Начать хотелось бы с особенностей наших программ.

Использование OpenCV.

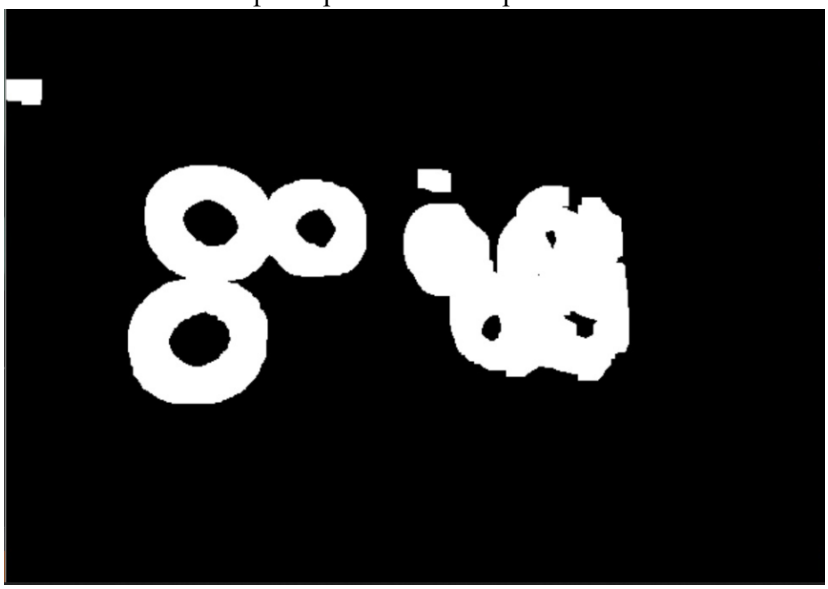
Для определения количества колец в автономном периоде мы решили, как и в прошлом году использовать OpenCV. OpenCV легко устанавливается в среду разработки Android Studio, в частности библиотеки EasyOpenCV. Распознавание будет происходить в отдельном параллельном потоке, чтобы не нагружать главный. Камера направлена так, чтобы видеть только кольца из своей половины поля, как с центральной позиции, так и с крайней.

Принцип распознавания:

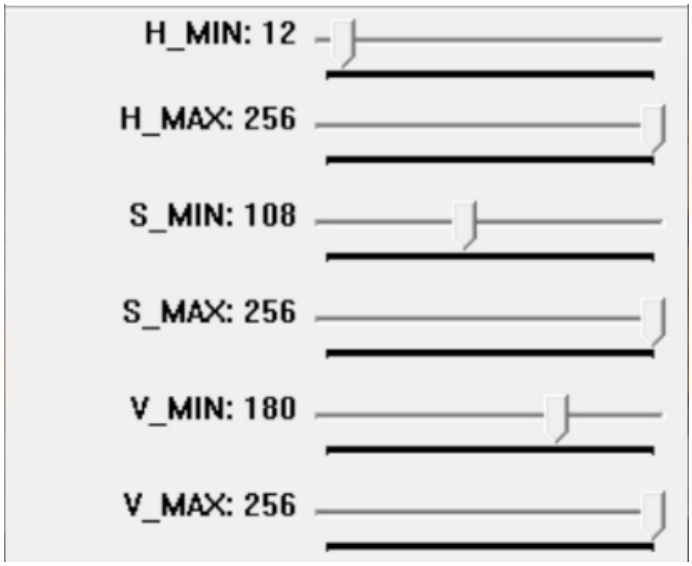
1. Получение матрицы пикселей (изображения) с веб-камеры с частотой 30 кадров в секунду
2. Перевод изображения из формата rgb в формат hsv. Формат hsv расшифровывается как Hue, Saturation, Value — тон, насыщенность, значение. И OpenCV использует для вычислений параметры hsv палитры.
3. Фильтруем изображение по цвету колец, для получения пятна из колец.

Изначальное изображение:

Отфильтрованное изображение:



Параметры фильтрации в формате hsv:

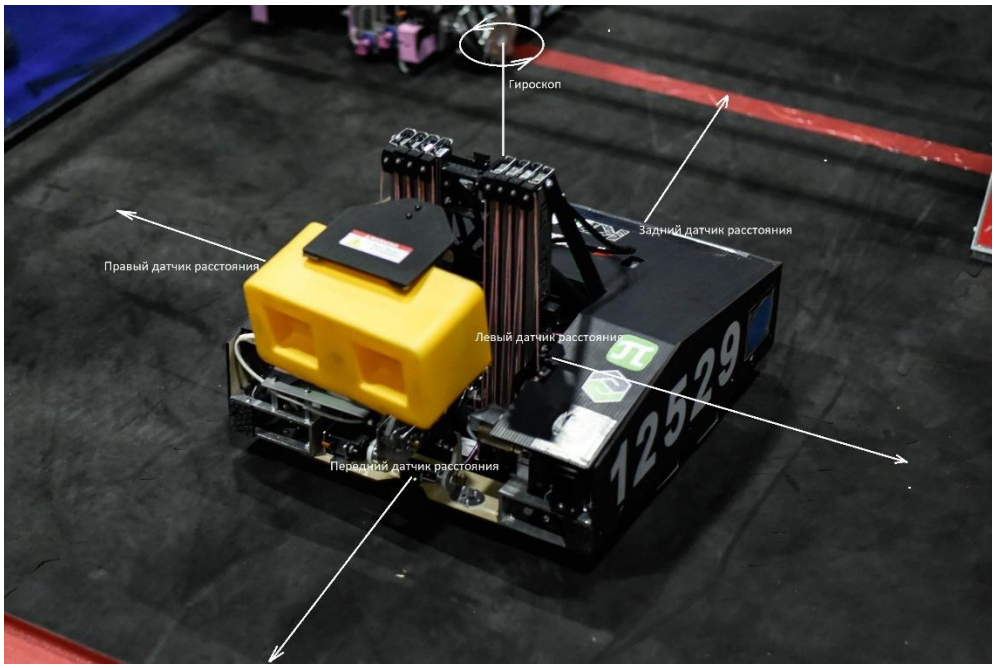


4. Определяем ширину и высоту отфильтрованного пятна встроенными функциями OpenCV.
5. Находим площадь отфильтрованного пятна.
6. Сравниваем найденную площадь с подсчитанными заранее значениями, и исходя из этих вычислений определяем количество колец.

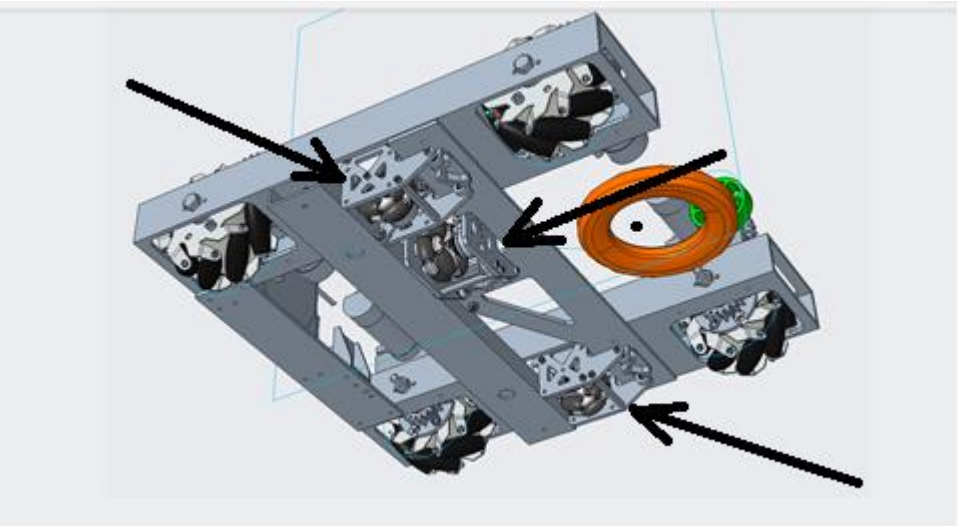
Распознавание отключается после первого определения, так как дальше оно не требуется и напрягать лишними потоками программу не требуется.

Одометрия.

В прошлом году для перемещения по полю мы использовали 4 датчика расстояния и гироскоп, и это было достаточно точно, ведь любая обратная связь помогает роботу держаться заданной траектории. Пример установки датчиков:



В этом году мы получили детали для одометрии и с самого начала сезона начали её настраивать и тестировать. Установили мы 2 горизонтальных и 2 вертикальных енкодера как показано на рисунке ниже:

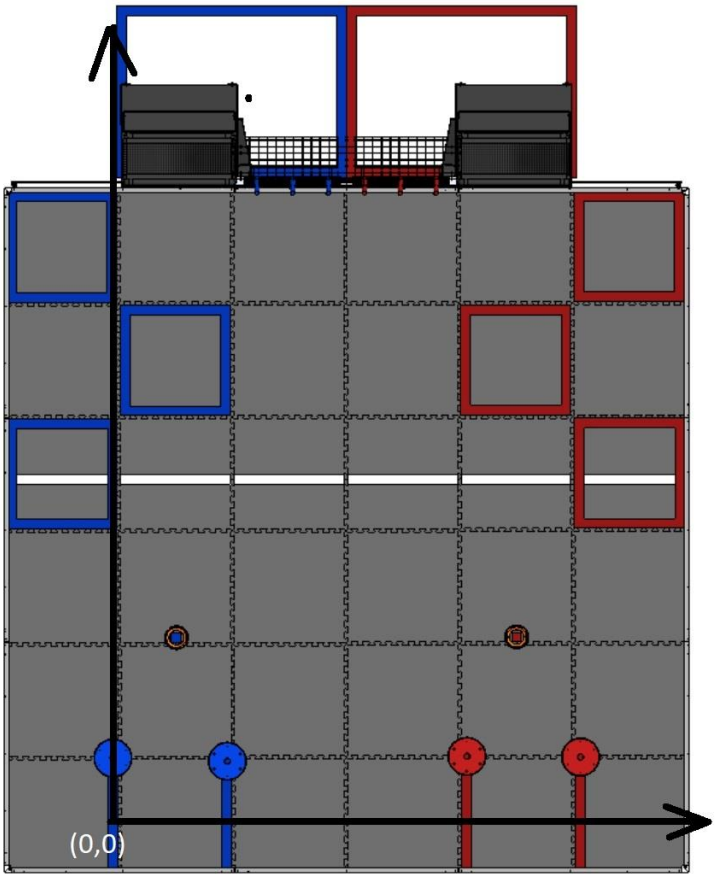


Также, нами были выведены формулы для подсчёта положения робота на поле в реальном времени.

robotXPosition = robotXPosition + (p* sin(robotOrientationRadians) + n*cos(robotOrientationRadians));
robotYPosition = robotYPosition + (p* cos(robotOrientationRadians) - n*sin(robotOrientationRadians));

Где p – это перемещение робота по вертикальной оси, n – это перемещение робота по горизонтальной оси.

Поле для нас стало координатной плоскостью, где оси расположились как показано на следующем рисунке:



Точка (0,0) — это центр робота во время его старта.

Для того чтобы добраться до необходимой точки с необходимым углом была написана функция, которая, учитывая расстояние до необходимой точки вычисляет необходимые мощности для моторов, тем же методом сложения, как и в управлении у 1 оператора, и так будет работать пока робот не попадёт в зону ошибки. Также есть защита от застревания робота – если робот находится на одной и той же координате более 4 секунд и мощность на моторы не 0, то функция завершается.

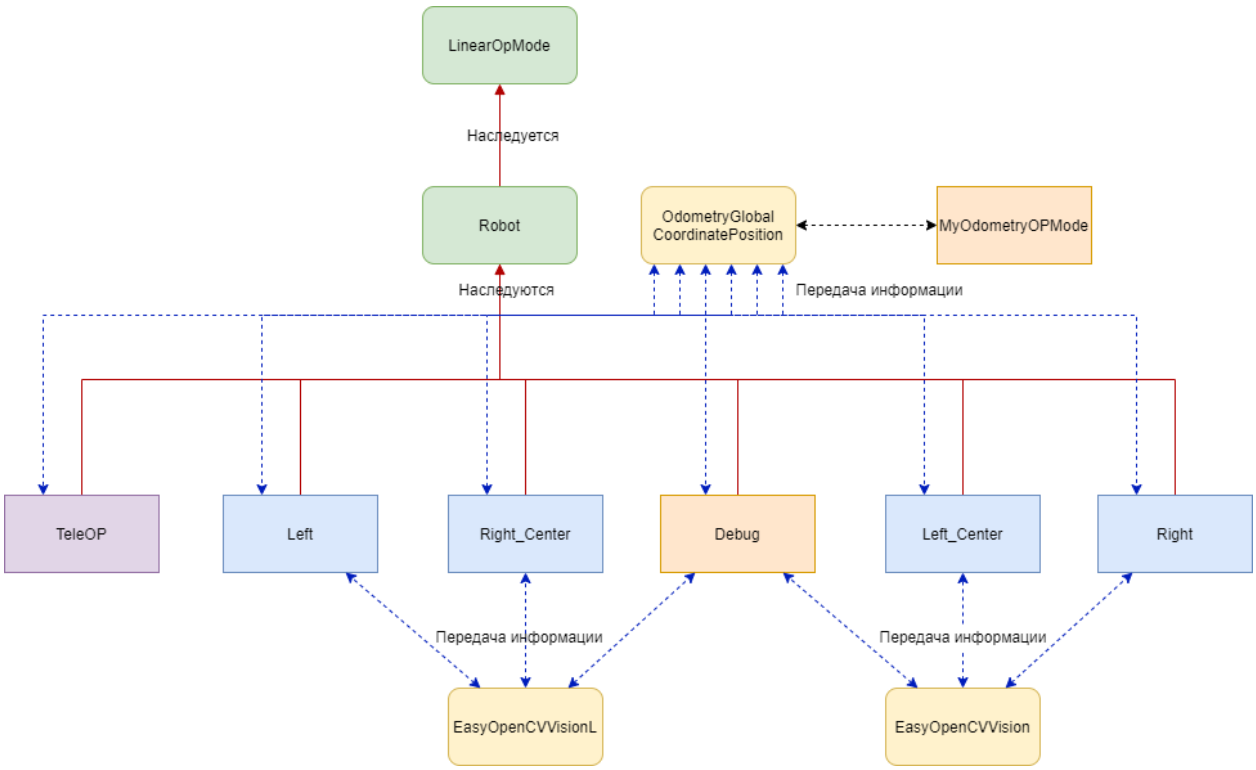
ПИД регулятор.

Структура нашей программы:

В этом году, как и в предыдущих, для программирования мы используем Android Studio и язык программирования Java. В данном году мы решили реализовать одометрию и вновь OpenCV, поэтому наш код немного изменился. Добавились папки для OpenCV и для Одометрии и теперь программа содержит такие классы, как:

1. Robot - это абстрактный класс, который содержит в себе переменные и методы, необходимые для применения в других классах нашей программы
2. AutoOPs
   2.1. Left - программа для автономного периода с левой позиции возле борта
   2.2. Left_Center- программа для автономного периода с левой центральной позиции
   2.3. Right- программа для автономного периода с правой позиции возле борта
   2.4. Right_Center- программа для автономного периода с правой центральной позиции
   2.5. Debug – проверка правильности распознавания количества колец в начале матча и вывод данных с датчиков и со всех енкодеров.
3. Odometry
   3.1. OdometryGlobalCoordinatePosition – отслеживание координат робота (работает в параллельном потоке)
   3.2. MyOdometryOpMode – программа для проверки передвижения робота по полю
4. Vision
   4.1. EasyOpenCVVision – распознавание количества колец, когда кольца находятся справа от робота
   4.2. EasyOpenCVVisionL– распознавание количества колец, когда кольца находятся слева от робота
5. TeleOP2020 – программа для управляемого режима

Для наглядного примера взаимодействия классов можно увидеть uml-диаграмму:



Где синие стрелки обозначают передачу информации (обратную связь) между классами. Красные обозначают наследование.

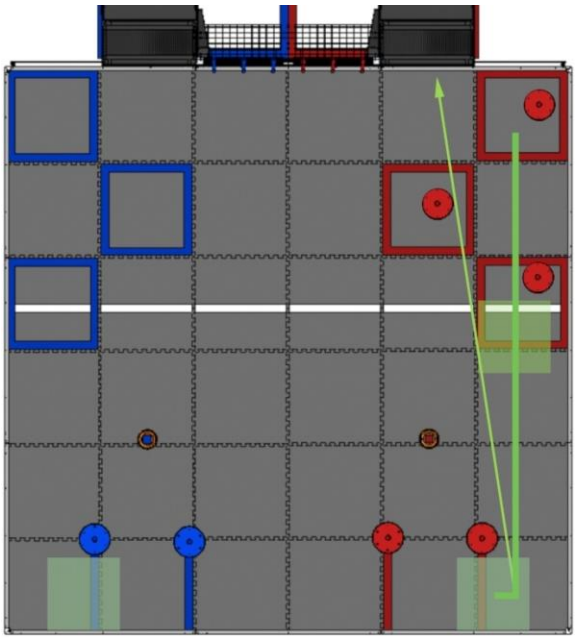
План автономного периода.

На первой встрече, когда мы уже могли стрелять план нашего автономного периода был простой, и некоторым командам даже показался нечестным. На этой встрече мы использовали датчики для перемещения, поэтому робот стрелял почти из изначального своего положения, так как это было точнее всего.

2 встреча СПБ.

Стартовая позиция всегда вдоль борта.

1. распознавание количества колец;
2. перемещение вдоль борта в зону стрельбы;
3. стрельба кольцами в верхние ворота;
4. доставка вобла в соответствующую target zone;
5. парковка на линии.



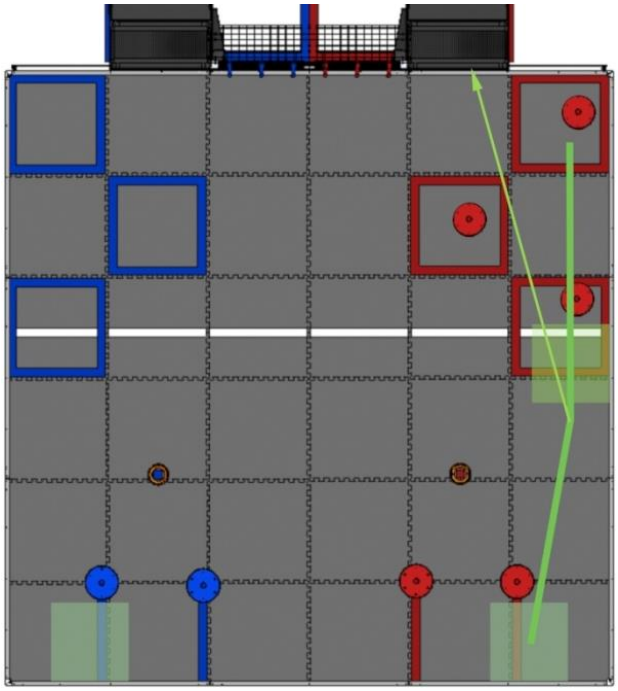
3 встреча СПБ.

Мы доработали наш автономный период и подключили одометрию и теперь робот мог спокойно перемещаться по полю и с улучшенной обратной связью мог совершать хорошее наведение из любой точки на поле.

Автономный период:

Стартовая позиция вдоль борта.

1. распознавание количества колец;
2. перемещение в зону стрельбы;
3. стрельба кольцами в верхние ворота;
4. доставка вобла в соответствующую target zone;
5. парковка на линии.



3 и 4 встреча СПБ. Отборочные СПБ

3 и 4 встречи спб были тестовые для новых версий наших программ и ещё со старым оборудованием. А на отборочных СПБ мы подключили control hub и видеокамеру Logitech и уже с новым оборудованием показали достойный результат.

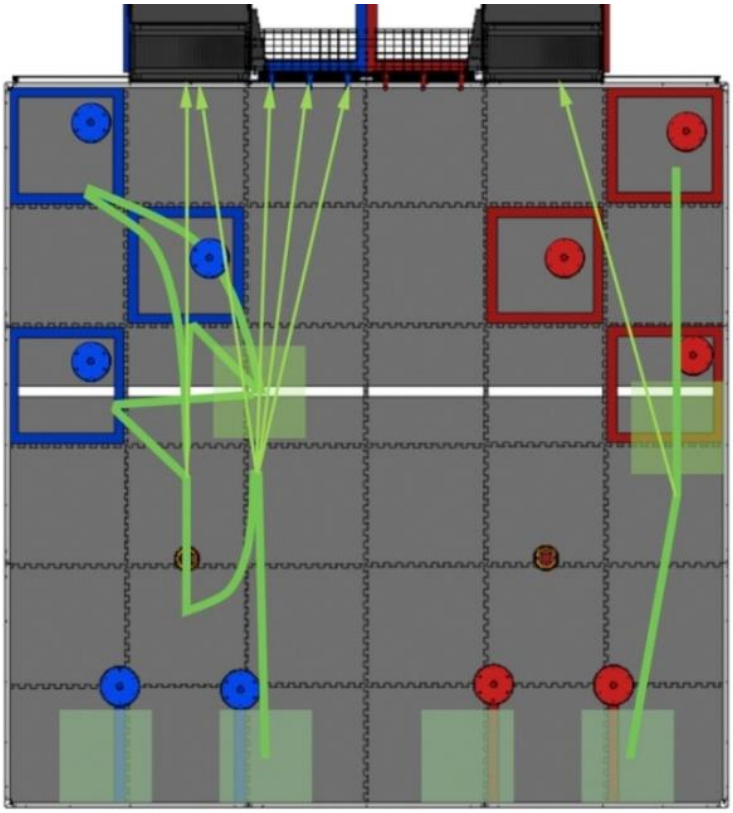
Автономный период:

Стартовая позиция вдоль борта.

1. распознавание количества колец;
2. перемещение в зону стрельбы;
3. стрельба кольцами в верхние ворота;
4. доставка вобла в соответствующую target zone;
5. парковка на линии.

Стартовая позиция ближе к центру поля.

1. распознавание количества колец;
2. перемещение в зону стрельбы;
3. стрельба по мишеням;
4. вариант А – доставка вобла в соответствующую target zone;



вариант B – возврат к кольцу, его захват и стрельба в верхние ворота;

вариант C - возврат к стопке колец, ее разбивание, сбор трёх колец, стрельба в верхние ворота;

5. вариант B и C - доставка вобла в соответствующую target zone;
6. парковка на линии.

План управляемого периода.

Программа управляемого периода, в отличии от прошлого года у нас поменялась, так как была добавлена одометрия. В этом году необходимо отслеживать положение робота на поле и после сбора необходимого количества колец быстро возвращаться на точку стрельбы. Поэтому для 1 оператора было добавлено взаимодействие с одометрией и вынесено всё в отдельный поток, чтобы не нагружать программу. Действия второго оператора также вынесены в отдельный поток и заключаются в сборе колец, вобблов и стрельбе.

Изначально был вариант отдать и стрельбу и перемещение робота одному человеку, но после тренировок и реальных стрессовых ситуаций на соревнованиях наши операторы решили оставить всё как есть, ведь они действовали очень слаженно.

Подробнее о действиях операторов можно узнать в пункте назначения геймпадов

Назначения геймпадов

1 оператор

1 оператор отвечает за движение робота и автоматическую стрельбу по мишеням

2 оператор

2 оператор отвечает за стрельбу кольцами и захватом воббла.

Подробнее про возможности управления.

1 Оператор

Движение и поворот робота осуществляется суммированием значений, получаемых с геймпада. Все стики и триггеры у геймпада1 (геймпад первого оператора):

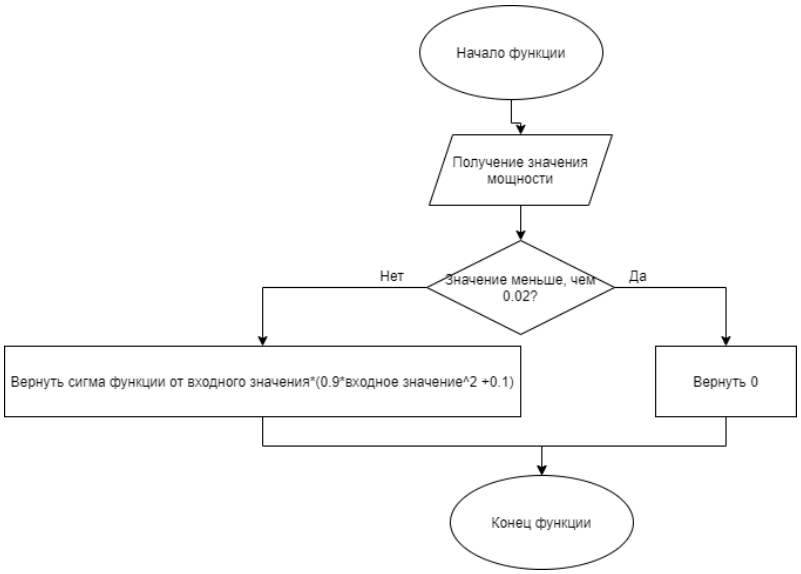
Передний_Правый_Мотор = - (левый_стик_y+крестовина_вверх-крестовина_вниз)(вперед-назад) - левый_стик_x(влево-вправо)+поворот(+левый триггер- правый триггер + правый_стик_x)

Передний_Левый_Мотор = (левый_стик_у+крестовина_вверх-крестовина_вниз)(вперед-назад) - левый_стик_х(влево-вправо)+поворот(+левый триггер- правый триггер + правый_стик_х)

Задний_Правый_Мотор = - (левый_стик_у+крестовина_вверх-крестовина_вниз)(вперед-назад) + левый_стик_х(влево-вправо)+поворот(+левый триггер- правый триггер + правый_стик_х)

Задний_Левый_Мотор = (левый_стик_у+крестовина_вверх-крестовина_вниз)(вперед-назад) - левый_стик_х(влево-вправо)+поворот(+левый триггер- правый триггер + правый_стик_х)

Для более удобного и точного управления роботом 1 оператору была сделана математическая функция для вычисления мощности для моторов, которая исключает движение робота на малых отклонениях стиков и делает функцию мощности нелинейной. На вход функции поступают вычисленные выше значения мощностей для моторов:



1 оператор также отвечает за автоматическую стрельбу по мишеням в конце управляемого периода и автоматический возврат к позиции стрельбы. Эти функции находится у первого оператора, так как для них важно положения робота, а именно 1 оператор отвечает за них.

2 оператор

Во время наших тестов мы столкнулись с проблемой застревания колец в обойме, так как операторы плохо контролировали эту ситуацию. Мы приняли решение сделать активным сбор колец тогда и только тогда, когда наша обойма находится в нижнем положении. Также для устранения ситуаций с несколькими кольцами, лежащими друг на друге, добавлена функция активации и деактивации бампера для разбивания колец.

Также выставление нужного положения захвата воббла было реализовано как через стик, для более точной настройки в непредвиденных ситуациях, так и через енкодер, для стандартных сценариев использования.

Так как автоматическая стрельба не даёт 100 результата по 3 кольцам, то была добавлена кнопка для установки меньшей, чам для стрельбы по воротам, скорости стрельбы по мишеням.