МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

Уразаев Александр Александрович

Разработка программной части робота-дезинфектора

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

6В07113 – Робототехника и мехатроника

Алматы 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой РТиТСА
кандидат технических наук,
профессор
_____Ожикенов К. А.
«___» _____ 2024 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: «Разработка программной части робота-дезинфектора»

6В07113 – Робототехника и мехатроника

Выполнил
Уразаев Александр Александрович.

Рецензент
Кандидат технических наук, доцент
Кафедры «Физика» КазНПУ им.Абая
_____ Жаменкеев Е.К.
«___» май 2024 г.

Научный руководитель
Магистр технических наук,
старший преподаватель
_____ Баянбай Н.А.
«___» май 2024 г.

Алматы 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

6В07113 – Робототехника и мехатроника

УТВЕРЖДАЮ
Заведующий кафедрой РТиТСА
кандидат технических наук,
профессор
_____Ожикенов К. А.
«___» _____ 2024 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Уразаеву Александру Александровичу
Тема: «Разработка программной части робота-дезинфектора»
Утверждена приказом ректора_____ №548пс от «07» 12 2024 г.
Срок сдачи законченной работы «03» 06 2025 г.
Исходные данные к дипломному проекту:
Теоретические материалы о существующих роботах-дезинфекторах
Теоретические материалы о существующих программах для программирования плат Arduino
Теоретические материалы по Arduino Nano
Теоретические материалы по PID-регуляторам
Перечень подлежащих разработке в дипломном проекте вопросов:
а) Обзор существующих роботов-дезинфекторов и существующих программ для разработки программ для платформы Arduino
б) Математический анализ данных с датчиков, определение основных и дополнительных функций план робота

в) Разработка алгоритма работы робота и обзор наиболее важных участков кода
г) Сборка макета и тестирование функций робота
Перечень графического материала (с точным указанием обязательных чертежей): представлены слайдов презентации работы
Таблицы: 3
Рисунки: 27
Рекомендуемая основная литература: из 15 *наименований*

ГРАФИК

подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечания
Исследовательская часть	17.02.2024	Выполнено
Теоретическая часть	13.03.2024	Выполнено
Практическая часть	22.04.2024	Выполнено
Специальная часть	28.05.2024	Выполнено

Подписи

консультантов и норм контролера на законченную дипломную работу (проект)с указанием относящихся к ним разделов работы (проекта)

Наименование разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Нормоконтроль	Игембай Е.А., магистр техники и технологии, преподаватель	03.06.2024	
Основная часть	Баянбай Н.А., магистр технических наук, старший преподаватель	03.06.2024	
Расчетная часть	Баянбай Н.А., магистр технических наук, старший преподаватель	03.06.2024	
Программная часть	Баянбай Н.А., магистр технических наук, старший преподаватель	03.06.2024	

Научный руководитель: _____ Баянбай Н.А.

Задание принял к исполнению обучающийся: _____ Уразаев А.А.

Дата «03» 06 2024

АҢДАТПА

Бұл дипломдық жұмыс дезинфекциялық роботтың бағдарламалық жасақтаманы жазылуына арналған. Жазылған жұмыстың негізі бағдарламалау әдістемесін таңдауға айтарлықтай әсер еткен аппараттық құрал, сондай-ақ кодты құру процесін жеңілдетуге бағытталған бағдарламалық құралдар болды. Прототиптің жұмыс алгоритмін жасауға негіз болған дезинфекциялық роботтардың қолданыстағы аналогтары талданды. Жазылған бағдарлама тек дәлелденген прототипте ғана емес, сонымен қатар басқа ұқсас роботтарда да қолдануға арналған. Бағдарламалық жасақтама жүктелген Робот бөлмелердегі қабырғаларды дезинфекциялау процесін жеңілдетуге арналған.

АННОТАЦИЯ

Данная дипломная работа посвящена разработке программного обеспечения для робота-дезинфектора. Основой разработки послужила аппаратная часть, которая существенно влияла на выбор методологии программирования, а также имеющиеся библиотеки и программные средства, направленные на упрощение процесса создания кода. Были проанализированы существующие аналоги роботов-дезинфекторов, что послужило основой для разработки алгоритма работы прототипа. Разработанная программа предназначена не только для использования в проверенном прототипе, но и в других аналогичных роботах. Робот, на который было загружено разработанное программное обеспечение, предназначен для упрощения процесса дезинфекции стен в помещениях.

ABSTRACT

This diploma project is devoted to the development of software for a disinfection robot. The basis of the development was the hardware, which significantly influenced the choice of programming methodology, as well as the available libraries and software tools aimed at simplifying the code creation process. The existing analogues of disinfection robots were analyzed, which served as the basis for the development of the prototype algorithm. The developed program is intended not only for use in a proven prototype, but also in other similar robots. The robot, on which the developed software was downloaded, is designed to simplify the process of disinfection of walls in rooms.

СОДЕРЖАНИЕ

Введение		7
1	Исследовательская часть	8
1.1	Актуальность проекта	8
1.2	Методы дезинфекции	8
1.3	Обзор роботов-дезинфекторов на рынке	8
1.4	Существующие решения создания программ для роботов	12
2	Теоретическая часть	17
2.1	Принципиальная схема робота	17
2.2	Электронные компоненты	17
2.3	Принцип движения робота	18
2.4	Основные и дополнительные функции робота	24
2.5	Алгоритм работы робота	26
3	Практическая часть	31
3.1	Финансовый расчёт	31
3.2	Готовый макет с загруженной программой	32
3.3	Проверка функций робота	32
Заключение		34
Список использованной литературы		35
Приложение А		36

ВВЕДЕНИЕ

В современном мире, несмотря на достижения в медицине, продолжают возникать проблемы с распространением инфекций из-за нежелания делать прививки, игнорирования гигиенических правил и недостаточной социальной ответственности. Опыт пандемии коронавируса, вследствие которого в Казахстане погибло 19 тысяч человек показал, что общество не готово к подобным кризисам [1].

Для успешной борьбы с инфекционными угрозами требуется системный подход, включающий в себя не только меры профилактики и лечения, но и развитие передовых методов дезинфекции. Традиционные методы дезинфекции, такие как влажная уборка и обработка химическими средствами, а также обработка ультрафиолетовыми лампами в медицинских учреждениях, остаются важными, однако современные технологии предлагают новые решения, способные значительно упростить этот процесс.

Среди новых решений наиболее перспективными являются роботы-дезинфекторы, способные ориентироваться в пространстве и дезинфицировать его по ходу своего перемещения. Разработка подобных роботов требует внимания не только аппаратная часть, в которую входят управляющий микропроцессор, система движения и сбора информации об окружении, но и программное обеспечение, управляющее его движением путём анализа собранных данных и поставленной задачи.

Целью дипломного проекта является разработка и реализация программного кода для роботов-дезинфекторов. Программа, с помощью которой работает устройство – одна из важнейших составляющих любой компьютеризированных любой системы, особенно когда это касается безопасности человека.

1. Исследовательская часть

1.1 Актуальность проекта

С каждым годом население Земли устойчиво растёт. В ряде малоразвитых странах уровень существует значительная проблема с доступом к медицинской помощи. В купе с повышенным уровнем антисанитарии приводит к увеличению риска возникновения очагов инфекционных заболеваний и её дальнейшего распространения. Из недавних событий можно вспомнить как в Алмате, прошлой осенью произошла вспышка кори. Как раз в стационарах, куда помещают заражённых и необходима дезинфекция. Конечно, дезинфекция не является абсолютом, но заранее помыть яблоко гораздо проще чем вылечить больного. И чем качественнее дезинфекция будет производиться, тем проще медикам будет бороться с болезнями.

1.2 Методы дезинфекции

В современных роботах-дезинфекторах применяются два основных метода дезинфекции: химическая обработка и ультрафиолетовое облучение. Химическая обработка включает использование различных химических реагентов, таких как хлорноватистая кислота, диоксид хлора, пероксиуксусная кислота и перекись водорода [2]. Однако данный метод имеет существенный недостаток в виде постоянного пополнения реагентов, что требует дополнительных затрат времени и ресурсов. Более эффективным и рациональным методом дезинфекции является облучение помещений ультрафиолетовым излучением. Важным преимуществом ультрафиолетового излучения является то, что оно способно обрабатывать помещение целиком, в то время как химические реагенты дезинфицируют только в той области, где были нанесены.

1.3 Обзор роботов-дезинфекторов на рынке

Распыляющие роботы-дезинфекторы:

На рисунке 1.1 представлен – робот китайской компании BIOBASE. BKS-Y-800 – может автоматически перемещаться к зоне для полного распыления и поддерживает управление с помощью мобильного приложения для обеспечения разделения человека и машины и минимизации воздействия на персонал. Благодаря проработанному ПО его легко обслуживать и управлять его действиями, что является несомненным преимуществом среди конкурентов [3].

Рисунок 1.1 – Робот BKS-Y-800

Puductor (рисунок 1.2), – произведённый компанией Pudu Robotics, базирующейся в Шэньчжэне. Движение осуществляется автономно, независимо от человека. В его систему определения пространства входят два датчика от компании Intel – Realsense RGBD, что позволяет роботу быстро определять препятствия на пути и корректировать свой маршрут [4]. В сравнении с другими роботами-дезинфекторами данного способа дезинфекции, Puductor имеет самый большой резервуар для дезинфицирующей жидкости – на 23.8 л.

Рисунок 1.2 – Робот Puductor

Puductor 2, показанный на рисунке 1.3 – усовершенствованная модель Puductor. Как и его предшественник, имеет полное автоматическое управление. На данный момент это единственный робот, совмещающий в себе распылительную и ультрафиолетовую дезинфекцию. При обнаружении человека в радиусе 3-х метров робот перестаёт распылять дезинфицирующий раствор, а также перекрывает поток излучения от ламп [5].

Рисунок 1.3 – Робот Puductor 2

В таблице 1.1 приведено сравнение технических характеристик роботов-дезинфекторов с распыляющим методом дезинфекции.

Таблица 1.1 – Сравнение характеристик распыляющих роботов

Модель робота	BKS-Y-800	Puductor	Puductor 2
Изображение			
Внешний размер (мм)	500x1335x700	516x500x1288	544x538x1290
Эффективность распыления	2 – 3 л/ч	2 л/ч	0.5 – 2 л/ч
Объем резервуара	16 л	23.8 л	15 л
Скорость движения	0 – 0.6 м/с	0 – 1.2 м/с	0 – 1.2 м/с
Источник питания	AC 220 ± 10% В	AC 220 ± 10% В	AC 220 ± 10% В
Потребляемая мощность	150 Вт	150 Вт	150 Вт
Время работы	6 ч	4 ч	6 ч
Время зарядки	4 ч	4.5 ч	4 ч
Общий вес	72 кг	35 кг	45 кг

Роботы, дезинфицирующие ультрафиолетовым излучением:

Представленный на рисунке 1.4 робот SIFROBOT-6.53 – робот, созданный компанией SIFROBOT, основанной в Калифорнии в 2005 году. Для дезинфекции робот оснащён 8-ю ультрафиолетовыми лампами, конструкционно охватывающие 360º вокруг робота [6]. Относительно конкурентов имеет малый вес в 45 кг, что позволяет тратить больше энергии на излучение, нежели на перемещение. Траекторию движения может строить самостоятельно, а также её можно назначить вручную.

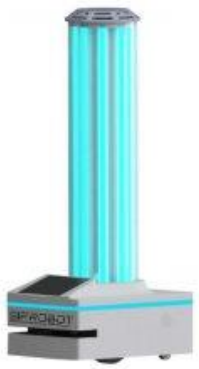


Рисунок 1.4 – Робот SIFROBOT-6.53

SEIT-UV (рисунок 1.5) – произведён турецкой компанией Milvus Robotics. За 10 минут этот робот может обработать комнату в 25 кв м. Имеет множество функций предотвращения облучения человека, такие как: 2D LIDAR, 3D-камера, кнопки аварийной остановки и индикаторные светодиоды. [7]

Рисунок 1.5 – Робот SEIT-UV

Model C – робот, изображённый на рисунке 1.6, датской компании UVD Robots [8]. В сравнении с конкурентами имеет самый большой вес – 140 кг, из-за чего имеет минимальное время работы в 2.5 часа. При этом время заряда аккумулятора составляет 4 ч. Этот робот тратит большую часть энергии на передвижение, нежели на излучение, что делает его менее эффективным.

Рисунок 1.6 – Робот Model C

В таблице 1.2 приведено сравнение технических характеристик роботов-дезинфекторов, использующих ультрафиолетовый метод дезинфекции.

Таблица 1.2 – Сравнение характеристик роботов с УФ лампой

Модель робота	SIFROBOT-6.53	SEIT-UV	Model C
Изображение			
Внешний размер (мм)	635x486x1530	516x500x1288	930x660x1710
Длина волны	253.7 нм	254 нм	254 нм
Скорость движения	0 – 0.7м / с	0 – 1.5м / с	0 – 1,5 м/с
Источник питания	AC 220 ± 10% В	AC 220 ± 10% В	AC 220 ± 10% В
Потребляемая мощность	200 Вт	150 Вт	200 Вт
Время работы	6 ч	3 ч	2.5 ч
Время зарядки	2 ч	3 ч	4 ч
Общий вес	45 кг	120 кг	140 кг

1.4 Существующие решения создания программ для роботов

На сегодняшний день существует множество способов создания программ для роботов, которые обеспечивают их функциональность. Основные методы включают использование различных языков программирования, специализированных платформ и сред разработки, а также интеграцию сенсоров и исполнительных механизмов.

ROS (Robot Operating System) – это набор программных библиотек и инструментов, предназначенных для упрощения процесса создания сложного и

масштабируемого программного обеспечения для роботов [9]. Для создания роботов на базе ROS используются языки программирования Python и C++. В число инструментов данного фреймворка входят:

Rviz. Визуализатор данных робота. Позволяет отображать данные с сенсоров, траектории движения и другие параметры в реальном времени, что облегчает отладку и настройку.

Gazebo. Позволяет тестировать программное обеспечение робота в виртуальной среде, моделируя физические взаимодействия и работу сенсоров.

С помощью различных библиотек, таких как slam_toolbox и Nav2 можно создавать роботизированные системы, способные самостоятельно ориентироваться в пространстве и корректировать маршрут используя технологию SLAM (simultaneous localization and mapping).

MATLAB. MATLAB это высокоуровневый язык программирования и интерактивная среда для численных вычислений, визуализации и программирования [10]. Он предоставляет мощный язык программирования с богатым набором функций для математических операций, включая линейную алгебру, статистику, и обработку сигналов. Matlab и его расширение Simulink часто используются для моделирования, симуляции и управления роботами и автоматизированными системами. На сегодняшний день Matlab имеет большое количество дополнительных пакетов, расширяющих его возможности в различных областях, таких как машинное обучение (Statistics and Machine Learning Toolbox), обработка изображений (Image Processing Toolbox), контроль и управление (Control System Toolbox), и многие другие.

CopeliaSim – программное обеспечение для моделирования и симуляции роботов и автоматических систем разработанное компанией Coppelia Robotics. CopeliaSim предоставляет широкий спектр возможностей для создания и тестирования различных робототехнических систем, начиная от промышленных манипуляторов и мобильных роботов и заканчивая более сложными системами [11]. В данной среде разработки возможно самостоятельное создание пространства для симуляции, имитация различных датчиков (расстояния, лазерные, камеры). CopeliaSim поддерживает различные языки программирования, такие как Python, C/C++, MATLAB, Lua, что позволяет разработчикам интегрировать свои собственные алгоритмы и контроллеры в симулируемую среду.

Raspberry Pi. Это одноплатный компьютер, использующийся в качестве основного вычислительного процессора в создаваемых роботизированных системах [12]. Raspberry Pi позволяет создавать роботов как на своей собственно операционной системе Raspberry Pi OS, так и интегрироваться с ROS. В собственной операционной системе предусмотрены различные библиотеки, такие как:

RPi.GPIO: для управления GPIO – общими входами-выходами на Raspberry Pi.

OpenCV: Библиотека для компьютерного зрения, позволяющая роботам распознавать объекты и ориентироваться в пространстве.

picamera: для работы с камерой Raspberry Pi.

Arduino. Это популярная платформа для создания электронных проектов и роботов. Включает в себя различные микроконтроллеры Arduino Uno, Arduino Mega, Arduino Nano и т. д., а также собственную среду разработки Arduino IDE [13]. Благодаря своей простоте данная платформа получила огромную популярность, в связи с чем создание роботов на базе Arduino является самым простым способом создания робота из всех перечисленных. Для плат Arduino было создано множество библиотек, упрощающих создание кода. Также, помимо Arduino IDE для упрощения работы с микроконтроллером разработано множество сред разработки и моделирования:

Tinkercad. Платформа разработана компанией Autodesk для обучения молодых пользователей блочному программированию и построению схем. Tinkercad позволяет проводить симуляцию созданных схем, что ограничивает риски повреждения реальных компонентов в ходе их тестирования (рисунок 1.7).

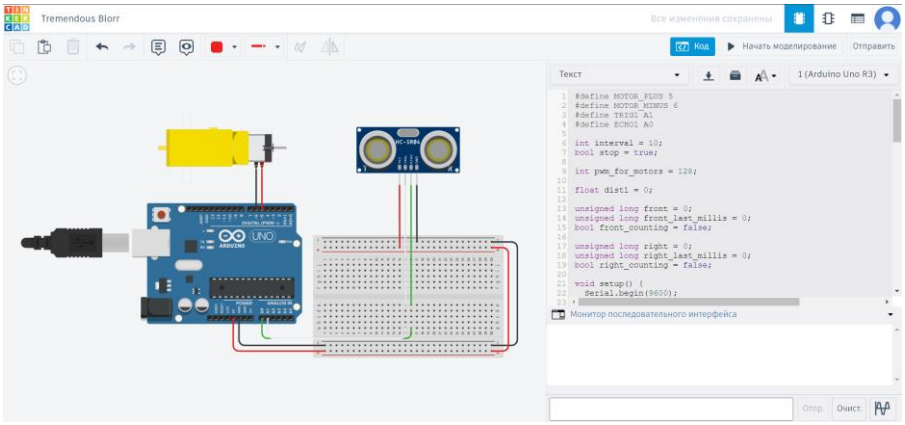


Рисунок 1.7 – Интерфейс программы Tinkercad.

Fritzing – это программное обеспечение для проектирования электронных схем, разработки печатных плат и создания схематических диаграмм для электронных проектов. Большим преимуществом данной программы является обширная библиотека электронных компонентов и модулей. Также любой пользователь может создать собственный электронный компонент и поделиться им на официальном форуме. Файл расширения программы (.fzz) содержит в себе визуальное представление электронной схемы, принципиальную схему, печатную плату с электронными компонентами и файл с кодом который должен управлять микроконтроллером. Интерфейс данной программы изображён на рисунке 1.8.

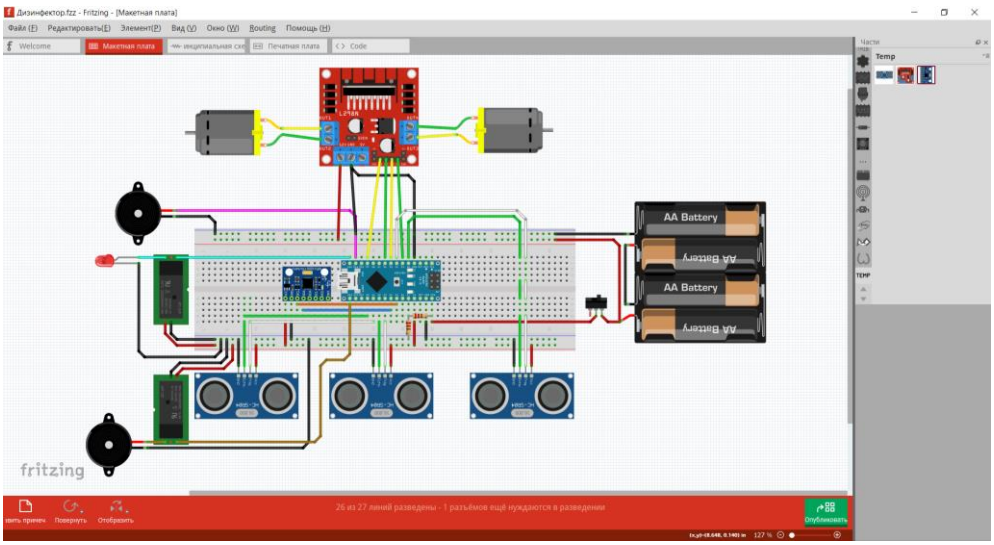


Рисунок 1.8 – Интерфейс программы Fritzing.

Wokwi – это онлайн-платформа для разработки и тестирования встраиваемых систем и проектов. Она предоставляет веб-интерфейс, который позволяет пользователям создавать, моделировать и тестировать аппаратные проекты без необходимости установки дополнительного программного обеспечения на своем компьютере. Платформа поддерживает различные микроконтроллеры, такие как: Arduino, ESP8266, ESP32, STM32, Pi Pico и др. На данный момент Wokwi обладает лучшей средой симуляции электронных схем на базе микроконтроллеров (рисунок 1.9).

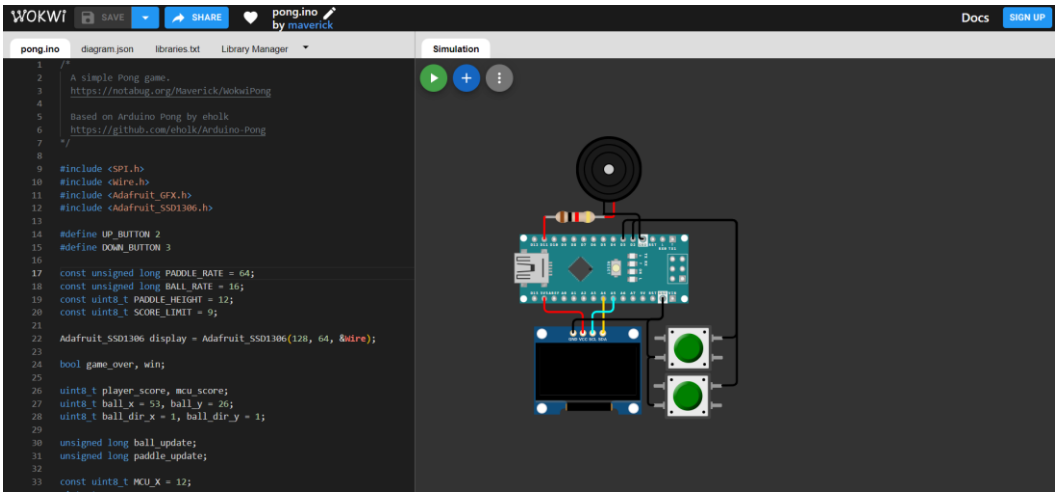


Рисунок 1.9 – Интерфейс программы Wokwi.

Proteus – это программное обеспечение для моделирования электронных схем, разработки печатных плат и симуляции микроконтроллеров. Программа обеспечивает точную симуляцию электронных схем, включая работу различных компонентов, таких как резисторы, конденсаторы, микроконтроллеры, интегральные схемы и другие. Программа поставляется с обширной

библиотекой электронных компонентов, что позволяет пользователям легко создавать свои схемы и проекты. Единственным недостатком в сравнении с предыдущими программами является не самый интуитивно понятный интерфейс (рисунок 1.10).

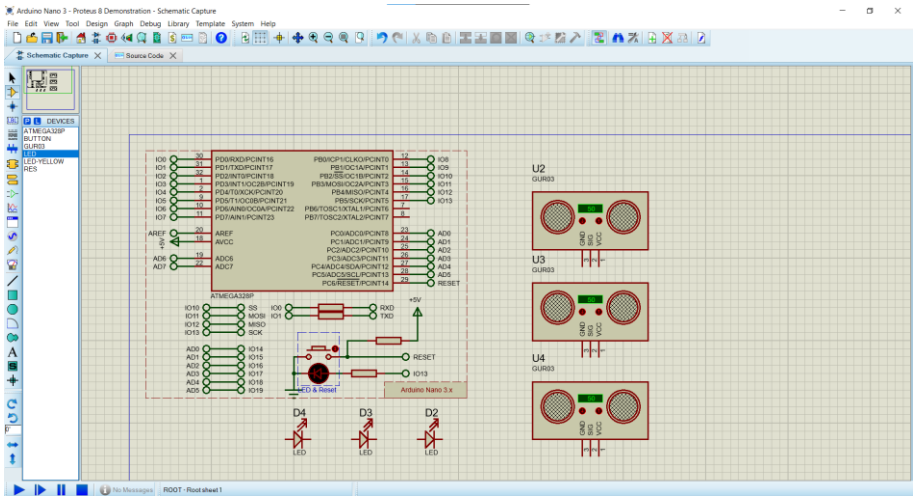


Рисунок 1.10 – Интерфейс программы Proteus.

2. Теоретическая часть

2.1 Принципиальная схема робота

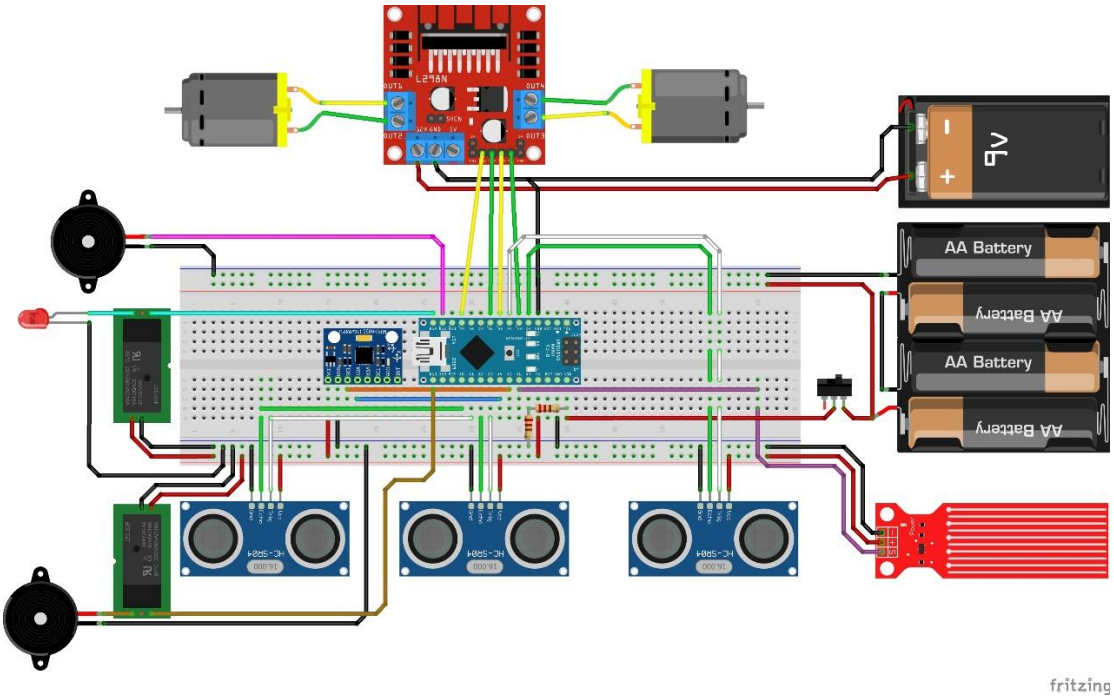


Рисунок 2.1 – Принципиальная схема робота

2.2 Электронные компоненты робота

– Микроконтроллер Arduino nano;
– Гироскоп + Акселерометр MPU-6050;
– Ультразвуковые датчики HC-SR04;
– Драйвер L298N;
– TT мотор-редукторы с колёсами;
– Пьезоэлемент Arduino
– Реле JQC-3FF-S-Z;
– Светодиодная лента;
– Ультразвуковой распылитель;
– Датчик уровня воды T1592;
– Отсек для батареек;
– Выключатель;
– Резисторы на 10 кОм;
– 2 кабеля USB
– Макетная плата;
– Комплект проводов;

2.3 Принцип движения робота

Разработка принципов движения робота начинается с постановления его основных функций. В моём случае это движение робота по помещению вдоль стены и дезинфицирование её поверхности в процессе движения.

Для реализации движения робота я буду использовать «правило одной руки», которое гласит: двигаясь по лабиринту, надо все время касаться правой или левой рукой его стены. Данный метод позволит обработать все стены в помещении на уровне высоты робота.

В качестве “руки” будут использованы ультразвуковые датчики HC-SR04. Данный датчик измеряет время, которое прошла звуковая волна от микрофона до объекта и обратно в динамик. Далее измеренное значение времени передаётся на микроконтроллер. Для обработки сигнала с датчика будет использована формула, для преобразования времени в расстояние:

$$S = \frac{v \cdot t}{2} \tag{2.1}$$

где: S – расстояние до объекта;

v – скорость звука в воздухе;

t – время полета ультразвукового сигнала от датчика до объекта и обратно.

Модуль MPU-6050 представляет собой MEMS (micro-electromechanical systems) датчик, в конструкции которого имеется гироскоп, акселерометр и датчик температуры. Для данного робота понадобятся только гироскоп и акселерометр. Общение датчика с микроконтроллером происходит через интерфейс I2C. Датчик передаёт данные об ускорении по трём осям (акселерометр) и угловой скорости по трём осям (гироскоп).

Ориентацию объекта в пространстве принято определять через углы относительно его осей координат, которые обозначаются как крен (roll), тангаж (pitch) и рыскание (yaw). Эти параметры описывают вращение объекта вокруг соответствующих осей (рисунок 2.2).

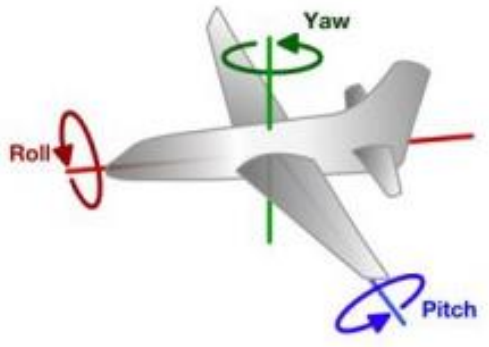


Рисунок 2.2 – Ориентация объекта в пространстве

Для нахождения этих углов используются одновременно данные с гироскопа и акселерометра, которые математически обрабатываются для получения окончательных значений углов.

Данные с акселерометра обозначим через AccX, AccY и AccZ. Для вычисления углов крена (и тангажа на основе данных акселерометра используются следующие формулы:

$$accAngleX = arctan\left(\frac{AccY}{\sqrt{AccX^2 + AccZ^2}}\right) \cdot \frac{180}{\pi}, \quad (2.2)$$

$$accAngleY = arctan\left(\frac{-1 \cdot AccX}{\sqrt{AccY^2 + AccZ^2}}\right) \cdot \frac{180}{\pi} \quad (2.3)$$

где: accAngleX – составляющая угла крена;
accAngleY – составляющая угла тангажа.

Углы на основе данных с гироскопа обозначим через GyroX, GyroY и GyroZ. Их составляющие углов вычисляются по формулам:

$$gyroAngleX = GyroX \cdot Time \quad (2.4)$$

$$gyroAngleY = GyroY \cdot Time \quad (2.5)$$

$$yaw = GyroZ \cdot Time \quad (2.6)$$

где: Time – время, в течение которого на гироскоп воздействует угловая скорость.

Для итогового вычисления углов крена и тангажа используется комбинация вычисленных углов акселерометра и гироскопа с коэффициентами 0.96 и 0.04:

$$roll = 0.96 \cdot gyroAngleX + 0.04 \cdot accAngleX, \quad (2.7)$$

$$pitch = 0.96 \cdot gyroAngleY + 0.04 \cdot accAngleY. \quad (2.8)$$

В модели робота этот датчик необходим для поддержания его траектории, обеспечивая контроль отклонения от начального угла движения. Для корректировки движения робота, чтоб он ехал точно по прямой, в код будет добавлен PID регулятор

PID регулятор – это тип управляющего устройства, используемого в автоматическом регулировании систем [14]. Он состоит из трех основных

компонентов: пропорциональной, интегральной и дифференциальной составляющих. При отклонении робота от курса, текущее значение угла становится отличным от начального, таже и для расстояния от боковой стороны робота до стены, на основе этого я создал 2 переменные, отклонение угла обозначенное через yawError и отклонение расстояния, обозначенное через distError их значение вычисляется через разницу между начальным углом и текущим:

$$yawError = targetYaw - yaw, \tag{2.9}$$

$$distError = targetDistance - dist \tag{2.10}$$

где: yaw – начальное значение угла;
targetYaw – текущее значение угла;
dist – начальное значение расстояния;
targetDistance – текущее значение расстояния.

Из значений ошибки угла и ошибки мы получаем единую ошибку (combinedError) отклонения путём суммирования этих ошибок:

$$combinedError = yawError + distError \tag{2.11}$$

Для устранение этой ошибки используется PID регулятор. Его пропорциональная, интегральная и дифференциальные составляющие рассчитываются следующим образом:

Пропорциональная – значение ошибки в текущий момент, умноженное на коэффициент пропорциональности:

С точки зрения математики:

$$P = K_p \cdot e(t) \tag{2.12}$$

где: P – пропорциональная составляющая регулятора;
Kp – коэффициент пропорциональности;
e(t) –значение ошибки в текущий момент.

С точки зрения программы:

$$proporcional = K_p \cdot combinedError \tag{2.13}$$

где: proporcional – пропорциональная составляющая регулятора.

Интегральная – сумма ошибок от момента появления первой ошибки до текущей, умноженных на разницу между временем появления первой ошибки и появлением последней, умноженное на коэффициент интеграции:

С точки зрения математики:

$$I = K_i \int_0^t e(t)dt \tag{2.14}$$

где: I – интегральная составляющая регулятора;
Ki – коэффициент интеграции.

С точки зрения программы:
Сперва создаётся переменная, которая является суммой произведений значений ошибок на время, между ошибками:

$$sumError \mathrel{+}= combinedError \cdot deltaTime \tag{2.15}$$

где: sumError – сумма ошибок;
deltaTime – время между предыдущей ошибкой и текущей.

Далее для получения интегральной составляющей эта сумма умножается на коэффициент интеграции:

$$integral = K_i \cdot sumError \tag{2.16}$$

где: integral – интегральная составляющая регулятора.

Дифференциальная – отношение разницы между текущей ошибкой и предыдущей и времени текущей ошибки и прошлой, умноженной коэффициент дифференциации:
С точки зрения математики:

$$D = K_d \cdot \frac{de(t)}{dt} \tag{2.17}$$

где: D – дифференциальная составляющая регулятора;
Kd – коэффициент дифференциации.

С точки зрения программы:

$$derivative = K_d \cdot (combinedError - previousError)/deltaTime \tag{2.18}$$

где: derivative – дифференциальная составляющая регулятора;
previousError – значение предыдущей ошибки.

Общее корректирующее значение регулятора является суммой его составляющих:

С точки зрения математики:

$$u(t) = P + I + D \tag{2.19}$$

где: u(t) – корректирующее значение в текущий момент времени;

С точки зрения программы:

$$correction\ = proporcional + integral\ + derivative \tag{2.20}$$

где: correction – корректирующее значение в текущий момент времени;

Значение коэффициентов подбирается экспериментально, основываясь на лучшем результате коррекции.

Возвращаясь к принципу “одной руки”, можно составить следующие действия, которые будет выполнять робот:

После включения робота проверяется информация с боковых датчиков. Датчик, расстояние которого будет составлять 20 сантиметров, будет использован в качестве той самой руки. Также робот проверяет расстояние с переднего датчика, по данным с этих двух датчиков и корректировке положения с помощью MPU-6050 робот будет двигаться следующим образом:

Если расстояние от бокового датчика до стены составляет 20 сантиметров и расстояние от переднего датчика до стены впереди больше 20 сантиметров, то робот едет прямо (рисунок 2.2).

Если расстояние от бокового датчика до стены составляет 20 сантиметров и расстояние от переднего датчика до стены впереди меньше или равно 20 сантиметрам, то робот поворачивает направо (рисунок 2.3).

Если расстояние от бокового датчика до стены стало разительно больше (в коде указано 40 сантиметров) и расстояние от переднего датчика до стены больше 20 сантиметров, то робот поворачивает налево (рисунок 2.4).

Визуализировать 3 этих случая можно следующим образом:

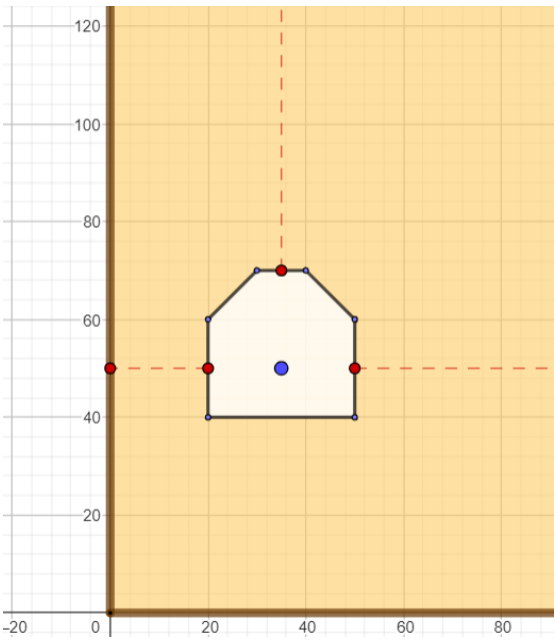


Рисунок 2.3 – Движение вперёд

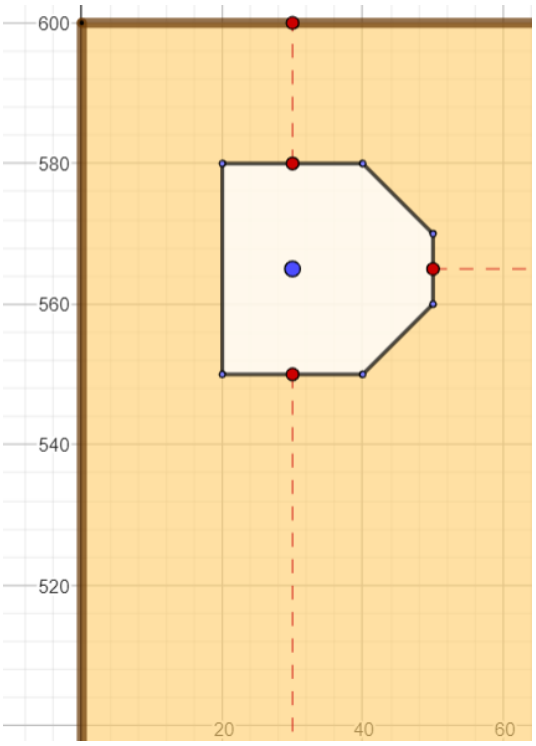


Рисунок 2.4 – Поворот направо

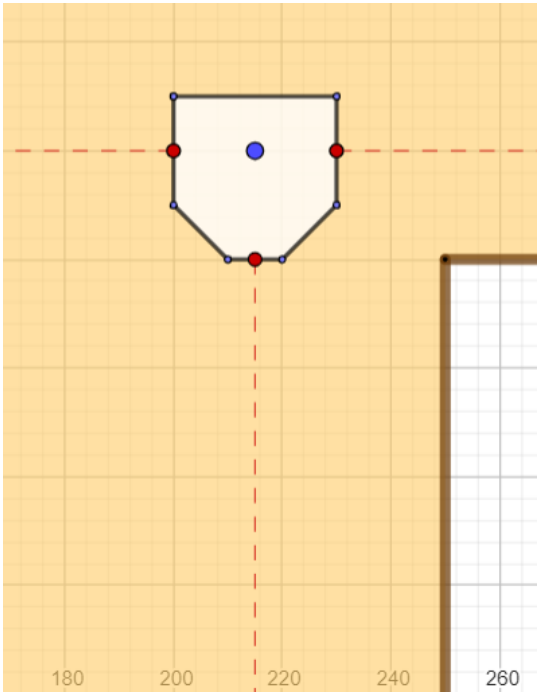


Рисунок 2.5 – Поворот налево

2.4 Основные и дополнительные функции робота

Помимо движения главной задачей робота-дезинфектора – дезинфицировать, в данном случае поверхность стен. В качестве дезинфицирующих средств были выбраны ультрафиолетовая лампа и ультразвуковой распылитель, распыляющий химический раствор. В целях безопасности, на прототипе ультрафиолетовая лампа была заменена светодиодной лентой, а ультразвуковой распылитель будет распылять воду. Данного подхода достаточно для демонстрации возможностей робота. Заменить светодиодную ленту на УФ лампу, можно поставив повышающий преобразователь 5 DC – 220 AC, и также управлять её работой через реле. Заменить воду на раствор тоже крайне просто.

Также для разработки программной части будет полезно рассмотрение следующих гипотез:

Гипотеза 1: робот проехал весь периметр помещения.

В своей работе я предусмотрел следующий вариант решения этой гипотезы:

В раздели математики под названием «Теория графов» существует понятие «Циклический маршрут» [15]. Это граф, начальная и конечная вершины которого совпадают. Путь робота можно представить в виде графа с вершинами, в которых он поворачивает, что показано на рисунке 2.6.

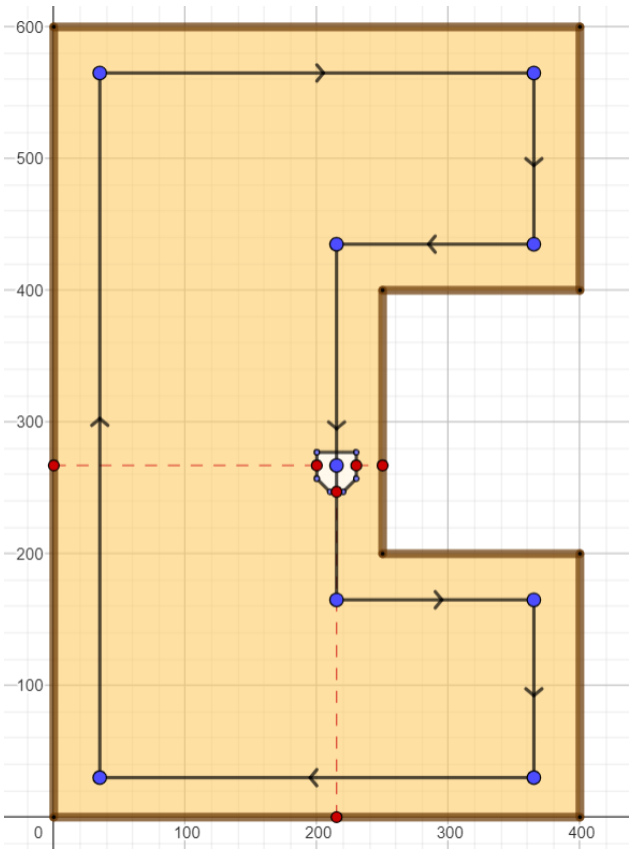


Рисунок 2.6 – Траектория движения робота

Путь, пройденный роботом по данной траектории, складывается из расстояний, которые он проехал вперёд, влево, вправо и назад. Для упрощения вычислений я буду использовать не пройденное расстояние а время, которое робот проехал в том, или ином направлении. Совпадение временных промежутков, пройденных в противоположных направлениях, будет означать возвращение робота в первоначальное положение, после чего последует его остановка.

Гипотеза 2: в результате воздействия внешних факторов робот не удержал равновесие и упал.

В своём коде и в схеме мною был предусмотрен пьезоэлемент. При отклонении угла крена или тангажа больше чем на 5°, активируется участок кода, который будет подавать ток на пьезоэлемент, который в свою очередь начнёт подавать звуковой сигнал. Также при этом будут отключены системы дезинфекции, чтобы оператор или находящиеся рядом люди вернули робота в исходное положение.

Гипотеза 3: аккумулятор робота разрядился.

В схеме робота имеется делитель напряжения, плата Arduino nano принимает с него сигнал, при падении напряжения питания ниже 5.5 В, робот отключит системы дезинфекции и включит пьезоэлемент.

Гипотеза 4: в баке робота закончилась дезинфицирующая жидкость.

Отсутствие жидкости для распыления в контексте моего робота представляет весомую угрозу. Ультразвуковые распылители охлаждаются за счет жидкости, которую испаряют. Без неё перегрева распылителей и их последующего выхода не избежать. В целях обезопасить конструкцию от данного исхода в конструкции предусмотрен датчик уровня жидкости T1592. При отсутствии воды в баке датчик подаст сигнал на отключение распылителей, что остановит их возможный нагрев.

2.5 Алгоритм роботы робота

Исходя из функций, которые робот должен выполнять, был разработан алгоритм, представленный на рисунке 2.6:

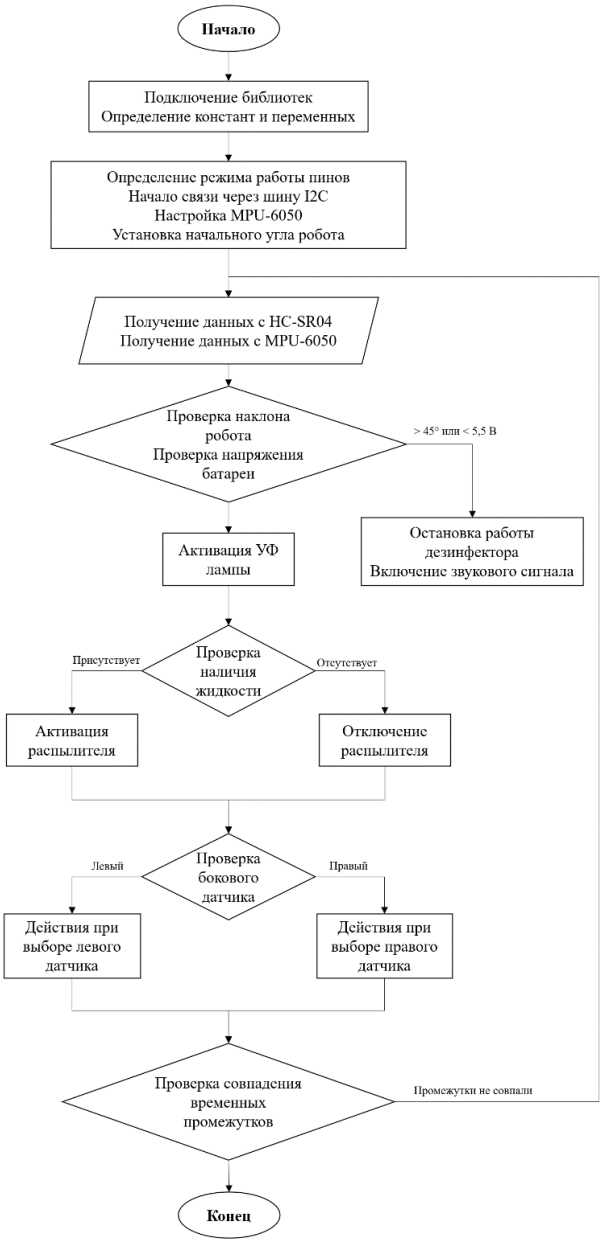


Рисунок 2.6 – Алгоритм работы робота

1) В блоке подключения библиотек я подключаю библиотеку “Wire.h” для коммуникации по протоколу I2C с датчиком MPU-6050. Также я определяю константы и переменные, необходимые для корректной работы программы.

2) Во второй блок описывает функцию setup() (рисунок 2.7), в которой настраивается режим работы пинов, инициализируется датчик MPU-6050 и устанавливается нулевое значение для ориентации робота.

```
void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);
  calculateError();
  delay(20);
  currentTime = micros();
  pinMode(MOTOR1_PLUS, OUTPUT);
  pinMode(MOTOR1_MINUS, OUTPUT);
  pinMode(MOTOR2_PLUS, OUTPUT);
  pinMode(MOTOR2_MINUS, OUTPUT);
  pinMode(TRIG1, OUTPUT);
  pinMode(ECHO1, INPUT);
  pinMode(TRIG2, OUTPUT);
  pinMode(ECHO2, INPUT);
  pinMode(TRIG3, OUTPUT);
  pinMode(ECHO3, INPUT);
  pinMode(BATTERY, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(SPRAYER, OUTPUT);
  pinMode(BUZZER, OUTPUT);
}
```

Рисунок 2.7 – Функция setup()

3) Далее, управление передается функции loop(), где сначала считываются данные с ультразвуковых датчиков через функцию getDist(), а также данные с датчика MPU-6050 функцией getAngles() (риунок 2.8).

```
float getDist(int trig_pin, int echo_pin) {
  digitalWrite(trig_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_pin, LOW);
  uint32_t duration = pulseIn(echo_pin, HIGH);
  return duration / 58.3;
}

void getAngles(){
  readAcceleration();
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - AccErrorX;
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) - AccErrorY;
  previousTime = currentTime;
  currentTime = micros();
  elapsedTime = (currentTime - previousTime);
  readGyro();
  GyroX -= GyroErrorX;
  GyroY -= GyroErrorY;
  GyroZ -= GyroErrorZ;
  gyroAngleX += GyroX * elapsedTime;
  gyroAngleY += GyroY * elapsedTime;
  gyroAngleZ += GyroZ * elapsedTime;
  roll = 0.96*gyroAngleX + 0.4*accAngleX;
  pitch = 0.96*gyroAngleY + 0.4*accAngleY;
  yaw += GyroZ * elapsedTime;
}
```

Рисунок 2.8 – Функции getDist() и getAngles()

4) В следующем блоке проверяется наклон робота и уровень заряда батареи. Если один из этих параметров выходит за допустимые пределы, происходит остановка дезинфекции и срабатывает пьезоэлемент. В противном случае код продолжает выполняться.

5) На следующем шаге активируется функция getDisinfect(), которая активирует светодиодную ленту, после чего проверяется условие наличия жидкости в баке робота. Если вода присутствует, то активируется ультразвуковой распылитель если нет, то распылитель не включается. Этот участок алгоритма показан на рисунке 2.9.

```
void getDisinfect() {
  digitalWrite(LED, HIGH);
  waterState();
  if (waterPresent){
    digitalWrite(SPRAYER, HIGH);
  }
  else {
    digitalWrite(SPRAYER, LOW);
  }
}
```

Рисунок 2.9 – Функции getDisinfect()

6) Затем программа определяет, с какой стороны от робота находится стена, сравнивая текущее расстояние, измеренное боковыми датчиками, с порогом в 20 сантиметров.

7) В зависимости от активированного датчика запускается алгоритм

движения робота. На рисунке 2.10 показаны действия робота, в зависимости от стены, вдоль которой он едет.

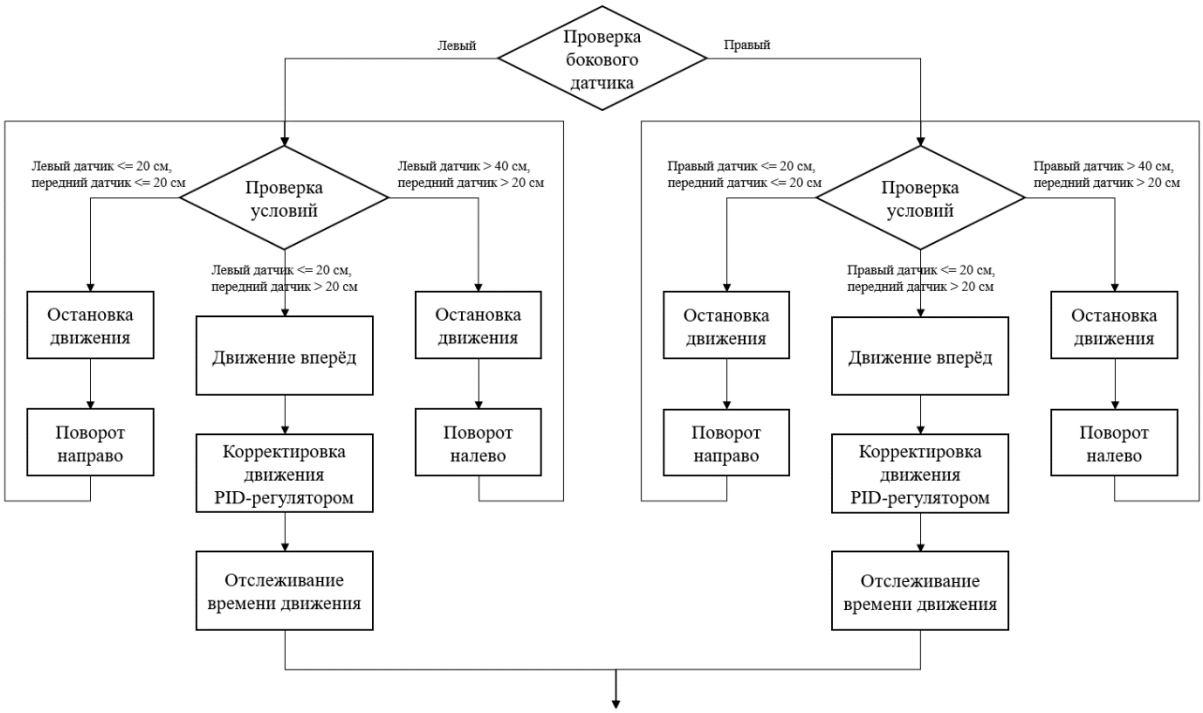


Рисунок 2.10 – Алгоритм движения при активном левом датчике

8) В зависимости от условий робот будет поворачивать вправо, влево или ехать прямо. Для каждого движения предусмотрена функция. Время, затраченное на поворот, не учитывается в расчёте пройденного пути.

```
void turnLeft() {
  analogWrite(MOTOR1_PLUS, pwm_for_motors);
  analogWrite(MOTOR1_MINUS, 0);
  analogWrite(MOTOR2_PLUS, pwm_for_motors);
  analogWrite(MOTOR2_MINUS, 0);
}
```

Рисунок 2.11 – Пример функции движения

9) При движении вперёд активируются функции PID-регулятора (рисунок 2.12) и измерения времени, в течение которого движется робот.

```
void getPID(float angle, float distn, float deltaTime){
  float yawError = (targetYaw + angle) - yaw;
  float distError = targetDistance - distn;
  float combinedError = yawError + distError;

  integral += combinedError * deltaTime;
  float derivative = (combinedError - previousError) / deltaTime;
  float correction = Kp * combinedError + Ki * integral + Kd * derivative;
  previousError = combinedError;

  motor1Speed = pwm_for_motors + correction;
  motor2Speed = pwm_for_motors - correction;
}

void getTime(uint32_t& side, uint32_t& side_last_millis, bool& side_counting) {
  if (!side_counting) {
    side_counting = true;
    side_last_millis = millis();
  }
  uint32_t currentMillis = millis();
  side += (currentMillis - side_last_millis);
  side_last_millis = currentMillis;
}
```

Рисунок 2.12 – Функция PID-регулятора и отсчёта времени

10) Далее выполняется проверка совпадения временных промежутков, в течение которых робот двигался в противоположные стороны (рисунок 2.13).

```
if (front > 0 && back > 0 && right > 0 && left > 0 && front - back <= fault && left - right <= fault) {
  front = 0;
  right = 0;
  back = 0;
  left = 0;
  do_it = true;
  return;
}
```

Рисунок 2.13 – Проверка пройденного расстояния

11) Если робот не объехал весь периметр помещения, то он продолжит ехать. Если же проехал, то программа обнулит пройденное расстояние и завершит дезинфекцию.

3. Практическая часть

3.1 Финансовый расчёт

Таблица 3.1 – Финансовый расчёт робота

Товар	Кол-во	Цена
Плата Arduino nano	1	1400
Гироскоп + акселерометр MPU-6050	1	900
Ультразвуковые датчики HC-SR04	3	1800
Драйвер L298N	1	600
TT мотор-редукторы с колёсами	2	1600
Пьезоэлемент Arduino	1	400
Реле JQC-3FF-S-Z	2	800
Светодиодная лента	1	5000
Ультразвуковой распылитель	1	700
Отсек для батареек	1	300
Пальчиковая батарейка	4	300
Батарея крона 9В	1	500
Выключатель	1	100
Резистор 10кОм	2	30
Итого		**14430**

3.2 Готовый макет с загруженной программой

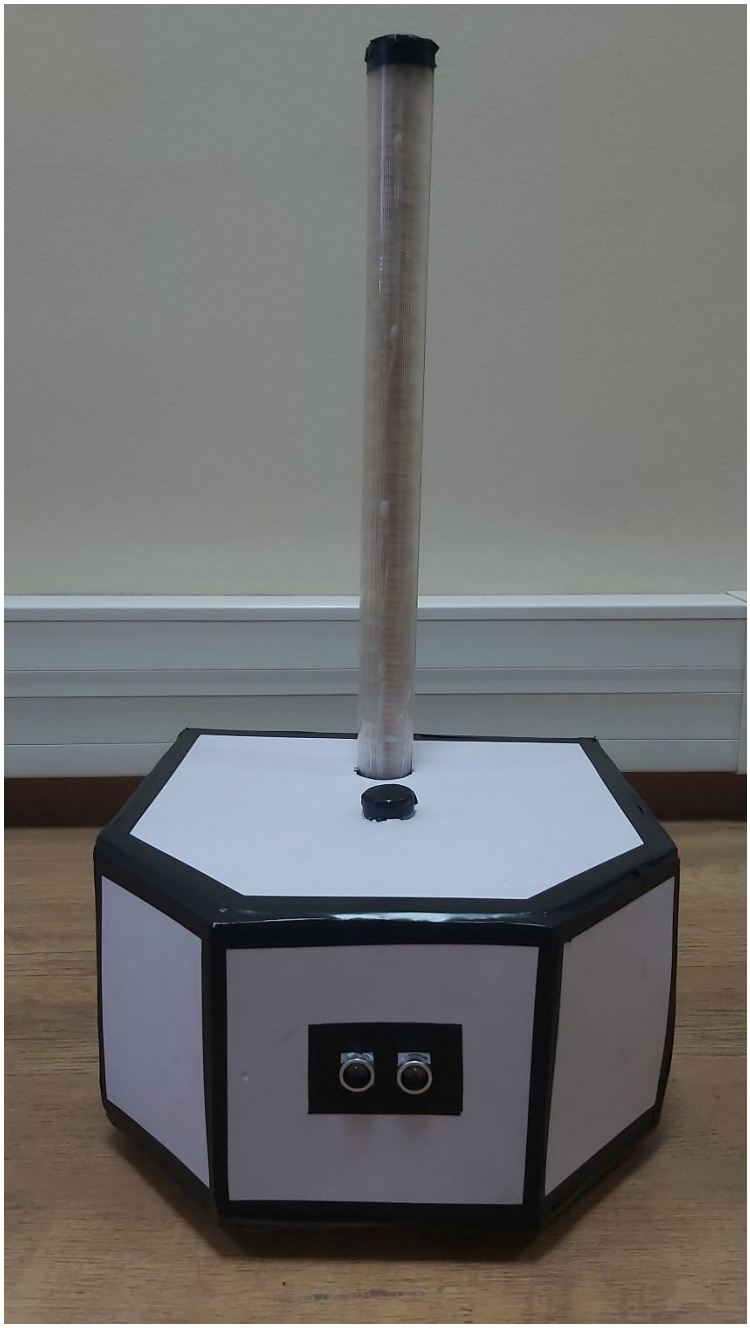


Рисунок 3.1 – Макет робота с программой

3.3 Проверка функций робота

Рисунок 3.2 – Дезинфекция ультрафиолетом

Рисунок 3.3 – Дезинфекция химическим раствором

ЗАКЛЮЧЕНИЕ

В данной дипломной работе были исследованы различные роботы-дезинфекторы, имеющиеся на рынке, рассмотрены методы разработки программной части роботов, а также программы для создания схем и кода для платформы Arduino.

В рамках разработки программы для робота, были рассмотрены подходы к управлению движением робота, включая его методы навигации по заданным траекториям, системы с обратной связью с использованием датчиков расстояния HC-SR04 и угловой ориентации, а также алгоритмы ПИД-регулирования для корректного управления моторной системой. Разработанные алгоритмы позволяют роботу эффективно перемещаться внутри помещений, тем самым обеспечивая полное и качественное покрытие поверхностей распыляемым дезинфицирующим средством.

Были разработаны механизмы контроля распыления, включая систему контроля уровня жидкости и автоматическое выключение распылителя при недостатке дезинфицирующего раствора. Разработаны системы обнаружения падения робота и разрядки аккумуляторов. Итоговый алгоритм полностью соответствует поставленным задачам.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Диана Слинькова, Три года, один месяц и 24 дня: Минздрав подвёл итоги пандемии Covid-19 – Электронный ресурс https://informburo.kz/novosti/tri-goda-odin-mesyac-i-24-dnya-minzdrav-podvyol-itogi-pandemii-covid-19

[2] Андрей Канавалов, Роботы-дезинфекторы: назначение и перспективы использования – Электронный ресурс https://vektorus.ru/blog/robot-dezinfektor.html

[3] Виктор Михайлов, Робот распыляющей дезинфекции BKS-Y-800 – Электронный ресурс https://ru.biobase.com/product/atomizing-disinfection-robot

[4] Николай Гальцкий, Puductor – Электронный ресурс https://robotplace.io/product/puductor/

[5] Николай Гальцкий, Робот дезинфектор Puductor 2 – Электронный ресурс https://robotec.ru/products/servisnye-roboty/roboty-dezinfektory/robot-dezinfektor-puductor-2.html

[6] Xao Chen, Autonomous UVC Disinfection Robot: SIFROBOT-6.53 – Электронный ресурс https://sifsof.com/product/autonomous-uvc-disinfection-robot-sifrobot-6-53/

[7] MilVUS ROBOTICS, SEIT-UV logo Cutting-edge UV Disinfectant – Электронный ресурс https://milvusrobotics.com/products/seit-uv

[8] UVO ROBOTS, Autonomous UV-C disinfection robot – Электронный ресурс https://uvd.blue-ocean-robotics.com/features

[9] ROS (Robot Operating System), wikipedia – Электронный ресурс https://ru.m.wikipedia.org/wiki/ROS_(операционная_система)

[10] MATLAB, wikipedia – Электронный ресурс https://ru.wikipedia.org/wiki/MATLAB

[11] CoppeliaSim, wikipedia – Электронный ресурс https://en.m.wikipedia.org/wiki/CoppeliaSim

[12] Raspberry Pi, wikipedia – Электронный ресурс https://ru.wikipedia.org/wiki/Raspberry_Pi

[13] Arduino, wikipedia – Электронный ресурс https://ru.m.wikipedia.org/wiki/Arduino

[14] PID – регулятор, wikipedia – Электронный ресурс https://ru.m.wikipedia.org/wiki/ПИД-регулятор

[15] Цикл (Теория графов), wikipedia – Электронный ресурс https://ru.m.wikipedia.org/wiki/Цикл_(теория_графов)

ПРИЛОЖЕНИЕ А

```
#include <Wire.h>

#define MOTOR1_PLUS 11
#define MOTOR1_MINUS 10
#define MOTOR2_PLUS 9
#define MOTOR2_MINUS 6
#define ECHO1 A0
#define TRIG1 A1
#define ECHO2 4
#define TRIG2 5
#define ECHO3 A2
#define TRIG3 A3
#define BATTERY A5
#define LED 13
#define SPRAYER 12
#define BUZZER 6


const int MPU = 0x68;
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY;
float gyroAngleX, gyroAngleY, gyroAngleZ;
int roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;

const float alpha = 0.98;

bool time_count = false;
bool do_it = false;
int fault = 10;

int pwm_for_motors = 128;
int motor1Speed, motor2Speed;

int interval1 = 3920;
uint32_t tmr1 = 0;
uint32_t lastTime = 0;

float dist1 = 0;
```

```
float dist2 = 0;
float dist3 = 0;

float targetDistance = 20.0;
float distanceTolerance = 2.0;

int targetYaw = 0;
int angle = 0;

float deltaTime = 0;

const float Kp = 2.0;
const float Ki = 0.5;
const float Kd = 1.0;

float previousError = 0;
float integral = 0;

uint32_t front = 0;
uint32_t front_last_millis = 0;
bool front_counting = false;

uint32_t back = 0;
uint32_t back_last_millis = 0;
bool back_counting = false;

uint32_t left = 0;
uint32_t left_last_millis = 0;
bool left_counting = false;

uint32_t right = 0;
uint32_t right_last_millis = 0;
bool right_counting = false;

const float R1 = 10000.0;
const float R2 = 10000.0;
const float referenceVoltage = 5.0;
const float voltageThreshold = 5.5;
float batteryVoltage = 0;

const uint32_t intervalOn = 500;
const uint32_t intervalOff = 2000;
```

```
uint32_t previousMillis = 0;
bool piezoState = LOW;

const float sampleTime = 0.1;
uint32_t lastPIDTime = 0;

const int threshold = 500;
bool waterPresent = false;

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);
  calculateError();
  delay(20);
  currentTime = micros();
  pinMode(MOTOR1_PLUS, OUTPUT);
  pinMode(MOTOR1_MINUS, OUTPUT);
  pinMode(MOTOR2_PLUS, OUTPUT);
  pinMode(MOTOR2_MINUS, OUTPUT);
  pinMode(TRIG1, OUTPUT);
  pinMode(ECHO1, INPUT);
  pinMode(TRIG2, OUTPUT);
  pinMode(ECHO2, INPUT);
  pinMode(TRIG3, OUTPUT);
  pinMode(ECHO3, INPUT);
  pinMode(BATTERY, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(SPRAYER, OUTPUT);
  pinMode(BUZZER, OUTPUT);
}

void loop() {

  getAngles();

  if (roll >= 25 || roll <= -25   ||   pitch >= 25 || pitch <= -25){
    getBuzzer();
  } else {
    digitalWrite(BUZZER, LOW);
```

```
  }

  uint32_t currentTime1 = millis();

  if (currentTime1 - lastPIDTime >= sampleTime * 1000) {
    deltaTime = (currentTime - lastPIDTime) / 1000.0;
    lastPIDTime = currentTime;

    dist1 = getDist(TRIG1, ECHO1);
    dist2 = getDist(TRIG2, ECHO2);
    dist3 = getDist(TRIG3, ECHO3);
  }

  if (front > 0 && back > 0 && right > 0 && left > 0 && front - back <= fault && left
- right <= fault) {
      front = 0;
      right = 0;
      back = 0;
      left = 0;
      do_it = true;
      return;
  }

  if (!do_it) {
    getDisinfect();
    if ((dist1 >= 18 || dist1 <= 22) && dist2 > 20) {
      getPID(angle, dist1, deltaTime);
      moveForward(motor1Speed, motor2Speed);
      getCount(yaw);
    } else if ((dist1 >= 18 || dist1 <= 22) && dist2 <= 20) {
      stopCount();
      angle = angle + 90;
      stopMotors();
      while (yaw < (targetYaw + angle)) {
        getAngles();;
        turnRight();
      }
      stopMotors();
    } else if (dist1 > 40 && dist2 > 20) {
      stopCount();
      tmr1 = millis();
      while (millis() - tmr1 < interval1){
```

```
      getPID(angle, dist1, deltaTime);
      moveForward(motor1Speed, motor2Speed);
    }
    angle = angle - 90;
    stopMotors();
    while (yaw > (targetYaw + angle)) {
      getAngles();;
      turnLeft();
    }
    stopMotors();
  }

  if ((dist3 >= 18 || dist3 <= 22) && dist2 > 20) {
    getPID(angle, dist3, deltaTime);
    moveForward(motor1Speed, motor2Speed);
    getCount(yaw);
  } else if ((dist3 >= 18 || dist3 <= 22) && dist2 <= 20) {
    stopCount();
    angle = angle + 90;
    stopMotors();
    while (yaw < (targetYaw + angle)) {
      getAngles();
      turnRight();
    }
    stopMotors();
  } else if (dist3 > 40 && dist2 > 20) {
    stopCount();
    tmr1 = millis();
    while (millis() - tmr1 < interval1){
      getPID(angle, dist3, deltaTime);
      moveForward(motor1Speed, motor2Speed);
    }
    angle = angle - 90;
    stopMotors();
    while (yaw > (targetYaw + angle)) {
      getAngles();
      turnLeft();
    }
    stopMotors();
  }
}
if (do_it){
```

```
    stopDisinfect();
  }
}

void calculateError() {
  c = 0;
  while (c < 200) {
    readAcceleration();
    AccErrorX += (atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 / PI);
    AccErrorY += (atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 /
PI);
    c++;
  }
  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  c = 0;
  while (c < 200) {
    readGyro();
    GyroErrorX += GyroX;
    GyroErrorY += GyroY;
    GyroErrorZ += GyroZ;
    c++;
  }
  GyroErrorX = GyroErrorX / 200;
  GyroErrorY = GyroErrorY / 200;
  GyroErrorZ = GyroErrorZ / 200;
}

void readAcceleration() {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
}

void readGyro() {
  Wire.beginTransmission(MPU);
  Wire.write(0x43); // Начинаем с регистра 0x43 (GYRO_XOUT_H)
  Wire.endTransmission(false);
```

```
  Wire.requestFrom(MPU, 6, true);
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
}

void getBuzzer() {
  unsigned long currentMillis = millis();

  if (piezoState == LOW) {
    if (currentMillis - previousMillis >= intervalOff) {
      previousMillis = currentMillis;
      piezoState = HIGH;
      digitalWrite(BUZZER, piezoState);
    }
  } else {
    if (currentMillis - previousMillis >= intervalOn) {
      previousMillis = currentMillis;
      piezoState = LOW;
      digitalWrite(BUZZER, piezoState);
    }
  }
}

float getDist(int trig_pin, int echo_pin) {
  digitalWrite(trig_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_pin, LOW);
  uint32_t duration = pulseIn(echo_pin, HIGH);
  return duration / 58.3;
}

void moveForward(int speed1, int speed2) {
  speed1 = constrain(speed1, 0, 255);
  speed2 = constrain(speed2, 0, 255);
  analogWrite(MOTOR1_PLUS, speed1);
  analogWrite(MOTOR1_MINUS, 0);
  analogWrite(MOTOR2_PLUS, 0);
  analogWrite(MOTOR2_MINUS, speed2);
}

void turnLeft() {
```

```
  analogWrite(MOTOR1_PLUS, pwm_for_motors);
  analogWrite(MOTOR1_MINUS, 0);
  analogWrite(MOTOR2_PLUS, pwm_for_motors);
  analogWrite(MOTOR2_MINUS, 0);
}

void turnRight() {
  analogWrite(MOTOR1_PLUS, 0);
  analogWrite(MOTOR1_MINUS, pwm_for_motors);
  analogWrite(MOTOR2_PLUS, 0);
  analogWrite(MOTOR2_MINUS, pwm_for_motors);
}

void stopMotors() {
  analogWrite(MOTOR1_PLUS, 0);
  analogWrite(MOTOR1_MINUS, 0);
  analogWrite(MOTOR2_PLUS, 0);
  analogWrite(MOTOR2_MINUS, 0);
}

void getPID(float angle, float distn, float deltaTime){
  float yawError = (targetYaw + angle) - yaw;
  float distError = targetDistance - distn;
  float combinedError = yawError + distError;

  integral += combinedError * deltaTime;
  float derivative = (combinedError - previousError) / deltaTime;
  float correction = Kp * combinedError + Ki * integral + Kd * derivative;
  previousError = combinedError;

  motor1Speed = pwm_for_motors + correction;
  motor2Speed = pwm_for_motors - correction;
}

void getTime(uint32_t& side, uint32_t& side_last_millis, bool& side_counting) {
  if (!side_counting) {
    side_counting = true;
    side_last_millis = millis();
  }
  uint32_t currentMillis = millis();
  side += (currentMillis - side_last_millis);
  side_last_millis = currentMillis;
```

```
}

void stopTime(uint32_t& side_last_millis, bool& side_counting) {
  if (side_counting) {
    side_counting = false;
    side_last_millis = millis();
  }
}

void getCount(float current_angle){
  int normalizedAngle = static_cast<int>(angle) % 360;
  if (normalizedAngle < 0) {
    normalizedAngle += 360;
  }

  if ((normalizedAngle >= 350 && normalizedAngle <= 360) || (normalizedAngle >=
0 && normalizedAngle <= 10)) {
    getTime(front, front_last_millis, front_counting);
  } else if (normalizedAngle >= 80 && normalizedAngle <= 100) {
    getTime(left, left_last_millis, left_counting);
  } else if (normalizedAngle >= 170 && normalizedAngle <= 190) {
    getTime(back, back_last_millis, back_counting);
  } else if (normalizedAngle >= 260 && normalizedAngle <= 280) {
    getTime(right, right_last_millis, right_counting);
  }
}

void stopCount(){
  stopTime(front_last_millis, front_counting);
  stopTime(back_last_millis, back_counting);
  stopTime(left_last_millis, left_counting);
  stopTime(right_last_millis, right_counting);
}

void checkBattery(){
  int sensorValue = analogRead(BATTERY);
  float voltage = sensorValue * (referenceVoltage / 1023.0);
  batteryVoltage = voltage * ((R1 + R2) / R2);
}

void getDisinfect() {
  digitalWrite(LED, HIGH);
```

```
  if (waterPresent){
    digitalWrite(SPRAYER, HIGH);
  }
  else {
    digitalWrite(SPRAYER, LOW);
  }
}

void stopDisinfect() {
  digitalWrite(LED, LOW);
  digitalWrite(SPRAYER, LOW);
}


void getAngles(){
  readAcceleration();
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) -
AccErrorX;
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) -
AccErrorY;
  previousTime = currentTime;
  currentTime = micros();
  elapsedTime = (currentTime - previousTime);
  readGyro();
  GyroX -= GyroErrorX;
  GyroY -= GyroErrorY;
  GyroZ -= GyroErrorZ;
  gyroAngleX += GyroX * elapsedTime;
  gyroAngleY += GyroY * elapsedTime;
  gyroAngleZ += GyroZ * elapsedTime;
  roll = 0.96*gyroAngleX + 0.4*accAngleX;
  pitch = 0.96*gyroAngleY + 0.4*accAngleY;
  yaw += GyroZ * elapsedTime;
}
```

НЕКОММЕРЧЕСКОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО
«КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ К.И. САТПАЕВА»

ОТЗЫВ

дипломного проекта (работы)

студента специальность 6B07113 – «Робототехника и мехатроника»

Уразаева Александра Александровича

На тему: **«Разработка программной части робота-дезинфектора»**

В своей дипломной работе Уразаев А.А. успешно разработал программную часть автономного робота-дезинфектора, предназначенного для дезинфекции стен помещений. Основной идеей дипломной работы является создание алгоритма действий, выполняемых роботом, в зависимости от внешних факторов. Разработанная программа полностью соответствует поставленным требованиям.

В процессе разработки программной части студент рассмотрел существующих роботов-дезинфекторов, различные методы разработки и управления роботизированными системами, а также исследовал различные программные решения для упрощения работы с платформой Arduino.

Студент разработал: схему движения робота, метод обработки данных с датчиков, а также функцию коррекции движения. Алгоритм прошел тестирование, показав стабильную работу и эффективность в выполнении поставленных задач.

Работа хорошо структурирована, грамотно оформлена и содержит все необходимые расчеты и рисунки. Дипломная работа Уразаева А.А. свидетельствует о его готовности к профессиональной деятельности и заслуживает высокой оценки.

Научный руководитель
Магистр технических наук, старший преподаватель
_____Баянбай Н.А.

«___» ___06___ 2024 г.

Ф КазНИТУ 704-21. Отзыв научного руководителя

НЕКОММЕРЧЕСКОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени К.И.САТПАЕВА»

РЕЦЕНЗИЯ

дипломного проекта Уразаева Александра Александровича
по специальности 6В07113 – «Робототехника и мехатроника»
Satbayev University

На тему: «Разработка программной части робота-дезинфектора»

Выполнено:
а) графическая часть на 27 листах
б) пояснительная записка на 46 страницах

ЗАМЕЧАНИЯ К РАБОТЕ

Данная дипломная работа освещает методы разработки программного обеспечения для автономных роботизированных систем на примере робота-дезинфектора. Студент осуществил глубокий анализ практической и теоретической информации, касающейся данной темы, демонстрируя высокий уровень знаний в области робототехники.

В работе автором изучены современные программы для создания управляющего алгоритма роботизированных систем. Также приложены принципиальная схема робота и математический анализ данных с датчиков.

Особое внимание уделено безопасности использования робота, предусмотрены различные сценарии выполнения его основных функций, включая завершение дезинфицирующего раствора, заряда аккумулятора и выведение робота из равновесия.

Эта работа значима как научное исследование и как практический вклад в разработку программной части различных роботизированных систем. Подробный анализ разработанного алгоритма позволяет брать данную работу за основу в дальнейшем конструировании роботов

ОЦЕНКА РАБОТЫ

Дипломная работа соответствует всем требованиям, предъявляемым к дипломным работам, и представляет собой ценный вклад в развитие робототехники в разработке программной части робототехники. Уразаева А.А. заслуживает высокой оценки 94% за качественное исследование и практическую применимость результатов.

Рецензент
Кандидат технических наук, доцент
Кафедры «Физика» КазНПУ им.Абая
_____ Жаменкеев Е.К
(подпись)
«28» мая 2024 г.

StrikePlagiarism.com

SATBAYEV UNIVERSITY

Дата отчета 5/29/2024

Дата редактирования ---

Отчет не был оценен.

Метаданные

Название

Разработка программной части робота-дезинфектора

Автор

Уразаев Александр Александрович

Научный руководитель / Эксперт

Нурлан Баянбай

Подразделение

ИАиИТ

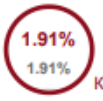
Тревога

В этом разделе вы найдете информацию, касающуюся текстовых искажений. Эти искажения в тексте могут говорить о ВОЗМОЖНЫХ манипуляциях в тексте. Искажения в тексте могут носить преднамеренный характер, но чаще, характер технических ошибок при конвертации документа и его сохранении, поэтому мы рекомендуем вам подходить к анализу этого модуля со всей долей ответственности. В случае возникновения вопросов, просим обращаться в нашу службу поддержки.

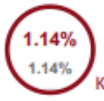
Замена букв		0
Интервалы		0
Микропробелы		0
Белые знаки		0
Парафразы (SmartMarks)	a	3

Объем найденных подобий

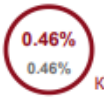
КП-ия определяют, какой процент текста по отношению к общему объему текста был найден в различных источниках.. Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.



КП1



КП2



КЦ

25
Длина фразы для коэффициента подобия 2

4127
Количество слов

32174
Количество символов

Поиск контента ИИ

Интегрированный модуль поиска контента AI. Нажмите «Подробнее», чтобы узнать больше о результатах и алгоритме поиска.

Коэффициент вероятности ИИ

Подобия по списку источников

Ниже представлен список источников. В этом списке представлены источники из различных баз данных. Цвет текста означает в каком источнике он был найден. Эти источники и значения Коэффициента Подобия не отражают прямого плагиата. Необходимо открыть каждый источник и проанализировать сожержание и правильность оформления источника.

10 самых длинных фраз

Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

1	РАЗРАБОТКА АТОМАТИЗИРОВАННОГО РОБОТА-ДЕЗИНФЕКТОРА.docx [redacted] 4/3/2024 Satbayev University (ИПАиЦ)	47	1.14 %
2	РАЗРАБОТКА АТОМАТИЗИРОВАННОГО РОБОТА-ДЕЗИНФЕКТОРА.docx [redacted] 4/3/2024 Satbayev University (ИПАиЦ)	12	0.29 %
3	Разработка интеллектуальной системы автоматического управления процессами газовой области [redacted] 5/29/2023 Satbayev University (ИАиИТ)	11	0.27 %
4	https://stud.kz/referat/show/78611	9	0.22 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (1.70 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	РАЗРАБОТКА АТОМАТИЗИРОВАННОГО РОБОТА-ДЕЗИНФЕКТОРА.docx [redacted] 4/3/2024 **Satbayev University (ИПАиЦ)**	**59 (2)**	**1.43 %**
2	Разработка интеллектуальной системы автоматического управления процессами газовой области [redacted] 5/29/2023 Satbayev University (ИАиИТ)	11 (1)	0.27 %

из программы обмена базами данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из интернета (0.22 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://stud.kz/referat/show/78611	9 (1)	0.22 %

Список принятых фрагментов (нет принятых фрагментов)

ПОРЯДКОВЫЙ НОМЕР	СОДЕРЖАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)