Областное государственное бюджетное общеобразовательное учреждение
"Томский физико - технический лицей"

Техническое описание робота команды Таёжные Ёжики

Работу выполнили:
ученица МБОУ лицей при ТПУ г. Томска М.С. Цыганкова
ученик ОГБОУ «ТФТЛ» Г.А. Пильщиков

Наставник проекта: С.В. Косаченко
Заместитель директора по
информационным технологиям ОГБОУ
«ТФТЛ»

Томск
2025

СОДЕРЖАНИЕ

АНОТАЦИЯ

Данная работа содержит описание работы над роботом для решения задач RCJ Rescue Maze. Особенностью нашего робота заключается в самодельные элементы конструкции и использовании микрокомпьютера Raspberry Pi 4 для определения жертв. Стратегии выполнения самого задания мы выбрали следующую. Наш робот передвигается по лабиринту по приоритетам прямо, лево право. Во время движения он просчитывает свою текущую координату относительно клетки старта и заносит информацию о посещении, наличие/присутствии стен, жертв, болота/ямы в карту, при этом робот в может пренебрегать приоритетом для изучения ранее не изученных клеток. Эта стратегия позволяет изучить как можно больше территории и найти как можно больше жертв.

Рисунок 1 — фото команды

Опыт участия и успехи команды в различных робототехнических соревнованиях и фестивалях

- RoboScience Hackathon 2024 - 1 место в состязании Robocup Junior Rescue Maze
- Международный фестиваль Робофинист 2022 — 3 место в категории Robocup Junior Rescue Maze, г.Санкт-Питербург
- Кубок Губернатора Томской области 2024 — 1 место в состязании роботов с техническим зрением памяти Виктора Ширшина
- Кубок Губернатора Томской области 2023 — 1 место в состязании роботов с техническим зрением памяти Виктора Ширшина
- Кубок Губернатора Томской области 2022 — 2 место в состязании роботов с

техническим зрением памяти Виктора Ширшина

- XIII Региональная олимпиада по образовательной робототехнике школьников Томской области 2024 — 2 место в состязании Robocup Junior Rescue Maze
- XII Региональная олимпиада по образовательной робототехнике школьников Томской области 2023 — 1 место в состязании Robocup Junior Rescue Simulation Webots Erebus
- Открытый Российский чемпионат по робототехнике 2023 — 2 место в состязании Robocup Junior Rescue Simulation Webots Erebus
- Победа во всероссийском конкурсе научно-технологических проектов “Большие вызовы”
- 2022 год — Участие Цыганковой Марии и Пильщикова Григория в фестивале подводной робототехнике «АкваРобоФест» в г.Асино

ВВЕДЕНИЕ

Предполагаемый результат проекта

При разработке проекта планировалось создать уменьшенный прототип аппарата, для полностью автономного проведения поисково-спасательной операции, анализируя и приспосабливаясь к окружающей среде, следовательно, в соответствии ГОСТ Р60, робота-спасателя для выполнения поставленных задач.

Команда для реализации проекта

1) Пильщиков Григорий - капитан команды. Разработал алгоритмы компьютерного зрения, занимался пайкой электрических компонентов, занимался сборкой робота.
2) Цыганкова Мария - член команды. Разработала алгоритмы движения, навигации и картографирования, составляла чертежи деталей конструкции робота, чертежи плат занималась подключением электрических компонентов, сборкой робота.
3) Косаченко Сергей Викторович - наставник команды.

Цель работы

Разработать уменьшенный прототип автономного робота для помощи проведения поисково-спасательных операций.

Задачи

1. Изучить уже существующие решения, аналоги с похожим функционалом
2. Разработать работоспособный макет робота
3. Протестировать робота, проанализировать полученные результаты
4. Доработать минусы, появившиеся в результате тестирования

ОСНОВНАЯ ЧАСТЬ РАБОТЫ

Планирование работ

1) Формулировка технического задания, расчет конструкции.

Необходимые ресурсы/компоненты: ноутбук, программы для черчения.

Промежуточный результат: разработанная и рассчитанная конструкция робота, список необходимых для сборки компонентов.

2) Сборка и тестирование шасси.

Необходимые ресурсы/компоненты: металлические несущие балки, двигатели с энкодорами, шестерни/колеса, гусеничная лента, набор инструментов.

Промежуточный результат: готовое рабочее шасси с возможности регулировать натяжение гусениц без разбора механизма.

3) Сборка каркаса робота, установка контроллера.

Необходимые ресурсы/компоненты: контроллер, ЧПУ станок, фанера, гайки / шайбы / болтики, резьбовые шпильки, набор инструментов.

Промежуточный результат: готовый корпус робота.

4) Пайка и подключение электрических компонентов.

Необходимые ресурсы/компоненты: паяльник, канифоль, олово, флюс, третья рука, пинцет, соединительные провода, подобранные и приобретенные электрические компоненты.

Промежуточный результат: готовая ходовая часть робота.

5) Разработка ПО, тестирование начального прототипа.

Необходимые ресурсы/компоненты: ноутбук, кабель для программирования, место для испытаний.

Промежуточный результат: начальный прототип робота с возможностью передвижения в замкнутых пространствах.

6) Установка камер, разработка системы компьютерного зрения.

Необходимые ресурсы/компоненты: ноутбук, микрокомпьютер, камеры, система подсветки камеры.

Промежуточный результат: готовый прототип робота с системой обнаружения жертв.

7) Разработка систем навигации и картографирования.

Необходимые ресурсы/компоненты: ноутбук, кабель для программирования, симулятор для быстрого тестирования алгоритмов.

Промежуточный результат: экспериментальный прототип ПСР, готовый к тестированию.

8) Тестирование и анализ результатов.

Необходимые ресурсы/компоненты: ноутбук, кабель для программирования, полигон для испытаний, монитор для снятия телеметрии, монитор.

Промежуточный результат: выявленные плюсы/минусы текущей версии ПСР

9) Доработка и усовершенствование.

Необходимые ресурсы/компоненты: в зависимости от того, что будем улучшать.

ИТОГ: Экспериментальный прототип автономного робота для проведения поисково-спасательных операций.

Формулировка технического задания проекта, разработка концепции робота

Для простоты тестирования, за основу требований к роботу были взяты требования и регламенты различной сложности состязаний роботов-спасателей для студентов и школьников. Мы смоделировали спасательную миссию: робот должен по максимуму обследовать помещение (рисунок 2), построить его карту, отметить и сообщить об опасных или потенциально опасных местах, при обнаружении пострадавших, незамедлительно сообщить спасателям и скинуть комплект для оказания первой помощи, продолжить миссию, после окончании операции вернуться обратно к месту старта. Таким образом, нетрудно сформулировать и сами требования к роботу.

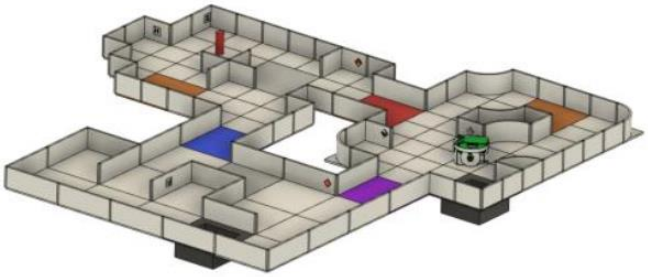


Рисунок 2 – модель поля проведения миссии

- Шасси робота должно быть достаточно прочным, надежным и проходимым, для преодоления препятствий различной трудности.
- Шины или гусеницы должны быть выполнены из нескользящего материала, для увеличения сцепления с поверхностью.
- При использовании гусеничного шасси, необходимо разработать системе удобной регулировки их натяжения.
- Следует предусмотреть энкодоры на каждый из моторов.
- Робот должен обладать датчиками и сенсорами для перемещения по объекту и построении его карты.
- Робот должен контролировать обстановку не только вокруг себя, но и под самим роботом.
- Робот должен обладать камерами и системой компьютерного зрения для обнаружения подозрительных вещей.
- Для системы компьютерного зрения, необходимо выделить микрокомпьютер, задачей которого будет являться анализ изображения и отправка сигнала в случае обнаружения уже на сам контроллер.
- Питание контроллера и микрокомпьютера следует разделить, то есть необходимо предусмотреть раздельное питание с удобством быстрой смены аккумуляторов.
- Для устойчивости конструкции, необходимо стремиться сместить центр тяжести робота как можно ниже.
- Для питания контроллера следует использовать высокотоковые аккумуляторы типа

18650.

- Необходимо предусмотреть стабилизатор питания для некоторых датчиков.
- Дополнительно, необходимо также предусмотреть систему выгрузки комплектов первой помощи.
- Для удобства тестирования, необходимо обеспечить наличие дисплейного модуля для вывода информации.

Таким образом, была ращработана следующая концепция робота. В качестве контроллера была выбрана плата meAURIGA, так как она достаточно мощная и представляет из себя по сути Arduino Mega 2560 с уже припаянными необходимыми нам компонентами. В качестве микрокомпьютера была выбрана Raspberry Pi 4, так как это полноценная замена компьютеру на Linux, которая позволяет выполнять все поставленные нами задачи. Конструкция робота предполагает несколько этажей с разным функционалом: на нулевом располагаются моторы, подсветка для камер и датчик цвета, который мы выбрали для контроля состояния поля под роботом, так как при помощи параметра освещенности можно реагировать на ямы до того как их заметил гироскоп; на первом — аккумуляторы для meAURIGA, пауэрбанк Raspberry Pi (RPI); на втором — сама meAURIGA, Raspberry, жесткий диск, преобразователь логических уровней, двунаправленный, 4-х канальный, переходная плата с портов meAURIGA RJ12 на разъемы с шагом 2.45 мм, стабилизатор питания с 5В до 3.3В и инфракрасные дальномеры; на третьем — два экрана для вывода информации, кнопка для быстрого запуска и сами камеры. В пространстве между 2-ым и 3-ем этажами робота располагается бокс для проводов, жесткий диск и meAURIGA. Выгрузка комплектов происходит за счет трубы с самими комплектами первой помощи и гладкой шестерни, которая приводится в движение за счет сервопривода. Шасси плоское, гусеничное, ход заднеприводный. Основа шасси собрана из металлических балок, однако они находятся под углом и скреплены при помощи металлической оси с гайками и шайбами. За счет такой конструкции шасси робот становится более проходимым и пригодным к изменениям натяжения гусениц без разборки всей конструкции.

Описание стратегии

Робот перемещается в лабиринте по изначальному приоритету прямо, лево, право, но после захождения в тупик возвращается исследовать неисследованные пути, при этом выбор неизведанного прохода происходит по алгоритму право, лево, вперед. Такой алгоритм позволяет исследовать как можно большую часть лабиринта. При нахождении жертвы он привязывает ее к конкретной координате, что помогает избежать повторных обнаруже-

ний. Для самого же обнаружения у нас есть нейросеть, которая быстро реагирует в передвижении и алгоритм метод точек для более точного результата. Для экономии памяти бы дробим каждый байт на биты и оперируем ими. Таким образом, используя все возможности нашего робота, мы способны выполнить всю миссию.

Разработка механической части робота

Разработка шасси

Начальным этапом разработки робота стал этап разработки шасси для робота. Перед началом разработки было решено, что шасси стоит сделать гусеничным, так как это увеличивает проходимость роботу. Первая версия шасси робота была собрана полностью из цельных металлических балок, так как это увеличивает прочность шасси, имела форму треугольника и состояла из 6-ти небольших шестерней (рисунок 3).

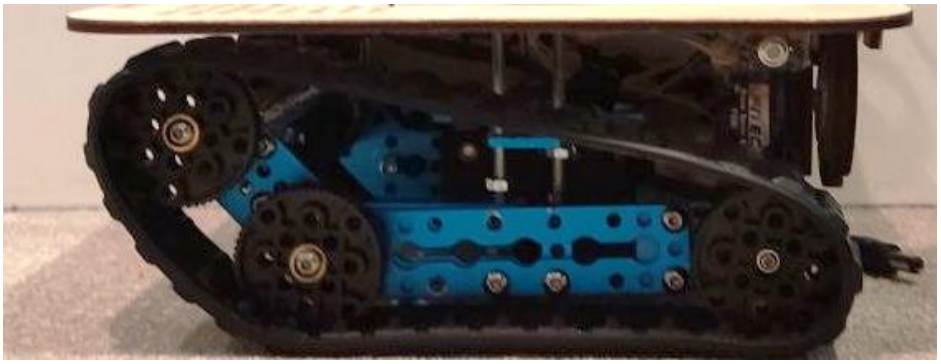


Рисунок 3 – первая версия шасси для робота

Во время тестирования шасси были выявлены две главные проблемы этой версии – отсутствие механизма регулировки натяжения гусениц и переворачивания из-за неудобств треугольной конструкции шасси. Первое, что было решено менять в этой версии это именно треугольную форму, однако металлические балки в конструкции все еще должны были остаться, так как в противном случае шасси может потерять в надежности. В результате чего была изменена конструкция шасси – оно стало полностью плоским (т.е. не имеет наклона, который был в предыдущих версиях), что позволило увеличить устойчивость робота и понизить его центр тяжести (рисунок 4).

Рисунок 4 – вторая версия шасси для робота

В последней версии этого робота в шасси добавили механизм для возможности регулировки натяжения гусениц. Система натяжения для регуляции без разбора шасси была собрана из металлической оси, гаек и шайб. В результате чего, стало возможным изменение натяжения гусениц на месте проведения миссии при помощи гаечного ключа и плоскогубцев. Благодаря этим изменениям у нас появилась возможность ездить как на гусеницах, так и без них, а также увеличилось пространство 1-го этажа, за счет чего нам удалось опустить 2-й и сместить центр тяжести ниже, что также увеличило его проходимость (рисунки 5, 6, 7).

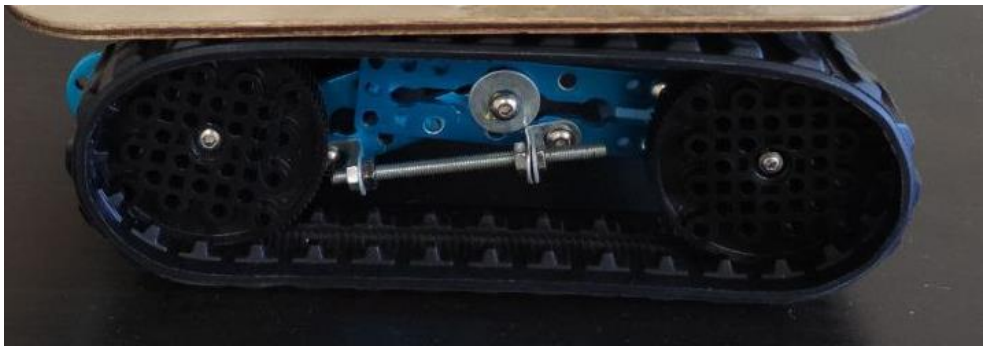


Рисунок 5 – третья версия шасси для робота

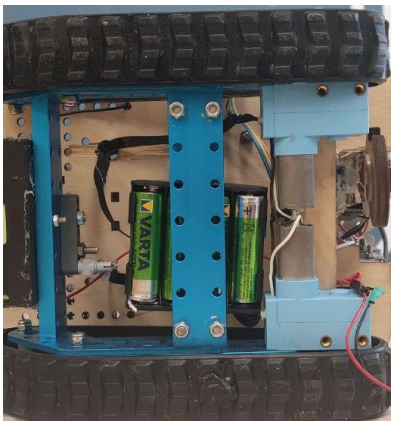


Рисунок 6, 7 – третья версия шасси для робота с разных сторон

Разработка корпуса

Следующим этапом разработки стала разработка рамы робота. Форма рамы была выбрана прямоугольной с закругленными углами, которые позволят роботу избежать возможных зацеплений и залипаний угла о угол. Рама была смоделирована в системе автоматизированного проектирования (САПР), при моделировании были учтены отверстия для крепежей последующих деталей, протяжки проводов, шестерни для выгрузки и крепежа основных осей, соединяющих этажи (рисунок 8).

Далее были разработаны крепежи для датчиков расстояния. Перед проектировкой они были протестированы, после чего было рассчитано минимальное, но оптимальное расстояние для их корректной работа, именно на основе этого расстояния и были выбраны положения крепежей на раме. Кроме того, на этом этапе также был разработан и смоделирован каркас для 3-го этажа (рисунок 9).

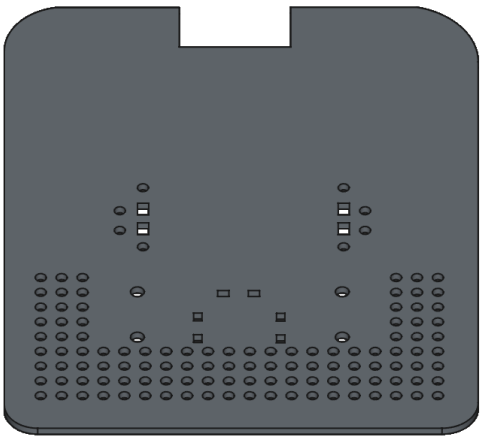


Рисунок 8 – модель готовой рамы робота

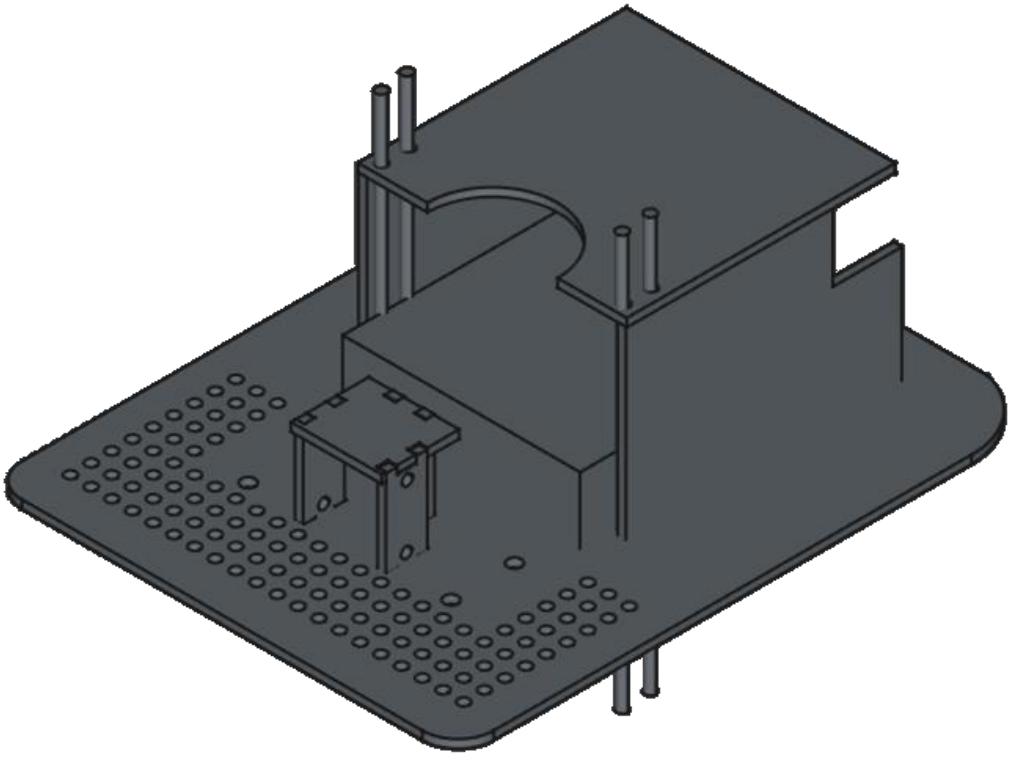


Рисунок 9 – модель основного корпуса робота

После окончании проектировки корпуса робота начался этап его изготовления. В качестве материалов для изготовления прототипа робота была выбрана фанера, так как для нас это был наиболее доступный и очень прочный материал. Все детали были вырезаны на ЧПУ станке и обработаны. В результате, чего был собран готовый корпус робота (рисунок 10).

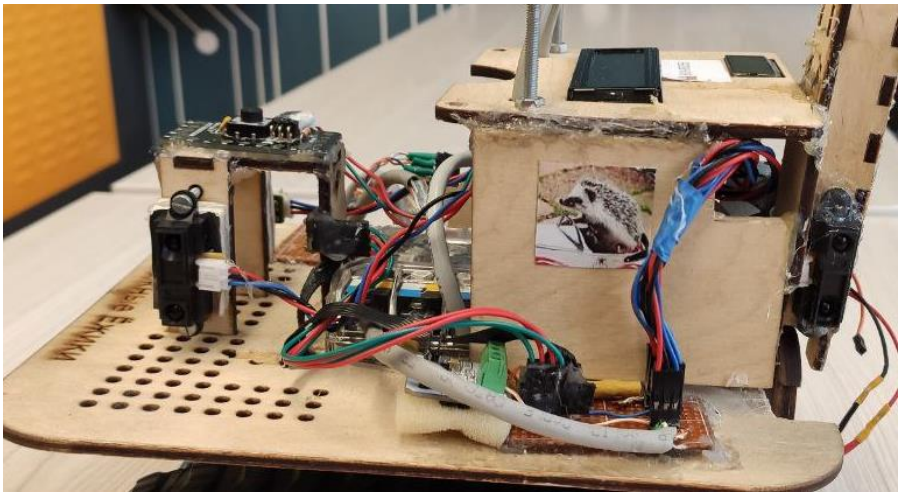


Рисунок 10 – итоговая версия корпуса робота в сборе

Для разработки механизма выгрузки, также использовали САПР. Сам же механизм состоит из трубы-контейнера, содержащего спасательные комплекты первой помощи, одного сервопривода и псевдошестерни, на которую под действием силы тяжести падает комплект (конструкция выполнена так, что в отсек на шестерне помещается ровно один). Сама выгрузка происходит посредством вращения этого сервопривода (рисунок 11).

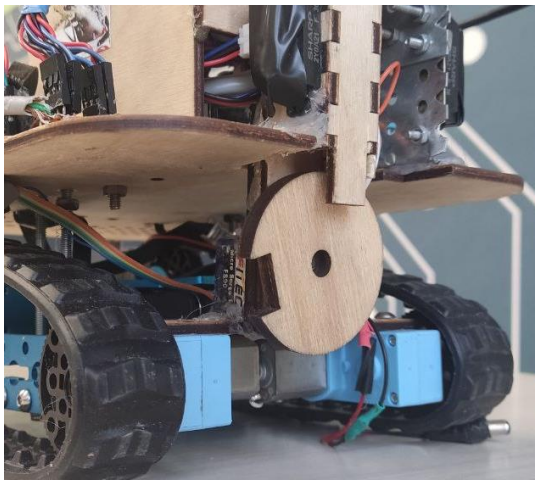


Рисунок 11 – механизм выгрузки в сборе

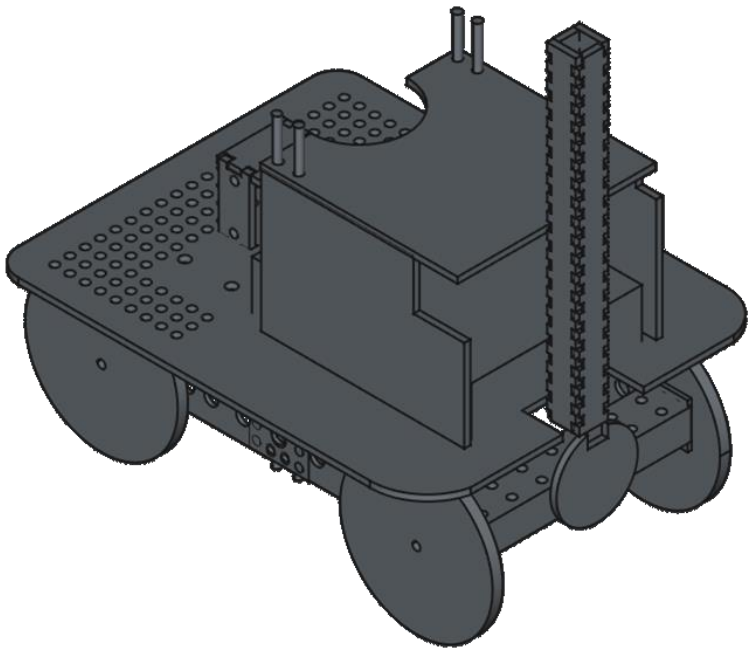


Рисунок 12 – итоговая модель корпуса робота

Разработка электрической части робота

Пайка и подключение электрических компонентов

В качестве контроллера робота была выбрана плата meAURIGA, так как она достаточно мощная и представляет из себя по сути Arduino Mega 2560 с уже припаянными необходимыми нам компонентами, такими как гироскоп или модуль Bluetooth. В качестве микрокомпьютера была выбрана Raspberry Pi 4. Также были выбрали инфракрасные дальномеры sharp 2y0a21 f 36, так как они достаточно точные и дешевые, датчик цвета HiTechnic NCO1038, так как он наиболее доступный и простой в подключении, веб камеры, считываемые RPI, которая анализирует данные с камеры и передает на meAURIGA, двигатели постоянного тока DC Motor-37, так как они достаточно легкие и мощные (скорость без нагрузки – 350 об/мин и редукция 39,6), высокоточные, уже со встроенными оптическими энкодорами точностью до 360°, а также поддерживает выбранную нами плату управления. Драйвера для двигателей уже встроены в контроллер. Данные с RPI на meAURIGA мы передаем по самодельному протоколу, используя для приема-передачи данных 4 GPIO с RPI и 4 цифровых пинов от meAURIGA, и из-за разницы напряжения логических выводов, пришлось использовать преобразователь логических уровней, для подключения моторчика выгрузки комплектов мы также спаяли переходную плату с портов meAURIGA RJ12 на разъемы с шагом 2.45 мм. Кроме того, на роботе стоит стабилизатор питания с 5В на 3.3В. Для более удобного старта на роботе вынесена кнопка, запускающая самого робота. Распиновку meAURIGA и RPI 4, мы брали с официальной технической документации.

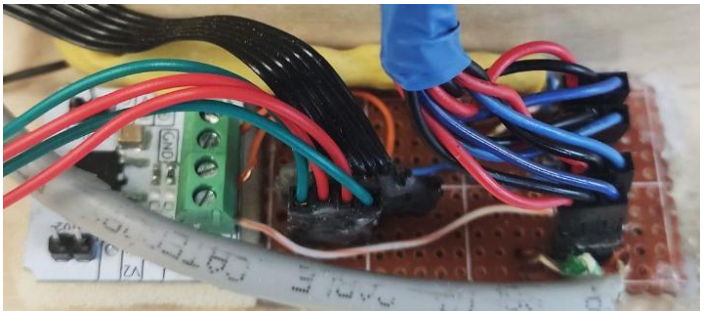


Рисунок 13 – готовая плата для подключения дальномеров

Перед изготовлением и распайкой электрических компонентов для робота необходимо было разработать и рассчитать их структурные и электрические схемы в САПР (см Приложение 1, 2). Так, для питания датчика цвета Hitechnic NCO1038, требуется 3.3В, когда контроллер выдает 5В, следовательно, возникла потребность в стабилизаторе питания с 5В на 3.3В, для подключения дальномеров, нам пришлось спроектировать и распаять переходную плату (рисунок 13) с RJ-12 на стандартные разъемы с шагом 2.45 мм, что облегчило подключение датчиков. А также для "связи" между RPI и

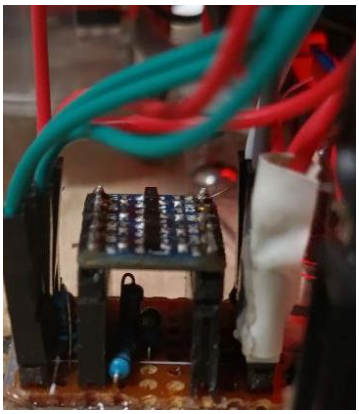


Рисунок 14 – готовая плата преобразователя логических уровней

meAURIGA было необходимо спроектировать и спаять плату-переходник (рисунок 14), с посадочным местом под плату преобразователя логических уровней, между GPIO пинами RPI и цифровыми пинами meAURIGA, со стягивающими резисторами, для нейтрализации шумов.

После расчета напряжения для каждого электрического компонента робота, все электрические узлы были распаяны, собраны в общую схему, после чего протестированы и закреплены на роботе, а провода аккуратно уложены и скрыты в боксе для проводов (рисунки 15, 16). Для питания общей электрической схемой были выбраны литий ионные аккумуляторы 18650, в количестве двух штук, так как таким образом на вход контроллера подается ≈8.4В, что является наиболее оптимальным для спокойного управления двигателями. Для RPI же выделили отдельное питание от обычного пауэрбанка на 5В, это обусловлено тем, что при одном источнике питания RPI может начать питаться от meAURIGA и перегореть.

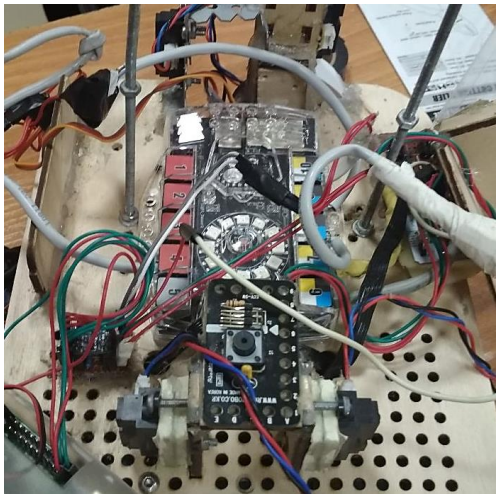


Рисунок 15 – собранная электрическая часть робота

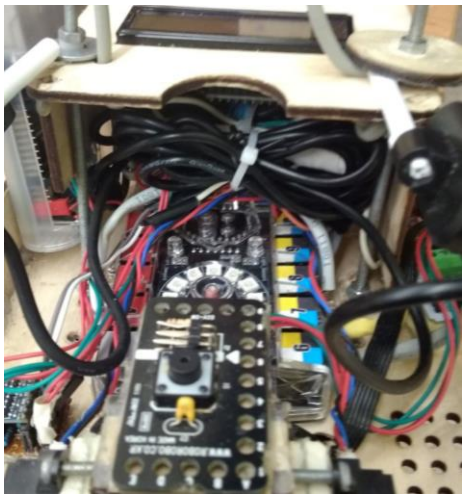


Рисунок 16 –электрическая часть робота после фиксации всех элементов и укладки проводов

Разработка программного обеспечения

Программное обеспечение робота делится на две глобальные задачи: алгоритмы передвижения и навигации (ПО для meAURIGA),а так же алгоритмы компьютерного зрения (ПО для Raspberry).

Контроллер прошивали через текстовую среду разработки Arduino IDE, так как она наиболее доступна и поддерживает библиотеки для meAURIGA. Этот алгоритм состоит из нескольких подпрограмм: передвижение, общение с RPI, построение карты и навигация по ней. Стоит отметить, что для простоты и точности тестирования ПО для контроллера робота, использовался стимулятор Webots Erebus, где были воссозданы некоторые полигон для проведения миссий и сам робот (рисунки 17, 18). Программу же писали в обычной текстовой среде программирования на языке C++, для удобства тестирования каждая подпрограмма писалась как отдельная программа, и уже позже собиралась в одну основную. Контроль над передвижением осуществляется через оптические энкодоры, встроенные в моторы, для простоты построения карты полигон как-бы мысленно делился на клетки 30х30 см. Во время движений, поворотов, выравниваний и так далее робот постоянно собирает и анализирует показание сторонних датчиков (не дальномеры), что позволяет роботу оперативно реагировать на изменения в окружающей среде. Поворот же осуществляется при помощи гироскопа, располагающегося на самой meAURIGA, и состоит из самого поворота и доворота (выравнивания) для точного попадания в найденный проход. Для более точного перемещения в сильно замкнутых пространствах, используем пропорционально-дифференциальный регулятор, на основе показаний датчиков, что позволяет легко проходить подобные участки, не останавливаясь для выравнивания. Кроме того, при помощи гироскопа производится контроль над положением робота по оси OY, а именно, наличие ям или горок. Для более раннего обнаружения больших ям мы также используем датчик цвета, который способен определить яму еще до того, как центр тяжести перевесил в эту яму.

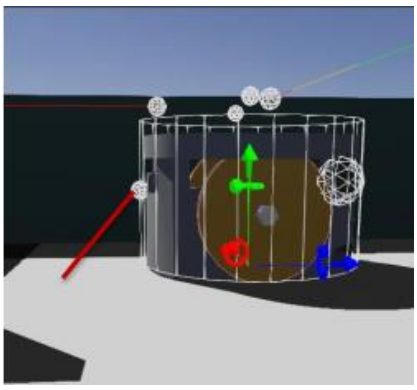


Рисунок 17 – модель робота в симуляторе

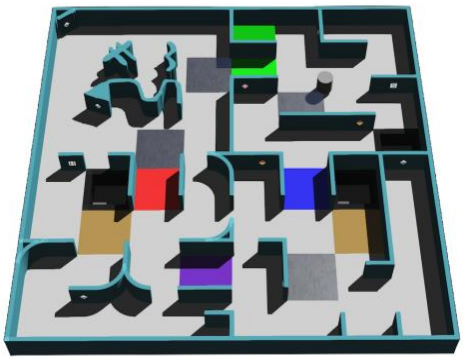


Рисунок 18 – модель полигона для миссии в симуляторе

Картографирование выглядит следующим образом: карта представляет собой довольно большой массив из структур. Каждая структура составит из нескольких однобайтных переменных, хранящих информацию о наличие препятствий на каждом из дальнометров, информацию о наличие жертв или иной информации для спасателей, а также количестве посещений этой ячейки. Для экономии памяти мы использовали по 2 байта на ячейку, так как для хранения информации о наличие препятствий, жертв, ям нам достаточно по одному биту памяти, следовательно при помощи побитогово распределения памяти нам удалось сократить её использование в 40 раз, что помогло увеличить максимальный размер карты.

Фрагмент кода, для создания основных элементов хранения карты на C++:

```
struct myNode{
        uint8_t wall=0; //1 байт памяти отвечающий за объектах внутри клетки
        uint8_t count=0; // 1 байт памяти для количества посещений клетки
};
myNode mymap[40][40]; // сама карта

int cord_x=15, cord_y=25; // контроль текучей координаты
int napravlenie=2; // контроль направления робота относительно старта

uint8_t tam_l, tam_r;// для расчета смежных направлений относительно ведущего
```

Таким образом, во время посещения роботом следующей клетки, робот обновляет информацию о ячейке и пересчитывает побочные переменные, необходимые для навигации. Сама же навигация робота строится на основе алгоритмов построения маршрутов в графе: DFS и алгоритм Дейкстры (блок схему можно прочитать в Приложении 4). Робот при посещении новой клетки смотрит, может ли он продолжать свое движение прямо, если нет, смотрит находится ли он на развилке, если вариантов много, он помечает на карте, что тут не дообследованный путь и продолжает свое движение, если же он в тупике, строит кратчайший маршрут до последней (ближайшей) не дообследованной развилки и продолжает его обследование уже с нее. Окончание мисси же может быть вызвано окончанием памяти, выходом из строя какого-либо компонента, или же зацикливанием карты (в данном случае будет считаться, что робот полностью обследовал помещение, рисунок 19).

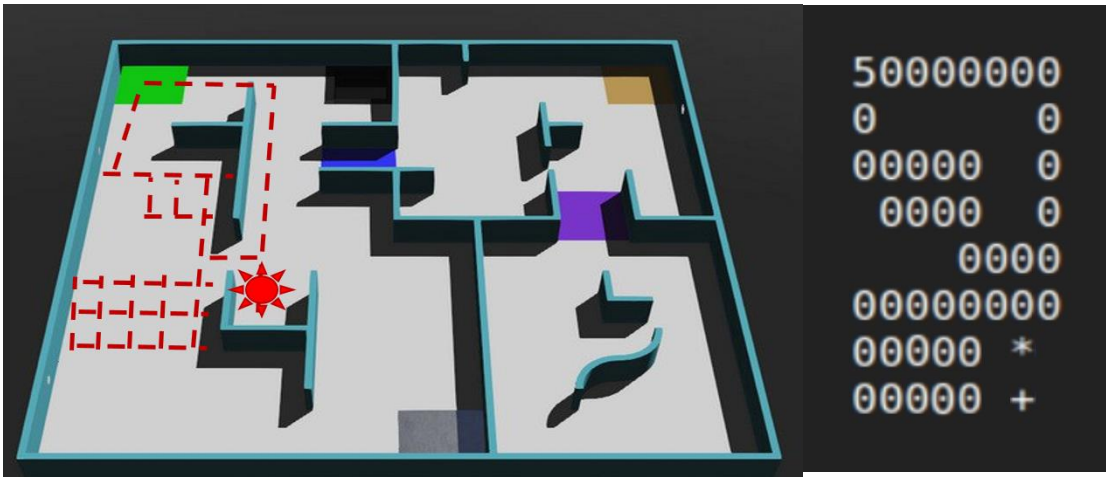


Рисунок 19- вариант вывода карты на экран (5-стартовая ячейка, * - робот, + - направление, в которое он смотрит)

Для программирования RPI, также были составлены несколько разных программ, которые позже собирались в одну. Так, для более точного определения возможных знаков или надписей на языке C++ был разработан алгоритм чтения букв, который мы назвали «метод точек», а для быстрого реагирования на изменения в окружающей среде был обучен фреймворк искусственного интеллекта YOLO v8, написанный на языке Python.

В регламенте соревнований роботов спасателей, требуют функции определения знаков опасности и чтения некоторых букв (рисунок 20). Для этого нам понадобилось проводить бинаризацию на цвета, характерные для знаков опасности и далее – само определение знаков/чтение букв. Однако мы не можем предсказать освещение во время миссии, поэтому нам понадобилось разработать алгоритм смены параметров автобинаризации в зависимости от освещения на поле. Алгоритм, который мы придумали ищет на изображении самый светлый, самый темный, среднестатистическое значение пикселя и формулой рассчитывает данные для бинаризации на черный (чем темнее освещение на поле, тем больше делителей). Таким образом, робот может находить линию даже в сумраке (рисунок 21, 22).

Рисунок 20 – знаки, определяемые нами

Примерная формула, которую мы используем на псевдокоде:

текущий_пиксель<(min+ср_знач)/2+ min

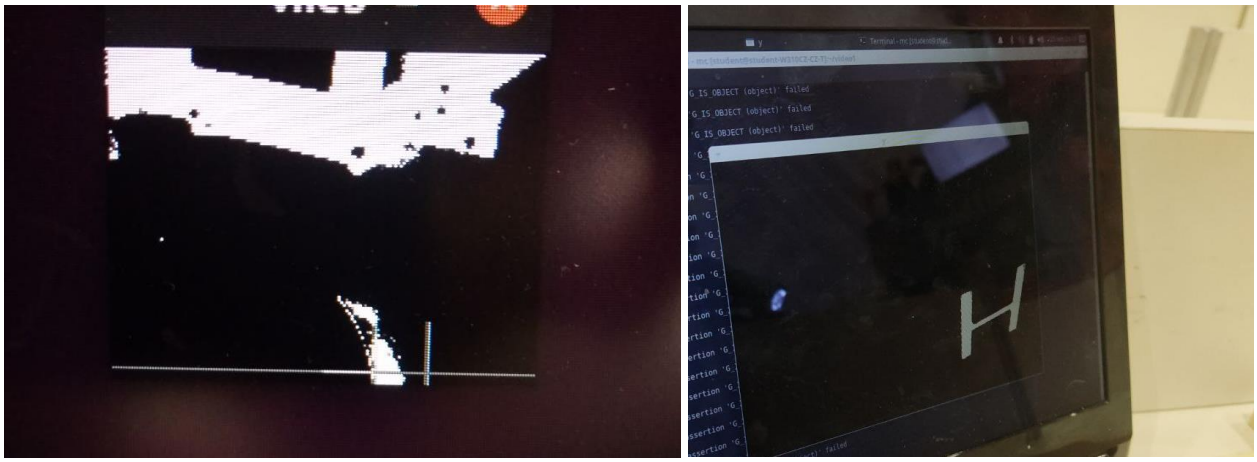


Рисунок 21, 22 – результат работы алгоритма автобинаризации

Само обнаружение же происходит с помощью камеры и библиотеки OpenCV 4. Для начала настраивается бинаризация (смотрится по количеству цветных пикселей) для одного из цветов: красного, жёлтого и чёрного. После бинаризации определяем контуры и проверяем самые большие контуры, подходят ли они под параметры знаков. Если обнаружены чёрные контуры, подходящего размера, то вычисляем 3 точки прямоугольника этого контура: верхнюю срединную и нижнюю срединную, с помощью них мы и определяем какая буква и ищем в них контур на чёрной бинаризации: если верхний и нижний - черные, то это s, если нижний и верхний - не черные - то это H, а если нижний – чёрный - то это U (рисунок 23, 24). Как можно заметить, данный метод основывается на различие начертания букв. Сейчас нам достаточно всего 2-х точек, так как мы читаем всего 3 буквы, однако с увеличением количества букв, необходимых для чтения нам достаточно просто увеличить количество точек и их расположение.

Рисунок 23 – «метод точек» наглядно

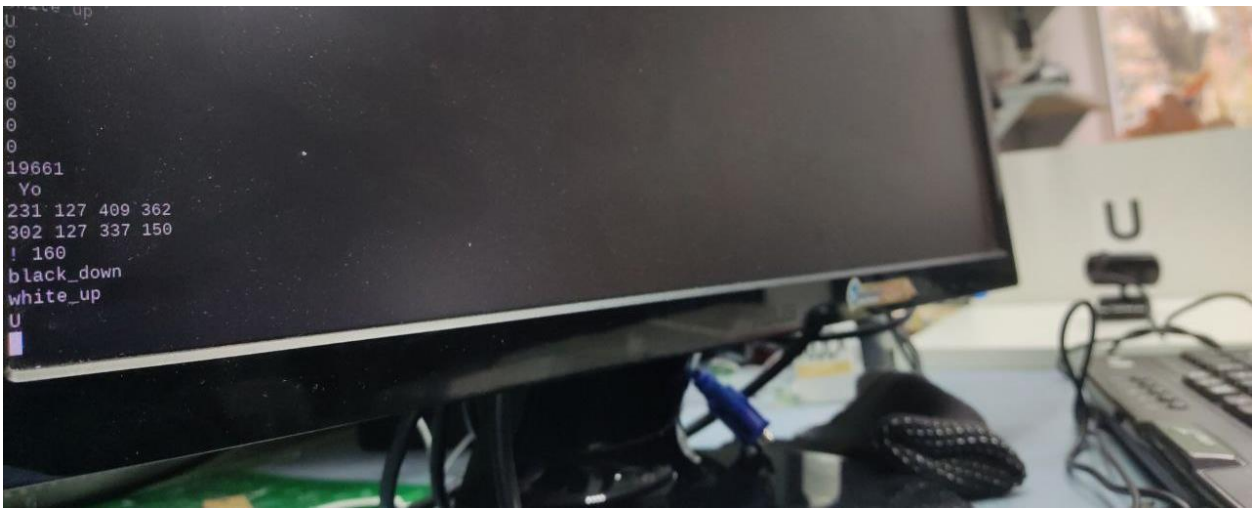


Рисунок 24 – результат работы алгоритма «метод точек»

Еще одним вариантом распознавания букв, а также вариантом распознавания жертв стало обучение фреймворка искусственного интеллекта YOLO v8 (рисунок 25). Механизм работы нашей программы предельно простой: у нас имеется заранее натренированная модель на датасете поля, преобразуем каждый кадр в тензор, сравниваем с тензора и в модели, получаем боксы детектирования, если условный "коэффициент сходства" в подобном боксе больше определённого значения, то узнаем, что за жертва/буква привязана к нему. Для реализации этого алгоритма собственно нужно было подготовить сам датасет: снять фото с камеры робота, и с помощью спец инструментов провести разметку. Так как мы использовали нейросеть YOLO v8, то и тренировали мы модель YOLO v8 (n/m/x) на нашем датасете. Далее экспортировали в расширение torchscript, передавали файл распберри, т.к. программируем RPI на C++, а YOLO обучается на Python - использовалась библиотека libtorch.

Рисунок 25 – итог работы алгоритма
распознавания на YOLO v8

Тестирование робота

Перед тестированием самого робота на каждом этапе его разработки проводились тестирования каждого элемента ПСР. Так, конструкция в первую очередь тестируется на прочность и если ее недостаточно, то элементы дорабатываем. Таким образом в результате таких тестов мы заменили пластиковые и картонные детали на металлические и фанерные. При тестировании датчиков также приходилось проверять, что кабели не мешают их обзору, в результате чего и появился короб для проводов. Кроме того, при разработке шасси нам приходилось постоянно тестировать его проходимость - забираться на горки, проезжать через лежачие препятствия разной сложности. В результате у нас есть робот с хорошей проходимостью.

Так как для подключения большей части датчиков к meAURIGA нам приходилось паять, основным тестированием датчиков было проверка на отсутствие короткого замыкания. Кроме того, перед непосредственно пайкой и подключением, нам было необходимо отдельно проверять датчики при помощи Arduino Uno на работоспособность.

Тестирование ПО, вне симулятора производилось непосредственно на поле при выполнении миссии. Для обучения YOLO мы запустили робота и сохраняли кадры с камеры на роботе раз в 0,5/1 секунд. А позже для тестирования распознавания жертв и букв мы использовали распечатанные буквы, так как это удобнее, чем на поле.

Тестирования прохождения самой мисси проводили на полигонах для соревнований макетов роботов спасателей. Полигоны собраны из ЛДСП и симулируют выполняемую миссию (рисунки 26,27,28), также на полигонах можно встретить препятствия, мусор, подъемы-спуски, рампы и лестницы. Робот же выполнял миссии в нескольких режимах: с передачей информации людям, и обособлено, без вмешательств, так как нам не интересны показания датчиков, нам было достаточно подключить планшет к RPI и отслеживать, что она пишет в командную строку. Таким образом мы можем оперативно получать информацию о нахождении жертв прямо на планшет.

Видеозаписи большей части испытаний можно посмотреть на нашем облачном диске (см. ссылки на авторов).

Рисунок 26, 27 – примеры полей для испытаний

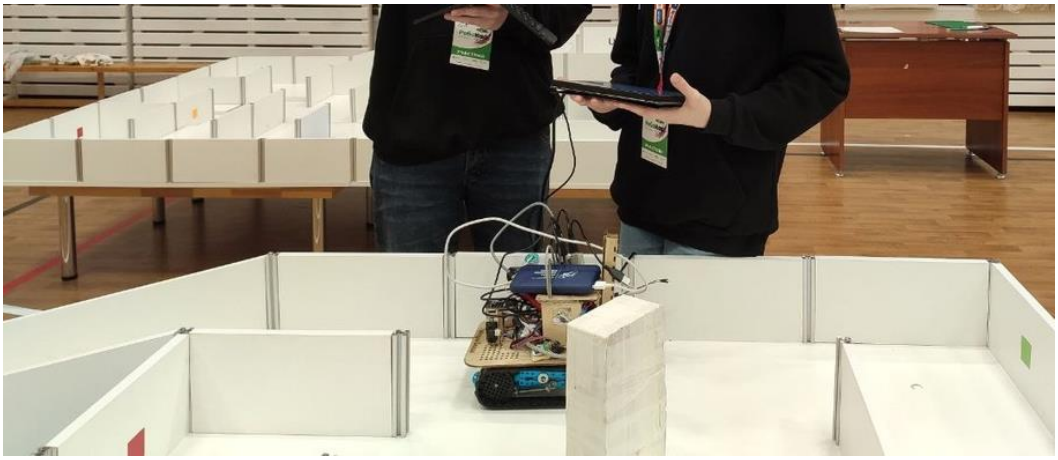


Рисунок 28 – отладка робота на поле

Таким образом, после доработки промежуточных результатов разработки на кождом из этапов, были пройдены несколько этапов разработки и итераций робота (рисунки 29-34), получила и протестировала рабочий эксперементальный прототип робота для помощи проведения поисково-спасательных операций (рисунок 38).

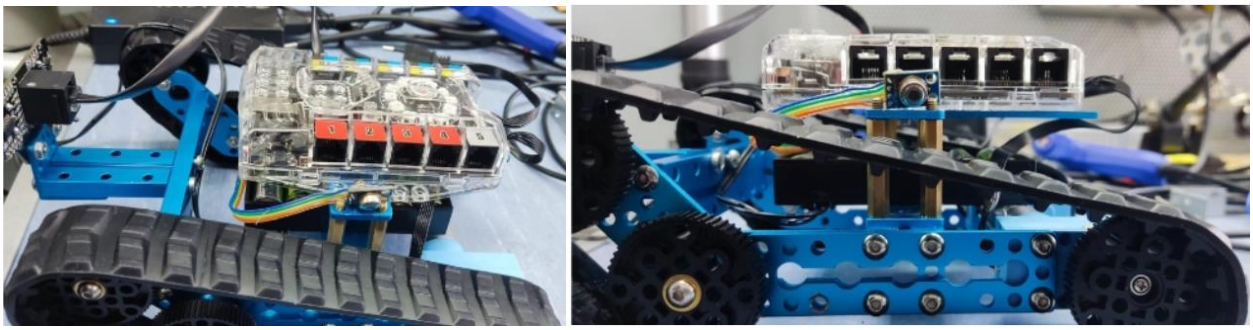


Рисунок 29,30 – первая версия робота

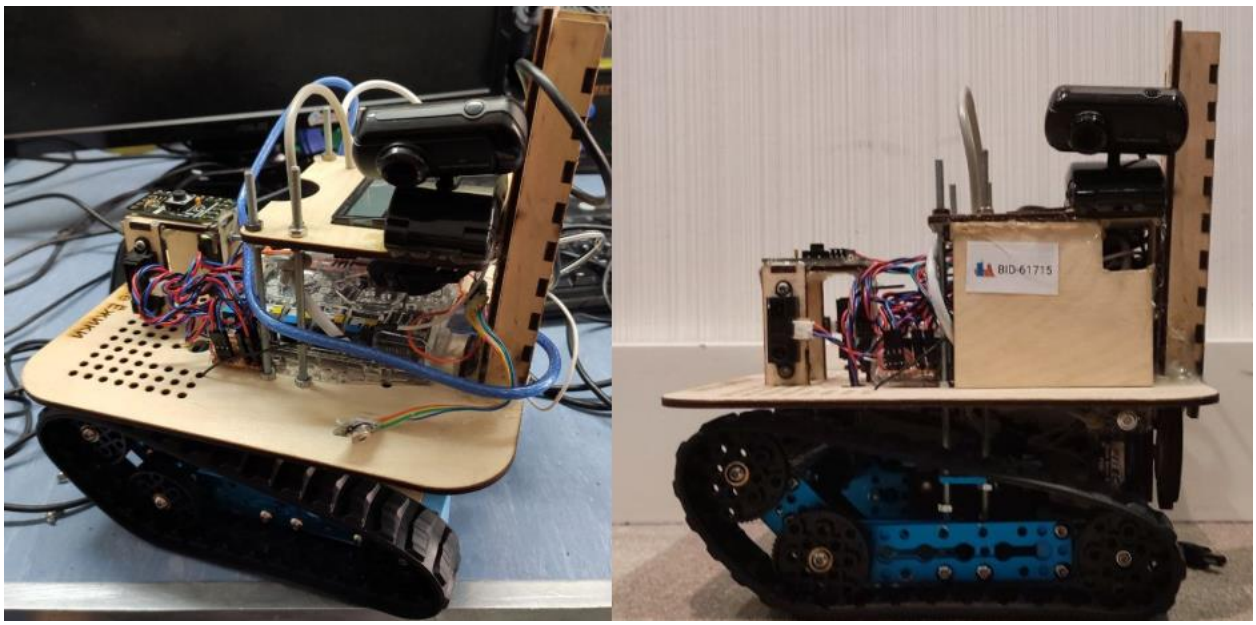


Рисунок 31,32 – вторая версия робота

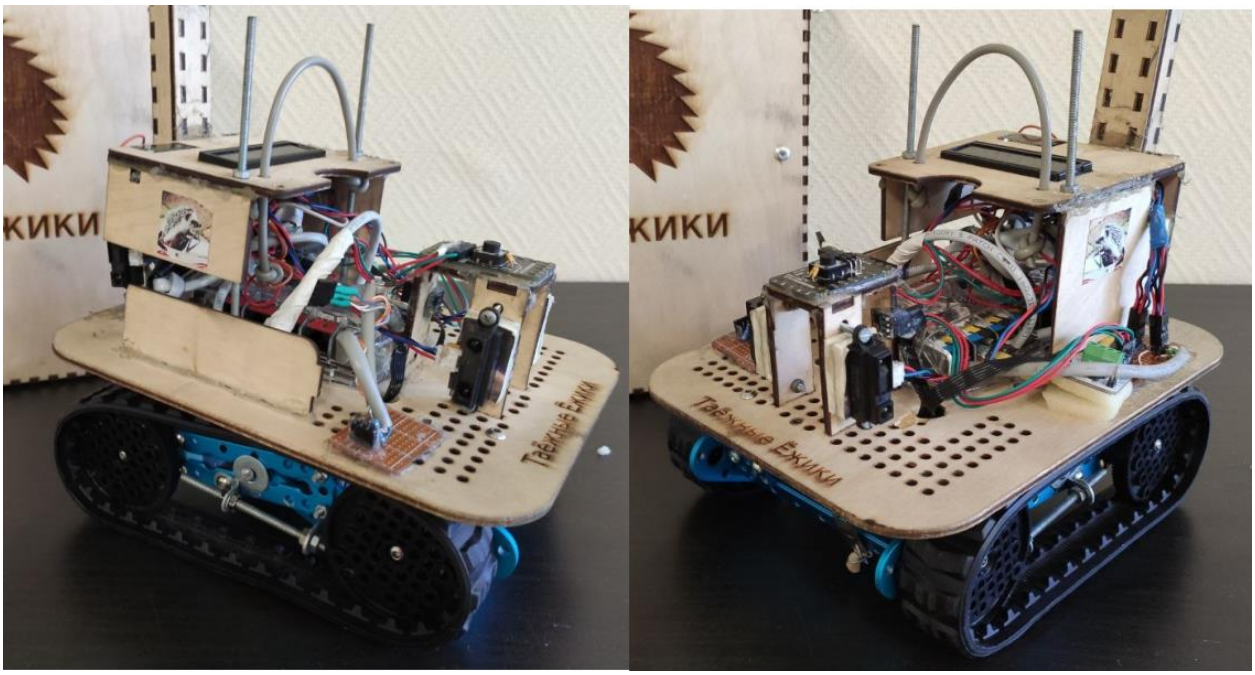


Рисунок 33,34 – третья версия робота

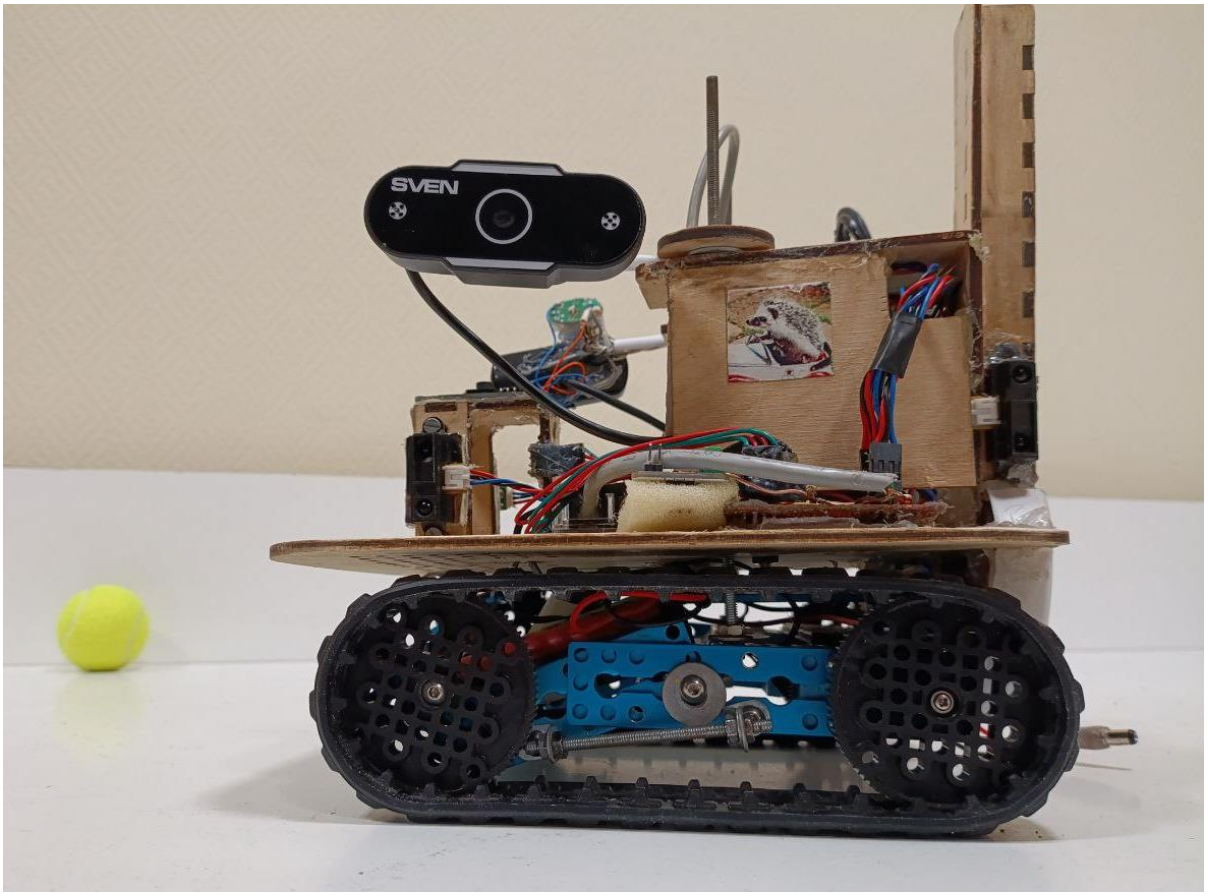


Рисунок 35 – итоговый прототип робота для проведения поисково-спасательных операций

ЗАКЛЮЧЕНИЕ

Достигнутые результаты

В результате проделанной работы желаемый результат был достигнут, был разработан и протестирован рабочий прототип робота для проведения поисково-спасательных операций. Данный робот способен в автономном режиме обследовать территорию, строить его карту и определять наличие знаков опасности.

Проблемы, с которыми мы столкнулись на этапе реализации

Во время разработки и модифицирования нашего робота команда столкнулась со следующими проблемами.

1) При разработке чертежей робота, необходимо было сразу предусмотреть все необходимые отверстия и крепежи, из-за постоянной модификации конструкции, необходимо заново переделывать их. (основной платформы у нас 3 варианта, последнюю из которых также пришлось модифицировать).

2) Освещение на тестовых полях было очень плохим и к тому же менялось с течением суток, настройка бинаризации очень долгий процесс, из-за чего было принято решение разработать самодельный алгоритм автобинаризации.

3) Изначально мы передавали данные с Raspberry на meAURIGA через Serial и по какой-то причине Raspberry пыталась питаться через meAURIGA, а не через пауэрбанк, из-за чего сгорала Raspberry. В результате было принято решение перейти с serial на UART. А позже – и на логические сигналы. Возникшая проблема была решена.

4) Зачастую у Raspberry сбивается время, и программа не компилируется, выводя ошибку «Время изменения файла находится в будущем», для решения данной проблемы мы копируем новый каталог под новым временем и сносим старый с неправильным.

5) Библиотеки MeAuriga.h. Мы не могли найти подходящие команды для программирования робота. Мы провели анализ и разбор библиотеки и смогли определить необходимую команду.

6) Для масштабной и подробной карты необходимо много памяти, которой сильно не хватало. Решением стало использование методов побитовых операций и сжатие переменных до 1 байтных.

7) Raspberry постоянно поедает sd карты, они ломаются и порой даже не поддаются форматированию, мы не смогли до конца решить данную проблему, поэтому просто делаем бэкапы и заменили microSD на жесткий диск

8) Обучение нейроситей, оказалось довольно удача затратным занятием, так как очень сложно предугадать, сколько ресурсов вам понадобится, чтобы нейросеть обучилась,

но не переобучилась. В связи с чем было сделано множество вариаций этой нейросити и со временем научились это высчитывать.

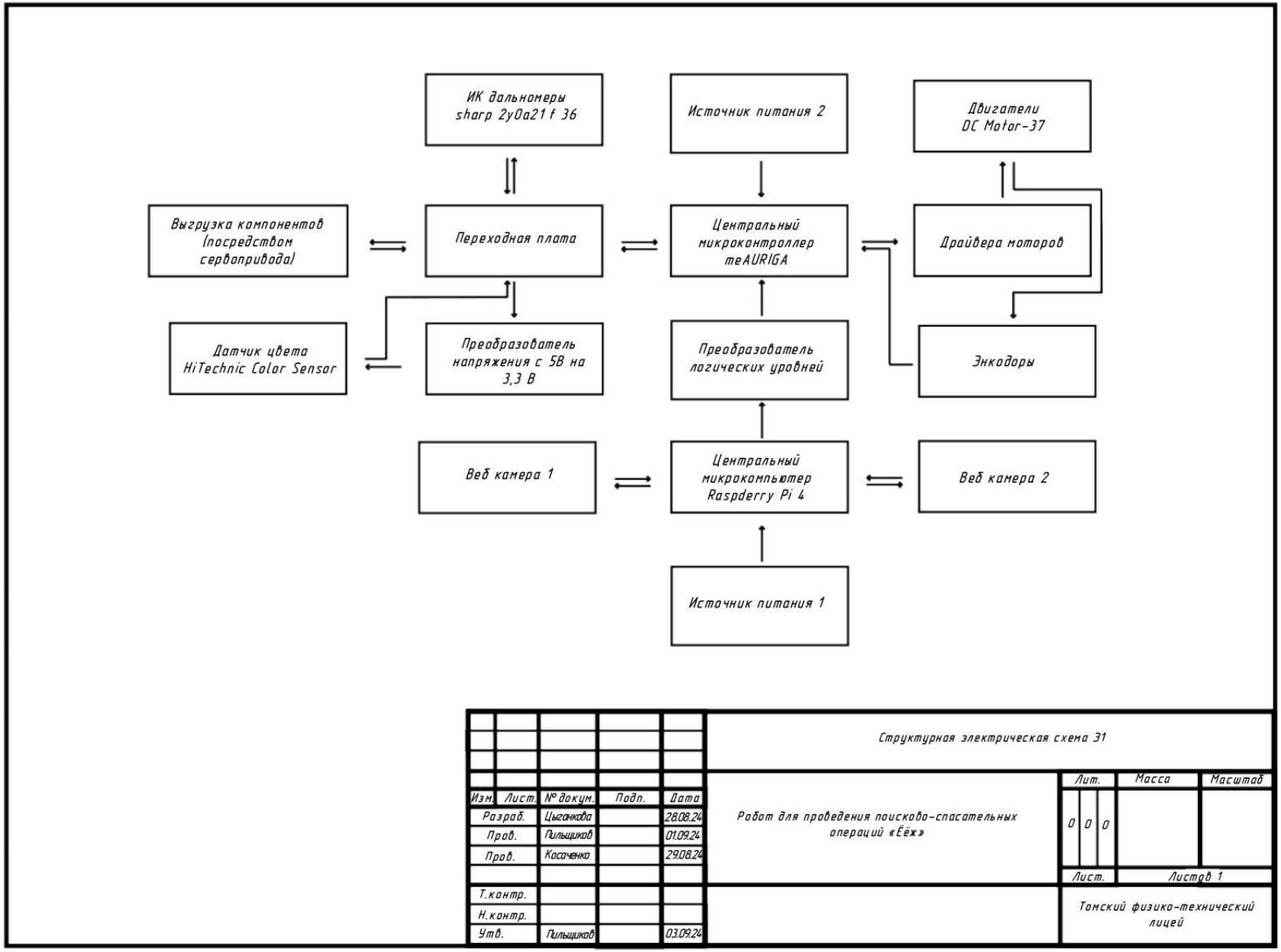
9) Технологии машинного обучения тратят довольно много ресурсов, к тому же мы не отказывались от метода точек, в связи с чем возникла проблема с использованием Raspberry Pi Zero и произошёл переход на Raspberry Pi 4.

Планы на дальнейшую доработку

Для данного проекта был закуплен датчик– лидар. В связи с чем уже ведется разработка корпуса под него, а также модернизация алгоритма навигации, так как теперь по сути мы можем анализировать не 5 лазеров, а все 360 и данная модернизация позволит улучшить качество построенных нами карт. Также планируется продолжать обучать нейросеть для определения человека и степень его повреждения при ЧС.

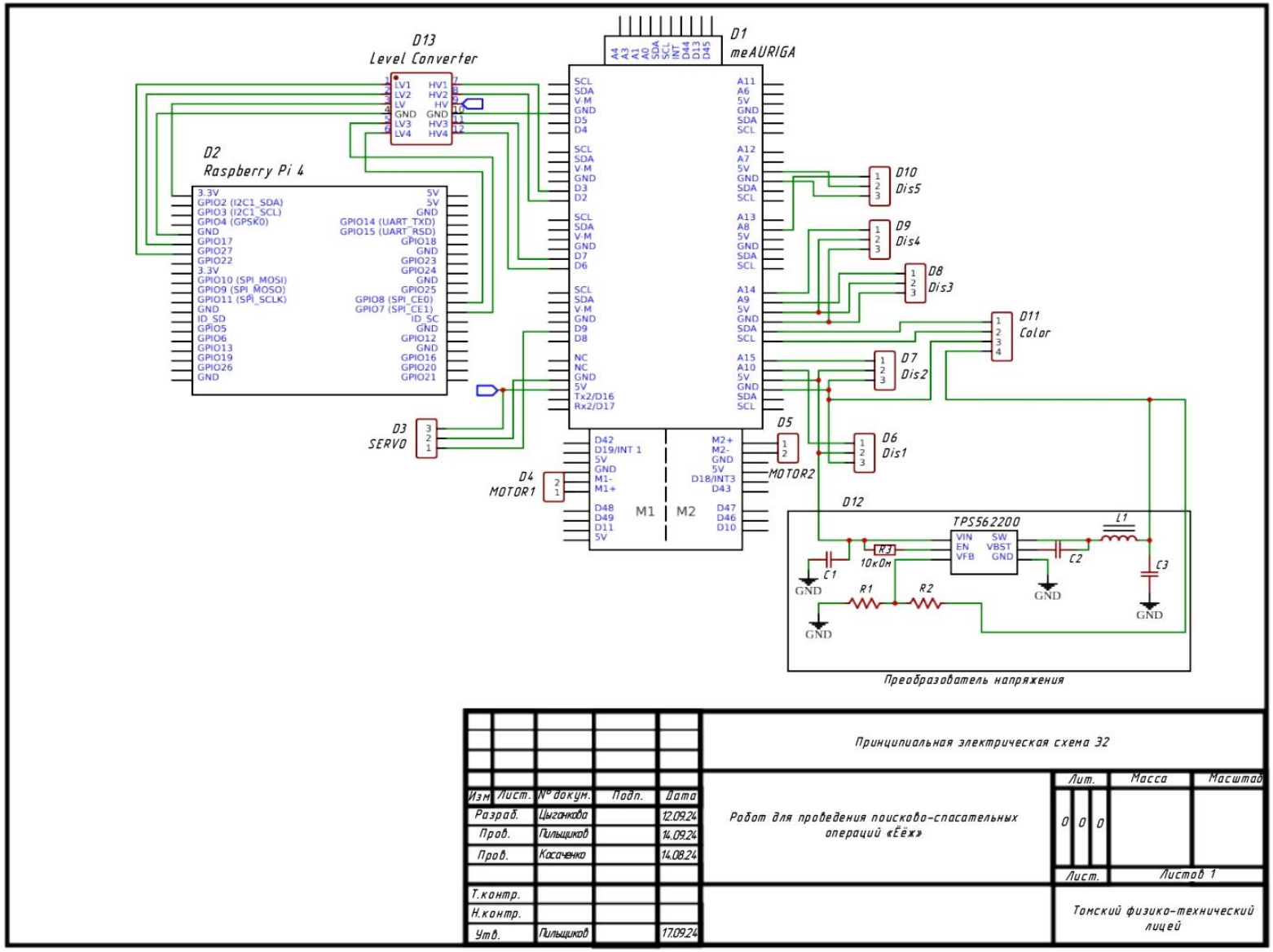
Приложение 1

Структурная схема Э1



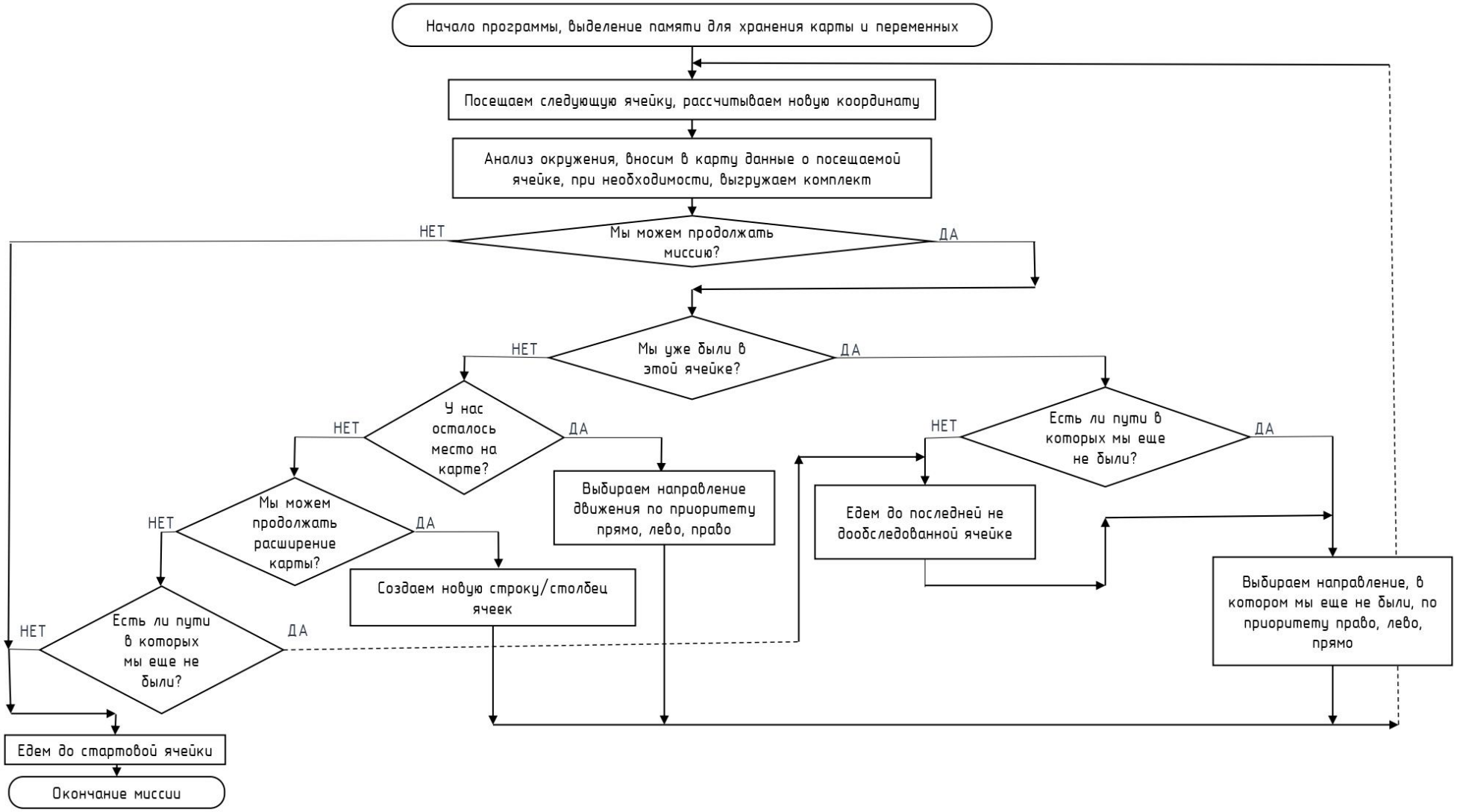
Приложение 2

Электрическая схема Э2



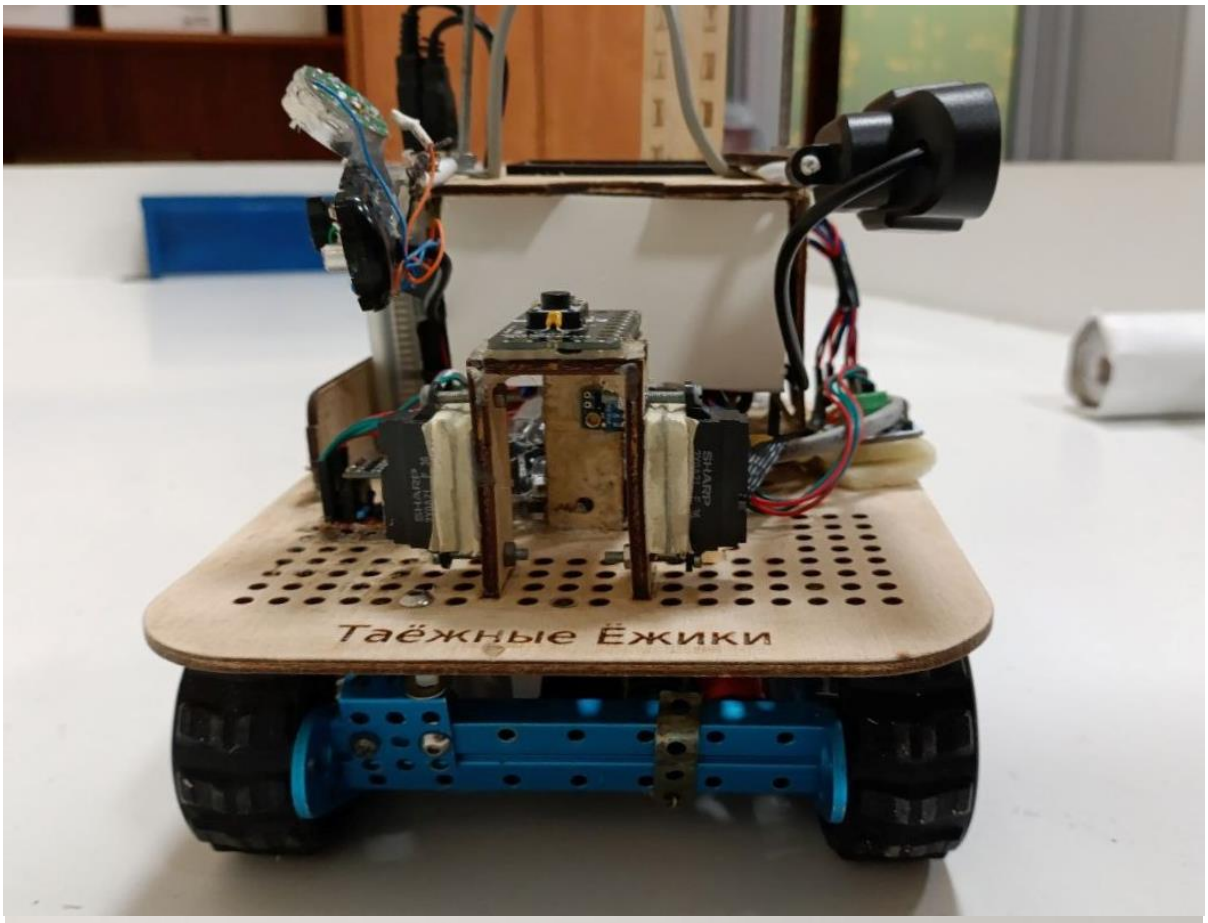
Приложение 3

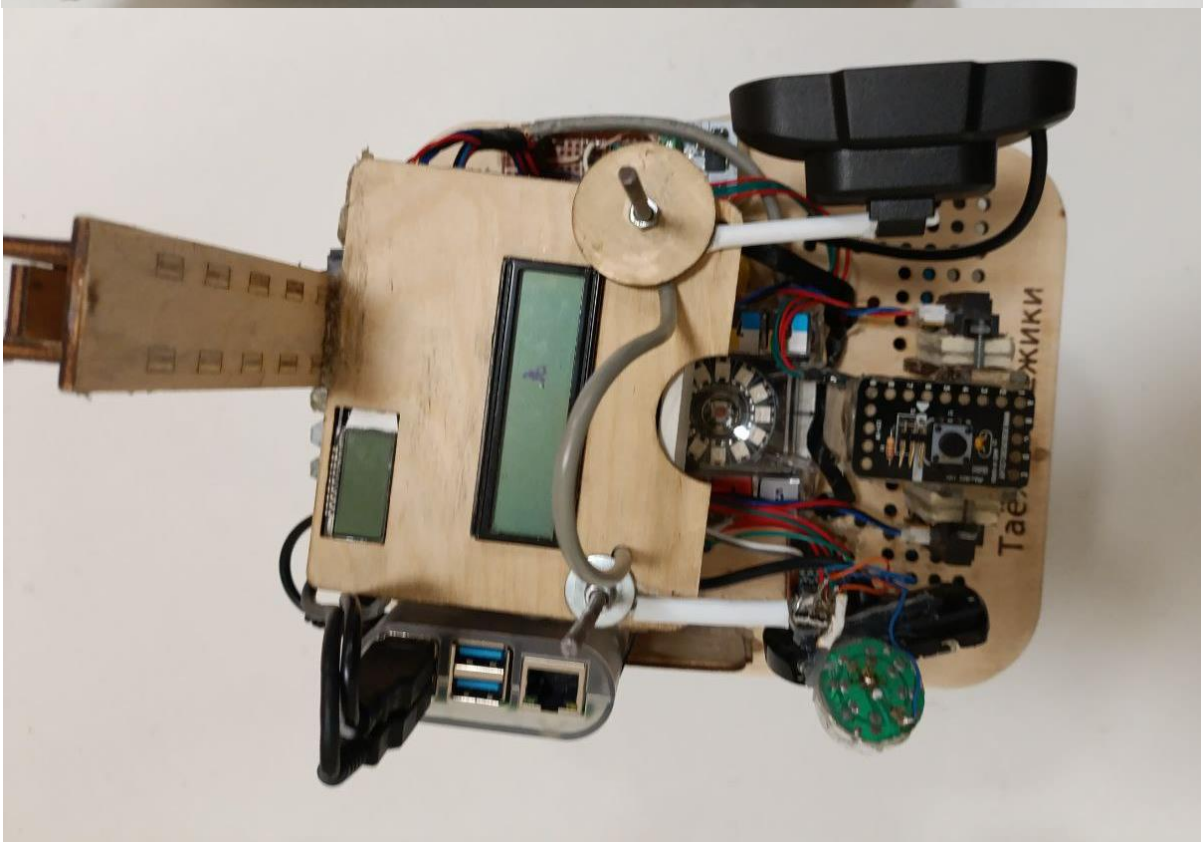
Блок-схема алгоритма навигации

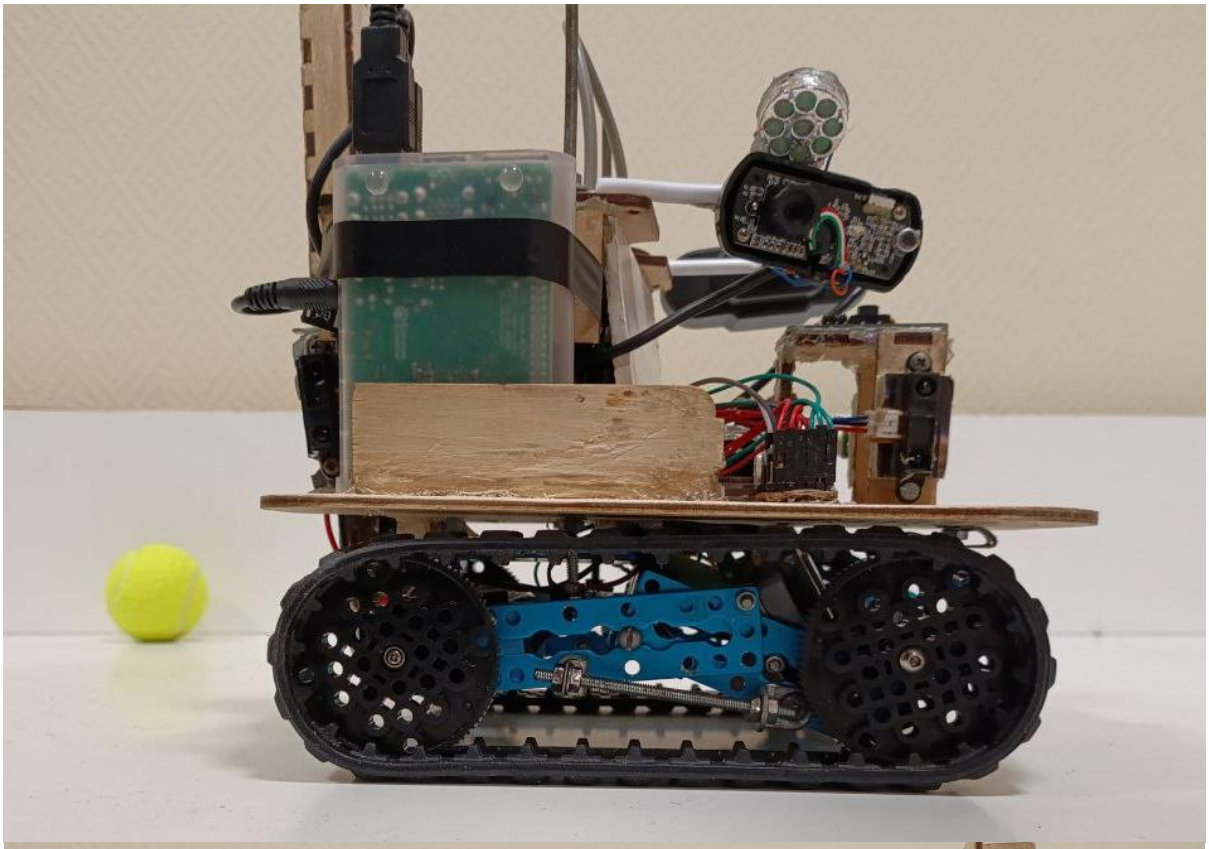


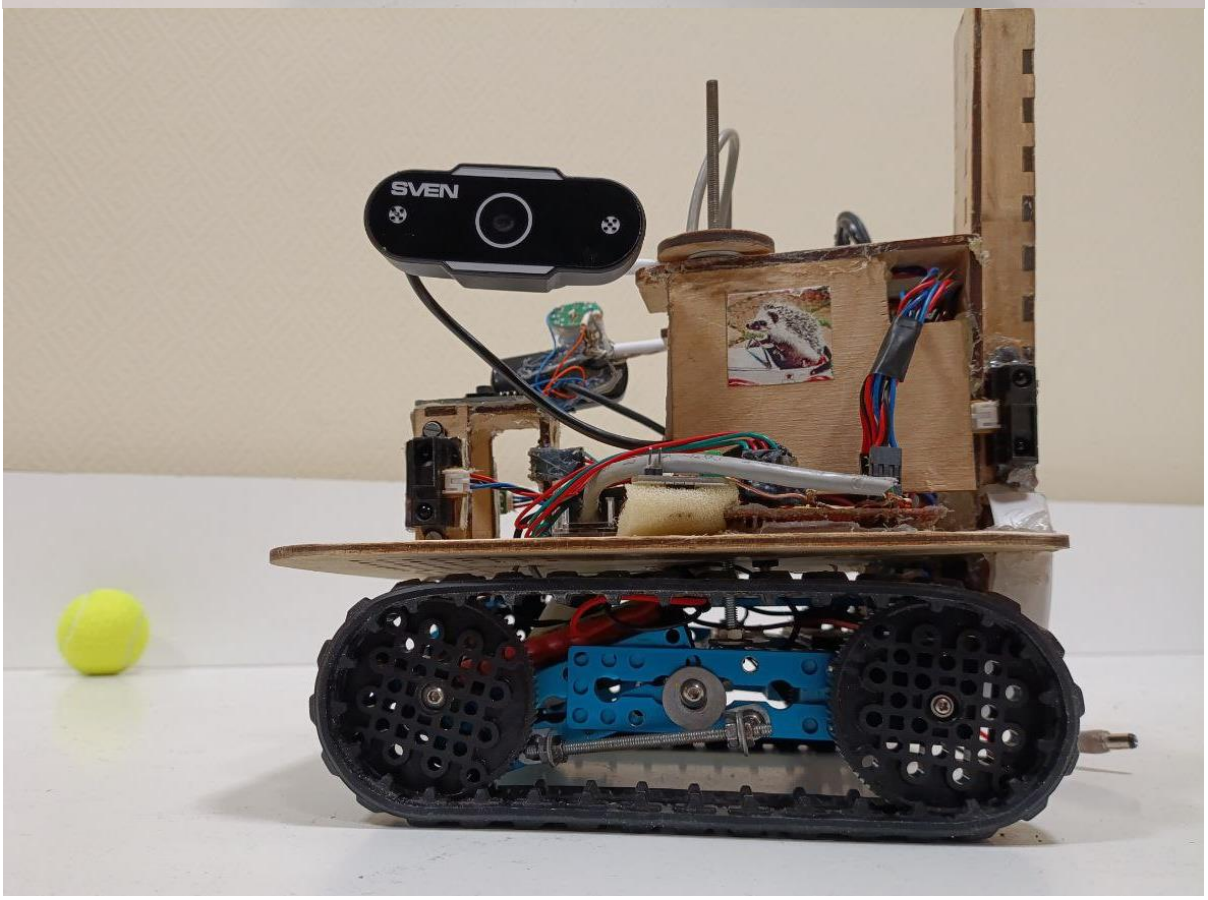
Приложение 4

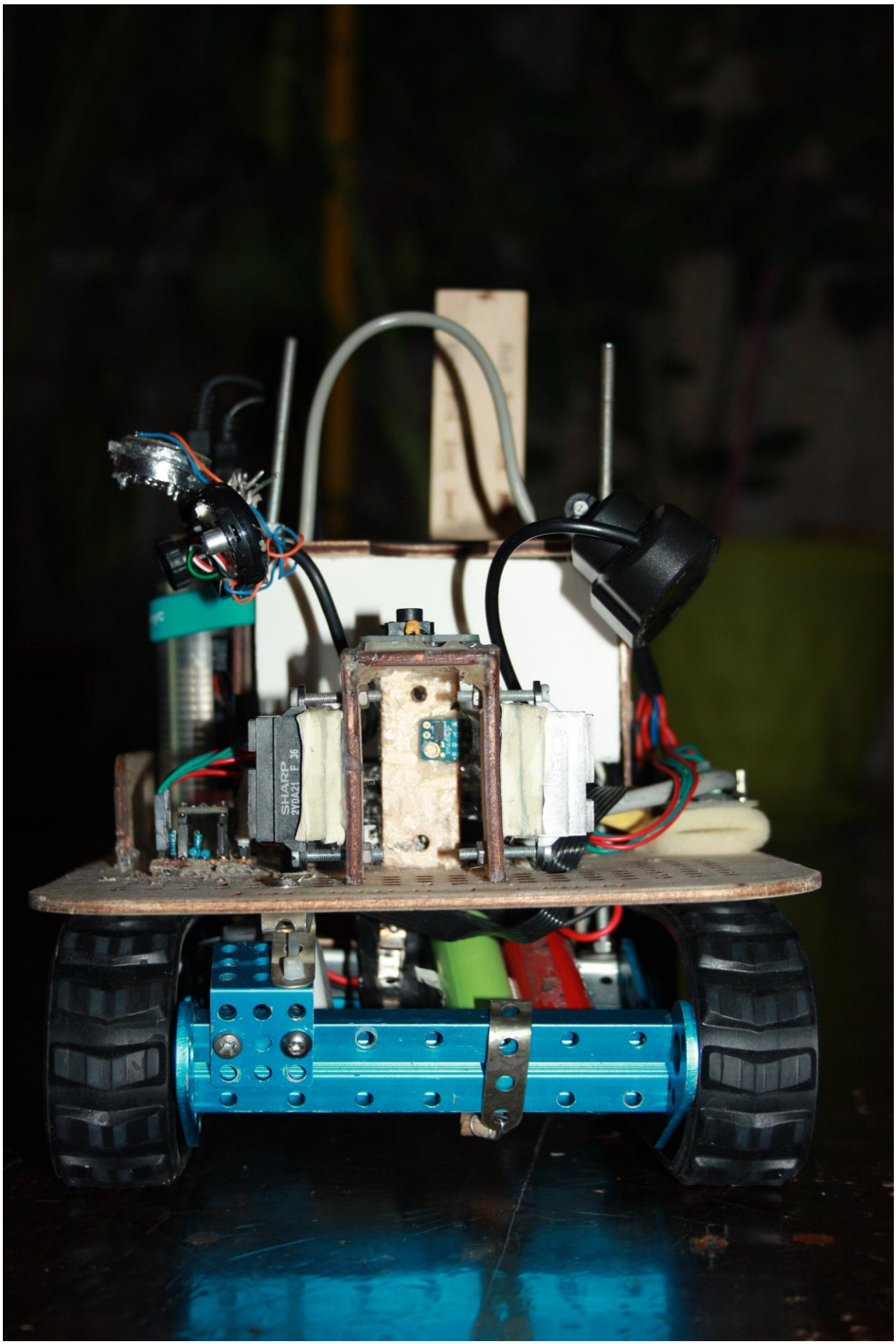
Фото итогового прототипа ПСР

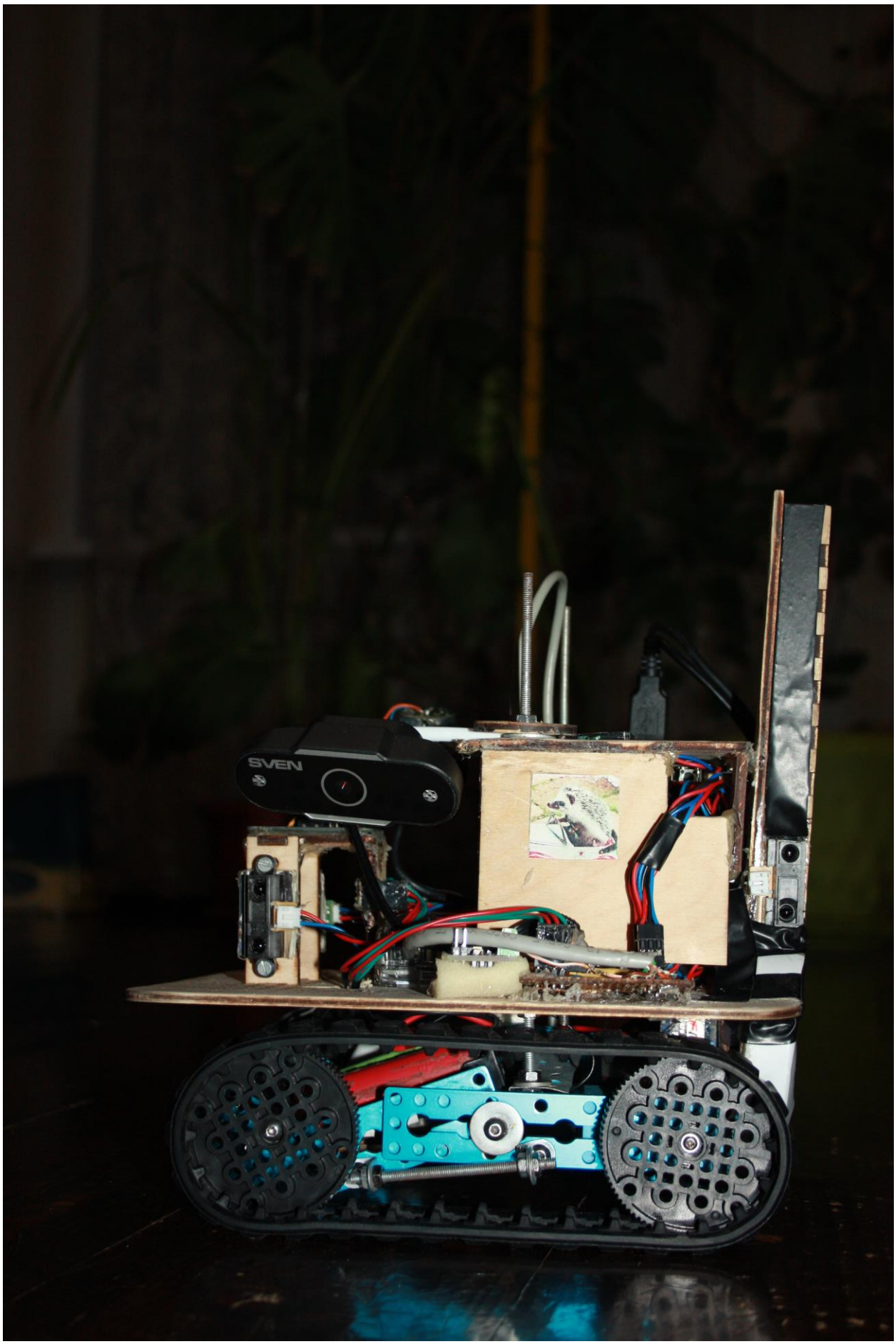


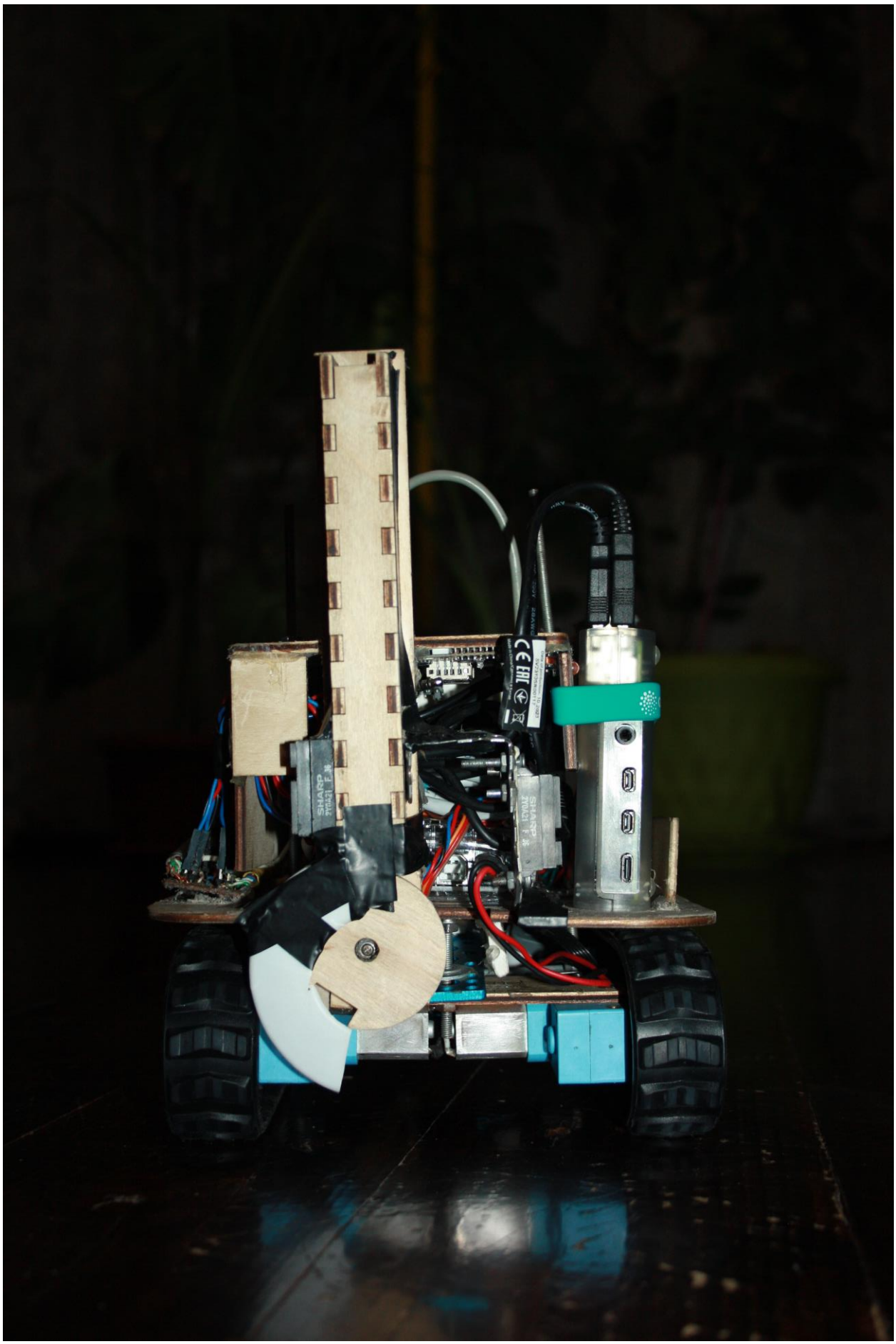


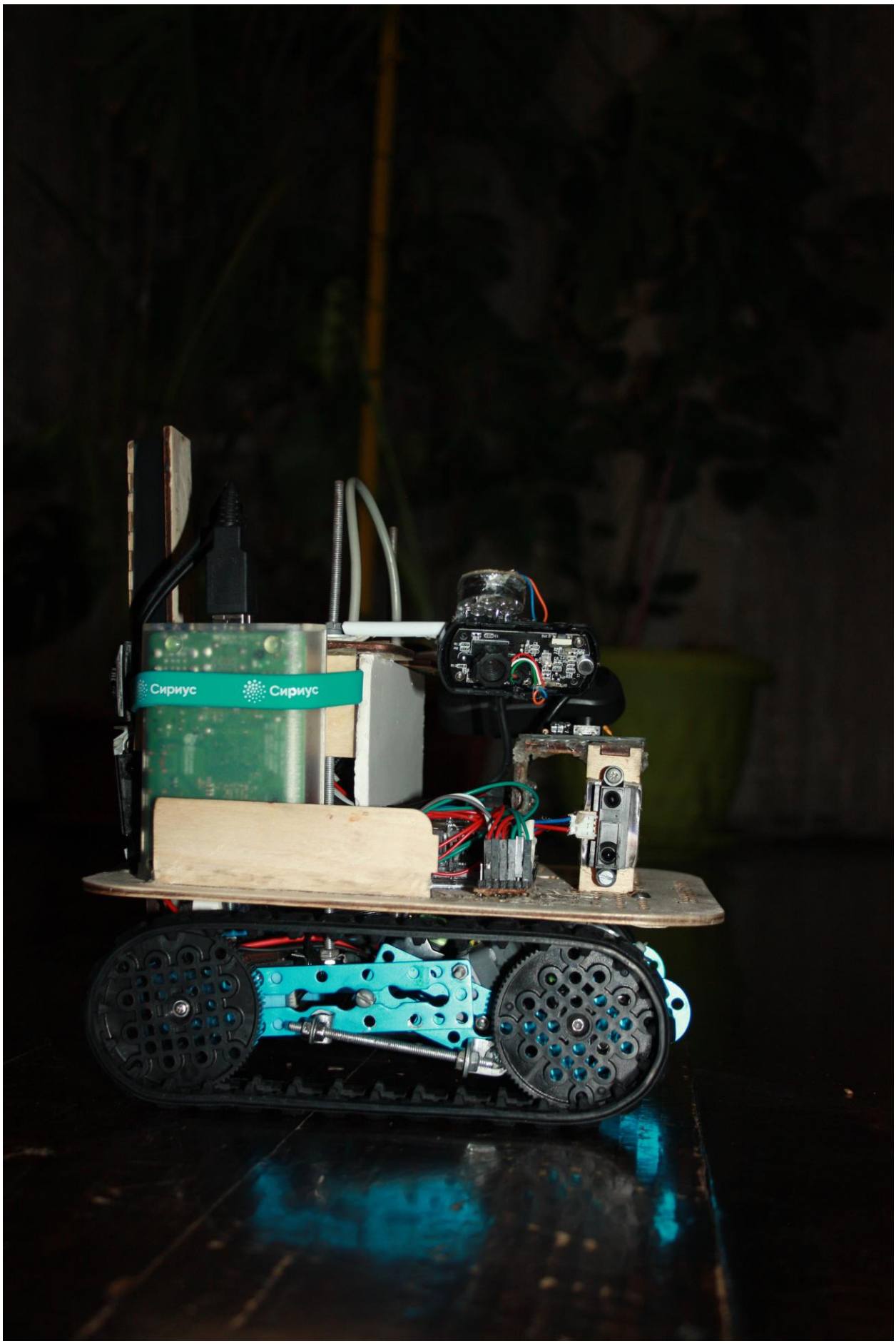



SVEN


SHARP
2Y0A21 F 36


SVEN


SHARP
SHARP


Сириус
Сириус

Ссылки на нас

Здесь вы можете подробнее прочитать про этот и другие проекты команды «Таёжные Ёжики», изучить чертежи, код, датасеты, а также посмотреть на все испытания и тесты робота.

https://github.com/Tsygankova-Maria/TE2022/tree/main (наш GitHab)

https://drive.google.com/drive/folders/1fmRRXI1r2HDXeKdHgOfJ3aFNQoEvYZH1?usp=sharing (облачный диск, где собираются видео испытаний робота)

https://drive.google.com/drive/folders/1vHIvbDYvG99-oNjlotWKDVUvc7Kxek05?usp=sharing (облачный диск, где собирается техническая документация робота)

https://www.youtube.com/@user-cj5jf6zu6l (наш YouTube)

https://www.youtube.com/@SergKosachenko (YouTube нашего тренера, здесь можно посмотреть некоторые записи испытаний)

Мы каждый год публекуем свои материалы и будем рады помочь Вам при подготовке!

Контактные данные авторов

М.С. Цыганкова: maria.tsygankova.tftl@gmail.com

Г.А. Пильщиков: p.g.a.2019@mail.ru

С.В. Косаченко: kosachenkosv@yandex.ru