**Прототип беспилотного автомобиля обладающего техническим зрением для автономного движения.**

Программа:

Разработчик. Программист. Цифровые профессии

Специализация

Программист Python

ФИО

Надымов Александр Васильевич

Город Канск

Год 2024

Оглавление

[Введение 3](#_Toc159159675)

[Глава 1. Беспилотный автомобиль 5](#_Toc159159676)

[Глава 2. Компьютерное зрение 6](#_Toc159159677)

[Глава 3. Работа микроконтроллера 18](#_Toc159159678)

[Глав 4. Общая программа 21](#_Toc159159679)

[Глава 5. Заключение 28](#_Toc159159680)

[Список литературы 29](#_Toc159159681)

# **Введение**

**Тема проекта:** Разработать прототип беспилотного автомобиля обладающего техническим зрением для автономного движения

**Цель:** создать прототип беспилотного автомобиля

**Какую проблему решает:** рассматривает возможность применения технического зрения в беспилотных автомобилях.

**Задачи:**

1. Изучить материал по данной теме
2. Собрать прототип автомобиля
3. Написать программу
4. Протестировать работу прототипа
5. Сделать вывод.

**Проблема**: С каждым годом, количество автомобилей на дорогах страны растет. По данным официальной статистики ГИБДД за 2023 год, (рисунок 1) в стране произошло почти 120000 дорожно-транспортных происшествий, из них 90% произошли по вине человека [2]

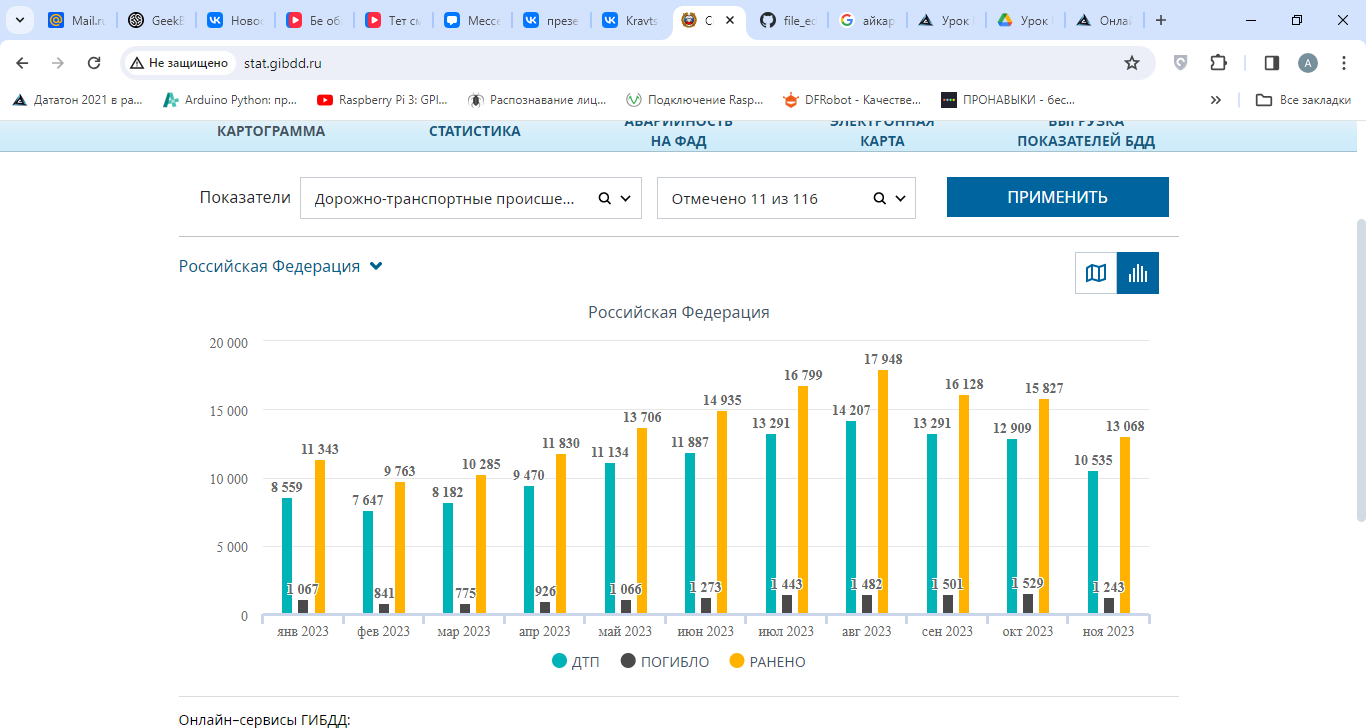


Рисунок 1 График ДТП за 2023 год

Основные причины ДТП связанны с: осознанным нарушением правил дорожного движения, несоблюдением дистанции, вождение в нетрезвом состоянии, выезд на полосу встречного движения, превышение скорости, нарушение правил обгона, невнимательность [3] т.е. человеческий фактор.

Предлагаемое решение позволит снизить влияние человеческого фактора, тем самым сократить количество ДТП.

Прототип состоит из камеры, компьютера, управляющего контроллера, драйвера мотора, сервомотора, колесной базы масштаба 1:10.

При работе с техническим зрением, будет использован язык программирования Python с установленными библиотеками OpenCV, NumPy, а также C++ для работы контроллера.

Камера передает изображение на компьютер со скоростью несколько кадров секунду. Всё изображение - это одна большая переменная, имеющая своё имя. Переменные, в которых хранятся изображения, представляют собой массивы. Смысл массива в том, чтобы можно было работать не только со всем изображением, но и с каждым его пикселем. Каждый пиксель имеет своё "имя", состоящее из названия массива и индексов: номера строки и номера столбца, в котором находится пиксель.

Компьютер обрабатывает изображения и передает данные на контроллер, где находится дорожная разметка, какие знаки увидел, какие препятствия встретил, какой сигнал светофора.

Контроллер, на основе полученных данных, от компьютера, мотор драйвера и и сервомотора упраляет рулем и мотором.

Схема подключения копонентов отображена на (рисунке 2)



Рисунок 2 Схема подключения компонентов

# **Глава 1. Беспилотный автомобиль**

Беспилотный автомобиль — это транспортное средство, оборудованное системой автоматического управления, которое может передвигаться без участия человека [6].

Преимущества беспилотного автомобиля

* кардинальная минимизация ДТП и почти полное исключение человеческих жертв, отсюда значительное снижение расходов на страхование и медицину быстрого реагирования, снижение нагрузки на пожарных, полицейских, МЧС;
* снижение стоимости транспортировки грузов и людей за счёт экономии на заработной плате и времени отдыха водителей, а также экономии топлива, экономии на ремонте, так как нет аварий;
* снижение потребности в индивидуальных автомобилях за счёт развития сервисов беспилотных такси и развития беспилотного общественного транспорта;
* повышение эффективности использования дорог за счёт полного соблюдения автономными автомобилями правил дорожного движения (при массовом выходе беспилотных автомобилей на дороги);
* Плавный уход рутинных профессий (водитель такси, автобуса, грузовика, трамвая, троллейбуса, тракторист, комбайнёр, машинист...) и появление новых, связанных с разработкой и обслуживанием автономного транспорта.
* перевозка грузов в опасных зонах, во время природных и техногенных катастроф или военных действий без риска для жизни и здоровья людей;
* в более отдалённой перспективе снижение глобальной экологической нагрузки как за счёт количественной оптимизации парка автомобилей, так и за счёт более широкого использования для их передвижения альтернативных видов энергии.[5]

Уровни автономности

Классификация автоматизации автомобилей разработана Сообществом автомобильных инженеров (SAE) и содержит 6 уровней.

Уровень 0. Никакой автоматизации, водитель выполняет всю работу.

Уровень 1, «hands on», «помощь водителю». Водитель и система вместе управляют автомобилем. Пример: водитель рулит, а система регулирует мощность двигателя, сохраняя заданную скорость (круиз-контроль) или регулирует мощность двигателя и управляет тормозом, сохраняя заданную скорость, а при необходимости снижая, чтобы соблюдать дистанцию (адаптивный круиз-контроль). Другим примером является автоматическая парковка (англ.)рус., когда скорость определяется водителем, а руление автоматическое.

Уровень 2, «hands off», «частичная автоматизация». Система полностью управляет автомобилем, осуществляя ускорение, торможение и рулёжку. Водитель следит за ездой и готов вмешаться в любой момент, если система не может правильно отреагировать. Несмотря на название «hands off», такие системы часто требуют от водителя держать руки на руле, как подтверждение готовности вмешаться.

Уровень 3, «eyes off», «условная автоматизация». От водителя не требуется немедленной реакции. Он может, например, писать сообщения или смотреть фильм. Система сама реагирует на ситуации, требующие немедленных действий, таких как экстренное торможение. От водителя требуется готовность вмешаться в течение какого-то ограниченного времени, определённого производителем.

Уровень 4, «mind off», «широкая автоматизация». Отличается от уровня 3 тем, что от водителя не требуется постоянного внимания. Например, он может лечь спать или покинуть место водителя. Полностью автоматическое вождение осуществляется лишь в некоторых пространственных областях (геозонах) или в некоторых ситуациях, например, в пробках. Вне таких мест или ситуаций система способна прекратить вождение и припарковать машину, если водитель не взял управление на себя.

Уровень 5, «steering wheel optional», «полная автоматизация». Никакого человеческого вмешательства не требуется.

# **Глава 2. Компьютерное зрение**

Компьютерное зрение - это область компьютерных наук, математики и электротехники. Он включает в себя способы получения, обработки, анализа и понимания изображений и видео из реального мира, чтобы имитировать человеческое зрение. Кроме того, в отличие от человеческого зрения, компьютерное зрение также может использоваться для анализа и обработки глубинных и инфракрасных изображений.

Компьютерное зрение также связано с теорией извлечения информации из изображений и видео. Система компьютерного зрения может принимать различные формы данных в качестве входных данных, включая, но не ограничиваясь ими, изображения, последовательности изображений и видео, которые могут передаваться из нескольких источников для дальнейшей обработки и извлечения полезной информации для принятия решений.

Типичные задачи компьютерного зрения включают в себя следующее:

* Распознавание и классификация объектов
* Обнаружение движения и анализ
* Реконструкция изображения и сцены [3]

OpenCV (Open Source ComputerVision) - это библиотека функций программирования для компьютерного зрения. Это кроссплатформенная библиотека, что означает, что она может быть реализована и работать в разных операционных системах. Основное внимание уделяется обработке изображений и видео. В дополнение к этому, он имеет несколько графических интерфейсов и функций обработки событий для удобства пользователя.

Numpy — это пакет языка Python, предназначенный для работы с большими многомерными массивами и матрицами. Содержит в себе большую библиотеку высокоуровневых (и очень быстрых) математических функций для операций с этими массивами1, 2. В дальнейшем этот пакет нам понадобится для работы с компьютерным зрением. Данный пакет является обязательным компонентом для работы с OpenCV [2].

Нам нужно получить изображение, обработать изображение и распознать объекты

Получение изображения.

Для получения изображения с камеры и вывода изображения на экран используем следующий код [9]:

# Подключаем библиотеку  
import cv2 as cv  
  
# Видеозахват с камеры и присваеванм переменной cap  
cap = cv.VideoCapture(0)  
# В цикле считываем изображение в переменную frame и выводим картинку  
while True:  
 ret, frame = cap.read()  
 cv.imshow('frame', frame)  
 # выводим как выглядит переменная frame  
 print(frame)  
 # выводим тип переменной frame  
 print(type(frame))  
 # для корректной остановки программы используем условие по нажатию клавиши  
 if cv.waitKey(1) == ord('q'):  
 break  
# останавливаем видео захват и закрываем все окна  
cap.release()  
cv.destroyAllWindows()

фрагмент вывода информации в консоль

[[ 3 10 21]  
 [ 3 10 21]  
 [ 4 10 24]  
 ...  
 [188 208 216]  
 [195 213 223]  
 [200 218 228]]]  
<class 'numpy.ndarray'>

В выводе хорошо видно что, картинка это массив, каждый элемент которого состоит из массива с 3 числами, а это означает, что работа с изображением это работа с многомерным массивом. Каждый элемент массив это пиксель, а числа это цвет в формате BGR (Blue, Green, Red).

Распознавание дорожных знаков по цветам

Дорожный знаки имеют основные цвета красный и синий, именно эти цвета можно выделить. Из исходного изображения делаем черно-белое (бинаризуем по порогу). Разрешенные цвета станут белыми, а запрещенные – черными. для получения значений порога бинаризации используем специальную программу.

import cv2 as cv  
  
def nothing(x):  
 pass  
  
# Получаем изображение с камеры  
cap = cv.VideoCapture(0)  
# создаем окно  
cv.namedWindow('result')  
# создаем ползунки для минимальных и максимальных значений цветов  
cv.createTrackbar('minB', 'result', 0, 255, nothing)  
cv.createTrackbar('minG', 'result', 0, 255, nothing)  
cv.createTrackbar('minR', 'result', 0, 255, nothing)  
cv.createTrackbar('maxB', 'result', 0, 255, nothing)  
cv.createTrackbar('maxG', 'result', 0, 255, nothing)  
cv.createTrackbar('maxR', 'result', 0, 255, nothing)  
  
signal = cv.imread('znak2.jpg')  
  
while True:  
 ret, frame = cap.read()  
 # сохраняем текущее положение ползунка  
 minB = cv.getTrackbarPos('minB', 'result')  
 minG = cv.getTrackbarPos('minG', 'result')  
 minR = cv.getTrackbarPos('minR', 'result')  
 maxB = cv.getTrackbarPos('maxB', 'result')  
 maxG = cv.getTrackbarPos('maxG', 'result')  
 maxR = cv.getTrackbarPos('maxR', 'result')  
  
 mask2 = cv.inRange(signal, (minB, minG, minR), (maxB, maxG, maxR))  
 cv.imshow('mask2', mask2)  
 cv.imshow('signal', signal)  
  
 if cv.waitKey(1) == ord('q'):  
 break  
cap.release()  
cv.destroyAllWindows()

Возьмем исходное изображение со знаками (рисунок3). Преобразуем изображение из формата BGR в HSV (рисунок 4). Применим функцию размытия (рисунок 5) и передвигая ползунки (Рисунок 6) и добиваемся изображения, на котором будут видны только знаки (Рисунок 7).

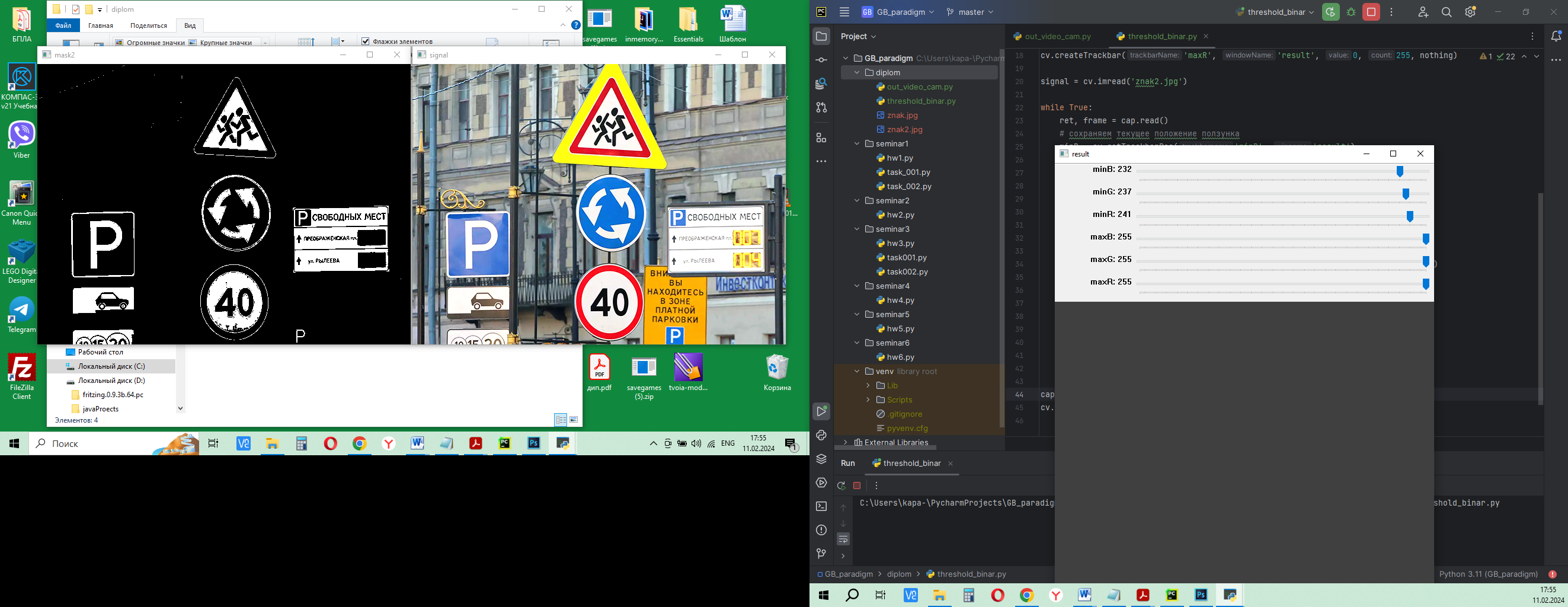


Рисунок 3 исходное изображение

|  |  |
| --- | --- |
| Рисунок 4 до применение функции размытия | Рисунок 5 после применения функции размытия |

|  |  |
| --- | --- |
| Рисунок 6 | Рисунок 7 после бинаризации |

С помощью морфологических преобразований Image Dilate (Рисунок 9) и Image Erosion (Рисунок 9), удалим небольшие шумы на уже бинаризованной картинке и увеличим видимость нужных объектов.

|  |  |
| --- | --- |
| Рисунок 8 функция Image Erosion | Рисунок 9 функция Image Dilate |

После подбора необходимых значений, программа выглядит следующим образом:

import cv2 as cv  
  
signal = cv.imread('znak.jpg')

# подобранные значения  
minBlue = 102  
minGreen = 109  
minRed = 134  
maxBlue, maxGreen, maxRed = 255, 255, 255  
  
while True:  
 cv.imshow('signal', signal)  
 hsv = cv.cvtColor(signal, cv.COLOR\_BGR2HSV)  
 hsv = cv.blur(hsv, (5, 5))  
  
 mask = cv.inRange(hsv, (minBlue, minGreen, minRed), (maxBlue, maxGreen, maxRed))  
 mask = cv.erode(mask, (5, 5), iterations=2)  
 mask = cv.dilate(mask, None, iterations=2)  
 cv.imshow('dilate', mask)  
  
 if cv.waitKey(1) == ord('q'):  
 break  
  
cv.destroyAllWindows()

Применим к найденным изображениям функцию поиска контуров findConturs, отсортируем найденные контуры и выведем их на копии исходного изображения. Из всех контуров нам необходимы только самые большие контуры. Отсортируем их по убыванию и выведем самый большой из них. Функция sorted не может работать с пустыми массивами, поэтому сортировку и выделение контуров выполняем, только когда они есть (Рисунок 10)

|  |  |
| --- | --- |
| Рисунок 10 | Рисунок 11 |

Далее из всего изображения выделяем прямоугольную область с найденным знаком, с помощью функции нахождения координат boundingRect() (Рисунок 11).

Из общего изображения вырезаем найденный знак с помощью стандартных операций над массивами (Рисунок 12).

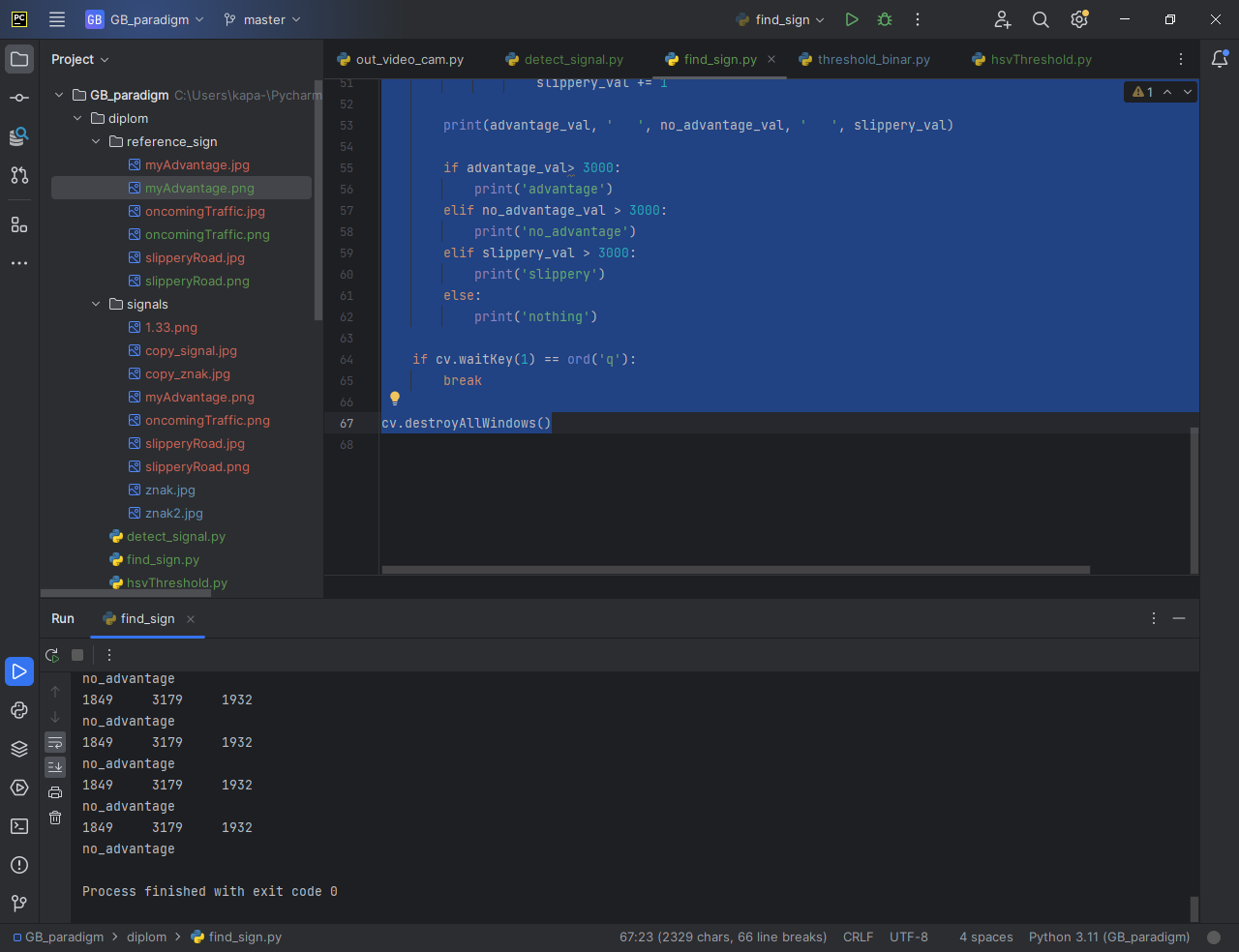
|  |  |
| --- | --- |
| Рисунок 12 |  |

Код программы выглядит следующим образом

import cv2 as cv  
  
signal = cv.imread('signals/znak2.jpg')  
minBlue = 102  
minGreen = 109  
minRed = 134  
maxBlue, maxGreen, maxRed = 255, 255, 255  
cap = cv.VideoCapture(0)  
while True:  
 signal\_copy = signal.copy()  
 hsv = cv.cvtColor(signal, cv.COLOR\_BGR2HSV)  
 hsv = cv.blur(hsv, (5, 5))  
  
 mask = cv.inRange(hsv, (minBlue, minGreen, minRed), (maxBlue, maxGreen, maxRed))  
 mask = cv.erode(mask, None, iterations=2)  
 mask = cv.dilate(mask, None, iterations=4)  
 cv.imshow('dilate', mask)  
  
 contours, hierarchy = cv.findContours(mask, cv.RETR\_TREE, cv.CHAIN\_APPROX\_NONE)  
  
 if contours:  
 contours = sorted(contours, key=cv.contourArea, reverse=True)  
 cv.drawContours(signal\_copy, contours, -1, (255, 0, 255), 2)  
 cv.imshow('countours', signal\_copy)  
  
 (x, y, w, h) = cv.boundingRect(contours[0])  
 cv.rectangle(signal\_copy, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 cv.imshow('rect', signal\_copy)  
  
 roiImg = signal[y:y + h, x:x + w]  
 cv.imshow('detect', roiImg)  
  
 if cv.waitKey(1) == ord('q'):  
 break  
  
cv.destroyAllWindows()

Для распознавания знаков необходимо полученное изображение и эталонное изображение. Эталонное изображение считываем и уменьшаем их размер до 64 х 64 пикселя. Выбранный размер зависит от скорости работы компьютера, чем больше размер, тем точнее определения, но медленнее работает распознавание, и наоборот, чем меньше размер, тем меньше точность определения. Полученное изображение уменьшаем до такого же значения и все изображения бинаризуем (делаем их черно-белыми). Далее попиксельно сравниваем полученное бинаризованное изображение с бинаризованными эталонными изображениями. По наибольшему количеству совпадающих пикселей определяем полученный знак. Если полученных значений мало, значит распознали не знак.

Вывод значений в консоль



Весь код программы

import cv2 as cv  
  
signal = cv.imread('signals/oncomingTraffic.png')  
minBlue = 0  
minGreen = 109  
minRed = 114  
minInRange = (minBlue, minGreen, minRed)  
maxBlue, maxGreen, maxRed = 255, 255, 255  
maxInRange = (maxBlue, maxGreen, maxRed)  
pixel = 64  
min\_pixels = (pixel, pixel)  
  
advantage = cv.inRange(cv.resize(cv.imread('reference\_sign/myAdvantage.jpg'), min\_pixels), minInRange, maxInRange)  
no\_advantage = cv.inRange(cv.resize(cv.imread('reference\_sign/oncomingTraffic.jpg'), min\_pixels), minInRange,  
 maxInRange)  
slippery = cv.inRange(cv.resize(cv.imread('reference\_sign/slipperyRoad.jpg'), min\_pixels), minInRange, maxInRange)  
  
while True:  
 signal\_copy = signal.copy()  
 hsv = cv.cvtColor(signal, cv.COLOR\_BGR2HSV)  
 hsv = cv.blur(hsv, (5, 5))  
  
 mask = cv.inRange(hsv, (minBlue, minGreen, minRed), (maxBlue, maxGreen, maxRed))  
 mask = cv.erode(mask, None, iterations=2)  
 mask = cv.dilate(mask, None, iterations=4)  
 cv.imshow('signal', mask)  
  
 contours, hierarchy = cv.findContours(mask, cv.RETR\_TREE, cv.CHAIN\_APPROX\_NONE)  
  
 if contours:  
 contours = sorted(contours, key=cv.contourArea, reverse=True)  
 cv.drawContours(signal\_copy, contours, -1, (255, 0, 255), 2)  
  
 (x, y, w, h) = cv.boundingRect(contours[0])  
 cv.rectangle(signal\_copy, (x, y), (x + w, y + h), (0, 255, 0), 2)  
  
 roiImg = signal[y:y + h, x:x + w]  
 roiImg = cv.inRange(cv.resize(roiImg, min\_pixels), minInRange, maxInRange)  
 cv.imshow('detect', roiImg)  
  
 advantage\_val = 0  
 no\_advantage\_val = 0  
 slippery\_val = 0  
 for i in range(pixel):  
 for j in range(pixel):  
 if roiImg[i][j] == advantage[i][j]:  
 advantage\_val += 1  
 if roiImg[i][j] == no\_advantage[i][j]:  
 no\_advantage\_val += 1  
 if roiImg[i][j] == slippery[i][j]:  
 slippery\_val += 1  
  
 print(advantage\_val, ' ', no\_advantage\_val, ' ', slippery\_val)  
  
 if advantage\_val> 3000:  
 print('advantage')  
 elif no\_advantage\_val > 3000:  
 print('no\_advantage')  
 elif slippery\_val > 3000:  
 print('slippery')  
 else:  
 print('nothing')  
  
 if cv.waitKey(1) == ord('q'):  
 break  
  
cv.destroyAllWindows()

Пример приведен для трех знаков, но по желанию можно добавлять необходимое количество.

Распознавание сигналов светофора

Для распознавания предварительно загрузим картинки светофора в отдельную папку. Приводим изображение найденного светофора к одному размеру, обрезаем посторонний фон (Рисунок 13), все полученные изображения переводим в формат HSV. В формате HSV составляющая V – отвечает за яркость, а значит, используя только эту составляющую, можно получить наиболее яркую область светофора (Рисунок 14). Изображение станет черно белое. Белый цвет означает, какой цвет горит на светофоре.

Разделяем кадр с изображением светофора на три части и верхнюю, среднюю и нижнюю. Складываем числа, которые отвечают за цвет пикселя на черно-белой картинке в этих трех частях, получим три суммы, сумму чисел для красной (верхней) зоны, для желтой (средней) зоны и зеленой (нижней) зоны. И сравнив эти суммы, получим, в какой из зон наибольшее число (Рисунок 15), а значит, в этой зоне зажжен сигнал светофора.

|  |  |
| --- | --- |
| Рисунок 13 формат BGR | Рисунок 14 из HSV только V |

|  |
| --- |
| Рисунок 15 вывод в консоль результаты вычисления |

Программа распознавания сигналов светофора выглядит следующим образом:

import cv2 as cv  
import numpy as np  
  
heightLT = 210  
widthLT = 70  
i = 1  
while True:  
 # считываем картинку  
 rgb\_light = cv.resize(cv.imread('lightTraffic/' + str(i) + '.jpg'), (widthLT, heightLT))  
 cv.imshow('light', rgb\_light)  
  
 # обрезаем фон  
 cutTraffic = rgb\_light[10:200, 12:58]  
 cv.imshow('cut', cutTraffic)  
  
 # переводим в формат HSV и выделяем самый яркий свет  
 hsv = cv.cvtColor(cutTraffic, cv.COLOR\_BGR2HSV)  
 v = hsv[:, :, 2]  
 cv.imshow('v', v)  
  
 # складываем значения пикселей в каждой зоне  
 redSum = np.sum(v[0:63, 0:46])  
 yelSum = np.sum(v[64:127, 0:46])  
 greSum = np.sum(v[128:190, 0:46])  
 print('redSum', redSum, ' ', 'yelSum', yelSum, ' ', 'greSum', greSum)  
  
 # увидеть выделенные области  
 # cv.rectangle(cutTraffic, (0, 0), (46, 63), (0, 0, 255), 2)  
 # cv.rectangle(cutTraffic, (0, 64), (46, 127), (0, 255, 255), 2)  
 # cv.rectangle(cutTraffic, (0, 128), (46, 190), (0, 2500, 0), 2)  
 # cv.imshow('region', cutTraffic)  
  
 # сравниваем полученные суммы  
 if greSum > yelSum and greSum > redSum:  
 print('green')  
 elif yelSum > greSum and yelSum > redSum:  
 print('yellow')  
 else:  
 print('red')  
  
 key = cv.waitKey(1)  
 if key == ord('n'):  
 i += 1  
 cv.destroyAllWindows()  
  
 if cv.waitKey(1) == ord('q'):  
 break  
  
cv.destroyAllWindows()

Распознавание дорожной разметки

Есть исходное изображение с камеры (Рисунок 16). Камера расположена на автомобиле по центру и смотрит перед собой. Из исходной картинки вырезаем область интересов, чтобы видеть только разметку и не обращать внимания на постороннее изображение, кроме того увеличивается скорость обработки выделяемого изображения. Бинаризуем выбранную область (Рисунок 17), подбирая пороги бинаризации чтобы исключить ненужные элементы. Вычисляем где находиться самая белая линия и запоминаем их расположение, Вычисляем положение белой линии относительно центра кадра и находим положение автомобиля относительно центра траектории (Рисунок 18).

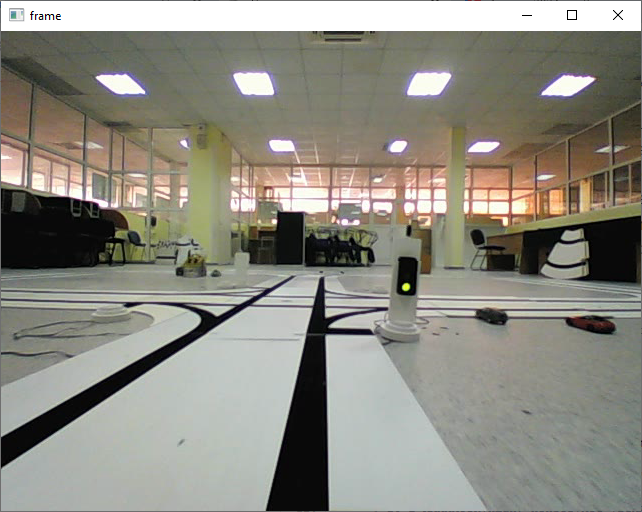


Рисунок 16 изображение с камеры



Рисунок 17 вырезана область интересов и бинаризация

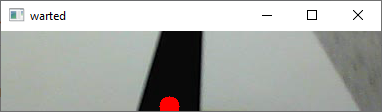


Рисунок 18 получен и отмечен центр линии

Программа движения по линии приведена ниже. Изображение с камеры заменено на изображение с видео ролика.

import serial  
import time  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
# ser = serial.Serial('/dev/ttyACM0',115200, timeout=1)  
# ser.flush()  
  
cap = cv.VideoCapture('videoLine/allSignsLight.avi')  
k = 0.11915 # 0.11915  
while True:  
 ret, frame = cap.read()  
 frameCopy = frame.copy() # копируем frame в переменную frameCopy  
 cv.imshow("frame", frameCopy)  
 fps = cv.waitKey(25)  
  
 road = frameCopy[400:480, 130:510] #  
 road2 = road.copy()  
 cmap = cv.cvtColor(road2, cv.COLOR\_BGR2GRAY)  
 ret, thresh1 = cv.threshold(cmap, 50, 255, cv.THRESH\_BINARY)  
 cv.imshow("thresh1", thresh1)  
 r = 0  
 for i in thresh1[78]:  
 if i < 50: # поменял знак  
 r += 1  
 cnt = np.argmin(thresh1[78]) # поменял np.argmin на np.argmax  
 centr = cnt + r // 2  
  
 cv.line(road2, (200, 200), (200, 100), (0, 250, 0), 2)  
 cv.circle(road2, (centr, 75), 10, (0, 0, 255), -1)  
 cv.imshow("warted", road2)  
 cv.imshow("frame", frameCopy)  
  
 d = centr - 190  
  
 print(d)  
 time.sleep(0.001)  
 if cv.waitKey(1) == ord('q'):  
 break  
cap.release()  
cv.destroyAllWindows()

# **Глава 3. Работа микроконтроллера**

В качестве микроконтроллера будем использовать ArduinoUNO и программировать микроконтроллер на языке программирования C++[9].

Обмен данными между Raspberry Pi и платой Arduino выолним через последовательный интерфейс [10].

Последовательная передача данных – это просто способ их передачи. Данные отправляются последовательно, по одному биту (1 байт = 8 бит), в отличие от параллельной связи, когда одновременно отправляется много битов.

На плате Arduino Uno есть один UART, который можно использовать с USB-кабелем (Рисунок 19).

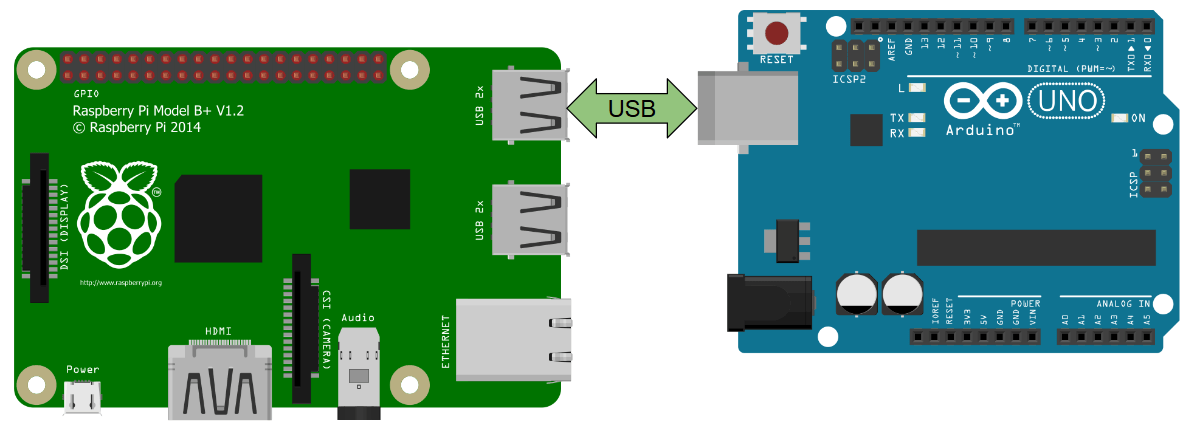


Рисунок 19 подключение микрокомпьютера и микроконтроллера

Для Arduino не требуется никаких специальных настроек. Просто убедитесь, что вы скачали и установили среду разработки Arduino.

Однако, на вашем Raspberry Pi требуется несколько вещей, чтобы интерфейс заработал. При подключении к Arduino через USB, мы обнаружили ее по адресу /dev/ttyACM0.

Установим библиотеку последовательного интерфейса Python на Raspberry Pi, в данном случае использовалась библиотека pySerial со скорость передачи 115200.

Код программы на Raspberry:

import serial  
import time  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
ser = serial.Serial('/dev/ttyACM0',115200, timeout=1/30)  
ser.flush()  
cap = cv.VideoCapture(0)  
  
  
znak\_number = 55  
R = 7  
  
k = 0.1  
  
while True:  
 ret, frame = cap.read()  
 frameCopy = frame.copy() # копируем frame в переменную frameCopy   
 cv.imshow("frameCopy", frameCopy)  
   
 ser.write(str(R).encode('utf-8'))  
 ser.write(b'%d \n' %R)  
##################### определяем знаки #########################  
   
  
 cv.imshow("znak", znak)  
 number = int.from\_bytes(ser.read(), byteorder='big')  
 if number == 18:   
 ser.write(str(znak\_number).encode('utf-8'))  
 ser.write(b'%d \n' %znak\_number)  
 ser.write(b'%d \n' %R)  
  
  
 ser.write(str(znak\_number).encode('utf-8'))  
 print(znak\_number)  
 print(R)  
 znak\_number = 0  
 time.sleep(0.01)  
  
 print(R)  
 if cv.waitKey(1)==ord('1'):  
 break  
cap.release()  
cv.destroyAllWindows()

Код программы на Arduino

#include <Multiservo.h>

#include <Wire.h>

Multiservo myservo;

int pos = 52;

int posR = 52;

const uint8\_t pwmpin = 11;

const uint8\_t enpin = 8;

const uint8\_t inApin = 9;

const uint8\_t inBpin = 10;

int speedStreet = 190;

int speedHigh = 40;

int speedWolker = 70;

unsigned long timing;

float setpoint = 52.0;

float kp = 1.0;

float ki = 0.005;

float kd = 15;

float integral = 0.0;

float prevErr = 0.0;

float dt = 0.01;

float err;

void setup()

{

Serial.begin(115200);

Serial.setTimeout(50);

pinMode(inApin, OUTPUT);

pinMode(inBpin, OUTPUT);

pinMode(pwmpin, OUTPUT);

digitalWrite(enpin, HIGH);

Wire.begin();

myservo.attach(11);

myservo.write(pos);

}

void loop()

{

if (Serial.available()>0) {

int pos = Serial.parseInt();

if (millis() - timing > 100){

timing = millis();

err = setpoint - pos;

err = map(err, -22, 23,77,27);

integral += err\*dt;

float D = (err - prevErr) / dt;

prevErr = err;

pos = err \* kp + integral \* ki + D \*kd;

posR = constrain(pos, 27, 77); // можно изменить угол 27 77

myservo.write(posR);

}

Serial.println(pos);

digitalWrite(inBpin, LOW); // включаем ехать вперед

digitalWrite(inApin, HIGH);

if (posR < 62 && posR > 42)

{

analogWrite(pwmpin, speedStreet);

}

else

{

analogWrite(pwmpin, speedWolker);

}

flushSerial();

}

}

void flushSerial(){

while (Serial.available()){

Serial.read();

}

}

# **Глава 4. Общая программа**

Программа для микрокомпьютера:

import serial  
import time  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
ser = serial.Serial('/dev/ttyACM0', 115200, timeout=1 / 100)  
ser.flush()  
cap = cv.VideoCapture(0)  
  
nerovno = cv.inRange(cv.resize(cv.imread('Nerovno.jpg'), (64, 64)), (88, 133, 46), (255, 255, 255))  
nPriori = cv.inRange(cv.resize(cv.imread('nPriori.jpg'), (64, 64)), (88, 133, 46), (255, 255, 255))  
nPrior2 = cv.inRange(cv.resize(cv.imread('nPriori2.jpg'), (64, 64)), (88, 133, 46), (255, 255, 255))  
parcking = cv.inRange(cv.resize(cv.imread('parcking2.jpg'), (64, 64)), (88, 70, 70), (255, 255, 255))  
stop = cv.inRange(cv.resize(cv.imread('stop.jpg'), (64, 64)), (88, 133, 46), (255, 255, 255))  
  
znak\_number = 0  
R = 0  
k = 0.09  
while True:  
 ret, frame = cap.read()  
 frameCopy = frame.copy() # копируем frame в переменную frameCopy   
 cv.imshow("frameCopy", frameCopy)  
 road = frameCopy[280:480, 120:520]  
 road2 = road.copy()  
 cv.imshow("road2", road2)  
 cmap = cv.cvtColor(road, cv.COLOR\_BGR2GRAY)  
 ret, thresh1 = cv.threshold(cmap, 50, 255, cv.THRESH\_BINARY)  
 cv.imshow("thresh1", thresh1)  
 r = 0  
 for i in thresh1[190]:  
 if i < 50: # поменял знак  
 r += 1  
 cnt = np.argmin(thresh1[190]) # поменял np.argmin на np.argmax   
 centr = cnt + r // 2  
  
 # cv.line(road2, (200, 200), (200,150), (0, 250, 0), 2)  
 cv.circle(road2, (centr, 195), 10, (0, 0, 255), -1)  
 cv.imshow("warted", road2)  
  
 d = centr - 200  
 y = d \* k  
 L = (52 + y) // 1  
 # ser.write(str(R).encode('utf-8'))  
  
 # ser.write(b'%d \n' %R)  
 ##################### определяем знаки #########################  
 znak = frameCopy[40: 250, 400: 635]  
 cv.imshow('znak', znak) # закоментируй это для работы  
 # frameCopy = frame.copy()  
 znakGRAY = cv.cvtColor(znak, cv.COLOR\_BGR2GRAY)  
  
 Edge = cv.Canny(znakGRAY, 100, 255)  
 cv.imshow('Edge', Edge)  
 contours, hierarchy = cv.findContours(Edge, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)  
  
 if contours == 0:  
 continue  
 for cnt in contours:  
 epsilon = 0.01 \* cv.arcLength(cnt, True)  
 approx = cv.approxPolyDP(cnt, epsilon, True)  
 area = math.fabs(cv.contourArea(cnt)) # вычисляем площадь контура  
  
 if area < 5000 or area > 15000:  
 continue  
 x, y, w, h = cv.boundingRect(cnt) # получаем координаты верхнего угла, ширину и высоту  
 sideRatio = w / h # получаем соотноение сторон  
 if sideRatio < 0.4 or sideRatio > 1.2:  
 continue  
 if sideRatio > 0.4 and sideRatio < 0.6:  
 cv.rectangle(znak, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 detectZnak = cv.resize(znak[y: (y + h), x: (x + w)], (60, 120))  
 # cv.imshow("detectZnak", detectZnak) # чтобы видеть знак раскоментируй  
  
 cutedFrame = detectZnak[20:101, 12:48]  
  
 v = cv.cvtColor(cutedFrame, cv.COLOR\_BGR2HSV)[:, :, 2]  
 # cv.imshow("v", v)  
  
 red\_sum = np.sum(v[0:27, 0:36])  
 yellow\_sum = np.sum(v[28:54, 0:36])  
 green\_sum = np.sum(v[55:81, 0:36])  
  
 cv.rectangle(cutedFrame, (0, 0), (36, 27), (0, 0, 255), 2)  
 cv.rectangle(cutedFrame, (0, 28), (36, 54), (0, 0, 255), 2)  
 cv.rectangle(cutedFrame, (0, 55), (36, 81), (0, 0, 255), 2)  
  
 if green\_sum > yellow\_sum and green\_sum > red\_sum:  
 R = 5  
 # print("green")  
 elif yellow\_sum > green\_sum and yellow\_sum > red\_sum:  
 R = 6  
 # print("yellow")  
 elif red\_sum > green\_sum and red\_sum > yellow\_sum:  
 R = 7  
 print("red")  
 else:  
 print("red")  
 R = 7  
 else:  
 cv.rectangle(znak, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 detectZnak = znak[y: (y + h), x: (x + w)]  
  
 detectZnak = cv.inRange(cv.resize(detectZnak, (64, 64)), (88, 100, 46), (255, 255, 255))  
  
 stop\_val = 0  
 nerovno\_val = 0  
 nPriori\_val = 0  
 nPrior2\_val = 0  
  
 for i in range(64):  
 for j in range(64):  
 if detectZnak[i][j] == stop[i][j]: stop\_val += 1  
 if detectZnak[i][j] == nerovno[i][j]: nerovno\_val += 1  
 if detectZnak[i][j] == nPriori[i][j]: nPriori\_val += 1  
 if detectZnak[i][j] == nPrior2[i][j]: nPrior2\_val += 1  
 # if detectZnak[i][j] == priorit[i][j]: priorit\_val +=1  
  
 if stop\_val > 3000:  
 R = 1  
 elif nerovno\_val > 3000:  
 R = 2  
 elif nPriori\_val > 3000:  
 R = 3  
 elif nPrior2\_val > 3000:  
 R = 3  
 # elif priorit\_val > 3000:  
 # R = 4  
 else:  
 R = 0  
  
 # cv.imshow("znak", znak)  
 # number = int.from\_bytes(ser.read(), byteorder='big')  
 # if number == 18:   
 # # ser.write(str(znak\_number).encode('utf-8'))  
 # ser.write(b'%d \n' %znak\_number)  
 ser.write(b'%d \n' % R)  
 ser.write(b'%d \n' % L)  
 print(R)  
 print(L)  
  
 if cv.waitKey(1) == ord('1'):  
 break  
cap.release()  
cv.destroyAllWindows()

программа для микроконтроллера:

#include <SoftwareSerial.h>

#include <Wire.h>

#include <Multiservo.h>

Multiservo myservo;

int pos = 90;

int posR = 90;

// motor driver CJNMCU

// схема подключение мотора

// https://www.xsimulator.net/community/attachments/motomonstersetup-jpg.27135/

const uint8\_t EN = 8;

const uint8\_t INA = 9;

const uint8\_t INB = 10;

const uint8\_t PWM = 11;

unsigned long timing = 0;

// скорость мотора

uint8\_t motorSpeed = 55;

uint8\_t motorSpeedBack = 60;

uint8\_t motorStop = 0;

uint8\_t motorSpeedHigh = 80;

uint8\_t motorSpeedLow = 40;

uint8\_t znakNumber = 0;

uint8\_t znak = 0;

uint8\_t NPr = 0;

uint8\_t sc = 0;

uint8\_t zp = 0;

uint8\_t start = 0;

void setup() {

Serial.begin(115200);

pinMode(EN, OUTPUT);

pinMode(INA, OUTPUT);

pinMode(INB, OUTPUT);

pinMode(PWM, OUTPUT);

digitalWrite(EN, HIGH);

Wire.begin();

myservo.attach(11);

myservo.write(pos);

delay(6000);

}

void loop() {

if (Serial.available() > 0)

{

znakNumber = Serial.parseInt();

comeZnak();

if (start == 0) {

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

}

if (start == 1 || znakNumber == 5) {

comeZnak();

}

}

}

/////////////////////////////////

void znakNerovno() {

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeedLow);

myservo.write(85);

delay(1000);

while (millis() - timing > 2000) {

timing = millis();

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeedLow);

znak = 2;

}

}

//////////////////////////////////

void znakPriorit() {

if (zp == 0) {

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeedHigh);

delay (1000);

zp = 1;

}

if (zp == 1) {

znak = 0;

}

}

//////////////////////////////////

void forvard() {

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

}

//////////////////////////////////

void stopCar() {

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

delay (2000);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (700);

}

//////////////////////////////////

void back() {

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

delay (500);

digitalWrite(INA, LOW);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorSpeedBack);

delay (500);

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

delay (2000);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (2000);

}

//////////////////////////////////

void parking() {

myservo.write(pos);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (500);

myservo.write(102);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (200);

myservo.write(87);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (500);

myservo.write(77);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (200);

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

delay (5000);

znakNumber = 0;

myservo.write(48);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeedLow);

delay (500);

myservo.write(45);

digitalWrite(INA, HIGH);

digitalWrite(INB, LOW);

analogWrite(PWM, motorSpeed);

delay (1000);

}

//////////////////////////////////

void comeZnak() {

if (Serial.available() > 0)

{

znakNumber = Serial.parseInt();

if (znakNumber < 10) {

switch (znakNumber) {

case 0:

znak = 0;

forvard();

flushSerial();

break;

case 1:

znak = 1;

stopCar();

flushSerial();

break;

case 2:

znak = 2;

znakNerovno();

flushSerial();

break;

case 3:

znak = 3;

back();

flushSerial();

break;

case 4:

znak = 4;

parking();

flushSerial();

break;

case 5:

znak = 5;

start = 1;

forvard();

flushSerial();

break;

case 6:

znak = 6;

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

break;

flushSerial();

case 7:

znak = 7;

digitalWrite(INA, HIGH);

digitalWrite(INB, HIGH);

analogWrite(PWM, motorStop);

break;

}

}

if (znakNumber > 10) {

posR = constrain(znakNumber, 65, 115); // можно изменить угол 22 82

myservo.write(posR);

}

}

}

void flushSerial() {

while (Serial.available()) {

Serial.read();

}

}

# **Глава 5. Заключение**

В итоге всей работы был собран, запрограммирован и апробирован прототип беспилотного автомобиля. При работе с прототипом были получены навыки программирования в Python, установка дополнительных библиотек, работа с файлами, изображениями, потоковым видео, обработка изображений, написание документации, работа с github, передача данных из одного устройства на другое. Прототип был успешно опробован на соревнованиях Роботраффик (Рисунок 19), ссылка на видео https://github.com/uokansk/GB\_paradigm/blob/master/diplom/robotraffic.mp4

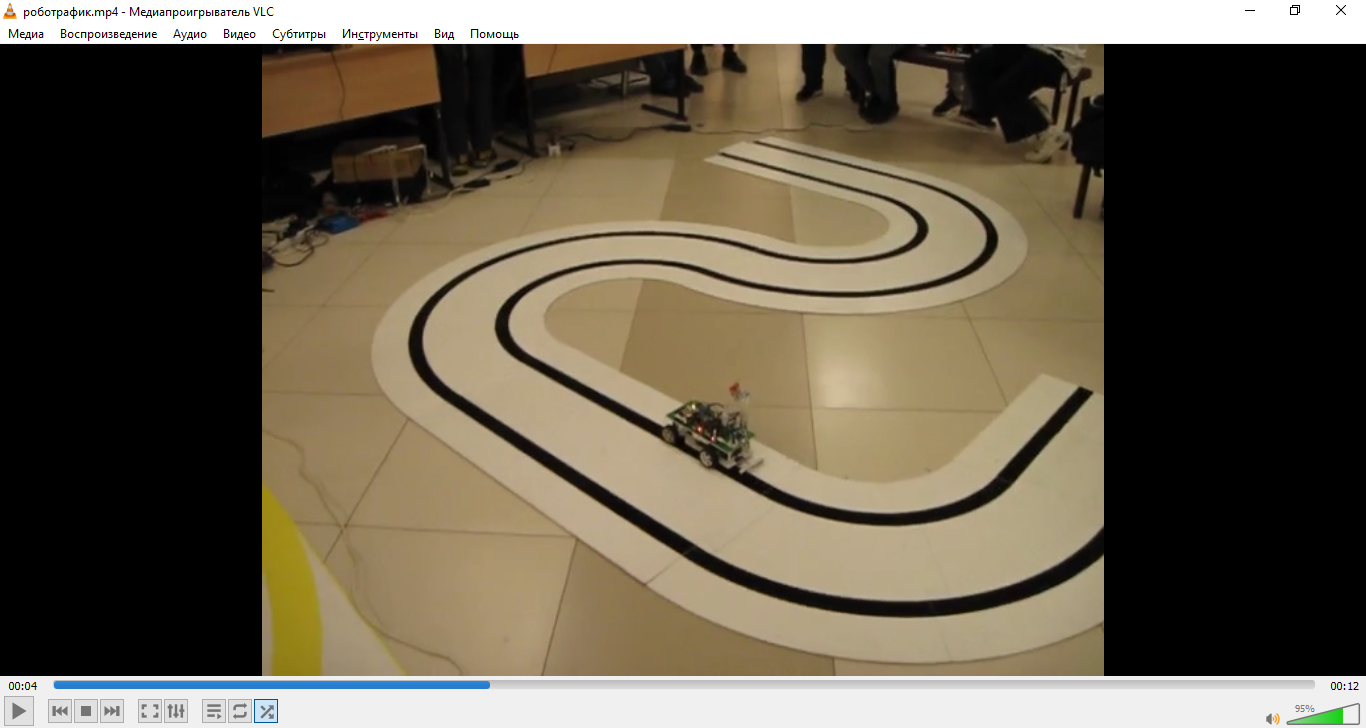


Рисунок 20 Испытание прототипа

В данном виде использовать программу для настоящих беспилотных автомобилей нельзя, но приемы и методы полученные в ходе создания данного проекта вполне применимы для разработки беспилотного транспорта.

# **Список литературы**

1. Книга: Шакирьянов Э.Д. КОМПЬЮТЕРНОЕ ЗРЕНИЕ НА PYTHON М. : Лаборатория знаний, 2021.-160 с.
2. Книга: Ashwin Pajankar. Raspberry Pi Computer Vision Programming. First published: May 2015 Birmingham B3 2PB, UK.
3. Официальный сайт Министерства Внутренних Дел [Электронный ресурс]. URL: <http://stat.gibdd.ru/> .
4. Официальный сайт Автошколы при академии президента [Электронный ресурс]. URL: <https://avtoshkola-drive.ru/content/7-osnovnyk-hprichin-DTP/> .
5. Сайт ПДД мастер [Электронный ресурс]. URL: [https://pddmaster.ru/rabota/rezhim-truda.html#](https://pddmaster.ru/rabota/rezhim-truda.html) .
6. Сайт Википедия [Электронный ресурс]. [https://ru.wikipedia.org/wiki/беспилотный\_автомобиль](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BF%D0%B8%D0%BB%D0%BE%D1%82%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C)
7. ХАБР – сообщество IT специалистов [Электронный ресурс]. URL: <https://habr.com/ru/post/431758/>
8. Сайт. OpenCV Tutorials. URL: <https://docs.opencv.org/4.x/d9/df8/tutorial_root.html>
9. Сайт. Аппаратная платформа ардуино URL: <https://arduino.ru/>
10. Сайт. Интернет магазина Вольтик. Подключение Raspberry Pi к Arduino через UART интерфейс. URL: <https://voltiq.ru/raspberry-pi-arduino-serial-communication>