# Техническое описание робота

Состязание: состязания роботов с техническим зрением памяти Виктора Ширшина
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 модифицировали робота категории RCJ Rescue Maze для участия в состязании роботов с техническим зрением памяти Виктора Ширшина, разработали 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 2022 — 2 место в состязании роботов с техническим зрением памяти Виктора Ширшина

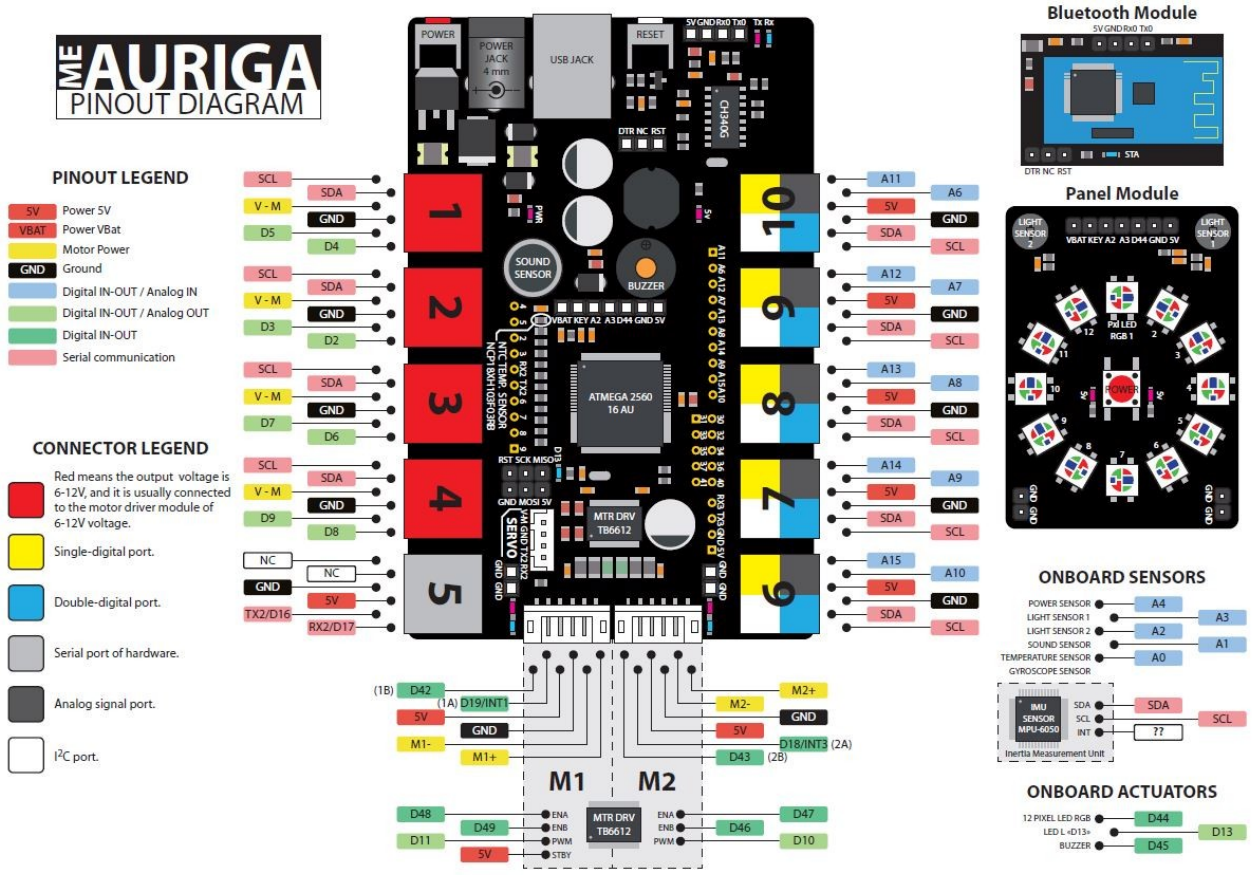
XII Региональная олимпиада по образовательной робототехнике школьников Томской области 2023 — 1 место в состязании Robocup Junior Rescue Simulation Webots Erebus
Откратый Российский чемпионат по робототехнике 2023 — 2 место в состязании Robocup Junior Rescue Simulation Webots Erebus
2022 год — Участие Цыганковой Марии и Пильщикова Григория в фестивале подводной робототехнике «АкваРобоФест» в г.Асино

## Описание Робота

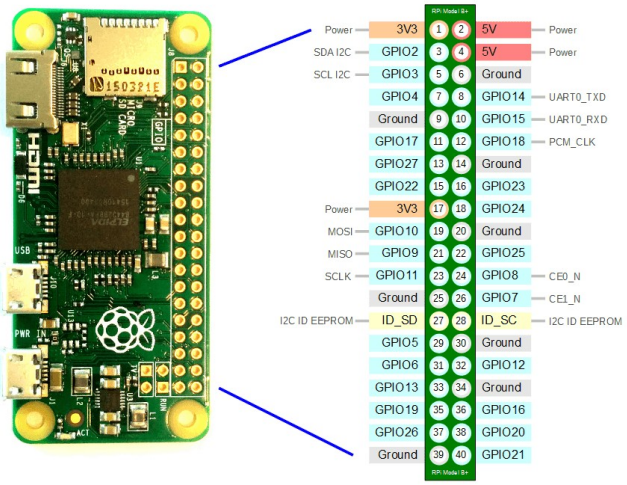
Стратегия выполнения задания роботом:
Робот должен дождаться разрешающего сигнала семафора и после этого следовать по черной линии до финиша. Определять цвет и форму семафора мы решили с помощью технического зрения: если цвет семафора красный, то стоять на месте, и если цвет семафора зелёный начать движение по линии. Для того, чтобы робот мог стартовать ровно на линии старта камера присоединена к сервоприводу, при запуске камера повернута к семафору, после того как появился зеленый свет она поворачивается и начинает движение по линии. Из-за того что освещение на поле не предсказуемо, а проводить повторную бинаризацию линии долго был разработан особый алгоритм автобинаризации, за счет которой робот может видеть линию при любом освещении не требует дополнительных настроек (подробнее в разделе «Определение черно линии»).

Использование датчиков:
В нашем роботе используется 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 I2Shub 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

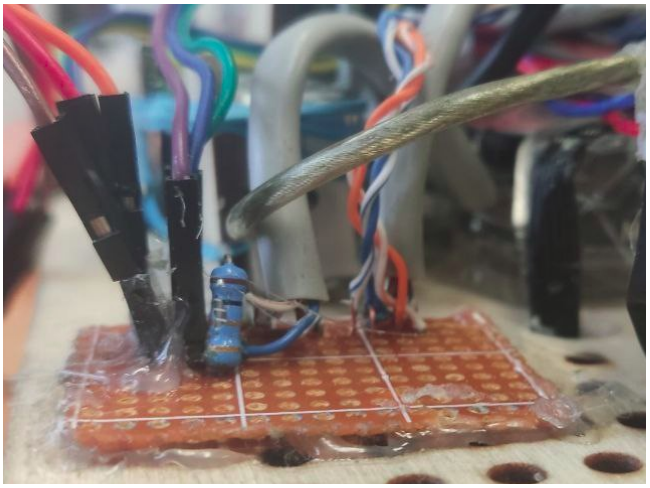



Распиновка Raspberry Pi Zero,

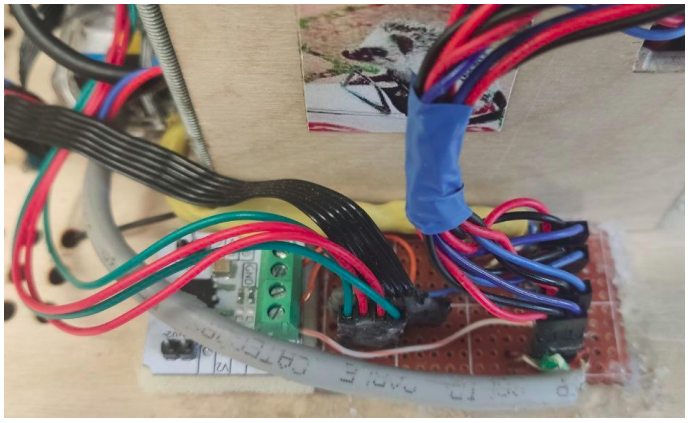
| Pin | Name | Function | Color | Pin Numbering |
|---|---|---|---|---|
| 1 | ANA | Analog interface, +9V Supply | white |  |
| 2 | GND | Ground | black | |
| 3 | GND | Ground | red | |
| 4 | IPOWERA | +4.3V Supply | green | |
| 5 | DIGIAI0 | I²C Clock (SCL), RS-485 A | yellow | |
| 6 | DIGIAI1 | I²C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

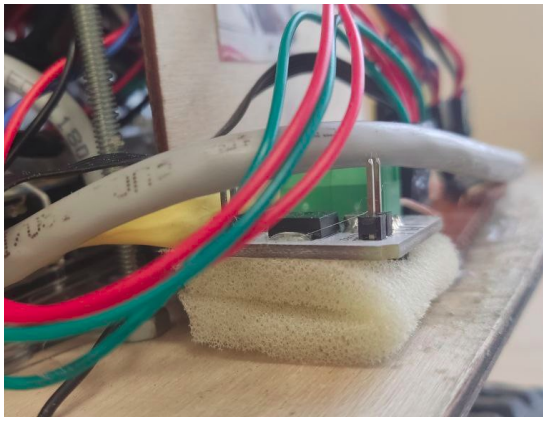


Распиновка порта MakeBlock RJ12



Самодельный I2Shub



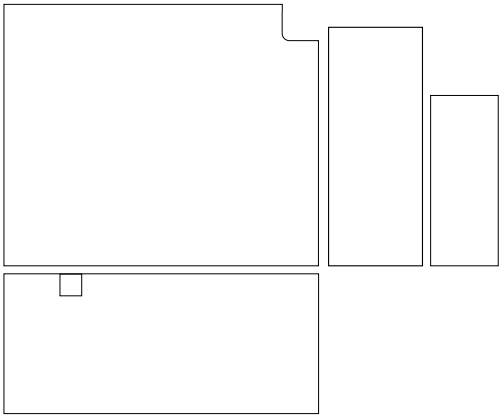


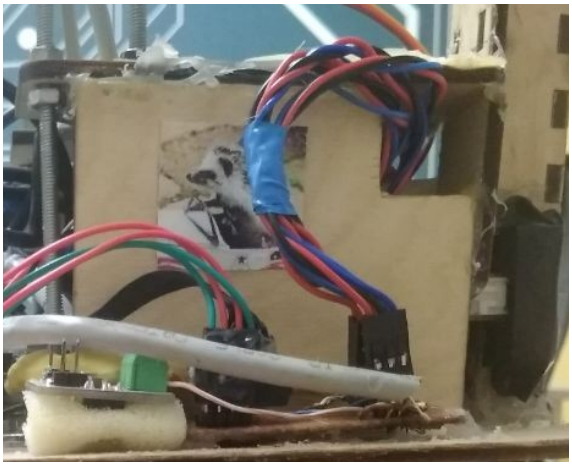
переходная плата с портов MakeBlock RJ12 на разъемы с шагом 2.45 мм, стабилизатор питания с 5 В до 3.3 В

(больше распиновок и схем можно найти на нашем GitHub)

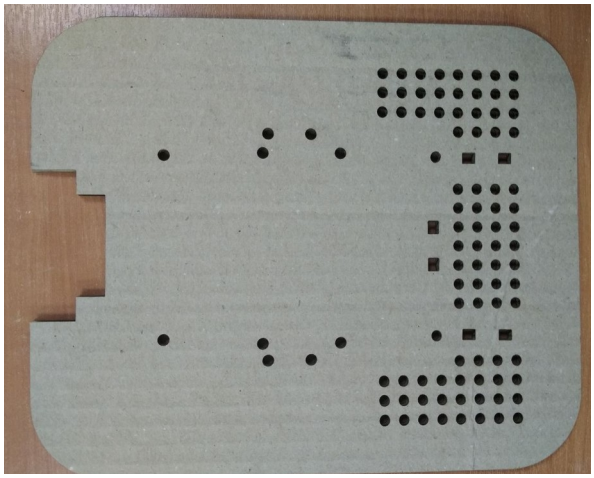
Конструкция робота:
Конструкция робота предполагает несколько этажей с разным функционалом: на нулевом располагаются моторы, на первом — аккамулятор для meAURIGA и пауэрбанк для Raspbberry, на втором — сама meAURIGA и  Raspbberry, USBhub, самодельный I2Shub, переходная плата с портов MakeBlock RJ12 на разъемы с шагом 2.45 мм, стабилизатор питания с 5 В до 3.3 В, на третьем — два экрана для вывода информации и кнопка для быстрого запуска и на четвертом сама камера и моторчик для её мобильности. 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

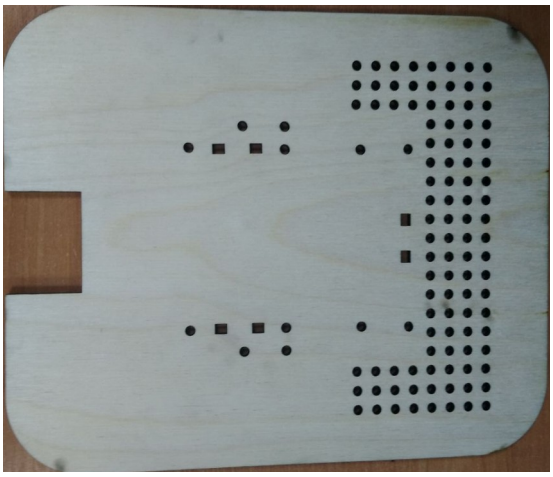
## Чертежи и макеты корпуса робота



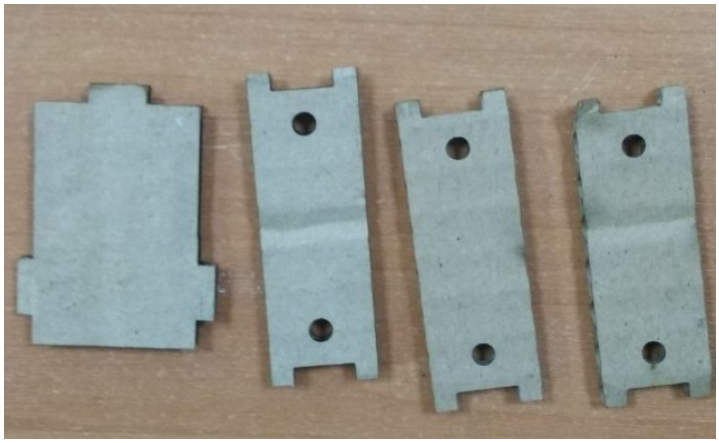


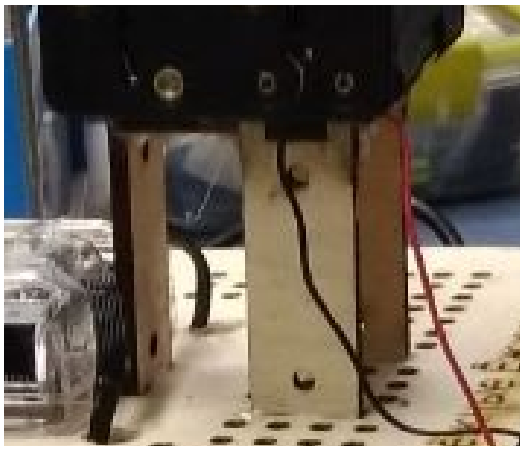
Чертеж и собранная конструкция корпуса для проводов и meAURIGA





1-я и 2-я версия основной платформы робота

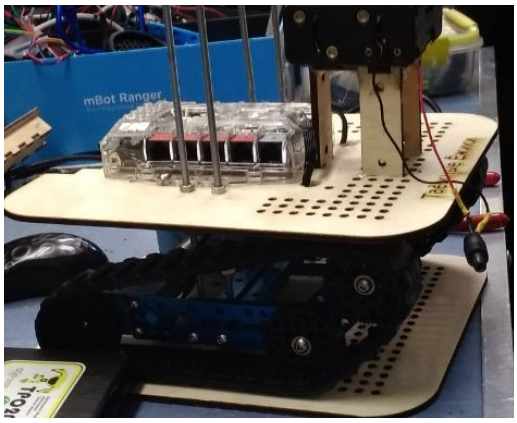


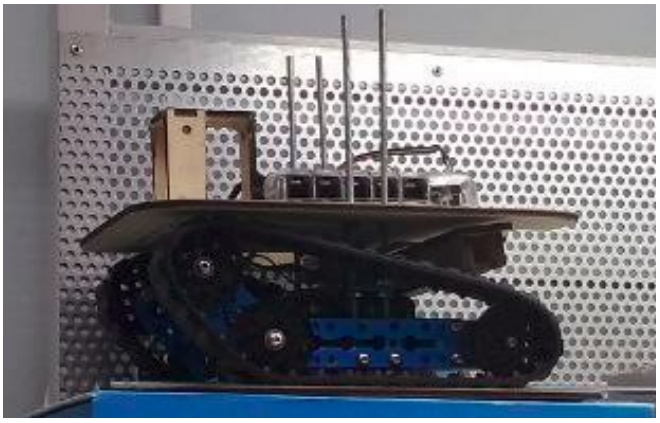


Картонный макет и итоговый крепеж для кнопки

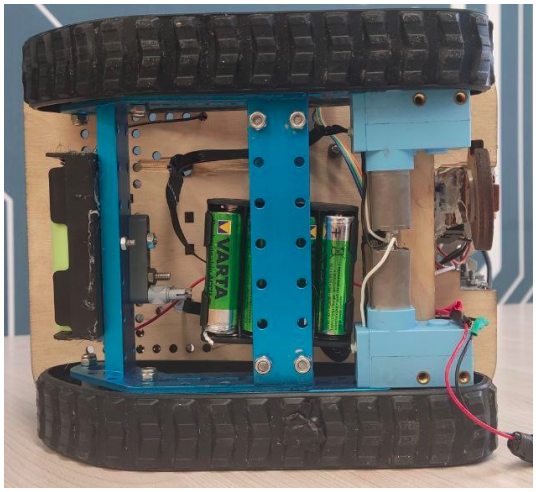
(Больше чертежей и схем можно найти на нашем GitHub)

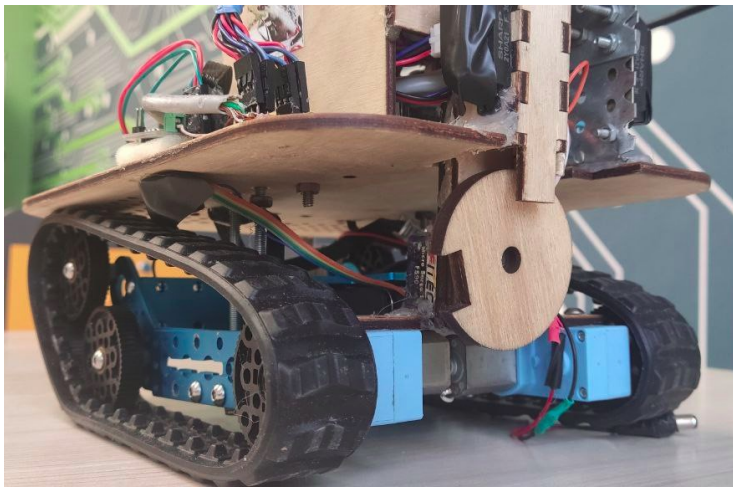
Фото Робота



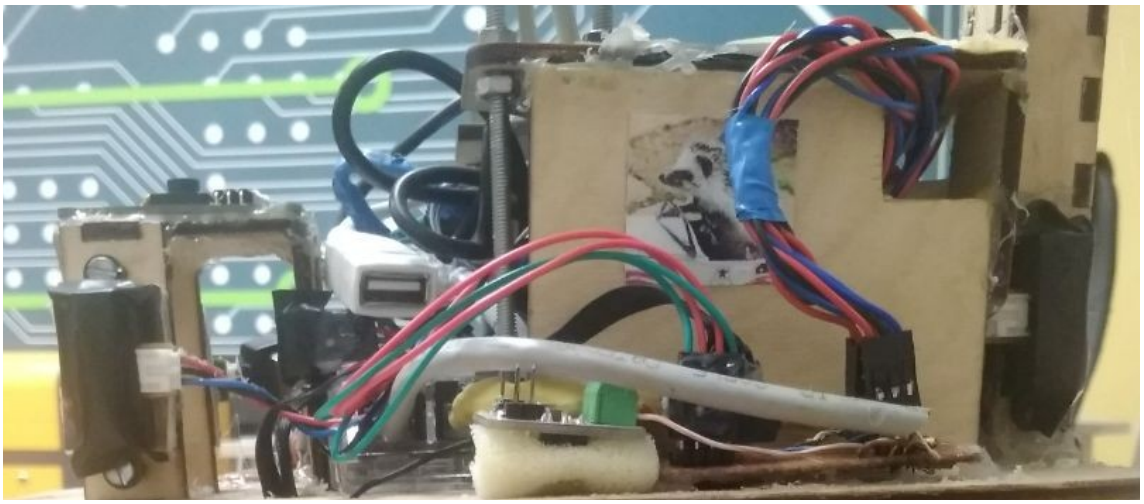



Первая версия шасси для робота

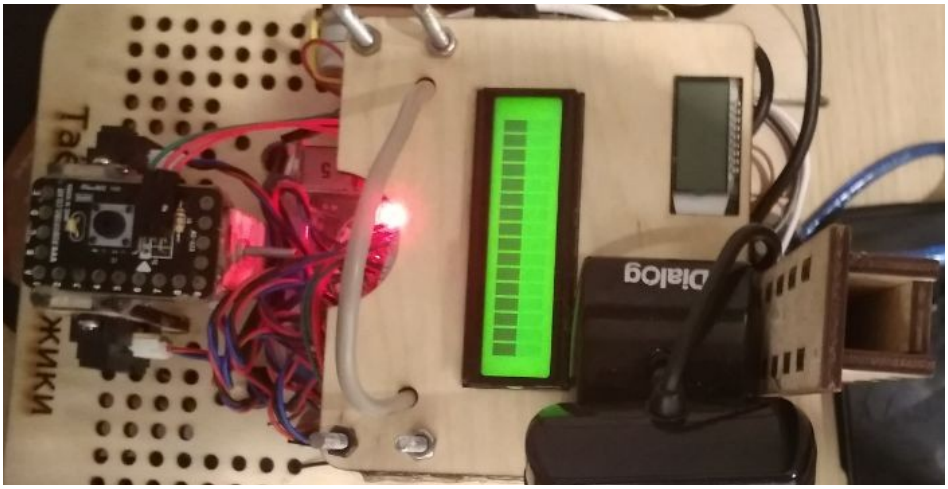


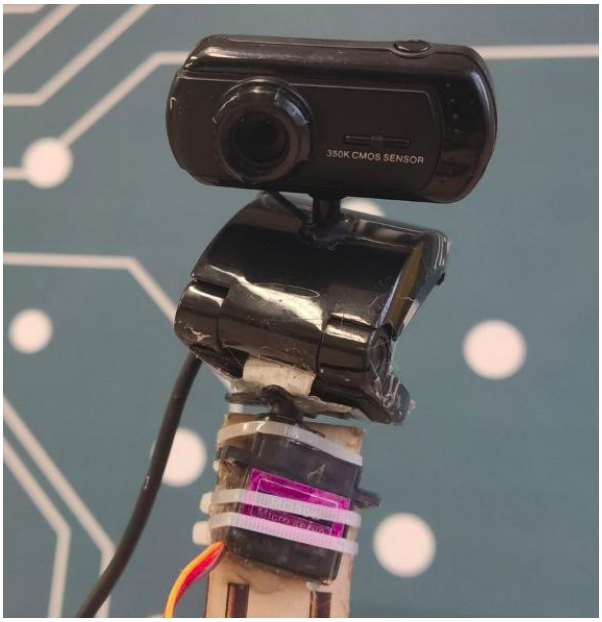
0-й и 1-й этажи робота



2-й этаж робота



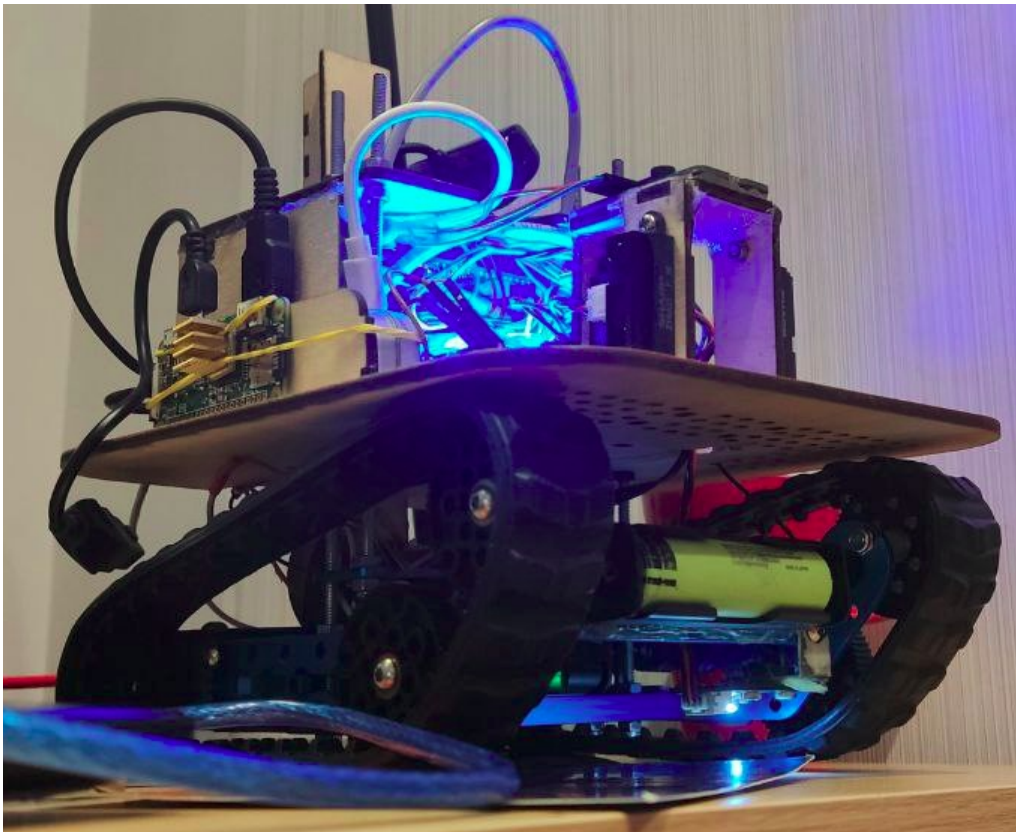

3-й этаж робота

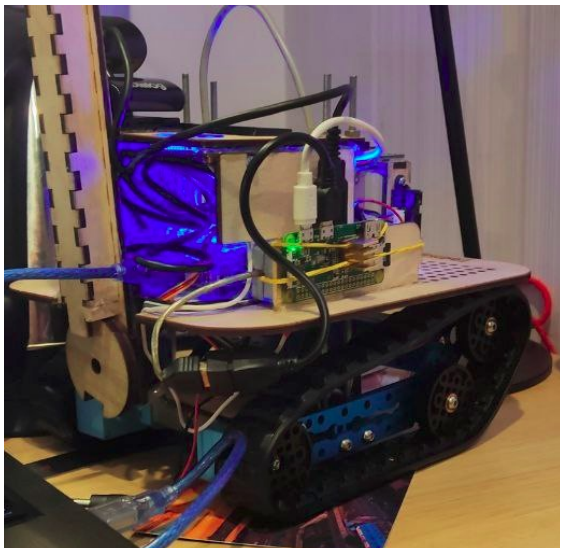


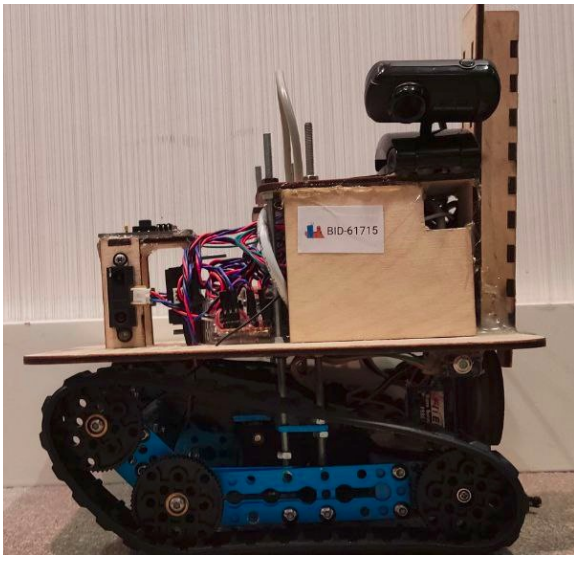

4-й этаж робота

Робот целиком

\

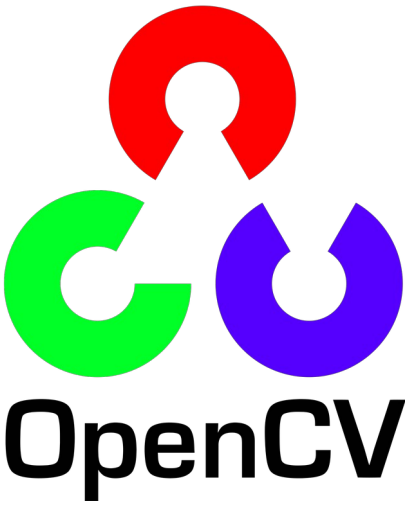






## Программное обеспечение

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 IDLE.





## 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, выделяя зеленый цвет, и среди выделенных фигур мы ищем элипс (зелёный).
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 íŕćńňŕ ESC - âűőîäčě
            break;
        }
    }
```

## Бинаризация для определения зелёного семафора в кадре

Сначала мы создаём кадр и в нём ищем контур круга (эллипса). Если такой эллипс найден, то этот контур обводится зелёным цветом, и робот начинает своё движение по линии.
Фрагмент кода на OpenCV 4, который ищет контуры:

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(imgDil,contours,hierarchy,RETR_EXTERNAL,
CHAIN_APPROX_SIMPLE);
or(int i = 0; i <contours.size(); i++){
 int area = contourArea(contours[i]);
 vector<vector<Point>>conPoly(contours.size());
 vector<Rect>boundRect(contours.size());
 string objectType;
  tur = area;
 // cout << "! " << tur << endl;
  if(tur >= 50){
   dich = 1;
  }
  if(area > 20000 && area < 70000){
   number = 1;
   float peri = arcLength(contours[i],true);
   approxPolyDP(contours[i],conPoly[i],0.003 * peri,true);
   boundRect[i] = boundingRect(conPoly[i]);
   cout<<"AREA: "<<area<<endl;
   drawContours(img, conPoly,i,Scalar(0,255,0),2);
 }
}
```

## Определение черно линии

Освещение на поле не предсказуемо, в прошлом году нам требовалась проводить бинаризацию для черной линии перед каждом заездом, что задействовало очень много времени. Поэтому мы решили разработать алгоритм автобинаризации. Алгоритм, который мы придумали ищет на изображении самый светлый, самый темный и средне-статистическое значение пикселя и формулой расчитывает 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
```

## Программа управления для контроллера шасси

Робот по камере находит отклонение центра робота от линии и стремится уменьшить отклонение от линии благодаря П-Регулятору. Следовательно, робот должен выравниваться и ехать по линии. Программу управления для контроллера мы писали на Arduino IDE.
Фрагмент кода реализации П регулятора:

```
// ======= П регулятор
// Рассчитываем отклонение работа от желаемого центра
float error = ((x + xLeft) / 2) - (640/2);
// Рассчитываем пропорциональную часть регулятора
//angle = 90 + (error * 1.2);
// Ограничиваем угол поворота сервопривода
//if(angle < 60) angle = 60;
//if(angle > 120) angle = 120;

        int b = 130 - (error * 0.3); int a = 130 + (error * 0.3);

        if(a > 254) { a = 254; }
        if(a < 1) { a = 1; }

        if(b > 254) { b = 254; }
        if(b < 1) { b = 1; }
```

## 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 придумать все необходимые отверстия и крепежи, из-за постоянной модификации конструкции, необходимо заново переделывать их. (одной 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 долгой, из-за чего было принято решение придумать 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. Поэтому было принято решение перейти с serial на UART.
4) Зачастую у Raspbberry сбивается время и программа не компилируется, выводя ошибку «Время изменения файла находится в будущем», всякий такой раз нам приходилось копировать каталог под новым временем и сносить старый с неправильным.
5) Библиотеки MeAuriga.h. Мы не могли найти подходящие команды для программирования робота. Решением проблемы стало перебирание примеров из этой библиотеки, благодаря этому нам удалось найти нужные команды.

6) В процессе пайки нужно быть очень внимательным, иначе можно сжечь много важных схем. Команда уже сожгла одну Raspberry и многое другое. Поэтому команда в процессе изготовления робота обучалась искусству пайки.

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

Наши спонсоры

Ссылки:

GitHub

youtube