**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»**

**ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите**  Заместитель директора по  учебно-методической работе  \_\_\_\_\_\_\_\_\_\_\_ \_\_Е. Г. Конакина\_\_\_\_\_  (Подпись) (И.О.Ф.)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка программно-аппаратного робототехнического комплекса для поиска утечки опасных газов в замкнутом пространстве

специальность 09.02.07 группа 42919/4

Студент (ка) Эфендиев В.К.

(подпись) (ФИО)

Руководитель Болсуновская М.В.

(подпись) (ФИО)

Санкт-Петербург

2023

# СОДЕРЖАНИЕ

[Введение 4](#_Toc136385227)

[1 Общая часть 7](#_Toc136385228)

[1.1 Анализ предприятия 7](#_Toc136385229)

[1.2 Постановка задачи 9](#_Toc136385230)

[1.3 Анализ рынка существующих решений 11](#_Toc136385234)

[2 Техническая часть и программирование РОБОГАЗ 18](#_Toc136385235)

[2.1 Теоретическая часть 18](#_Toc136385236)

[2.2 Практическая часть 23](#_Toc136385237)

[2.3 Тестирование РОБОГАЗ 46](#_Toc136385253)

[3 Экономическая часть 49](#_Toc136385254)

[3.1 Технико-экономическое обоснование 49](#_Toc136385255)

[3.2 Расчет экономической эффективности от внедрения программного продукта 50](#_Toc136385256)

[3.3 Расчет капитальных затрат на проектирование и внедрение 51](#_Toc136385257)

[3.4 Расчет эксплуатационных расходов 54](#_Toc136385258)

[4 Техника безопасности и охрана труда 60](#_Toc136385262)

[4.1 Анализ условий труда программиста 60](#_Toc136385263)

[4.2 Расчет искусственного освещения в помещении 62](#_Toc136385264)

[4.3 Электробезопасность на предприятии 64](#_Toc136385265)

[4.4 Пожарная безопасность на предприятии 66](#_Toc136385266)

[Заключение 68](#_Toc136385267)

[Список использованных источников 70](#_Toc136385268)

[Приложение А Листинг кода программы arduino 72](#_Toc136385269)

[Приложение Б Листинг кода программы pysensors\_ data 80](#_Toc136385270)

# ВВЕДЕНИЕ

В промышленности используются различные вещества, в том числе и газы, что может привести к ситуациям, угрожающим здоровью людей. В данной работе представлены результаты разработки робототехнического комплекса на основе использования интеллектуальных систем для эффективного и точного выявления мест утечки газа.

Концепция разработки программно-аппаратного робототехнического комплекса для поиска утечки опасных газов в замкнутом пространстве является актуальной и востребованной среди производственных предприятий в сфере промышленной безопасности и спасательных мероприятий. В современных условиях необходимость быстрого и эффективного обнаружения утечек газов и своевременного принятия мер по их локализации становится все более актуальной, так как это напрямую связано с сохранением жизни и здоровья людей. В данной дипломной работе будет рассмотрена разработка робототехнического комплекса, который способен автоматически обнаруживать и локализовывать местоположение утечки опасных газов в замкнутых пространствах с помощью использования современных технологий и методов.

Конечный продукт работы – программно-аппаратный робототехнический комплекс для поиска и локализации утечки опасных газов в замкнутом пространстве (РОБОГАЗ). Данная разработка предназначена для использования в учебных целях, в спасательных мероприятиях, а также промышленности, где регулярно происходят аварии с выбросом вредных веществ. Он позволяет ускорить процесс обнаружения утечки, а также уменьшить риск для здоровья людей, дав возможность заменить их на РОБОГАЗ.

Для разработки данного комплекса необходимо было провести исследование и анализ существующих решений в данной области, а также выбрать наиболее подходящие для нашего проекта компоненты и технологии.

В процессе работы были использованы современные инструменты и технологии программирования и инженерии, такие как 3D-моделирование, симуляция и программирование микроконтроллеров.

В данной работе детально рассматривался каждый этап разработки комплекса: от выбора аппаратной платформы и компонентов до разработки программного обеспечения и тестирования комплекса. Также были проведены эксперименты для оценки эффективности и точности работы робототехнического комплекса.

Итак, данная работа направлена на создание нового решения в области обнаружения утечки опасных газов, которое будет эффективным и безопасным для людей, иметь небольшую стоймость разработки, а также автономность управления [2]. Разработка является важным шагом в обеспечении безопасности и охраны окружающей среды.

Задачи исследования:

1. Изучение существующих методов поиска утечек газов и сравнение их с предлагаемым методом, на основе которого будет разработан робототехнический комплекс;
2. Анализ опасных газов, которые могут быть обнаружены с помощью данного комплекса, и определение их химических свойств и особенностей поведения в замкнутом пространстве;
3. Разработка алгоритмов работы комплекса для поиска утечек, включая выбор и оптимизацию сенсоров, методов сбора и обработки данных, а также варианта движения робота;
4. Тестирование и оценка эффективности разработанного комплекса на различных типах замкнутых пространств и с разными уровнями утечек газов;
5. Разработка методов движения робота в помещении и поддержки робототехнического комплекса, включая создание автономного управления и анализа полученных данных;
6. Исследование экономической и социальной эффективности использования данного комплекса для поиска утечек опасных газов в различных областях применения, таких как промышленность, медицина и др.

# 1 Общая часть

# 1.1 Анализ предприятия

Лаборатория с 2004 года занимается наукоемкими исследованиями и коммерческими разработками в области алгоритмов и систем сбора и анализа данных. В 2017 была сформирована межинститутская научно-исследовательская лаборатория «Алгоритмы и системы потоковой обработки данных» (МНИЛ «АСПОД») в структуре Объединенного научно-технологического института. С 2018 года МНИЛ «АСПОД» включена в структуру Центра НТИ СПбПУ с названием «Промышленные системы потоковой обработки данных» (ПСПОД) Центра НТИ СПбПУ.

Результаты исследований применяются при создании аналитических информационных систем для компаний, заинтересованных в цифровизации бизнеса и стремящихся использовать передовые технологии в промышленности, сельском хозяйстве, строительстве, медицине, сфере IT, ЖКХ и др.

Ключевыми областями экспертизы являются:

1. Аналитика данных (анализ большого объёма разнородных данных для выявления в них закономерностей и получения информации, необходимой для принятия решений);
2. Системы технического зрения (анализ цифрового изображения с видеокамер для повышения точности и эффективности работы оборудования и процессов);
3. Системы видеоаналитики (разработка алгоритмов и ПО для анализа цифрового изображения в сфере видеонаблюдения и безопасности: распознавания, детектирования, визуализации объектов или сцен, поиска изображений);
4. Математическое моделирование и проектирование;
5. Машинное обучение (методы сбора и обработки данных на основе математических моделей и алгоритмов, которые позволяют строить прогнозную аналитику на основе исторических данных о системе);
6. Автономный транспорт (осуществление управления без водителя с помощью различных датчиков (напр. камер, лидаров) и компьютерных алгоритмов (рисунок 1));
7. Геоинформационные системы (обработка, анализ и визуализация пространственно-координатных данных для повышения эффективности процессов управления, хранения и представления информации, её обработки и анализа с целью поддержки принятия решений).
8. BIM-проектирование (Building Information Modeling – сбор и комплексная обработка всей архитектурно-конструкторской, технологической, экономической и другой информации о здании со всеми её взаимосвязями и зависимостями, а также совокупность мероприятий по управлению жизненным циклом объекта).

# 1.2 Постановка задачи

Первоначальная задача состояла в создании РОБОГАЗ как финального задания для учеников, участвующих в НТО (Национальная Техническая Олимпиада). В последующем данный инструмент может использоваться для обучения студентов робототехнике, а также поиску и локализации местоположения утечек в различных ситуациях.

Для этого требуется решить следующие задачи:

1. Разработать аппаратную часть комплекса, включающую в себя датчики газов, устройства для сбора и передачи данных, а также элементы управления;
2. Создать программное обеспечение для сбора, обработки и анализа данных, получаемых от датчиков, с целью определения наличия и местонахождения утечки;
3. Разработать алгоритм управления роботом, который будет перемещаться по замкнутому пространству для обнаружения утечки;
4. Провести тестирование комплекса на специально подготовленной площадке для определения эффективности работы и точности распознавания местонахождения утечки;
5. Провести экспериментальные исследования на различных объектах для оценки работоспособности комплекса в различных условиях;
6. Оптимизировать работу комплекса с помощью анализа результатов исследования и внесения соответствующих корректировок в аппаратную и программную части комплекса.

После постановки задач были составлены функциональные и технические требования.

К функциональным требованиям относятся:

1. Комплекс должен обладать способностью обнаруживать наличие опасных газов в замкнутом пространстве и определять местоположение утечки с точностью до 1 метра;
2. Комплекс должен иметь возможность анализировать данные об обнаруженной утечке, концентрации газа, размере зоны поражения и т.д.;
3. Комплекс должен обеспечивать управление роботом, который будет осуществлять поиск утечки и сбор данных. Управление должно осуществляться с помощью программного интерфейса, который позволит оператору контролировать движение робота и получать данные в режиме реального времени;
4. Робот должен быть оснащен системой автоматического выключения в случае опасной ситуации или сбоя системы управления;
5. Комплекс должен обеспечивать удобный и понятный интерфейс для пользователей, который позволит им получать данные об обнаруженной утечке и управлять роботом;
6. Комплекс должен быть способен интегрироваться с другими системами, например, системами автоматической пожарной сигнализации или системами безопасности зданий.

Технические требования:

1. Операционная система: Linux;
2. Дистрибутив: Ubuntu 20.04;
3. Верхний уровень: микрокомпьютер Raspberry Pi 3 либо выше, или его аналог Orange Pi 3 либо выше;
4. Процессор: SoC Allwinner H6 или аналог, или лучше;
5. ОЗУ: 2 ГБ или больше;
6. Свободное пространство на диске: 64 ГБ;
7. Нижний уровень: Arduino Mega 2560;
8. Датчки: Датчик газа MQ-6, датчики дистанции VL6180X;
9. Лазерный сканер: RPLidar A1.

# 1.3 Анализ рынка существующих решений

Для описанного выше комплекса существует несколько аналогов на рынке. Рассмотрим некоторые из них. Также плюсы и минусы всех упомянутых комплексов можно рассмотреть в таблице 1.

Одним из наиболее известных и широко используемых решений является «GasFindIR» (рисунок 1) компании FLIR Systems. Эта система включает в себя инфракрасную камеру, которая может обнаруживать утечки метана и других газов. Также существует ряд других устройств, например, «ToxiRAE Pro» (рисунок 2) и «MultiRAE Lite» (рисунок 3), которые используются для обнаружения различных типов газов.



Рисунок 1 – Использование GasFindIR при поиске утечек



рисунок 2 – Использование ToxiRAE Pro на производстве



Рисунок 3 – Устройство MultiRAE Lite

Однако существующие на рынке решения имеют ряд ограничений. Например, многие из них могут работать только при определенных условиях, таких как освещенность или температура окружающей среды. Также некоторые из них могут быть очень дорогими, или для их использования необходимы специализированные знания.

Таким образом, существует потребность в разработке нового решения, которое будет более универсальным и доступным для использования, в том числе в сложных условиях. Предлагаемый комплекс, сочетающий в себе аппаратные и программные средства, может решить эту проблему и обеспечить эффективный и точный поиск утечек газов в замкнутых пространствах.

Анализ рынка существующих решений для комплекса по поиску утечек опасных газов в замкнутом пространстве позволяет выявить несколько основных типов аналогов, которые уже существуют на рынке и применяются в различных областях.

Одним из таких аналогов являются портативные газоанализаторы, которые могут использоваться для обнаружения утечек газов в замкнутых помещениях. Эти устройства могут иметь различные функции, например, осуществлять измерение концентрации газа, сигнализировать о превышении установленных пороговых значений, а также могут быть оснащены системой логирования данных. Примером такого газоанализатора является Dräger X-am 2500 (рисунок 4), который может обнаруживать до четырех газов одновременно и имеет встроенную систему логирования данных.



Рисунок 4 – устройство Dräger X-am 2500 и его комплектация

Еще одним аналогом может быть система контроля окружающей среды, которая может использоваться для мониторинга концентрации газов в замкнутых помещениях. Такие системы могут быть оснащены различными датчиками, например, датчиками CO2, датчиками температуры и влажности, а также датчиками концентрации определенных газов. Примером такой системы может быть Netronix WMC-100, которая предназначена для мониторинга качества воздуха внутри помещений. К сожалению, изображения данного устройства отсутствуют в свободном доступе.

Также на рынке существуют специализированные системы для обнаружения утечек газов, которые могут использоваться в промышленных условиях. Эти системы могут быть оснащены различными датчиками, например, датчиками метана, датчиками хлора, датчиками сероводорода и т.д. Примером такой системы может быть Honeywell Analytics Sensepoint XCD (рисунок 5), которая может обнаруживать различные типы газов и имеет встроенную систему оповещения в случае обнаружения утечки газа.



Рисунок 5 – Система Honeywell Analytics Sensepoint XCD

Таким образом, анализ рынка существующих решений позволяет выявить, что существуют различные типы устройств и систем для обнаружения утечек опасных газов в замкнутых помещениях. Тем не менее, каждое из этих решений имеет свои преимущества и недостатки.

Таблица 1. - Плюсы и минусы комплексов

|  | GasFindIR | ToxiRAE Pro | MultiRAE Lite | Dräger X-am 2500 | Netronix WMC-100 | Honeywell Analytics Sensepoint XCD |
| --- | --- | --- | --- | --- | --- | --- |
| Широкий диапазон обнаруживаемых газов | + | + | + | + | + | + |
| Высокая точность измерений | + | + | + | + | + | + |
| Высокая чувствительность к газам | + | + | + | + | + | + |
| Гибкая настройка и программируемость | - | - | - | - | + | - |
| Удобный и понятный интерфейс | + | + | + | + | + | + |
| Сложность первоначальной настройки | + | + | + | - | - | - |
| Возможные ограничения в работе в условиях высокой влажности | + | + | + | - | - | - |

Рассмотрев данную таблицу, можно увидеть лишь устройства и системы для поиска утечек газа под названием «Газоанализатор», что для олимпиады и нашей задачи совершенно не подходит. Поэтому было принято решение о создании комплекса, который сможет выполнять ту же задачу, но только автономно, без участия человека.

Газоанализаторы могут подойти для устранения утечек газа в малых количествах, однако при более серьезных утечках человек может пострадать. Сравнение комплекса РОБОГАЗ, разработанного для выпускной квалификационной работы, с системами и устройствами, указанными в таблице, позволяет выделить ряд преимуществ комплекса:

1. Низкая стоимость разработки;
2. Открытый исходный код;
3. Возможность развития и модификации;
4. Легкая первоначальная настройка;
5. Автономность;
6. Габариты робота.

# 2 Техническая часть и программирование РОБОГАЗ

# 2.1 Теоретическая часть

При разработке комплекса РОБОГАЗ использовалось 3 уровня управления:

1. Нижний уровень (подсистема управления) – данный уровень отвечает за непосредственное управление моторами для колес и датчиками робота. Здесь реализуются обработка сигналов от датчиков и выполнение низкоуровневых команд;
2. Верхний уровень – данный уровень объединяет в себе работу среднего и верхнего уровня. На данном уровне происходит обработка алгоритма движения, формулируются задачи и цели работы комплекса. В данной работе тесно связан с низким уровнем.

Для разработки и программирования нижнего уровня использовался микроконтроллер Adruino MEGA 2560 (рисунок 6). Это одна из самых мощных и распространенных платформ для создания электронных проектов и робототехнических систем. Обладает множеством цифровых и аналоговых входов и выходов, а также интерфейсов для подключения датчиков, моторов, LCD-экранов и других устройств. При выполнении задачи микроконтроллер используется для сбора данных с датчиков дистанции и газа, а также управления моторами и парсинга данных.

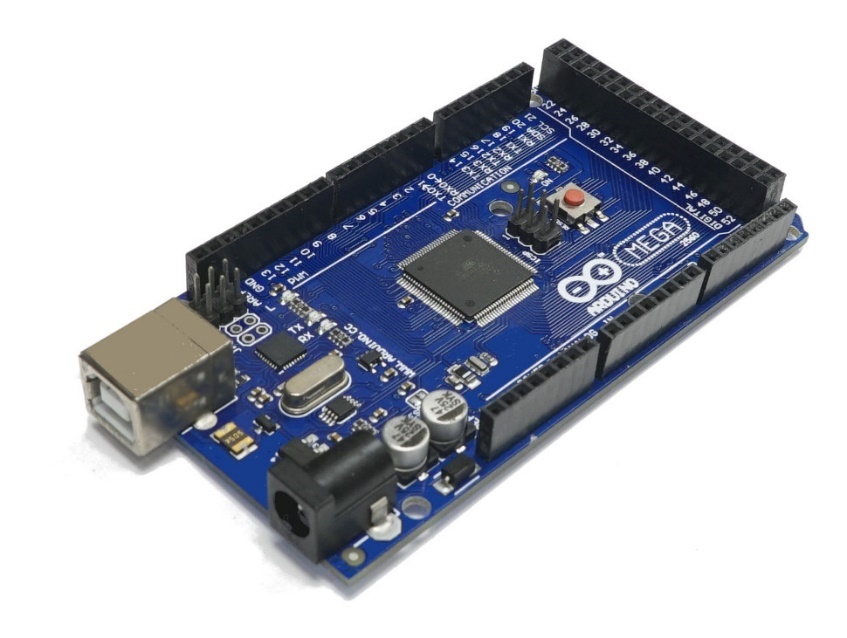


Рисунок 6 – Arduino MEGA 2560

К Arduino подключаются: 6 датчиков дистанции (VL6180X) и 1 датчик газа (MQ-6) для сбора информации и отправки данных на верхний уровень (эти устройства рассматриваются в пункте **2.2.5** и **2.2.6**), а также 2 мотора для управления движением робота [6, 7].

Для работы с верхним уровнем был выбран микрокомпьютер Orange Pi 3 LTS (рисунок 7). Это одноплатный компьютер (SBC), основанный на архитектуре ARM. Он предлагает процессор Allwinner H6, четыре ядра Cortex-A53 и графический процессор Mali-T720. Устройство также оснащено различными интерфейсами, включая HDMI, USB, Ethernet и Wi-Fi, что обеспечивает возможность подключения к различным периферийным устройствам и сетям. К Orange Pi 3 подключаются 2 устройства: RPLIDAR A1 (Рисунок 8) и Arduino соответственно. Подключение происходит при помощи серийного порта (данный способ рассматривается в пункте 2.2.4).

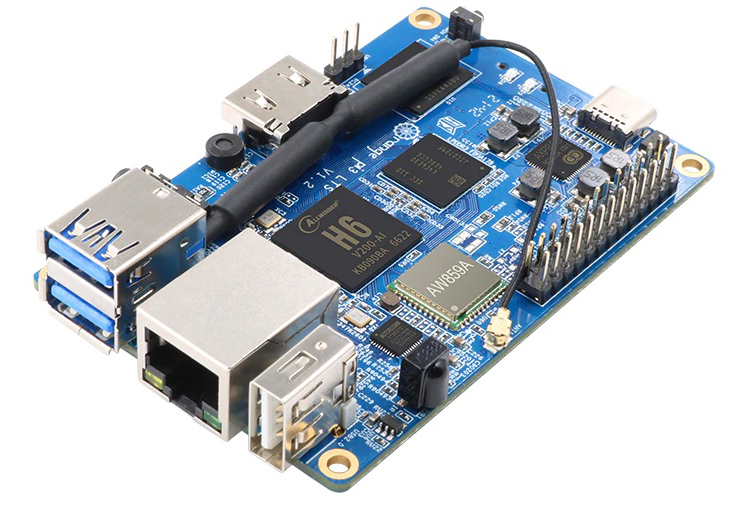


Рисунок 7 – Orange Pi 3 LTS



Рисунок 8 – 2D сканер помещения RPLIDAR A1

Программное обеспечение разрабатывалось под управлением операционной системы Linux с установленным дистрибутивом Ubuntu 20.04. Далее требуется установка ROS (Robot Operating System) [5, 10] – гибкой и распределенной системы для разработки программного обеспечения в области робототехники. Несмотря на наименование «операционная система», ROS на самом деле является не операционной системой в привычном понимании, а, скорее, фреймворком и набором инструментов для разработки робототехнических приложений.

При разработке ROS предоставляет множество инструментов, библиотек и протоколов, которые помогают упростить работу по созданию программного обеспечения для роботов. Также имеются средства для управления процессами, коммуникацией между компонентами системы, управления аппаратными устройствами робота, визуализацией данных и многого другого.

В основе системы ROS лежит концепция «узлов» («nodes»), которые представляют собой отдельные компоненты системы, применяемые для выполнения различных задач. Например, задачи, которые были выполнены в рамках выпускной квалификационной работы, затрагивали управление моторами робота, обработку данных с датчиков дистанции и газа, использование написанного алгоритма движения внутри помещения и получения информации, обработку данных с устройства RPLidar A1. Главная особенность узлов в том, что они обмениваются информацией между собой с помощью «topics» – каналов, которые предоставляются для передачи данных. Данная система позволяет разрабатывать узлы на нескольких поддерживаемых языках. Основными являются C++ [11, 12] и Python [1, 3, 4]. Сама система очень упрощает разработку робототехнических систем, обмен данными, а также интеграцию компонентов, что позволяет разработчику сосредоточиться на создании функциональности робота, а не на низкоуровневых деталях системы.

Для программирования верхнего и нижнего уровня использовалась среды разработки Arduino IDE [13] и Visual Studio Code от компании Microsoft[14]. В среду Visual Studio Code, были установлены дополнительные библиотеки для программирования, подключения датчиков к микроконтроллеру и получения данных с них:

1. Wire.h – данная библиотека используется для связи микроконтроллера с устройствами и модулями через интерфейс I2C. I2C это шина связи, использующая всего две линии. С помощью этого интерфейса Arduino может всего по двум проводам обмениваться данными со множеством устройств.;
2. VL6180X.h – данная библиотека упрощает настройку датчика дистанции, считывание с него данных и уровень внешней освещенности через шину I2C;
3. TroykaMQ.h – данная библиотека используется для считывания данных с датчиков газа MQ;
4. ros.h – это заголовочный файл, содержащий объявление сообщения типа std\_msgs/String, который находится в пакете std\_msgs. Файл генерируется автоматически из файла String.

# 2.2 Практическая часть

# 2.2.1 Структура программы для Arduino

Написание кода для Arduino выполняется на языке C++. Основная структура программ для Arduino состоит из двух функций: setup() и loop().

Первая отвечает за команды, которые будут производиться один раз при старте микроконтроллера и настраивать все подключённые устройства. Вторая отвечает за команды, которые будут выполняться в цикле работы микроконтроллера. Интервал итераций цикла команд в loop() можно регулировать функцией delay(), которая принимает один аргумент: тип unsigned long, представляющий из себя количество времени задержки в миллисекундах, который обязательно потребуется в будущем при настройке устройств в функции setup(). Все написание кода для Arduino происходило в Arduino IDE и Visual Studio Code. В конце разработки программного обеспечения для нижнего уровня имеется выходной артефакт под названием machine.ino. В нем описаны методы работы моторов, принятие данных с датчиков и их отправка на верхний уровень.

# 2.2.2 Занятость портов, их настройка и мониторинг данных с верхнего уровня

При подключении верхнего уровня необходимо запоминать порт, через который идёт взаимодействие с Arduino, поскольку при его смене придётся менять соответствующую настройку в среде разработки.

В случае, если настроенный порт занят каким-либо другим процессом или программой, Arduino перестанет отвечать на команды.

Чтобы убедиться, что данные с каких-либо датчиков корректно передаются, или для любых других проверок, в ходе работы будет полезен мониторинг порта благодаря встроенному набору функций Serial. Чтобы данные выводились и могли быть прочитаны, необходимо:

1. Наличие в коде Serial.begin(115200); – инициирует последовательное соединение и задает скорость передачи данных в бит/c (бод) (тип аргумента – long). При работе имеются допустимые значения: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 или 115200. Вызывается в setup();
2. Использование в коде одного или нескольких методов Serial.print(); Serial.println(); – выводит информацию, передаваемую в аргументы методов. Первый метод выполняет вывод информации без перевода на новую строку, второй позволяет произвести вывод информации с переходом на новую строку;
3. Подключив и загрузив код на Arduino, открыть монитор порта в зависимости от IDE, выставить значение «бод» идентичное инициированному в коде.

# Работа с Arduino MEGA 2560 в Arduino IDE

Для работы через Arduino IDE необходимо открыть файл machine.ino в программе, а также настроить среду через вкладку «Инструменты» («Tools»). «Плата» («Board»): Arduino Mega or Mega 2560; «Процессор» («Processor», устанавливается автоматически): ATmega2560 (Mega 2560); «Порт» («Port»): установите свой выбранный (к примеру, /dev/ttyUSB0 или COM4); «Программатор» («Programmer»): AVRISP mkII или AVR ISP.

Для проверки кода на ошибки можно воспользоваться кнопкой «Проверить» («Verify»). Для загрузки готового кода на микроконтроллер используется кнопка «Загрузить» («Upload»). Они находятся в левой верхней части окна (рисунок 9). В правом верхнем углу имеется кнопка для открытия «Монитора порта» («Serial Monitor»), что может пригодиться при отладке [7].

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Проверка и загрузка кода в Arduino IDE

# 2.2.4 Работа с Arduino MEGA 2560 в IDE Visual Studio Code

При открытии папки с проектом и файла формата .ino в Visual Code в нижней правой части окна появляется синяя линия с 3-мя базовыми настройками для Arduino (рисунок 10) выбор модели микроконтроллера, программатора и серийного порта.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Выбор конфигурации для работы с Arduino

Сначала выбираем тип микроконтроллера – Arduino Mega или Mega 2560 (Arduino AVR Boards). После этого Visual Code создаст ещё одно поле с процессором, которое заполняется автоматически (должно быть ATmega2560 (Mega 2560)) (рисунок 11).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11 – Итоговый вариант настройки типа Arduino

В качестве программатора был выбран «AVR ISP».

Перед выбором серийного порта подключаем к нему Arduino, иначе Visual Code может его не увидеть.

Если в ходе написания программы возникает необходимость воспользоваться инструментами отладки для Arduino (например, открыть и посмотреть серийный порт), можно применить комбинацию клавиш Ctrl+Shift+P: откроется строка поиска, в которой все возможности расширения микроконтроллера помечены «Arduino:» (рисунок 12).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 12 – Инструменты расширения для Arduino в Visual Studio Code

Для проверки кода на ошибки можно воспользоваться кнопкой «Verify» (комбинация клавиш: Ctrl+Alt+R). Для загрузки готового кода на микроконтроллер используется кнопка «Upload» (комбинация клавиш: Ctrl+Alt+U). Они находятся в правой верхней части окна (рисунок 13).

Изображение выглядит как текст, снимок экрана, Измерительный инструмент, Шрифт

Автоматически созданное описание

Рисунок 13 – Загрузка и проверка кода в Visual Studio Code

# 2.2.5 Работа с датчиками дистанции VL6180X

Начиная работать с данными датчиками, необходимо подключить их к плате Arduino. Подключаемые элементы можно разделить на 3 типа: питание, управление и передача информации (рисунок 14).

Изображение выглядит как Электронная техника, электроника, Электронное устройство, схема

Автоматически созданное описание

Рисунок 14 – Датчик дистанции и освещенности VL6180X

Характеристики датчика VL6180X:

1. Инфракрасный излучатель;
2. Длина волны излучателя: 850нм;
3. Измеряемое расстояние: 1-100 мм;
4. Тип интерфейса: I2C.

За питание отвечают порты VIN и GND. Порт VIN подключается к 5V, а GND – к GND; оба находятся в секции «POWER» на плате микроконтроллера (рисунок 15).

Изображение выглядит как текст, электроника, схема, Электронная техника

Автоматически созданное описание

Рисунок 15 – Питание устройств на Arduino Mega 2560

Передача данных от датчиков происходит по выводным портам SCL и SDA (шина I2C). Они подключаются к соответствующим пинам на Arduino в секции «COMMUNCIATION» (рисунок 16).

Изображение выглядит как схема, текст, Электронная техника, электроника

Автоматически созданное описание

Рисунок 16 – Секции передачи информации и управления устройствами

Для управления датчиками, включения, инициализации, раздачи адреса необходимо подключить SHDN (контакт SHDN предназначен для перевода регулятора напряжения в режим энергосбережения; это происходит при подаче на данный контакт низкого логического напряжения) на датчике к одному из разъемов на Arduino в секции «PWM» (рисунок 16).

Входы и выходы на Arduino сильно ограничены по количеству, не исключена ситуация, когда количество устройств помимо датчиков дистанции будет превышать доступное число для подключения. Чтобы успешно подключить все необходимые устройства, имеется специальный макет (рисунок 17).

Изображение выглядит как пластик, в помещении

Автоматически созданное описание

Рисунок 17 – Макет

При написании кода для датчиков использовалась библиотека VL6180X.h от Pololu [8]. Она предоставляет разработчику достаточно гибкий функционал для работы с данными устройствами. Существуют более простые библиотеки, однако они не позволяют сменить ограничение дистанции. Изначально у предоставленных датчиков максимальная дальность – 20 см, что при габаритах робота и помещения может быть недостаточно.

После подключения библиотеки через #include необходимо создать объекты класса VL6180X под каждый датчик в глобальной области видимости.

При запуске функции setup() или иной, вызываемой в ней, необходимо выставить режим пина управления (SHDN) всех датчиков с помощью функции pinMode(<arg1>, <arg2>), где <arg1> – номер пина, а <arg2> – режим, в нашем случае – OUTPUT. Для читаемости и понятности кода во избежание запутанности, использовались константные значения с именами (например, SHT\_LOX1) для номеров пинов, адресов (ко всем подключаемым устройствам).

После установки режима пинов необходимо отключить их все функцией digitalWrite(<arg1>, <arg2>), где <arg1> – номер пина, а <arg2> – значение, в нашем случае – LOW. Устанавливается задержка через delay() от 10 миллисекунд. Далее необходимо включить все пины датчиков, аналогично выключению, но вместо LOW ставим HIGH. Вновь устанавливаем задержку.

Далее необходимо отключить все пины датчиков, кроме одного, вновь поставить задержку. Инициализировать датчик методом .init() объекта VL6180X, установить настройки по умолчанию методом .configureDefault(), дальность датчика методом .setScaling(2) – это увеличит дальность до 40 см. Также имеется режим «3» с дальностью до 60 см; по умолчанию установлен – «1» до 20 см. Затем использовать его адрес методом .setAddress(<адрес>). Далее включаем следующий пин другого датчика, устанавливаем задержку. Повторяем для него и последующих датчиков дистанции.

После описанной настройки датчиков VL6180X, если всё прошло успешно, появляется возможность получать информацию с них с помощью метода .readRangeSingleMillimeters(), возвращающего дистанцию в миллиметрах, используя тип данных uint16\_t.

# 2.2.6 Работа с датчиком MQ – 6

Датчик газа MQ-6 (рисунок 18) имеет 4 выхода (рисунок 19), из которых важными являются: питание и информация.

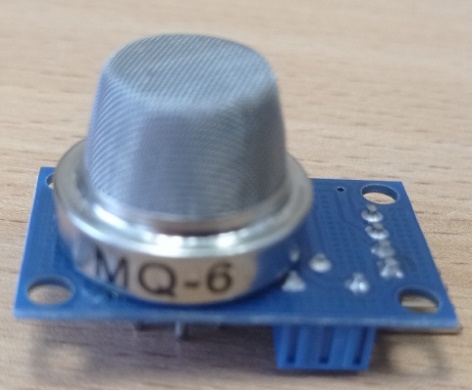


Рисунок 18 – Датчик газа MQ-6

Изображение выглядит как Электронная техника, электроника, Компонент схемы, Электронный компонент

Автоматически созданное описание

Рисунок 19 – Обратная сторона датчика газа

Питание датчика подключается аналогично Motor Control Board: GND и VCC к GND и 5V на Arduino, соответственно, секция «POWER». Данные с датчика могут быть прочтены по выходу A0, который подключается в порт A0 на Arduino в секцию «ANALOG IN» (рисунок 20).

Изображение выглядит как схема, Электронная техника, электроника, Электронный компонент

Автоматически созданное описание

Рисунок 20 – Подключение MQ – 6 к Arduino Mega 2560

Есть несколько способов работы с датчиками семейства MQ: с использованием сторонних библиотек и без. Основным решением при разработке стала библиотека TroykaMQ.h [9].

После подключения библиотеки необходимо создать объект класса MQ6 с указанием пина подключения (по данному руководству – это А0): MQ6 <название\_объекта>(А0). В начальной конфигурации микроконтроллера (функция setup()) необходимо, прежде всего, выставить задержку, чтобы дать датчику прогреться (рекомендуемое время прогрева – 20-60 секунд, можно воспользоваться функцией delay()). Затем откалибровать датчик, вызвав метод .calibrate() (датчик должен находиться в чистом воздухе в этот момент). Если известно точное значение сопротивления датчика в чистом воздухе, Ro, в помещении, то можно самостоятельно указать его в аргументе метода, тип аргумента – float). На этом настройки окончены и датчик готов к работе. В рамках библиотеки TroykaMQ.h предоставляются следующие возможности: чтение откалиброванного сопротивления (метод .getRo()), чтение отношения сопротивления в откалиброванном воздухе и текущем (метод .readRatio()) и вывод количества газа в единицах ppm (метод .readLPG()).

Для работы с датчиком без внешних библиотек понадобятся две переменные типа float: в одну записываются данные, поступающие с аналогового пина А0, во вторую – преобразованное значение. В ходе работы микроконтроллера (функция loop()) для повышения точности считывания берём несколько значений с пина за одну итерацию loop(), а затем делим на количество, чтобы получить среднее значение. Далее это значение преобразуем по формуле (1):

, (1)

где: averangeValueFromA0 – среднее значение за определённое количество итераций считывания, например, 100; newValue – новое значение, которое уже анализируется на наличие газа.

Для подтверждения наличия в области постороннего газа показатель датчика должен превысить определённое значение. В случае без использования библиотеки норма составляет примерно 0,24 (значение newValue из формулы 1), появление газа в помещении можно фиксировать при значении примерно >1,00. При работе с TroykaMQ.h норма составляет ~4 ppm (метод .readLPG()) – повышенными считаются значения, переступающие порог в несколько сотен (при больших концентрациях газа показания могут исчисляться в десятках и сотнях тысяч ppm).

# 2.2.7 Работа с моторами и платой управления

На рисунке 21 в левый разъем подключено внешнее питание. Чтобы плата питалась именно от стороннего источника, необходимо переставить жёлтую заглушку в положение, как на изображении. Сверху, в секциях «М1» и «М2» подключаются 2 мотора. Справа подключены провода, ведущие к микроконтроллеру: M1A, M1B и M2A, M2B – два мотора соответственно подключаются в секцию «PWM», как и управление датчиков дистанции. GND и VCC – питание, подключаются к GND и 5V на Arduino в секцию «POWER».

Изображение выглядит как Электронная техника, электроника, Электронный компонент, Компонент схемы

Автоматически созданное описание

Рисунок 21 – Подключенная Motor Control Board

Для основной работы с моторами в коде не требуется особых библиотек. Необходимо, как и у датчиков дистанции, выставить режим пинов (4 пина для 2-х моторов) на OUTPUT с помощью функции pinMode(). После этого моторами можно управлять путём записи значений в пины функциями digitalWrite() (для установки направления движения, значений HIGH и LOW можно использовать пин B с Motor Control Board; 2 аргумента: номер пина и значение) и analogWrite(), используемый для установки скорости движения (значения до 255, можно использовать пин A с Motor Control Board; 2 аргумента: номер пина и значение).

# 2.2.8 Отправка полученных данных с датчиков на верхний уровень

На верхнем уровне роботом управляет фреймворк ROS под управлением Linux.

Во время разработки для передачи данных на верхний уровень использовалась программа, написанная в файле machine.ino. В нем для связи с верхним уровнем имеется:

1) Импорт библиотек – ros.h (основная), std\_msgs/String.h (представляет собой структуру данных, а не простой тип; значение в объекте хранится в поле .data), geometry\_msgs/Twist.h (подробнее о нём в пункте 2.6);

2) Создание основных объектов классов – NodeHandle (главный модуль взаимодействия), Publisher (будет отсылать сообщения наверх), Subscriber (будет принимать сообщения сверху);

3) Обработчик получения команд с верхнего уровня (метод getROScmdVel()), который будет регулировать скорость и направление движения (подробнее о нём в пункте 2.6);

4) Инициализация всех объектов происходит в функции setup() (блок #ifdef ROS ... #endif);

5) Пример отправки пустой строки наверх можно наблюдать в функции loop().

# 2.2.9 Подготовка и настройка РОБОГАЗ

Как говорилось ранее, разработка ПО для робототехнического комплекса велась на микрокомпьютере Orange Pi 3 LTS. Также была проведена работа по моделированию корпуса для робота для печати на 3D принтере (рисунок 22, 23).

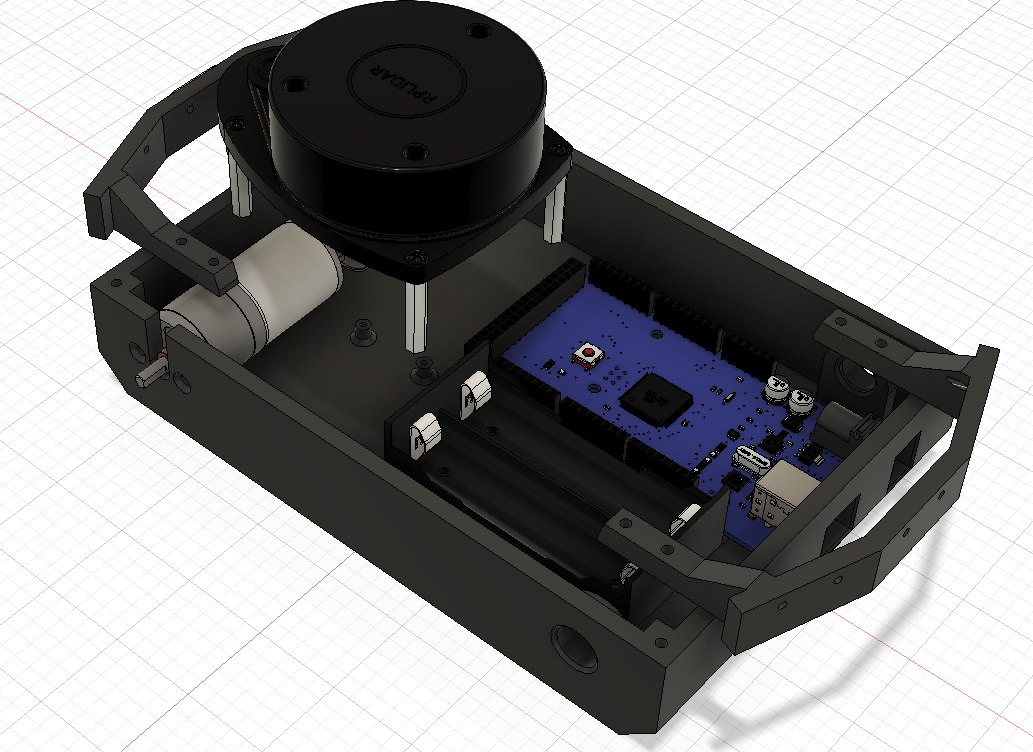


Рисунок 21 – Моделирование расположения нижнего уровня в РОБОГАЗ

Изображение выглядит как текст, черный

Автоматически созданное описание

Рисунок 22 – Моделирование верхнего уровня РОБОГАЗ

Так РОБОГАЗ выглядел после печати корпуса и сборки для дальнейшего программирования алгоритма движения и подключения верхнего уровня к нижнему (рисунок 23).

Изображение выглядит как в помещении, кабель, электроника, Электрическая проводка

Автоматически созданное описание

Рисунок 23 – собранная модель РОБОГАЗ

Основным файлом для написания алгоритма парсинга данных с датчиков дистанции, газа и RPLidar является pysenors\_data\_node.py. Данный файл лежит в корневой папке проекта по пути: orangepi/catkin\_ws/src/NTO\_Robot/pysensors\_data/src/. В данном файле имеются:

1. Основной класс программы – KeyboardTeleop. Данный класс представляет собой основное рабочее пространство программы, в котором пишется весь код;
2. Инициализация основных элементов ROS, настроенные «Subscribers» и «Publisher».
3. Основной метод run() для запуска алгоритма;
4. Метод onIRSensor() вызывается автоматически при получении данных датчиков с нижнего уровня; здесь необходимо правильно интерпретировать приходящее строковое значение, чтобы работать с показаниями датчиков в дальнейшем;
5. Метод onLaserSensor() вызывается автоматически при получении данных с лидара. В нём предоставляется пробный вариант кода, который может быть частично или полностью изменён;
6. Метод update\_velocity() вызывается в ходе программы в методе run(). Это самый основной метод для работы, поскольку в нём необходимо реализовать алгоритм прохождения лабиринта (определение по данным с датчиков и лидара направления и скорости движения);
7. В конце метода update\_velocity()имеется заготовка отправки ответа на нижний уровень с указаниями, куда необходимо будет записать значения линейного (twist.linear.x) и углового (twist.angular.z) движений.

Перед запуском робота требуется подключение всех устройств и ввод трёх команд в терминал Linux. Ввод команд необходим, поскольку робот полностью автономен и не имеет управления с помощью клавиатуры или геймпада (как в некоторых случаях). Порядок действий для запуска робота:

1. Подключить устройство RPLidar и Arduino. Рекомендуется подключать в таком порядке, чтобы данные устройства были подключены в настроенные для них разъемы. В программе данные порты обозначаются как /dev/ttyUSB0 для Lidar и /dev/ttyUSB1 для Arduino соответственно;
2. Запустить главное ядро системы ROS и начать считывать данные с Lidar, используя команду: roslaunch rpplidar\_ros rplidar.launch. По умолчанию команда будет искать подключаемое устройство в настроенный порт (указано в 1 пункте);
3. Следующее действие открывает порт для получения данных с нижнего уровня. Данную команду нужно вводить либо в новой вкладке, либо в другом терминале: rosrun rosserial\_python serial\_node.py \_port:=/dev/ttyUSB1 \_baud:=115200 (параметры \_port и \_baud могут быть изменены в зависимости от настройки порта);
4. Следующий этап — это получение данных. После ввода команд происходит запрос данных с нижнего уровня, но первые 10 секунд не будет поступать никакой информации; она появится только после прогрева датчика газа MQ – 6 и запуска микроконтроллера Arduino;
5. Выполнив все обозначенные выше условия, небходимо запустить программу, написанную для верхнего уровня, с помощью консольной команды, либо в новом терминале, либо в новой вкладке: rosrun pysensors\_data pysensors\_data.py;
6. При успешном выполнение пунктов 1-5 робот готов к запуску.

# 2.2.10 Прием данных с нижнего уровня

Каждый раз при получении данных с нижнего уровня вызывается метод onIRSensor(self, msg). За это отвечает Subscriber sub\_front. Данные хранятся в msg в виде строки String, для её изъятия необходимо использовать msg.data, из которой предстоит извлечь значения путём преобразования строки для дальнейших манипуляций с ними.

В ROS термин «msg» относится к файлам сообщений (message), которые определяют структуру данных, используемых для обмена информацией между узлами ROS. Файлы сообщений (msg) определяют типы данных, которые могут быть переданы через топики (topics) или сервисы (services) в ROS-системе.

Файлы сообщений описывают поля данных, которые могут быть переданы в сообщении. Каждое поле имеет свой тип данных, такие как целые числа, вещественные числа, строки, массивы и т.д. Определение сообщений (msg) включает в себя указание названия сообщения и набора полей данных, которые должны быть переданы.

В случае ошибок имеет смысл проверить корректность присылаемых данных в теме SensorsData с помощью rqt и Topic Monitor (подробнее об этом в конце пункта 2.2.15).

# 2.2.11 Работа с RPLIDAR в коде

Если следовать предложенному варианту, то область лидара делится на 24 секции (пункт 2.2.12, рисунок 24). Создадим пустой список под секции ([]) и заполним его «0» в методе \_\_init\_\_ предложенного класса.

Subscriber sub\_lidar будет собирать данные с него и вызывать метод onLaserSensor (тип сообщения – LaserScan, структура которого описана по ссылке: http://docs.ros.org/en/noetic/api/sensor\_msgs/html/msg/LaserScan.html).

# 2.2.12 Обработка полученных данных

Для примера описания решения задачи будет рассмотрено решение алгоритмом левой руки.

Источниками данных при работе робота являются датчики дистанции и Lidar. Предлагаемый вариант решения выглядит следующим образом:

1. Выделить область (рисунок 24) после чего объединить и усреднить значения, входящие в эту область, выбрать набор значений лидара, располагающихся под нужным углом;

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рисунок 24 – «Сегментация» данных с устройства RPLIDAR

1. Определить области лидара, с которых программно будут считываться данные по положению робота;
2. Основным источником для маневрирования является Lidar, по его данным можно определить положение работа относительно стены, вдоль которой движется робот. При решении требуется определиться, на каком расстоянии от стены лучше двигаться;
3. Завести переменные, которые будут содержать угловое и линейное ускорение, так как эти значения почти наверняка придётся править после тестирования, как минимум потому, что точно указать угловую скорость, ни разу не запустив робота, скорее всего, не удастся;
4. При работе с датчиками требуется указать скорость и направление поворота в зависимости от датчика. В случае если передний датчик можно затормозить, сдать назад или задать поворот по желанию.

В качестве условий для лидара могут выступать:

1. При отдалении от стены более чем на n по диапазону данных лидара, следящих за расстоянием до стены слева, придавать роботу угловую скорость, помимо линейной. Следует помнить, что при повороте имеет смысл смотреть на разные точки RPLIDAR – а (рисунок 25);
2. В случае если в выбранном диапазоне значений n не превышен, можно двигаться вперед, пока не превышено значение n (также оно может отличаться по желанию) по диапазону лидара, следящего за расстоянием до стен около робота;
3. С диапазоном правой стороны процесс работы аналогичен.

Изображение выглядит как диаграмма, линия, дизайн

Автоматически созданное описание

Рисунок 25 – Схема поворота (с учетом изменения положения в пространстве RPLIDAR – а)

# 2.2.13 Алгоритм прохождения маршрута

Для написания алгоритма перемещения в замкнутом пространстве имелся очень обширный выбор доступных вариантов:

1. Алгоритм правой руки – робот следует вдоль правой стены в помещении, поворачивая только направо, пока не достигнет цели или выхода (рисунок 26). В случае если нет возможности повернуть направо, робот движется вперед. Если же возникла ситуация, в которой преграда находится справа или прямо, комплекс поворачивает налево. Сам алгоритм основан на следующем принципе: если продолжать движение по правой стене, рано или поздно можно добраться до выхода или цели. Однако такой алгоритм может быть использован, если помещение односвязано или многосвязано, но с петлей вокруг выхода;.

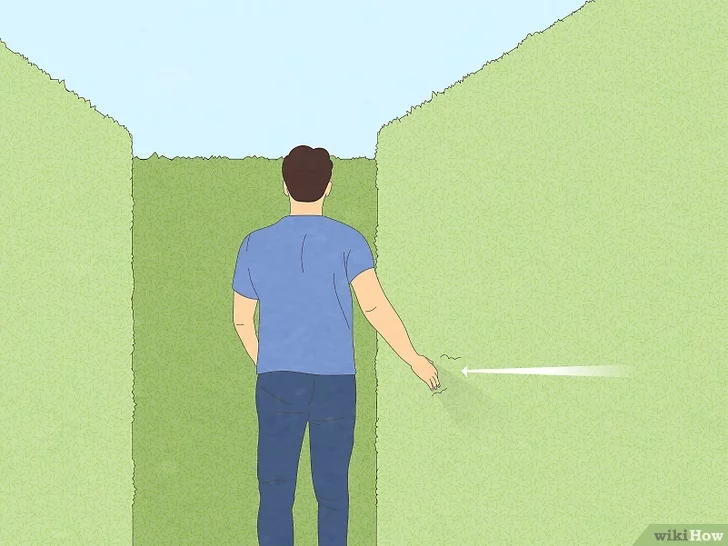


Рисунок 26 – Алгоритм правой руки

1. Алгоритм левой руки – этот алгоритм аналогичен алгоритму правой руки, но ориентация робота происходит по левой руке;
2. Алгоритм поиска в глубину – сам принцип работы данного алгоритма представляет собой исследования роботом всех доступных путей в помещении от начальной позиции (рисунок 27). Если, например, комплекс достигнет развилки, он случайно выберет поворот. После робот движется прямо пока не упрётся в тупик или не наткнётся на ещё одну развилку. Углубление будет продолжаться до тех пор, пока робот не упрется в тупик или, в случае если точка назначения находится в выбранном направлении, не достигнет конечной точки. В случае если конечная точка не достигнута, робот отъедет назад до последней посещённой развилки и свернёт на неисследованный путь. Он постепенно расширяет свою область поиска, пока не достигнет выхода или цели;

Изображение выглядит как Прямоугольник, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 27 – Пример алгоритма поиска в глубину

1. Алгоритм поиска в ширину – данный алгоритм представляет помещение как граф и позволяет найти кратчайшие пути из одной вершины невзвешенного (ориентированного или неориентированного) графа до всех остальных вершин (рисунок 28). Под кратчайшим путем подразумевается путь, содержащий наименьшее число ребер;

Изображение выглядит как прямоугольный, Прямоугольник, шаблон, Красочность

Автоматически созданное описание

Рисунок 28 – Пример алгоритма поиска в ширину (зеленая линия – правильный путь)

1. Алгоритм А\* – этот алгоритм использует эвристическую оценку для выбора наилучшего пути до цели (рисунок 29). Робот вычисляет стоимость прохода через каждую ячейку лабиринта с учетом расстояния до цели и стоимости перемещения. Он выбирает наименьшую стоимость для продолжения движения и постепенно приближается к цели.

Изображение выглядит как диаграмма, Прямоугольник, План, снимок экрана

Автоматически созданное описание

Рисунок 29 – Пример алгоритма А\*

Для прохождения маршрута роботом был написал алгоритм левой руки, но из-за его минуса, связанного с очень плохой навигацией в помещении, было принято решение со временем провести модификацию до алгоритма поиска в глубину. В будущем данный комплекс будет дорабатываться, и алгоритм будет изменен на поиск в глубину для более точной ориентировки в помещении и повышения функционала робота.

# 2.2.14 Управление движением РОБОГАЗ

Ранее было сказано, что робот полностью автономный, но все же есть способ им управлять. Управление движением робота осуществляется через специальный топик – «cmd\_vel». В нем публикуются данные структуры Twist (внутри два вектора: для линейного движения вперёд-назад – linear; для углового движения – angular; конкретнее – здесь используется ускорение по осям и вокруг осей). Векторы внутри Twist содержат 3 числа типа float64 – x, y, z. Робот будет перемещаться на плоскости (без подъёмов и спусков, но только в рамках тестирования комплекса), так что для работы понадобятся два значения – linear.x (для движения вперёд-назад) и angular.z (для задания поворота). Эти значения могут быть в диапазоне от -m до m (значение m – десятичное число float64, которое может выбрать разработчик), где знак отвечает за направление: «+» – движение вперёд и вправо, а «-» – движение назад и влево для linear.x и angular.z соответственно. Само число характеризует скорость движения и поворота РОБОГАЗ в м/с (метрах в секунду).

В файле разработки содержится метод update\_velocity(self) для кода по определению текущей скорости и направления. Была разработана заготовка создания и отправки в cmd\_vel сообщения Twist; в нее необходимо ввести рассчитанное значение для линейного и углового движения.

Как упоминалось в разделе программирования нижнего уровня, моторы принимают скорость в виде pwm (диапазон от 0 до 255), так что понадобятся дополнительные написанные вручную функции на Arduino по интерпретации значений из cmd\_vel в направление и скорость движения для моторов.

# 2.2.15 Отладка РОБОГАЗ перед запуском

Для проверки настроенного движения робота нет необходимости запускать основную программу pysensors\_data\_node.py. Есть возможность открыть инструмент «rqt» (команда rqt в новом терминале или вкладке при выполненной команде roslaunch rplidar\_ros rplidar.launch). Это довольно простой и удобный инструмент для отслеживания и управления во время разработки. Основными функциями для данной задачи являются: Topic Monitor (Вкладка Plugins => Topics => Topic Monitor) и Robot Steering (Вкладка Plugins => Robot Tools => Robot Steering). Topic Monitor (рисунок 30) позволяет отслеживать все темы в системе (к примеру, cmd\_vel, SensorsData, scan), что пригодится при проверке корректности присылаемых данных с нижнего уровня. Для отслеживания определённого топика необходимо поставить галочку левее него и раскрыть список. Robot Steering (рисунок 31) представляет собой удобный инструмент отправки данных в cmd\_vel с помощью интуитивно понятных ползунков и клавиш. Существует возможность ограничивать максимальные значения ползунков.

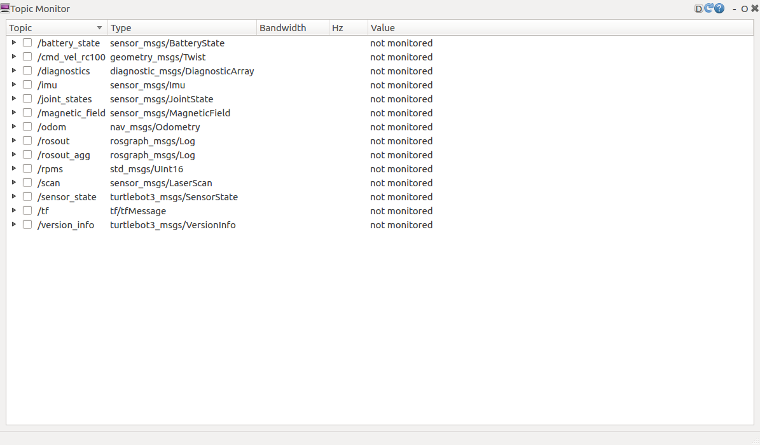


Рисунок 30 – Интерфейс Topic Monitor (количество и названия Topic зависят от текущих в системе)

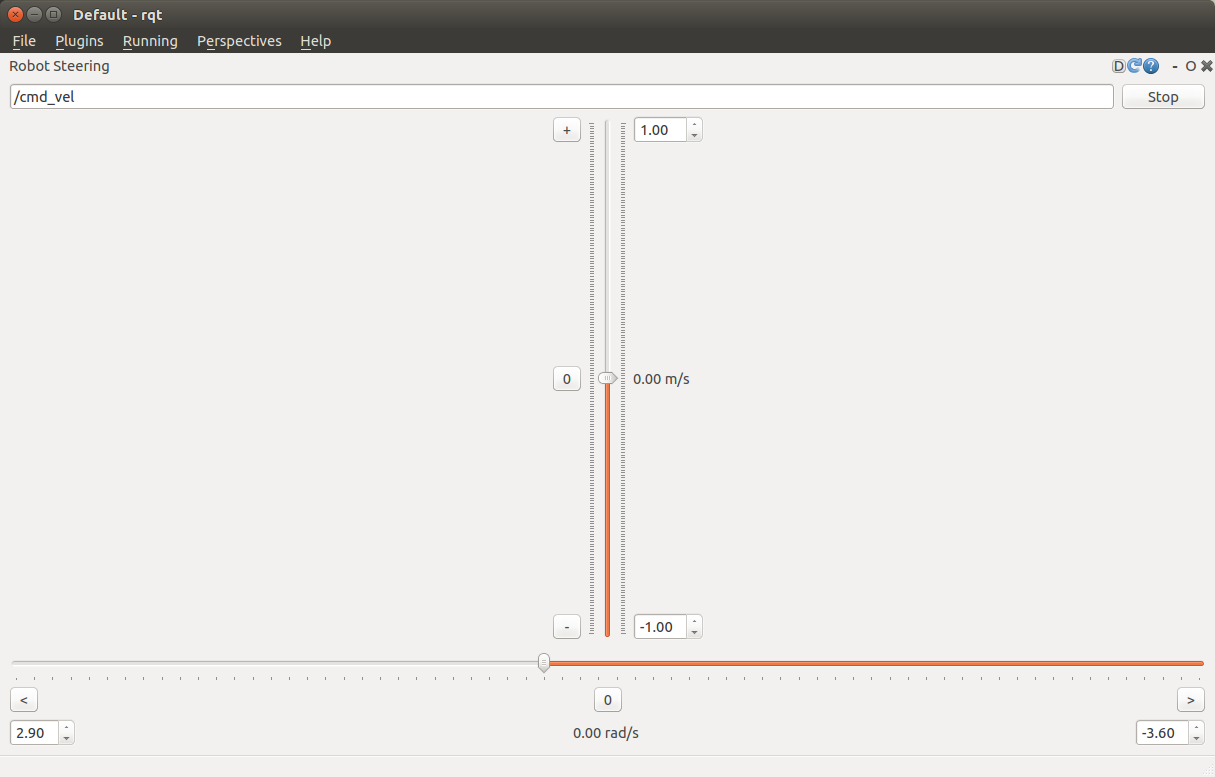


Рисунок 31 – Интерфейс Robot Steering (может незначительно меняться в зависимости от OS)

# 2.3 Тестирование РОБОГАЗ

Тестирование – это процесс проверки программного обеспечение на соответствие требованиям, выявление ошибок и обеспечение гарантирование качества. Оно выполняется с целью обеспечения надежной и стабильной работы программы, позволяет выявить ошибки и дефекты в программном обеспечении и не только, что в последующем помогает улучшить качество и надежность программного продукта.

Во время написания алгоритма движения для РОБОГАЗ были выявлены несколько ошибок:

1. РОБОГАЗ крутился вокруг своей оси (замена 2 задних колес на маленькое колесо);
2. Недостаточная плавность поворота (исправление вычислений в алгоритме движения);
3. Недостаточная чувствительная датчика газа;
4. Сильный нагрев микрокомпьютера Orange Pi при работе с ROS.

Во время разработки и тестирования возникли ошибки, которые указаны в 1-4 пункте. Опытным путем удалось исправить их все, за исключением пунктов 3 и 4. Датчик газа MQ-6 выдавал значения в ROS ниже, чем в самостоятельном запуске. Также на процессор был установлен радиатор, что позволило охладить его и избавиться от постоянных зависаний.

Для тестирования был построен лабиринт, который соответствовал размерам 3 метра в длину на 3 метра 60 сантиметров в ширину, а также имел высоту 50 см. Точка старта робота определена буквой А, а финиш буквой B (рисунок 32). РОБОГАЗ имел задачу пройти от точки старта до точки финиша, в том числе локализовать местоположение газа по мере движения к выходу. Для запуска робота требуется ввод 3 команд в терминал Linux (описаны в пункте 2.2.9). В момент впрыскивания газа в лабиринт при превышении порога >4 ppm (появление газа фиксируется при >=1.0 ppm, а норма составляет >= 4ppm) данные фиксируется в файле pysensors\_data.py. Уведомление о превышении газа приходит в виде строки в терминал Linux.

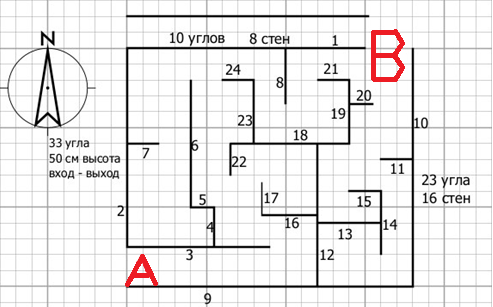


Рисунок 32 – Проектирование лабиринта в программе Sweet Home 3D

Окончательное тестирование проходило на полосе препятствий в Гидробашне Политех перед олимпиадой НТО (рисунок 33).

Изображение выглядит как в помещении, мебель, дизайн интерьера, домашнее растение

Автоматически созданное описание

Рисунок 33 – Полоса препятствий (лабиринт) для тестирования РОБОГАЗ

После учета и исправления всех ошибок при разработке, комплекс РОБОГАЗ проехал полосу препятствий, обнаружив выпущенный газ в замкнутом пространстве.

# 3 Экономическая часть

# 3.1 Технико-экономическое обоснование

В данном разделе даётся технико-экономическое обоснование разработки робототехнического комплекса для Лаборатории «Промышленные системы потоковой обработки данных» Центра НТИ СПбПУ.

Цель работы: разработать программно-аппаратный робототехнический комплекс для поиска утечек опасных газов в замкнутом пространстве, который предполагает:

* разработку аппаратной части комплекса с подключением датчиков, устройств и элементов управления;
* создание программного обеспечения для сбора, анализа и обработки данных;
* разработку алгоритма перемещения робота;
* тестирование комплекса на специально подготовленной площадке;
* оптимизацию работы комплекса с учетом ошибок, выверенных во время тестирования.

# 3.2 Расчет экономической эффективности от внедрения программного продукта

Критерием эффективности, создания и внедрения новых средств автоматизации является ожидаемый экономический эффект. Он определяется по формуле 2:

(2)

где Эр – годовая экономия;

Ен – нормативный коэффициент. Является константой и равен 0,15;

Кп – капитальные затраты на проектирование и внедрение, включая первоначальную стоимость программы.

Годовая экономия складывается из экономии эксплуатационных расходов и экономии в связи с повышением производительности труда пользователя. Таким образом, получаем формулу 3:

(3)

где Р1 и Р2 – эксплуатационные расходы до и после внедрения разрабатываемой программы;

– экономия от повышения производительности труда пользователей.

# 3.3 Расчет капитальных затрат на проектирование и внедрение

При оценке экономического эффекта с учетом всех деталей капитальные затраты на проектирование и внедрение рассчитываются с учетом длительности работ на этом этапе.

Рассмотрим подробнее расчет капитальных затрат на проектирование и внедрение системы автоматизации. Под проектированием понимается совокупность работ, которые необходимо выполнить, чтобы спроектировать систему, часть системы или поставленную задачу. Под внедрением понимается комплекс работ по вводу в промышленную эксплуатацию системы с возможными её доработками.

Для расчета затрат на этапе проектирования необходимо определить продолжительность каждой работы. Продолжительность работ определяется либо по нормативам.

Капитальные затраты на этапе проектирования Кп рассчитываются по формуле 4:

(4)

где Zп – заработная плата специалистов на всех этапах проектирования и внедрения;

Mn – затраты на использование ЭВМ на этапах проектирования и внедрения;

Н – накладные расходы на этапе проектирования и внедрения.

Одним из основных видов затрат на этапе проектирования является заработная плата специалиста, которая определена или рассчитывается по формуле 5:

(5)

где Zп – заработная плата разработчика на этапе проектирования;

Zд – дневная заработная плата разработчика на этапе проектирования;

Тп – общая ожидаемая длительность работ (в днях);

Ас – процент отчислений на социальное страхование;

Ап – процент премий.

Расчет заработной платы специалиста производится из расчета оклада по должности «Техник-программист», который составляет 50 тыс./мес., в день оплата составляет 2272 руб. 73 коп. Премия составляет 10% от суммы оклада, т. е. 5000 руб.

Ожидаемая длительность работ на этапе проектирования берется нормативная, и составляет 9 дней, длительность работ на этапе разработки – 46 дней, на этапе документирования – 15 дней.

= 2272.73 70 = 227500 руб.

В общем случае расходы на машинное время состоят из расходов на процессорное и дисплейное время. Расходы на машинное время рассчитываются по формуле 6:

(6)

где tд, tп – процессорное и дисплейное время, необходимое для решения задач (в часах);

Сд, Сп *–* стоимости одного часа дисплейного и процессорного времени.

При расчете Мп следует учитывать время на подготовку исходных текстов программ, их отладку и решение контрольных примеров. Поскольку программа разработана на быстродействующих компьютерах, в дополнительном процессорном времени необходимость отсутствует. Таким образом, значения Cп и tп принимаются нулевыми.

Продолжительность работ 70 дней. Стоимость одного часа процессорного и дисплейного времени составляет 20 рублей. По трудовому кодексу продолжительность рабочего дня составляет 8 часов, из которых час отводится на обед. Таким образом, в день на разработку уходит 7 часов, тогда Мп будет равно значению:

Расходы на эксплуатационные принадлежности на этапе проектирования определяются подсчетом затрат на их приобретение по оптовым или свободным ценам. Ниже представлены суммы расходов на приобретение эксплуатационных принадлежностей:

* стоимость среды разработки (Visual Studio – бесплатно);
* стоимость пачки бумаги – 400 руб.;
* стоимость картриджа для принтера – 1300 руб.;
* стоимость интернета – тариф 1000 рублей в месяц, т. е. примерно 33 рубля в день, в пересчете на 70 дней – 2310руб.;

Итого накладные расходы на этапе проектирования и внедрения будут составлять 4010 рублей. Следовательно, капитальные затраты на этапе проектирования составят:

Кп = 227500 + + 4010 = 241310 руб.

# 3.4 Расчет эксплуатационных расходов

В эксплуатационные расходы входят:

* содержание персонала по обслуживанию комплекса технических средств;
* расходы на функционирование программы;
* прочие расходы.

# 3.4.1 Расчет эксплуатационных расходов

Расходы по различным видам работающих определяются по формуле 7:

(7)

где ni – численность персонала i вида, связанного с выполнением работ;

zi – среднегодовая заработная плата работника;

Ac – процент отчислений на социальное страхование;

An – среднегодовой процент премии (если предусмотрена).

Предполагается, что с разработанной программой будет работать 1 человек. Также предполагается, что сопровождение и устранение неполадок в системе будет занимать около 2-х часов в неделю. Заработная плата – 3000 рублей в день при 8-часовом рабочем дне (375 рублей в час). За месяц при 2-х рабочих часах в неделю – 375 \* 2 \* 4 = 3000 рублей в месяц. Среднегодовая заработная плата равна 36000 рублей в год. Следовательно:

Z = 36000 \* 1,30 \* 1,15 = 53820 руб.

# 3.4.2 Расходы на функционирование программы

Расходы на функционирование программы складываются из затрат на машинное время и затрат на эксплуатацию различных принадлежностей (накладные расходы). Из формулы (4) произведем расчет расходов на функционирование программы

При этом можно оценить расходы до внедрения (М1) и после внедрения (М2) и сравнить полученные значения. При внедрении программы уменьшается время работы с одной и той же задачей, за счет чего появляется экономия.

До внедрения РОБОГАЗ, решением данной задачи занимались сотрудники, при помощи специальной техники и бумажного тестирования. В качестве исходных данных будем использовать данные, приведенные в таблице 2.

До внедрения робототехнического комплекса работа для получения и изучения информации велась вручную и при использовании ПК для самостоятельного изучения, что занимало около 4 часов в день. Стоимость одного часа процессорного и дисплейного времени составляет 20 рублей. В году 240 рабочих дней, следовательно:

*=* 19200 руб.

Работа за компьютером после внедрения программы значительно сократит время на анализ исследований человеком и в среднем при работе одного человека займет около 1 часа в день. В соответствии с этим, расходы на функционирование программы составят:

= 4800 руб.

# 3.4.3 Прочие расходы

Прочие расходы составляют от 1 до 3-х процентов от суммы всех эксплуатационных расходов. Прочие расходы до внедрения программы определяются по формуле 8:

(8)

Соответственно, прочие расходы до внедрения программы равны:

Прочие расходы после внедрения программы определяются по формуле 9:

(9)

Соответственно, прочие расходы после внедрения программы равны:

Таким образом, эксплуатационные расходы составляют:

* до внедрения программы (формула 10);
* после внедрения программы (формула 11).

(10)

(11)

Если пользователь i вида работ с применением программы экономит часов, то повышение производительности труда (в процентах) определяется по формуле 12:

(12)

где – время, которое планировалось пользователем для выполнения работы j вида до внедрения программы (в минутах);

– экономия времени (в минутах).

Таблица 2. Работы пользователей

| Вид работ | До автоматизации () | После автоматизации () | Экономия времени, мин. | Повышение производительности труда   (в %) |
| --- | --- | --- | --- | --- |
| Ввод информации | 60 | 20 | 40 | 100 |
| Проведение расчётов | 60 | 5 | 10 | 200 |
| Редактирование данных | 45 | 15 | 15 | 100 |
| Поиск и выборка данных | 60 | 15 | 45 | 300 |

Экономия, связанная с повышением производительности труда пользователя (Рп), определяется по формуле 13:

(13)

где Zn – заработная плата пользователя за месяц;

Pi – повышение производительности труда (в %).

Предполагается, что заработная плата пользователя программы составляет 50000 рублей в месяц. Экономия, связанная с повышением производительности труда пользователя, будет равна:

Тогда, по формуле (3) получим годовую экономию Эр:

В итоге получаем следующую ожидаемую экономическую эффективность Э по формуле (2):

Вывод: Для предприятия Лаборатория «Промышленные системы потоковой обработки данных» Центра НТИ СПбПУ внедрение продукта является экономически выгодным. Продукт предоставляет компании возможность находить утечки опасных газов в замкнутых помещениях, а также использовать данный комплекс для обучения студентов по направлению «Робототехника». Данный комплекс поможет находить утечки в сжатые сроки, определять их характер и объем, что можно использовать для уменьшения затрат на персонал и не подвергать человеческие жизни опасности. Также данный комплекс может использоваться для развития образовательного учреждения по направлению «Робототехника».

В процессе расчета экономический эффективности необходимо учитывать одно свойство автоматизации: чем больше средств и времени потрачено на автоматизацию, тем выше экономический эффект от внедрения. Объясняется это довольно просто: если качественно подойти к выбору программного продукта, качественно проработать, описать и отладить все процессы на этапе проектирования и внедрения, то впоследствии будет потрачено гораздо меньше средств на эксплуатацию программы.

# 4 Техника безопасности и охрана труда

# 4.1 Анализ условий труда программиста

Рабочее место программиста – это секция помещения, где он проводит большую часть рабочего времени. Совокупность показателей технических и программных средств определяет качество и производительность работы, наряду с габаритными показателями мебели и её удобством.

Помещение для работы представляет собой кабинет длиной 6 м, шириной 6 м и высотой потолков 2.5 м.

Согласно санитарным правилам, монитор на столе нужно располагать на расстоянии 60–70 см от глаз пользователя, но не ближе 50 см с учетом размеров алфавитно-цифровых знаков и символов. При использовании жидкокристаллических дисплеев на каждый компьютер должно приходиться не менее 4,5 м2 площади. На дисплей ПК не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Для снижения нагрузки на технические средства, влекущей за собой перегрев компьютера и ноутбука, используются программные средства, позволяющие снизить нагрузку на технические средства за счёт отказа в использование ненужных служб.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

* электромагнитные поля и излучения от технических средств;
* статическое электричество;
* статические перегрузки, вызываемые неподвижной работой за компьютером;
* перенапряжение органов зрения.

В целях выявления вредных и (или) опасных факторов производственной среды и трудового процесса, оценке уровня их воздействия на работника с учетом отклонения их фактических значений от установленных уполномоченным Правительством Российской Федерации федеральным органом исполнительной власти гигиенических нормативов условий труда и применения средств индивидуальной и коллективной защиты работников проводится специальная оценка условий на рабочих местах. Оценке подлежат все имеющиеся в организации рабочие места.

Работа по специальной оценке условий труда заключается в исследовании следующих факторов на рабочих местах организации:

* химические факторы;
* физические факторы;
* биологические факторы;
* тяжесть и напряженность трудового процесса.

Для уменьшения нагрузки на органы зрения, а также снятия мышечного напряжения во время рабочего дня проводится проветривание в течение 10–15 минут каждые 3 часа.

# 4.2 Расчёт искусственного освещения в помещении

Через зрительный анализатор человек получает около 80% от общего объема информации. Качество поступающей информации, в основном, зависит от освещения в помещении. Если оно неудовлетворительное, это приводит к утомлению организма в целом. Кроме того, ненадлежащее освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени ухудшают видимость настолько, что вызывают полную потерю ориентации в пространстве работающих, снижают производительность труда и увеличивают вероятность возникновения брака продукции. В связи с этим, необходимо достаточное искусственное освещение рабочего места.

Условия работы для расчета показателя освещенности рабочего места программиста являются:

* помещение размером 36 квадратных метров;
* высота помещения 2,50 метров, длина 6 метра, ширина 6 метра;
* светильники с люминесцентными лампами /светодиодные в количестве одной штуки, новые;
* крупногабаритная мебель отсутствует.

Основным критерием, по которому определяется необходимое количество осветительных приборов, является нормируемый уровень освещенности.

Базовым методом проектирования осветительной установки является метод коэффициентов использования, позволяющий вручную проводить все вычислительные процедуры при решении относительно простых светотехнических задач.

По этому методу необходимое количество светильников в осветительной установке определяется с помощью формулы 14:

, (14)

где N – количество светильников, шт;

Е – нормируемая освещенность, лк, определяется по СП 52.13330.2016 «Естественное и искусственное освещение»;

S – площадь помещения, м2.

Кз – коэффициент запаса;

U – коэффициент использования

Фл – световой поток одной лампы в светильнике, лм;

n – количество ламп в светильнике.

Коэффициент использования характеризует эффективность использования светового прибора в помещении. Для его определения необходимо знать индекс помещения φ и коэффициент отражения стен, пола и потолка.

Индекс помещения рассчитывается по формуле 15:

, (15)

где  – индекс помещения;

S – площадь помещения, м2.

h1 – высота подвеса светильников над рабочей поверхностью, м;

h2 – высота рабочей поверхности, м;

а – длина помещения, м;

b – ширина помещения, м.

=

Полученное количество светильников совпадает с числом светильников, установленных в помещении, что соответствует нормативным показателям искусственного освещения согласно СП 52.13330.2016 «Естественное и искусственное освещение».

# 4.3 Электробезопасность на предприятии

При поступлении на работу сотрудник должен пройти инструктаж по технике безопасности и электробезопасности. Работника знакомят с основными правилами по технике безопасности, предлагают внимательно прочитать действующие на предприятии инструкции, поясняя при этом отдельные правила и требования.

Инструктаж по технике безопасности при выполнении конкретной работы проводит руководитель соответствующего производственного участка, показывая инструктируемому правильные безопасные приемы работы.

Повторный инструктаж проводится для рабочих независимо от их квалификации, стажа и опыта работы, не реже 1 раза в 6 месяцев по программе инструктажа на рабочем месте.

При нарушении работающими правил и инструкций по технике безопасности, технологической и производственной дисциплины, а также в случаях изменения технологического процесса или вида работы проводят дополнительные инструктажи.

Все инструктажи оформляются записями в специальном журнале с указанием номеров или шифров инструкций. Журнал о проведении инструктажа хранится у руководителя подразделения.

Работник на рабочем месте должен соблюдать общие меры электробезопасности. При этом запрещается:

* прикасаться к задней панели системного блока при включенном питании;
* допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
* производить самостоятельно вскрытие и ремонт оборудования;
* пользоваться неисправными электроприборами и электропроводкой;
* ремонтировать электроприборы самостоятельно;
* подвешивать электропровода на гвоздях, металлических и деревянных предметах, перекручивать провод, закладывать провод и шнуры на водопроводные трубы и батареи отопления, вешать что-либо на провода, вытягивать за шнур вилку из розетки;
* прикасаться одновременно к персональному компьютеру и к устройствам, имеющим соединение с землей (радиаторы отопления, водопроводные краны, трубы и т. п.);
* пользоваться самодельными электронагревательными приборами и электроприборами с открытой спиралью;
* наступать на переносимые электрические провода, лежащие на полу.

При работе с электроприборами и оргтехникой (персональные компьютеры, принтеры, сканеры, копировальные аппараты, факсы, бытовые электроприборы, приборы освещения):

* Автоматические выключатели и электрические предохранители должны быть всегда исправны;
* Изоляция электропроводки, электроприборов, выключателей, штепсельных розеток, ламповых патронов и светильников, а также шнуров, с помощью которых включаются в электросеть электроприборы, должны быть в исправном состоянии;
* Электроприборы необходимо хранить в сухом месте, избегать резких колебаний температуры, вибрации, сотрясений;
* Для подогрева воды пользоваться сертифицированными электроприборами с закрытой спиралью и устройством автоматического отключения, с применением несгораемых подставок.

# 4.4 Пожарная безопасность на предприятии

Инструктаж по пожарной безопасности проводится по программе, разработанной ответственным по охране труда организации, с учетом требований стандартов, правил, норм и инструкций о мерах пожарной безопасности. Продолжительность инструктажа устанавливается в соответствии с утвержденной программой. Инструктаж по пожарной безопасности, как правило, проводится совместно с инструктажем по технике безопасности и в те же сроки.

Инструктаж по пожарной безопасности проходят все работники организации, независимо от их образования, стажа работы по данной профессии или должности, временные работники, командированные, обучающиеся и студенты, прибывшие на практику.

О проведении инструктажа по пожарной безопасности работник, проводивший инструктаж, делает запись в журнале учета проведения инструктажей по пожарной безопасности, с обязательной подписью инструктируемого и инструктирующего. При регистрации внепланового инструктажа по пожарной безопасности указывают причину его проведения.

Все работники организаций должны допускаться к работе только после прохождения противопожарного инструктажа, а при изменении специфики работы проходить дополнительное обучение предупреждению и тушению возможных пожаров в порядке, установленном руководителем.

На проведение первичного противопожарного инструктажа необходимо отводить не менее 1 ч. Инструктируемые должны ознакомиться:

* с действующими на объекте правилами пожарной безопасности и инструкциями;
* с наиболее пожароопасными производственными участками, где запрещается курить, применять открытый огонь;
* с возможными причинами возникновения пожара и мерами его предупреждения;
* с практическими действиями в случае возникновения пожара: вызов пожарной помощи, использование первичных средств пожаротушения, место расположения ближайшего телефона и ознакомление с правилами поведения в случае возникновения пожара, эвакуации людей и материальных ценностей.

При первичном инструктаже инструктирующий обязан рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения пожаров и загораний, указать место для курения, ознакомить вновь поступившего с имеющимися на объекте средствами пожаротушения, показать ближайший телефон и объяснить правила поведения в случае возникновения пожара.

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и т. д.).

Соблюдение рассмотренных в данном разделе мероприятий позволяет снизить утомляемость и травматизм, повысить производительность труда, обеспечить комфортные условия трудовой деятельности.

# ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработан программно-аппаратный робототехнический комплекс (РОБОГАЗ), предназначенный для эффективного обнаружения и локализации утечек опасных газов в замкнутых пространствах. Код был загружен на GitHub [15]. Процесс разработки включал в себя проведение анализа проблемы утечки газов, определение требований, выбор подходящих аппаратных компонентов, разработку необходимых алгоритмов и создание удобного интерфейса управления.

Основными достижениями в работе являются:

1. В рамках комплекса была использована аппаратная платформа, обеспечивающая мобильность и маневренность робота. Были выбраны и интегрированы специализированные датчики для обнаружения газов, что позволило достичь высокой чувствительности и точности при поиске утечек пропана и бутана;
2. Были разработаны алгоритмы для эффективного обнаружения и локализации утечек газов. С помощью программного обеспечения осуществляется управление роботом, сбор и анализ данных с датчиков;
3. Разработанный комплекс был подвергнут испытаниям, включающим симуляции различных сценариев утечек и реальные испытания в контролируемых условиях. Полученные результаты позволяют сделать вывод о высокой эффективности и точности работы комплекса в обнаружении и локализации утечек газов;
4. Разработанный комплекс имеет широкие перспективы применения в различных отраслях, таких как промышленность, нефтегазовая отрасль, химическая промышленность и другие области, но только после смены датчиков газа на более узкоспециализированные и настройки комплекса под определенную отрасль, если он будет в таковой нуждаться;
5. Данный комплекс будет совершенствоваться в будущем для определения других видов утечек, лучшей безопасности, автономности и перемещения в замкнутом пространстве.

Из минусов комплекса хотелось бы отметить:

1. Низкую безопасность – для перемещения робота используются коллекторные двигатели; также возникла проблема с печатью колес для комплекса – их пришлось покрыть клеящейся антискользящей лентой (рисунок 23), что при работе с газом может дать искру и привести к возникновению взрыва, однако в будущем данная неполадка будет устранена;
2. Слабый алгоритм перемещения – из-за недостатка знаний не удалось в кратчайшие сроки реализовать алгоритм движения в глубине, поэтому робот использует алгоритм левой руки для перемещения по замкнутому пространству, что в дальнейшем также будет исправлено и доработано.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Марк Лутц, Изучаем Python / 2019 год. - 1272 с.

2 Григорий Сапунов, Основы конструирования и программирования роботов / 2019 год. - 256 с.

3 Python 3.10 documentation [Электронный ресурс]. – Режим доступа : https://docs.python.org/3.10. – Загл. с экрана.

4 Документация по языку Python 3 [Электронный ресурс]. – Режим доступа : https://docs-python.ru. – Загл. с экрана.

5 Морган Куинн, Программирование роботов на ROS с использованием Python / 2020 год. - 384 с.

6 Майк МакГирв, Arduino для начинающих: От идеи до реализации / 2019 год. - 352 с.

7 Аrduino для начинающих. Пошаговое руководство [Электронный ресурс]. – Режим доступа : https://all-arduino.ru/arduino-dlya-nachinayushhih. – Загл. с экрана.

8 Pololu Arduino library for VL6180X distance and ambient light sensor [Электронный ресурс]. – Режим доступа: https://github.com/pololu/vl6180x-arduino. – Загл. с экрана.

9 TroykaMQ [Электронный ресурс]. – Режим доступа : https://github.com/amperka/TroykaMQ. – Загл. с экрана.

10 ROS Wiki: Documentation [Электронный ресурс]. – Режим доступа : http://wiki.ros.org/Documentation. – Загл. с экрана.

11 C++ documentation [Электронный ресурс]. – Режим доступа : https://devdocs.io/cpp. – Загл. с экрана

12 Документация C++, C и Assembler [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170. – Загл. с экрана

13 Arduino IDE документация [Электронный ресурс]. – Режим доступа : https://arduino.ru/Hardware/ArduinoBoardMega2560.– Загл. с экрана

14 Visual Studio documentation [Электронный ресурс]. – Режим доступа : https://code.visualstudio.com/docs.– Загл. с экрана

15 Git Hub documentation [Электронный ресурс]. – Режим доступа : https://docs.github.com/ru/codespaces.– Загл. с экрана

# ПРИЛОЖЕНИЕ А Листинг кода программы arduino

#include <Wire.h>

#include <VL6180X.h>

#include <TroykaMQ.h>

#include "ros.h"

#include "std\_msgs/String.h"

#include "std\_msgs/UInt8.h"

#define LOX1\_ADDRESS 0x30

#define LOX2\_ADDRESS 0x31

#define LOX3\_ADDRESS 0x32

#define LOX4\_ADDRESS 0x35

#define LOX5\_ADDRESS 0x34

#define LOX6\_ADDRESS 0x33

#define SHT\_LOX1 2

#define SHT\_LOX2 3

#define SHT\_LOX3 4

#define SHT\_LOX4 7

#define SHT\_LOX5 6

#define SHT\_LOX6 5

#define BUF\_LEN 128

#define PIN\_MQ6 A0

//#define SERIAL\_DEBUG

#define ROS

// Датчики дистанции

// lox1 - forward-left

// lox2 - forward-center

// lox3 - forward-right

// lox4 - back-left

// lox5 - back-center

// lox6 - back-right

VL6180X lox1, lox2, lox3, lox4, lox5, lox6;

// Датчик газа

MQ6 mq6(PIN\_MQ6);

// Моторы

const int motor1b\_dir = 9;

const int motor1a\_pwm = 8;

const int motor2b\_dir = 11;

const int motor2a\_pwm = 10;

// Связь с ROS

// --- Ros Handler Init --- //

ros::NodeHandle NodeHandler;

std\_msgs::String SensorsMessage;

ros::Publisher pub("SensorsData", &SensorsMessage);

// Message stream for ROS

char msg[BUF\_LEN];

String s\_msg;

//===============================================================

// ROS (обработка команд с верхнего уровня)

//===============================================================

void getROSCommands(const std\_msgs::String& msg) {

String commandStr = msg.data;

String speedStr = msg.data;

int command = commandStr.substring(0, 1).toInt();

int ispeed = commandStr.substring(2, commandStr.length()).toInt();

switch (command) {

case 1:

setForwardMoving(ispeed);

break;

case 2:

setBackMoving(ispeed);

break;

case 3:

setRightMoving(ispeed);

break;

case 4:

setLeftMoving(ispeed);

break;

default:

setStopMoving();

break;

}

}

ros::Subscriber<std\_msgs::String> sub("Commands", getROSCommands);

//===============================================================

// Sensors

//===============================================================

void setID() {

// all reset

digitalWrite(SHT\_LOX1, LOW);

digitalWrite(SHT\_LOX2, LOW);

digitalWrite(SHT\_LOX3, LOW);

digitalWrite(SHT\_LOX6, LOW);

digitalWrite(SHT\_LOX5, LOW);

digitalWrite(SHT\_LOX4, LOW);

delay(10);

// all unreset

digitalWrite(SHT\_LOX1, HIGH);

digitalWrite(SHT\_LOX2, HIGH);

digitalWrite(SHT\_LOX3, HIGH);

digitalWrite(SHT\_LOX6, HIGH);

digitalWrite(SHT\_LOX5, HIGH);

digitalWrite(SHT\_LOX4, HIGH);

delay(10);

// activating LOX1 and reseting LOX2

digitalWrite(SHT\_LOX1, HIGH);

digitalWrite(SHT\_LOX2, LOW);

digitalWrite(SHT\_LOX3, LOW);

digitalWrite(SHT\_LOX6, LOW);

digitalWrite(SHT\_LOX5, LOW);

digitalWrite(SHT\_LOX4, LOW);

delay(10);

lox1.init();

lox1.configureDefault();

lox1.setScaling(2);

lox1.setAddress(LOX1\_ADDRESS);

digitalWrite(SHT\_LOX2, HIGH);

delay(10);

lox2.init();

lox2.configureDefault();

lox2.setScaling(2);

lox2.setAddress(LOX2\_ADDRESS);

digitalWrite(SHT\_LOX3, HIGH);

delay(10);

lox3.init();

lox3.configureDefault();

lox3.setScaling(2);

lox3.setAddress(LOX3\_ADDRESS);

digitalWrite(SHT\_LOX4, HIGH);

delay(10);

lox4.init();

lox4.configureDefault();

lox4.setScaling(2);

lox4.setAddress(LOX4\_ADDRESS);

digitalWrite(SHT\_LOX5, HIGH);

delay(10);

lox5.init();

lox5.configureDefault();

lox5.setScaling(2);

lox5.setAddress(LOX5\_ADDRESS);

delay(10);

digitalWrite(SHT\_LOX6, HIGH);

lox6.init();

lox6.configureDefault();

lox6.setScaling(2);

lox6.setAddress(LOX6\_ADDRESS);

}

void readSensors() {

s\_msg = "";

s\_msg.concat(lox1.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(lox2.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(lox3.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(lox4.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(lox5.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(lox6.readRangeSingleMillimeters());

s\_msg.concat(";");

s\_msg.concat(mq6.readLPG());

}

//===============================================================

// Motors

//===============================================================

void setForwardMoving() {

digitalWrite(motor1b\_dir, HIGH);

analogWrite(motor1a\_pwm, 1);

digitalWrite(motor2b\_dir, HIGH);

analogWrite(motor2a\_pwm, 1);

}

void setBackMoving() {

digitalWrite(motor1b\_dir, LOW);

analogWrite(motor1a\_pwm, 255);

digitalWrite(motor2b\_dir, LOW);

analogWrite(motor2a\_pwm, 255);

}

void setLeftMoving() {

digitalWrite(motor1b\_dir, HIGH);

analogWrite(motor1a\_pwm, 1);

digitalWrite(motor2b\_dir, LOW);

analogWrite(motor2a\_pwm, 1);

}

void setRightMoving() {

digitalWrite(motor1b\_dir, LOW);

analogWrite(motor1a\_pwm, 1);

digitalWrite(motor2b\_dir, HIGH);

analogWrite(motor2a\_pwm, 1);

}

void setForwardMoving(int ispeed) {

digitalWrite(motor1b\_dir, HIGH);

analogWrite(motor1a\_pwm, ispeed);

digitalWrite(motor2b\_dir, HIGH);

analogWrite(motor2a\_pwm, ispeed);

}

void setBackMoving(int ispeed) {

digitalWrite(motor1b\_dir, LOW);

analogWrite(motor1a\_pwm, ispeed);

digitalWrite(motor2b\_dir, LOW);

analogWrite(motor2a\_pwm, ispeed);

}

void setLeftMoving(int ispeed) {

digitalWrite(motor1b\_dir, HIGH);

analogWrite(motor1a\_pwm, ispeed);

digitalWrite(motor2b\_dir, LOW);

analogWrite(motor2a\_pwm, 1);

}

void setRightMoving(int ispeed) {

digitalWrite(motor1b\_dir, LOW);

analogWrite(motor1a\_pwm, 1);

digitalWrite(motor2b\_dir, HIGH);

analogWrite(motor2a\_pwm, ispeed);

}

void setStopMoving() {

digitalWrite(motor1b\_dir, LOW);

analogWrite(motor1a\_pwm, 1);

digitalWrite(motor2b\_dir, LOW);

analogWrite(motor2a\_pwm, 1);

}

//===============================================================

// Setup

//===============================================================

void setup() {

Serial.begin(115200);

Wire.begin();

pinMode(SHT\_LOX1, OUTPUT);

pinMode(SHT\_LOX2, OUTPUT);

pinMode(SHT\_LOX3, OUTPUT);

pinMode(SHT\_LOX6, OUTPUT);

pinMode(SHT\_LOX5, OUTPUT);

pinMode(SHT\_LOX4, OUTPUT);

pinMode(motor1b\_dir, OUTPUT);

pinMode(motor1a\_pwm, OUTPUT);

pinMode(motor2b\_dir, OUTPUT);

pinMode(motor2a\_pwm, OUTPUT);

setID();

delay(60000);

mq6.calibrate();

// Init ROS Nodes

#ifdef ROS

NodeHandler.getHardware()->setBaud(115200);

NodeHandler.initNode();

NodeHandler.advertise(pub);

NodeHandler.subscribe(sub);

#endif

}

//===============================================================

// Loop

//===============================================================

void loop() {

readSensors();

memset(msg, '\0', sizeof(msg));

s\_msg.toCharArray(msg, BUF\_LEN);

SensorsMessage.data = msg;

#ifdef SERIAL\_DEBUG

Serial.println(SensorsMessage.data);

#endif

#ifdef ROS

pub.publish(&SensorsMessage);

NodeHandler.spinOnce();

#endif

delay(500);

}

# ПРИЛОЖЕНИЕ Б Листинг кода программы pysensors\_ data

#!/usr/bin/env python3

import rospy, math

from std\_msgs.msg import String

from geometry\_msgs.msg import Twist

from sensor\_msgs.msg import LaserScan

# 12

# 13 11

# 14 ^ 10

# 15 / \ 9

# 16 | 8

# 17 | 7

# 18 | 6

# 19 5

# 20 4

# 21 3

# 22 2

# 23 1

# 24

# Место для потенциальных глобальных значений

# ...

# Основной класс программы

class KeyboardTeleop:

def \_\_init\_\_(self):

rospy.init\_node('keyboard\_teleop')

self.pub = rospy.Publisher('/cmd\_vel', Twist, queue\_size=10)

self.sub\_front = rospy.Subscriber('SensorsData', String, self.onIRSensor)

# Добавьте здесь инициализацию переменных для значений датчиков с нижнего уровня

# ...

self.sub\_lidar = rospy.Subscriber('/scan', LaserScan, self.onLaserSensor)

# Список под секции лидара

self.distances = []

self.lidar = LaserScan.angle\_max

for i in range(SEGMENTS):

self.distances.append(0)

# Переменные для записи скорости и направления

self.target\_linear\_vel = 0.0

self.target\_angular\_vel = 0.0

# Запуск алгоритма

def run(self):

rate = rospy.Rate(10)

try:

# Цикл работы программы, его наполнение может быть полностью изменено, кроме команды rate.sleep()

while not rospy.is\_shutdown():

self.update\_velocity()

# ...

rate.sleep()

except rospy.ROSInterruptException:

pass

# Вызывается при получении данных датчиков с нижнего уровня

def onIRSensor(self, msg):

print("placeholder")

# Вызывается при получении данных с лидара (можно экспериментировать с наполнением)

def onLaserSensor(self, msg):

for i in range(SEGMENTS):

self.distances[i] = 9999

angle = msg.angle\_min

for r in msg.ranges:

if math.isinf(r):

r = 12

a = angle \* 180 / math.pi

idx = int(a / (360 / SEGMENTS)) % SEGMENTS

self.distances[idx] = min(self.distances[idx], r)

angle = angle + msg.angle\_increment

def update\_velocity(self):

# Место для вашего кода вычисления направления и скоростей

# ...

# Вывод всех сегментов с лидара для отладки и тестов

for i in range(SEGMENTS):

print(f"Seg {i} - {self.distances[i]}")

twist = Twist()

rate = rospy.Rate(10)

twist.linear.x = """ Ваше выбранное значение """; twist.linear.y = 0.0; twist.linear.z = 0.0

twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z = """ Ваше выбранное значение """

self.pub.publish(twist) # Данная строчка отсылает данные движения на нижний уровень в тему "cmd\_vel"

rate.sleep()

if \_\_name\_\_ == "\_\_main\_\_":

teleop = KeyboardTeleop()

teleop.run()