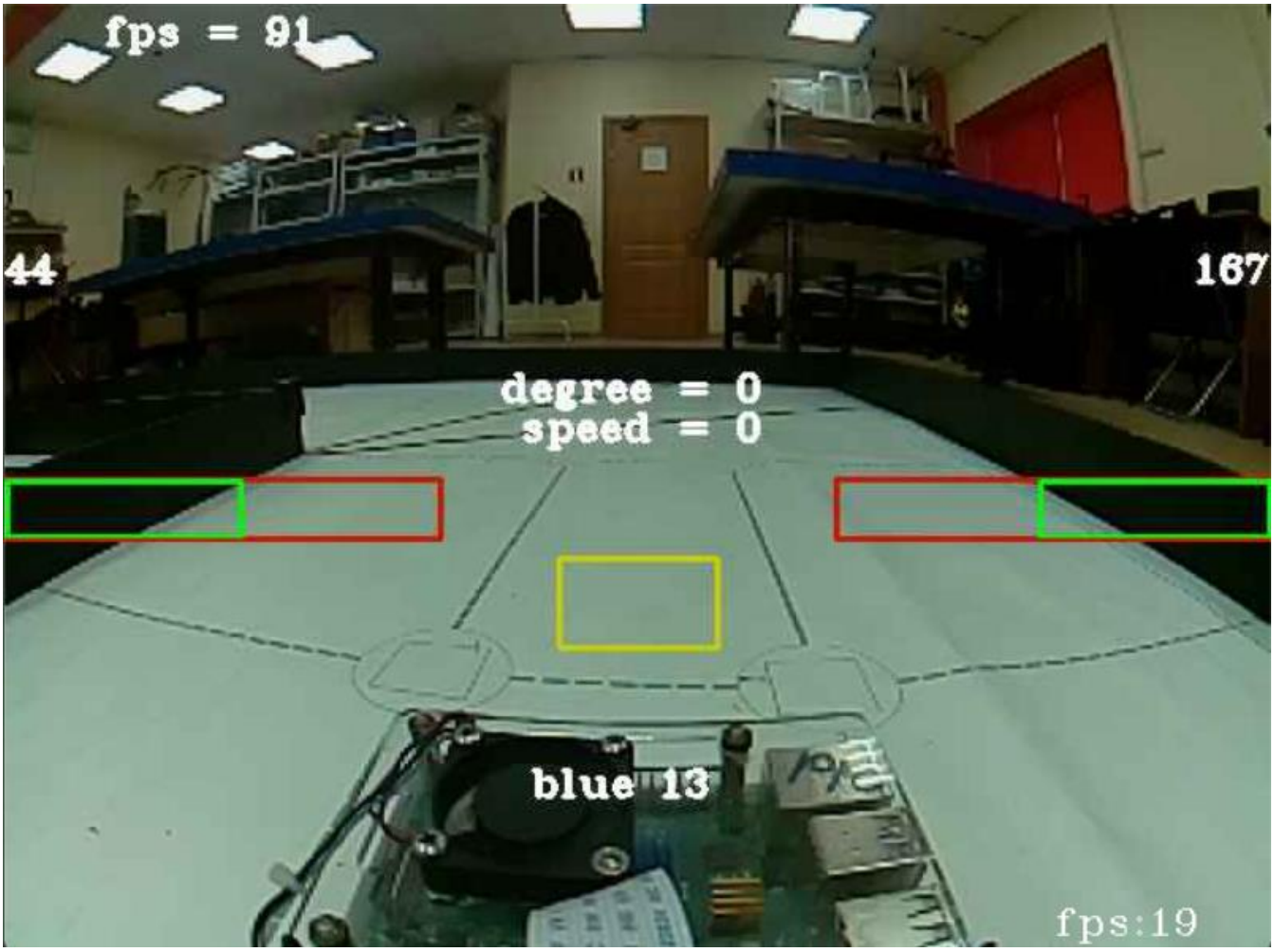
Пояснения к программам

Квалификация

Для начала следует взглянуть на поле глазами робота



На этом фото мы можем видеть поле, телеметрию (текст на экране), а также прямоугольники. Эти прямоугольники — это графическое отображения областей интереса. Они очерчивают границы области поиска тех или иных объектов, например, два красных прямоугольника очерчивают границы областей поиска бортиков, а небольшой жёлтый прямоугольник по центру — область поиска синей и оранжевой линии. В дальнейшем для простоты и краткости область интереса и прямоугольник мы объединим в понятие датчик. То есть на кадре присутствует 3 датчика: 2 датчика бортика и один датчик перекрёстков (перекрёстком называется синяя или оранжевая линия, в зависимости от направления движения).

Датчики

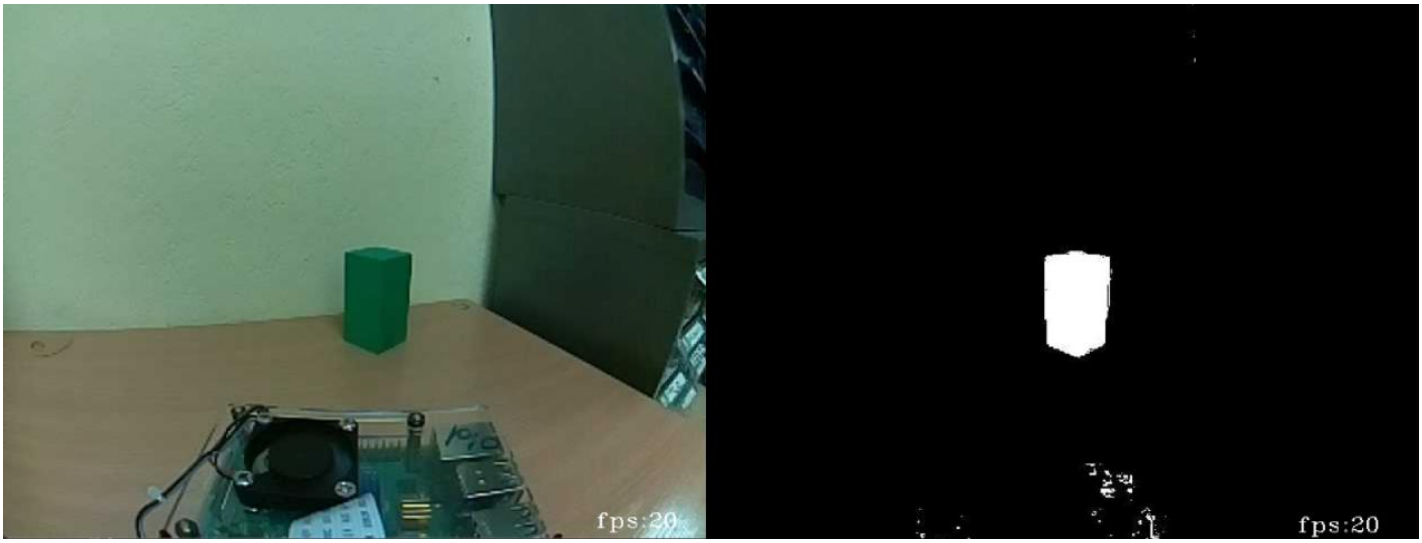
Датчики нужны для того, чтобы определить позицию объекта (бортика, знака, линии) по цвету в модели HSV. Все датчики устроены похожим образом и имеют базовый набор действий:

Скопировать часть кадра, перевести полученное изображение в цветовую модель HSV, создать маску, найти на этой маске контуры, найти наибольший контур, передать его координаты.

В этом наборе есть действие, требующее пояснения, таковым является создание маски.

Создание маски — это процесс создания массива нулей и единиц на основе изображения в цветовом пространстве HSV и диапазона значений. Алгоритм создания таков:

Если цвет пикселя на исходном изображении входит в данный диапазон, то пиксель с такими же координатами на маске становится белым, иначе пиксель будет чёрным.



Выше представлена наглядная демонстрация создания маски

Алгоритм движения

После разбора основных составляющих можно перейти к общему алгоритму движения.

Если кратко, алгоритм таков:

Получить кадр с камеры;
Обработать датчики;
Передать значения, полученные с датчиков, в ПД регулятор;
Проверить наличие перекрёстка;
Передать значение угла поворота из ПД регулятора на микроконтроллер;
Проверить количество перекрёстков, если оно равно 12, остановится.

Но если рассматривать более подробно, то для начала надо бы посмотреть на код программы.

```
wait_for_key()
while 1:
    frame = robot.get_frame(wait_new_frame=1)

    if state == 1:
        red, green, blue = 0, 0, 0
        detect_line_pro()
        pd_regulator(dat1, dat2)
        search_cross()
        if search_cross_time + 0.5 > time.time():
            if color_line == 'orange':
                red = 84
                green = 54
            if color_line == 'blue':
                blue = 80
                green = 20
```

Этот код мы взяли из программы qualification.py и убрали все комментарии. Тут, очевидно, не вся программа, это только фаза движения, которая, в прочем, представляет собой весь алгоритм.

`wait_for_key()` — это функция ожидания нажатия кнопки, она посылает на микроконтроллер сообщение '9999999999999999999$', которое свидетельствует о том, что распберри запустилась, и начала выполнять программу. Параллельно происходит чтение сообщений с микроконтроллера, которые содержат всего одну цифру, ноль либо единицу. Ноль — кнопка не нажата, один — кнопка нажата. Как только приходит сообщение о нажатой кнопке, распберри выходит из этой функции.

`frame = robot.get_frame(wait_new_frame=1)` — этой строчкой мы получаем снимок с камеры, который мы будем использовать в дальнейшем.

`if state == 1:` — переменная state отвечает за режимы работы робота, 1 — выполнение задания, 2 — активное торможение, 3 — остановка.

`red, green, blue = 0, 0, 0` — эти три переменные отвечают за цвета RGB светодиода.

`detect_line_pro()` — эта функция отвечает за работу датчиков бортиков, но не отрисовывает их на экране. Внутри у неё не так много кода, потому что она использует две другие функции и упрощает работу с ними. Её задача просто приравнять глобальные переменные dat1 и dat2, к показаниям датчиков.

`pd_regulator(dat1, dat2)` — эта функция обыкновенный ПД-регулятор с некоторыми дополнениями в виде частных случаев, в роде: если левый датчик равен нулю, следовательно мы на повороте и только что проехали угол, следовательно сейчас необходимо сделать поворот влево.

`search_cross()` — эта функция занимается поиском перекрёстков на соответствующем датчике. Эта же функция записывает время зон в список.

А далее идет индикация обнаружения перекрёстков.

И это весь алгоритм движения, дальше только активное торможение и остановка.

На raspberry находится и выполняется основная программа, что подразумевает больше трудностей и особенностей, и соответственно больше алгоритмов. Для удобства разделим все алгоритмы на две группы, а именно на алгоритмы поиска и алгоритмы движения, а имеющиеся группы разделим по заездам на квалификационный и финальный.

Алгоритмы поиска

Вставить картинку

Как мы можем видеть для прохождения квалификационного заезда не нужно много датчиков (датчиками условно называются области интереса, в которых мы что-то ищем, например на двух датчиках по краям экрана мы ищем черный бортик).

На записи квалификации мы видим только три датчика, два датчика для чёрного бортика и один датчик для так называемых перекрёстков (перекрёстками условно называются оранжевые и синие линии). У всех датчиков, несмотря на разные задачи и разные объекты поиска, имеется примерно одинаковый набор действий по нахождению предметов интереса. Классический набор действий включает в себя: выбор части изображения (области интереса), размытие её по Гауссу, перевод изображения в цветовое пространство HSV, получение маски (маской называется черно-белое изображение, полученное посредством изменения цвета всех пикселей изображения, все пиксели, попавшие в установленный диапазон, становятся белыми, а все остальные становятся чёрными), поиск контуров на этой маске, и последующее нахождение координат этих контуров. Полученные координаты контуров обычно и являются выходными данными датчиков.

Теперь разберём датчики в отдельности, начнём с датчиков чёрного бортика (далее датчики линии).

Датчики линии занимаются поиском черного цвета и в набор их действий входят все вышеописанные операции. Левый датчик определяет самое правое положение черного цвета, а правый датчик определяет самое левое положение чёрного цвета. Исходя из показаний этих двух датчиков считается управляющее воздействие, но к нему мы вернёмся в разделе “Алгоритмы движения”.

Датчики линии постоянно видят свой объект интереса, в отличии от датчика перекрёстка. Датчик перекрёстка, как следует из названия, ищет перекрёстки и считает их количество. Но перекрёсток находится на экране не один кадр, следовательно возможны ложные срабатывания и некорректное подсчитывание кругов. Исходя из этого мы написали алгоритм, решающий эти проблемы. При обнаружении перекрёстка мы запускаем таймер, и регистрируем перекрёсток только если с предыдущего перекрёстка прошло какое-то время, зависящее от скорости.

Также в алгоритм работы всех датчиков включено ограничение по минимальной площади контура.

В программе для финального заезда добавляются ещё два датчика, для поиска знаков. Они отличаются только тем, что они ищут разные цвета.

Алгоритмы движения

Самый основной алгоритм движения, который используется в каждой из программ — ПД-регулятор.

ПД-регулятор — это алгоритм, высчитывающий управляющее воздействие, исходя из показаний датчиков. В квалификации мы используем один и тот же регулятор на протяжении всего заезда, но бывают ситуации в которых регулятор не помогает, и для таких ситуаций нужны особые алгоритмы. Например: при потере одного датчика (потеря датчика — ситуация, в которой показания датчика равны нулю) мы должны повернуть в направлении этого датчика, то есть приблизиться к бортику, от которого мы отдалились. А если потеряны оба датчика, робот должен произвести поворот в направлении движения, которое мы определяем по цвету первого перекрёстка.

В программе для финального заезда добавляются ещё два ПД-регулятора, для зеленого и красного знаков.