Титульный листЗадание

РЕФЕРАТ

Разработка аппаратно-программного комплекса для внедрения ROS2

Руководитель ВКР – доцент, кандидат технических наук завидущий кафедрой Захаров М. В.

Выпускная квалификационная работа объемом ?? с. Содержит ?? рисунков, ?? источников и ?? приложений.

Ключевые слова: платформа, разработка, Raspberry Pi, STM32, ROS2, 3D печать, обучение, Linux.

Цель работы – разработать и собрать многофункциональную платформу для изучения ROS2.

Структура ВКР: состоит из введения, четырех глав, списка использованных источников, приложений.

В первой главе анализируется необходимость разработки платформы для изучения ROS2, рассматриваются готовые проекты и разрабатывается техническое задание на разработку платформы.

Во второй главе осуществляется проектирование платформы, и ее сборка. Описываются аппаратные элементы, принцип их выбора и методы работы с ними.

В третьей главе описывается разработка программного обеспечения платформы.

В четвертой главе представляются примеры лабораторных работ по изучению платформы ROS2, а также предоставляются методические указания по использованию этой платформы и работе с ее элементами.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| « \_ » 2022г. |  |  | А.Н. Крайников |
|  | (подпись) |  | (инициалы, фамилия) |

ОГЛАВЛЕНИЕ

[Нормативные ссылки 5](#_Toc104721821)

[Определения, обозначения и сокращения 6](#_Toc104721822)

[Введение 7](#_Toc104721823)

[1 Аналитиз проблемы 9](#_Toc104721824)

[1.1 Актуальность проблемы 9](#_Toc104721825)

[1.2 Постановка проблемы 10](#_Toc104721826)

[1.3 Возможные пути решения 11](#_Toc104721827)

[1.4 Описание пути решения 14](#_Toc104721828)

[2 Проектирование и сборка аппаратной части 16](#_Toc104721829)

[2.1 Разработка принципиальной схемы элементов 16](#_Toc104721830)

[2.2 Моделирование корпуса 16](#_Toc104721831)

[2.3 Изготовление манипулятора 16](#_Toc104721832)

[2.4 Отладочная плата STM32F1 16](#_Toc104721833)

[2.5 Микрокомпьютер RPI 4B 16](#_Toc104721834)

[2.6 Разработка панели управления 16](#_Toc104721835)

[3 Разработка программной части 17](#_Toc104721836)

[3.1 Программа для передвижения робота 17](#_Toc104721837)

[3.2 Программа для управления манипулятором 17](#_Toc104721838)

[3.3 Программа для панели управления 17](#_Toc104721839)

[3.4 Программа для интернет интерфейса 17](#_Toc104721840)

[3.5 Программа для автономной навигации 17](#_Toc104721841)

[4 Применение платформы в орбразовательном процессе 18](#_Toc104721842)

[4.1 Примеры лабораторных работ 18](#_Toc104721843)

[4.2 Методические указания к использованию робота 18](#_Toc104721844)

[Заключение 19](#_Toc104721845)

[Список использованных источников 20](#_Toc104721846)

# Нормативные ссылки

# Определения, обозначения и сокращения

ПО – программное обеспечение.

ПЛК – программируемый логический контроллер.

UART – универсальный асинхронный приёмопередатчик.

USART - универсальный синхронно/асинхронный приёмопередатчик.

I2C – последовательная асимметричная шина

SWD – интерфейс последовательной проводной отладки

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

ROS2 – фреймворк для программирования робототехники

# Введение

(Обоснование темы, актуальность) В данной работе рассматривается тема разработки аппаратно-программного комплекса для внедрения «ROS2». ROS2 – это фреймворк для программирования в сфере робототехники, который значительно упрощает процесс программирования на всех его этапах от начала работки до поддержки итогового продукта и его развития. ROS2 также является усовершенствованной версией фреймворка ROS. Оба фреймворка могут применяться на идентичных аппаратных базах и различаются лишь инструментальной базой программного уровня.

Причиной рассмотрения данной темы, иначе говоря – проблемой, можно выделить: низкий уровень обучения студентов соответствующих направлений инновациям в области робототехники и автоматизации.

Актуальность данной проблемы основывается на предполагаемых последствиях. Одним из наиболее важных последствий можно выделить: снижение уровня актуальности знаний и умений выпускников соответствующих направлений подготовки, что в свою очередь может привести к снижению качества выпускаемой продукции, производимой с помощью автоматизации технологических процессов, снижению безопасности на производстве и снижению скорости развития соответствующих сфер.

(Цель и Задачи) За цель можно определить: разработать аппаратно-программный комплекс для изучения фреймворка ROS2.

Задачи можно выделить следующие:

- наличие манипулятора, имитирующего промышленные аналоги;

- возможность изучения и изменения аппаратной части платформы;

- возможность внедрения ROS и его изучения;

- доступность элементарной базы, для ремонта и усовершенствования;

- сравнительно низкая стоимость.

(Оценка современного состояния решаемо задачи) Существуют различные способы решения данной проблемы, но общим для них является то, что необходимо усовершенствование процесса обучения студентов, обучающихся на соответствующих специальностях, путем внедрения теории и практики работы с системой ROS2.

Проанализировав готовые решения, можно сделать вывод, что на рынке не представлены или представлены в малом количестве решения, которые удовлетворяют нашей проблематике. Однако мы имеем возможность самостоятельно собрать платформу, подходящую под нашу проблематику.

(Основание и исходные данные) За исходные данные при разработке можно принять документацию на схожие проекты или используемые проекты. Основанием для данной темы будем считать условно низкий процент инновационных дипломных проектов.

(Методы и средства) Наиболее действенным методом решения поставленной проблемы, будет внедрение в учебный процесс дисциплины или тем, в уже используемые дисциплины, связанные с новейшими технологии в области робототехники и автоматизации. Для проведения подобных занятий будет необходима платформа для практического применения полученных на теории навыков.

(Ожидаемые результаты) Среди ожидаемых результатов можно выделить большую вовлеченность студентов в использование современных технологий автоматизации и получение платформы для изучения фреймворка ROS2 и сопутствующих

## 1 Аналитиз проблемы

### 1.1 Актуальность проблемы

На данный момент «перспективы индустрии робототехники в мире оптимистичны» о чем сказал Милтон Гэррии в свое обращении от 17 июня 2021 года. Эти перспективы также передаются на область промышленной робототехники в России. Например, по итогам 2019 года в России было продано на 40% больше промышленных роботов по сравнению с 2018 годом, подобные темпы развития сохраняются и по нынешнее время.

Одна из причин развития промышленной робототехники заключается в том, что роботизированное и автоматизированное производство обладает большей надежностью, по отношению к ручному труду. Выгодность вложений в промышленную робототехнику показала пандемия 2020 года, по итогам которой большинство «робокомпаний» (68%) получили большую выручку чем на год ранее.

Вследствие актуальности развития промышленной робототехники в мире наблюдается рост проектов, связанных роботизацией в промышленной сфере. Например, юго-западный научно-исследовательский институт и команда «ROS-Industrial» разработали устройство, позволяющее промышленным роботам сканировать и манипулировать металлическими объектами. В основе программного кода данного проекта используется фрейворк ROS2.

На основе поставленной проблемы можно предполагать о возможных последствиях, если не уделять должное внимание данной проблеме.

Одним из возможных последствий является спад скорости развития сферы промышленной автоматизации. По оценкам всемирного экономического форума в 2018 году порядка 29% всех рабочих часов приходится на роботов, к 2025 году эта доля превысит половину. Уже сейчас роботы выполняют 31% всех работ, связанных с физическим трудом. На данный момент по информации «World Robotics 2019» в среднем в мире на 10 000 рабочих промышленной сферы приходится 99 роботов в то время, как в России на 10 000 работников всего 5 роботов. Это нам показывает не только спад скорости развития автоматизации производств в России относительно других стран, но и показывает сферу, в которой Россия нуждается в инновациях.

Используя полученные данные, можно смело предположить, что главный тренд автоматизации – это роботизация.

### 1.2 Постановка проблемы

Использование фреймворка ROS 2 позволяет упростить процесс роботизации, а соответственно и автоматизации, в производстве, благодаря своей архитектуре. Также, как показала нам пандемия, очень важным качеством для автоматизированного производства является адаптивность и масштабирование самого производства. Благодаря распределенной структуре кода ROS2 позволяет упростить видоизменение производства, что влияет на его адаптивность под спрос продукции.

Однако на данный момент во многих учебных планах направлений автоматизация технологических процессов и производств не предусмотрено изучение основ робототехники и фреймворка ROS2, как одного из самых популярных в данной сфере. Из чего можно сделать вывод, что выпускники данного и подобных направлений обладают сниженным уровнем знаний об инновациях и не имеют навыков работы с ними

Основываясь на перечисленных выше факторах и факта моего обучения по направлению автоматизации технологических процессов и производств была поставлена проблема низкого уровня обучения студентов соответствующих направлений инновациям в области робототехники и автоматизации.

В данном случае мы будем говорим об узконаправленном решении проблемы не преподавания инновационных технологий в сфере промышленности. Мы будем рассматривать инновации в промышленности именно на примере роботизации промышленности, как на одной из наиболее актуальных темы.

### 1.3 Возможные пути решения

Выделенная проблема может быть решена несколькими путями. Если учитывать, что полноценно изменить учебный процесс является достаточно сложной задачей, можно дополнить учебный процесс новыми дисциплинами, связанными с робототехникой и роботизацией производства. Данный путь решения имеет возможность повлиять на нашу проблематику в лучшую сторону, однако он может увеличить нагрузку на студентов, что в результате может привести к худшему уровню обучения другим предметам. Поэтому данный путь не подходит.

Намного более выгодным путем решения будет усовершенствование уже преподаваемых дисциплин, иначе говоря, их актуализация. Примерами данных дисциплин являются дисциплины, связанные с программированием микроконтроллеров и микрокомпьютеров. В рамках данных дисциплин можно рассматривать различные фреймворки, операционные системы (ядра) и паттерны программирования, применяющиеся в робототехнике. В рамках дисциплин связанных с использованием программируемых логических контроллеров есть возможность рассматривать их применение на примере роботов-манипуляторов, которые наиболее активно используются при роботизации производств. При изучении аппаратной части автоматизированных систем можно уделять больше внимания роботизированным системам, как подвиду автоматизированных систем, в том числе использованию пневмоприводом и электроприводов не как исполнительных органов, реагирующих на изменение показаний датчиков с помощью регуляторов, а как органов, реагирующих на изменение среды через более сложные датчики (камеры и лидары) и через более сложные с программной стороны системы регулирования.

Далее нам необходимо уточнить путь нашего решения, а именно то, на основе какой элементной базы будет происходить обучение данным дисциплинам. В данном случае принимается за аксиому то, что корректировка дисциплин приведет к необходимости изменение аппаратной базы, а также то, что лучшим решением является корректировка как теоретической части дисциплин, так и практической.

Давайте рассмотрим наборы для изучения фрейворка ROS2, тем самым конкретизировав нашу тему на внедрение изучения фремворка ROS2 в процесс обучения, как тему, решающую поставленную нами более широкую проблему.

Одним из самых популярных наборов для изучения ROS в России является набор «TurtleBro». Стоимость и комплектацию данного набора можно наблюдать на рисунке 1.1.

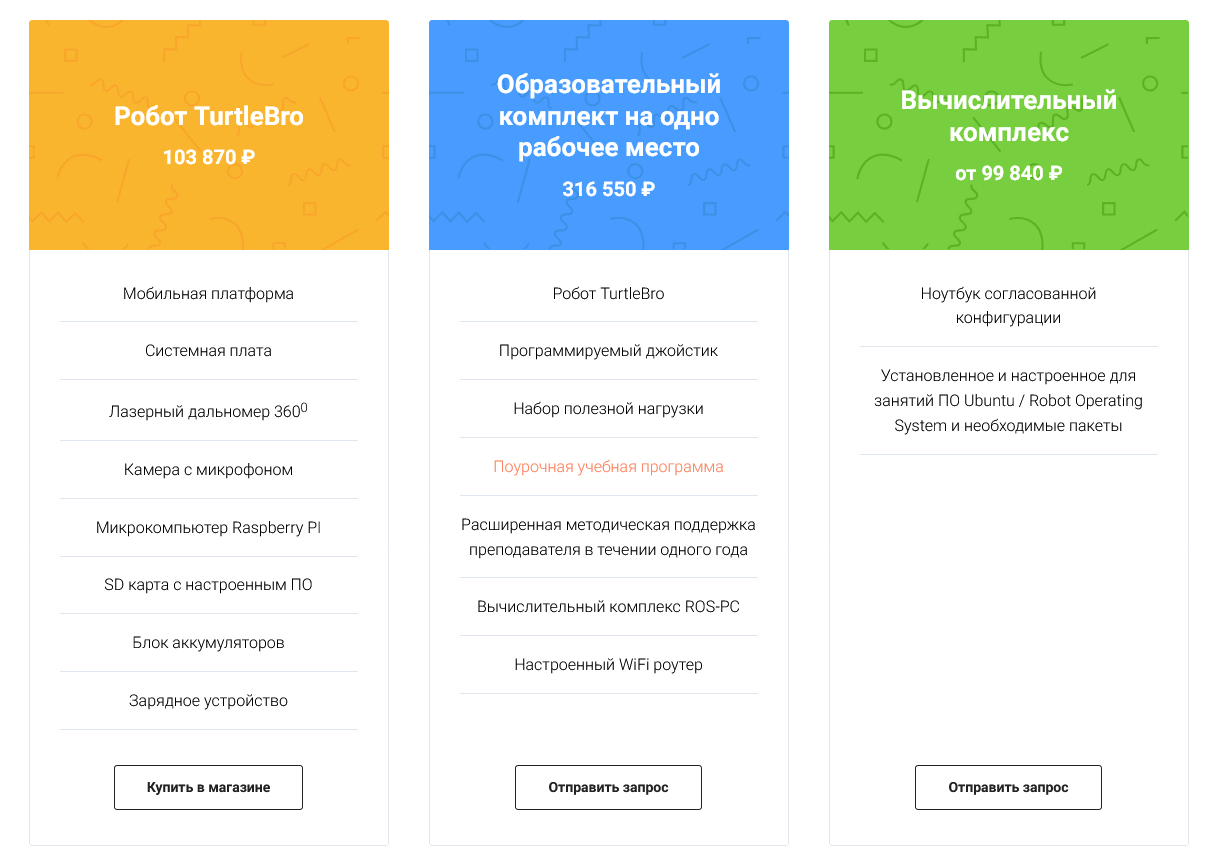


Рисунок 1.1 – Комплектации набора «TurtleBro»

Одним из самых популярных в мире комплектов для изучения ROS2 является набор «TurtleBot». Стоимость и комплектации данного работа вы можете наблюдать на рисунке 1.2.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.1 – Комплектации набора «TurtleBot»

После краткого анализа самых популярных предложений на рынке следует перейти к расчету стоимости самостоятельной сборки набора. При расчете стоимости принимались самые дешевые предложения на рынке, элементы стоимостью ниже пятисот рублей не рассчитывались, так как была поставлена задача рассчитать ориентировочную, а не точную стоимость набора. Комплектация и стоимость элементов можно наблюдать в таблице 1.1.

Таблица 1.1. – Комплектация набора собственной сборки

|  |  |
| --- | --- |
| Название элемента | Стоимость, руб. |
| Raspberry Pi 4 В | 9 500 |
| Lidar Delta 2A | 4 800 |
| STM32F1 (Driver) с MPU6050 | 4 200 |
| Камера с IR-CUT | 1 150 |
| Аккумуляторный блок | 4 500 |
| Платформа с двигателями | 5 400 |
| Две катушки филамента для 3D принтера | 3 000 |
| Набор серводвигателей | 4 000 |
| Блок питания/заряда | 1 200 |
| Итого | 37 750 |

Исходя из рассчитанной стоимости, можно прогнозировать итоговую стоимость проекта до 50 тысяч рублей, что почти в два раза меньше, средней стоимости минимального комплекта на рынке. Однако стоит учитывать, что качество данного комплекта будет значительно ниже, нежели у коммерческих аналогов, а также в стоимость не включены многие пункты, например работа разработчика и работа сборщика.

В данном случае оптимальный выбором будет собрать комплект самостоятельно по причине более низкой стоимости и более простого/дешёвого ремонта, который будет необходим в процессе постоянного использования студентами комплекта для изучения.

### 1.4 Описание пути решения

Необходимо более точно описать выбранное решение.

Решение по самостоятельной сборке комплекта для изучения ROS2 с целью создания учебно-методического комплекса можно условно разделить на 3 этапа: проектирование, сборка, написание методического материала.

Первый этап – это проектирование. На этапе проектирование необходимо разработать систему робота, определить используемые аппаратные и программные элементы с учетом возможности расширения комплекта дополнительными элементами и учетом возможности быстрого и дешёвого ремонта, рассчитать итоговую стоимость комплекта

На втором этапе необходимо произвести сборку и программирование робота. Важными аспектами является ведение документа, описывающего сборку, а также комментирование программного кода.

Третий этап подразумевает написание лабораторных работ и дидактического материала, которые могут использоваться в процессе обучения.

## 2 Проектирование и сборка аппаратной части

### 2.1 Разработка принципиальной схемы элементов

На основе поставленных задач мы можем смоделировать принципиальную схему, которая представлена в приложении А. Упрощенную схему можно наблюдать на рисунке 2.1.



Рисунок 2.1 – Условная схема аппаратных компонентов платформы

В общем случае на данной схеме слева представлены элементы управления и датчики, а в правой части исполнительные органы, по центру находится микрокомпьютер, и отладочная плата с микроконтроллером «STM32 F1» на борту, а также распаянным датчиком гироскопа/акселерометра и двумя H-мостами, позволяющими управлять четырьмя двигателями.

Данная схема позволяет перенести часть вычислений для автоматического управления на внешний компьютер, управлять платформой в ручном режиме, строить карты помещений, производить обработку изображений, а также взаимодействовать с внешней средой благодаря манипулятору.

Также в данной схеме предусмотрен аккумуляторный блок, и блок управлений роботом, позволяющий произвести быструю настройку и управление микрокомпьютером, а соответственно и всеми частями платформы, без подключения. Данные блоки позволят роботу обладать большей автономностью.

На основе имеющейся принципиальной схемы мы может произвести подбор компонентов. В данном случае не будет производиться подбор внешнего компьютера и внешнего устройства ручного управления, так как эти элементы не являются основными и в любой момент могут быть добавлены и убраны, что не повлияет на общую систему. Также при выборе компонентов будут описываться важные характеристики элемента, кроме очевидных.

Начать следует с выбора микрокомпьютера, так как это повлияет на выбор остальных компонентов. Среди представленных на рынке микрокомпьютеров, имеющих вывод «GPIO», который нам необходим, лидерами являются микрокомпьютер «Raspberry Pi 4В 8gb» и «Jetson nano». Оба микрокомпьютера обладают необходимыми техническими требования, а именно: гребенкой пинов GPIO, USB портами, возможность подключения камеры через шлейф, возможностью подключения к микрокомпьютеру экрана и низким вольтажом питания, что позволяет питать микрокомпьютер от аккумуляторного блока. Также оба микрокомпьютера имею схожу стоимость, в базовой комплектации порядка 10 000 рублей. В данном случае выгоднее будет отдать предпочтение микрокомпьютеру «Raspberry Pi 4В 8gb», так как он обладает большей оперативной памятью, 8 гигабайт против 2 и обладает более современным, а как следствие и более мощным процессором, того же семейства. Однако микрокомпьютер «Jetson nano» имеет видеокарту с 256 cuda-ядрами, что позволяет лучше ему справлять с видеопотоком и нейросетями, но в данном случае это не является нашей задачей.

После определения микроконтроллера можно перейти к выбору систем сбора данных (датчикам) в нашем случае это будет камера и лидар. В данном случае большинство представленных на рынке камер имеют схожие характеристики. Нам необходимо, чтобы камера имела угол обзора более 160°, качество съемки более 1080p и шлейф подключения к микрокомпьютеру «Raspberry Pi 4В 8gb». Выберем самое дешевое и популярное предложение. Таким оказалось «OV5647 160». Данная камера является одной из самых популярных на рынке и обладает сравнительно низкой стоимостью, при этом удовлетворяет нашим задачам. Необходимый нам лидар должен обладать возможностью 2D сканирование на 360°, расстоянием между просветами менее одного градуса, дальностью работы более 5 метров и возможностью быстрого подключения и принятия данных в операционной среде «Linux». «Lidar Delta 2A» обладает характеристиками превосходящими необходимые нам и сравнительно низкой стоимостью. Прямых аналогов в данной ценовой категории не имеет, либо имеет, но не достаточно развитых.

Для встроенного блока управления роботом нам необходим экран и энкодер с функцией кнопки, это позволит нам производить отладку робота при его изучении. Экран должен обладать маленьким размером, порядка одного дюйма, качеством изображения более 8 тысяч пикселей и возможностью подключения экрана через интерфейс i2c. В данном случае, нам подойдет модуль экрана «TZT OLED 0.96». Он обладает всеми необходимыми функциями и имеет большое количество готовых программных пакетов для работы с ними. Энкердер в нашем случае должен быть круговым, иметь функцию кнопке и обладать более, чем 10-тью импульсами на полный круг. Нам подойдет эндокер «ЕС11», он обладает всеми необходимыми характеристиками, сравнительно низкой ценой и хорошими отзывами. В данном случае нет смысла искать аналоги, так как выбранные элементы нас полностью устраивают.

Отладочная плата микроконтроллера должна содержать в себе H-мост, датчик гироскоп/акселерометр, возможность подключения двигателей с энкодерами и распаянным на плате стабилизатором напряжения. Контроллер, который используется в отладочной плате, должен иметь возможность работы на частоте 8 МГц, хотя бы по одному интерфейсу i2c и uart и производительной мощностью, достаточной для управления четырьмя серводвигателями и как минимум двумя блоками колес с эндокерами, а также возможностью передачи пакетов данных по uart интерфейсу не реже 10 раз в секунду. Платформа и двигатели должны иметь площадь более 50 см квадратных, и возможностью выдерживать вес в пять килограмм, это необходимо для установки манипулятора и остальных элементов робота. В данном случае были выбрана комплектная платформа и отладочная плата от компании «MoebiusTech Store». Они предлагают отладочную плату на контроллер «STM32 F1» и платформу с четырьмя колесами Илона, крепящимися к двух этажной акриловой плите. Данный набор удовлетворяет всем необходимым нам характеристикам и не имеет подобных аналогов, либо имеет, но слабо развитых. Отладочная плата имеет аналог на базе микроконтроллера семейства «AVR», однако данный мироконтроллер обладает меньшей шиной данных и соответственно меньшей вычислительной способностью. Также семейство микроконтроллеров «STM32» более часто встречается в робототехнике и автоматизации промышленного уровня, поэтому обучение на данном контроллере является более выгодным.

Манипулятор, установленный на роботе должен иметь возможность замены типа захвата и выдерживать вес в точке захвата до 200 грамм, а также имитировать манипулятор используемый в промышленной робототехнике. Был сделан выбор в пользу 3D печати пластиком манипулятора «EEZYbotARM mk2». Модели для печати на данным манипулятор находятся в открытом доступе, а 3D печать облегает модернизацию робота, также данный манипулятор может выдержать вес более 200 грамм и является уменьшенной копией (1:7) промышленного манипулятора «ABB IRB 460».

Драйвер ШИМ сигнала для управления сервоприводами манипулятора должен иметь более 16-ти ШИМ пинов, возможность подключение внешнего питания и интерфейс общения с микроконтроллером «I2C». Самым распространённым и удовлетворяющим всем характеристикам является драйвер PCA9685PWM.

При выборе сервоприводов следует опираться на возможность сервопривода поворачиваться на угол до 180° и быть достаточно распространённым для возможности быстро замены, также сервопривод должен иметь момент силы более 10 кг/см. Всем характеристикам удовлетворяет модель сервопривода «MG995-all metal», также она обладает повышенной прочностью и износостойкость и является одним из лидеров среди сервоприводов в своей ценовой категории.

### 2.2 Проектирование корпуса

При проектировании корпуса стояли следующие задачи:

* удобно разместить все компоненты робота в корпусе;
* размер робота не должен превышать в высоту 50 сантиметров, в длину 30 сантиметров и в ширину 20 сантиметров;
* создать возможность удобного разборки и сборки робота;
* вес робота должен быть равномерно распределен между колесами;

Учитывая данные факторы, был разработан корпус робота в программе КОМПАС-3D. Его модель вы можете наблюдать на рисунке 2.2.

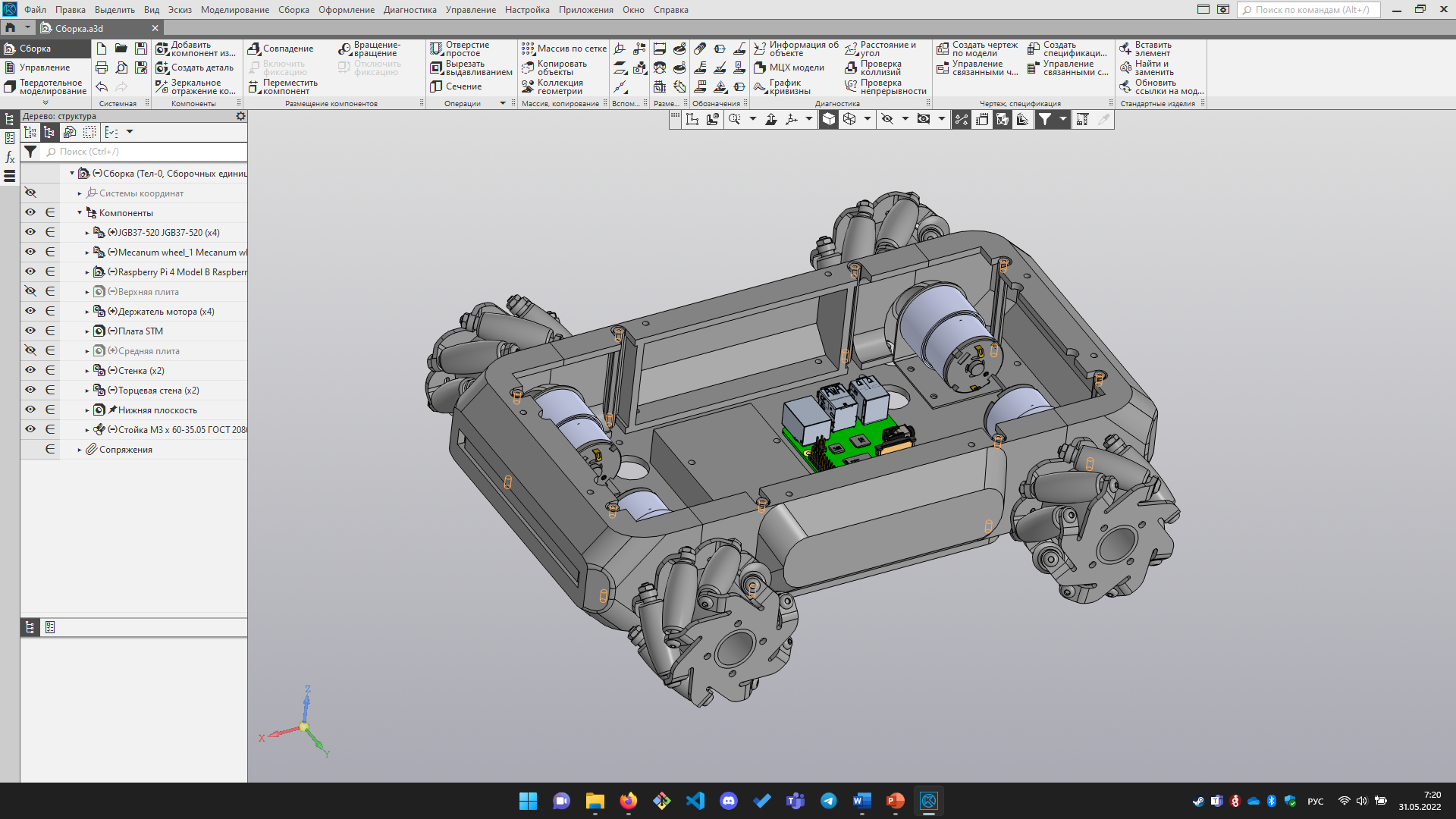


Рисунок 2.2 – Модель корпуса робота

Для модели были взять условные модели микрокомпьютера, отладочной платы микроконтроллера и колес Илона. Верхняя пластина, она же «крышка» робота и средняя пластина для установки аккумуляторов, были скрыты, для удобного отображения внутреннего расположения элементов робота.

Данный корпус можно условно разделить на 3 уровня по расположению элементов.

Первый уровень – нижняя пластина, она же «поддон». На этом уровне расположены стойки м3 длиной 6 см. и стойки м3 длиной 3 см. Первый вид стоек служит креплением между нижней и верхней пластиной, а второй для крепления средней пластины. Также на первом уровне расположены крепления колес, стенки боковые и стенки торцевые, платы микроконтроллера и микрокомпьютера.

Второй уровень – средняя пластина. На этом уровне расположен аккумуляторный блок, понижающий стабилизатор напряжения и сборка коннекторов, для разводки питающих напряжений.

Третий уровень – верхняя пластина. На этом уровне расположен манипулятор и панель управления, которые находятся в задней части робота. Также в передней части робота расположен лидар, вынесенный на стойках м3, и камера, которая находится под ним и направлена вперед робота.

На торцевых стенка робота предусмотрено углубление со сквозными каналами для установки rgb светодиодной ленты, которая может использоваться для подсветки участка перед роботом и сзади робота, но в основном представляет лишь косметическое улучшение внешнего вида.

На пластинах предусмотрены отверстия для крепления элементов и подачи проводки. Пластины будут изготовлены из прозрачного акрила толщиной 4мм. Данный материал достаточно прочный и дешевый. Также изготовление пластин из данного материала можно заказать в рекламных мастерских. По итогу заказа данных пластин стоимость их изготовления составляла около 350 рублей за пластину площадью 160 мм на 300 мм. Среднюю плиту было принято решение изготовить по средствам 3D печати.

Моторы, колеса и их крепления поставлялись комплектом и соответствующие отверстия для крепления были просто перенесены на корпус.

Стенки и средняя плита изготавливались по средствам 3D печати с пластиком PLA и заполнением равном 20%. Суммарное время печати примерно равно 20 часам. Для печати корпуса и манипулятора по подсчетам требовалось примерно 800 граммов пластика без учета неудавшихся печатей. По итогам расчёта было принято решение закупить один килограмм пластика PLA фирмы «Bestfilament» c толщиной нити 1.75 мм. Был выбран белый цвет по личным предпочтениям разработчика.

### 2.3 Изготовление манипулятора

Модель манипулятора было принято взять из проекта с открытыми исходниками. Был выбран манипулятор «EEZYbotARM mk2» от разработчика Carlo Franciscone. Модель данного манипулятора вы можете наблюдать на рисунке 2.3.

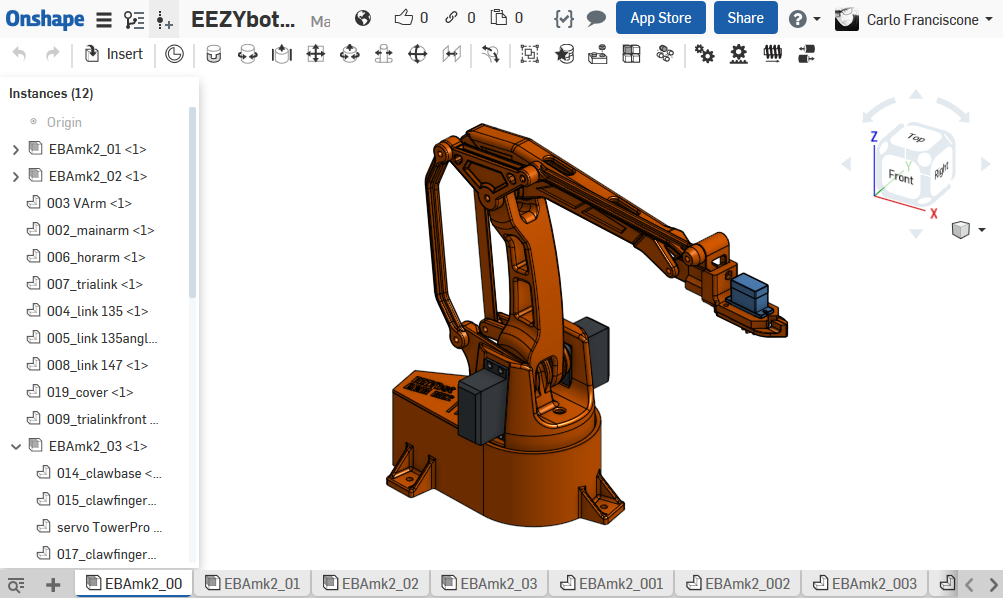


Рисунок 2.3 – Модель манипулятора

Манипулятор «EEZYbotARM mk2» подходит по своим размерам, однако его функционал не в полной мере соответствует целям платформы. Вследствие чего было принято решение заменить захват манипулятора и основание, а также использовать пакет для расширения «коробки» возможных положений захвата. Все изменения были также взяты из открытых источников, указанных в списке использованных источников.

В нашем случае манипулятор приводится в действие с помощью четырех сервоприводов, которые отвечают за поворот манипулятора, за захват, за сгиб «локтя» и сгиб «запястья».

Данный манипулятор используется кинематику промышленного манипулятора «ABB IRB460» с уменьшенным соотношением 1:7. Благодаря данной кинематике захват робота всегда остается в положении параллельном основании.

Манипулятор изготавливался по средствам 3D печати с использованием PLA пластика и заполнением равном 40%. Суммарное время печати примерно равно 30 часам.

Также на захвате манипулятора планировалось использовать камеру, для возможности внедрения обработки изображений и программирования автоматического захвата, однако вследствие резкого подорожания необходимых элементов и увеличения срока доставки до более чем двух месяцев, было принято решение реализовать данного робота с камерой лишь в передней части. Это не позволит управлять манипулятором без прямого зрительного контакта пилота и робота. Однако данный функционал планируется к внедрению после нормализирования указанных выше факторов.

### 2.4 Отладочная плата STM32F1

Компанией «MoebiusTech» разработана отладочная плата с микроконтроллером «STM32F103RCT6», двумя H-мостами «TB6612» и гироскопом/акселерометром «MPU6050». Данная плата входит в часть комплект с моторами «GB37-520», на которых дополнительно установлен энкодер, для реализации возможностей одометрии. Отдельно на плате выведенj два USB порта, одни для питания Raspberry PI, второй для передачи данных между микроконтроллером и микрокомпьютер.

Программирование данной платы возможно посредствам внешнего программатора «ST-Link». Питание платы осуществляется внешним источником постоянного питания от 6 до 12 вольт. В нашем случае это будет сборка из литий-полимерных аккумуляторов с напряжением 7.4 вольта Фото отладочной платы вы можете наблюдать на рисунке 2.4.

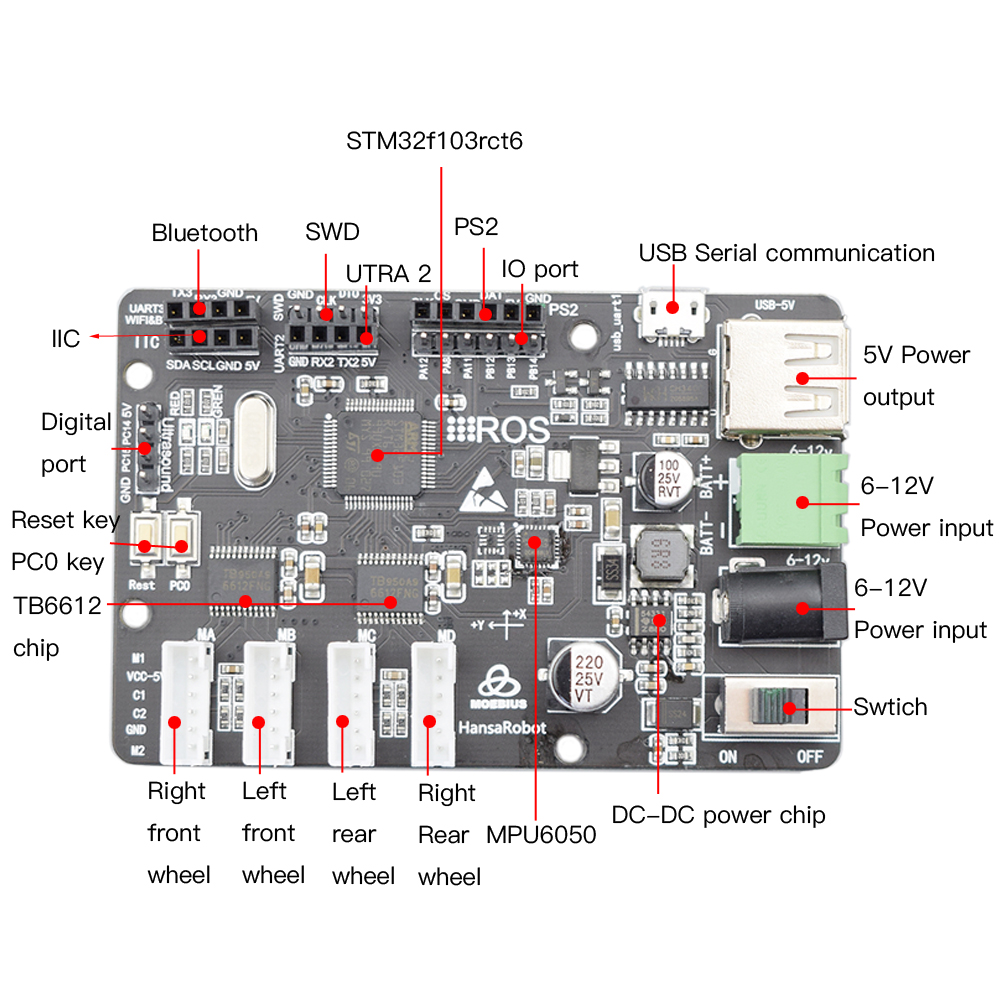


Рисунок 2.4 – Отладочная плата Moebius

Электрическую схему платы вы можете наблюдать в приложении Б.

Данная плата является самым дешевым вариантом, имеющий весь необходимый функционал из найденных.

### 2.5 Микрокомпьютер RPI 4B

Микрокомпьютер Raspberry Pi 4 Model B был выбран в качестве микрокомпьютера для обработки видео сигнала и коммуникации с внешним пультом управления.

Данный микрокомпьютер обладает интерфейсом «WI-FI» и «Bluetooth», что позволит передавать управляющий сигнал по сети или с помощью Bluetooth соединения. Это одни из самых удобных интерфейсов беспроводной передачи данных. Также данный компьютер имеет четыре порта USB, что подключения периферии и отдельный порт для подключения шлейфа камеры.

В нашем случае будет использоваться версия на восемь мегабайт оперативной памяти, с процессором «Broadcom BCM2711». Также на микрокомпьютере установлена гребенка пинов «GPIO», для подключения низкоуровневой периферии.

Прямым аналогом данного микрокомпьютера является плата «Jetson nano». Фактический это прямой аналог, из важных в нашем случае различий можно выделить то, что Raspberry имеет 8 гигабайт оперативной памяти, против 4 у Jetson, так же Raspberry обладает более современным процессором. Однако, Jetson имеет 256 Cuda ядер, которые очень важны во время обучения и использования нейросетей.

В нашем случае перед нами не стоит цель использования нейросетей, и как более дешевый и более мощный аналог был выбран микрокомпьютер Raspberry PI 4В.

Для питания Raspberry требуется постоянный источник питания 5 вольт 3 ампера. Как мы разбирали выше, отладочная плата Moebius имеет выход USB, с подключенным преобразователем напряжения, настроенным на 5 вольт и максимальным выходным током 3 ампера. Поэтому проблем с питанием не должно возникнуть.

Так же в дополнение к микрокомпьютеру был заказан алюминиевый корпус с двумя вентиляторами. Фото микрокомпьютера вы можете наблюдать на рисунке 2.5.

Изображение выглядит как текст, электроника, цепь

Автоматически созданное описание

Рисунок 2.5 – Микрокомпьютер Raspberry PI 4B

### 2.6 Лидар 3irobotix Delta-2A

Лидар позволяет сканировать окружавшее пространство по плоскости, посылая и принимая инфракрасный сигнал с вращающейся башни.

В нашем случае этот лидар был выбран, как самый дешёвый лидар, обладающий необходимым функционалом.

Радиус сканирования данного сигнала составляет от 0,13 до 8 метров, учитывая препятствие в виде белой стены. При этом лидар посылает один сигнал до препятствия раз в 0,3–0,8 градуса. Потребление лидара составляет 500 миллиампер. При этом коммуникация с компьютером происходит по средствам TTL преобразователя сигнала. Одним шлейфом он подключается к лидару, и двумя USB портами подключается к компьютеру. Поставляется лидар с комплектом ПО для работы, а именно: пакетом для ROS, программой на языке С, и программой для операционной системы Windows.

Фото данного лидара можно наблюдать на рисунке 2.6.



Рисунок 2.6 – Лидар Delta-2A

### 2.6 Разработка панели управления

Для взаимодействия с роботом и программирование микрокомпьютера будет использоваться «SSH» подключение, однако оно возможно только когда компьютеры клиент и сервер находятся в одной сети. И вывести его в одну сеть иногда проблематично. Для этого разрабатываться панель управления, чтобы можно было изменить основные настройки робота и получить отладочную информацию, например о подключении к Wi-Fi сети или уровне заряда аккумулятора.

На панели управления будет установлен энкодер с кнопкой, «OLED» дисплей и кнопка питания. Модель корпуса вы можете наблюдать на рисунке 2.7.

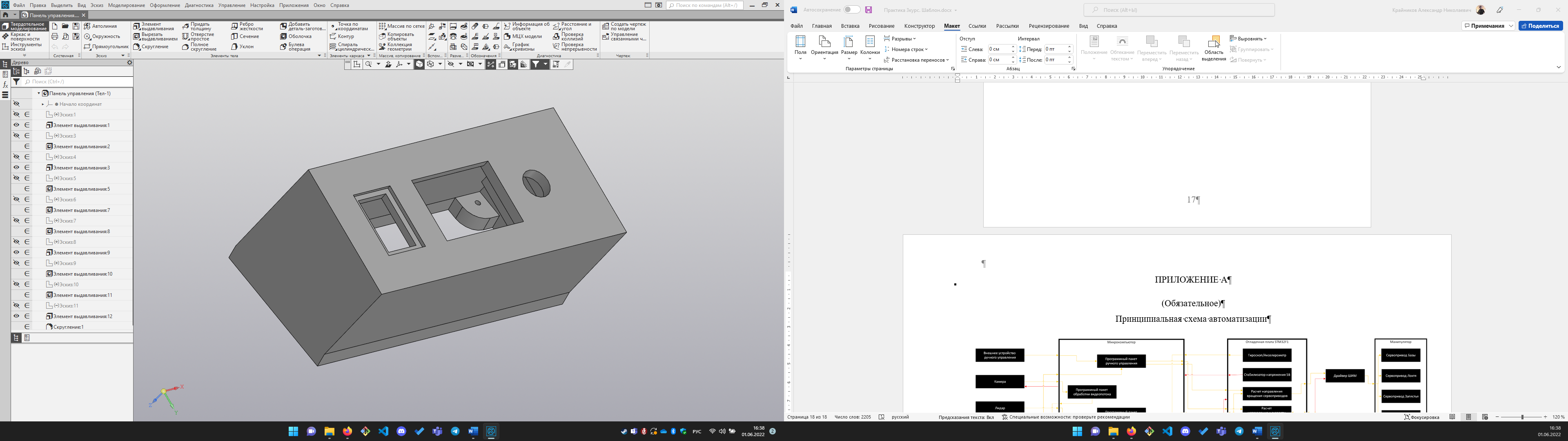


Рисунок 2.7 – Корпус панели управления

Кнопка включения разрывает цепь питания микроконтроллера, что приводит к отключению всех систем. Дисплей позволит выводить информацию о роботе, а в совокупности с энкодером изменять настройки робота без подключения к нему.

В частности, планируется реализовать вывод на экран имени Wi-Fi сети в которой находится робот, его IP адреса в этой сети и текущий заряд аккумулятора. Изменять с этой панели планируется следующие настройки: Wi-Fi сеть, к которой подключен робот, перевести микрокомпьютер в перезагрузку/выключить его, включить автономное патрулирование по заранее построенной карте.

### 2.7 Сборка робота

Сборка робота начинается с установки моторов, стоек, микроконтроллера и микрокомпьютера. Моторы подключаются к плате микроконтроллера по специальным шлейфам. На плате порты моторов пронумерованы, как «MA», «MB», «MC», «MD». В нашем случае моторы следует подключать начиная с переднего правого и против часовой стрелки, если смотреть на платформу сверху. Собранную нижнюю пластину можно наблюдать на рисунке 2.8.

Изображение выглядит как внутренний

Автоматически созданное описание

Рисунок 2.7 – Сбор нижней плиты платформы

Далее необходимо установить стенки корпуса и закрепить их к нижней плите. Для удобного распределение проводов по корпусу, были дополнительно напечатаны на 3D принтере специальные крепления, которые липкой лентой удерживают несколько проводов в определенном месте корпуса. Установленные стенки на платформу робота можно наблюдать на рисунке 2.8.

Изображение выглядит как электроника, фотоаппарат, проектор

Автоматически созданное описание

Рисунок 2.8 – Сборка корпуса робота

Далее необходимо собрать верную пластину, установить на нее собранный манипулятор, лидар, камеру и панель управления. Алгоритм сборки можно найти на сайте разработчика манипулятора, указанного в списке использованных источников. Собранную панель управления можно наблюдать на рисунке 2.9.

Изображение выглядит как рука

Автоматически созданное описание

Рисунок 2.9 – Собранная панель управления.

Подключаем провода к панели управления, шилду управления сервоприводами и на термоклей крепим TTL преобразователь от лидара. После чего для удобства объединяем все провода в единое подобие шины. Собранную верхнюю пластину можно наблюдать на рисунке 2.10.

Изображение выглядит как автомат

Автоматически созданное описание

Рисунок 2.10 – Собранная верхняя пластина

Далее собираем среднюю плиту, на нее необходимо установить холдер для аккумуляторов типа 18650, стабилизатор напряжение и три клемника для разводки питания на 5 вольт, питания напрямую от аккумулятора и 0 вольт (землю). Соединить проводами в соответствии с схемой подключения компонентов. Собранную среднюю плиту можно наблюдать на рисунке 2.11.

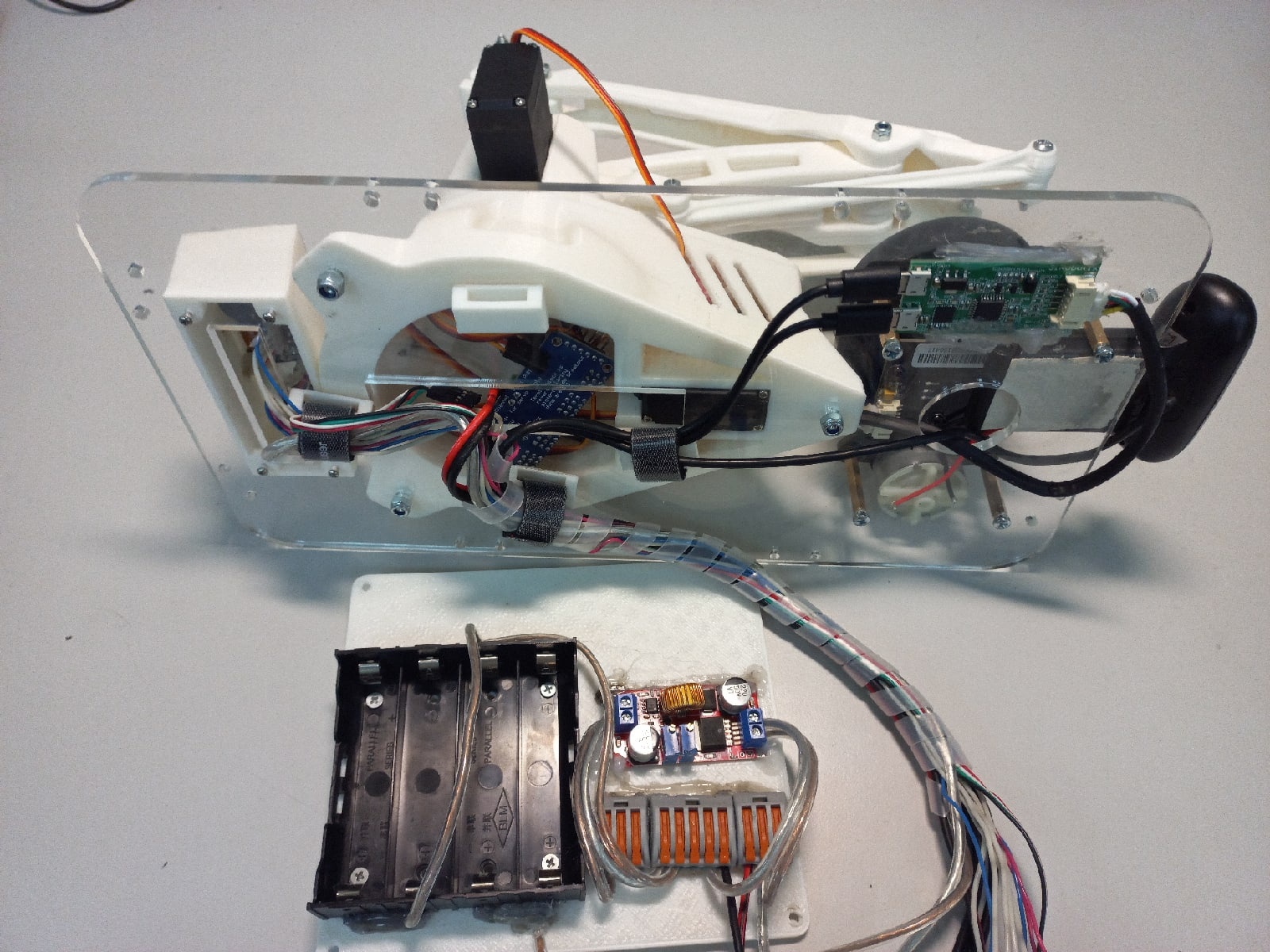


Рисунок 2.10 – Собранная средняя пластина

Далее необходимо соединить провода от аккумуляторов с кнопкой установленной на панели управления. Провода с аккумулятора после ключа (кнопки) подключить в плату микроконтроллера. С шины 5 вольт подключить провод питания на шилд управления сервоприводами, также с этой шины запитать светодиодные ленты, предварительно настроив стабилизатор напряжения на 5 вольт выход. Провода от энкодера и экрана подключить в соответствующие пины микрокомпьютера. После этого устанавливаем среднюю плиту на стойки 30 мм. И сверху прикручиваем верхнюю плиту к стойкам 60 мм. На этом сборка робота завершена. Фотографию итогового робота вы можете наблюдать на рисунке 2.11.

Изображение выглядит как прибор

Автоматически созданное описание

Рисунок 2.11 – Собранный робот

## 3 Разработка программной части

### 3.1 Архитектура

В общей своей структуре существует три вычислительных мощности, выполняющих программный код: персональный компьютер, микроконтроллер, микрокомпьютер. Отдельно условно можно выделить Wi-Fi, хоть она и не будет исполнять никакой код, а будет выступать лишь как передатчик информации, однако для удобства мы делим ее как отдельно структуру, в которой будут содержаться топики, места стечения информации. Весь код можно условно разделить на единицы, ноды, функции, библиотеки, которые можно приписать к тем или иным структурам. Условный граф распределения программного кода можно наблюдать на рисунке 3.1.

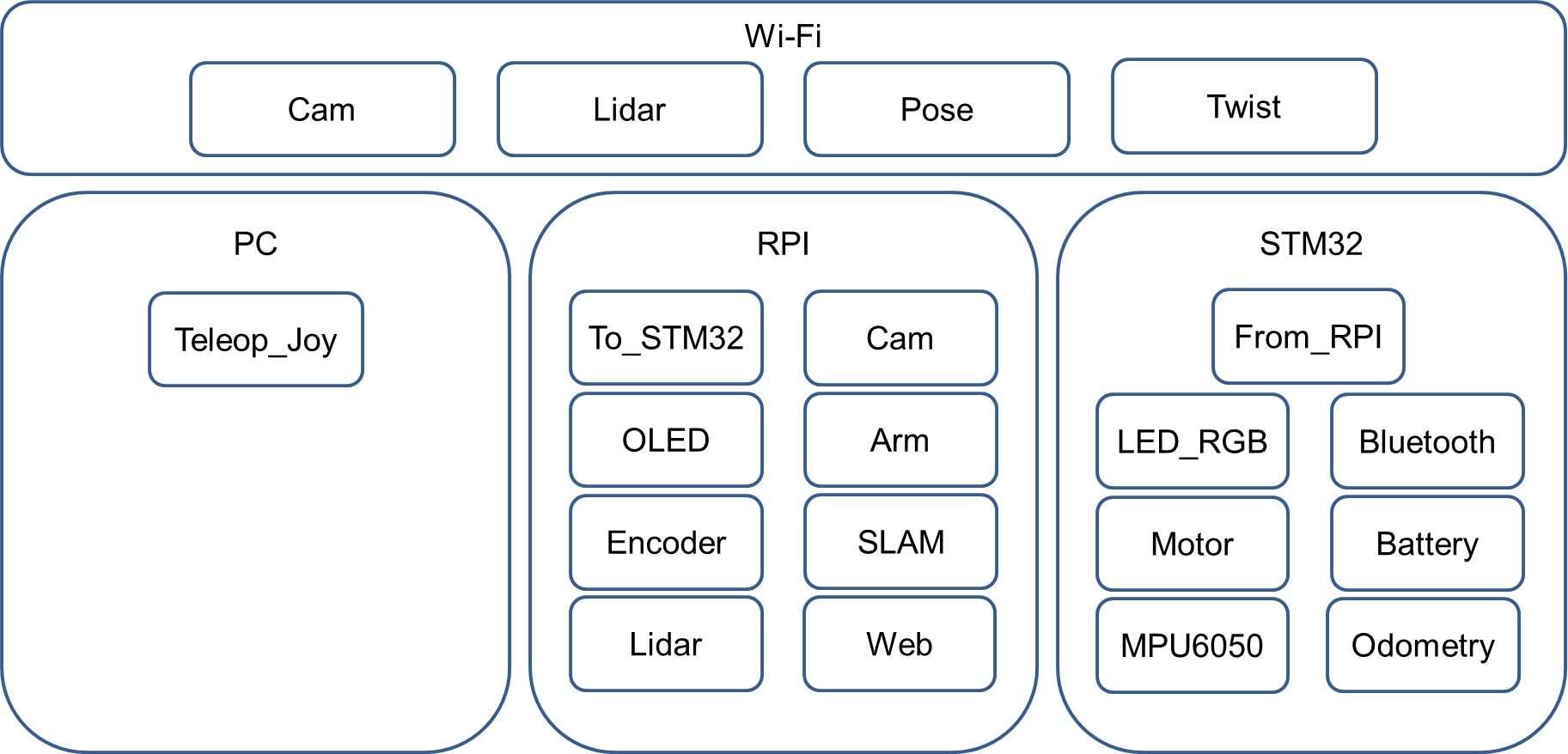


Рисунок 3.1 – Распределение программного кода

Персональный компьютер будет отвечать лишь за сбор данных с джойстика и передачу этих данных в топики «Twist» и «Pose», которые отвечают за перемещение робота и управление манипулятором. Отдельно можно выделить, что компьютер может считывать данные с топиков со всех топиков, указанных в структуре «Wi-FI», для отображения данных.

Микрокомпьютер же поддерживает связь с микроконтроллером, передает данные с камеры в топик «Сam», данные с лидара в топик «Lidar», а также считывает данные с других топиков и производит соответствующие действия. Отдельно в функции микрокомпьютера можно выделить отображение отладочной информации на дисплее робота, считывание энкодера и расчет данных для «SLAM» навигации.

Микроконтроллер отвечает за поддержание связи с микрокомпьютером, изменения состояния светодиодной ленты, считывания данных с энкодеров колес, гироскоп/акселерометра, и, конечно, за управление моторами.

Итоговая версия программного обеспеченья может отличать от описанной здесь. Также в работе будет рассмотрен лишь код настройки и код примера управления некоторыми элементами робота, потому что описание итогового кода сильно усложнит понимание принципа работы программного кода. Однако, итоговый программный код вы можете наблюдать на интернет-ресурсе [какой-то].

### 3.2 Персональный компьютер

В нашем случае персональный компьютер не будет выполнять много задач. На персональном компьютере будет исполнять пакет ROS2 «teleop\_twist\_joy», который будет считывать данные с джостика, подключенного к компьютеру и передавать данные в топик «Twist» и топик «Joy». Однако, нам понадобится только первый топик. Данный пакет будет установлен из источников, представленных в свободном доступе.

Также на компьютере можно будет запускать ПО «rviz2» и «rqt», подобные программы понадобятся для графического отображение данных с лидара, камеры, а также отображения построенной камеры и положения робота на ней. Интерфейс программы «rviz2» с построенной картой можно наблюдать на рисунке 3.2. Данная программа поставляется вместе ROS2.

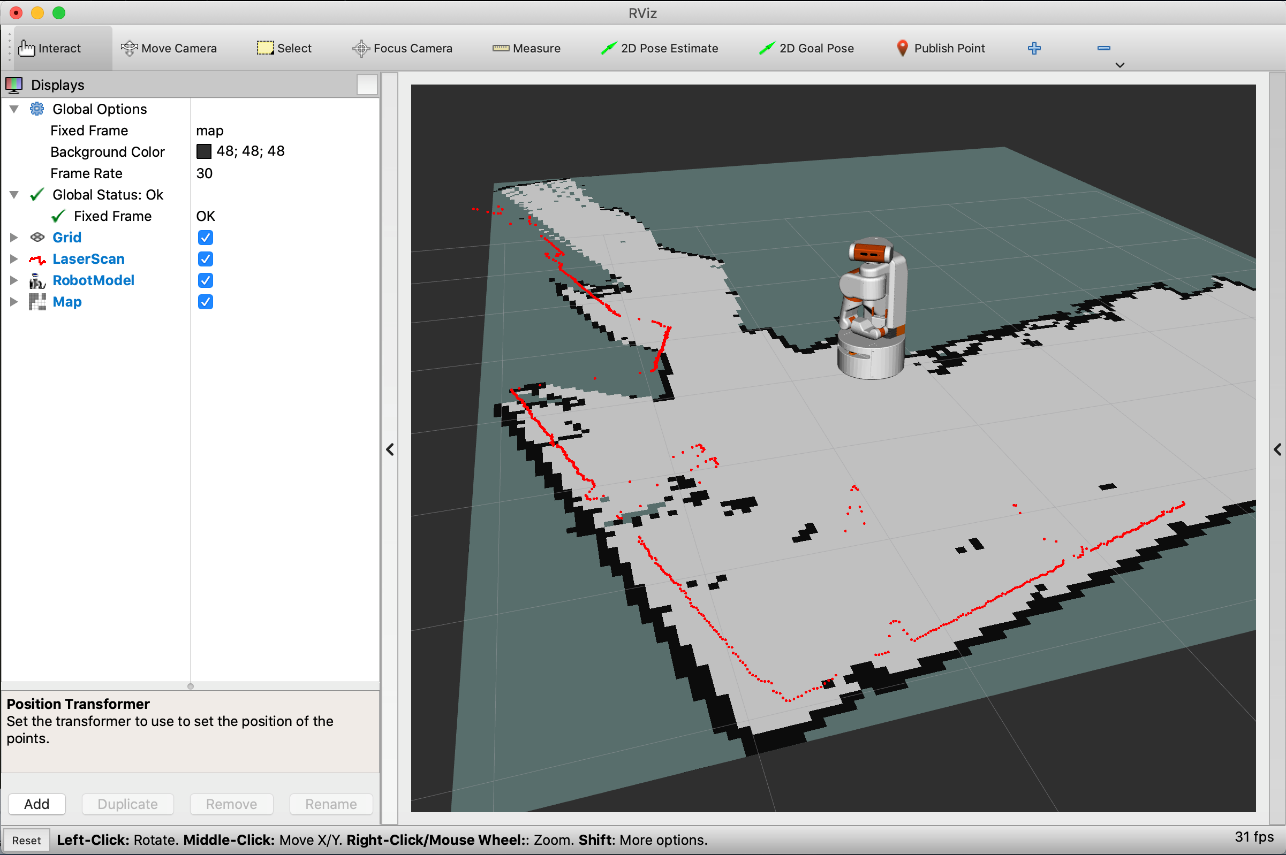


Рисунок 3.2 – Интерфейс программы rviz2

### 3.3 Управление моторами

Управление моторами базируется на вращение моторов с помощью драйверов «TB6612» и считывании данных с энкодеров. Эти данные можно будет использовать как для синхронизирования вращения колес, так и для одометрии, то есть расчёта пройденного расстояния.

Один чип «TB6612» может управлять двумя моторами. В данном случае у моторов есть четыре состояния: «блокировка», «остановка», «вращение по часовой», «вращение против часовой». Эти состояния задаются благодаря высокому и низкому сигнала на четырех ножках. Управление скоростью вращения колес используется дополнительная ножка, принимающая ШИМ-сигнал. Итого для управления четырьмя моторами нужно восемь пинов состояния и четыре пина скорости. Дополнительно нам нужно два пина и отдельный таймер для считывания вращения колес с энкодера

Для этого необходимо инициализировать двадцать пинов, как пины выхода. К четырем из них нужно подключить по каналу одного из таймеров. И к восьми по парам нужно подключить четыре различных тамера. Далее нужно настроить таймер и ШИМ-сигнал. Все эти настройки можно выставить в программе «СubeMX» с помощью графического интерфейса. Указанные настройки сгенерируют код для контроллера. Данный код вы можете наблюдать в листинге 1.

Листинг 1 – Код для настройки управления колесами

|  |
| --- |
| void MiniBalance\_Motor\_Init(void)  {  GPIO\_InitTypeDef GPIO\_InitStructure;  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOD, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE);    GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_12;  GPIO\_Init(GPIOA, &GPIO\_InitStructure);  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_4 | GPIO\_Pin\_5;  GPIO\_Init(GPIOB, &GPIO\_InitStructure);    GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4 | GPIO\_Pin\_5 | GPIO\_Pin\_12;  GPIO\_Init(GPIOC, &GPIO\_InitStructure);  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;  GPIO\_Init(GPIOD, &GPIO\_InitStructure);  INA1=0;INB1=0;INC1=0;IND1=0;  INA2=0;INB2=0;INC2=0;IND2=0;    Set\_MotorPWMA(0);  Set\_MotorPWMB(0);  Set\_MotorPWMC(0);  Set\_MotorPWMD(0);  }  void MiniBalance\_PWM\_Init(u16 arr,u16 psc)  {  MiniBalance\_Motor\_Init();    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM8, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE);    RCC->APB2ENR|=1<<13;  RCC->APB2ENR|=1<<4;  GPIOC->CRH&=0XFFFFFF00;  GPIOC->CRH|=0X000000BB;    GPIOC->CRL&=0X00FFFFFF;  GPIOC->CRL|=0XBB000000;  TIM8->ARR=arr;  TIM8->PSC=psc;  TIM8->CCMR1|=6<<4;  TIM8->CCMR1|=6<<12;  TIM8->CCMR2|=6<<4;  TIM8->CCMR2|=6<<12;    TIM8->CCMR1|=1<<3;  TIM8->CCMR1|=1<<11;  TIM8->CCMR2|=1<<3;  TIM8->CCMR2|=1<<11;  TIM8->CCER|=1<<0  TIM8->CCER|=1<<4;  TIM8->CCER|=1<<8;  TIM8->CCER|=1<<12;  TIM8->CCR1=0;  TIM8->CCR2=0;  TIM8->CCR3=0;  TIM8->CCR4=0;  TIM8->BDTR |= 1<<15;  TIM8->CR1=0x8000;  TIM8->CR1|=0x01;  }  #include "encoder.h"    void Encoder\_Init\_TIM2(void)  {  // RCC->APB2ENR|=1<<0;  AFIO->MAPR|=1<<8;  RCC->APB1ENR|=1<<0;  RCC->APB2ENR|=1<<2;  RCC->APB2ENR|=1<<3;  GPIOA->CRH&=0X0FFFFFFF;  GPIOA->CRH|=0X40000000;    GPIOB->CRL&=0XFFFF0FFF;  GPIOB->CRL|=0X00004000;  TIM2->DIER|=1<<0;  TIM2->DIER|=1<<6;  MY\_NVIC\_Init(1,3,TIM2\_IRQn,1);  TIM2->PSC = 0x0;  TIM2->ARR = ENCODER\_TIM\_PERIOD;  TIM2->CR1 &=~(3<<8);  TIM2->CR1 &=~(3<<5);    TIM2->CCMR1 |= 1<<0;  TIM2->CCMR1 |= 1<<8;  TIM2->CCER &= ~(1<<1);  TIM2->CCER &= ~(1<<5);  TIM2->SMCR |= 3<<0;  TIM2->CR1 |= 0x01;  }  int Read\_Encoder(u8 TIMX)  {  int Encoder\_TIM;  switch(TIMX)  {  case 2:  {  Encoder\_TIM = (short)TIM2 -> CNT;  TIM2 -> CNT=0;  }  break;    case 3:  {  Encoder\_TIM = (short)TIM3 -> CNT;  TIM3 -> CNT=0;  }  break;  case 4:  {  Encoder\_TIM = (short)TIM4 -> CNT;  TIM4 -> CNT=0;  }  break;  case 5:  {  Encoder\_TIM= (short)TIM5 -> CNT;  TIM5 -> CNT=0;  }  break;  default: Encoder\_TIM=0;  }  return Encoder\_TIM;  }  void TIM2\_IRQHandler(void)  {  if(TIM2->SR&0X0001)  {  }  TIM2->SR&=~(1<<0);  } |

### 3.3 Считывание данных с MPU6050

Работа с данным датчиком осуществляется по интерфейсу «i2c», путем считывания и записи данных в регистры и имеет много особенностей. Всего данный модуль имеет 3 датчика: трех осевой акселерометр, трех осевой гироскоп и датчик температуры.

Для начала нам необходимо инициализировать пины для обмена данными по i2c. После этого необходимо задать состояния конфигурационных регистров и произвести калибровку датчиков. Обычно это происходит в процессе инициализации устройства. Код, описывающий считывание данных и инициализацию устройства указан в листинге 2.

Листинг 2 – Код считывания данных с MPU6050

|  |
| --- |
| void MPU6050\_I2C\_init(void) {  I2C\_InitTypeDef I2C\_InitStructure;  GPIO\_InitTypeDef GPIO\_InitStructure;  RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_I2C1, ENABLE);  RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6 | GPIO\_Pin\_7;  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_10MHz;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;  GPIO\_Init(GPIOB, &GPIO\_InitStructure);  I2C\_InitStructure.I2C\_Mode = I2C\_Mode\_I2C;  I2C\_InitStructure.I2C\_DutyCycle = I2C\_DutyCycle\_2;  I2C\_InitStructure.I2C\_OwnAddress1 = 0x13; // Собственный адрес  I2C\_InitStructure.I2C\_Ack = I2C\_Ack\_Enable;  I2C\_InitStructure.I2C\_AcknowledgedAddress = I2C\_AcknowledgedAddress\_7bit;  I2C\_InitStructure.I2C\_ClockSpeed = 100000; // 100 кГц  I2C\_Cmd(I2C1, ENABLE);  I2C\_Init(I2C1, &I2C\_InitStructure);  }  //=============== Запись данных Data в регистр Reg по I2C ===============  void MPU6050\_Write(uint8\_t Reg, uint8\_t Data) {  while (I2C\_GetFlagStatus(I2C1, I2C\_FLAG\_BUSY));  I2C\_GenerateSTART(I2C1, ENABLE);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_MODE\_SELECT));  I2C\_Send7bitAddress(I2C1, (0x68 << 1), I2C\_Direction\_Transmitter);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED));  I2C\_SendData(I2C1, Reg); // Передаём адрес регистра  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED));  I2C\_SendData(I2C1, Data); // Передаём данные  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED));  I2C\_GenerateSTOP(I2C1, ENABLE);  }  //=============== Чтение данных из регистра Reg по I2C ===============  uint8\_t MPU6050\_Read(uint8\_t Reg) {  static uint8\_t Data;  while (I2C\_GetFlagStatus(I2C1, I2C\_FLAG\_BUSY));  I2C\_GenerateSTART(I2C1, ENABLE);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_MODE\_SELECT));  I2C\_Send7bitAddress(I2C1, (0x68 << 1), I2C\_Direction\_Transmitter);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED));  I2C\_Cmd(I2C1, ENABLE);  I2C\_SendData(I2C1, Reg); // Передаём адрес регистра  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED));  I2C\_GenerateSTART(I2C1, ENABLE);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_MODE\_SELECT));  I2C\_Send7bitAddress(I2C1, (0x68 << 1), I2C\_Direction\_Receiver);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_RECEIVER\_MODE\_SELECTED));  I2C\_AcknowledgeConfig(I2C1, DISABLE);  I2C\_GenerateSTOP(I2C1, ENABLE);  while (!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_RECEIVED));  Data = I2C\_ReceiveData(I2C1); // Принимаем данные  I2C\_AcknowledgeConfig(I2C1, ENABLE);  return Data;  }  int main(void) {  uint16\_t X, Y, Z;  volatile uint32\_t i32;  for(i32 = 0; i32 < 100000; i32++) {}; // Программная задержка  MPU6050\_I2C\_init(); // Настраиваем I2C  for(i32 = 0; i32 < 100000; i32++) {};  //Датчик тактируется от встроенного 8Мгц осциллятора  MPU6050\_Write(0x6B, 0x00); // Register\_PWR\_M1 = 0, Disable sleep mode  //Выполнить очистку встроенных регистров датчика  MPU6050\_Write(0x6A, 0x01); // Register\_UsCtrl = 1  for (i32 = 0; i32 < 1000; i32++) {};  while (1) {  X = MPU6050\_Read(0x3B) << 8; X |= MPU6050\_Read(0x3C);  Y = MPU6050\_Read(0x3D) << 8; Y |= MPU6050\_Read(0x3E);  Z = MPU6050\_Read(0x3F) << 8; Z |= MPU6050\_Read(0x40);  X = MPU6050\_Read(0x43) << 8; X |= MPU6050\_Read(0x44);  Y = MPU6050\_Read(0x45) << 8; Y |= MPU6050\_Read(0x46);  Z = MPU6050\_Read(0x47) << 8; Z |= MPU6050\_Read(0x48);  // Вывод переменных  }  } |

### 3.4 Работа с OLED дисплеем

Работа с дисплеем таже осуществляется по интерфейсу i2c. В этом случае мы будем работать с дисплеем через Raspberry Pi и библиотеки «Adafruit SSD1306», «Python Imaging Library» для языка Python. В общем случае, определенные пиксели экрана закрашиваются и таким образом рисуются определенные символы. Для обновления экрана необходимо закрасить его «черным». Так же, следует включать экран только на период работы с ним. С этим связаны особенности работы OLED экрана. Код, пример работы представлен в листинге 3.

Листинг 3 – Работа с OLED дисплеем

|  |
| --- |
| import Adafruit\_GPIO.I2C as I2C  import Adafruit\_SSD1306  from PIL import Image, ImageDraw, ImageFont  import subprocess  RST = 0  disp = Adafruit\_SSD1306.SSD1306\_128\_64(rst=RST)  disp.begin()  disp.clear()  disp.display()  width = disp.width  height = disp.height  image1 = Image.new('1', (width, height))  draw = ImageDraw.Draw(image1)  draw.rectangle((0,0,width,height), outline=0, fill=0)  padding = -2  top = padding  bottom = height-padding  x = 0  font = ImageFont.load\_default()  while True:  draw.rectangle((0,0,width,height), outline=0, fill=0)  # выводим на экран две строчки текста  disp.clear()  disp.display()  draw.text((x, top), "OLED Interfacing " , font=font, fill=255)  draw.text((x, top+8), "Circuit Digest", font=font, fill=255)  draw.text((x, top+16), "For more Videos", font=font, fill=255)  draw.text((x, top+25), "Visit at", font=font, fill=255)  draw.text((x, top+34),"https://", font=font, fill=255)  # выводим на экран изображение  disp.image(image1)  disp.display()  time.sleep(2)  if disp.height == 64:  image = Image.open('img1.png').convert('1')  else:  image = Image.open('img1.png').convert('1')  disp.image(image)  disp.display()  time.sleep(2)  if disp.height == 64:  image = Image.open('img3.jpg').convert('1')  else:  image = Image.open('img3.jpg').convert('1')  disp.image(image)  disp.display()  time.sleep(2)  if disp.height == 64:  image = Image.open('img4.jpg').convert('1')  else:  image = Image.open('img4.jpg').convert('1')  disp.image(image)  disp.display()  time.sleep(2) |

## 4 Лабораторные работы

### 4.1 Техническое обслуживание робота

Цель: изучить принципы проведения технического обслуживания робота.

Краткие теоретические сведенья:

Задание:

1. Проверить правильность подключения всех элементов робота.
2. Проверить вольтаж аккумуляторного блока.
3. Запустить робота
4. Обновить пакеты приложений и гит репозитории
5. Проверить получение изображения с камеры
6. Проверить получение данных с лидара
7. Проверить получение данных с MPU6050
8. Проверить данные одометрии
9. Проверить управление манипулятором
10. Проверить управление двигателями
11. Заполнить документ о техническом обслуживании

Дополнительная литература:

### 4.2 Базовый функционал

Цель: изучить принцип работы базового функционала робота

Краткие теоретические сведенья:

Задание:

Дополнительная литература:

### 4.3 Автономная навигация

Цель: изучить принцип работы автономной навигации с построением карты

Краткие теоретические сведенья:

Задание:

Дополнительная литература:

# Заключение

По итогам выполнения поставленных задач на преддипломную практику, был выполнен анализ возможности изготовления робота и проведена полная его сборка.

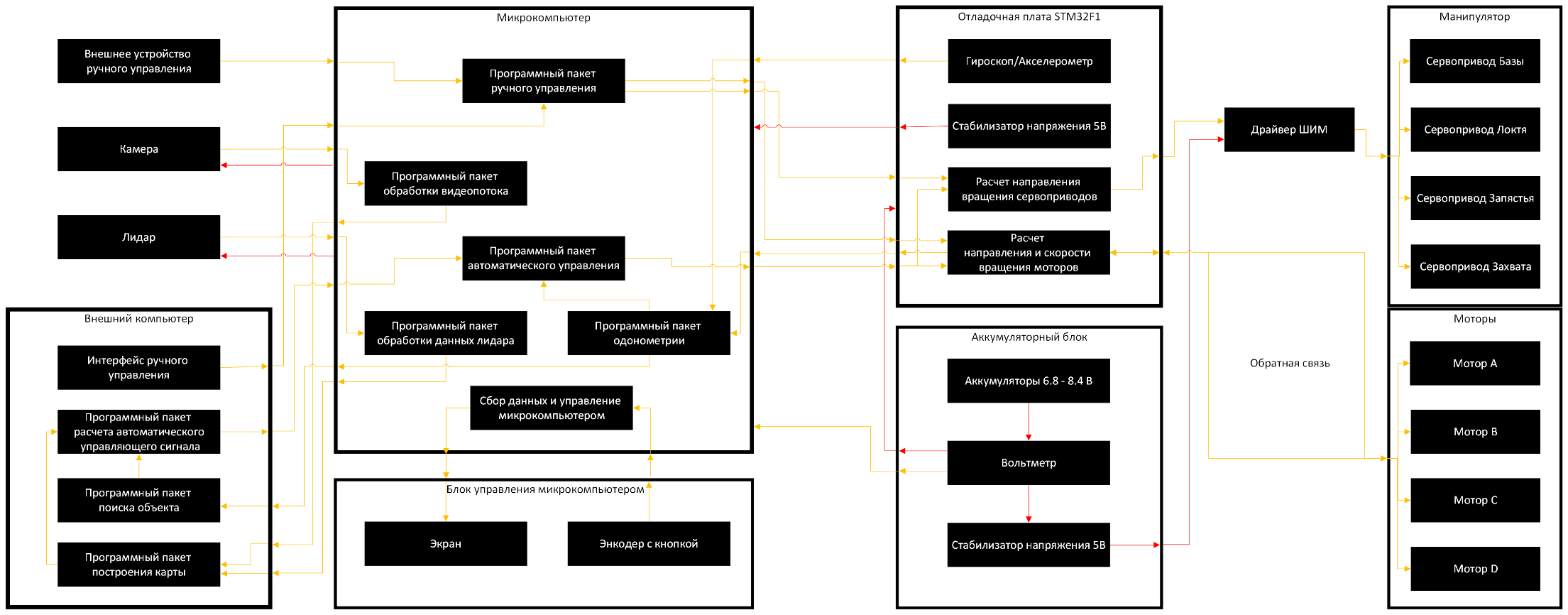
# Список использованных источников

1. Название со страницы [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: ссылка на сайт, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
2. Роботизация промышленного производства на базе искусственного интеллекта [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: https://cyberleninka.ru/article/n/robotizatsiya-promyshlennogo-proizvodstva-na-baze-iskusstvennogo-intellekta/viewer, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
3. Команда ROS-Industrial разрабатывает роботизированное устройство для улучшения машинного зрения [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: https://robroy.ru/komanda-ros-industrial-razrabatyivaet-robotizirovannoe.html, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
4. Промышленные роботы в России [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Промышленные\_роботы\_в\_России, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
5. АСУ ТП (рынок России) [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: https://www.tadviser.ru/index.php/Статья:АСУ\_ТП\_(рынок\_России), свободный (дата обращения: 24.12.2021). – Загл. с экрана.
6. TurtleBro [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: http://www.turtlebro.ru/, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
7. ROBOTIS TURTLEBOT [Электронный ресурс]: [офиц. сайт] / Электрон. дан. – Режим доступа: https://robotgeeks.ru/collection/robotis-turtlebot, свободный (дата обращения: 24.12.2021). – Загл. с экрана.
8. https://www.thingiverse.com/thing:4362304
9. https://www.thingiverse.com/thing:4632254

# Приложение А

(Обязательное)

Принципиальная схема автоматизации



# Приложение Б

(Обязательное)

Электрическая схема отладочной платы Moebius

.

СВЕДЕНИЯ О САМОСТОЯТЕЛЬНОСТИ ВЫПОЛНЕНИЯ РАБОТЫ

Выпускная квалификационная работа «Разработка-аппаратного программного комплекса для внедрения ROS2» выполнена самостоятельно.

Используемые в работе материалы и концепции из публикуемой литературы и других источников имеют ссылки на них.

Электронный экземпляр выпускной квалификационной работы в формате pdf размещен на странице онлайн-курса «ГИА\_15.03.04 Автоматизация систем управления производством (21–22)»

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| « » 2022г. |  |  | А.Н. Крайников |
|  | (подпись) |  | (инициалы, фамилия) |