Содержание

Введение 7

О компании 7

1 Цель работы 9

2 Функционал робота 10

3 Программные решения. ПО платформы 11

3.1 Теория о ROS 11

3.2 Теория о MQTT 13

3.3 Опыт взаимодействия с ROS 15

3.3.1 Ознакомительный процесс 15

3.3.2 Практический результат 16

Заключение 19

Перечень использованных информационных ресурсов 20

Введение

На преддипломную практику мною была поставлена задача, разработки программного обеспечения для четырехколесной робототехнической платформы. А именно поиск подходящих робототехнических решений, с целью оптимизации их под конкретную платформу, имеющую стандартный функционал мобильных роботов, выполняющих локализацию себя в пространстве и картопостроение, за счёт собранных данных. Особенность заключается в условиях работы платформы, а именно на реабилитационных территориях с повышенным уровнем радиационного фона, требующим дополнительный класс защиты, а также наличие инструмента построения карты радиоактивного излучения.

О компании

В качестве места прохождения преддипломной практики мною было выбрана компания «Бастион». «**Бастион**» — российская научно-производственная **компания**, осуществляющая с 1991 г., изображенная на рисунке 1. Основной вектор деятельности компании направлен на производство электрооборудования. Но также компания имеет опыт создания различных систем, мобильных приложений и программного обеспечения под уникальные аппаратные решения.

Рисунок 1 – Компания Бастион

От компании «Бастион» входе практики были предоставлены технологические решения, позволяющие процесс разработки продукта, независимо от вида деятельности. Начиная от производства корпусных элементов, путем гибки материала, 3D-печатью пластиковых комплектующих, производства собственных печатных плат, экспертностью в разработке ПО, заканчивая наличием огромной компонентной базы, значительно ускоряющей процесс разработки.

1 Цель работы

Условия работы робототехнического комплекса подразумевают агрессивную среду, в которой возникает большой ряд сложностей, связанный с защитой электроники от быстрого износа, под действием радиационного фона, защитой корпуса от радиационного загрязнения, а также защитой каналов связи от наводящихся помех. Входе выполнения работ я не буду учитывать контекст агрессивной среды, а лишь буду анализировать существующие решения с целью запуска платформы. Описывая основные шаги, которые мне удалось пройти, подкрепляя данные ссылками на официальные источники.

2 Функционал робота

Прежде чем говорить о разработке программного обеспечения и анализе существующих решений, стоит коротко описать аппаратную часть робототехнической платформы. Робот представляет собой полноприводную четырехколесную платформу, актуаторамии которой являются коллекторные электродвигатели. Обратная связь от моторов фиксируют два инкрементальных энкодера, соединённым при помощи зубчатой ременной передачи с валом колеса. Энкодеры расположены противолежащих углах корпуса платформы. Также в роботе имеются ультразвуковые датчики расстояния, детектор радиоактивных частиц, а также 2D-лидар и веб-камера. Внешний вид робота представлен на рисунке 2.

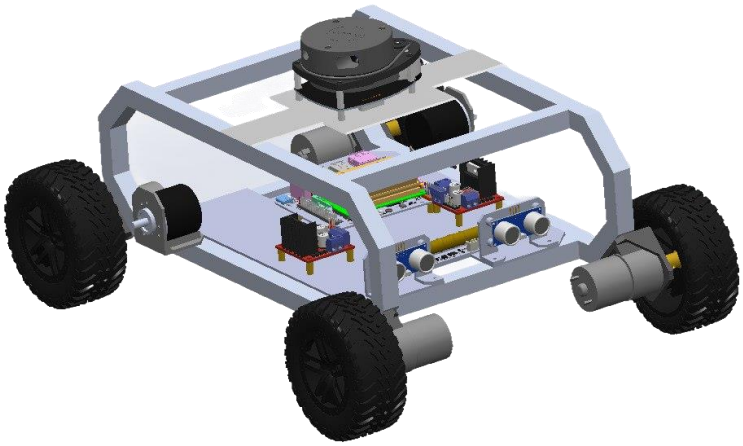

Рисунок 2 – Внешний вид робота

Обработка данных с датчиков происходит на 32-битном контроллере STM32F103. После данные с МК передаются на микрокомпьютер Raspberry Pi4.

3 Программные решения. ПО платформы

Существует большое кол-во софта для запуска тех или иных аппаратных систем, большое кол-во затрудняет выбор, по причине не понимая всех недостатков и преимуществ. Кроме того, большую роль в использовании софта играет его документация, учебные материалы по использованию, а также аналогичные проекты, реализованные с похожими характеристиками. И я, как и любой другой новичок-разработчик не в силах сразу понять какой инструмент, когда лучше применить, и в принципе как его использовать, в той или иной ситуации.

Используемым мною софтом или же программным обеспечением, выбранным для разработки ПО робототехнической платформы, является мета-операционная система ROS. Исходя из понимая философии ROS, изложенной на официальном сайте.

3.1 Теория о ROS

Операционная система робота представляет собой набор программных библиотек и инструментов, которые упрощают процесс создания приложений для роботов. От драйверов до самых современных технологий алгоритмов и мощных инструментов разработчика. И ключевая особенность ROS, это открытый исходный код [1].

Немного стоит упомянуть о концепции ROS [2]. Первый уровень концепции, **файловая система ROS** – основная задача, которой организация файлов на диске. Так основной единицей в рамках программного фреймворка ROS, является **пакет**. Пакет содержит в себе исполняемые файлы, библиотеки и т.п. Далее пакет имеет **описание,** в котором находится все основная информация о пакете. После идут **типы сообщений**, которые хранятся в отдельной папке с расширением *.msg.*. **Типы служб,** также находятся в папке с специальным расширением *.srv.* и такие файлы определяют запрос и ответ для структуры данных в сервисах ROS.

Второй уровень концепции, **вычислительный граф ROS** – это одноранговая сеть систем ROS, назначение которой – обработка всех данных. Основными понятиями вычислительной графики ROS являются узлы, Мастер ROS, сервер

параметров, сообщения и службы. На приведенном ниже рисунке 3, отображена схема, иллюстрирующая структуру связи в узле ROS1.

Рисунок 3 – Структура связи ROS

Такая структура общения позволяет функционирующей системе:

1. Сохранять **надежность**, если один узел выходит из строя, это не влияет на всю систему, по той причине, что другие узлы могут функционировать.
2. **Эффективно** общаться между узлами системы отправляя данные лишь когда это нужно, что снижает ненужный трафик в сети.
3. Использование **масштабируемости**, система может быть легко масштабирована с добавлением новых узлов, которые могут публиковать или подписываться на новые темы.
4. Обладать **гибкостью,** модель публикации-подписки и клиент-сервер обеспечивает гибкость в обмене данными, позволяя узлам быть издателями или подписчиками информации в зависимости от их ролей.
5. Обладать **модульностью,** т.е. использование узлов позволяет разрабатывать системы модульно, где каждый узел отвечает за отдельную задачу.

Стоит отметить, что схема, отраженная на рисунке 3, частично применима для ROS2, хоть он и является улучшенной версией ROS, но лично мне оно дает понимание

передачи сообщений между узлами, да и в принципе мне, как начинающему разработчику схема более понятна для восприятия. Также структура отражает похожий инструмент работы и взаимодействия MQTT, работающий на созвучном сетевом протоколе MQTT, чуть позже немного опишу принцип работы, который позволяет лучше понять концепцию ROS.

И заключительный уровень концепции, это уровень общения ROS. **Сообщество ROS** состоит из разработчиков и исследователей, создающих и поддерживающих пакеты ROS. Они обмениваются между собой информацией о существующих и недавно созданных пакетах и другими новостями, связанными со структурой ROS.

Собрав воедино все достоинства ROS, мы имеем неплохую экосистему поддерживаемую большим комьюнити, имея большую базу знаний и библиотеки контроллеров, алгоритмов и т.д., что делает использование максимально простым и понятным.

Но стоит отметить, что опыт моего использования операционной системы для роботов не увенчался большим успехом. Камнем преткновения у меня служит долгий поиск подходящих репозиториев с готовыми программными решениями, подходящими подмою конкретную задачу, так как из большого разнообразия выбирать трудно. Возникает эта проблема на фоне низкой подготовленности, меня как специалиста.

Выше я упоминал о MQTT, в моем случае протокол дал мне понимание и объяснение работы с основными инструментами мета-операционной системы для роботов. Принцип работы MQTT так же заключается в формате «издатель-подписчик»

3.2 Теория о MQTT

MQTT — это протокол публикации/подписки, предназначенный для подключения устройств Интернета вещей [3]. В отличие от парадигмы запроса/ответа HTTP, MQTT работает на основе событий, позволяя отправлять сообщения клиентам.

Этот архитектурный подход обеспечивает высокую масштабируемость решений за счет разделения производителей и потребителей данных, устраняя зависимости между ними. Двумя ключевыми компонентами для установки соединения MQTT для публикации и подписки сообщений являются **клиенты MQTT** и **брокер MQTT** , как показано на рисунке 4 ниже:

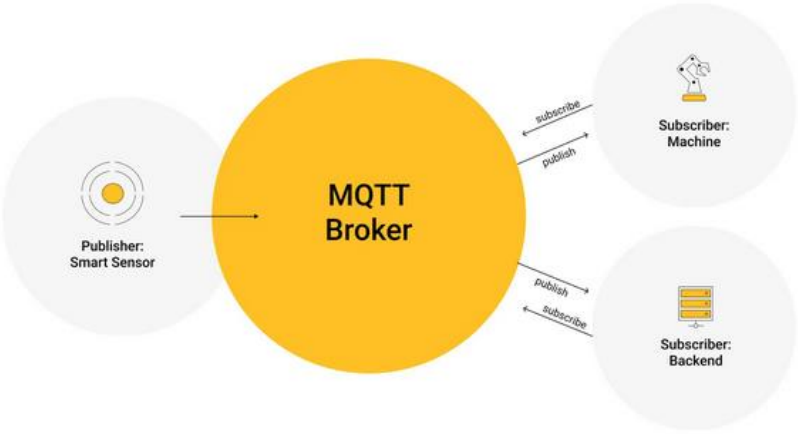

Рисунок 4 – Архитектура публикации/подписки MQTT

Возникает вопрос, а кто есть кто и как их отличать, да и в принципе в чем отличается их функционал. В основе MQTT лежат брокеры MQTT и клиенты MQTT [3]. Брокер MQTT является посредником между отправителями и получателями, отправляя сообщения соответствующим получателям. Клиенты MQTT публикуют сообщения брокеру, а другие клиенты подписываются на определенные темы для получения сообщений. Каждое сообщение MQTT включает тему, и клиенты подписываются на интересующие их темы. Брокер MQTT поддерживает список подписчиков и использует его для доставки сообщений соответствующим клиентам.

Познакомившись с софтом этих двух инструментов, прослеживается связь между устройством работы, а точнее форматом общения между двумя и более устройствами/узлами путем использования архитектуры издатель подписчик, что в свою очередь значительно уменьшает кол-во используемого трафика, повышает надежность системы, увеличивает степень автономности систем с использованием выше описанных технологий.

3.3 Опыт взаимодействия с ROS

Во время прохождения практики я взаимодействовал с ROS2 на примере базового пакета turtlesim [4]. Я пытался разобраться в структуре работы основных инструментов таких как: топики, узлы, сервисы и клиенты, а также в структуре работы топиков и узлов.

3.3.1 Ознакомительный процесс

На рисунке 5 изображен процесс запуска пакета turtlesim и получение данных о типах сообщений в темах публикуемых, узлами пакета.

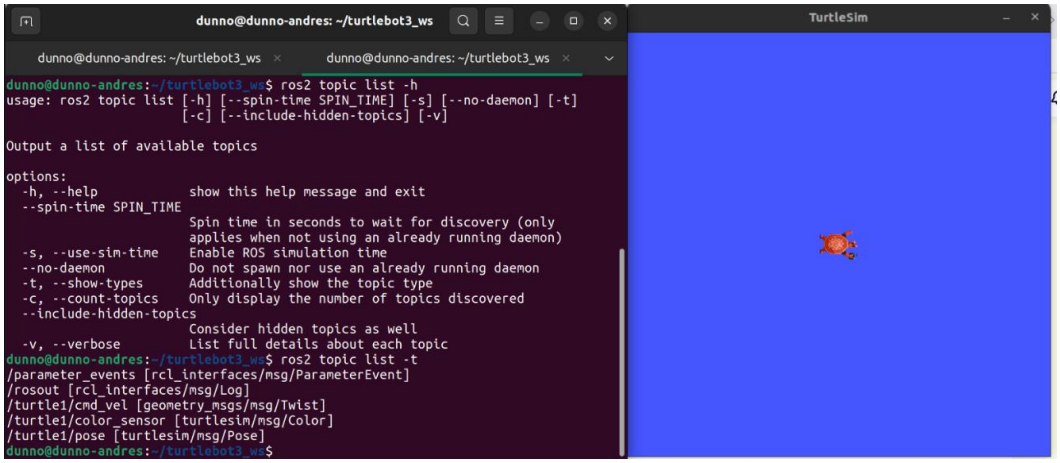

Рисунок 5 – Демонстрация работы пакета turtlesim

Также на рисунке 6 представлена структурная схема работы пакета отражает как различные компоненты ROS взаимодействуют друг с другом. И можем наблюдать, что управляющие данные публикуются в общий топик процесса (/turtle1), после чего используются нодой (turtlesim) для изменения местоположения.

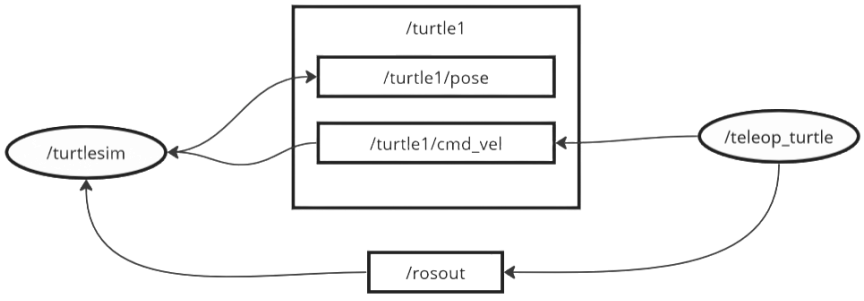

Рисунок 6 – Структурная схема общения пакета

3.3.2 Практический результат

Для более детального понимания процесса разработки ПО на рисунке 7 представлена примерная структурная схема блоков, имеющихся в роботе, как в физическом уровне, так и в цифровом. Более подробно о схеме и процессе ее воспроизведения на физической модели. У нас имеется три вычислительных устройства предназначенных для обработки различных процессов, разделяющихся между собой по виду деятельности, начиная с STM32, устройства производят различного рода вычисления. На уровне STM32 вычисления сводятся к обработке сигналов с датчиков, генерации управляющих сигналов и пересылкой их на другое устройство.

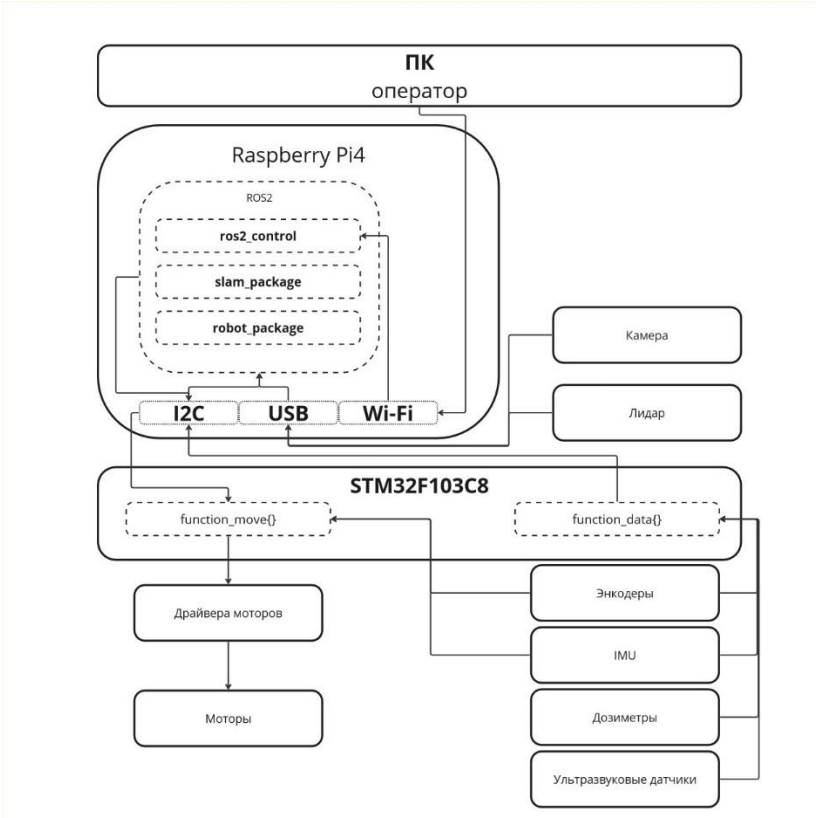

Рисунок 7 – Схема структурных блоков

Обработанные данные принимает микрокомпьютер Raspberry Pi4, на нем уже стоит ответственность за решение поставленной задачи с user-ПК, а также отправкой операционных команд на микроконтроллер.

В качестве микроконтроллера на время практики, использовался контроллер семейства AVR с чипом ATmega328P и чтобы запустить мобильную четырех

колесную платформу буду использовать лишь 2 драйвера мотора и 2 энкодера, дополнительную обвязку добавлю позже. Среда разработки Arduino IDE, почему не stm32, низко-профильность меня как разработчика не позволяет быстро выполнять прошивку контроллеров и требует от меня дополнительных знаний и навыков, которых я при себе пока не имею, но постепенно осваиваю.

В качестве удобного подключения к микрокомпьютеру я буду использовать SSH – сетевой криптографический протокол, позволяющий пользователям удаленно подключаться к различным серверам, обеспечивая безопасность соединения и простоту в использовании, также позволяющий создавать ssh-тунели, для подключения независимо от местоположения сервера и клиента.

Также для визуализации работы с ROS, я настроил GUI по SSH. Для этого я использовал X-сервер на Windows для управления портами локального дисплея. После подключения к серверу мне достаточно, обратиться из сервера по IP к локальной машине, указав порт публикации потока дисплея, команда (export DISPLAY=192.168.31.191:0.0) отображена на 8 рисунке.

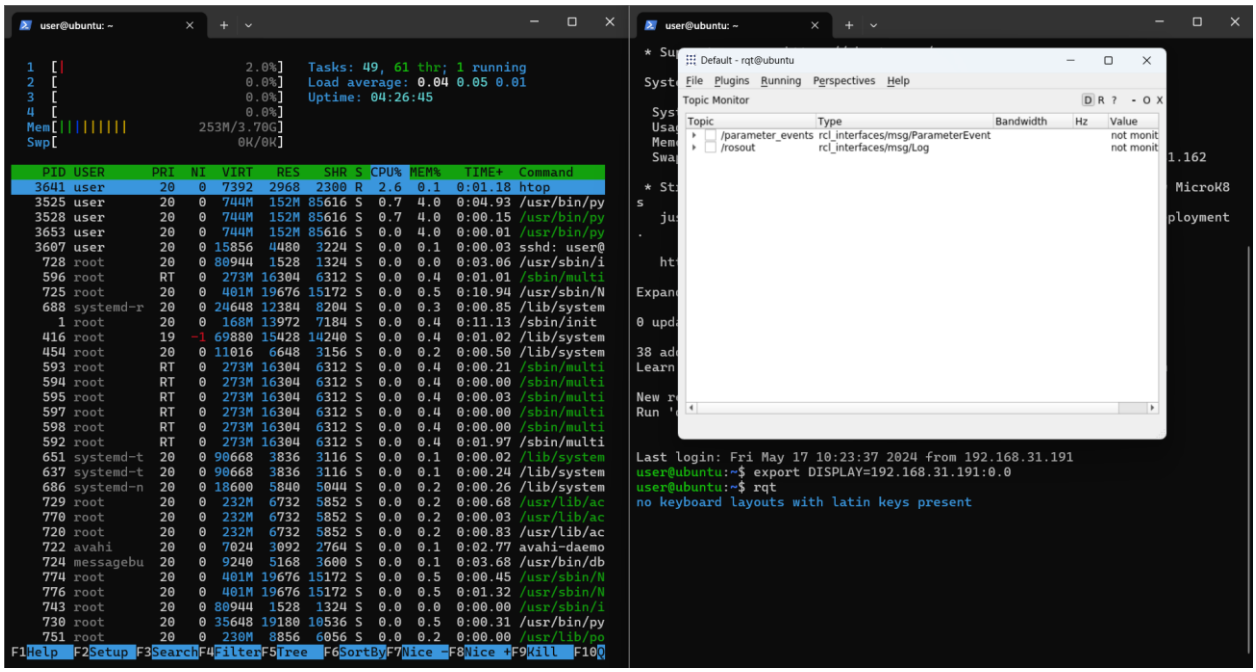

Рисунок 8 – Пример использования GUI on SSH

Также исходя из монитора задач мы видим, что этот процесс не сильно нагружает систему, если использовать его в обычных целях, но соединение конечно

не подходит для просмотра видеороликов на YouTube. Стоит отметить что присутствует некая задержка в скорости отклика.

Для лучшего понимания устройства работы с ROS, начал следовать туториалам, находящихся на просторах сети. Я видел различные учебные ресурсы, позволяющие познакомиться с ROS, но они были слишком удалены от действительности работы с железом. Мне попался иностранный блогер, ведущий активную деятельность области робототехники. У него мне удалось найти приемлемое объяснение и разжевывание процессов, необходимых для робототехники [5].

По итогам следования деятельности мне удалось удаленно управлять моторами, задавать ШИМ-сигнал, обрабатывать данные с энкодеров и наконец-то пощупать своими руками ROS на железе. На рисунке 9 отображена структурная схема физических элементов, благодаря ей, я шаг за шагом осознаю удобность применения ROS

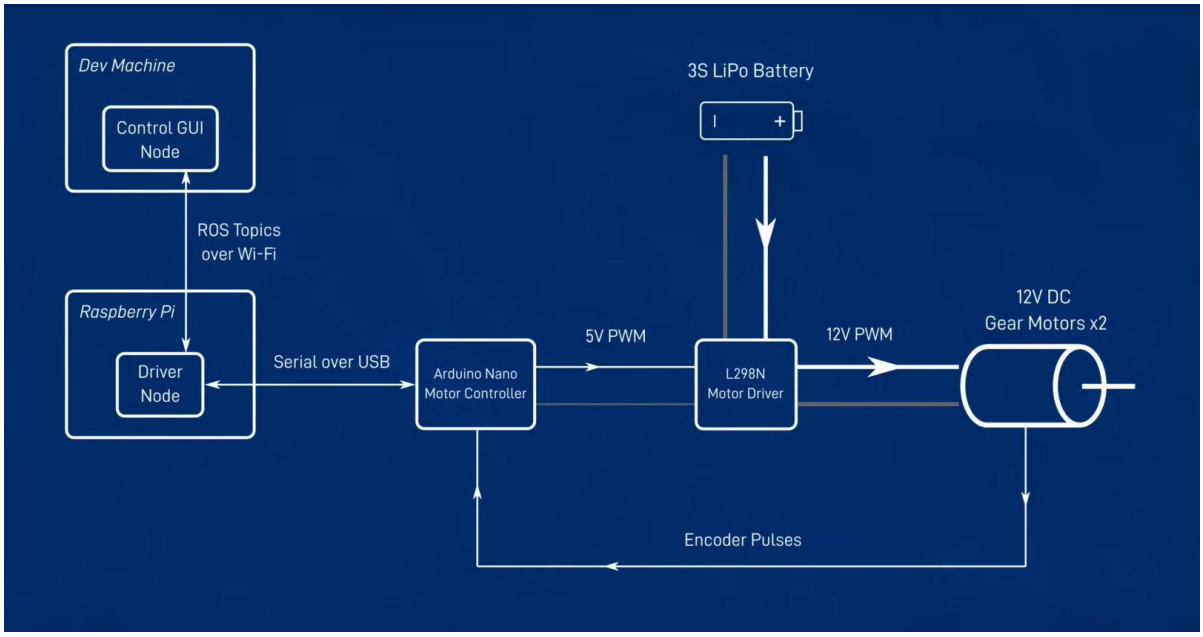

Рисунок 9 – Структурная схема из гайда

Дальше мой прогресс пока не продвинулся, следую материалам и пытаюсь переложить программное решение на свою платформу.

Заключение

Входе прохождения практики в дочерней компании Бастиона "Интерстеллар софт" у меня осталось большое кол-во вопросов к себе как будущему специалисту, были выявлены крупные бреши в моей специальности разработчика. Теперь буду закрывать пробелы всеми возможными способами, увеличивать уровень своего профессионализма в сфере робототехники. Мне удалось познакомится с протоколом связи MQTT, инструмент для работы с Iot, с возможностью асинхронного программирования микроконтроллеров, путем распределение задач внутри различных прерываний, а также я улучшил свои познания в области проектирования печатных плат и схемотехники. Я выполнил свою задачу на 1/3 если не больше, есть над чем работать и к чему стремиться.

Перечень использованных информационных ресурсов

1. ROS домашняя страница [Электронный ресурс] – URL: https://www.ros.org/ (Дата обращения 14.05.2024).
2. Джозеф Л. Изучение робототехники с помощью Python / пер. с анг. А. В. Корягина. — Оформление, издание, перевод, ДМК Пресс, 2019. — Сведения объеме (250 с.). — ISBN (978-5-97060-749-7).
3. Домашняя страница знакомства с MQTT [Электронный ресурс] – URL: https://www.hivemq.com/ (Дата обращения 10.05.2024)
4. Документация ROS2. Учебные пособия [Электронный ресурс] – URL: https://docs.ros.org/en/foxy/Tutorials.html (Дата обращения 20.04.2024)
5. Учебный материал по созданию робота с использованием ROS [Электронный ресурс] – URL: https://articulatedrobotics.xyz/tutorials/ready-for-ros/what-you-need-for-ros (Дата обращения 12.05.2024)