

RoboCup Russia Open 2022
Technical Description Paper (TDP)

RCJ Soccer Open

ALPHA

Участники:

Романов Никита, @thenromanov
Макушинский Егор, @egormlh

Тренер:

Мустафин Сергей, @mus_serg

2022

Содержание

1 Введение	3
2 Конструкция роботов	4
2.1 Предыдущие версии	4
2.2 Компонентная база	5
2.2.1 Микроконтроллер	5
2.2.2 Моторы	5
2.2.3 Драйверы моторов	5
2.2.4 Камера	6
2.2.5 Гироскоп	6
2.2.6 Соленоид	6
2.2.7 Источник питания	7
2.3 Материнская плата	7
2.4 Датчик линии	8
2.5 Сферическое зеркало	8
2.6 Прочие детали	9
3 Стратегия	10
4 Программная часть	11
4.1 Управление движением	11
4.2 Обработка изображения	11
5 Финальная версия робота	13

1 Введение

Команда **Alpha** участвует в соревновании RoboCup уже четвертый сезон. Члены команды разработали конструкции роботов, их программное обеспечение и алгоритмы работы, используя прошлый опыт участия, а также материалы, предоставленные другими участниками RoboCup из разных стран. В этом документе мы объясним особенности наших технических и программных решений, а также расскажем о дальнейших планах развития.

Команда трижды участвовала в национальном этапе RoboCup, каждый раз улучшая своих роботов, их программное обеспечение и результат участия. Так, в 2018 году она заняла второе место в категории RCJ Soccer Lightweight, благодаря чему получила шанс представлять Россию на тихоокеанском чемпионате в Иране в декабре 2018. Там ребята смогли приобрести новые знания, опыт, знакомства. В 2019 команда продолжила участие в Lightweight лиге и успешно выступила на всероссийском этапе RoboCup. Заняв первое место, команда прошла на международный финал RoboCup 2019 в Австралии. В 2021 году команда продолжила участие и, перейдя в Open лигу, заняла первое место на RoboCup Russia Open 2021.

В этом сезоне команда состоит из двух участников: Никиты Романова, который занимался прототипированием и написанием алгоритмов для роботов, и Егора Макушинского, который разработал и изготовил электрические схемы, используемые в роботе. Руководителем команды является методист ГАОУ ЦПМ Мустафин Сергей Владимирович.

2 Конструкция роботов

2.1 Предыдущие версии

Первые версии наших роботов (см. Рис. 1), с которыми мы выступали в Lightweight лиге, состояли из нескольких слоев, вырезанных из фанеры. Материнская плата была разведена и вытравлена саморучно, а большинство соединений было выполнено с помощью проводов. Несущие крепления были напечатаны на 3D-принтере.

К сезону 2019 мы обновили конструкцию (см. Рис. 2). Все слои робота представляли собой печатные платы, изготовленные на заводе. Для крепления использовались металлические стойки, а соединение компонентов выполнялось посредством вертикальных плат.

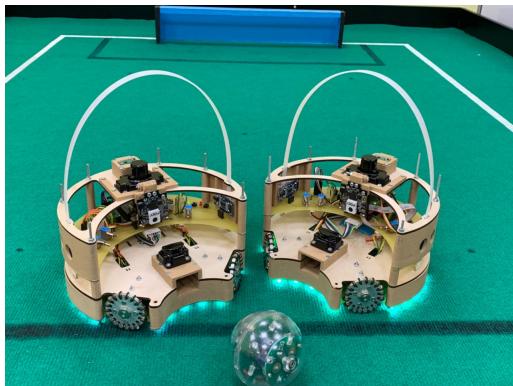


Рис. 1 Lightweight 2018



Рис. 2 Lightweight 2019

В 2021 году наша команда перешла в Open лигу. В новых роботах (см. Рис. 3) мы по-прежнему использовали платы в качестве слоев робота. Теперь вместе множества датчиков для определения мяча использовалось сферическое зеркало. Также с этого сезона в роботах появляется соленоид, который используется в качестве удара, и используются первые модификации дриблинга для контроля мяча.



Рис. 3 Open 2021

Разрабатывая конструкцию роботов на текущий сезон, мы сохранили ключевые принципы создания моделей и адаптировали их под изменившиеся правила лиги.

2.2 Компонентная база

2.2.1 Микроконтроллер

В качестве головного контроллера мы используем Teensy 3.6 (см. Рис. 4). Плата построена на высокопроизводительном 32-разрядном ARM-процессоре с ядром Cortex-M4 с частотой 180 МГц и имеет 256 КБ оперативной памяти, 1 МБ энергонезависимой Flash-памяти, а также широкий набор интерфейсов.



Рис. 4 Teensy 3.6

2.2.2 Моторы

В роботе используются 4 мотора Pololu (см. Рис. 5). Этот мотор-редуктор состоит из мощного щеточного двигателя постоянного тока напряжением 12 В в сочетании с металлической цилиндрической коробкой передач 20.4:1. На моторах установлены алюминиевые всенаправленные колеса GTF Robots 50mm Omni Wheel (см. Рис. 6) с отверстием диаметром 4 мм.



www.pololu.com

Рис. 5 Мотор Pololu



Рис. 6 Колесо GTF Robots

2.2.3 Драйверы моторов

В роботе используются 2 драйвера Pololu TB67H420FTG (см. Рис. 7). Этот драйвер упрощает использование щеточного двигателя постоянного тока, и работает в двухканальном режиме для независимого двунаправленного управления двумя двигателями. Он имеет широкий диапазон рабочих напряжений от 10 В до 47 В и может подавать непрерывное напряжение 1,7 А на каждый канал двигателя.



Рис. 7 Pololu TB67H420FTG

2.2.4 Камера

Чтобы определить мяч, мы используем сферическое зеркало и камеру Pixy2 (см. Рис. 8). Датчик изображения Charmed Labs Pixy 2 CMUcam5 меньше, быстрее и мощнее, чем у оригинальной Pixy. Pixy2 может запоминать до 7 сигнатур (цветов), находить сотни объектов в одном кадре и обрабатывать до 60 кадров в секунду, поэтому эту камеру очень удобно использовать в Open лиге.



Рис. 8 Pixy2

2.2.5 Гироскоп

Для ориентирования в пространстве мы используем GY-521(см. Рис. 9). Модуль GY-521 построен на базе микросхемы MPU6050. На плате модуля также расположена необходимая обвязка MPU6050, включая подтягивающие резисторы интерфейса I2C. Гироскоп используется для измерения линейных ускорений, а акселерометр – угловых скоростей. Совместное использование акселерометра и гироскопа позволяет определить движение тела в трехмерном пространстве.



Рис. 9 GY-521

2.2.6 Соленоид

В качестве удара в роботе используется соленоид Магнитек ОМ-0626В (Рис. 10). Для удара мы повышаем напряжение с 10.8 В до 100-150 В.



Рис. 10 Магнитек ОМ-0626В

2.2.7 Источник питания

В качестве источника питания используется LiPo аккумулятор TATTU 3S с рабочим напряжением 11,1 В и емкостью 1300 mAh.



Рис. 11 Аккумулятор

2.3 Материнская плата

Форма материнской платы была спрототипирована в Autodesk Fusion 360, а электрическая схема была разведена в KiCAD (см. Рис. 12). Геометрически, плата разведена так, что моторы расположены под углом 90° относительно друг друга, но при этом имеют смещение в центр. Такое расположение позволяет увеличить нишу для захвата мяча, а также высвобождает пространство для аккумулятора. На плате расположены 2 понижающих DC-DC преобразователя: на 3.3 В для питания датчика и 5 В для питания периферии, установлены разъемы для микроконтроллера, драйверов моторов, камеры и гироскопа. Так же на плате установлены компоненты управления и контроля: тумблеры питания, светодиоды и кнопки. Финальная версия платы была изготовлена на заводе (см. Рис. 13).

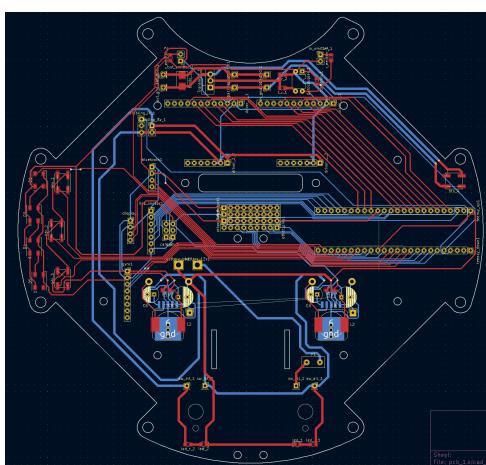


Рис. 12 Схема материнской платы

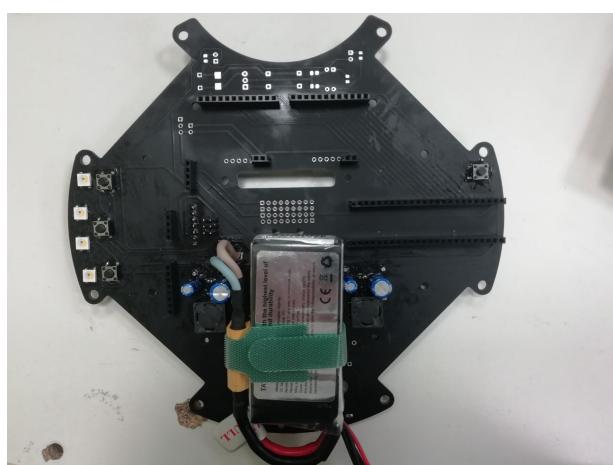


Рис. 13 Материнская плата

2.4 Датчик линии

Датчик линии имеет кольцевую форму и состоит из шестнадцати фоторезисторов, каждый из которых подсвечивается отдельным светодиодом. Рабочее напряжение датчика составляет 3.3 В. Для считывания используются два восьмиканальных мультиплексора. Электрическая схема была разведена в KiCAD, а финальная версия была заказана на заводе (см. Рис. 14).

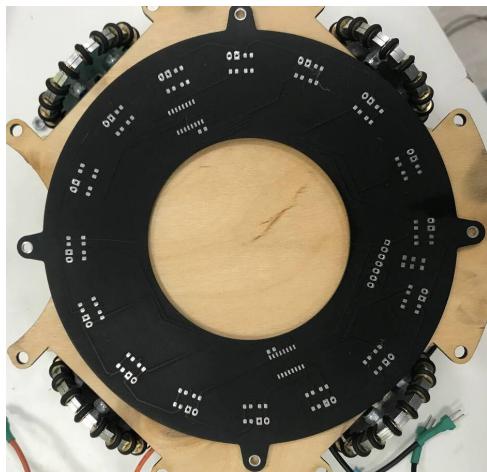


Рис. 14 Датчик линии

2.5 Сферическое зеркало

В текущей конструкции мы использовали то же сферическое зеркало, что и в предыдущей версии робота. График, в соответствии с которым было изготовлено зеркало представлен на рисунке 15. Модель зеркала напечатана на полимерном принтере и обработана NiCr покрытием с помощью процесса гальванизации (см. Рис. 17).

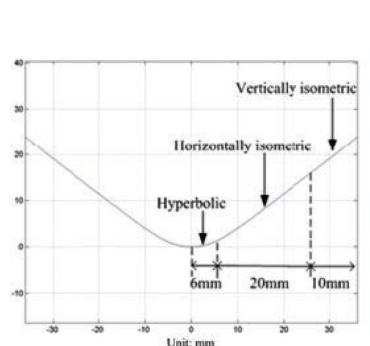


Рис. 15 Форма зеркала



Рис. 16 Покрытие заготовки



Рис. 17 Сферическое зеркало

2.6 Прочие детали

Различные стенки, стойки и крепления были спроектированы в САПР AutoDesк Fusion 360 и напечатаны на 3D-принтере PLA пластиком.

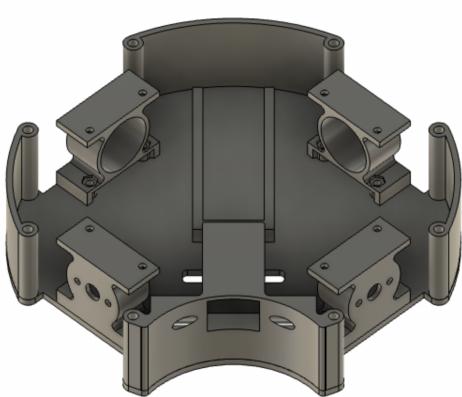


Рис. 18 Детали робота

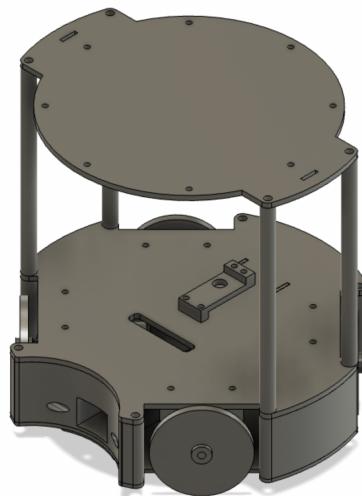


Рис. 19 3D-прототип робота

3 Стратегия

Из-за высокого темпа игры, и широких ворот мы решили отказаться от тактик двух нападающих и смены ролей вратаря и нападающего между роботами. Роботы строго делятся по ролям: нападающий и вратарь.

Вратарь почти не выезжает дальше штрафной зоны и защищает ворота. Основным ориентиром вратаря являются ворота, которые он обороняет. Робот центрирует свое положение относительно их, поскольку в ходе игры они никогда не будут перекрыты. При этом он сохраняет расстояние относительно ворот, чтобы не заехать в штрафную зону, и выезжает только при непосредственной близости мяча.

Нападающий же наоборот обычно не въезжает в зону своего вратаря, чтобы роботы не сталкивались, не мешали друг другу и не было двойной обороны. В ходе игры нападающий выравнивается относительно ворот соперника, и центрируется в их сторону, чтобы в момент удара не было необходимости в развороте. При потере ворот нападающий использует гироскоп регулирования направления.

4 Программная часть

4.1 Управление движением

Для управления движением робота необходима базовая функция, позволяющая по заданным скорости моторов и направлению включать каждый мотор с такой скоростью, что робот перемещается в нужном направлении. Такое регулирование осуществляется посредством подачи необходимых коэффициентов на каждый мотор. Подходящими коэффициентами являются тригонометрические функции вида $\pm \cos(\alpha \pm \frac{\pi}{4})$. Таким образом функция управления движением имеет вид:

```
1 void move(double speed, double dir, double k) {
2     M1.start(speed * cos(dir - 0.785398163397448) + k);
3     M2.start(speed * -cos(dir + 0.785398163397448) + k);
4     M3.start(speed * -cos(dir - 0.785398163397448) + k);
5     M4.start(speed * cos(dir + 0.785398163397448) + k);
6 }
```

4.2 Обработка изображения

Настройка камеры и обучение сигнатур производится в утилите PixyMon. Обработка изображения происходит на процессоре камеры и по протоколу SPI на контроллер поступает массив из найденных блоков (объектов). Каждый блок содержит 7 параметров:

- **sig** – порядковый номер сигнатуры
- **x** – центральный пиксель блока по оси X
- **y** – центральный пиксель блока по оси Y
- **width** - размер блока по оси X
- **height** - размер блока по оси Y
- **index** - порядковый номер блока, id
- **age** - возраст блока (кол-во последовательных кадров, на которых был найден этот блок)

Для работы с мячом мы используем ряд примитивов вычислительной геометрии.

В первую очередь необходимо уметь вычислять расстояние между парой точек на плоскости. Для этого используется формула Евклидова расстояния:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Другим важным примитивом является вычисление угла между векторами. Он используется для того, чтобы находить угол между направлением "прямо" для робота и мяча. Для его вычисления рассмотрим ряд определений.

Скалярное произведение (англ. dot product) двух векторов равно произведению их длин и косинуса угла между ними. Обозначается $\vec{v}_1 \cdot \vec{v}_2$, задаётся формулой

$$\vec{v}_1 \cdot \vec{v}_2 = |\vec{v}_1| \cdot |\vec{v}_2| \cdot \cos \alpha$$

где α - угол, на который надо повернуть \vec{v}_1 , чтобы он стал коллинеарен \vec{v}_2 .

При работе с координатами, для него справедлива следующая формула:

$$\vec{v}_1 \cdot \vec{v}_2 = v_{1x}v_{2x} + v_{1y}v_{2y}$$

Векторное произведение (англ. cross product, также называется *косым* или *псевдоскалярным*) для двух векторов равно произведению их длин и синуса угла между ними — причём знак этого синуса зависит от порядка операндов. Обозначается $\vec{v}_1 \times \vec{v}_2$, задаётся формулой

$$\vec{v}_1 \times \vec{v}_2 = |\vec{v}_1| \cdot |\vec{v}_2| \cdot \sin \alpha$$

При работе с координатами, для него справедлива следующая формула:

$$\vec{v}_1 \times \vec{v}_2 = v_{1x}v_{2y} - v_{1y}v_{2x}$$

Тогда пусть α - ориентированный угол между векторами \vec{v}_1 и \vec{v}_2 . Сопоставляя формулы для векторного и скалярного произведений этих векторов, имеем

$$\operatorname{tg} \alpha = \frac{\vec{v}_1 \times \vec{v}_2}{\vec{v}_1 \cdot \vec{v}_2} = \frac{v_{1x}v_{2y} - v_{1y}v_{2x}}{v_{1x}v_{2x} + v_{1y}v_{2y}}$$

Зная тангенс угла между векторами, мы легко найдем угол между прямыми, на которых лежат векторы \vec{v}_1 и \vec{v}_2 :

$$\alpha = \left| \operatorname{arctg} \frac{\vec{v}_1 \times \vec{v}_2}{\vec{v}_1 \cdot \vec{v}_2} \right|$$

В языках семейства С есть встроенная функция вычисления арктангенса *atan2()*.

Тогда, реализовав структуру вектора и перегрузив ряд операторов, получим реализацию следующего вида:

```

1 struct r {
2     int x, y;
3     r () {};
4     r (int x, int y) : x(x), y(y) {};
5 };
6
7 int operator*(r a, r b) { return a.x * b.x + a.y * b.y; }
8 int operator^(r a, r b) { return a.x * b.y - a.y * b.x; }
9 double angle(r a, r b) { return atan2(a ^ b, a * b); }

```

5 Финальная версия робота