

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт радиоэлектроники и информационных технологий

Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы и сети
(наименование)

Кафедра Вычислительные системы и технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра
(бакалавра, магистра, специалиста)

Студента Курганского Романа Сергеевича группы 16-В-2
(Ф.И.О.)

на тему Аппаратно-программная система организации движения роботов в колонне
(наименование темы работы)

СТУДЕНТ:

Курганский Р.С.
(подпись) (фамилия, и., о.)
04.07.2020
(дата)

КОНСУЛЬТАНТЫ:

1. По _____

(подпись) (фамилия, и., о.)

(дата)

РУКОВОДИТЕЛЬ:

Гай В.Е.
(подпись) (фамилия, и., о.)
04.07.2020
(дата)

2. По _____

(подпись) (фамилия, и., о.)

(дата)

РЕЦЕНЗЕНТ:

(подпись) (фамилия, и., о.)

(дата)

3. По _____

(подпись) (фамилия, и., о.)

(дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ

Жевнерчук Д.В.
(подпись) (фамилия, и.о.)
06.07.2020
(дата)

ВКР защищена _____
(дата)

протокол № _____

с оценкой _____

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Кафедра «Вычислительные системы и технологии»

УТВЕРЖДАЮ

Зав. кафедрой ВСТ

Жевнерчук Д.В.

«12» мая 2020 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы
по направлению подготовки (специальности) 09.03.01 «Информатика и
вычислительная техника»

(код и наименование)

студенту Курганскому Роману Сергеевичу группы 16-В-2
(Ф.И.О.)

1. Тема ВКР «Аппаратно-программная система организации движения роботов в колонне»

(утверждена приказом по вузу от 15.04.2020г. №867/5)

2. Срок сдачи студентом законченной работы 03.07.2020г.

3. Исходные данные к работе

Язык разработки Python, Симулятор роботов Webots, Документация по библиотекам
Webots

4. Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих
разработке)

Введение

1. Требования к продукту

2. Анализ требований к разрабатываемому продукту

3. Разработка структуры системы движения робота в колонне

4. Разработка программных средств

5. Тестирование системы

Заключение

Список литературы

Приложение

5. Перечень графического материала (с точным указанием обязательных чертежей)

1. Структурная схема системы управления движением робота

2. Структурная схема алгоритма движения ведущего робота под управлением системы
GPS

3. Модель робота с автопилотом

4. Объекты, определенные радаром

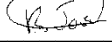
6. Консультанты по ВКР (с указанием относящихся к ним разделов)

Руководитель: Гай В.Е.

Нормоконтроль: Жевнерчук Д.В.

7. Дата выдачи задания: 12.05.2020г.

Код и содержание Компетенции	Задание	Проектируемый результат	Отметка о выполнении
ПК-1 — Способность определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов ограничений	Разработка структурной схемы системы и алгоритмов	Алгоритмы реализации движения роботов в организованной колонне	
ПК-2 — Способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	Реализовать модель робота и разработанные алгоритмы на одном из языков программирования с использованием виртуальной среды	Моделирование робота в среде Webots и реализация программы на языке Python	
ПК-3 — Способность анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями	Составление и оформление пояснительной записки в соответствии с ГОСТ	Пояснительная записка к выпускной квалификационной работе	

Руководитель  Гай В.Е.
(подпись)

Задание принял к исполнению 12.05.2020г.
(дата)

Студент  Курганский Р.С.
(подпись)

**МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)**

АННОТАЦИЯ

к выпускной квалификационной работе

**по направлению подготовки (специальности) 09.03.01 «Информатика и
вычислительная техника»**

студенту Курганскому Роману Сергеевичу группы 16-В-2

по теме «Аппаратно-программная система организации движения роботов в колонне»

Выпускная квалификационная работа выполнена на 40 страницах, содержит 4
схемы, 11 изображения, библиографический список из 5 источников, 3 приложений.

Тема выпускной квалификационной работы: «Аппаратно-программная система
организации движения роботов в колонне».

Актуальность: в текущий момент времени происходит переход на автоматическое
управление в большинстве видах деятельности человека, основанием для этого служит
существенное упрощение жизнедеятельности, автоматизация также коснулась и
автомобильного транспорта.

Объект исследования: автоматизация движения автомобильного транспорта и роботов.

Предмет исследования: алгоритмы и методы организации движения роботов в колонне.

Цель исследования: разработка аппаратно-программной системы организации движения
роботов в колонне.

Задачи исследования: исследовать существующие типы систем позиционирования
роботов в окружающей среде; исследовать основные датчики используемые в
беспилотных автомобилях и роботах; разработать собственные алгоритмы
обеспечивающие движения роботов в организованной колонне; разработать собственную
программную систему реализующую обработку информации из внешней среды и
генерирующую команду движения робота; выполнить тестирование с целью проверки
работоспособности разработанной аппаратно-программной системы.

Методы исследования: Создание и моделирование движения роботов в виртуальной среде
симулятора; Анализ систем позиционирования роботов и датчиков для разработки
алгоритмов движения роботов.

Структура работы: выпускная квалификационная работа состоит из введения, пяти глав, заключения, списка литературы и приложения.

В введении дается описание проблемы, лежащей в основе данной работы.

В 1 разделе «Требования к продукту» составлено техническое задание на разработку.

Во 2 разделе «Анализ требований к разрабатываемому продукту» производится выбор программных средств, симулятора роботов, обзор существующих систем позиционирования роботов на местности, приводится обоснование выбора системы, в соответствии с которым будет производиться выбор датчиков и разработка алгоритмов обеспечения движения в колонне.

В 3 разделе «Разработка структуры системы движения робота в колонне» разрабатывается структурная схема, алгоритмы решения каждого из этапов решения задачи.

В 4 разделе «Разработка программных средств» разрабатываются программные средства для решения поставленной задачи.

В 5 разделе «Тестирование системы» описывается метод тестирования системы и полученные результаты.


В заключении приводятся основные выводы по работе.

Выводы:

1. Разработана аппаратно-программная система организации движения роботов в колонне
2. Тестирование системы подтвердило её работоспособность и возможность использования для решения поставленной задачи.

Рекомендации:

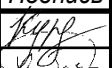
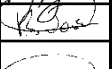
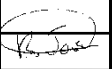

1. Дальнейшее развитие проекта.
2. Оптимизация существующего кода.

 /Курганский Р.С.
подпись студента /расшифровка подписи

«03» июля 2020 г.

Оглавление

Введение.....	5
1. Требования к продукту	6
1.1. Назначение разработки и область применения.....	6
1.2. Технические требования	6
2. Анализ требований к разрабатываемому продукту	7
2.1. Выбор симулятора	7
2.2. Выбор языка программирования.....	9
2.3. Выбор среды разработки.....	10
2.4. Выбор системы позиционирования роботов.....	12
3. Разработка структуры системы движения робота в колонне	14
3.1. Разработка общей структуры системы	14
3.2. Разработка модели робота.....	16
3.3. Разработка алгоритма движения ведущего робота под управлением системы GPS	18
3.4. Разработка алгоритма движения ведущего робота под управлением системы автопилота	20
3.5. Разработка алгоритма движения ведомых роботов.....	21
4. Разработка программных средств.....	23
5. Тестирование системы.....	34
5.1. Описание методики тестирования	34
5.2. Проверка работы системы ведомого робота	34
5.3. Проверка работы системы ведущего робота с системой GPS.....	36
5.4. Проверка работы системы ведущего робота с системой автопилота.....	37
Заключение.....	40
Список литературы.....	41
Приложение.....	42

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)			
Изм.	Лист	№ докум.	Подпись	Дата	Аппаратно-программная система организации движения роботов в колонне Пояснительная записка	Лит.	Лист	Листов
Разраб.		Курганский Р.С.		04.07				
Провер.		Гай В.Е.		04.07			4	60
Н. контр.		Гай В.Е.		04.07		Кафедра "Вычислительные системы и технологии"		
Утверд.		Жевнерчук Д.В.		06.07				

Введение

В текущий момент времени происходит переход на автоматическое управление в большинстве видах деятельности человека, основанием для этого служит существенное упрощение жизнедеятельности.

Одной из видов такой деятельности являются автомобили и техника на автоматизированном управлении. Это автомобили или же техника, которые оборудованы системами автоматического управления и способны передвигаться без управления со стороны человека.

Данная техника нужна для повышения безопасности на дорогах, повышения уровня производительности поездок, а также в целом для понижения участия водителя в процессе управления транспортом. Основными разработчиками программного и аппаратного обеспечения беспилотных автомобилей являются такие компании как Tesla, Apple, Google, Uber, Audi и BMW [5].

Также использование беспилотных грузовых автомобилей в колонне позволяет повысить аэродинамическую эффективность, что ведет к сокращению потребления топлива.

Основные задачи, которые должна решать техника с возможностью автоматического организованного движения в колонне являются: распознавание разметки, отслеживание ведущего робота, позиционирование с помощью системы GPS.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						5
Изм.	Лист	№ докум.	Подп.	Дата		

1. Требования к продукту

1.1. Назначение разработки и область применения

Разрабатываемая аппаратно-программная система предназначена для автоматического управления роботами обеспечивающая их движение в организованной колонне. Может использоваться как отдельный модуль из нее или же как комплекс программ для решения поставленной задачи. Она может быть применена в различных сферах, таких как транспортная или же в промышленном производстве. В транспортной сфере решаются такие задачи, как уменьшение участия человека в управление транспортом, увеличение безопасности и экономии топлива. За счет чего достигается большая экономическая эффективность.

В производственной отрасли так же наблюдается тенденции перехода на автоматизацию. Например, колонны роботов доставщиков самостоятельно доставляют компоненты для сборки между различными цехами на производстве.

1.2. Технические требования

Требования, предъявляемые разрабатываемой системой к ЭВМ:

1. Операционная система с графическим интерфейсом – Microsoft Windows, Linux;
2. Требования к аппаратному обеспечению определяются операционной системой;
3. Устройство ввода: клавиатура;
4. Устройство вывода: дисплей;

Требования к функционалу разрабатываемой системы организации движения роботов в колонне:

1. Система должна обеспечивать движение роботов в последовательной колонне;
2. Система должна обеспечивать автоматическое движение роботов по дорожной разметке;
3. Система должна обеспечивать автоматическое движение роботов по GPS к заданной координате;
4. Система должна обеспечивать остановку при обнаружении неподвижного препятствия

2. Анализ требований к разрабатываемому продукту

2.1. Выбор симулятора

Одним из значительных пунктов в разработке системы является выбор программы симулятора в которой будет выполняться основная часть работ по проектированию роботов и моделированию их работы. Существует достаточное количество симуляторов, однако не многие из них предоставляют необходимые инструменты для реализации данной системы. Рассмотрим каждый из них.

Webots – это программа для моделирования роботов. Она позволяет пользователям создавать виртуальные сцены и предметы имеющие такие физические свойства как масса, сила трения и др. Пользователь может создавать статические объекты или активные объекты (роботы). Роботы могут иметь различные способы передвижения: колесные роботы, шагающие роботы или же летающие роботы. Помимо всего, они могут быть оборудованы разнообразными устройствами, например: инфракрасный датчик, колеса, камера, датчик расстояния, радар и многие другие. Отдельно для каждого робота пользователь может написать необходимую программу для управления, называемую контроллером, чтобы добиться требуемого поведения. После симуляции и отладки реализованный контроллер можно перенести на реальную модель робота. Также Webots включает в себя встроенную среду разработки для контроллера. Имеет поддержку таких языков как Python, Java, C/C++ и является полностью свободной.

Gazebo – программный комплекс, представляет собой среду для симулирования работы виртуальных роботов с различными сенсорами в окружении всевозможных объектов. Приложение состоит из графической части и части по имитированию взаимодействия твердых объектов, позволяя моделировать динамику и кинематику механизмов роботов (включая моменты взаимодействия с телами внешней среды) и формировать физически правдоподобные показания виртуальных датчиков. Симулятор имеет гибкий дизайн и удобный интерфейс, поддерживающий одновременную работу с несколькими устройствами. Gazebo является открытой и свободной программой для использования. Однако поддерживает только язык C++ и требует его хорошего знания для работы с ним.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист 7
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>		

V-Rep – робосимулятор с широким функционалом, поддерживается на операционных системах Windows, Linux, MacOS. Симулятор свободно распространяется для использования в личных целях. Также имеются библиотеки для программирования роботов с помощью C/C++, Python, Java, Matlab. Поставляется с набором готовых моделей стационарных и мобильных роботов, управлять которыми можно программируя файлы контроллера. Для управления определенными моделями имеются выделенные наборы ползунков. Во всех роботах уже предустановлена базовую программу и моделирование осуществляется согласно реальным законам физики.

Проанализировав все вышеперечисленные симуляторы роботов, был выбран Webots. Основными факторами такого решения стали поддержка многих языков программирования, удобный интерфейс, интегрированная среда разработки и наличие уже готовых моделей роботов, автомобилей и необходимых сцен для моделирования движения роботов в колонне.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						8
Изм.	Лист	№ докум.	Подп.	Дата		

2.2. Выбор языка программирования

Для разрабатываемой системы организации движения роботов в колонне могут быть применены следующие поддерживаемые в выбранном симуляторе языки программирования: C++, Java, Python. Рассмотрим каждый из приведенных выше языков.

Python – высокоуровневый язык программирования, который хорошо интегрируется с существующими программами и имеет простой синтаксис, тем самым повышая читаемость кода. Включает в себя поддержку многих библиотек, таких как Numpy, Math, а также интеграцию с языками C/C++. Язык кроссплатформенен, что является плюсом при использовании его на различных устройствах, однако ему необходим предустановленный интерпретатор [2].

Java – кроссплатформенный, объектно-ориентированный и сетевой язык программирования, является одним из самых популярных языков. Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной машине. Она обрабатывает байт-код и передает инструкции оборудованию, как это делает интерпретатор. При этом байт-код, в отличие от текста, обрабатывается значительно быстрее. Вместе с тем, синтаксис сложнее чем у Python, а разница в скорости выполнения не так велика при относительно небольших размерах программ.

C++ – язык программирования, обладающий высокой скоростью выполнения программного кода, а также поддерживающий работу с графикой. Язык программирования содержит в своем составе богатую стандартную библиотеку. C++ предоставляет широкие возможности для решения всевозможных задач, но при этом имеет более сложный синтаксис и необходимость постоянного контроля за памятью. Так же язык является самым быстрым по выполнению, так как он сразу компилируется, а не транслируется или интерпретируется. Однако это необходимо выполнять отдельно для каждой архитектуры.

Проанализировав плюсы и минусы, и характеристики приведенных языков программирования, можно сделать вывод, что наиболее подходящим и оптимальным вариантом для разработки будет использование языка Python.

2.3. Выбор среды разработки

Для разработки программного кода в проекте будет использоваться интегрированная среда разработки (IDE), программа, предназначенная для разработки программного обеспечения, содержащая такие инструменты как редактор, предназначенный для обработки кода, средства сборки, выполнения и отладки, а также может иметь поддержку системы контроля версий. Существует множество сред разработки, обычно предназначенные для нескольких языков, однако здесь будут рассматриваться только специализированные под Python. Выбор среды разработки играет важную роль в процессе реализации программных средств, так как от этого зависит насколько эффективен данный процесс. Рассмотрим некоторые популярные среды разработки, которые специализированы для работы с выбранным языком Python: Pycharm, Spyder и встроенный редактор Webots.

PyCharm - редактор кода PyCharm предоставляет обширную поддержку языка Python. Предлагает большой набор инструментов из коробки: встроенный отладчик и инструмент запуска тестов, профилировщик Python, полнофункциональный встроенный терминал, инструменты для работы с базами данных. Инструменты обнаружения ошибок, автозавершения кода и автоматического исправления кода. Предоставляет функции интеллектуального поиска, позволяющая переходить к любому классу, файлу, символу или окну инструмента. Также, PyCharm имеет удобный интерфейс для взаимодействия с основными системами контроля версий, такими как Git, SVN.

Spyder - свободная среда разработки для Python, предоставляющая набор таких возможностей как редактирование, анализ, отладка и профилирование. Это комплексный инструмент разработки, способный к исследованию данных, интерактивному выполнению, глубокой проверке и доступный интерфейс визуализации. С помощью ruflakes в реальном времени код программы проверяется на ошибки, и в случае их нахождения пользователь получает подсказки. Одной из уникальных инструментов является возможность удобной работы с переменными. В процессе выполнения программы они отображаются на панели в виде списка с возможностью просмотра их значений, что очень удобно при отладке программы и поиске логических ошибок. Также в Spyder имеется возможность просмотра документации или исходных кодов любых объектов Python (классов, функций, модулей). Также доступна онлайн документация

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						10
Изм.	Лист	№ докум.	Подп.	Дата		

модулей Python (сгенерированная в формат html). Для документации в Python используется библиотека Sphinx.

Встроенная среда разработки Webots – представляет собой редактор кода использующий установленный на машине интерпретатор Python и содержащий набор необходимых базовых функций, такие как автозавершение кода, подсветка синтаксиса, а также удобные инструменты поиска. Плюсом данной среды является то, что она интегрирована в симулятор, что позволяет быстро отлаживать, редактировать код и сразу же запускать симуляцию с внесенными изменениями.

Рассмотрев все приведенные варианты в качестве среды для разработки была выбрана встроенная среда разработки Webots. Она предоставляет все необходимые возможности для разработки, так же является более легковесной по сравнению с другими перечисленными средами.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	<i>Лист</i>
						11
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>		

2.4. Выбор системы позиционирования роботов

Существует несколько концепций позиционирования роботов в пространстве, рассмотрим основные из них и определим какие будут наиболее подходящими для разрабатываемой системы.

Глобальная система

В глобальной системе происходит получение координат, таких как широта и долгота. Основными системами являются GPS, RTK-GPS, Глонасс, использующие спутники для нахождения позиции. Точность таких систем зависит от множества факторов, но в условиях, близких к идеальным наиболее развитая из данных систем, GPS, способна обеспечить точность с ошибкой в пределах метра. Применение систем такого типа ограничивается условиями, в которых находится техника. Системы глобального ограничены в возможностях их использования в замкнутых пространствах, таких как здания, закрытые парковки и т. д. Следовательно, использование такого типа систем позиционирования актуально использовать на открытой местности и больших расстояниях. С увеличением размеров робота возрастает значимость использования глобальной системы для его позиционирования.

Персональная система

Персональная система применяется при позиционировании отдельных составляющих робота и взаимодействии с окружающими предметами. Такая точность важна для роботов, у которых имеются манипуляторы. Этот тип систем применяется для позиционирования в условиях ограниченной территории, например, для координации робота-сборщика. Также персональные системы реализации движения применяются при следовании по заданной разметке или при передвижении по линии. Основная проблема при применении в реальной жизни данного типа систем является их узкая настройка под конкретную территорию, практически не могут адаптироваться под изменившиеся условия внешней среды. Преобладающая сфера использования персональных систем – работа в пределах заранее определенной территории.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						12
Изм.	Лист	№ докум.	Подп.	Дата		

Автономная система

В рамках автономной системы навигации применяются гироскопы, цифровые компасы. Основная сложность при использовании таких систем является то, что они чувствительны к различного рода неровностям рельефа, например, наклоны, ямы и т. д. Это может существенно ограничить их применение. Автономные системы навигации находят применение в условиях, когда качество сигналов управления извне недостаточен или отсутствует совсем. Этот аспект важен в среде с отсутствием возможности передавать сигнал.

Локальная система

Локальные системы используют для позиционирования некоторую точку, обычно стартовую. Данные системы могут выполнять позиционирование на относительно большого размера площадях, например, для беспилотных самолетов, работающих в рамках заданной местности. Система навигации A-GPS, использующая для позиционирования сотовые сети, также является локальной. В настоящее время наиболее часто применяются системы, использующие такие датчики, как лазерные, инфракрасные, ультразвуковые и т. д.

Рассмотрев приведенные системы позиционирования, основными в реализуемой структуре будут выбраны персональная и глобальная системы.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						13
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>		

3. Разработка структуры системы движения робота в колонне

3.1. Разработка общей структуры системы

Задачей при разработке системы организации движения роботов в колонне является создание алгоритма для движения для робота, а также реализация комплекса систем взаимодействия с внешней средой.

Система, обеспечивающая движение в колонне, подразумевает наличие как минимум двух роботов. Они представляют из себя ведущего и ведомого. Предусмотрена замена ролей при невозможности ведущего робота выполнять одну из своих функций. Обобщенная структура системы показана на рисунке 1.

1. Ведущий робот

Ведущий робот представляет собой автомобиль, оснащенный системами взаимодействия со внешней средой. Его основной задачей является реализация движения на основе полученных данных из внешней среды. Для обеспечения контроля в сценарии использования на автотранспорте, в салоне присутствует водитель, выбирающий типы движения и способный принять на себя контроль автомобилем. Осуществляет два типа движения:

Движение на автопилоте следующего в полосе – пользователь включает систему автоматического движения в полосе и задает необходимую скорость, обеспечивающая следование автомобиля по полосе дорожной разметки. Реализуя этот метод движения, система способна распознать автомобиль или препятствие при необходимости обеспечить торможение. В случае остановки, если в течении 10 секунд автомобиль или препятствие, которое находится в полосе не начнет перемещение, система передает управление водителю, чтобы он смог выполнить объезд. Также обеспечивает регулирование скорости при условии, что движущийся впереди автомобиль едет медленнее заданной водителем скорости.

Движение на основе системы GPS – пользователь вводит необходимую координату, к которой робот должен следовать, если робот находится на открытой местности или набор, состоящий из нескольких точек, для более точного движения и обхода уже известных препятствий. Также реализует набор по контролю скорости из системы автопилота.

2. Ведомый робот

Ведомый робот также является автомобилем, оснащенным теми же системами взаимодействия со внешним миром что и у ведущего робота. Однако активно работает только одна из них, она определяет робота впереди и обеспечивает повторение траектории движения ведущего. Это сделано для возможности замены ведущего автомобиля при выходе из строя одной из его систем. Во время движения в ведомом роботе отсутствует водитель, так как система всего лишь дублирует движения и не нуждается в контроле со стороны человека.



Рисунок 1 – Обобщенная структурная схема систем реализации движения роботов

3.2. Разработка модели робота

Перед тем как данные будут обработаны они должны будут поступить на системы взаимодействия с внешней средой. Для корректного выполнения все задач ведущего и ведомого робота необходимо определить, какие датчики необходимо установить на робота или автомобиль. Для каждой единицы в колонне будет предусмотрен единый набор датчиков. Рассмотрим выбранные устройства:

Камера – представляет собой обычную камеру, используется на ведущем роботе для распознавания дорожной разметки.

Радар – датчик, измеряющий расстояние и угол отклонения автомобиля или препятствия. Задействован на ведущем роботе для обеспечения адаптивного регулирования скорости или избегания столкновения. Работает в паре с камерой при включенном автопилоте или же с системой GPS. Используется на ведомом роботе как единственный датчик, на данных которого осуществляется управления.

GPS приёмник – устройство возвращающее текущие координаты робота на местности, применяется на ведомом роботе при включенном режиме управления по GPS. Также, как и камера работает совместно с радаром.

Стоит заметить, что для ведущего автомобиля радар передает данные только для регулировки скорости, тогда как в ведомого полностью позволяет управлять скоростью и направлением на основе своих показаний. Можно было бы заметить его на простой датчик расстояния, однако, при выходе из строя, например, камеры этот датчик не сможет предоставить необходимые данные для системы ведомого робота.

Рассмотренные выше устройства предоставляют системе роботов все необходимые данные для их корректного движения [4].

Для установки данных средств на модель в Webots, необходимо выбрать саму модель и слот, в который будет установлен необходимый датчик рисунок 2.



Рисунок 2 – Выбор модели и слота для датчика

Далее необходимо добавить из списка, выбранный модуль рисунок 3.

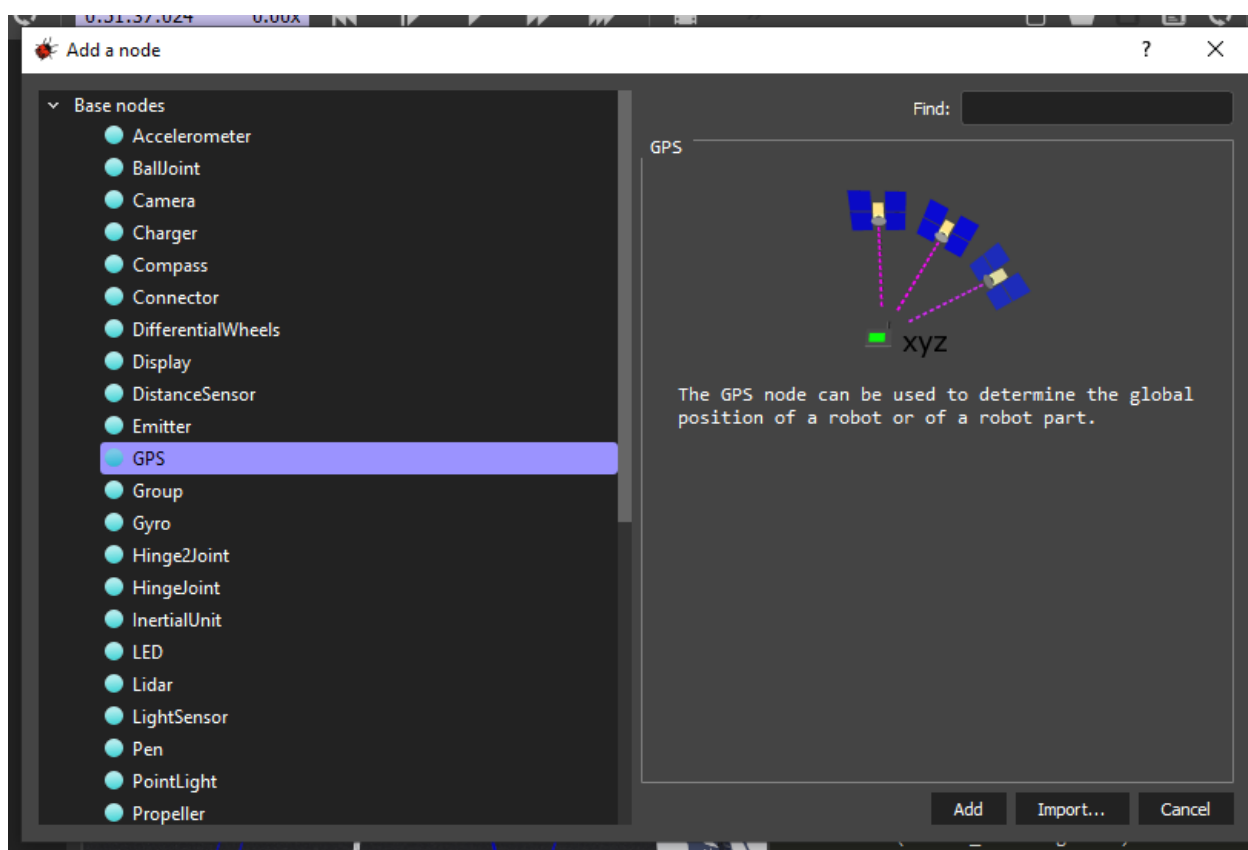


Рисунок 3 – Добавление GPS приемника в Webots.

3.3. Разработка алгоритма движения ведущего робота под управлением системы GPS

Для контроля робота под управлением системы GPS был реализован алгоритм, определяющий направление движения, используя вычисление углов между векторами – рисунок 4.

Рассмотрим его подробнее:

1. Инициализация и получение текущей координаты.
2. Задание пользователем нужной точки или последовательности этих точек.
3. Вычисляем вектор направления движения робота.
4. Вычисляем вектор от координаты робота к заданной точке.

5. Выполняем поворот вектора необходимого направления на 90° по часовой стрелке, это необходимо для определения стороны, в которую требуется повернуть колеса. При отрицательном скалярном произведении необходимо поворачивать колеса влево и наоборот.

6. Производим вычисление скалярного произведения с вектором и берем знак.

7. Вычисляем угол между векторами по формуле: $\arccos \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$

8. Полученный угол умножаем на знак, полученный от скалярного произведения со смещенным вектором, и передаем получившееся значение для установки угла колес.

Параллельно выполняется проверка дистанции до заданной точки по формуле:

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

При достижении дистанции меньше метра, берется следующая заданная точка или управление переходит в ручной режим.



Рисунок 4 – Схема работы регулятора угла установки колес по GPS.

В реализации движения по GPS, основной алгоритм выполняет задачу установки угла колес, скорость задается в ручном режиме. Активно задействованный радар и алгоритм регулирования скорости, заимствованный у ведомого робота, выполняет роль системы предотвращения столкновений и срабатывает при обнаружении препятствия.

3.4. Разработка алгоритма движения ведущего робота под управлением системы автопилота

Алгоритм, реализующий беспилотное движение ведущего робота, заключается в обработке изображении камеры, распознавание полосы дорожной разметки и корректировки угла поворота колес с помощью ПИД – регулятора.

На начальном этапе происходит инициализация оборудования – камеры и радара. Изображение, поступившее с камеры, представляет собой массив пикселей в формате RGB, далее массив обрабатывается функцией поиска заданных пикселей цвета разметки. При условии отсутствия на изображении пикселей необходимого цвета система предупреждает водителя об отсутствие условий для автопилотирования. Определяя смещение распознанных пикселей в определенную сторону изображения формируется угол и происходит его нормирование. Затем угол подается в программный фильтр значений, выполняя усреднение для устранения так называемых «шумов». Отфильтрованный результат поступает на вход ПИД – регулятора, производящий финальное преобразования угла, с помощью формулы:

$$u(t) = k_p * e(t) + k_i * \int_0^t e(t) dt + k_d * (y(t) - y(t-1))$$

, где $e(t)$ - ошибка (рассогласование), $u(t)$ - выходной сигнал регулятора [1]. Подбор коэффициентов для пропорциональной, интегральной и дифференциальной части осуществлялся эмпирическим путем, осуществляя постепенное увеличение каждого, начиная с пропорционального, до тех пор, пока не достигнем оптимального результата.

После всех описанных выше шагов выполняется установка направления колес. Также, как и на системе с GPS, радар выполняет активное сканирование полосы на наличие препятствия или автомобиля. В случае если препятствие или автомобиль, блокирующий движение, остается неподвижен, управление передается водителю для обхода и возврата в режим автопилота.

3.5. Разработка алгоритма движения ведомых роботов

Основной задачей ведомого робота, следуя описанной концепции, является повторение движения ведущего робота – рисунок 5. Для осуществления этого был использован данный алгоритм:

1. Инициализируем радар
2. На вход системы поступает расстояние в метрах и угол отклонения в радианах до идущего впереди робота
3. Обработка полученного расстояния
4. Установка скорости полученной после обработки дистанции
5. Установка угла колес в соответствии с отклонением



Рисунок 5 – Схема работы ведомого робота.

На этапе 3, обработка происходит с помощью отдельного алгоритма, который реализует пропорциональную часть ПИД – регулятора. Суть алгоритма заключается в определении дистанции, на которой должно осуществляться торможение и ускорение, также если расстояние находится в промежутке между ними происходит регулирование на основе текущей и предыдущей дистанции. Выполняется сравнение, если расстояние меньше предыдущего, скорость

уменьшается, аналогичным образом происходит корректировка скорости при отставании. Блок схема алгоритма представлена на рисунке 6.

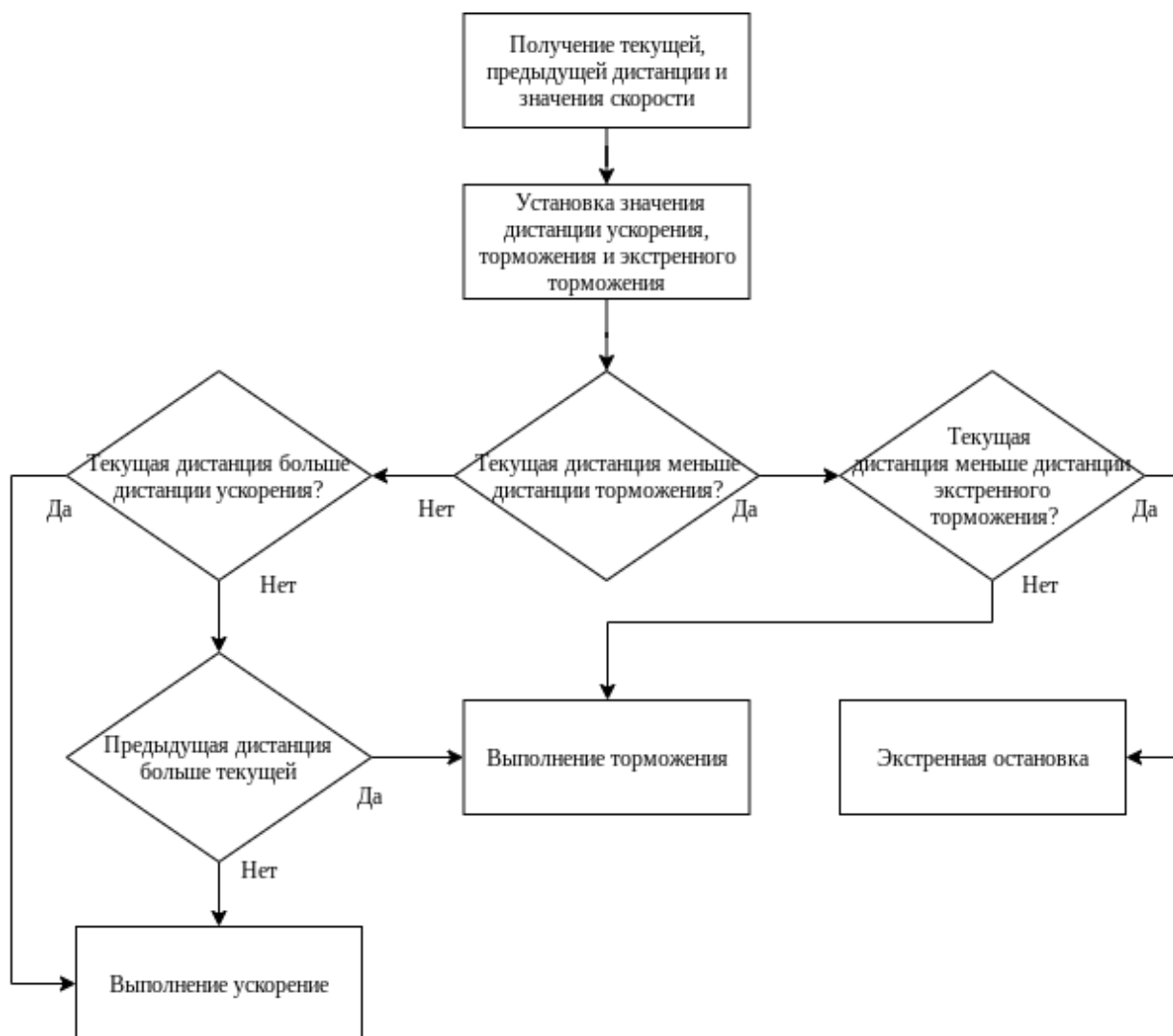


Рисунок 6 – Схема работы регулятора скорости.

4. Разработка программных средств

Программная система, обеспечивающая организацию движения в колонне, состоит из 3 основных частей. Две части, одна из которых реализует движение основываясь на данных GPS, а вторая выполняет функции автопилота базируясь на изображении с камеры, используются для управления ведущим роботом и часть, предназначенная для ведомого, выполняет повторение движения ведущего. Однако, помимо этого, необходимо выполнить реализацию модели робота в симуляторе для дальнейшей отладки и тестирования системы.

Рассмотрим модель робота, разработанной в симуляторе Webots. Для ускорения разработки выберем уже существующую модель грузовика, так как использование системы на данном типе транспорта является приоритетной. Далее выберем основные датчики: GPS, радар, камера. Добавим выбранные датчики на грузовик. Порядок расположения, следующий - два GPS приемника установлены на крыше грузовика, радар установлен в передней крайней части грузовика для увеличения области сканирования, камера слежения за полосой установлена на левом наружном зеркале заднего вида. На приведенном ниже рисунке продемонстрирована модель грузовик с уже установленными средствами взаимодействия с внешней средой, рисунок 7.



Рисунок 7 – Модель робота в симуляторе с установленными датчиками.

Модель робота в симуляторе описывается иерархической структурой, рисунок 8.

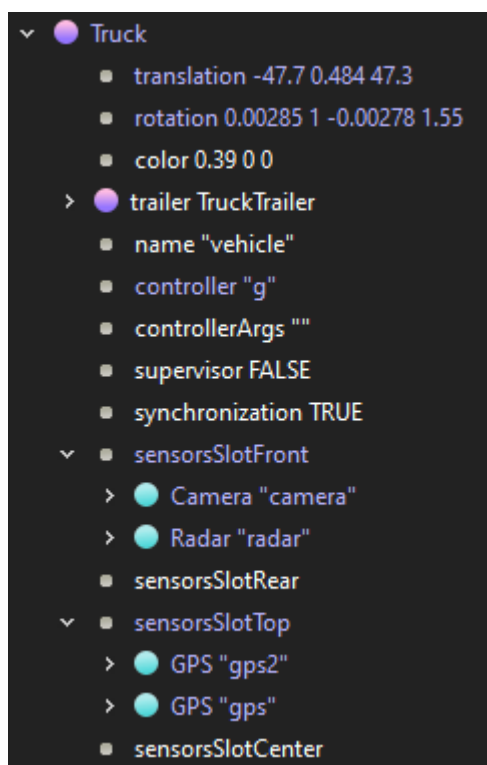


Рисунок 8 – Описание модели робота иерархической структурой

Можно увидеть, что в специально отведенные слоты были установлены все необходимые датчики. Помимо модели грузовика, также было добавлено уже имеющееся окружение города, в котором в дальнейшем будет выполняться симуляция.

Сконфигурировав модель в симуляторе, перейдем к рассмотрению части программной системы управления ведомым роботом. В ведомом роботе используется только радар, соответственно необходимо выполнить подключение атрибута по работе с ним из модуля controller, а также атрибут driver для управления грузовиком из модуля vehicle.

```
from controller import Radar
```

```
from vehicle import Driver
```

Основной цикл выполнения программы:

```
while (driver.step() != -1):
```

В каждом цикле выполняется проверка на наличие цели в области видимости радара данным условием:

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						24
Изм.	Лист	№ докум.	Подп.	Дата		

```
if (radar.getNumberOfTargets() > 0 ):
```

```
    for i in range(radar.getNumberOfTargets()):# получение данных с радара и занесение их в списки
```

```
        azim.append(radar.getTargets()[i].azimuth)
```

```
        dist.append(radar.getTargets()[i].distance)
```

При условии нахождения более одной цели перед радаром, ориентиром будет ближайшая находящаяся к нему.

```
if (radar.getNumberOfTargets() > 1 ):
```

```
    for i in range(radar.getNumberOfTargets()-1):# поиск ближайшей цели радара
```

```
        if dist[i] < dist[i+1] and dist[i] < shortest_distance
```

```
            count = i
```

```
            shortest_distance = dist[i]
```

```
            speed = speed_calibrating(dist[count], prev_dist, speed)
```

```
            set_speed(speed)
```

```
            change_manual_steer_angle(azim[count])
```

```
            prev_dist = dist[count]
```

```
speed = speed_calibrating(dist[0], prev_dist, speed)
```

```
set_speed(speed)
```

```
change_manual_steer_angle(azim[0])
```

```
prev_dist = dist[0]
```

Если радар не обнаружил ни одну цель, робот останавливается

```
else:
```

```
    set_speed(0)
```

```
    driver.setSteeringAngle(0)
```

```
pass
```

Метод, который выполняет регулировку называется speed_calibrating, он принимает на вход текущую и предыдущую дистанцию, также скорость.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						25
Изм.	Лист	№ докум.	Подп.	Дата		

В нем установлена оптимальная дистанция ускорения и торможения, вычисленная опытным путем, для движущегося в колонне грузовика.

```
def speed_calibrating(dist, prev_dist, speed):
```

```
    accel_dist = 11 # дистанция ускорения
```

```
    break_dist = 10.5 # дистанция торможения
```

```
    emergency_dist = 7 # дистанция экстренного торможения
```

```
        Выполняем проверки на необходимость ускориться или затормозить
```

```
    if dist < break_dist:
```

```
        if dist > 9.5:
```

```
            speed-=2
```

```
        else:
```

```
            speed-=3
```

```
    elif dist > accel_dist:
```

```
        speed += 2
```

```
    elif dist < emergency_dist:
```

```
        #print("Speed 0")
```

```
        speed = 0
```

При условии нахождения ведомого робота между двумя дистанциями выполняется сравнение предыдущей и текущей дистанции, таким образом выполняется поддержание скорости равной ведомому.

```
    elif dist < accel_dist and dist > break_dist:
```

```
        if prev_dist> dist:
```

```
            speed -= 1
```

```
        else:
```

```
            speed += 1
```

```
    if speed < 0:
```

```
        return 0
```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						26
Изм.	Лист	№ докум.	Подп.	Дата		

```
if speed > 40:
```

```
    speed = 40
```

```
return speed
```

Перейдем к обзору программной части ведущего робота под управлением системы с GPS.

Для работы с датчиками GPS нам понадобится подключить атрибут GPS из модуля controller. А также для вычисления векторов модуль math:

```
from controller import GPS
```

```
from vehicle import Driver
```

```
import math
```

В основном цикле выполняется проверка управляется ли грузовик вручную, моделирования ручного управления реализуется через клавиатуру. Включение автопилота происходит сменой флага в переменной. Если же включен автопилот, выполняется считывание показаний приемника GPS, и передача координат в метод обработки.

Метод принимает координаты передней и задней части грузовика, а также координату к которой необходимо проследовать.

```
def calc_angle(bx, by, ax, ay , tgtx, tgty):
```

```
    vectorA = [] # Список для вектора текущего направления
```

```
    vectorB = [] # Список для вектора целевого направления
```

```
    A = bx - ax
```

```
    B = by - ay
```

```
    vectorA.append(A)
```

```
    vectorA.append(B)
```

```
    A = tgtx - ax
```

```
    B = tgty - ay
```

```
    vectorB.append(A)
```

```
    vectorB.append(B)
```

Изм.	Лист	№ докум.	Подп.	Дата

Проводим проверку высчитывая скалярное произведение со смещенным вектором целевого направления, определяя таким образом сторону движения и в дальнейшем умножаем его знак на возвращаемый результат.

```

scalar_test = vectorA[0] * vectorB[1] + vectorA[1] * -vectorB[0]

scalar_prod = vectorA[0] * vectorB[0] + vectorA[1] * vectorB[1]

#print(scalar_prod)

lengthA = math.sqrt(math.pow(vectorA[0], 2) + math.pow(vectorA[1], 2))

lengthB = math.sqrt((math.pow(vectorB[0], 2) + math.pow(vectorB[1], 2)))

В переменную angle вносим результат вычисления угла между векторами

angle = scalar_prod / (lengthA * lengthB)

#print(math.cos(angle))

return math.degrees(math.acos(angle))*sign(scalar_test)

```

Также в основном цикле находится программная часть, включающаяся при обнаружении препятствия или другого автомобиля, и обеспечивающая регулирование скорости по радару, используя тот же метод что и у ведомого робота - speed_calibrating. При остановке включается счетчик, который по истечению 10 секунд, выводит предупреждающее сообщение и отключает автопилот.

```

if driver.getCurrentSpeed() == 0:

    time.sleep(1)

    sec += 1

    if sec == 10:

        sec = 0

        set_autopilot(False)

        print("Внимание неподвижное препятствие!\nПожалуйста, примите
управление.")

```

Перейдем к рассмотрению реализации программной части ведущего робота под управлением автопилота. Как и во всех предыдущих случаях изначально мы проводим подключение библиотек и инициализацию датчиков – камеры и радара.

Соответственно в данной программной части используется те же алгоритмы, что и в системе с GPS для проверки включения автопилота и регулирования скорости с предотвращением столкновения. Однако, основной принцип реализации управления другой. При условии, что автопилот включен в главном цикле происходит получение изображения с камеры и вызывается метод распознавания и определения угла полосы.

```
if autopilot:
```

```
    camera_image = camera.getImage()
```

```
    yellow_line_angle= filter_angle(process_camera_image(camera_image))#Обработка
```

```
    изображения
```

Метод принял на вход картинку дороги, обработал её и вернул рассчитанный угол в программный фильтр для устранения помех. При отсутствии на изображении дорожной разметки, фильтр вернет значение UNKNOWN, и будет выведено предупреждение о потере полосы и скорость начнет постепенно снижаться. Если же разметка обнаружена, отфильтрованный угол передается в ПИД регулятор.

```
        steer = 0
```

```
        if (yellow_line_angle != UNKNOWN ):
```

```
            #print(yellow_line_angle)
```

```
            line_following_steering = applyPID(yellow_line_angle)#Передача значения  
угла в ПИД
```

```
            steer = line_following_steering
```

```
            #steer *= 100
```

```
            #print(steer)
```

```
        else:
```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						29
Изм.	Лист	№ докум.	Подп.	Дата		

```
PID_need_reset = True
```

```
speed - = 3
```

```
print("*****")
```

```
print("Полоса потеряна!\nПримите управление.")#Вывод предупреждения
```

Подробнее рассмотрим основные функции обработки входного изображения. Первым преобразование осуществляет метод process_camera_image.

```
def process_camera_image(image):
```

```
    num_pixels = camera_height * camera_width
```

```
    REF = [95, 187, 203]# задаем целевой цвет пикселя, в соответствии с цветом разметки
```

```
    sumx = 0
```

```
    pixel_count = 0
```

```
    pixel = image
```

```
    x = 0
```

```
    i= 0
```

В цикле осуществляется проверка цвета каждого пикселя, если он отличается незначительно, то производится суммирование его номера в строке и подсчет количества [3]. Когда нужный цвета не обнаружены, метод возвращает значение UNKNOWN.

```
    while x < num_pixels:
```

```
        if (color_diff(pixel, REF, i) < 30):
```

```
            sumx += x % camera_width
```

```
            #sumx += x
```

```
            print(sumx)
```

```
            pixel_count += 1
```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						30
Изм.	Лист	№ докум.	Подп.	Дата		


```

        x += 1

        #print(pixel)

        i += 4

    if pixel_count == 0:

        return UNKNOWN

```

Обработав все пиксели, вычисляем их смещение на картинке, проводим нормирование для получения угла и вычитание для определения стороны поворота.

```

    return (sumx / pixel_count/ camera_width - 0.5 )

```

Функция filter_angle, усредняет полученные после вычисления значения угла. Размер фильтра FILTER_SIZE равен 3. Сначала производятся проверки был ли это первый вызов метода и пришло ли действительное значение.

```

def filter_angle(new_value):

    first_call = True

    old_value = []

    global UNKNOWN

    if (first_call or new_value == UNKNOWN ):

        first_call = False

        for i in range (FILTER_SIZE):

            old_value.append(0.0)

    else:

        for i in range((FILTER_SIZE - 1)):

            old_value[i] = old_value[i + 1]

    if (new_value == UNKNOWN):

        return UNKNOWN

    else:

        old_value[FILTER_SIZE - 1] = new_value

```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						31
Изм.	Лист	№ докум.	Подп.	Дата		

```

summ = 0.0

for i in range (FILTER_SIZE):

    summ += old_value[i]

return (summ / FILTER_SIZE)

```

При условии прохождения всех проверок происходит возврат усредненного значения угла.

Финальная стадия преобразование картинка в угол, который подается для установки колес происходит в ПИД регуляторе.

```

def applyPID(yellow_line_angle):

    oldValue = 0.0

    integral = 0.0

    global PID_need_reset

```

При условии первого вхождения производим сброс

```

if (PID_need_reset):

    oldValue = yellow_line_angle

    integral = 0.0

    PID_need_reset = False

if ( sign(yellow_line_angle) != sign(oldValue) ): #Компенсация эффекта насыщения

    integral = 0.0

```

Производим вычисления дифференциальной части:

```
diff = yellow_line_angle - oldValue
```

В интегральной компоненте проводим накопления при этом проверяя граничные значения:

```

if (integral < 30 and integral > -30):

    integral += yellow_line_angle

    oldValue = yellow_line_angle

```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						32
Изм.	Лист	№ докум.	Подп.	Дата		

Выполняем итоговое вычисление, умножая каждую составляющую на её коэффициент, а затем суммируя.

```
return KP * yellow_line_angle + KI * integral + KD * diff
```

5. Тестирование системы

5.1. Описание методики тестирования

Тестирование автомобилей и роботов на автоматическом управление делиться на три основные методики:

Тестирование алгоритмов и моделирование в симуляторе

Тестирование на закрытых площадках

Тестирование на дорогах общего пользования

Для разработанной системы будет использоваться тестирование в виртуальной среде. Тестирование в симуляторе позволит воспроизвести основные сценарии использования и проследить поведение системы в экстренных ситуациях.

Тестирование было разделено на несколько этапов:

- 1) Проверка работы системы ведомого робота
- 2) Проверка работы ведущего робота с системой GPS
- 3) Проверка работы ведущего робота с системой автопилота

5.2. Проверка работы системы ведомого робота

Тестирование системы реализации движения ведомого робота будет проводиться на двух сформированных моделях грузовиков в городском окружении – рисунок 9.

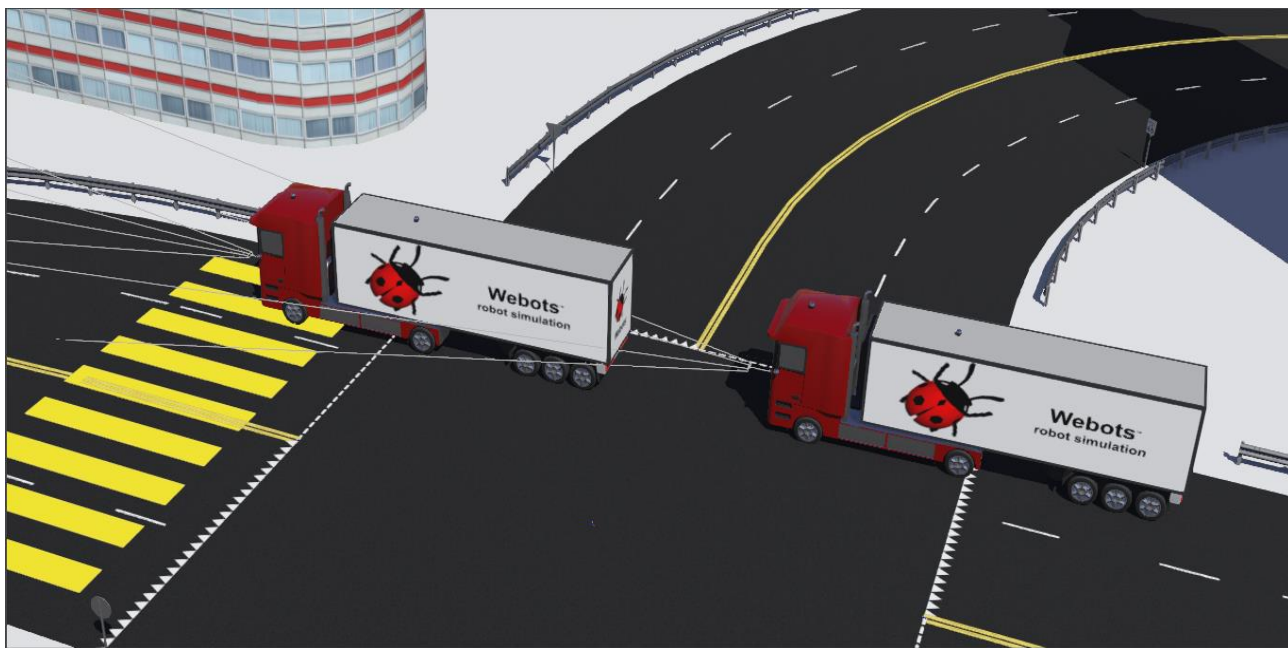


Рисунок 9 – Тестирование работы системы ведомого робота.

Белыми линиями обозначена область видимости радара.

Ведущая модель будет находится под ручным управлением, для успешного прохождения теста, ведомая модель должна корректировать угол поворота в соответствии с отклонение впереди идущего робота. Также она должна выполнять разгон, торможение и поддержание скорости ориентируясь на ведущую модель.

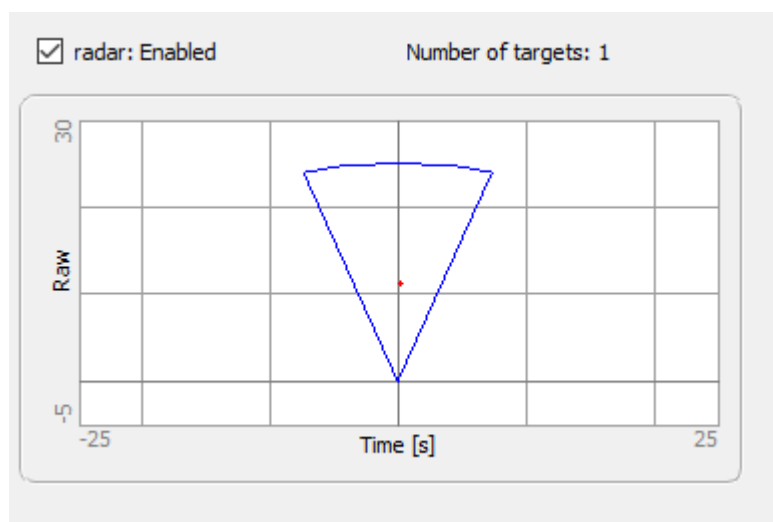


Рисунок 10 – Показание с датчика радара.

В ходе тестирования система ведомого автомобиля показала соответствие установленным требованиям.

5.3. Проверка работы системы ведущего робота с системой GPS

Тестирование робота будет выполняться с использованием одного робота. Также выбираем созданную модель и городское окружения. Для успешного прохождения тестирования модель должна проследовать до заданной координаты и остановиться. В качестве такой заданной точки установим на дороге бочку и маршрута выберем её координату рисунок 11.



Рисунок 11 – Итоговая точка движения.

Затем установим модель грузовика на повороте дороги – рисунок 12.

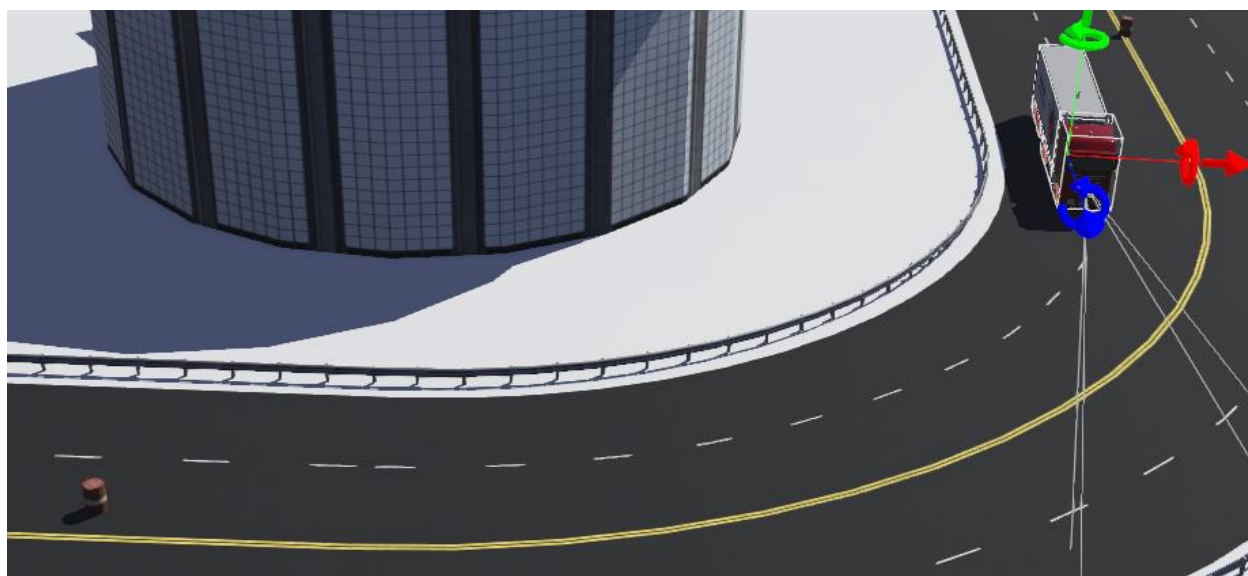


Рисунок 12 – Установка модели для формирования необходимых условий тестирования.

Робот корректно выполнил поворот и остановился, в консоль было выведено сообщение о достижении точки маршрута – рисунок 13.

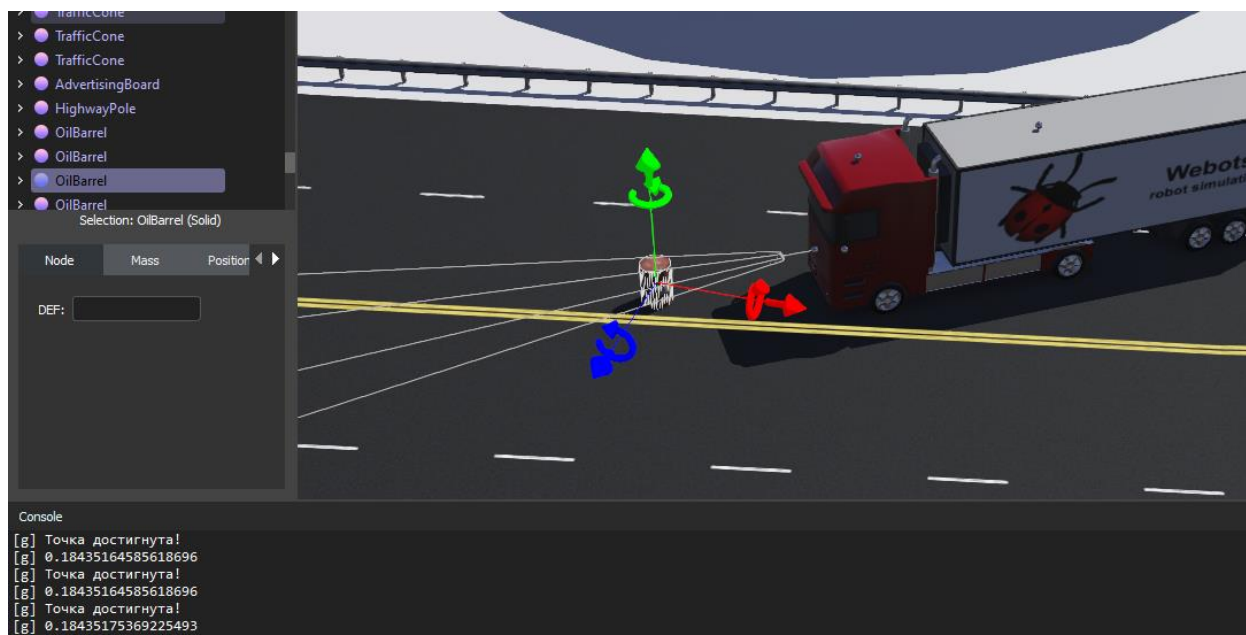


Рисунок 13 – Успешное выполнение теста системы с GPS.

5.4. Проверка работы системы ведущего робота с системой автопилота.

Финальным этапом тестирования разрабатываемой системы является проверка работы ведущего робота под управлением автопилота. Критерием успешного завершения проверки работоспособности этой системы считается движение грузовика в заданной полосе и предотвращение столкновения с объектом.

Сформируем необходимое окружения для выполнения тестирования. Установим грузовик на стартовой точке перед поворотом. После поворота установим неподвижный автомобиль – рисунок 14.



Рисунок 14 – Формирование сцены перед тестированием.

Запустив симуляцию, робот движется по полосе разметки, обнаружив приближение к неподвижному объекту он останавливается. Затем пользователю выводится сообщение с просьбой принять управление – рисунок 15.

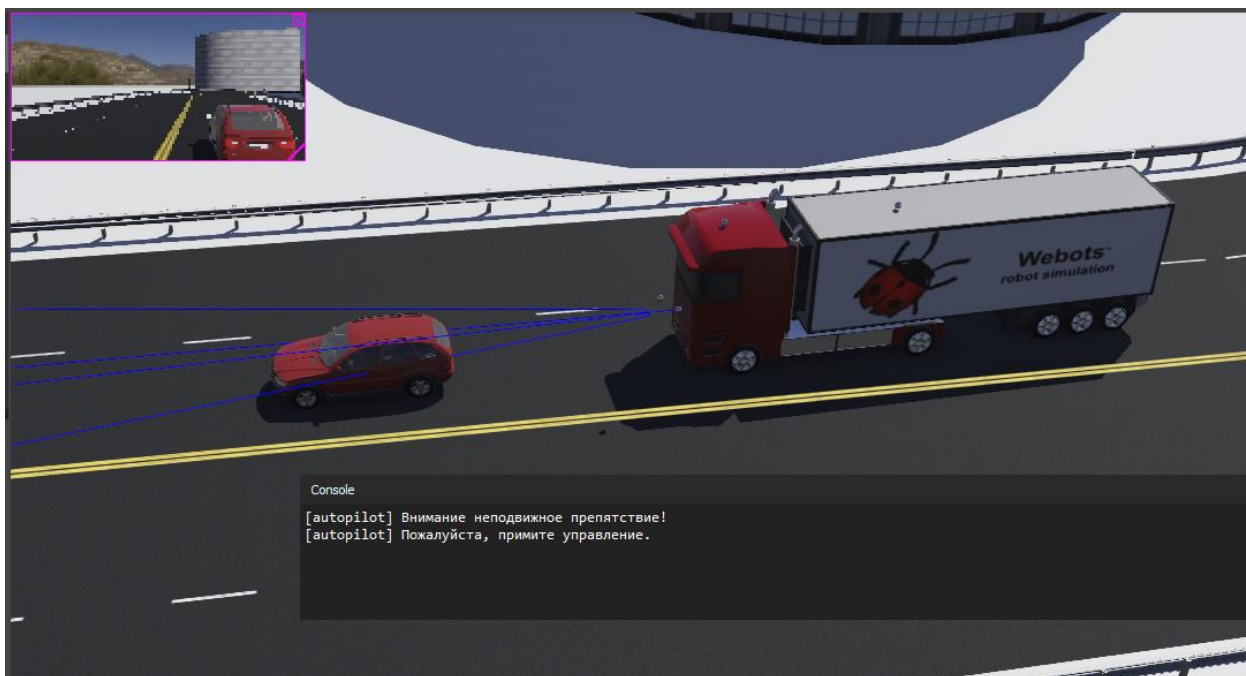


Рисунок 15 – Результат симуляции робота с автопилотом.

Изм.	Лист	№ докум.	Подп.	Дата

ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)

Лист

38

Из результатов проведенного для каждой из систем реализаций движения тестирования, можно сделать вывод, что система работает в соответствие с установленными требованиями.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	<i>Лист</i>
						39
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>		

Заключение

Итогом выполнения выпускной квалификационной работы была разработана и программно реализована система организации движения роботов в колонне. Для разработки данной программной системы был использован язык Python и симулятор Webots для отладки и проверки.

Созданная система выполняет соответствующие функциональные требования для организации движения роботов в колонне, что было подтверждено выполненным тестированием. Хотя реализация была выполнена на модели грузового транспорта, она так же может применяться и на различных промышленных роботах доставщиках.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	<i>Лист</i>
						40
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>		

Список литературы

1. Карпов В.Э. ПИД-управление в нестрогом изложении. [Электронный ресурс]. – Москва, 2012. – Режим доступа:
http://robofob.ru/materials/articles/pages/Karpov_mobline1.pdf
2. Доусон М. Програмируем на Python. – СПб.: Питер, 2014. – 416 с.
<http://бэкдор.рф/wp-content/uploads/Доусон-М.-Програмируем-на-Python-2014.pdf>
3. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
http://iro23.ru/sites/default/files/mark_lutc_programmirovanie_na_python.pdf
4. Roland Siegwart, Illah Reza Nourbakhsh, Davide Scaramuzz: Introduction to Autonomous Mobile Robots. The Mit Press 2004.
<https://pdfs.semanticscholar.org/4a37/fe05d825ae2554be2c0c90a19a39fe51c26b.pdf>
5. Knight, Will (22 October 2013). "The Future of Self-driving Cars". MIT Technology Review. Retrieved 22 July 2016.
<https://www.technologyreview.com/2013/10/22/175716/driverless-cars-are-further-away-than-you-think/>

					ВКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						41
Изм.	Лист	№ докум.	Подп.	Дата		

Приложение

Autopilot.py

```
import math

from controller import Robot
from controller import Keyboard

from vehicle import Driver

from controller import Radar
from controller import Camera

from controller import RadarTarget

from controller import Keyboard

import time


TIME_STEP = 50

KP = 0.25

KI = 0.006

KD = 2

FILTER_SIZE = 3

UNKNOWN = 99999.99


speed = 0

manual_steering = 0

steering_angle = 0

autopilot = True

sec = 0

azim = [0.0]

dist = [0.0]

prev_dist = 0
```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						42
Изм.	Лист	№ докум.	Подп.	Дата		

```

#Camera

camera_width = -1

camera_height = -1

camera_fov = -1.0

PID_need_reset = False


driver = Driver()

camera = driver.getCamera('camera')

camera.enable(TIME_STEP)

camera_width = camera.getWidth()

camera_height = camera.getHeight()

camera_fov = camera.getFov()

keyboard = Keyboard()

keyboard.enable(TIME_STEP)

radar = driver.getRadar('radar')

radar.enable(TIME_STEP)


print(camera_width)

def set_speed(kmh):

    # max speed

    if (kmh > 40):

        kmh = 40

    speed = kmh

    driver.setCruisingSpeed(kmh)

```

					БКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						43
Изм.	Лист	№ докум.	Подп.	Дата		

```

def sign(num):
    return -1 if num < 0 else 1

def color_diff(a, b, counter): #python
    diff = 0
    for i in range(3):
        d = a[counter + i] - b[i]
        diff += d if d > 0 else -d

    return diff

def process_camera_image(image):
    num_pixels = camera_height * camera_width
    REF = [95, 187, 203]
    sumx = 0
    pixel_count = 0

    pixel = image
    x = 0
    i= 0
    while x < num_pixels:
        if (color_diff(pixel, REF, i) < 30):
            sumx += x % camera_width
            #sumx += x
            print(sumx)
            pixel_count += 1

```

					БКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						44
Изм.	Лист	№ докум.	Подп.	Дата		

```

        x += 1

        i += 4

    if pixel_count == 0:

        return UNKNOWN

    return (sumx / pixel_count / camera_width - 0.5)

def filter_angle(new_value):
    first_call = True
    old_value = []
    global UNKNOWN
    if (first_call or new_value == UNKNOWN ):
        first_call = False
        for i in range (FILTER_SIZE):
            old_value.append(0.0)
    else:
        for i in range((FILTER_SIZE - 1)):
            old_value[i] = old_value[i + 1]
    if (new_value == UNKNOWN):
        return UNKNOWN
    else:
        old_value[FILTER_SIZE - 1] = new_value
        summ = 0.0
        for i in range (FILTER_SIZE):
            summ += old_value[i]
        return (summ / FILTER_SIZE)

```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						45
Изм.	Лист	№ докум.	Подп.	Дата		

```

def applyPID(yellow_line_angle):
    oldValue = 0.0
    integral = 0.0
    global PID_need_reset
    if (PID_need_reset):
        oldValue = yellow_line_angle
        integral = 0.0
        PID_need_reset = False

    if ( sign(yellow_line_angle) != sign(oldValue) ):
        integral = 0.0

    diff = yellow_line_angle - oldValue

    if (integral < 30 and integral > -30):
        integral += yellow_line_angle

    oldValue = yellow_line_angle
    return KP * yellow_line_angle + KI * integral + KD * diff

def set_autopilot(onoff):
    global autopilot
    if (autopilot == onoff):
        return
    autopilot = onoff

```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						46
Изм.	Лист	№ докум.	Подп.	Дата		


```

def speed_calibrating(dist, prev_dist, speed):
    accel_dist = 16
    break_dist = 15
    emergency_dist = 12
    if dist < break_dist:
        if dist > 15:
            print("Speed -2")
            speed-=2
        else:
            speed-=3
    elif dist > accel_dist:
        speed += 2
    elif dist < emergency_dist:
        speed = 0
    elif dist < accel_dist and dist > break_dist:
        if prev_dist> dist:
            speed -= 2
            print("Speed -1")
        else:
            speed += 1
    print("Return Speed", speed)
    if speed < 0:
        return 0
    if speed > 40:
        speed = 40

```

					БКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						47
Изм.	Лист	№ докум.	Подп.	Дата		

```
return speed
```

```
def change_manual_steer_angle(inc):  
    set_autopilot(False)  
    global manual_steering  
    new_manual_steering = manual_steering + inc  
    if (new_manual_steering <= 25 and new_manual_steering >= -25):  
        manual_steering = new_manual_steering  
        driver.setSteeringAngle(manual_steering * 0.02)  
    if (manual_steering == 0):  
        print("going straight")  
    else:  
        print("turning", manual_steering)
```

```
def check_keyboard():  
    key = keyboard.getKey()  
  
    if (key == Keyboard.UP):  
        global speed  
        speed = speed + 2  
        set_speed(speed)  
  
    if (key == Keyboard.DOWN):  
        speed = speed - 2  
        set_speed(speed)
```

					БКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						48
Изм.	Лист	№ докум.	Подп.	Дата		

```

    if (key == Keyboard.RIGHT):
        change_manual_steer_angle(+1)

    if (key == Keyboard.LEFT):
        change_manual_steer_angle(-1)

    if (key == Keyboard.CONTROL+ord('A')):
        set_autopilot(True)

set_speed(40)

# Main loop:
driver.setThrottle(0.3)
print(driver.getThrottle())

# - perform simulation steps until Webots is stopping the controller
while (driver.step() != -1):
    check_keyboard()

    if autopilot:
        camera_image = camera.getImage()

        yellow_line_angle = filter_angle(process_camera_image(camera_image))

        steer = 0

        if (yellow_line_angle != UNKNOWN ):
            #print(yellow_line_angle)

            line_following_steering = applyPID(yellow_line_angle)

            steer = line_following_steering

```

					БКР-ИГТУ-09.03.01-(16-B-2)-007-2020(ПЗ)	Лист
						49
Изм.	Лист	№ докум.	Подп.	Дата		

```

else:

    PID_need_reset = True

    print("*****")

    print("Полоса потеряна!\nПримите управление.")

    speed -= 1

driver.setSteeringAngle(steer)

if (radar.getNumberOfTargets() > 0 ):

    for i in range(radar.getNumberOfTargets()):      # получение данных
с радара и занесение их в списки

        azim[i] = radar.getTargets()[i].azimuth

        dist[i] = radar.getTargets()[i].distance

    speed = speed_calibrating(dist[0], prev_dist, speed)

    print(dist[0])

    set_speed(speed)

else:

    set_speed(40)

if driver.getCurrentSpeed() == 0:

    time.sleep(1)

    sec += 1

    if sec == 10:

        sec = 0

        set_autopilot(False)

        print("Внимание неподвижное препятствие!\nПожалуйста, примите
управление.")

    prev_dist = dist[0]

# Enter here functions to send actuator commands, like:

```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						50
Изм.	Лист	№ докум.	Подп.	Дата		

Pass

Radar.py

```
from controller import Robot
from controller import Keyboard
from vehicle import Driver
from controller import Radar
from controller import RadarTarget
from controller import Device

TIME_STEP = 64

azim = [0.0]
dist = [0.0]
prev_dist = 0
target = False
speed = 0

driver = Driver()
radar = driver.getRadar('radar')
radar.enable(TIME_STEP)

def speed_calibrating(dist, prev_dist, speed):
    accel_dist = 16
    break_dist = 15
    emergency_dist = 7
    if dist < break_dist:
        print("Breaking")
        if dist > 9.5:
```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						51
Изм.	Лист	№ докум.	Подп.	Дата		

```

        speed-=2
    else:

        speed-=3
    elif dist > accel_dist:

        print("Accelerating")
        speed += 2
    elif dist < emergency_dist:

        speed = 0
    elif dist < accel_dist and dist > break_dist:

        if prev_dist> dist:
            speed -= 2
        else:
            speed += 1

    if speed < 0:
        return 0
    if speed > 40:
        speed = 40
    return speed

def set_speed(kmh):
    # max speed
    if (kmh > 40):

```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						52
Изм.	Лист	№ докум.	Подп.	Дата		

```

        kmh = 40

    speed = kmh

    #print("setting speed to", kmh,"km/h")

    driver.setCruisingSpeed(kmh)

def change_manual_steer_angle(angle):
    new_manual_steering = angle
    driver.setSteeringAngle(new_manual_steering)

driver.setThrottle(0.1)

while (driver.step() != -1):

    if (radar.getNumberOfTargets() > 0 ):

        for i in range(radar.getNumberOfTargets()): # получение данных с
радары и занесение их в списки
            azim[i] = radar.getTargets()[i].azimuth
            dist[i] = radar.getTargets()[i].distance

        speed = speed_calibrating(dist[0], prev_dist, speed)
        set_speed(speed)
        change_manual_steer_angle(azim[0])
        prev_dist = dist[0]
        pass

```

GPS.py

```
from controller import GPS
from vehicle import Driver
import math
```

```
speed = 0
manual_steering = 0
steering_angle = 0
autopilot = True
sec = 0
azim = [0.0]
dist = [0.0]
prev_dist = 0
```

```
driver = Driver()
gps = GPS('gps')
gps2 = GPS('gps2')
```

```
gps.enable(64)
gps2.enable(64)
```

```
ay_prev = 0
ax_prev = 0
```

```
tgtx = 54.77
tgty = 103.169
```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						54
Изм.	Лист	№ докум.	Подп.	Дата		


```

def sign(num):
    return -1 if num < 0 else 1

def set_speed(kmh):

    if (kmh > 40):
        kmh = 40

    speed = kmh

    driver.setCruisingSpeed(kmh)

def change_manual_steer_angle(angle):
    manual_steering = 0
    new_manual_steering = angle
    if (new_manual_steering >= 25 ):
        manual_steering = 25
        driver.setSteeringAngle(manual_steering * 0.014)
    if (new_manual_steering <= -25):
        manual_steering = -25
        driver.setSteeringAngle(manual_steering * 0.014)
    else:
        driver.setSteeringAngle(new_manual_steering * 0.014)

```

					БКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						55
Изм.	Лист	№ докум.	Подп.	Дата		

```

def dist(bx, by, ax, ay):

    dist = math.sqrt(math.pow((bx - ax), 2) + math.pow((by - ay), 2))

    if (dist < 1):

        set_speed(0)

        print("Точка достигнута!")

def check_keyboard():

    key = keyboard.getKey()

    if (key == Keyboard.UP):

        global speed

        speed = speed + 2

        set_speed(speed)

    if (key == Keyboard.DOWN):

        speed = speed - 2

        set_speed(speed)

    if (key == Keyboard.RIGHT):

        change_manual_steer_angle(+1)

    if (key == Keyboard.LEFT):

        change_manual_steer_angle(-1)

    if (key == Keyboard.CONTROL+ord('A')):

        set_autopilot(True)

```

```

def calc_angle(bx, by, ax, ay , tgtx, tgty):

    vectorA = []
    vectorB = []

    A = bx - ax
    B = by - ay

    vectorA.append(A)
    vectorA.append(B)

    A = tgtx - ax
    B = tgty - ay
    vectorB.append(A)
    vectorB.append(B)

    scalar_test = vectorA[0] * vectorB[1] + vectorA[1] * -vectorB[0]
    scalar_prod = vectorA[0] * vectorB[0] + vectorA[1] * vectorB[1]

    lengthA = math.sqrt(math.pow(vectorA[0], 2) + math.pow(vectorA[1], 2))
    lengthB = math.sqrt((math.pow(vectorB[0], 2) + math.pow(vectorB[1], 2)))
    if lengthA * lengthB == 0:
        angle = 1
    else:
        angle = scalar_prod / (lengthA * lengthB)

    return math.degrees(math.acos(angle))*sign(scalar_test)

```

					БКР-НГТУ-09.03.01-(16-B-2)-007-2020(ПЗ)	Лист
						57
Изм.	Лист	№ докум.	Подп.	Дата		

```

def speed_calibrating(dist, prev_dist, speed):
    accel_dist = 16
    break_dist = 15
    emergency_dist = 12
    if dist < break_dist:
        if dist > 15:

            speed-=2
        else:
            speed-=3
    elif dist > accel_dist:

        speed += 2
    elif dist < emergency_dist:

        speed = 0
    elif dist < accel_dist and dist > break_dist:
        #print("Complete")
        if prev_dist> dist:
            speed -= 2
            print("Speed -1")
        else:
            speed += 1
    print("Return Speed", speed)
    if speed < 0:
        return 0
    if speed > 40:

```

					БКР-ИГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						58
Изм.	Лист	№ докум.	Подп.	Дата		

```

        speed = 40

    return speed

driver.setCruisingSpeed(5)

while driver.step() != -1:

    check_keyboard()

    if autopilot:

        ay_prev = gps2.getValues()[2]
        ax_prev = gps2.getValues()[0]

        angle = calc_angle(gps.getValues()[0], gps.getValues()[2], ax_prev,
ay_prev, tgtx, tgty)

        dist(gps.getValues()[0], gps.getValues()[2], tgtx, tgty)

        if (angle != -0):

            print(angle)

            change_manual_steer_angle(angle)

            if (radar.getNumberOfTargets() > 0 ):

                for i in range(radar.getNumberOfTargets()):      # получение данных
с радара и занесение их в списки

                    azim[i] = radar.getTargets()[i].azimuth

                    dist[i] = radar.getTargets()[i].distance

```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						59
Изм.	Лист	№ докум.	Подп.	Дата		

```

        speed = speed_calibrating(dist[0], prev_dist, speed)

        print(dist[0])

        set_speed(speed)
    else:

        set_speed(40)

    if driver.getCurrentSpeed() == 0:

        time.sleep(1)

        sec += 1

        if sec == 10:

            sec = 0

            set_autopilot(False)

            print("Внимание неподвижное препятствие!\nПожалуйста, примите
управление.")

            prev_dist = dist[0]

pass

```

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020(ПЗ)	Лист
						60
Изм.	Лист	№ докум.	Подп.	Дата		