Техническое описание робота

Состязание: Спасатели лабиринт Старшая RoboCupJunior Rescue Maze Secondary
Название команды: Таёжные Ёжики
Участники команды: Пильщиков Григорий, Цыганкова Мария
Тренер команды: Косаченко Сергей Викторович
Организация: ОГБОУ «Томский Физико-Технический лицей»

Оснивное содержание

Анотация: Цыганкова Мария и Пильщиков Григорий разработали и модифицировали робота длякатегории RCJ Rescue Maze, а также алгоритм движения и распознавания для выполнения поставленных целей.

Участники команды (слева направо): Пильщиков Григорий, Косаченко Сергей, Цыганкова Мария

Роли участников команды: Пильщиков Григорий и Цыганкова Мария конструируют и программируют роботов вместе.

Опыт участия и успехи команды в различных робототехнических соревнованиях и фестивалях

Международный фестиваль Робофинист 2022 — 3 место в категории Robocup Junior Rescue Maze, г.Санкт-Питербург
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
XII Региональная олимпиада по образовательной робототехнике школьников Томской области 2023 — 1 место в состязании Robocup Junior Rescue Simulation Webots Erebus

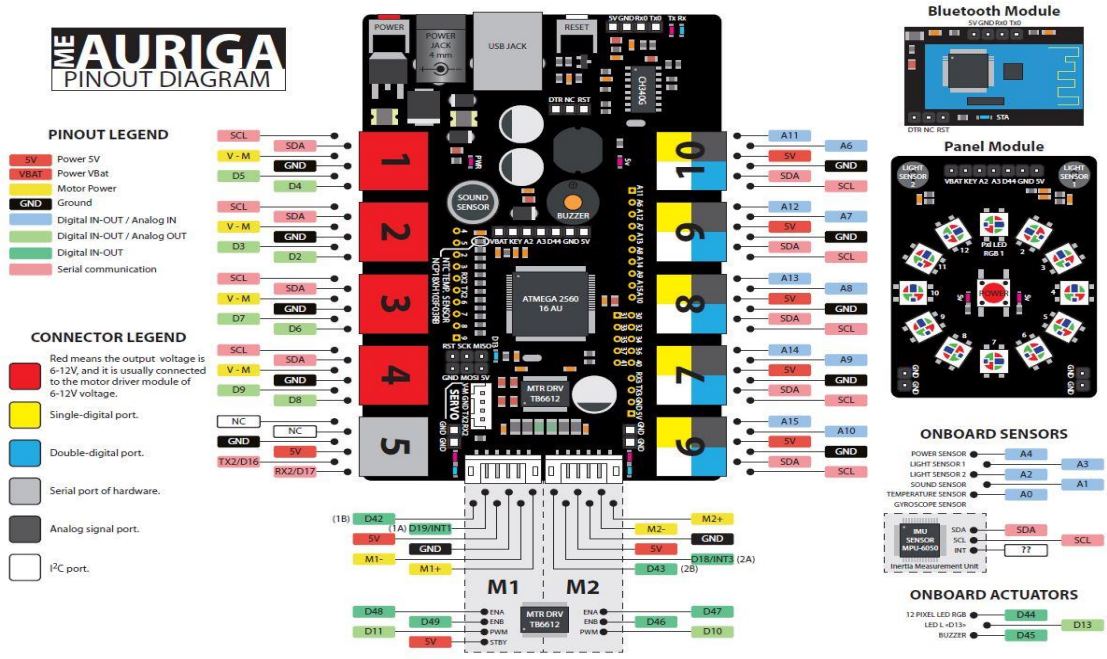
Откратый Российский чемпионат по робототехнике 2023 — 2 место в состязании Robocup Junior Rescue Simulation Webots Erebus
2022 год — Участие Цыганковой Марии и Пильщикова Григория в фестивале подводной робототехнике «АкваРобоФест» в г.Асино

Описание Робота

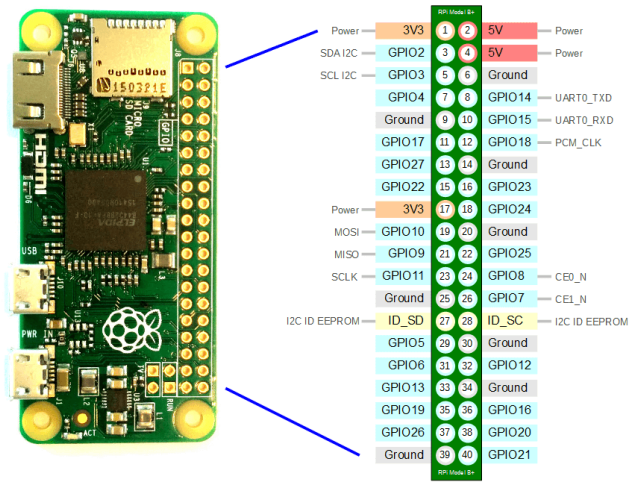
Стратегия выполнения задания роботом:
Наш робот ищет в лабиринте “жертв” по правилу "чередующийся руки". В момент запуска случайно выбирается ведущая рука для прохождения лабиринта, по которой в дальнейшем двигается робот. Во время движения робот анализирует стену: если он находит черный цвет, проверяет какую букву нашел, если нашел цвет, считает, что нашел цветную жертву. Кроме того, во время движения робот анализирует цвет тайла на котором находится для определения ямы и болота.

Использование датчиков:
В нашем роботе используется 5 инфракрасных дальномера sharp 2y0a21 f 36, HiTechnic NXT Color Sensor V2 и камера, считываемая Raspberry Pi Zero, которая передает данные на meAURIGA, кроме того у нас есть встроенные датчики считывания угловой скорости (энкодоры) с помощью которых управляем моторами из набора MakeBlock. Данные с Raspberry на meAURIGA мы передаем по UART для этого мы самостоятельно спаяли переходную плату, так называемый UART hub с Raspberry на порт MakeBlock RJ12, для подключения моторчика для поворота камерой мы также спаяли переходную плату с портов MakeBlock RJ12 на разъемы с шагом 2.45 мм. Кроме того, на роботе стоит стабилизатор питания с 5 В на 3.3 В. Для более удобного старта на роботе вынесена кнопка запускающая самого робота. Распиновку meAURIGA и Raspberry Pi Zero, мы брали с официальной тех. Документации.

Распиновки, которыми мы пользовались и самодельные платы робота



Распиновка meAURIGA

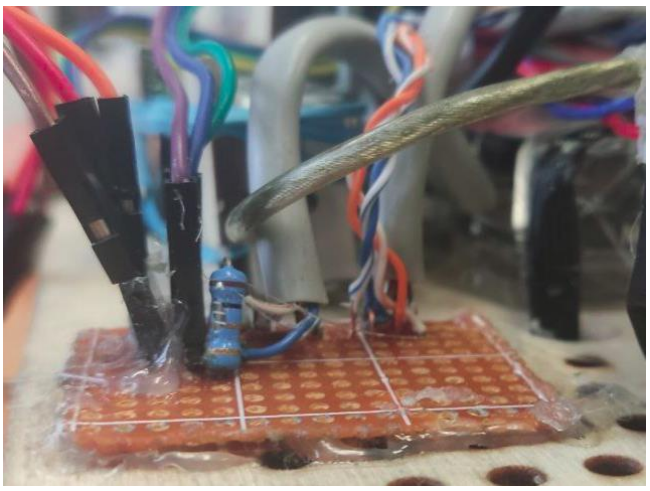


Распиновка Raspberry Pi Zero,

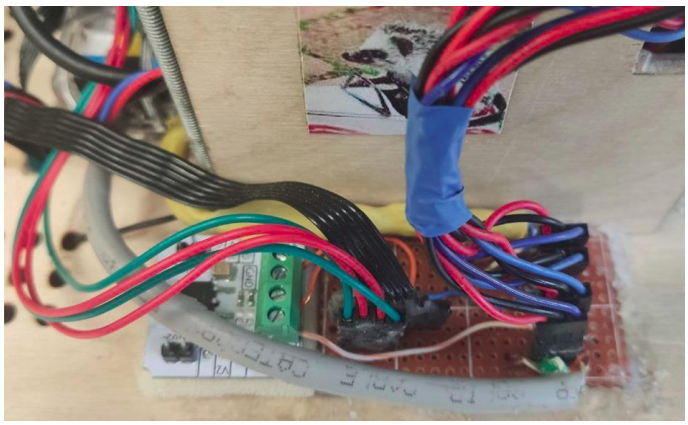
Pin	Name	Function	Color
1	ANA	Analog interface, +9V Supply	white
2	GND	Ground	black
3	GND	Ground	red
4	IPOWERA	+4.3V Supply	green
5	DIGIAI0	I²C Clock (SCL), RS-485 A	yellow
6	DIGIAI1	I²C Data (SDA), RS-485 B	blue

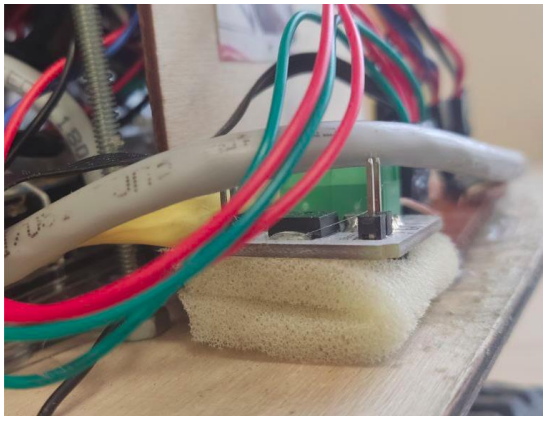
Распиновка порта MakeBlock RJ12

Инфракрасный дальномер sharp 2y0a21 f 36



Самодельный UARThub





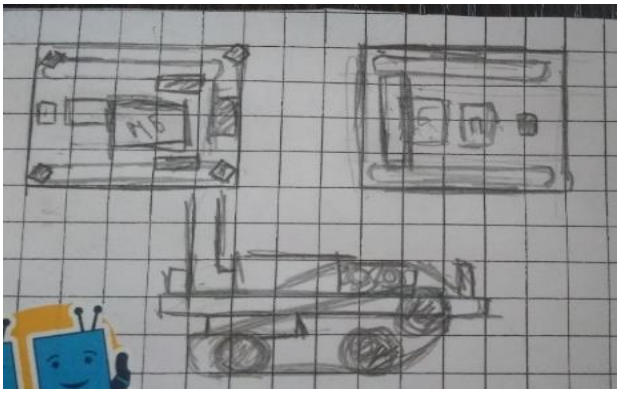
переходная плата с портов MakeBlock RJ12 на разъемы с шагом 2.45 мм, стабилизатор питания с 5 В до 3.3 В

(больше распиновок и схем можно найти на нашем GitHub)

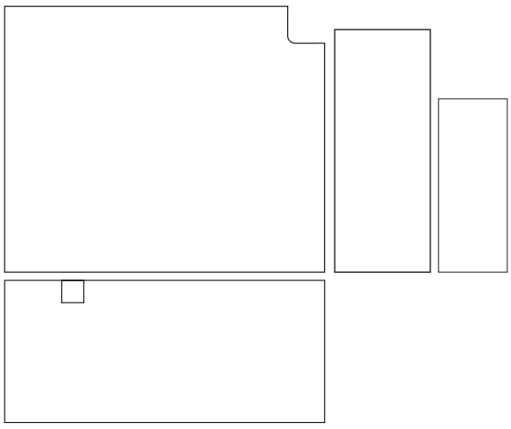
Конструкция робота:
Конструкция робота предполагает несколько этажей с разным функционалом: на нулевом располагаются моторы и Color Sensor, на первом — аккамулятор для meAURIGA и пауэрбанк для Raspbberry, на втором — сама meAURIGA и Raspbberry, USBhub, самодельный UART hub, переходная плата с портов MakeBlock RJ12 на разъемы с шагом 2.45 мм, стабилизатор питания с 5 В до 3.3 В и инфракрасные дальномеры, на третьем — два экрана для вывода информации и кнопка для быстрого запуска и сама камера. В пространстве между 2-ым и 3-ем этаже робота располагается коробочка для проводов и meAURIGA.
Сначала мы начертили все необходимые детали в векторном формате (в программе inkscape), после чего вырезали на фрезерном станке, изначально вырезали из картона, но позже для прочности перешли на фанеру, однако из-за не до конца спланированных отверстий, еще раз перечертили и перевырезали основную платформу. Используя вырезанные детали, детали из набора MakeBlock и спаянные нами вышеперечисленные платы собрали самого робота
Модернизации конструкции:

Первая версия шасси робота была собрана полностью из конструктора MakeBlock и имел всего 1 этаж. Во 2-й версии робота (можно увидеть на 1-х чертежах конструкции) робот состоял из 2-х этажей, где на 2-ом этаже располагались ультразвуковые датчики расстояния и meAURIGA, при размещении камеры команда решила сделать 3-й этаж, на котором и будет располагаться камера, еще позже на этом этаже появились экранчики для вывода информации, а на 2-ом этаже ультразвуковые датчики заменились лазерными. В 4-й версии этого робота полностью поменяли шасси для робота: они стали полностью плоскими (т.е. не имеют наклона гусениц, который был в предыдущий версиях) и стал иметь форму треугольника, для возможности регулировки натяжения гусениц. 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 увеличило его проходимость.

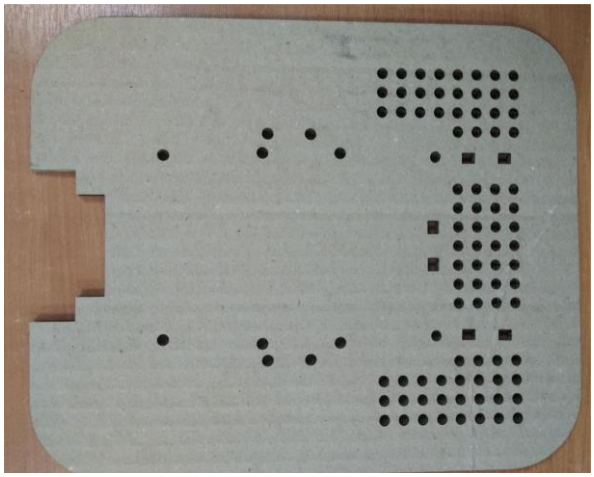
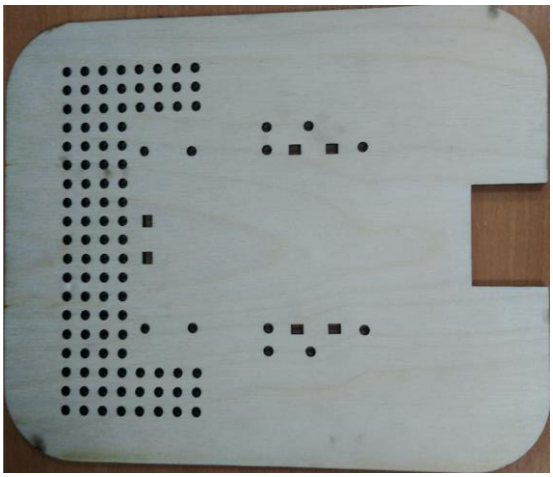
Чертежи и макеты корпуса робота



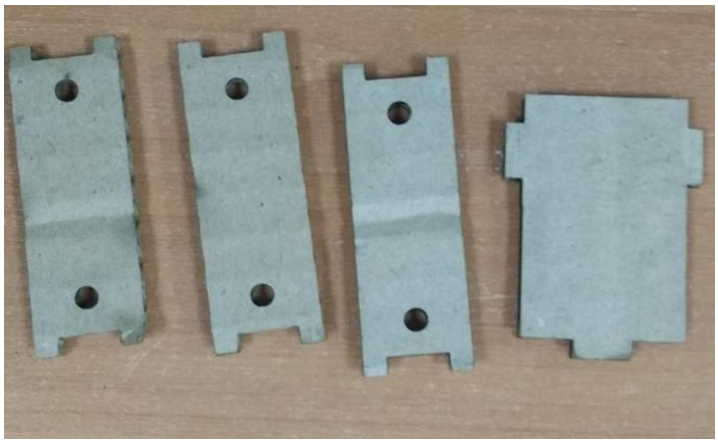
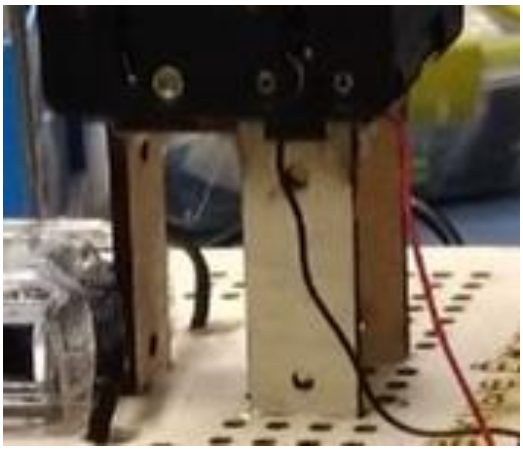
Эскиз/чертеж 2-й версии робота на бумаги и собранная версия



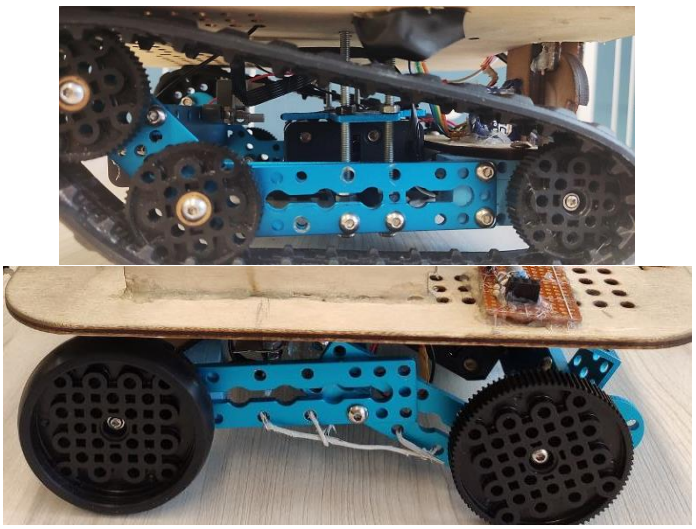
Чертеж и собранная конструкция корпуса для проводов и meAURIGA

1-я и 2-я версия основной платформы робота

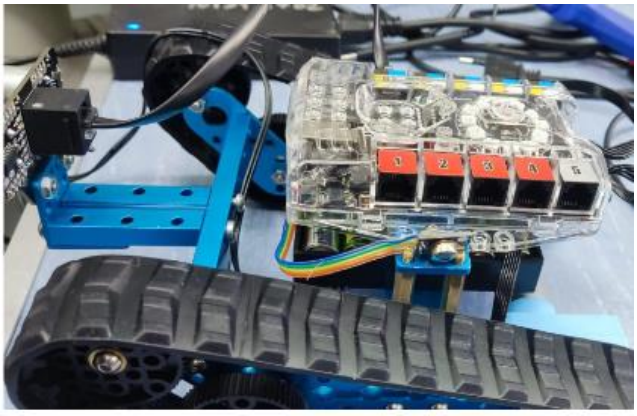
Картонный макет и итоговый крепеж для кнопки

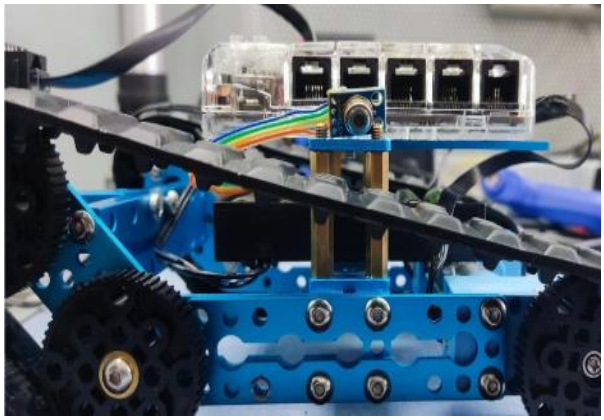


Старая и новая форма шасси

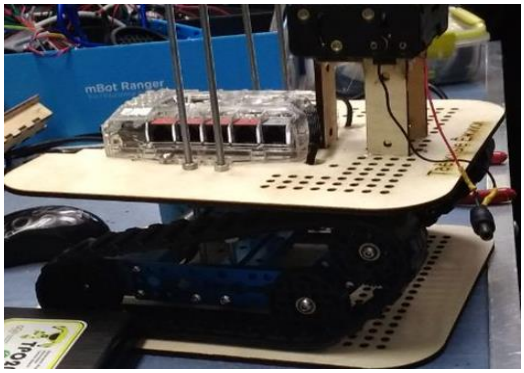
(Больше чертежей и схем можно найти на нашем GitHub)

Фото Робота

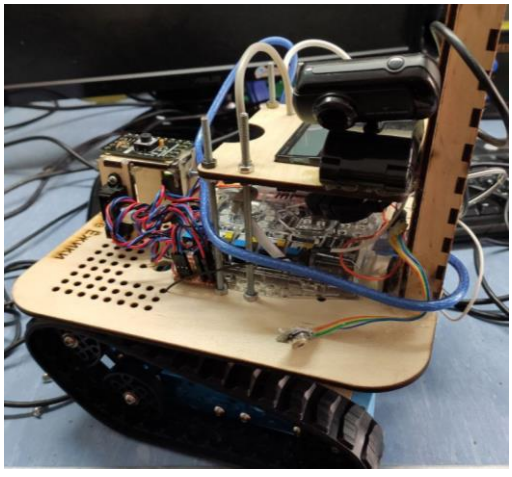


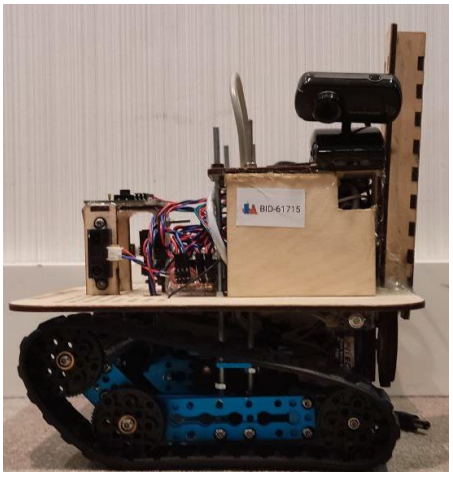


1-я версия робота



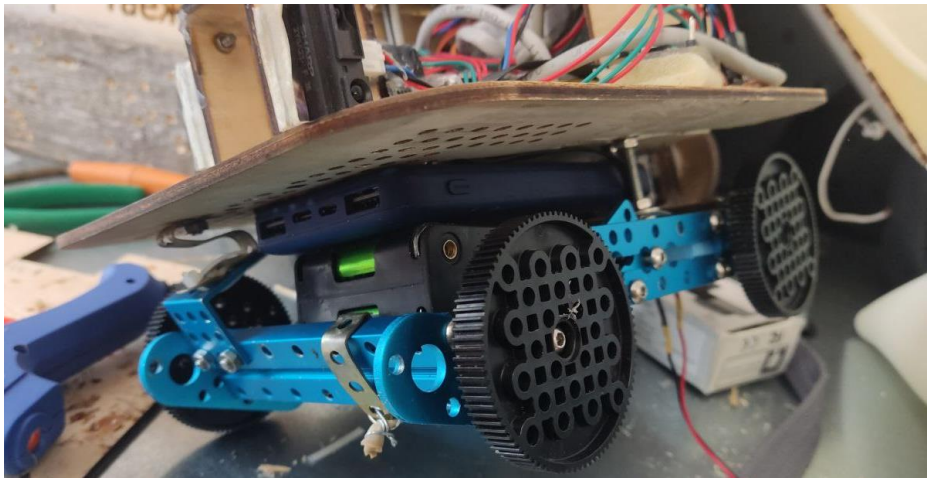
2-я версия робота



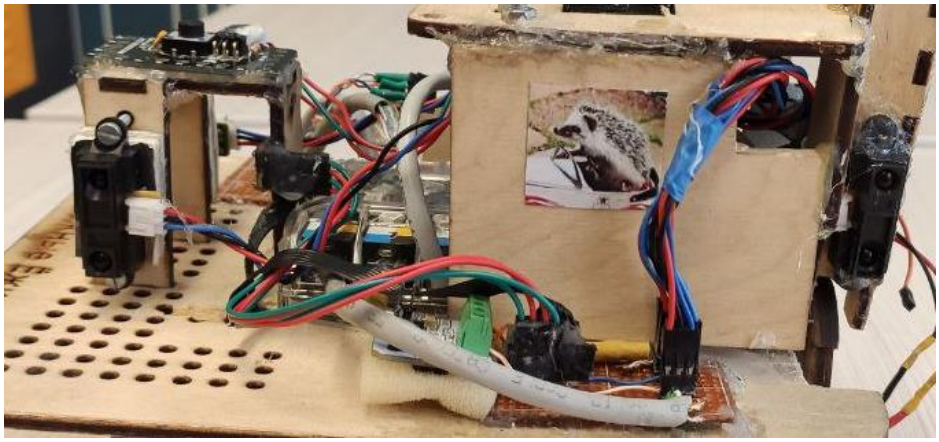


3-я версия робота

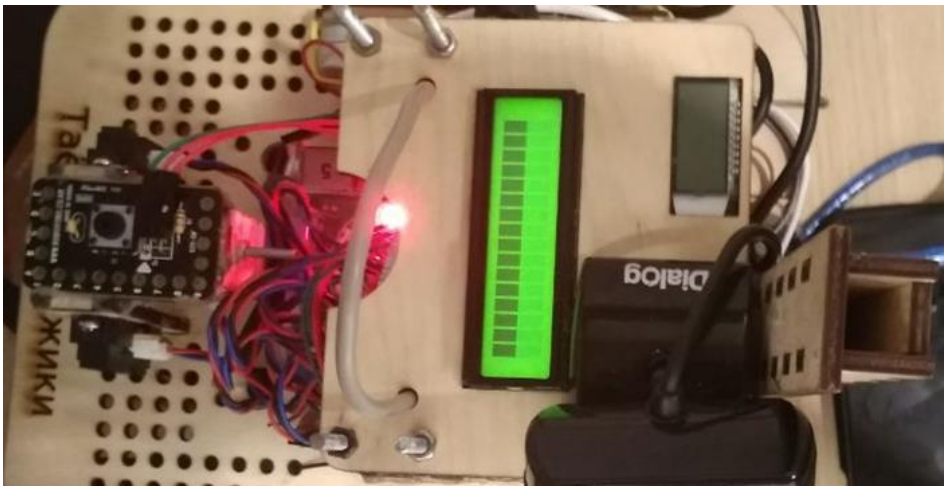
4-я версия робота:



0-й и 1-й этажи робота



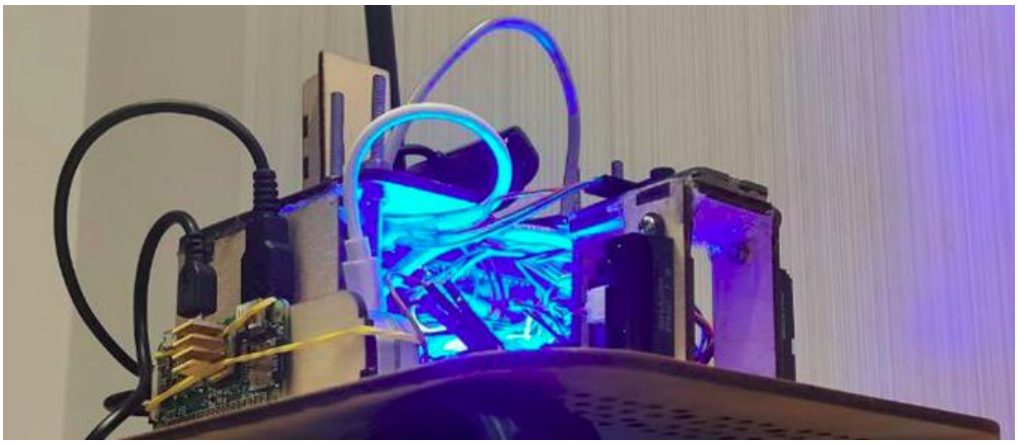
2-й этаж робота

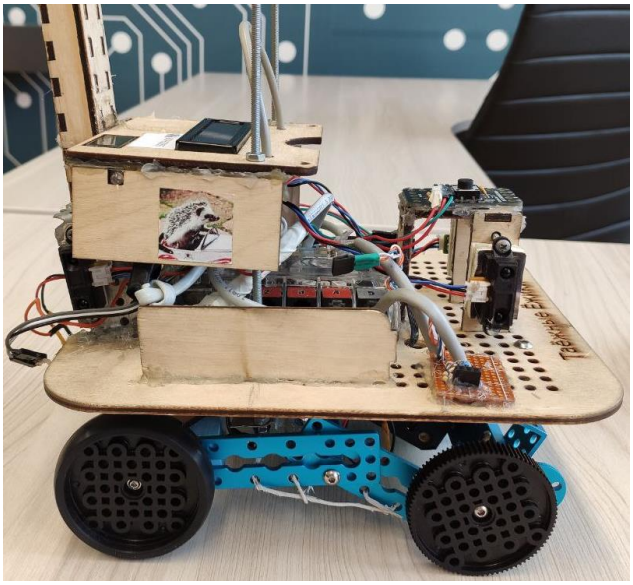


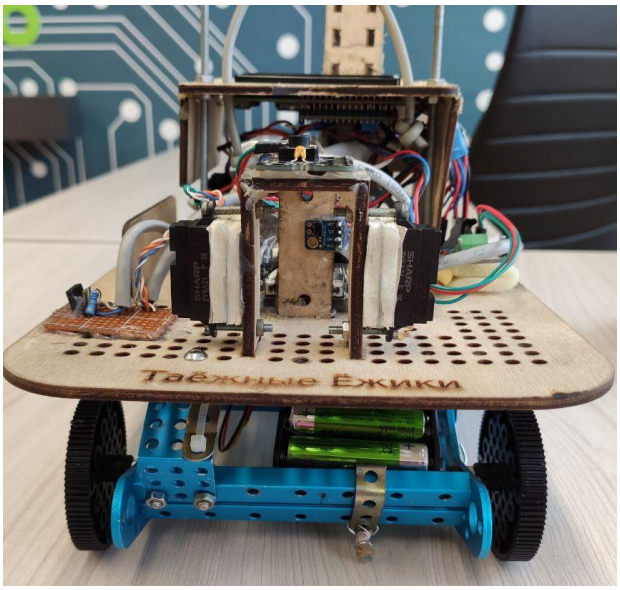
3-й этаж робота

Робот целиком

\







Программное обеспечение

В качестве программного обеспечения мы использовали написанную самостоятельно программу на C/C++ с использованием библиотеки OpenCV 4 на операционной системе Ubuntu 21.04, для программирования MakeBlock используем программу arduino IDE.

Работа с веб-камерой и OpenCV

Мы взяли веб-камеру с частотой 30 кадров в секунду. Благодаря этой веб-камере, мы захватываем с нее кадры и бинаризируем изображение, ища цвета цветных жертв и черного для букв.
Фрагмент программы, работающей с камерой на OpenCV 4

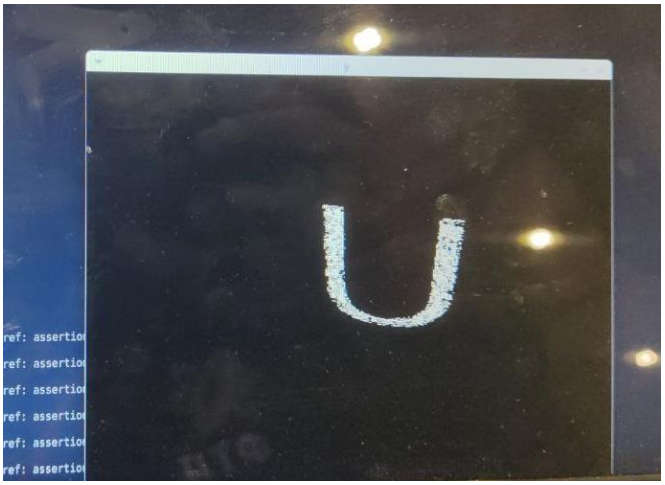
```
    //cv::Mat image = mur.getCameraOneFrame();
    cap >> image;
    cv::imshow("Image", image);
    cv::cvtColor(image, image, cv::COLOR_BGR2HSV);
    cv::Scalar lower(hMin, sMin, vMin);
    cv::Scalar upper(hMax, sMax, vMax);
    cv::inRange(image, lower, upper, image);
    cv::imshow("Bin", image);
    char c = cv::waitKey(10);
    if (c == 27) { // Íńěč íŕćŕňí ESC - âűőîäčě
       break;
    }
}
```

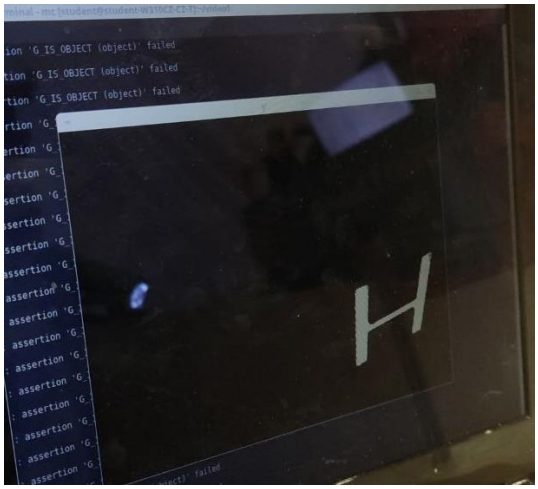
Алгоритм бинаризации черного цвета

Освещение на поле не предсказуемо, в прошлом году нам требовалась проводить бинаризацию для черного цвета перед каждым заездом, что задействовало очень много времени. Поэтому мы решили разработать алгоритм автобинаризации. Алгоритм, который мы придумали ищет на изображении самый светлый, самый темный и среднестатистическое значение пикселя и формулой рассчитывает данные для бинаризации на черный (чем темнее освящение на поле, тем больше делителей). Таким образом, робот может находить линию даже в сумраке.

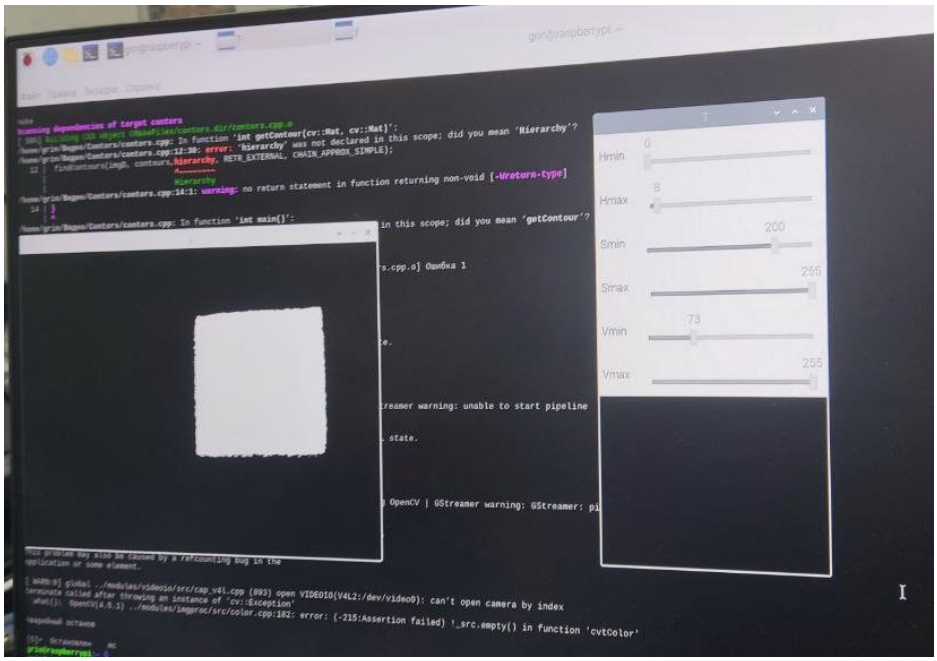
Примерная формула, которую мы используем на псевдокоде:

текущий_пиксель<(min+ср_знач)/2+ min





Автобинаризация на черный

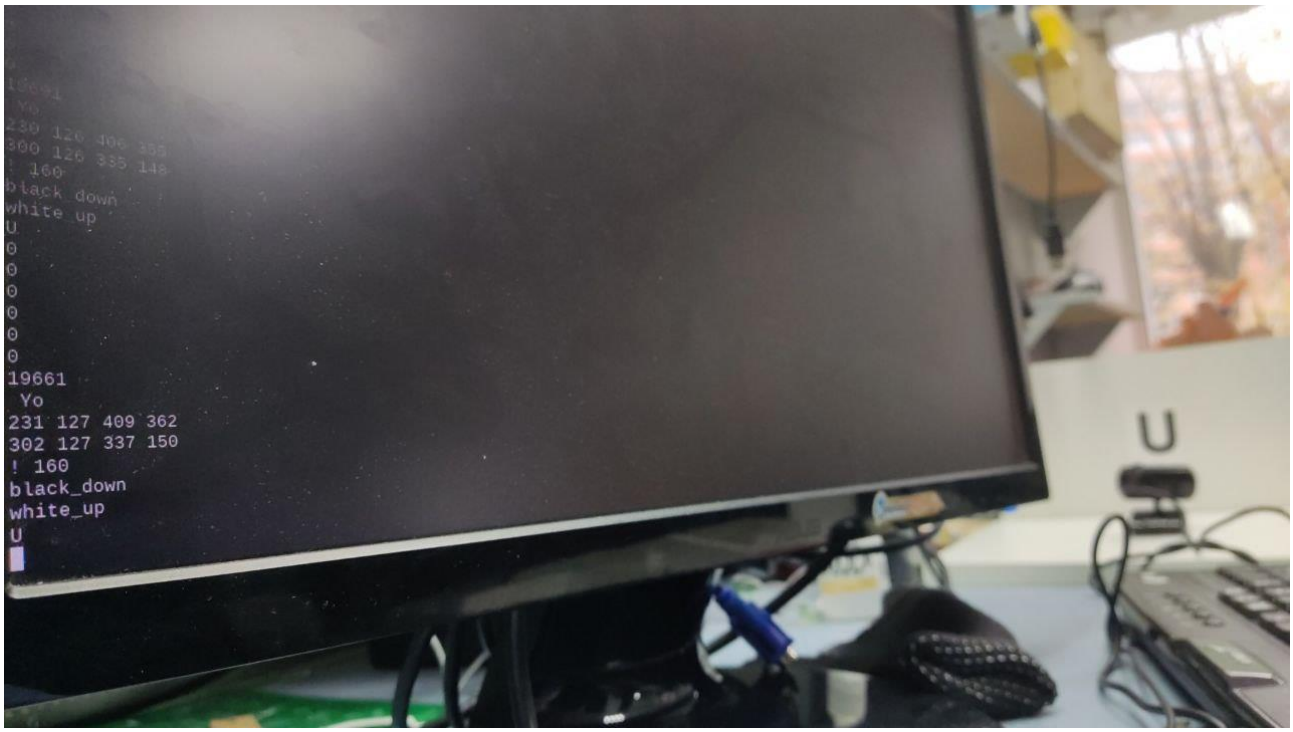


Ручная бинаризация на цветные метки

Определение жертв

Жертвы мы обнаруживаем с помощью камеры и библиотеки OpenCV 4. Для начала настраивается бинаризация для 1 из 3 цветов: красного, жёлтого и чёрного. После бинаризации определяем контуры и проверяем самые большие контуры, подходят ли они. 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 - чёрные, то это s, если нижний и верхний - не чёрные - то это H, а если нижний – чёрный - то это U.

Точки, анализируемые нами



Определение букв

Обсуждение и заключение

Решение проблем:
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 тренировочных 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 Raspbberry на meAURIGA через Serial и по какой-то причине Raspbberry пыталась питаться через meAURIGA, а не через пауэрбанк, из-за чего

сгорала Raspbberry. В результате было принято решение перейти с serial на UART. Возникшая проблема была решена.
4) Зачастую у Raspbberry 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 библиотеки и смогли определить необходимую команду.

Чему научились члены команды:
Цыганкова Мария и Григорий Пильщиков познакомились и научились применять функции из OpenCV, работать с Raspberry Pi Zero, захватывать изображение с веб-камеры, применили бинаризацию для обнаружения зеленого семафора и др.

Планы на будущее:
Команда будет продолжать обучаться техническому зрению, в дальнейшем планируем усовершенствовать и уменьшить конструкцию робота, поставить вторую камеру и научиться считывать информацию одной Raspberry с нескольких камер. Разработать упрощенный, но более эффективный алгоритм технического зрения.

Благодарности:
Благодарим нашего тренера Косаченко Сергея Викторовича за то, что он проводил с нами время и направлял нас на правильные пути и решения. Благодарим НПП Томскую Электронную компанию коммерческую помощь. А также мы благодарим Томский Физико-Технический Лицей за предоставленные нам ноутбуки и другое оборудование.

Наш тренер

ТФТЛ ТЭК

Наши спонсоры

Ссылки на нас:

GitHub

youtube