Проектирование цифровых систем управления», Маханько А.А., 2022
12. Лоторейчук, Е.А. Теоретические основы электротехники.: Учебник / Е.А. Лоторейчук. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 320 c.
13. Миловзоров, О.В. Электроника: Учебник для бакалавров / О.В. Миловзоров, И.Г. Панков. - М.: Юрайт, 2017. - 407 c.
14. ГОСТ 2.710-81 «Единая система конструкторской документации. Обозначения буквенно-цифровые в электрических схемах»;
15. ГОСТ 2.721-74 «Единая система конструкторской документации. Обозначения условные графические в схемах. Обозначения общего применения»;
16. Шейнин А.М., Шейнин В.А. Алгоритмы и программы решения оптимальных задач надёжности машин. Учебное пособие. – М.: МАДИ, 1981. – 112 с.
17. Курс лекций по дисциплине «Интеллектуальные системы управления», Маликов А.И., 2022
18. Нечеткое, нейронное и гибридное управление: учеб. пособие / Ю.Н. Хижняков. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2013. – 303 с.
19. Интеллектуальные системы управления на основе методов нечеткой логики. / Усков А.А., Круглов В.В. – Смоленск: Смоленская городская типография, 2003. – 117 с.
20. Курс лекций по дисциплине «Современная прикладная теория управления», Гаркушенко В.И., 2022
21. Голубева Н. В. Математическое моделирование систем и процессов. СПб.: Лань, 2013. 192 с.
22. Тарасик В. П. Математическое моделирование технических систем. Минск: Новое знание, 2013. 584 с.
23. Иванова Г.С. Программирование [Электронный ресурс]: учебник. М.: Кнорус, 2019. 267 с. Доступ из ЭБС «Вейцман В.М. Проектирование информационных систем [Электронный ресурс]: учеб. пособие. СПб.: Лань, 2019. 316 с. Доступ из ЭБС «Лань».
24. Гниденко, И. Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017. – 235 с.
25. Мясникова, Н.А. Алгоритмы и структуры данных [Электронный ресурс]: учебное пособие / Мясникова Н.А. – М.: КноРус, 2021. – 185 с. <https://www.book.ru/book/936642>.
26. Карпова Т. С. Базы данных: модели, разработка, реализация [Электронный ресурс]: учеб. пособие. М.: Национальный Открытый Университет "ИНТУИТ", 2016. 403 с. Доступ из ЭБС «Лань».
27. ГОСТ 34.003-90 Библиографическая ссылка. Автоматизированные системы. Термины и определения. – М.: Стандартинформ, 1990. – 23 с.
28. Назаров, С.В. Эффективность и оптимизация компьютерных систем : монография / Назаров С.В. – М. : Русайнс, 2019. – 219 с. <https://www.book.ru/book/934529>
29. Ипатова Э. Р., Ипатов Ю. В. Методологии и технологии системного проектирования информационных систем [Электронный ресурс]: учебник. М.: Флинта, 2016. 256 с. Доступ из ЭБС «Book.ru».
30. Маран М. М. Программная инженерия [Электронный ресурс]: учеб. пособие. СПб.: Лань, 2018. 196 с. Доступ из ЭБС «Лань».
31. Шварц М. Интернет вещей с ESP8266. – СПБ.: БХВ-Петербург, 2018. – 192 с.
32. Меле А. Django 4 в примерах. – М.: ДМК Пресс, 2023. – 800 с.

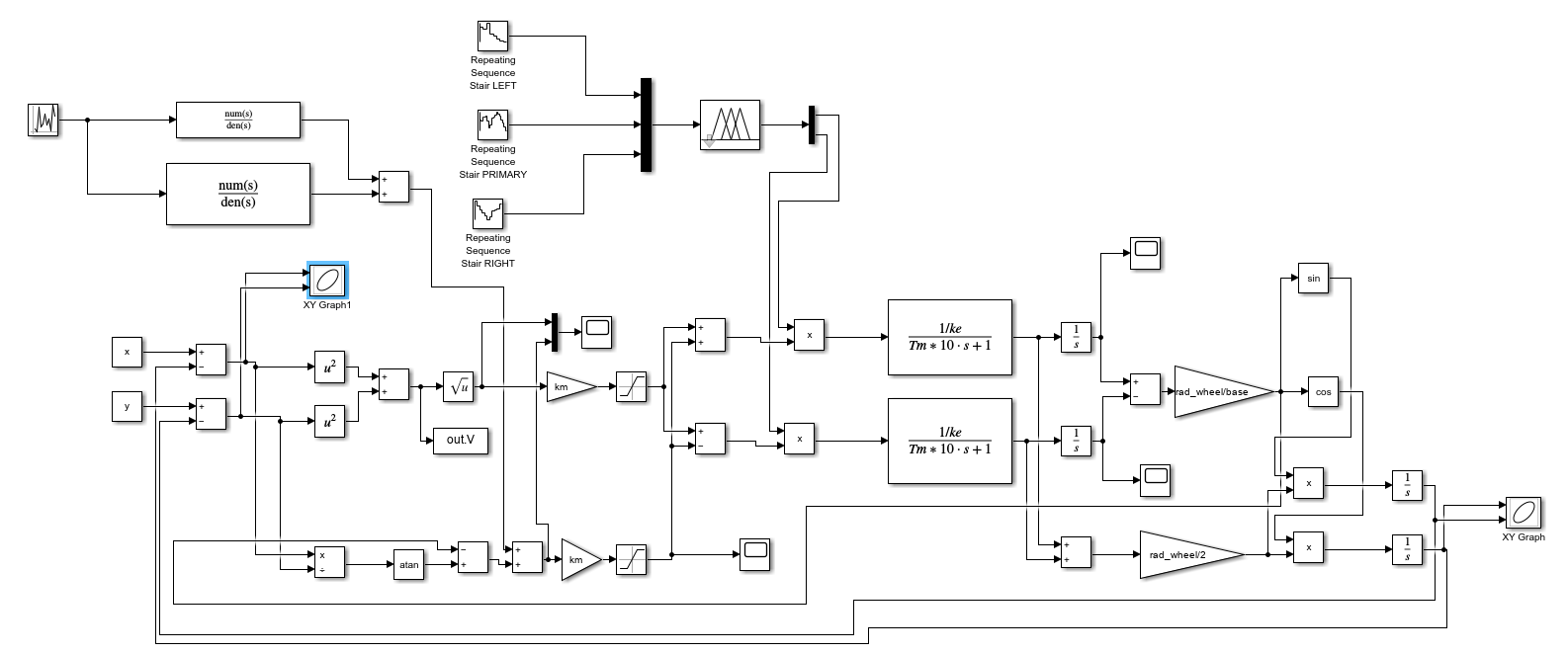
## Приложение А



## Приложение Б



## Приложение В



## Приложение Г

clear all;

clc;

rad\_wheel = 0.02;

limit = 20;

err = 0.001;

x = 2;

y = 2;

ts = 80;

base = 0.025;

a = rad\_wheel/base;

% Коэффициенты двигателя %

n = 166; % скорость в об/мин

U = 12; % Номинальное напряжение

R = 30; % сопротивление обмотки

I = 1.2; % ток

J = 0.0000068; % Момент инерции

L = 0.025; % Индуктивность

M0 = 0.84; % Крутящий момент покоя

m = 0.188; % вес

d = 0.005; % диаметр вала

w = (pi\*n)/30; % угловая скорость двигателя

Tm = (J\*w)/M0; % электромеханическая постоянная

i = 20; % коэффициент редукции

M = (m\*(d/2)^2)/2\*(w\*i)/Tm; % Крутящий момент двигателя

% M = 1.6;

km = i\*(M/I); % коэффициент передачи по току

ke = U/w; % коэффициент передачи противоЭДС

% Внешнее воздействие %

A1 = 1;

A2 = 0;

alpha1 = 0.14;

alpha2 = 0;

betta = 0;

out.V.signals.values = 0;

## Приложение Д

#include <SoftwareSerial.h>

#include <iarduino\_VCC.h>

#include <NewPing.h>

// создаём объект для работы с программным Serial

// и передаём ему пины TX и RX

SoftwareSerial mySerial(10, 11);

// Код ошибки: код 01; Радиус поворота выше скорости; Код 02; Скорость выше 255;

#define CON\_MOTOR1     0

#define CON\_MOTOR2     0

// Motor shield использует четыре контакта 4, 5, 6, 7 для управления моторами

// 4 и 7 — для направления, 5 и 6 — для скорости

#define SPEED\_1      5

#define DIR\_1        4

#define SPEED\_2      6

#define DIR\_2        7

// serial-порт к которому подключён Wi-Fi модуль

#define WIFI\_SERIAL mySerial

#define MAX\_VOLTAGE   5.0

String command\_l, command\_r, command\_p, command\_b;

int command; //Int to store app command state.

int Speed = 204; // 0 - 255.

int LED = 13;

int Speedsec;

int buttonState = 0;

int lastButtonState = 0;

int Turnradius = 0; //Set the radius of a turn, 0 - 255 Note:the robot will malfunction if this is higher than int Speed.

int brakeTime = 45;

int brkonoff = 1; //1 for the electronic braking system, 0 for normal.

byte PIN\_TRIG\_LEFT = 14;

byte PIN\_ECHO\_LEFT = 15;

byte PIN\_TRIG\_RIGHT = 16;

byte PIN\_ECHO\_RIGHT = 17;

byte PIN\_TRIG\_PRIMARY = 18;

byte PIN\_ECHO\_PRIMARY = 19;

byte MAX\_DISTANCE = 40;

NewPing sonar\_left(PIN\_TRIG\_LEFT, PIN\_ECHO\_LEFT, MAX\_DISTANCE);

NewPing sonar\_right(PIN\_TRIG\_RIGHT, PIN\_ECHO\_RIGHT, MAX\_DISTANCE);

NewPing sonar\_primary(PIN\_TRIG\_PRIMARY, PIN\_ECHO\_PRIMARY, MAX\_DISTANCE);

void setup() {

  // открываем последовательный порт для мониторинга действий в программе

  // и передаём скорость 9600 бод

  Serial.begin(9600);

  Serial.print("Serial init OK\r\n");

  // открываем Serial-соединение с Wi-Fi модулем на скорости 115200 бод

  WIFI\_SERIAL.begin(9600);

   for(int i = 4; i <= 7; i++)

        pinMode(i, OUTPUT);

    pinMode(LED, OUTPUT); //Set the LED pin.

  Serial.begin(9600);  //Set the baud rate to your Bluetooth module.

}

void loop() {

  unsigned int distanceSm\_left = sonar\_left.ping(); // Создание сигнала, получение параметра его продолжительности в мкс (uS).

  unsigned int distance\_left = distanceSm\_left / US\_ROUNDTRIP\_CM;

  delay(20);

  unsigned int distanceSm\_right = sonar\_right.ping(); // Создание сигнала, получение параметра его продолжительности в мкс (uS).

  unsigned int distance\_right = distanceSm\_right / US\_ROUNDTRIP\_CM;

  delay(20);

  unsigned int distanceSm\_primary = sonar\_primary.ping(); // Создание сигнала, получение параметра его продолжительности в мкс (uS).

  unsigned int distance\_primary = distanceSm\_primary / US\_ROUNDTRIP\_CM;

  delay(20);

  float  battery\_value = analogRead\_VCC();

  String command = "";

  if (!WIFI\_SERIAL) {

    Serial.print("NOT AVALIABLE");

  }

  // Serial.print(WIFI\_SERIAL.readString());

  if (WIFI\_SERIAL.available()) {

    String str = WIFI\_SERIAL.readString();

    str.trim();

    Serial.println(str.c\_str());

    if (str == String("SL")) {

      command\_l = "SENL" + String(distance\_left);

      Serial.println(command\_l);

      WIFI\_SERIAL.println(command\_l);

      Serial.println("LEFT sended");

      delay(100);

    }

    else if (str == String("SR")) {

      command\_r = "SENR" + String(distance\_right);

      Serial.println(command\_r);

      WIFI\_SERIAL.println(command\_r);

      Serial.println("RIGHT sended");

      delay(100);

    }

    else if (str == String("SP")) {

      command\_p = "SENP" + String(distance\_primary);

      Serial.println(command\_p);

      WIFI\_SERIAL.println(command\_p);

      Serial.println("PRIMARY sended");

      delay(100);

    }

    else if (str == String("SB")) {

      command\_b = "BATT" + String(round((battery\_value/5.0)\*100));

      Serial.println(command\_b);

      WIFI\_SERIAL.println(command\_b);

      Serial.println("BATTERY sended");

      delay(100);

    }

  }

  command\_l = "", command\_r = "", command\_p = "", command\_b = "";

}

void manualDrive() {

  if (Serial.available() > 0) {

    command = Serial.read();

    Stop(); //Initialize with motors stoped.

    switch (command) {

      case 'F':

        go(FORWARD, Speed);

        break;

      case 'B':

        go(BACKWARD, Speed);

        break;

      case 'L':

        go(LEFT, Speed);

        break;

      case 'R':

        go(RIGHT, Speed);

        break;

      case '0':

        Speed = 100;

        break;

      case '1':

        Speed = 140;

        break;

      case '2':

        Speed = 153;

        break;

      case '3':

        Speed = 165;

        break;

      case '4':

        Speed = 178;

        break;

      case '5':

        Speed = 191;

        break;

      case '6':

        Speed = 204;

        break;

      case '7':

        Speed = 216;

        break;

      case '8':

        Speed = 229;

        break;

      case '9':

        Speed = 242;

        break;

      case 'q':

        Speed = 255;

        break;

    }

    Speedsec = Turnradius;

    if (brkonoff == 1) {

      brakeOn();

    } else {

      brakeOff();

    }

  }

}

void Stop() {

  analogWrite(SPEED\_1, 0);

  analogWrite(SPEED\_2, 0);

  analogWrite(DIR\_1, 0);

  analogWrite(DIR\_2, 0);

}

void brakeOn() {

  //Here's the future use: an electronic braking system!

  // read the pushbutton input pin:

  buttonState = command;

  // compare the buttonState to its previous state

  if (buttonState != lastButtonState) {

    // if the state has changed, increment the counter

    if (lastButtonState == 'F') {

      if (buttonState == 'S') {

        go(BACKWARD, Speed);

        delay(brakeTime);

        Stop();

      }

    }

    if (lastButtonState == 'B') {

      if (buttonState == 'S') {

        go(FORWARD, Speed);

        delay(brakeTime);

        Stop();

      }

    }

    if (lastButtonState == 'L') {

      if (buttonState == 'S') {

        go(LEFT, Speed);

        delay(brakeTime);

        Stop();

      }

    }

    if (lastButtonState == 'R') {

      if (buttonState == 'S') {

        go(RIGHT, Speed);

        delay(brakeTime);

        Stop();

      }

    }

  }

  // save the current state as the last state,

  //for next time through the loop

  lastButtonState = buttonState;

}

void go(int newDirection, int speed)

{

  boolean motorDirection\_1, motorDirection\_2;

  switch (newDirection) {

    case FORWARD:

        motorDirection\_1 = false;

        motorDirection\_2 = true;

        break;

    case BACKWARD:

        motorDirection\_1 = true;

        motorDirection\_2 = false;

        break;

    case LEFT:

        motorDirection\_1 = false;

        motorDirection\_2 = false;

        break;

    case RIGHT:

        motorDirection\_1 = true;

        motorDirection\_2 = true;

        break;

  }

  // Если мы ошиблись с подключением - меняем направление на обратное

  motorDirection\_1 = CON\_MOTOR1 ^ motorDirection\_1;

  motorDirection\_2 = CON\_MOTOR2 ^ motorDirection\_2;

  analogWrite(SPEED\_1, speed);

  analogWrite(SPEED\_2, speed);

  digitalWrite(DIR\_1, motorDirection\_1);

  digitalWrite(DIR\_2, motorDirection\_2);

}

## Приложение Е

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include "base64\_utils.h"

IPAddress mqttServer(185, 152, 81, 104);

const char\* SSID = "malcon";

const char\* PWD = "1234fav1234";

long lastime = 0, lassensor\_l = 0, lassensor\_r = 0, lassensor\_p = 0, lassensor\_b = 0;

String command\_left = "", command\_right = "", command\_primary = "", command\_battery = "";

#define ROBOT\_NAME "ROBOT1"

void callback(char\* topic, byte\* payload, unsigned int length) {

  Serial.println("Callback: ");

  Serial.print((char\*)payload);

  Serial.println();

}

WiFiClient wifiClient = WiFiClient();

PubSubClient mqttClient(mqttServer, 1883, callback, wifiClient);

void connectToWiFi() {

  Serial.print("Connecting to ");

  Serial.println(SSID);

  WiFi.begin(SSID, PWD);

  while (WiFi.status() != WL\_CONNECTED) {

    Serial.print(".");

    delay(500);

    // we can even make the ESP32 to sleep

  }

  Serial.print("Connected - ");

  Serial.println(WiFi.localIP());

}

void reconnect() {

  Serial.println("Connecting to MQTT Broker...");

  while (!mqttClient.connected()) {

    Serial.println("Reconnecting to MQTT Broker..");

    String clientId = "ESP8266Client-";

    clientId += String(random(0xffff), HEX);

    if (mqttClient.connect(clientId.c\_str())) {

      Serial.println("Connected.");

      // subscribe to topic

    }

  }

}

void setup() {

  Serial.begin(9600);

  connectToWiFi();

}

void sendData(String command, char\* type) {

  long now = millis();

  command = String(ROBOT\_NAME) + command;

  int command\_len = command.length();

  int encoded\_len = b64\_enc\_len(command\_len);

  char\* encoded\_data = new char[encoded\_len];

  int b64len = b64\_encode(encoded\_data, (char\*)command.c\_str(), command\_len);

  Serial.println("Publishing data..");

  Serial.println(type);

  String encoded = String(encoded\_data);

  encoded.trim();

  // read temperature from sensor

  mqttClient.publish("esp8266/sensor/", encoded.c\_str());

  // read humidity from sensor

  mqttClient.publish("esp8266/test", String(now / 1110).c\_str());

  delay(40);

}

String getData(char\* str) {

  String command = "";

  while (Serial.available()) {

    command = Serial.readString();

    command.trim();

  }

  delay(40);

  if (command.substring(0, 4) == str) {

    return command;

  }

  return command;

}

void loop() {

  if (!mqttClient.connected())

    reconnect();

  mqttClient.loop();

  long now = millis();

  if (now - lassensor\_l > 4000) {

    Serial.println("SL");

    delay(50);

    command\_left = getData("SENL");

    lassensor\_l = now;

    if (command\_left.length() > 0) {

      sendData(command\_left, "left");

    }

  }

  if (now - lassensor\_r > 7000) {

    Serial.println("SR");

    delay(50);

    command\_right = getData("SENR");

    lassensor\_r = now;

    if (command\_right.length() > 0) {

      sendData(command\_right, "right");

    }

  }

  if (now - lassensor\_p > 10000) {

    Serial.println("SP");

    delay(50);

    command\_primary = getData("SENP");

    lassensor\_p = now;

    if (command\_primary.length() > 0) {

      sendData(command\_primary, "primary");

    }

  }

  if (now - lassensor\_b > 12000) {

    Serial.println("SB");

    delay(50);

    command\_battery = getData("BATT");

    lassensor\_b = now;

    if (command\_battery.length() > 0) {

      sendData(command\_battery, "battery");

    }

  }

  command\_primary = "";

  command\_left = "";

  command\_right = "";

  command\_battery = "";

  // Publishing data to MQTT

}

## Приложение Ж

Файл settings.py

from pathlib import Path  
import os  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
SECRET\_KEY = 'django-insecure-n!(u$u$e%7frjs4e562j@jd+1s\_)##-=^5k7utoebil6h2-6^x'  
DEBUG = True  
  
ALLOWED\_HOSTS = ['\*']  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'src.apps.core'  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware','django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'src.urls'  
  
TEMPLATES\_FOLDER = 'templates/'  
TEMPLATES\_ROOT = os.path.join(BASE\_DIR, TEMPLATES\_FOLDER)  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [TEMPLATES\_ROOT],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'src.wsgi.application'  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
LANGUAGE\_CODE = 'ru'  
  
TIME\_ZONE = 'Europe/Moscow'  
  
USE\_I18N = True  
  
USE\_TZ = True  
STATIC\_URL = '/static/'  
  
STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')  
  
STATICFILES\_DIRS = (  
 os.path.join(BASE\_DIR, 'staticfiles'),  
)  
  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')  
  
MEDIA\_URL = '/media/'  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'  
  
MQTT\_SERVER = '185.152.81.104'  
MQTT\_PORT = 1883  
MQTT\_KEEPALIVE = 60

Файл manage.py

import os  
import sys  
  
  
def main():os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'src.settings')  
 try:  
 from django.core.management import execute\_from\_command\_line  
 except ImportError as exc:  
 raise ImportError(  
 "Couldn't import Django. Are you sure it's installed and "  
 "available on your PYTHONPATH environment variable? Did you "  
 "forget to activate a virtual environment?"  
 ) from exc  
 execute\_from\_command\_line(sys.argv)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Файл urls.py

from django.contrib import admin  
from django.urls import path  
from src.apps.core import views  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('publish/', views.publish\_message, name='publish'),  
 path('', views.index, name='home'),  
 path('robot/<int:robot\_id>/', views.robot, name='robot'),  
 path('robot\_create/', views.addrobot, name='addrobot'),  
 path('robot\_delete/<int:robot\_id>/', views.deleterobot, name='deleterobot'),  
 path('robot\_bluetooth/<int:robot\_id>/', views.change\_type\_robot, name='robot\_bluetooth'),  
 path('sensor\_data/', views.sensor\_data, name='sensor\_data')  
]

Файл wsgi.py

import os  
from django.core.wsgi import get\_wsgi\_application  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'src.settings')  
application = get\_wsgi\_application()

Файл asgi.py

import os  
from django.core.asgi import get\_asgi\_application  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'src.settings')  
application = get\_asgi\_application()

Файл models/battery.py

from django.db import models  
from django.core.validators import MinValueValidator, MaxValueValidator  
class Battery(models.Model):  
 robot = models.ForeignKey(  
 to='core.Robot',  
 on\_delete=models.CASCADE,  
 related\_name='battery',  
 related\_query\_name='battery',  
 verbose\_name='Ссылка на робота',  
 help\_text='Ссылка на робота',  
 )  
 battery\_num = models.IntegerField(  
 verbose\_name="Остаток заряда аккумулятора",  
 help\_text="Остаток заряда аккумулятора",  
 null=True,  
 blank=True,  
 validators=[MinValueValidator(0), MaxValueValidator(100)],  
 )  
 def \_\_str\_\_(self):  
 return f"Робот {self.robot.token} - остаток заряда {self.battery\_num}%"  
 class Meta:  
 verbose\_name = "Емкость батареи робота"  
 verbose\_name\_plural = "Состояние батарей роботов"

Файл models/robot.py

from django.db import models  
from django.shortcuts import reverse  
class Robot(models.Model):  
 ipaddr = models.CharField(  
 verbose\_name="IP-адрес устройства",  
 help\_text="IP-адрес устройства",  
 max\_length=16  
 )  
 token = models.CharField(  
 verbose\_name="Кодовое наименование устройства",  
 help\_text="Кодовое наименование устройства",  
 max\_length=16,  
 null=True,  
 blank=True  
 )  
 is\_connected = models.BooleanField(  
 verbose\_name="Подключен ли робот",  
 help\_text="Подключен ли робот?",  
 null=True,  
 default=False  
 )  
 first\_connect = models.DateField(  
 verbose\_name="Первое подключение робота",  
 help\_text="Первое подключение робота",  
 auto\_created=True,  
 null=True,  
 blank=True  
 )  
 manual\_manage = models.BooleanField(  
 verbose\_name="Ручное управление роботом",  
 help\_text="Ручное управление роботом",  
 default=False  
 )  
 def get\_absolute\_url(self):  
 return reverse('robot', kwargs={'robot\_id': self.pk})  
 def \_\_str\_\_(self):  
 return f"Робот {self.token}"  
 class Meta:  
 verbose\_name = "Робот"  
 verbose\_name\_plural = "Роботы"

Файл models/distance.py

from django.db import models  
class Distance(models.Model):  
 robot = models.ForeignKey(  
 to='core.Robot',  
 on\_delete=models.CASCADE,  
 related\_name='distance',  
 related\_query\_name='distance',  
 verbose\_name='Ссылка на робота',  
 help\_text='Ссылка на робота',  
 )  
 path = models.IntegerField(  
 verbose\_name="Расстояние (в см)",  
 help\_text="Расстояние (в см)",  
 null=True,  
 blank=True  
 )  
 date\_distance = models.DateTimeField(  
 verbose\_name="Дата и время прохождения",  
 help\_text="Дата и время прохождения",  
 null=True,  
 blank=True,  
 auto\_created=True  
 )  
 speed = models.FloatField(  
 verbose\_name="Скорость робота",  
 help\_text="Скорость робота",  
 null=True,  
 blank=True  
 )  
 direction = models.CharField(  
 help\_text="Направление движения",  
 verbose\_name="Направление движения",  
 null=True,  
 blank=True,  
 max\_length=16  
 )  
 class Meta:  
 verbose\_name = 'Расстояние робота',  
 verbose\_name\_plural = "Список движений роботов"

Файл models/sensor.py

from django.db import models  
from src.apps.core.enums import SensorTypeEnum  
class Sensor(models.Model):  
 robot = models.ForeignKey(  
 to='core.Robot',  
 on\_delete=models.CASCADE,  
 related\_name='sensor',  
 related\_query\_name='sensor',  
 verbose\_name='Ссылка на робота',  
 help\_text='Ссылка на робота',  
 )  
 sensor\_type = models.CharField(  
 verbose\_name="Тип датчика",  
 help\_text="Тип датчика",  
 null=True,  
 blank=True,  
 max\_length=100,  
 choices=[{x.name, x.value} for x in SensorTypeEnum]  
 )  
 date\_sensor = models.DateTimeField(  
 verbose\_name="Дата и время снятия показания",  
 help\_text="Дата и время снятия показания",  
 null=True,  
 blank=True,  
 auto\_created=True  
 )  
 value = models.IntegerField(  
 verbose\_name="Значение с датчика (см)",  
 help\_text="Значение с датчика (см)",  
 null=True,  
 blank=True  
 )  
  
 def \_\_str\_\_(self):  
 return f"Робот {self.robot} - датчик {self.sensor\_type}"  
  
 class Meta:  
  
 verbose\_name = "Датчик робота"  
 verbose\_name\_plural = "Датчики робота"

Файл admin.py

from django.contrib import admin  
  
from .models import (Robot, Distance,  
 Battery, Sensor)  
  
admin.site.register(Robot)  
admin.site.register(Distance)  
admin.site.register(Battery)  
admin.site.register(Sensor)

Файл apps.py

from django.apps import AppConfig  
class CoreConfig(AppConfig):  
 default\_auto\_field = 'django.db.models.BigAutoField'  
 name = 'src.apps.core'

Файл forms.py

from django import forms  
class RobotForm(forms.Form):  
 ipaddr = forms.CharField(  
 label="IP-адрес устройства",  
 max\_length=16  
 )  
 token = forms.CharField(  
 label="Кодовое наименование устройства",  
 max\_length=16,  
 )

Файл views.py

import json, base64, datetime  
from django.http import JsonResponse, HttpResponse, HttpResponseRedirect  
from django.shortcuts import render  
from src.apps.mqtt.app\_config import client as mqtt\_client  
from src.apps.core.enums import SensorTypeEnum  
from src.apps.core.models import Robot, Sensor  
from .forms import RobotForm  
  
def get\_sensor\_data(sensors):  
 k = 0  
 sensor\_data = []  
 for i in range(len(sensors)-1, -1, -1):  
 data = {  
 "x": k,  
 "y": sensors[i].value  
 }  
 sensor\_data.append(data)  
 k+=1  
 return sensor\_data  
def publish\_message(request):  
 request\_data = json.loads(request.body)  
 msg = base64.b64encode(bytes(request\_data['msg'], 'utf-8'))  
 rc, mid = mqtt\_client.publish(request\_data['topic'], msg)  
 return JsonResponse({'code': rc})  
def index(request):  
 robots = Robot.objects.all()  
 robots = robots.values('token', 'id', 'battery\_\_battery\_num', 'is\_connected', 'ipaddr')  
 context = {  
 "robots": robots,  
 }  
 return render(request, 'core/index.html', context=context)  
def robot(request, robot\_id):  
 if not Robot.objects.filter(id=robot\_id).exists():  
 return HttpResponseRedirect('/')  
 robot = Robot.objects.get(id=robot\_id)  
 if robot.manual\_manage == True:  
 return HttpResponseRedirect(f'/robot\_bluetooth/{robot\_id}')  
 sensor\_left = Sensor.objects.filter(sensor\_type=SensorTypeEnum.LEFT.name).order\_by('date\_sensor', '-id')[:10]  
 sensor\_left\_data = get\_sensor\_data(sensor\_left)  
 sensor\_prim = Sensor.objects.filter(sensor\_type=SensorTypeEnum.PRIMARY.value).order\_by('date\_sensor', '-id')[:10]  
 sensor\_prim\_data = get\_sensor\_data(sensor\_prim)  
 sensor\_right = Sensor.objects.filter(sensor\_type=SensorTypeEnum.RIGHT.value).order\_by('date\_sensor', '-id')[:10]  
 sensor\_right\_data = get\_sensor\_data(sensor\_right)  
 context = {  
 'robot': {  
 'id': robot.id,  
 'token': robot.token,  
 'ipaddr': robot.ipaddr,  
 'battery\_\_battery\_num': robot.battery.first().battery\_num if robot.battery.all().exists() else "Не указано"  
 },  
 'sensor\_left': sensor\_left\_data,  
 'sensor\_prim': sensor\_prim\_data,  
 'sensor\_right': sensor\_right\_data  
 }  
 print(context)  
 return render(request, 'core/robot.html', context=context)  
def addrobot(request):  
 if request.method == 'POST':  
 form = RobotForm(request.POST)  
 if form.is\_valid():  
 Robot.objects.create(  
 ipaddr=form.cleaned\_data['ipaddr'],  
 token=form.cleaned\_data['token'],  
 is\_connected=False,  
 first\_connect=datetime.date.today()  
 )  
 return HttpResponseRedirect('/')  
 return render(request, 'core/create.html')  
def sensor\_data(request):  
 print(request.GET)  
 type = request.GET.get('sensor\_type')  
 sensor = Sensor.objects.filter(sensor\_type=type.upper()).order\_by('-id')[0:10]  
 sensordata = get\_sensor\_data(sensor)  
 return HttpResponse(json.dumps(sensordata))  
def deleterobot(request, robot\_id):  
 if Robot.objects.filter(id=robot\_id).exists():  
 Robot.objects.get(id=robot\_id).delete()  
 return HttpResponseRedirect('/')  
def change\_type\_robot(request, robot\_id):  
 robot = Robot.objects.get(id=robot\_id)  
 robot.manual\_manage = True  
 robot.save()  
 context = {  
 'robot': robot  
 }  
 return render(request, 'core/bluetooth.html', context=context)

Файл enums/sensor\_type\_enum.py

import enum  
  
  
class SensorTypeEnum(enum.Enum):  
  
 LEFT = "Левый борт"  
 PRIMARY = "Передняя часть"  
 RIGHT = "Правый борт"

Файл templates/base.html

{% load static %}  
<!DOCTYPE html>  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
 <link type="text/css" href="{% static 'core/css/main.css' %}" rel="stylesheet"/>  
 <link type="text/css" href="{% static 'core/css/index.css' %}" rel="stylesheet"/>  
 <link type="text/css" href="{% static 'core/css/robot\_create.css' %}" rel="stylesheet"/>  
 <link type="text/css" href="{% static 'core/css/robot\_charts.css' %}" rel="stylesheet">  
 <title>{{ title }}</title>  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <script src="{% static 'core/js/canvasjs.min.js' %}"></script>  
 <script src="{% static 'core/js/jquery-3.7.0.min.js' %}"  
 integrity="sha256-2Pmvv0kuTBOenSvLm6bvfBSSHrUJ+3A7x6P5Ebd07/g="  
 crossorigin="anonymous"></script>  
</head>  
<body>  
<div class="wrapper">  
 <div class="nav\_color">  
 <p class="nav\_bar\_\_title nav\_bar\_\_el">Кафедра Автоматики и управления КНИТУ-КАИ</p>  
 </div>  
 <main class="content\_block">  
 <aside class="nav\_bar">  
 {% block mainmenu %}  
 <div style="display: flex; flex-direction: column; align-content: space-evenly">  
 <div class="bar">  
 &nbsp;  
 </div>  
 <div>  
 <ul>  
 <li>  
 <div style="border: 1px solid #3354C8;">  
 <a href="/robot\_create/">  
 <div class="li\_\_block nav\_bar\_\_el ">  
 <img src="{% static 'core/images/add\_button.svg' %}" alt="plus">  
 Добавить новое устройство  
 </div>  
 </a>  
 </div>  
 </li>  
 <li>  
 <div style="border: 1px solid #3354C8;">  
 <a href="/">  
 <div class="li\_\_block nav\_bar\_\_el ">  
 <img src="{% static 'core/images/list.svg' %}" alt="plus">  
 Вывести список всех устройств  
  
 </div>  
 </a>  
  
 </div>  
 </li>  
 </ul>  
 </div>  
  
 </div>  
  
 {% endblock %}  
 </aside>  
 <div class="content\_body">  
  
 {% block content %}  
 {% endblock %}  
 </div>  
 </main>  
  
</div>  
  
</body>  
</html>

Файл templates/bluetooth.html

{% extends 'core/base.html' %}  
{% load static %}  
{% block content %}  
<div>  
 <div>  
 <h1 class="heading">Робот №{{ robot.pk }}</h1>  
 </div>  
 <div style="text-align: center">  
 <img style="height: 20%; width: 20%; " src="{% static 'core/images/Bluetooth.png' %}" alt="Bluetooth">  
 </div>  
 <div style="text-align: center">  
 <p style="font-size: 48px">Включен Bluetooth-режим</p>  
 </div>  
</div>  
{% endblock %}

Файл templates/create.html

{% extends 'core/base.html' %}  
{% load static %}  
{% block content %}  
 <div>  
 <h1 class="heading">Добавить новый робот!</h1>  
 </div>  
 <form action="/robot\_create/" method="post">  
 {% csrf\_token %}  
 <table class="table\_create">  
 <tbody>  
 <tr class="tr\_create">  
 <td class="tr\_create">  
 <label for="ipaddr" class="label\_font">1. Введите IP-адрес устройства</label>  
 </td>  
 <td>  
 <input id="ipaddr" type="text" name="ipaddr" maxlength="16" required class="input\_create">  
 </td>  
 </tr>  
 <tr class="tr\_create">  
 <td class="tr\_create">  
 <label for="token" class="label\_font">2. Введите модель:</label>  
 </td>  
 <td>  
 <input id="token" type="text" name="token" maxlength="16" required class="input\_create">  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 <br>  
 <br>  
 <br>  
 <br>  
 <div style="text-align: center">  
 <button type="submit" class="robot\_button button\_create">Подключить!</button>  
 </div>  
  
  
 </form>  
  
{% endblock %}

Файл templates/index.html

{% extends 'core/base.html' %}  
{% load static %}  
{% block content %}  
 <div>  
 <h1 class="heading">Роботы</h1>  
 </div>  
 <div>  
 <ul class="robot\_\_list">  
 {% for robot in robots %}  
 <li class="robot\_block">  
 <div>  
 <div class="robot\_maininf">  
 <div class="robot\_img">  
 <img src="{% static 'core/images/robot\_img.svg' %}" alt="robot">  
 </div>  
 <div class="robot\_info">  
 <p>ID: {{ robot.id }}</p>  
 <p style="white-space: nowrap">IP-адрес: {{ robot.ipaddr }}</p>  
 </div>  
 </div>  
 <div>  
 <div class="flex\_block">  
 <p style="margin-right: 75px">Кодовое имя:</p>  
 <p>{{ robot.token }}</p>  
 </div>  
 <div class="flex\_block" style="justify-content: space-between">  
 <p>Осталось {{ robot.battery\_\_battery\_num }}% батареи</p>  
 </div>  
 <div style="text-align: center">  
 <button class="robot\_button" onclick="location.href = '{% url 'robot' robot.id %}' ">Подключиться</button>  
 </div>  
 </div>  
 </div>  
  
 </li>  
 {% endfor %}  
 </ul>  
  
 </div>  
  
{% endblock %}

Файл templates/robot.html

{% extends 'core/base.html' %}  
  
{% load static %}  
  
{% block content %}  
 <script type="text/javascript">  
 const url = '/sensor\_data/'  
 window.onload = function () {  
 const chart\_left = new CanvasJS.Chart("chart\_left", {  
 theme: "light1",  
 animationEnabled: true,  
 title: {  
 text: "Датчик левого борта",  
 fontFamily: "Roboto Slab",  
 fontSize: 22,  
 },  
 data: [{  
 type: "line",  
 dataPoints: {{ sensor\_left | safe }}  
 }]  
 })  
 chart\_left.render();  
  
 const chart\_prim = new CanvasJS.Chart("chart\_prim", {  
 theme: "light1",  
 animationEnabled: true,  
 title: {  
 text: "Датчик переднего борта",  
 fontFamily: "Roboto Slab",  
 fontSize: 22,  
 },  
 data: [{  
 type: "line",  
 dataPoints: {{ sensor\_prim | safe }}  
 }]  
 })  
 chart\_prim.render();  
  
 const chart\_right = new CanvasJS.Chart("chart\_right", {  
 theme: "light1",  
 animationEnabled: true,  
 title: {  
 text: "Датчик правого борта",  
 fontFamily: "Roboto Slab",  
 fontSize: 22,  
 },  
 data: [{  
 type: "line",  
 dataPoints: {{ sensor\_right | safe }}  
 }]  
 })  
 chart\_right.render();  
  
 function updateData(chart, sensor\_type) {  
 $.ajax({  
 url: `${url}?sensor\_type=${sensor\_type}`,  
 type: "GET",  
 dataType: "json",  
 success: (jsonResponse) => {  
 console.log(jsonResponse)  
 console.log(chart)  
 chart.options.data[0].dataPoints = jsonResponse;  
 chart.render();  
 }  
 })  
 setTimeout(updateData, 500, chart, sensor\_type);  
 }  
 updateData(chart\_left, "left");  
 updateData(chart\_prim, "primary");  
 updateData(chart\_right, "right");  
 }  
  
 </script>  
 <div>  
 <h1 class="heading">Робот №{{ robot.pk }}</h1>  
 </div>  
 <div class="flex\_block charts">  
 <div class="chart" id="chart\_left">  
 </div>  
 <div class="chart" id="chart\_prim">  
 </div>  
 <div class="chart" id="chart\_right">  
 </div>  
 </div>  
 <div class="data\_block">  
 <div class="data">  
 <table>  
 <tr>  
 <td class="tr\_ceil">IP-адрес:</td>  
 <td class="tr\_ceil">{{ robot.ipaddr }}</td>  
 </tr>  
 <tr>  
 <td class="tr\_ceil">Кодовое имя:</td>  
 <td class="tr\_ceil">{{ robot.token }}</td>  
 </tr>  
 <tr>  
 <td class="tr\_ceil">Заряд батареи:</td>  
 <td class="tr\_ceil">{{ robot.battery\_\_battery\_num }}%</td>  
 </tr>  
 </table>  
 </div>  
 <div class="control\_block">  
 <button class="robot\_button button\_create button\_control button\_type" onclick="location.href = '{% url 'robot\_bluetooth' robot.id %}'">Ручной режим</button>  
 <button class="robot\_button button\_create button\_control">Автоматическое движение</button>  
 <button class="robot\_button button\_create button\_control button\_delete" onclick="location.href = '{% url 'deleterobot' robot.id %}'">Удалить из системы</button>  
 </div>  
 </div>  
{% endblock %}