МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименования института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Промышленная электроника
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Система распознавания дорожных знаков»

Студент Н.А. Краснопевцева
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., Е.С.Глибин
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент, А.В. Кириллова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра Промышленная электроника
(наименование)

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент *Краснопевцева Наталья Александровна, ЭЛб-1601а*

1. Тема *Система распознавания дорожных знаков*

2. Срок сдачи студентом законченной бакалаврской работы *«14» июня 2020 г.*

3. Исходные данные к выпускной квалификационной работе

Обозреваемая и разрабатываемая система распознавания дорожных знаков должна отвечать всем стандартам безопасности и критериям современных автоматизированных систем управления, а также строиться на известных промышленных решениях в сфере автоматики.

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов, разделов)

Аннотация
Введение
1 Актуальность и обзор аналогов
2 Выбор комплектующих
2.1 Выбор одноплатного компьютера
2.2 Выбор камеры
3 Разработка структурной схемы
4 Разработка электрической схемы
4.1 Подключение вентилятора
4.2 Подключение камеры
4.3 Подключение Raspberry Pi к Arduino UNO
5 Выбор инструментов
5.1 Язык программирования Python
5.2 Библиотека компьютерного зрения OpenCV
5.3 Библиотека NumPy
6 Алгоритм распознавания образов
6.1 Цветовое пространство HSV
6.2 Операции математической морфологии
6.3 Выделение контуров
6.4 ROI
7 Программная часть
7.1 Блок-схема алгоритма
8 Результаты экспериментальных испытаний
8. 1 Влияние освещения на качество распознавания дорожных знаков
9 Безопасность и экологичность проекта
10 Экономический расчет
Заключение

Список используемой литературы

5. Ориентировочный перечень графического и иллюстративного материала

1. Структурная схема устройства

2. Блок-схема алгоритма передачи данных

3. Блок-схема алгоритма приема данных

4. Схема электрическая соединений

5. Блок-схема алгоритма программы

6. Экономический расчет

6. Консультант по разделам А.В. Кириллова

7. Дата выдачи задания «28» февраля 2020 г.

Руководитель бакалаврской работы _____ Е.С. Глибин

(подпись) (И.О. Фамилия)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра Промышленная электроника
(наименование)

КАЛЕНДАРНЫЙ ПЛАН

выполнения бакалаврской работы

Студент Н.А.Краснопевцева
по теме Система распознавания дорожных знаков

Наименование работ	Плановый срок выполнения	Фактический срок выполнения	Отметка о выполнении	Подпись руководителя / консультатнта
Анализ актуальности проекта, написание введения и обзорной главы	21.02.2020	19.02.2020		
Разработка структурной схемы разрабатываемого устройства.	13.03.2020	13.03.2020		
Разработка схемы электрической соединений	24.04.2020	24.04.2020		
Разработка алгоритмов программы. Написание остальных разделов работы	15.05.2020	12.05.2020		
Оформление пояснительной записки и плакатов	10.06.2020	10.06.2020		

Руководитель бакалаврской работы _____ Е.С.Глибин
(подпись) (И.О. Фамилия)

Аннотация

Выпускная квалификационная работа состоит из введения, десяти глав, 2 таблиц, 35 рисунков, заключения, списка литературы, включая зарубежные источники. Определены актуальность темы, цель работы, в заключении сделаны выводы о проделанной работе. Объем выпускной работы составляет 62 страницы.

В работе создана система распознавания дорожных знаков, основанная на методе поиска объекта по цвету через цветовое пространство HSV.

Целью работы является достижение приемлемого распознавания дорожных знаков, а также внедрение данной системы в систему поиска пути мобильного робота в замкнутом пространстве. Таким образом, готовый комплексный проект представляет собой движение мобильного робота по пути в соответствии с дорожными знаками.

На сегодняшний день задача построения систем автоматического управления робототехнических платформ в складской логистике набирает популярность среди IT разработчиков. Их цель – создание недорогих и доступных широкому кругу потребителей систем автоматизированного распознавания знаков.

Предложена структурная схема системы распознавания знаков. Обоснован выбор конструктивных элементов системы. Разработана блок-схема программы. Произведена оценка безопасности и экологичности проекта. Выполнен экономический расчет. Сделан вывод о проделанной работе.

Abstract

The title of the graduation work is «Traffic sign recognition system».

The senior paper consists of an introduction, nine parts, a conclusion, tables, list of references including foreign sources, diagrams and pictures.

The key issue of the thesis the creation of low-cost and accessible to a wide range of consumers systems for automated recognition of road signs. We need to choose the appropriate method.

The aim of this work is to develop a road sign recognition system and implement it in the mobile robot path search system.

The graduation work may be divided into several logically connected parts which are: market review and analysis of analogues; selection of components; development of a structural diagram; development of an electrical circuit; work with the computer vision library OpenCV; software part; experimental test results.

The program code for sign recognition has written in the Python programming language using computer vision technologies of The OpenCV open source library and is a method for recognizing objects by color through the HSV color space.

Finally, we present the results of experimental tests, which clearly show that the program code for the traffic sign recognition system is working properly.

In conclusion we'd like to stress this work is relevant in to solve the problem of implementing a sign recognition system in warehouse logistics, but it can also become the basis for developing a sign recognition system for cars.

Содержание

Введение

Компьютеры справляются со многими задачами гораздо лучше, чем человек. К примеру, их преимущество в работе с числами неоспоримо. Однако такая простая задача, как обнаружение на картинке привычных для нас объектов – домов или деревьев, может поставить машину в тупик. Это объясняется тем, что человек учится распознавать объекты всю жизнь и основывается на воспоминаниях о том или ином свойстве предмета. Машины же в свою очередь создавались для того, чтобы работать с числами.

Необходимость наделить машины зрением возникла относительно недавно. На сегодняшний день роботизированные системы успешно справляются с задачами, связанными со способностью «видеть» и интерпретировать увиденное. Они умеют считывать штрихкоды на товарах в супермаркете, распознавать номерные знаки автомобилей, анализировать записи с камер наблюдения и даже находить лица людей на фото. Таким образом, под компьютерным зрением понимается набор методов, позволяющих компьютерам видеть и извлекать информацию из изображений [1].

В настоящее время развитие робототехники во многом зависит от успеха в области компьютерного зрения. Сферы его применения довольно обширны: от промышленных средств наблюдения и мониторинга до автономных систем, принимающих решения на основе анализа полученной видеоинформации. Компьютерное зрение так же служит средством преобразования реальных трехмерных объектов в двумерные или одномерные. В результате такой операции компьютер может восстановить информацию об окружающем мире. Распознавание представляет собой процесс, который присваивает некоторому объекту идентификатор на основании его описателей (например, «здание» или «машина») [2].

В мире искусственного интеллекта и совершенствования технологий многие исследователи и крупные компании, такие как Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi и т. д., работают над автономными транспортными средствами и автомобилями с автоматическим управлением. На сегодняшний день проблемы безопасности дорожного движения в значительной степени вызваны субъективными причинами, связанными с водителем: невнимательностью, неправильным управлением автомобилем и несоблюдением правил дорожного движения. Разработка умных систем для автомобилей стала бы эффективным средством устранения этих человеческих факторов.

Следующей сферой применения системы распознавания знаков является складская логистика, а именно внутрискладское перемещение груза. Эффективное функционирование складов в системе логистики, независимо от их назначения и вида деятельности, возможно лишь при решении проблем, с которыми сталкиваются при создании складского хозяйства и рационализации действующих складов. К таким проблемам можно отнести разработку системы складирования [3].

Для достижения точности в этой технологии, транспортные средства и складские грузоперевозчики должны иметь возможность корректно интерпретировать визуальные данные и по полученной информации принимать соответствующие решения.

В рамках выпускной квалификационной работы предполагается разработка программного и аппаратного обеспечения для распознавания дорожных знаков, встроенная в систему поиска пути мобильного робота.

1 Актуальность и обзор аналогов

Распознавание дорожных знаков является одной из составляющих, необходимых для беспилотных автомобилей и автоматизирования складской перевозки грузов.

Машина должна самостоятельно определять разметку, ограничения, знаки и условия движения. Поскольку причиной большинства аварий считается нарушение скоростного режима, инженеры автомобильных компаний задались целью искоренить данную проблему. Для этого в машину устанавливается система распознавания знаков. К ее основным функциям относятся:

1. Определение и подтверждение информации о дорожных знаках.

2. Поиск информации в базе данных и уведомление водителя.

3. Предупреждение с помощью светового или звукового сигнала, если скорость движения не изменяется.

Возможности систем зависят от поколений разработки. Первоначальные решения могли распознавать только ограничители скорости, запреты на обгон и некоторые дополнительные знаки. Современные системы могут расшифровывать информацию о жилых зонах, начале и конце населенного пункта, конце зоны ограничений, въезд запрещен и многое другое. Системы находятся на начальном этапе развития, что сказывается на точности обнаружения знаков и их расшифровке. При всем этом, стоит отметить стоимость подобной системы. Помимо высокой цены автомобиля таких крупных компаний как Tesla, Uber, Google, Mercedes-Benz, Toyota, Audi и т.д., наличие системы распознавания дорожных знаков увеличивает его стоимость как минимум на 15-20 процентов.

Многие IT-разработчики, не связанные с крупными автомобильными компаниями, занимаются созданием недорогих и доступных широкому кругу потребителей подобных систем.

В сфере складской логистики крупнейшая торговая площадка в мире Amazon, известная своими роботизированными складами, начинает внедрять технологии компьютерного зрения в своих центрах [4].

До сих пор работники центра выполнения вручную проверяли штрих-код на каждом товаре, который находится в помещении, позволяя отслеживать этот пакет при его перемещении по складу. Таким образом роботы узнавали, какие именно товары им необходимо забрать и куда их перевести. Однако времена ручных сканеров прошли, и сейчас Amazon внедряет в свои центры камеры и сканеры с искусственным интеллектом, которые самостоятельно могут считывать штрих-код каждого продукта и необходимость в участии человека в этом процессе исчезла. Такие изменения значительно повысили эффективность складов.

Дочерняя компания Amazon «Kiva System» занимается разработкой высокотехнологичной складской робототехники. Одна из их разработок – мобильные роботы, передвигающиеся при помощи колес. Номер заказа вводится в базу данных, после чего программа находит ближайший транспортный робот и направляет его к модулю хранения с помощью штрих-кодов, располагающихся на полу склада [5]. Фотография роботов Amazon представлена на рисунке 1.1.

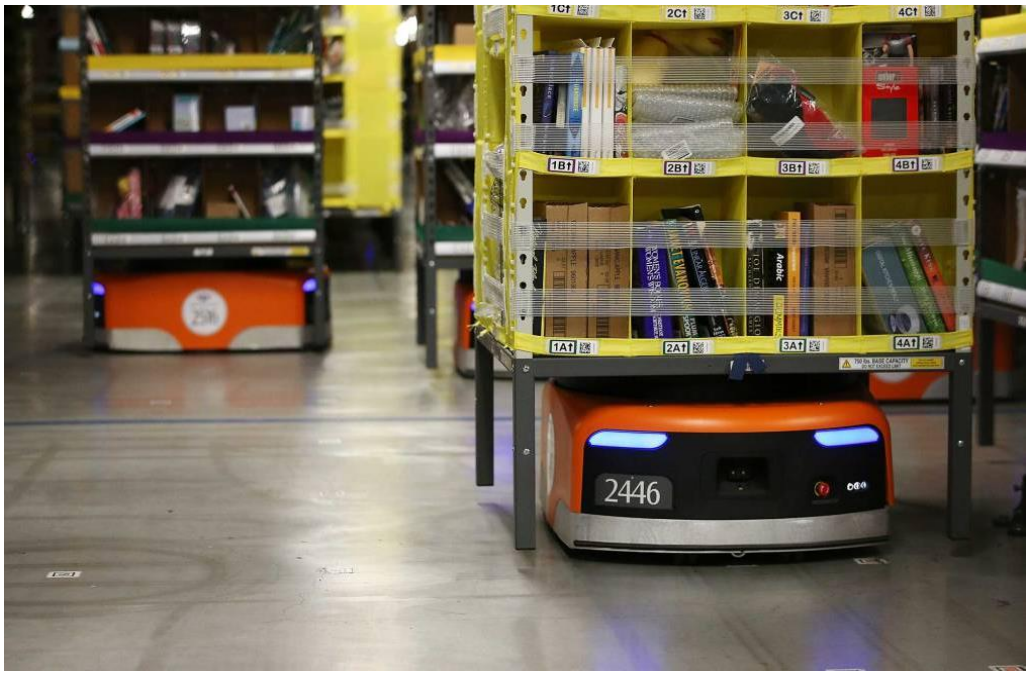

Рисунок 1.1 – Роботы Amazon

Однако такая технология доступна только крупным компаниям, финансово способным организовать подобную систему. Во всех других случаях необходима разработка менее дорогостоящих аналогов. К ним можно отнести систему распознавания дорожных знаков, установленных в различных частях склада для создания определенного маршрута робота-грузоперевозчика. Внедрение данной технологии распознавания знаков в систему складирования не только повысит эффективность работы склада в целом, но и уменьшит затраты, связанные с выплатами за перевоз груза работникам склада.

Таким образом, в данном разделе определена актуальность разработки подобных систем. На данный момент поиском решения автономных складских роботизированных платформ занимаются инженеры со всего мира. Внедрение элементов компьютерного зрения в данную систему позволяет увеличить ее функционал.

2 Выбор комплектующих

В качестве базы для аппаратного решения подобной системы повсеместно используются одноплатные компьютеры. Их преимущество заключается в компактности и, как следствие ее, мобильности. Также использование подобных компьютеров облегчается возможностью установки большого количества операционных систем на выбор пользователя. К компьютеру подключается камера, с которой происходит захват изображения. Компьютер получает информацию с камеры, обрабатывает ее и передает результат обработки на микроконтроллер, к которому подключен двигатель. Устройство и программное обеспечение мобильного робота является частью комплексной выпускной квалификационной работы, которая не описывается в данной работе.

В первую очередь для создания системы распознавания дорожных знаков необходимо выбрать подходящую модель одноплатного компьютера.

2.1 Выбор одноплатного компьютера

Самой популярной серией одноплатных компьютеров по праву является Raspberry Pi. Для решения поставленных задач, был сделан выбор в пользу одноплатного компьютера Raspberry Pi 2 Model B. Внешний вид данной модели представлен на рисунке 2.1.1.

Рисунке 2.1.1 – Raspberry Pi 2 Model B

В отличие от предыдущих моделей, Raspberry Pi второго поколения не использует относительно отсталый процессор BroadcomBCM2835 (одноядерный ARM11), а переходит на 4-ядерный процессор BroadcomBCM2836 с архитектурой ARMv7Cortex-A7. С обновлением процессора его тактовая частота также увеличилась с 700 МГц до 900 МГц. Объем памяти также удвоился, с 512 МБ до 1 ГБ. Согласно официальному заявлению, производительность оборудования нового поколения увеличилась в 6 раз, и с точки зрения улучшения производительности процессора это действительно так. Как и первая модель, второе поколение Raspberry Pi использует порты Ethernet 100M и USB 2.0 [6].

Для решения поставленных задач не рассматриваются более новые модели Raspberry Pi, поскольку в работе не используются сложные алгоритмы распознавания, например, распознавание лиц, а для распознавания знаков достаточно вычислительной мощности данной модели.

Raspberry Pi 2 Model B является оптимальным вариантом, поскольку имеет хорошую производительность, возможность установки модуля камеры, достаточное количество портов и расширений и тем самым выигрывает в соотношении цена/качество.

2.2 Выбор камеры

При разработке системы распознавания дорожных знаков важную роль играет камера, с которой будет осуществляться захват изображения. Обычные внешние камеры, подключающиеся через USB порты – не лучший вариант для работы с Raspberry Pi, поскольку они нагружают процессор и занимают дефицитные USB-порты одноплатного компьютера. Поэтому целесообразно выбрать специальный компактный камерный модуль.

К самым популярным моделям камер для Raspberry Pi относят Raspberry Pi Compatible Fisheye Camera, Raspberry Pi Camera V2 и Raspberry Pi Camera v2 NoIR [7]. Необходимо рассмотреть каждый из вариантов подробнее.

Raspberry Pi Compatible Fisheye Camera - это совместимая с Raspberry Pi камера типа «рыбий глаз», которую легко можно найти на различных торговых интернет-площадках: AliExpress, TaoBao, eBay и т.д. Она характеризуется широкоугольным обзором 175º. В ее основе лежит датчик Omnivision 5647 разрешением 5 мегапикселей (2592 x 1944 пикселей). Камера изображена на рисунке 2.2.1.

Рисунок 2.2.1 – Камера Raspberry Pi Compatible Fisheye Camera

Raspberry Pi Camera V2 – камера, оснащенная сенсором Sony IMX219 Exmor. Он позволяет захватывать, записывать и транслировать видео в форматах 1080p, 720p и VGA. Максимальное разрешение для фотографий достигает 3280×2464 пикселей. Камера изображена на рисунке 2.2.2.

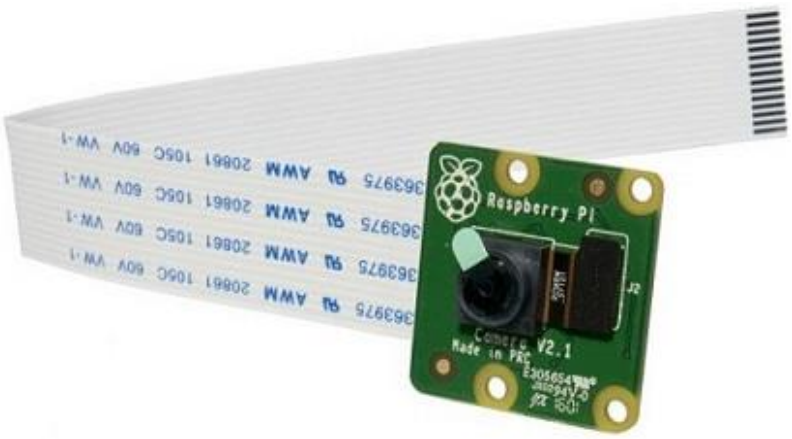

Рисунок 2.2.2 – Камера Raspberry Pi Camera V2

Raspberry Pi Camera V2 NoIR (рисунок 2.2.3) оснащена 8-мегапиксельным сенсором Sony IMX219. Характеристики камеры совпадают с моделью Raspberry Pi Camera V2, однако у данной модели есть одна отличительная способность: в ней не используется инфракрасный фильтр Это означает, что снимки, сделанные при дневном свете, будут выглядеть не так, как в жизни. Благодаря инфракрасному освещению эта модель способна видеть в темноте.

Рисунок 2.2.3 – Камера Raspberry Pi Camera V2 NoIR

Для наглядности представленных характеристик приведена таблица сравнений предложенных камер (таблица 2.2.1).

Таблица 2.2.1 – Характеристики модулей камеры для Raspberry Pi

	Модель сенсора	Максимальное разрешение	Разрешение видео	Угол обзора	Ночное видение
Raspberry Pi Compatible Fisheye Camera	Omnivision 5647	5 Мп (2592 x 1944)	1080p	175º	-
Камера Raspberry Pi Camera V2	Sony IMX219	8 Мп (3280×2464)	1080p	66°	-
Камера Raspberry Pi Camera NoIR	Sony IMX219	8 Мп (3280×2464)	1080p	66°	+

Проанализировав модели, был сделан выбор в пользу модуля камеры Raspberry Pi camera V2, поскольку он имеет отличные характеристики в плане качества изображения, а отсутствие широкого угла обзора и инфракрасного фильтра не является проблемой, поскольку захват изображения будет производиться со знаков, расположенных перед камерой, и при хорошем освещении.

Таким образом, в данном разделе был проведен подробный разбор нескольких типов камеры и моделей одноплатного компьютера. Была выбрана камера, обладающая достаточно хорошим качеством съёмки. Отмечается целесообразность использования менее дорогой версии компьютера, так как распознавание дорожных знаков не требует высокой вычислительной мощности.

3 Разработка структурной схемы

При разработке схемотехнической части электронного устройства в первую очередь необходимо составить структурную схему.

На рисунке 3.1 представлена структурная схема разрабатываемого устройства.

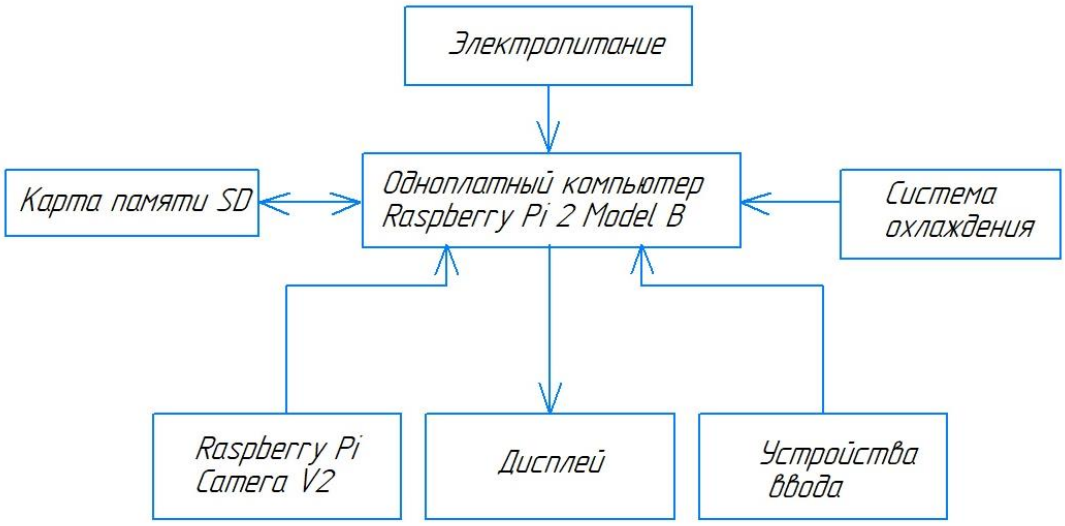

Рисунок 3.1 – Структурная схема разрабатываемого устройства

Как видно из рисунка, в разрабатываемое устройство включены:

– одноплатный компьютер Raspberry Pi 2 Model B;

– модуль камеры Raspberry Pi Camera V2;

– вентилятор, осуществляющий охлаждение быстро нагревающегося компьютера;

– карта SD;

– дисплей;

– устройства ввода (клавиатура и компьютерная мышь);

– источник электропитания с номинальным напряжением 5V и минимальным током 1,8 А.

Одноплатный компьютер Raspberry Pi 2 Model B будет осуществлять захват с камеры, преобразовывать полученную информацию и передавать ее на микроконтроллер Arduino UNO, связанному с ним последовательной связью по USB кабелю.

При отсутствии сигнала с Raspberry Pi мобильный робот будет двигаться согласно заранее заданному маршруту. В зависимости от полученного с Raspberry Pi сигнала, считанного с дорожного знака, мобильный робот будет двигаться в предписанном знаками направлении.

Таким образом, в данном разделе была разработана структурная схема устройства распознавания дорожных знаков.

4 Разработка электрической схемы

Перед подключением внешних устройств к одноплатному компьютеру, следует сначала разобраться в назначении основных контактов (пинов). Raspberry Pi 2 Model B имеет распиновку, изображенную на рисунке 4.1.

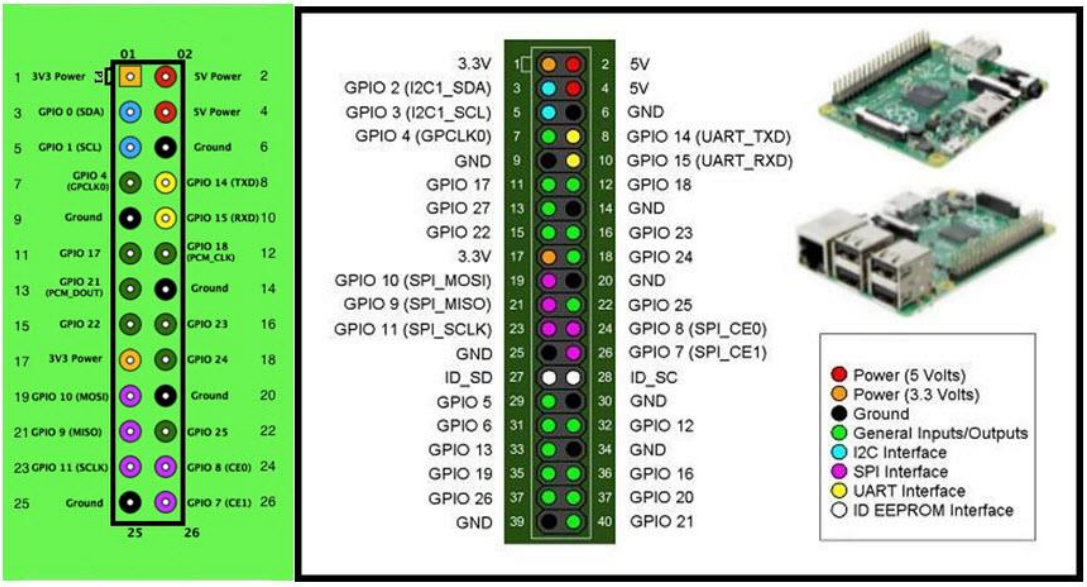

Рисунок 4.1 – Распиновка Rasberry Pi

У Raspberry Pi распиновка включает в себя два ряда штырьков. Совокупное количество же пинов равняется 40 [8].

Первый (располагается слева) предназначен для работы с устройствами, для которых требуется напряжение 3,3 Вольт. Второй (располагается справа) – для работы с устройствами, для которых требуется напряжение 5 Вольт.

Следующее, что нужно знать о распиновке GPIO Raspberry Pi – назначение всех штырьков. Всего существует три типа пинов:

- питающие (при включении подают электричество);
- порты (выводящие и принимающие информацию);

- заземляющие.

Питающие пины обозначены на плате словом «Power» (1,2,4,17 пины). Заземляющие – словом «Ground» (6, 9,14,20,25,30,34,39 пины). Выводящие и принимающие информацию порты обозначаются словом «BCM» и занимают все оставшиеся пины на плате.

Стоит отметить, что нумерация в Rasberry Pi выполняется по горизонтали. То есть: 1 – 3,3V, 2 – 5V, 3 – порт, 4 – 5V, 5 – порт, 6 – заземление, 7 – порт, 8 – первый порт для 5-вольтных устройств и т.д.

4.1 Подключение вентилятора

Подключение вентилятора к компьютеру считается обязательным шагом перед работой с Raspberry Pi. Это связано с тем, что одноплатные компьютеры довольно быстро нагреваются. При высоких температурах подобные устройства нередко выходят из строя и не подлежат дальнейшему их использованию. Схема подключения вентилятора к Raspberry Pi представлена на рисунке 4.1.1.

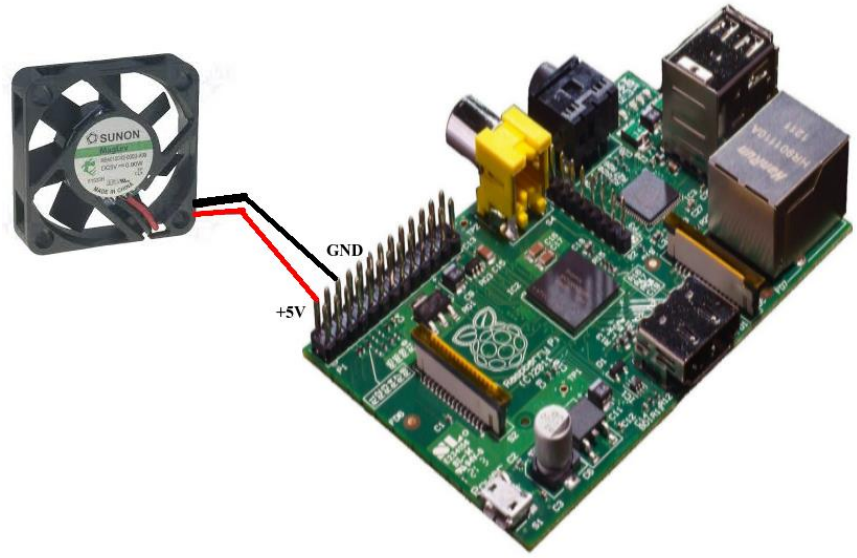

Рисунок 4.1.1 - Подключение вентилятора к Raspberry Pi

Как видно из рисунка, вентилятор располагается над радиатором и имеет только 2 провода: питание и заземление.

4.2 Подключение камеры

Модуль камеры Raspberry Pi имеет шлейф, который подключается в CSI-разъем одноплатного компьютера. Схема подключения модуля камеры к Raspberry Pi изображена на рисунке 4.2.1.

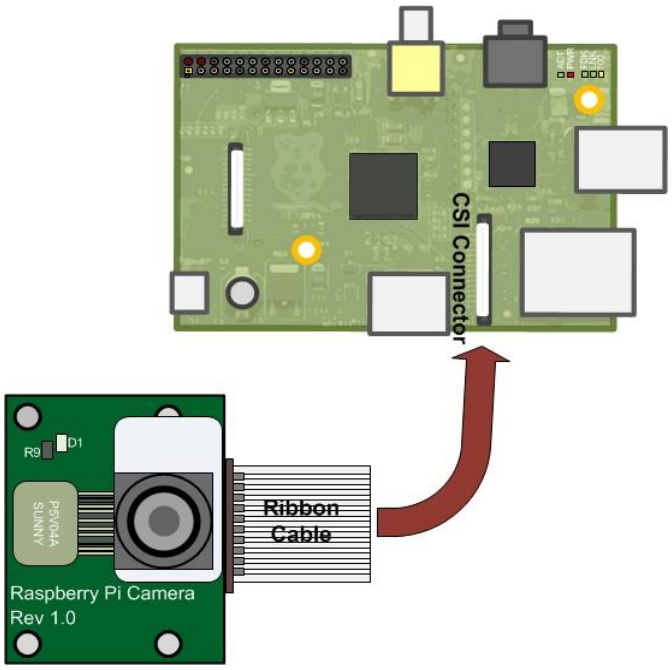

Рисунок 4.2.1 - Схема подключения модуля камеры к Raspberry

После этого необходимо программно обеспечить подключение камеры. Необходимо открыть утилиту конфигурирования и найти в ней вкладку «Interfaces». Оказавшись на ней, потребуется найти переключатель Camera, поставить его в положение «Enabled» и перезагрузить компьютер.

4.3 Подключение Raspberry Pi к Arduino UNO

Подключение Raspberry Pi к Arduino UNO является важным шагом при выполнении данной комплексной работы. Оно осуществляется за счет последовательной связи по USB-кабелю и загрузки двух программ: для Raspberry Pi и Arduino UNO.

Для работы с Raspberry Pi в первую очередь важно открыть все необходимые порты. На рисунке 4.3.1 изображена утилита конфигурирования, на которой стрелками указаны порты, которые необходимо перевести в положение «Enabled».

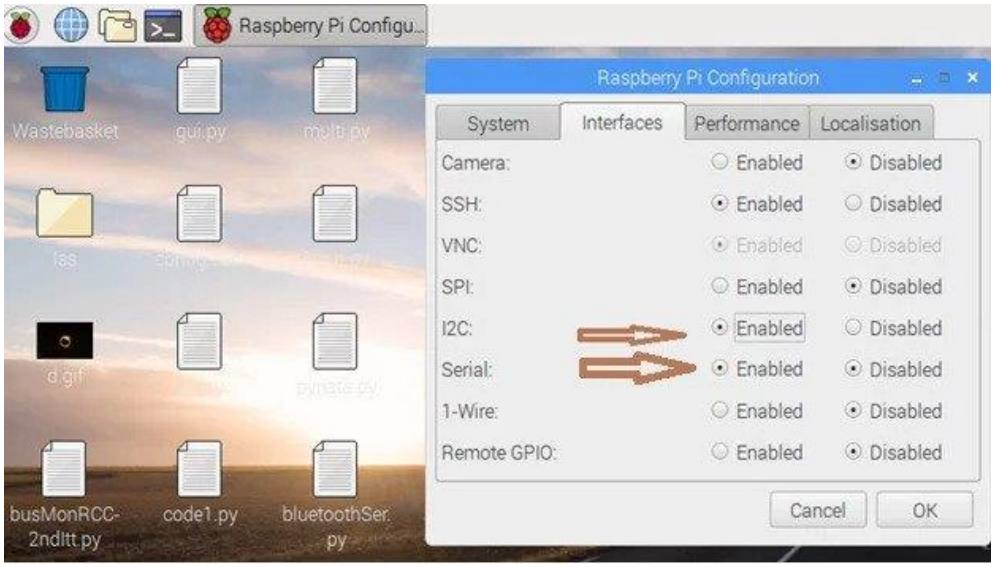

Рисунок 4.3.1 – Утилита конфигурирования Raspberry Pi

Программа, которую необходимо загрузить в Raspberry Pi выглядит следующим образом:

```
import serial
import RPi.GPIO as GPIO
import time
```

```
ser=serial.Serial("/dev/ttyUSB1",9600)
ser.baudrate=9600

while True:

   ser.write('r'.encode())
   print('r')
```

Разберем более подробно данную программу. В первую очередь необходимо подключить несколько важных модулей:

– модуль pyserial, предназначенный для работы с последовательным портом;

– модуль RPi.GPIO as GPIO для управления каналами GPIO на Raspberry Pi;

– модуль time для работы со временем.

Необходимо определить, к какому последовательному порту подключен микроконтроллер Arduino, и настроить скорость передачи данных:

```
ser=serial.Serial("/dev/ttyUSB1",9600)
ser.baudrate=9600
```

Мы используем стандартную скорость передачи данных 9600 бод. Эта скорость определяет, как быстро данные передаются по последовательной линии. Инвертировав скорость передачи данных, можно узнать, сколько времени требуется для передачи одного бита. Это значение определяет, как долго передатчик удерживает последовательную линию high/low или в какой период времени приемное устройство выполняет выборку своей линии [9].

При последовательном соединении устройств единственное требование заключается в том, что оба устройства должны работать с одинаковой скоростью.

Далее выводим символ «r» в монитор порта:

```
while True:
ser.write('r'.encode())
print('r')
```

Символ «r» обозначает, что программа распознала дорожный знак «Поворот направо». Соответственно обозначены и остальные дорожные знаки: «Поворот налево» – «l»; «Движение прямо» – «s». Написание одного символа «r» вместо целого слова «right» повышает быстродействие устройства.

Блок-схема алгоритма передачи данных с Raspberry Pi изображена на рисунке 4.3.2.

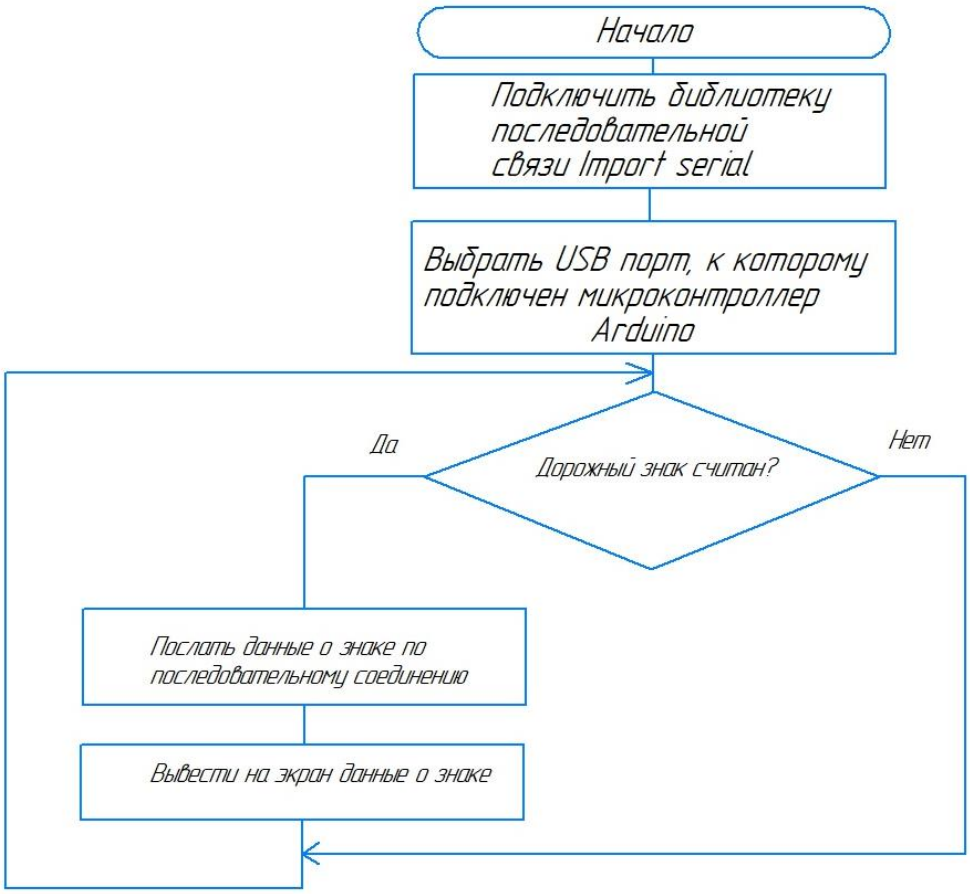

Рисунок 4.3.2 – Блок-схема алгоритма передачи данных

Программа, которую необходимо загрузить в микроконтроллер Arduino UNO, выглядит следующим образом:

```
void setup() {
Serial.begin(9600);
}
void loop() {
if(Serial.available()>0)
{
char b=Serial.read();
Serial.println(b);
delay(200);
}
}
```

Блок-схема алгоритма приема данных на Arduino UNO представлена на рисунке 4.3.3.

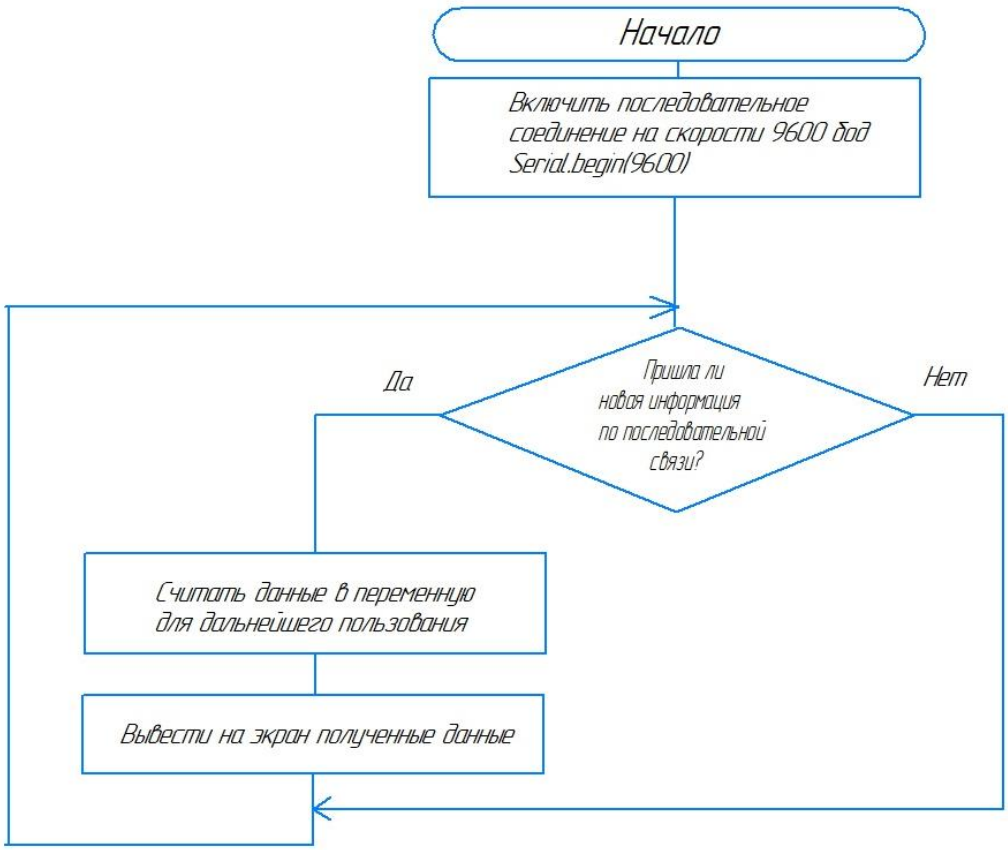

Рисунок 4.3.3 – Блок-схема алгоритма приема данных

В данной программе в первую очередь необходимо настроить монитор порта и выставить скорость передачи 9600 бод. В теле цикла проверить, пришла ли какая-либо информация. Если ответ положительный, то записать эту информацию в переменную и далее вывести ее в монитор порта. Дальнейшая обработка информации описывается в другой части комплексной выпускной квалификационной работы и не описана здесь.

После запуска основной программы, представленной в Приложении 1, Raspberry Pi распознает дорожный знак и подаст соответствующий сигнал на микроконтроллер Arduino UNO. В свою очередь, микроконтроллер подаст сигнал на моторы, и мобильный робот поедет по маршруту, предписанным установленными дорожными знаками.

4.4 Электрическая схема разрабатываемого устройства

На рисунке 4.4.1 представлена схема электрическая соединений устройства.

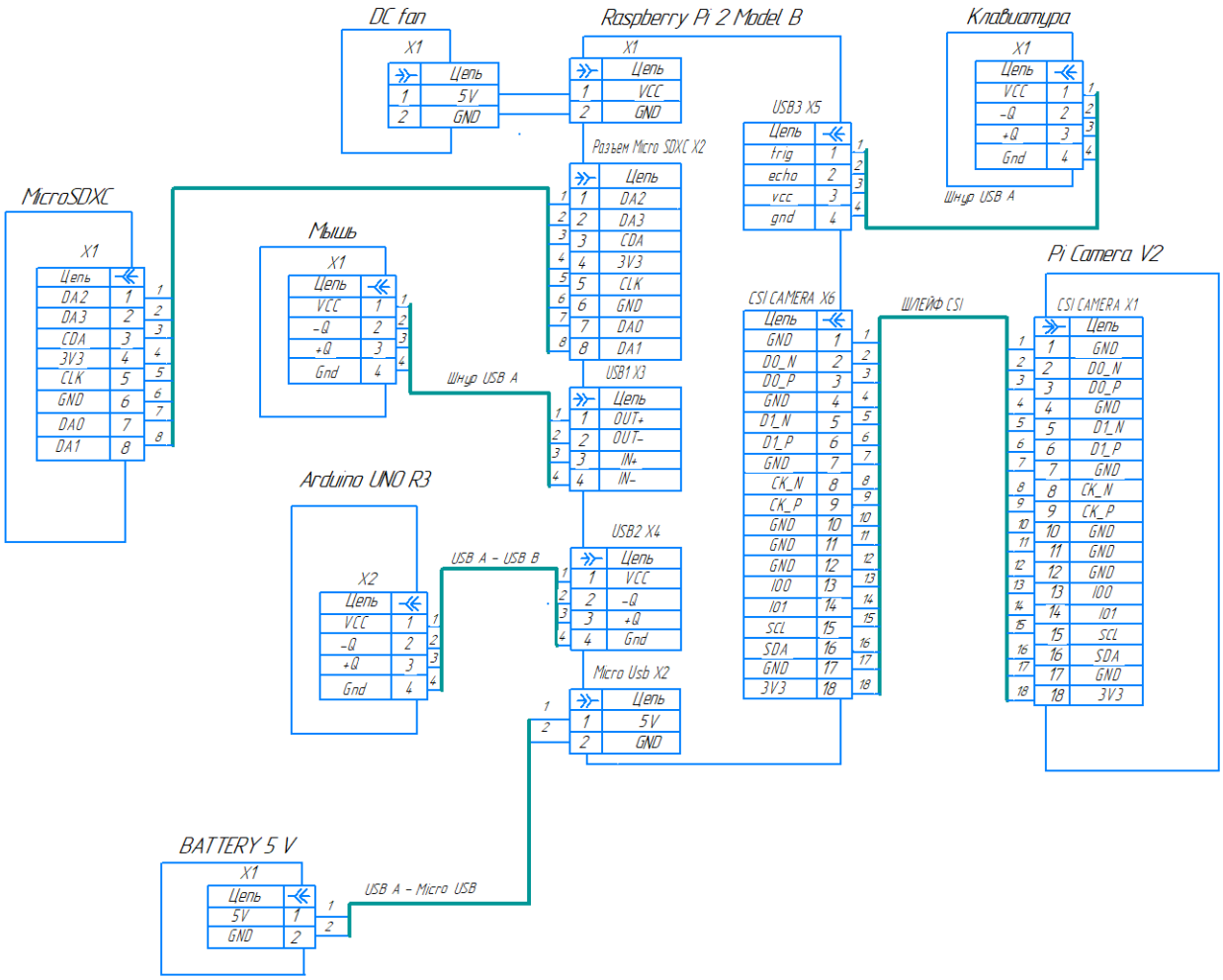

Рисунок 4.4.1 – Схема электрическая соединений

Отдельного внимания заслуживает CSI разъем для камеры Raspberry Pi. Контакты CAM1_CK_N и CAM1_CK_P обеспечивают тактовый импульс для полос данных MIPI для камеры. Они подключаются к контактам MIPI Clock Positive (MCP) и MIPI Clock (MCN) ICC камеры.

Контакты CAM1_D0_N и CAM1_D0_P – предназначены для вывода данных MIPI Data Positive (MDPI) и MIPI Data negative (MDN) для полосы данных 0 камеры 1.

Контакты CAM1_D1_N и CAM1_D1_P – предназначены для вывода данных MIPI Data Positive (MDPI) и MIPI Data negative (MDN) для полосы данных 1 камеры 1.

SCL и SDA – играют роль меньшей последовательной шиной и обеспечивают последовательную связь, благодаря которой пользователь может выбрать разрешение камеры. Эти контакты подключаются непосредственно к ведомому интерфейсу SCCB внутри IC камеры.

Вывод SCL обеспечивает стандартный входной тактовый сигнал последовательного интерфейса и стандартный последовательный интерфейс SDA для ввода/вывода данных [10].

В данном разделе был представлен перечень всех используемых электронных компонентов проекта и разработана схема электрическая соединений. Была написана программа и представлена блок-схема алгоритма соединения и передачи данных по последовательному порту с одноплатного компьютера на микроконтроллер. Также было подробно описано подключение камеры к одноплатному компьютеру при помощи CSI разъема.

5 Выбор инструментов

Для решения поставленных задач выпускной квалификационной работы был выбран язык программирования Python и библиотека компьютерного зрения с открытым исходным кодом OpenCV.

5.1 Язык программирования Python

Python – высокоуровневый язык программирования, ориентированный на повышение производительности разработчика и читаемости кода [11]. Данный язык программирования имеет минималистичный синтаксис, и в то же время стандартная библиотека включает в себя большой объем полезных функций.

В течение последних нескольких лет Python неуклонно растет в рядах языков программирования. В этом году он преодолел серию Java и занял 1 место [12]. Статистика популярности языков программирования за последние несколько лет представлена на рисунке 5.1.1.

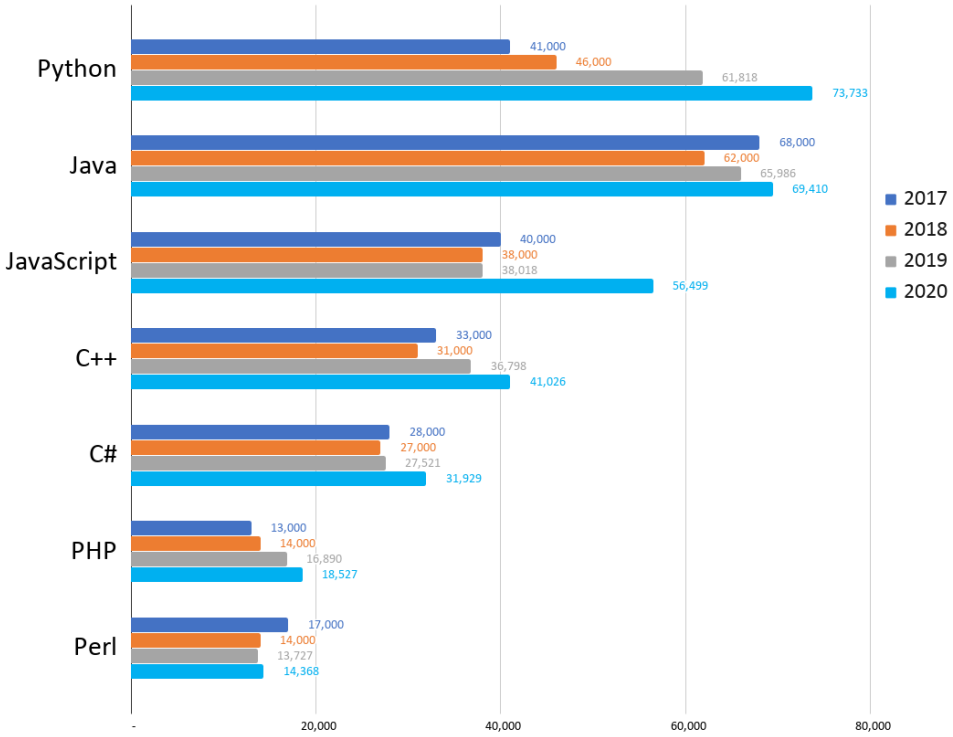

Рисунок 5.1.1 – Статистика популярности языков программирования

С помощью Google Trends – web-приложения от корпорации Google, показывающее, как часто определенный термин ищут по отношению к общему объему поисковых запросов Google, было определено, насколько люди заинтересованы в изучении Python. Результаты исследования представлены на рисунке 5.1.2.

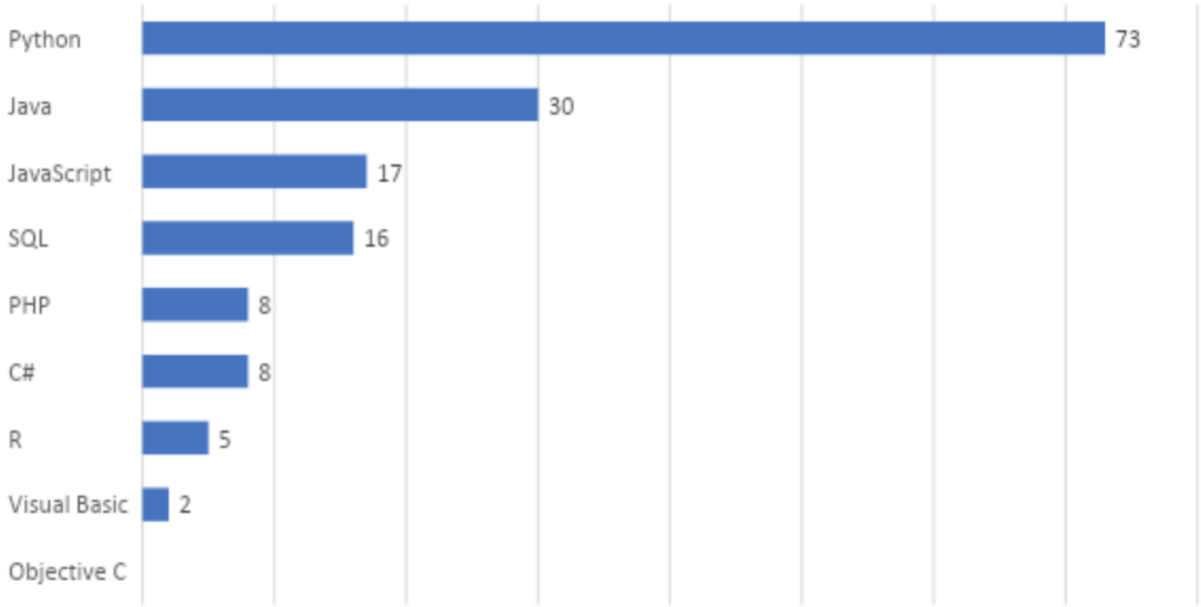

Рисунок 5.1.2 – Статистика поисковых запросов по языкам программирования

Проанализировав диаграмму, можно сказать, что Python получил почти столько же поисковых запросов, сколько и все остальные языки вместе взятые.

Успех Python обусловлен его универсальностью. В нем можно создать практически все что угодно. Кроме того, Python также используется для создания нейронных сетей для искусственного интеллекта. Частично это связано с обширной стандартной библиотекой, а также с тем, что она совместима с большинством основных систем и баз данных. Наконец, это язык с довольно простым синтаксисом, что делает его очень простым для чтения.

Python широко применяется в робототехнике. Пример простейшей программы для включения светодиода с помощью Raspberry Pi представлен на рисунке 5.1.3.

```
import RPi.GPIO as GPIO
import time

GPIO.output(pin,GPIO.HIGH)
```

Рисунок 5.1.3 – Программа для включения светодиода

На рисунке 5.1.4 приведен пример сравнения синтаксиса языка Python с синтаксисом языка С. В обоих случаях программа выводит надпись «Hello, world!», однако разница в длине кода очевидна.

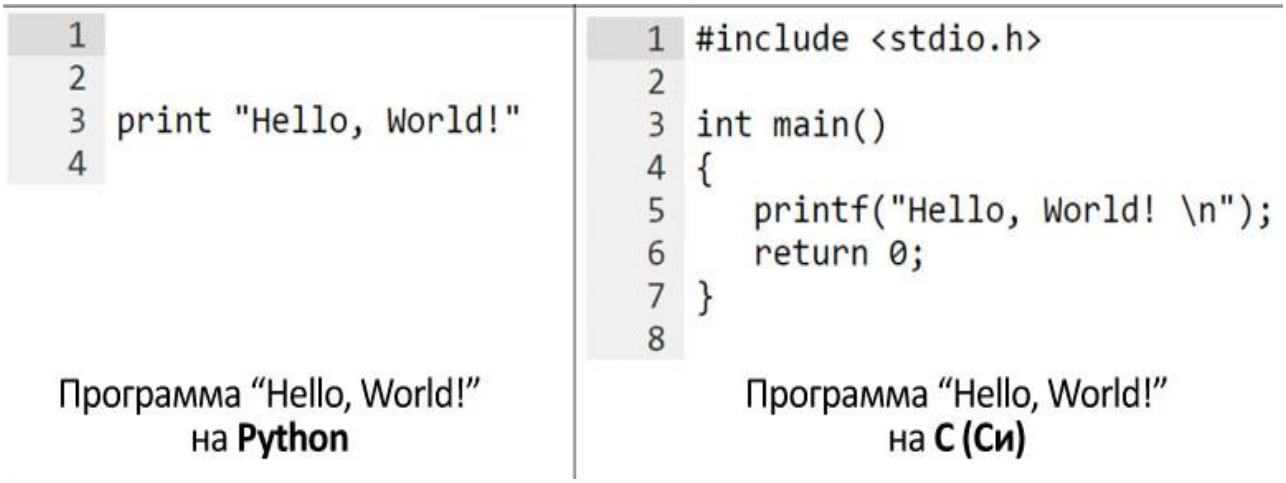

Рис. 5.1.4 – Сравнение синтаксиса языков программирования Python и С.

Принимая во внимание все вышеперечисленные достоинства языка Python, было принято решение об использовании его в рамках программного решения данной выпускной квалификационной работы.

5.2 Библиотека компьютерного зрения OpenCV

На сегодняшний день существует множество приложений в области компьютерного зрения: Computer Vision System Toolbox1 для MatLab, IMAQ Vision3, LabView2 и др. Но наиболее популярной является библиотека OpenCV.

OpenCV – библиотека компьютерного зрения с открытым исходным кодом. Цель ее разработки заключается в повышении вычислительной эффективности с уклоном на приложения реального времени [13]. Одним из главных преимуществ OpenCV является простота интерфейса, позволяющего людям довольно быстро осваивать технологии компьютерного зрения и на их основе строить сложные приложения.

С момента выхода альфа-версии в январе 1999 года, OpenCV была использована во многих приложениях и научно-исследовательских работах, самыми известными из которых являются:

– сшивка изображений спутниковых карт;

– выравнивание отсканированных изображений;

– уменьшение шума на медицинских снимках;

– анализ объектов, системы обнаружения вторжения;

– автоматический мониторинг;

– системы контроля, калибровка камеры;

– беспилотники;

– наземные и подводные аппараты.

Структура библиотеки OpenCV показана на рисунке 5.1.

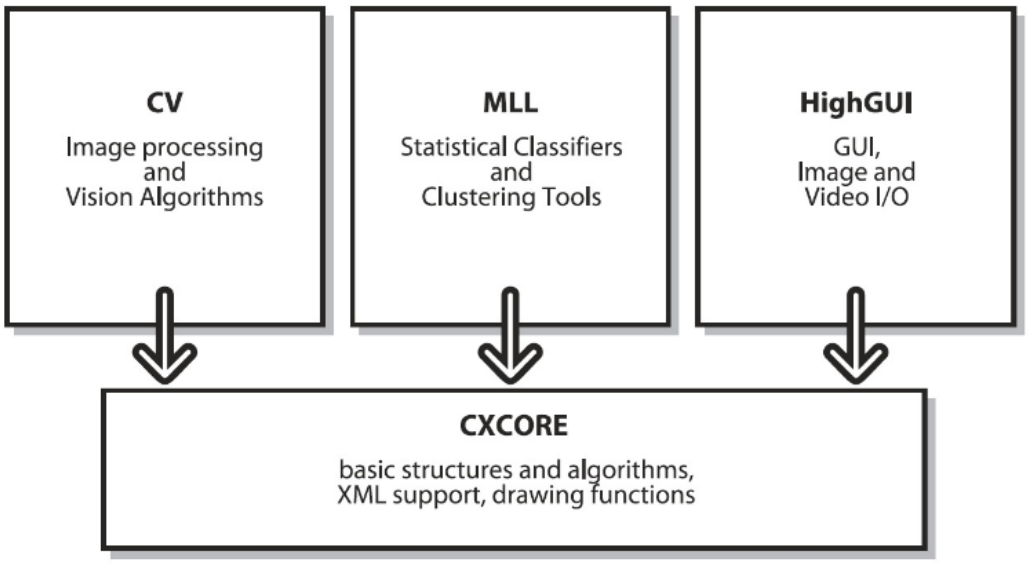

Рисунок 5.2.1 – Базовая структура OpenCV

Компонент CV содержит основные алгоритмы обработки изображений и высокоуровневые алгоритмы компьютерного зрения. ML содержит библиотеку машинного обучения, которая включает в себя средства статистической классификации и кластеризации. HighGUI содержит процедуры и функции ввода/вывода для хранения и загрузки видео и изображений. CXCore содержит основные структуры данных [14].

Благодаря простому интерфейсу и огромному количеству полезных функций, OpenCV идеально подходит для решения задачи распознавания дорожных знаков. Помимо библиотеки OpenCV нам необходимо так же подключить библиотеку NumPy для работы с математическими вычислениями.

5.3 Библиотека NumPy

NumPy – библиотека, предназначенная для научных вычислений в Python. Используя данную библиотеку, разработчик может выполнять следующие операции:

– математические и логические операции над массивами;

– преобразования Фурье и процедуры для манипуляции с формой;

– операции, связанные с линейной алгеброй.

Основной объект NumPy, который мы будем использовать в программе – однородный многомерный массив. Это таблица элементов (обычно чисел) одного типа, проиндексированных набором неотрицательных целых чисел. В NumPy размеры называются осями.

Например, координаты точки в трехмерном пространстве [1, 2, 1] имеют одну ось. Эта ось имеет 3 элемента, поэтому мы говорим, что она имеет длину 3. В приведенном ниже примере массив имеет 2 оси. Первая ось имеет длину 2, вторая ось имеет длину 3.

[[1., 0., 0.],

[0., 1., 2.]]

Класс массива NumPy называется ndarray. Он также известен под псевдонимом array. Необходимо знать, что numpy.array это не то же самое, что стандартный класс библиотеки Python array.array, который обрабатывает только одномерные массивы и предлагает меньшую функциональность. К наиболее важным атрибутам ndarray объекта относятся:

–ndarray.ndim - количество осей (размеров) массива;

– ndarray.shape - размеры массива. Это кортеж целых чисел, указывающих размер массива в каждом измерении. Для матрицы с n строками и m столбцами shape будет (n,m). Следовательно, длина shape кортежа равна числу осей ndim;

– ndarray.size - общее количество элементов массива и равняется произведению элементов shape;

– ndarray.dtype - объект, описывающий тип элементов в массиве. Можно создавать или указывать dtype, используя стандартные типы Python. Кроме того, NumPy предоставляет свои собственные типы: numpy.int32, numpy.int16 и numpy.float64;

– ndarray.itemsize - размер в байтах каждого элемента массива. Например, массив элементов типа float64имеет itemsize8 (= 64/8), а один из элементов типа complex32имеет itemsize4 (= 32/8). Это эквивалентно ndarray.dtype.itemsize.

– ndarray.data - буфер, содержащий фактические элементы массива. Обычно нам не нужно использовать этот атрибут, потому что мы будем обращаться к элементам в массиве, используя средства индексации [15].

Изображение является матрицей пикселей по размеру (высота x ширина). Обработка изображений в NumPy осуществляется следующим образом.

Если изображение черно-белое, то есть представлено в полутонах, каждый пиксель может быть представлен как единственное число в диапазоне от 0 (черный) до 255 (белый). Вот как выглядит фрагмент изображения (рисунок 5.3.1):

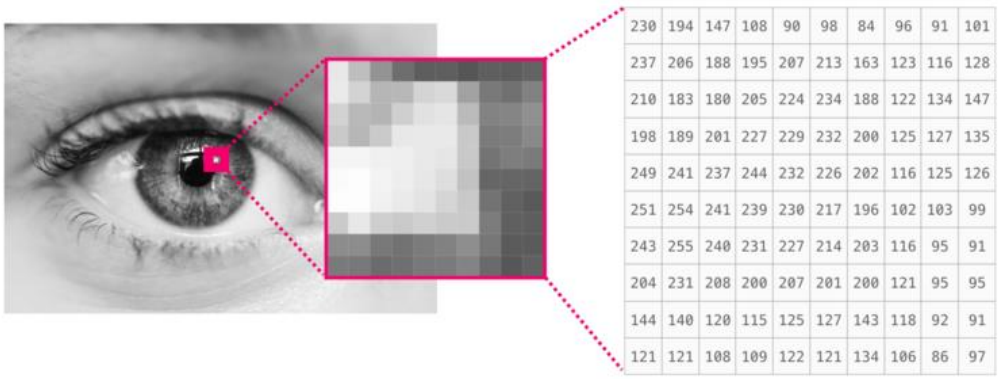

Рисунок 5.3.1 – Черно-белое изображение в матричном виде

В случае если изображение цветное – каждый пиксель представлен тремя числами. За основу берется цветовая модель RGB, где R – красный, G-зеленый, B – синий.

Поскольку каждая клетка вмещает только одно число, в этом случае нам необходима третья размерность и цветное изображение будет

представлено с помощью массива ndarray с размерностями: (высота x ширина x 3).

Рисунок 5.3.2 – Цветное изображение в матричном виде

С получившимся массивом мы можем делать все матричные преобразования с максимально быстрой и эффективной реализацией.

Логичным выводом из данного раздела может послужить то, что выбор языка программирования Python и библиотеки компьютерного зрения с открытым исходным кодом OpenCV является целесообразным, благодаря большому количеству модулей и методов.

6 Алгоритм распознавания образов

Изображения, полученные на выходе оптико-электронных преобразователей, искажены помехами. При анализе объектов на сложном фоне, фон также является помехой. С этой проблемой помогает справиться фильтрация – выделение интересующих нас области без их анализа. Большая часть методов фильтрации применяет какое-то единое преобразование ко всем точкам изображения. Результатом является оценка полезного сигнала изображения [16].

Существует множество способов считывания изображения компьютером, некоторые из которых необходимо комбинировать между собой для получения корректной информации об увиденном.

6.1 Цветовое пространство HSV

В отличие от чёрно-белых изображений, точка цветного изображения обычно кодируется тремя параметрами (трихроматическая теория), таким образом, картинка представляется как три однотипные матрицы.

Параметры, которые называют каналами изображения, можно выбирать разными способами. Наиболее известный - RGB (красный, зелёный, синий), каждый из трёх параметров RGB кодируется одним байтом, т.е. от 0 до 255.

Помимо RGB существуют и другие способы кодирования, например HSV (Hue, Saturation, Value) — тон, насыщенность, яркость.

Рисунок 2 показывает цветовое пространство HSV, преобразованное из цветового пространства RGB. Он представляет точки в цветовом

пространстве RGB перевернутым конусом, где H - это оттенок, S - это насыщенность, а V - это значение.

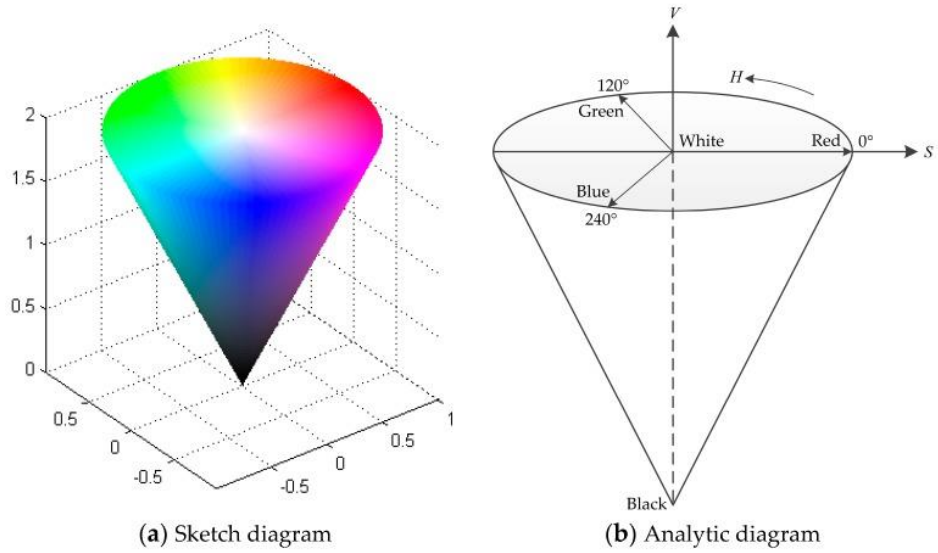

Рисунок 6.1.1 – Цветовое пространство HSV, преобразованное из цветового пространства RGB

H указывает на изменение цвета изображения. Положение спектрального цвета представлено углом, и разные значения цвета соответствуют разным углам. Красный, зеленый и синий находятся на расстоянии 120 ° друг от друга, то есть 0 °, 120 ° и 240 ° соответственно. S обозначает отношение текущей чистоты цвета к максимальной чистоте с максимальным значением 1 и минимальным значением 0. V представляет изменение яркости изображения. Максимальное значение 1 в белом, а минимальное значение 0 в черном. В цветовом пространстве HSV, учитывая, что V является набором фиксированных значений, а H и S сильно не связаны, цветовое пространство HSV имеет имеет более высокую скорость обнаружения, меньше подвержено влиянию освещения и обладает предпочтительным преимуществом сегментации.

Этот метод является наиболее подходящим для решения поставленной задачи, поскольку цвет является важной особенностью дорожного знака, и дорожный знак может быть быстро найден путем цветовой сегментации.

В библиотеке OpenCV за распознавание подходящего по цвету знака отвечает функция inRange(кадр, цвет_1, цвет_2), где

кадр — изображение, на которое мы накладываем фильтр;

цвет 1 — нижняя граница цвета;

цвет 2 — верхняя граница цвета.

Программа находит объект, подходящий под заданный диапазон цвета, убирает из кадра всё лишнее по цветовому признаку и получается черно-белое изображение. Нужный нам цвет становится белым, а все остальное черным, что очень сильно упрощает задачу детектирования [17].

Сегментация порога цветового пространства требуется после преобразования в цветовое пространство HSV.

Рисунок 6.1.2 показывает диаграмму шага сегментации порога цветового пространства.

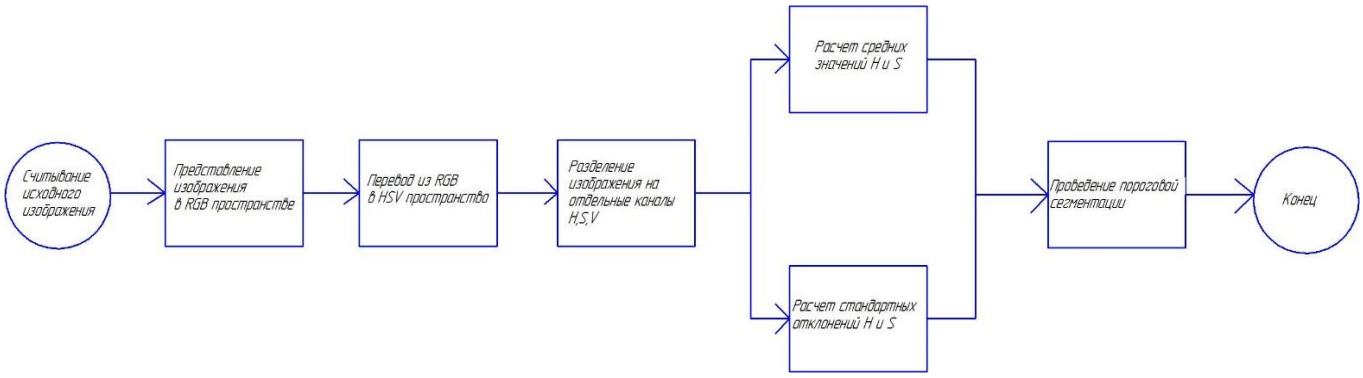

Рисунок 6.1.2 - Диаграмма шага сегментации порога цветового пространства.

После преобразования цветного изображение в двоичное, необходимо провести несколько морфологических операций.

6.2 Операции математической морфологии

Как правило, результат порогового преобразования содержит шум. Для борьбы с этой проблемой применяют операции математической морфологии. Каждая точка исходного изображения рассматривается с некоторой окрестностью. Форма и размер этой окрестности задаются структурным элементом, в простейшем случае это квадрат размера 3x3 точки. Для обработки изображений применяют две основные операции: сужение и расширение, а также их комбинации [18].

Морфологические преобразования - это простые операции, основанные на форме изображения. Обычно они выполняются на двоичных изображениях. Для этого нужны два входа: один - наше исходное изображение, второй - структурирующий элемент или ядро, которое определяет характер операции. Два основных морфологических оператора - это сужение и расширение. Также часто используются их разновидности, такие как открытие, закрытие. Рассмотрим принципы их действия с помощью следующего исходного изображения (рисунок 6.2.1):

Рисунок 6.2.1 - Исходное изображение

1. Основная идея сужения (Erode) - размытие границ объекта переднего плана. Это аналог логического И, точка принимает значение 1 (белый) если

все точки в окрестности белые. Таким образом, толщина или размер объекта переднего плана уменьшается. Это полезно для удаления небольших белых шумов отсоединения двух связанных объектов и т. д. Результат операции сужения представлен на рисунке 6.2.2.

Рисунок 6.2.2 - Результат операции сужения

Операция сужения в OpenCV задается следующим кодом:

erosion = cv2.erode(img, kernel, iterations = 1),

где img – исходное изображение;

kernel - ядро свёртки;

iterations – итерации.

2. Расширение (Dilate) – противоположность эрозии. Это аналог логического ИЛИ, точка принимает значение 1 (белый) если в окрестности есть хотя бы одна белая точка. Таким образом, увеличивается белая область на изображении или увеличивается размер объекта переднего плана. Обычно в таких случаях, после удаления шума, за сужением следует расширение. Сужение удаляет белые шумы, но при этом уменьшает наш объект. Таким образом мы его расширяем. Так как шум исчез, он не вернется, но площадь объекта будет увеличена. Такая операция бывает полезна при соединении

сломанных частей объекта. Результат операции расширения представлен на рисунке 6. 2.3.

Рисунок 6.2.3 - Результат операции расширения

Операция расширения в OpenCV задается следующим кодом:

dilation = cv2.dilate(img, kernel, iterations = 1),

где img – исходное изображение;

kernel - ядро свёртки;

iterations – итерации.

3. Открытие (opening) – последовательное выполнение сужения и расширения. Это просто еще одно название сужения, за которой следует расширение. Используется для удаления шума. Результат операции открытия представлен на рисунке 6.2.4.

Рисунок 6.2.4 - Результат операции открытия

Операция открытия в OpenCV задается следующим кодом:

```
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

4. Закрытие (closing) – последовательное выполнение расширения и сужения. Является противоположностью операции открытия. Операция используется при закрытии небольших отверстий внутри объектов переднего плана или маленьких черных точек на объекте [17]. Результат операции закрытия представлен на рисунке 6.2.5.

Рисунок 6.2.5 - Результат операции закрытия

Операция закрытия в OpenCV задается следующим кодом:

```
closing = cv2.morphologyEx( img,  cv2.MORPH_CLOSE,  kernel)
```

6.3 Выделение контуров

Ещё одним важным инструментом машинного зрения является функция выделения контуров. Выделив края и найдя контуры изображения, мы сразу отбрасываем лишнюю информацию. Этот подход имеет свои недостатки, например он чувствителен к шуму, но иногда с его помощью можно получить хороший результат.

В библиотеке OpenCV существует специальная функция для поиска контуров findContours() .

findContours(кадр, режим_группировки, метод_упаковки), где

кадр — изображение, на котором необходимо найти контуры.

режим_группировки — один из четырех режимов группировки найденных контуров:

CV_RETR_LIST — выдаёт все контуры без группировки; CV_RETR_EXTERNAL — выдаёт только крайние внешние контуры; CV_RETR_CCOMP — группирует контуры в двухуровневую иерархию. На верхнем уровне — внешние контуры объекта. На втором уровне — контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень;

CV_RETR_TREE — группирует контуры в многоуровневую иерархию.

метод_упаковки —один из трёх методов упаковки контуров: CV_CHAIN_APPROX_NONE — упаковка отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей; CV_CHAIN_APPROX_SIMPLE —склеивает все горизонтальные, вертикальные и диагональные контуры; CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS — применяет к контурам метод упаковки (аппроксимации) Teh-Chin. Он удаляет все лишние точки и сжимает контур, тем самым экономя память [19].

6.4 ROI

Region Of Interest (ROI) – определенная интересующая пользователя прямоугольная область на изображении, которую можно выделить для дальнейшей обработких [20]. Пример выделения ROI представлен на рисунке 6.4.1.

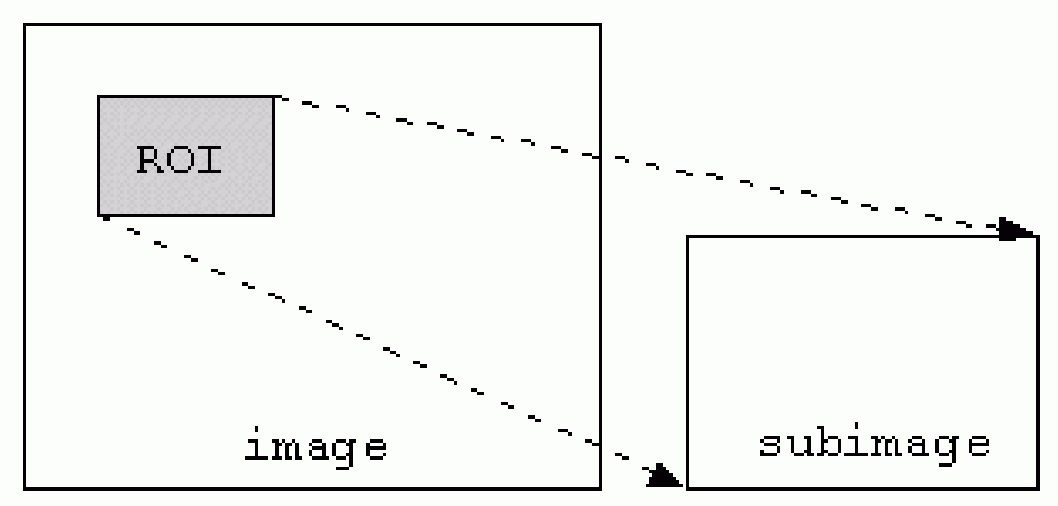

Рисунок 6.4.1 – Выделение ROI

После того как была определена ROI, большинство функций OpenCV

будут относиться к этой области, что значительно ускоряет работу программы для распознавания образов.

Для определения ROI в OpenCV используется следующая функция:

cvSetImageROI(IplImage*img,CvRect rect),

где img - исходное изображение;

rect - прямоугольная область внутри этого изображения.

Для удаления ROI используется функция:

cvResetImageROI(IplImage*img), где img- исходное изображение.

В данном разделе было подробно рассмотрено поэтапное преобразование получаемой картинки к виду, доступному для анализа на наличие дорожных знаков.

7 Программная часть

Программа основана на принципе поиска объекта по цвету через цветовое пространство HSV с дальнейшими морфологическими преобразованиями для удаления шумов. Листинг программы представлен в Приложении 1.

Алгоритм обнаружения дорожного знака можно описать следующим образом:

1. Задание параметров цвета – определение верхней и нижней границ.

2. Захват изображения – выделение самого большого участка заданного ранее цвета.

3. Преобразование изображения из BGR цветового пространства в HSV пространство.

4. Извлечение двоичного изображения необходимой области с цветом из HSV пространства.

5. Проведение морфологических операций для удаления шумов и зернистости.

6. Нахождение контура и обведение его в прямоугольник.

7. Деление площади на четыре зоны. Объявление координат и размеров ожидаемых «стрелок» слева, по центру, справа, сверху.

8. Выделение размера областей, в которых идет поиск пикселей.

9. Отслеживание доли белых пикселей.

10. При совпадении ожидаемых и полученных областей - вывод названия дорожного знака поверх оригинального изображения. При несовпадении – вывод надписи «None» и продолжение поиска - возвращение ко 2 пункту.

7.1 Блок-схема алгоритма

Блок-схема программы представлена на рисунке 7.1.1.

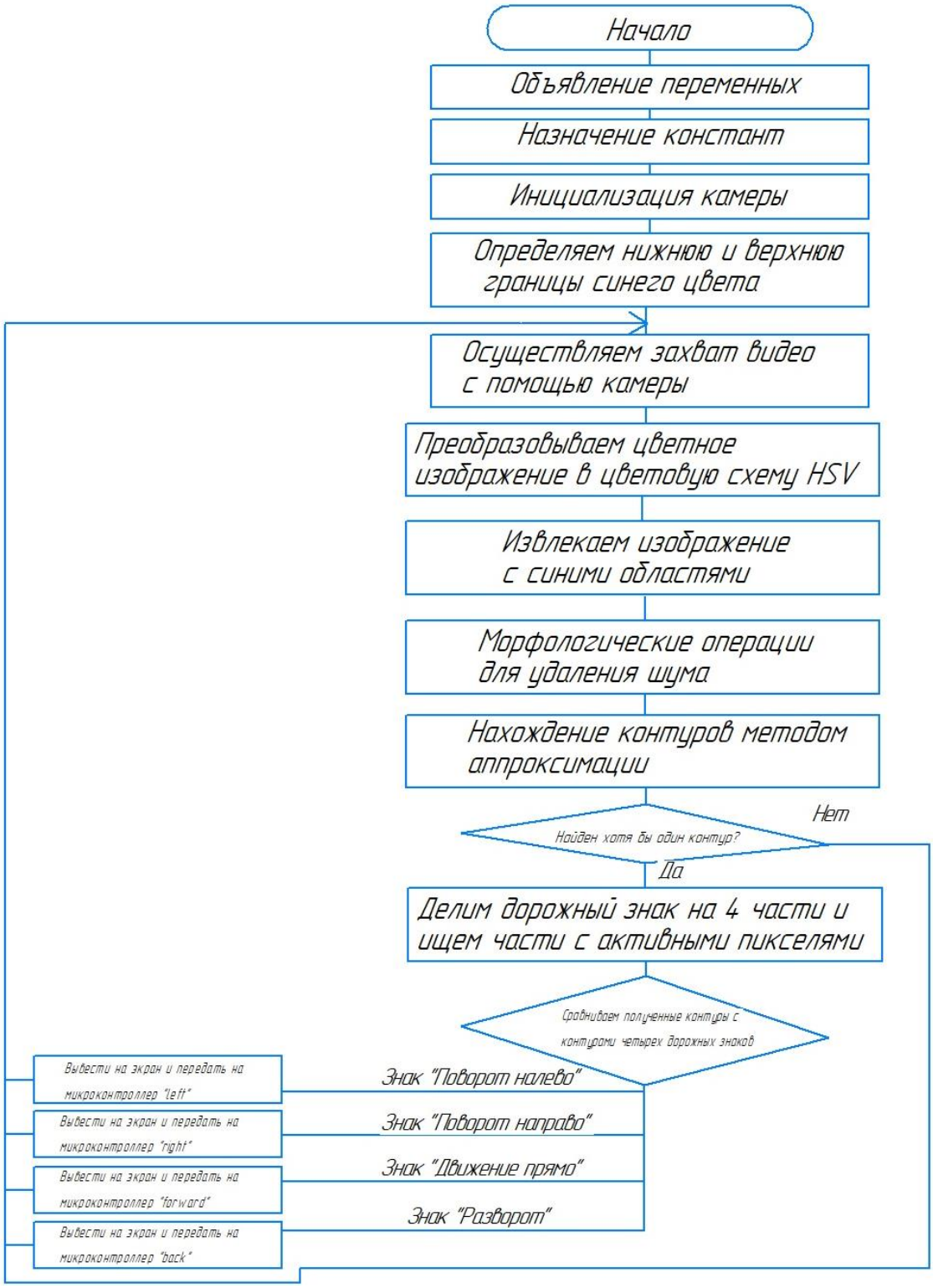

Рисунок 7.1.1 – Блок-схема программы

Проведем разбор отдельных ключевых функций программы.

Для определения дорожных знаков «Движения прямо», «Поворот направо», «Поворот налево» и «Разворот» нам нужно задать нижнюю и верхнюю границы характерного для этих знаков цвета, а именно синего цвета:

```
lower_blue = np.array([90,80,50])
upper_blue = np.array([110,255,255])
```

В элементы массивов мы записываем значения синего, зеленого и красного цвета в диапазоне от 0 до 255.

Далее для преобразования изображения из одного цветового пространства в другое используется функция cv2.cvtColor(input_image, flag), где flag определяет тип преобразованиях. OpenCV использует BGR формат. Конкретно для преобразования из BGR в HSV цветовое пространство мы используем функцию cv2.COLOR_BGR2HSV:

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Проверка попадания пикселей изображения в указанный диапазон от нижней до верхней границ заданного цвета, а так же извлечение изображения с интересующими нас областями синего цвета осуществляем за счет функции cv2.inRange():

```
mask = cv2.inRange(hsv, lower_blue, upper_blue)
```

Полученная картинка имеет шумы, которые мешают программе корректно распознать дорожный знак. В OpenCV морфологические операции проводятся с помощью функции cv2.morfologyEx() [21]. Проведение последовательных операций открытия и закрытия позволит получить на выходе изображение без различных шумов и зернистости:

```
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
```

На сглаженном изображении находим контуры с помощью функции findContours():

cnts=cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]

После того, как программа нашла необходимый контур, она должна обвести его в прямоугольник с помощью функции cv2.drawContours():

cv2.drawContours(frame,[largestRect],0,(0,0,255),2)

Указываем координаты и размеры ожидаемых «стрелок» слева, по центру, справа, сверху:

SIGNS_LOOKUP =

{

(1, 0, 0, 1): 'Right', # Поворот направо (активны левая и верхняя части)

(0, 0, 1, 1): 'Left', # Поворот налево (активны правая и верхняя части)

(0, 1, 0, 1): 'Straight', # Движение вперед (активны верхняя и центральная части)

(1, 0, 1, 1): 'Back', # Поворот назад (активны верхняя, левая и правая части)

}

Задаем порог:

THRESHOLD = 150

Инвертируем биты изображения:

image = cv2.bitwise_not(image)

Пиксели, значения которых меньше порога 150, преобразуются в черный цвет, а пиксели, значения которых больше этого порога — в белый цвет (рисунок 7.1.2).

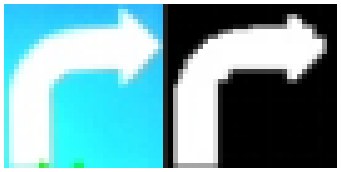

Рисунок 7.1.2 — Бинаризация изображения дорожного знака

Делим общую площадь на 10, получаем ширину и высоту:

(subHeight, subWidth) = np.divide(image.shape, 10)

Проводим полученные значения к целочисленному типу:

subHeight = int(subHeight)

subWidth = int(subWidth)

Выделяем размер областей, в которых идет поиск пикселей

leftBlock = image[4*subHeight:9*subHeight, subWidth:3*subWidth]

centerBlock = image[4*subHeight:9*subHeight, 4*subWidth:6*subWidth]

rightBlock = image[4*subHeight:9*subHeight, 7*subWidth:9*subWidth]

topBlock = image[2*subHeight:4*subHeight, 3*subWidth:7*subWidth]

Отслеживаем долю каждого участка:

leftFraction = np.sum(leftBlock)/(leftBlock.shape[0]*leftBlock.shape[1])

centerFraction = np.sum(centerBlock)/(centerBlock.shape[0]*centerBlock.shape[1])

rightFraction = np.sum(rightBlock)/(rightBlock.shape[0]*rightBlock.shape[1])

topFraction = np.sum(topBlock)/(topBlock.shape[0]*topBlock.shape[1])

Записываем в segments полученные данные:

segments = (leftFraction, centerFraction, rightFraction, topFraction)

Сравниваем с порогом и присваиваем каждому участку либо 0, либо 1:

segments = tuple(1 if segment > THRESHOLD else 0 for segment in segments)

Если полученные данные соответствуют какой-либо строчке массива SIGNS_LOOKUP, возвращаем распознанный знак, в противном случае ничего не возвращаем:

if segments in SIGNS_LOOKUP:

return SIGNS_LOOKUP[segments]

else:

return None

Выводим на экран полученное черно-белое изображение знака:

cv2.imshow("Warped", image)

Затем выводим название знака поверх оригинального изображения функцией cv2.putText():

cv2.putText(frame, detectedTrafficSign, (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 255, 0), 2).

Таким образом, разработана программа, основанная на принципе поиска объекта по цвету через цветовое пространство HSV с дальнейшими морфологическими преобразованиями для удаления шумов. Подробно описан каждый этап, начиная с задания параметров цвета верхней и нижней границы и заканчивая выводом сообщения о найденном знаке в последовательный порт с последующей передачей на микроконтроллер Arduino UNO.

8 Результаты экспериментальных испытаний

Результаты обнаружения дорожных знаков с помощью написанной программы можно увидеть на рисунках 8.1, 8.2, 8.3, 8.4.

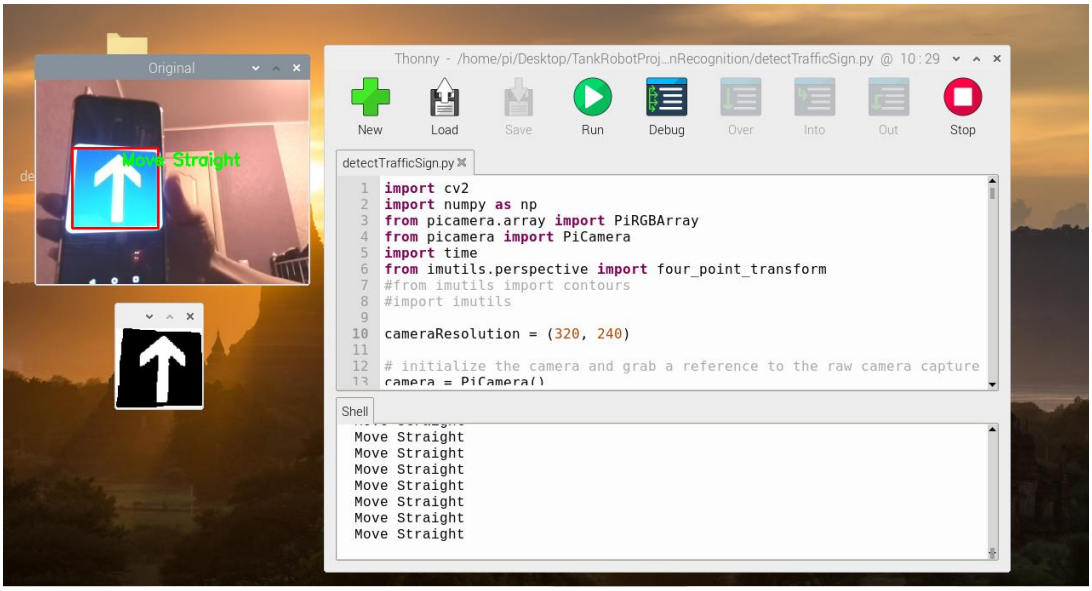

Рисунок 8.1 – Результат обнаружения дорожного знака «Движение прямо»

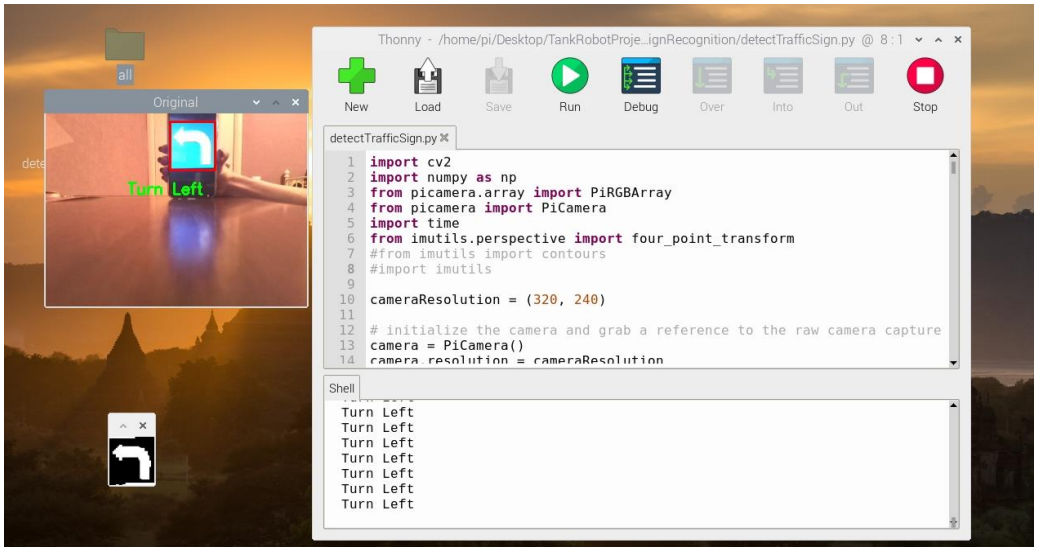

Рисунок 8.2 – Результат обнаружения дорожного знака «Поворот налево»

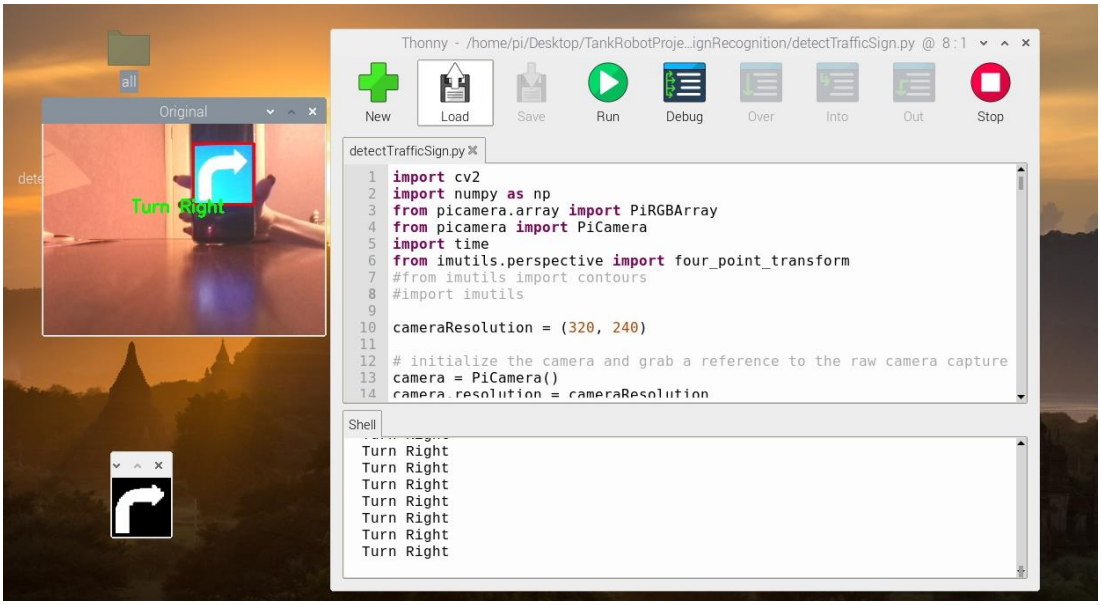

Рисунок 8.3 – Результат обнаружения дорожного знака «Поворот направо»

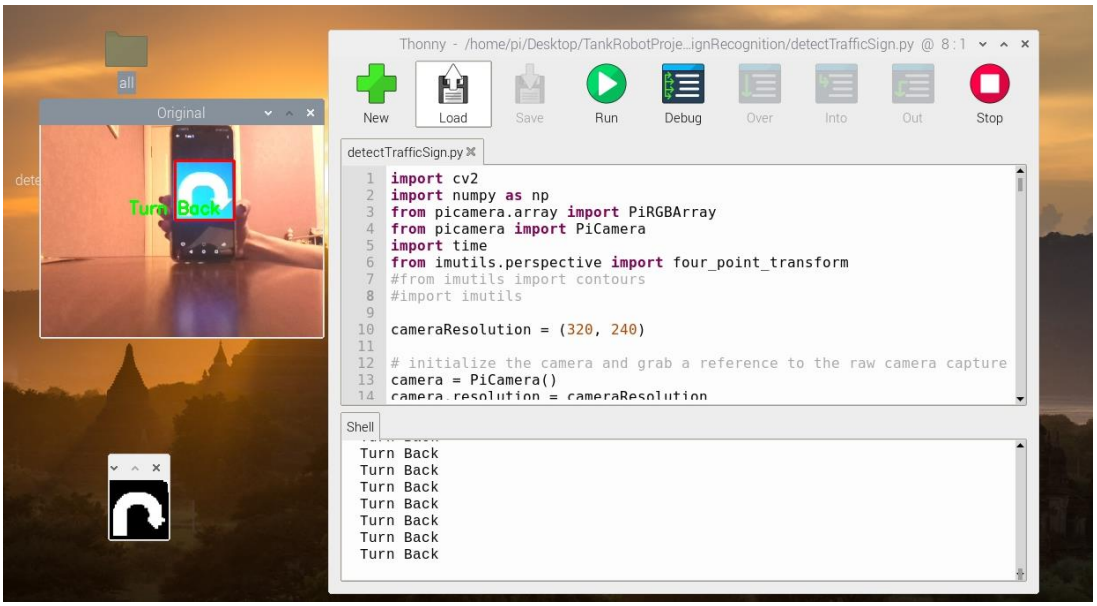

Рисунок 8.4 – Результат обнаружения дорожного знака «Разворот»

Из рисунков видно, что программа работает исправно и корректно распознала дорожные знаки «Движение прямо», «Поворот налево», «Поворот направо», «Разворот». Название знака выводится поверх оригинального изображения и прописывается в мониторе порта.

Слева снизу располагается окно с двоичным представлением знака.

После тестирования программы, устройство было установлено на роботизированную платформу и успешно считывало установленные знаки за 50 см до них. Тестирование проводилось в помещении с естественным освещением.

8.1 Влияние освещения на качество распознавания дорожных знаков

Для исследования влияния освещения на скорость и минимального расстояния для распознавания дорожных знаков, был произведен эксперимент: дорожный знак был показан системе через электронное устройство. С постепенным снижением яркости на устройстве, снижались так же минимальное расстояние для обнаружения дорожного знака и скорость его распознавания. Результаты исследования представлены в таблице 8.1.1.

Таблица 8.1.1 – Тестирование влияния освещения на качество распознавания дорожных знаков

Степень снижения освещения экрана, %	Минимальное расстояние для обнаружения дорожного знака, см	Скорость распознавания, с
Максимальная яркость	40	1
20	40	1
40	38	1
60	35	2
80	30	2

100	-	-

Проанализировав таблицу, можно сказать, что распознавание знака эффективно при хорошей его освещённости и совсем неэффективно при самом низком освещении, даже несмотря на то, что мы выбрали наименее зависимый от освещения метод распознавания объектов HSV. При этом тестирование проводилось в помещении с искусственным освещением, и мы можем наблюдать, что минимальное расстояние распознавания знака уменьшилось по сравнению с результатом испытаний при естественном освещении.

Таким образом, в данном разделе приведены изображения с результатами проведённых экспериментов. Отмечается качественное и безошибочное распознавание знаков при хорошем освещении и длительное распознавание при низком освещении.

9 Безопасность и экологичность проекта

Основная задача выпускной квалификационной работы заключается в написании программного кода. Следовательно, в процессе работы программист будет проводить большое количество в непосредственной близости от компьютера. Поскольку компьютер относится к электроустановкам, нужно предусмотреть следующие меры защиты, исключающие поражение человека электрическим током:

– наличие изоляции, предотвращающей «прямой» контакт с оголенными элементами электрических проводников;

– наличие защитного ограждения от внутреннего строения ЭВМ;

– исключение контакта электроустановки с жидкостью;

– исключение работы с выключателями или розетками, имеющими поврежденный корпус [22].

При работе с одноплатным компьютером для предотвращения перегрева и возможного его возгорания необходимо подключить к нему вентилятор. Однако стоит учесть, что при включенном с вентиляторе нельзя подносить пальцы в непосредственной близости от него.

Также необходимо предусмотреть защиту и от других вредных факторов, а именно: повышенного уровня шума от различных частей компьютера, недостаточная освещенность рабочего места и вредное электромагнитное излучение от монитора компьютера.

Под длительным воздействием шума у программиста появляется головная боль, повышается утомляемость, снижается память и концентрация внимания. Уровень шума на рабочем месте программиста не должен превышать 50дБА [23]. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами.

Недостаточная освещенность рабочего места влечет за собой перенапряжение зрительных органов, ослабление внимания и утомленность. Основные требования освещенности рабочих мест с компьютерами при выполнении зрительных работ предусматривают освещенность не менее 300лк. В качестве источников света при искусственном освещении должны применяться лампы с минимальным коэффициентом пульсации. Актуализированная редакция СНиП 23-05-95" и СанПиН 2.2.1/2.1.1.1278-03 определяют требования к пульсации света для работы с компьютерной техникой до 5% [24]. Кроме того, яркость экрана компьютера и степень освещенности рабочего места должны быть примерно одинаковыми.

Несмотря на противоречивые мнения о вреде длительного воздействия излучения от монитора компьютера, исчерпывающих данных в этой области не существует и исследования все еще ведутся. Однако установлены допустимые параметры неионизирующих электромагнитных излучений от монитора компьютера:

– напряженность электрической составляющей электромагнитного поля на расстоянии 50 см от поверхности монитора не должна превышать 10 В/м;

– Напряженность магнитной составляющей электромагнитного поля на расстоянии 50см от поверхности видеомонитора не должна превышать 0,3А/м;

– Напряженность электростатического поля для взрослых пользователей не должна превышать 20кВ/м [25].

Так, решением для снижения вредного воздействия излучения может стать применение мониторов с пониженным уровнем излучения, установка защитных экранов, а также соблюдение регламентированного режима труда и отдыха.

10 Экономический расчет

Стоимость каждого отдельно взятого компонента, входящего в состав разрабатываемого устройства, а также его итоговая стоимость приведена на рисунке 9.1.

№	Наименование	Кол	Цена, р
1	Raspberry Pi 2 Model B	1	2000
2	Кабель micro USB – USB	1	100
3	Внешний аккумулятор 5В 2.5А	1	1000
4	Raspberry Pi Camera V2	1	1300
5	Карта SD	1	400
6	Дисплей ЖК	1	700
7	Клавиатура	1	300
8	Мышь	1	200
9	Вентилятор	1	100
10	Пластиковый корпус для Raspberry Pi	1	300
		Итого	6400

Рисунок 9.1 – Экономический расчет

Таким образом, стоимость каждого отдельно взятого компонента невысока, что объясняется тщательным анализом рынка и сравнением всех доступных аналогов элементов.

Заключение

В рамках выполнения выпускной квалификационной работы была создана система распознавания дорожных знаков. Данная система внедрена в систему поиска пути мобильного робота. Готовый комплексный проект представляет собой движение мобильного робота в соответствии с дорожными знаками.

В первом разделе проанализированы аналоги, находящие свое применения в системах управления автомобилей и внутрискладских перемещениях грузов, отмечены их недостатки и отмечена актуальность разработки подобных систем.

Второй раздел посвящен выбору комплектующих для выполнения выпускной квалификационной работы. Обоснован выбор каждого элемента системы.

В третьем разделе представлены структурная схема разрабатываемого устройства и перечень всех используемых элементов.

В четвертом разделе была разработана схема электрическая соединений. Представлены схемы подключения вентилятора и модуля камеры к одноплатному компьютеру Raspberry Pi. Реализована аппаратная и программная связь одноплатного компьютера Raspberry Pi с микроконтроллером Arduino UNO.

В пятом разделе выбран необходимый инструментарий, изучены особенности его применения. Отмечаются преимущества использования языка программирования Python, библиотеки компьютерного зрения OpenCV и библиотеки NumPy в рамках выполнения выпускной квалификационной работы.

В шестом разделе описан метод распознавания знаков на основе поиска объекта по цвету через цветовое пространство HSV, отмечено его

преимущество перед остальными методами. Приводятся основные операции для удаления шумов с изображения, метод выделения контуров и интересующей области на изображении с помощью библиотеки компьютерного зрения OpenCV.

В седьмом разделе составлены блок-схемы алгоритма основных функций программы. Реализован алгоритм распознавания дорожных знаков.

В восьмом разделе представлены результаты обнаружения дорожных знаков с помощью написанной программы. Исследовано влияние негативных факторов, снижающих эффективность системы.

В девятом разделе определены меры техники безопасности при выполнении выпускной квалификационной работы.

В десятом разделе приведен экономический расчет разрабатываемого устройства.

Таким образом, результатом выполнения выпускной квалификационной работы является исправно работающая система распознавания дорожных знаков, внедренная в систему поиска пути мобильного робота.

Список используемой литературы

1. Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. — М.: Бином. Лаборатория знаний, 2006. — 752 с. — ISBN 5-94774-384-1.

2. E.R. Davies. Machine Vision : Theory, Algorithms, Practicalities. — Morgan Kaufmann, 2004.

3. Савенкова, Татьяна Ивановна. Логистика : учеб. пособие / Т. И. Савенкова. — 2-е изд., стер. — Москва : Издательство «Омега-Л», 2007. — 256 с.

4. Amazon Warehouses Upgraded With Computer Vision Technology [Электронный ресурс]. URL: https://www.scommerce.com/amazon-warehouses-upgraded-with-computer-vision-technology/

5. Роботы Amazon справляются со своими задачами в 4 раза быстрее человека [Электронный ресурс]. URL: https://habr.com/ru/post/395161/

6. Raspberry Pi 2 Model B Review [Электронный ресурс]. URL: https://www.pcmag.com/reviews/raspberry-pi-2-model-b

7. Raspberry Pi Camera Module Review and Tutorial Guide [Электронный ресурс]. URL: https://www.tweaktown.com/guides/5617/raspberry-pi-camera-module-review-and-tutorial-guide/index.html

8. Петин В. А. Микрокомпьютеры Raspberry Pi. Практическое руководство. — СПб.: БХВ-Петербург, 2015. — 240 с.: ил. — (Электроника) ISBN 978-5-9775-3519-9

9. Serial Communication [Электронный ресурс]. URL: https://learn.sparkfun.com/tutorials/serial-communication/all

10. Распиновка разъемов GPIO, DSI, CSI, 3.5 аудио/видео, I2S, тестовых точек в RaspberryPi [Электронный ресурс]. URL: https://pcminipro.ru/stati/raspinovka-razemov-gpio-dsi-csi-3-5-audio-video-i2s-testovyh-tochek-v-raspberrypi/

11. Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python. — Перевод с английского. — М.: ДМК Пресс, 2015. — с. — ISBN 978-5-9706-0330-7.

12. Top 7 Programming Languages Of 2020 Review [Электронный ресурс]. URL: https://www.codingdojo.com/blog/top-7-programming-languages-of-2020

13. Кэлер А., Брэдски Г. Изучаем OpenCV 3. Разработка программ компьютерного зрения на C++ с применением библиотеки OpenCV. - Издательство "ДМК Пресс", 2017. – ISBN 978-5-97060-471-7.

14. Gary Bradski, Adrian Kaehler. Learning OpenCV. - O'Reilly Media, 2008. - ISBN: 9780596516130

15. Quickstart tutorial [Электронный ресурс]. URL: https://numpy.org/doc/1.18/user/quickstart.html

16. В.Т. Фисенко, Т.Ю. Фисенко. Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. – 192 с.

17. Python, Машинное зрение, OpenCV. Часть 1 [Электронный ресурс]. URL: https://линуксблог.рф/python-mashinnoe-zrenie-opencv-chast-1/

18. Berthold K.P. Horn. Robot Vision. — MIT Press, 1986. — ISBN 0-262-08159-8 (Б.К.П. Хорн, Зрение роботов: перевод с англ. — М.: Мир, 1989).

19. E. R. DAVIES. Computer and Machine Vision: Theory, Algorithms, Practicalities- Elsevier, 2012. - ISBN: 978-0-12-386908-1

20. Ron Brinkmann (1999). The Art and Science of Digital Compositing. Morgan Kaufmann. pp. 184. ISBN 978-0-12-133960-9.

21. Д. Форсайт, Ж. Понс. Компьютерное зрение. Современный подход = Computer Vision: A Modern Approach. — М.: «Вильямс», 2004. — 928 с. — ISBN 5-8459-0542-7.

22. Розанов В.С. Безопасность жизнедеятельности. Электробезопасность. М., МИРЭА, 1999 г.

23. ГОСТ 12.1.009 – 76. Электробезопасность. Термины и определения. Введ. 01.01.77

24. Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий // tehlit : Техническая литература. 2015. URL: http://www.tehlit.ru/1lib_norma_doc/11/11776/index.htm /

25. ГОСТ 12.1.002-84 Система стандартов безопасности труда (ССБТ). Введ. 1986-01-01