МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка программных компонентов системы предотвращения столкновения для крупногабаритной техники

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-566.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ П.Г. Верман

Автор работы,
студент группы КЭ-403
_____ А.П. Батюшева

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-403
Батюшевой Анастасии Павловне,
обучающейся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка программных компонентов системы предотвращения столкновения для крупногабаритной техники.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. ГОСТ Р ИСО 11898-1-2015 Транспорт дорожный. Местная контроллерная сеть (CAN). Часть 1. Канальный уровень и передача сигналов Официальное издание. // М.: Стандартинформ, 2016. – 42 с.

3.2. Jetson Software Documentation. [Электронный ресурс] URL: https://docs.nvidia.com/jetson/tation (дата обращения: 29.01.2024 г.).

3.3. Kumar P., Narasimha Swamy S., Purohit G. Real-time, yolo-based intelligent surveillance and monitoring system using jetson tx2. // Data Analytics and Management: Proceedings of ICDAM 2019. – 461–471 pp.

3.4. Tkinter documentation. [Электронный ресурс] URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 29.01.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить анализ предметной области и провести обзор аналогов.

4.2. Спроектировать компоненты системы.

4.3. Реализовать компоненты.

4.4. Провести тестирование.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП П.Г. Верман

Задание принял к исполнению А.П. Батюшева

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

На данный момент, современный транспорт переживает существенные изменения, вызванные активным развитием автономных технологий. Внедрение беспилотных транспортных средств и автономных систем управления водителем становится все более распространенным явлением. Это создает потребность в эффективных системах предотвращения столкновений [1]. В связи с этим, разработка программных компонентов, способных взаимодействовать с автономными системами, становится ключевой задачей, при обеспечении согласованности в работе всего транспортного парка.

При этом, крупногабаритная техника, выполняющая работу в ограниченных пространствах, например, на сельскохозяйственных угодьях или на местах стройки, чаще сталкивается с высоким риском столкновений, и как следствие, повреждений техники, что в свою очередь влечет за собой увеличение сроков выполнения задач и увеличение финансовых затрат. Также работа в таких условиях не только угрожает безопасности оборудования, но и может представлять опасность как для операторов данной техники, так и для окружающей среды. Как следствие, разработка эффективных систем предотвращения столкновений становится неотъемлемой для предотвращения повреждений, что в конечном итоге приведет к сокращению операционных затрат и повышению производительности в сфере использования крупногабаритной техники.

Также стоит отметить, что задача разработки системы предотвращения столкновений подразделяется на разработку программной и аппаратной части. Целью данной работы является именно разработка программных компонентов, обеспечивающих корректную работу системы.

Постановка задачи

Целью выпускной квалификационной работы является разработка программных компонентов для системы автономного предотвращения

столкновения для крупногабаритной техники. Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области и провести обзор аналогов;
2) спроектировать компоненты системы;
3) реализовать компоненты;
4) провести тестирование.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 43 страницы, объем списка литературы – 29 источников.

В первой главе описывается предметная область, рассматриваются существующие на данный момент аналоги систем предотвращения столкновений и доступные технологии их реализации.

Вторая глава посвящена определению функциональных и нефункциональных требований к разрабатываемым компонентам системы, а также их проектированию.

В третьей главе описана реализация каждого из разрабатываемых компонентов системы.

Четвертая глава включает тестирование всех разрабатываемых компонентов.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Данная работа основана на современных технологических тенденциях в транспортной отрасли, где на данный момент происходит активное внедрение автономных транспортных средств и систем управления. В контексте крупногабаритной техники, такой как строительная и сельскохозяйственная, разработка программных компонентов для систем предотвращения столкновений становится одной из ключевых задач, по причине высокого риска столкновений.

Эта тема охватывает несколько важных аспектов. Основной из них – уменьшение аварий и повреждений крупногабаритной техники является важным моментом, решаемым разработкой эффективных систем предотвращения столкновений, основанных на сочетании современных датчиков и искусственного интеллекта.

Одним из основных направлений является также оптимизация операций крупногабаритной техники. Разработка программных компонентов направлена на снижение рисков простоев, повышение эффективности использования оборудования и, в конечном итоге, создание систем, способствующих экономическому эффекту от внедрения автономных решений в данной предметной области.

1.1. Обзор подобных систем

В ходе проведенного обзора аналогов были выявлены несколько систем со схожим функционалом.

SICK Safety Collision Prevention System

Система [2] представляет собой комплекс датчиков, включающих лазерные и радарные устройства, осуществляет активное мониторинг окружающей среды и обнаруживает препятствия в реальном времени. Интеграция искусственного интеллекта в алгоритмы обработки данных обеспечивает

адаптивность к изменяющимся условиям, повышая эффективность предотвращения столкновений. Иллюстрация принципа работы системы представлена на рисунке 1.

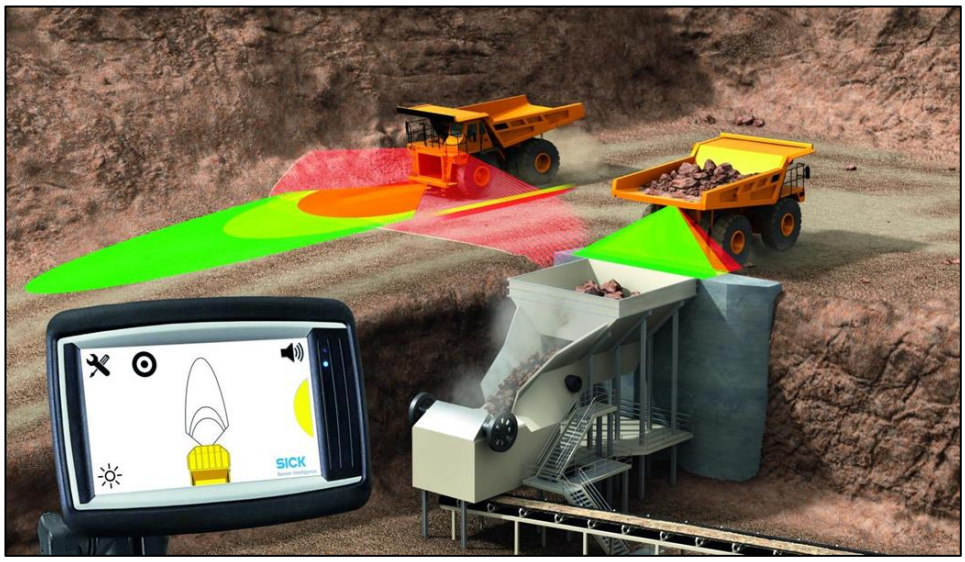

Рисунок 1 – Иллюстрация принципа работы
«SICK Safety Collision Prevention»

Caterpillar Cat Detect Safety System

Система [3] представляет собой комплексное решение для автоматизации контроля ситуации на строительных площадках, в качестве датчиков используются камеры и радары для обнаружения препятствий при работе специализированной техники. Интеллектуальные алгоритмы обработки изображений и данных с радаров позволяют классифицировать объекты, определять их наличие и траекторию их движения и выдавать предупреждения о присутствии объектов в поле движения машины. Автоматическое торможение в случае опасности также поддерживается искусственным интеллектом. Интерфейс программной части системы, использующийся оператором для контроля происходящей ситуации представлен на рисунке 2.

Рисунок 2 – Интерфейс системы «Caterpillar Cat Detect Safety»

Komatsu Intelligent Machine Control

Система [4] включает в себя GPS, камеры и устройство, сканирующее окружающее пространство с помощью лазерных лучей для создания трехмерной карты окружающей среды. Используя данные, собранные с этих устройств, алгоритмы искусственного интеллекта определяют препятствия и принимают решения о необходимых коррекциях маршрута и скорости, что в свою очередь, обеспечивает автоматическое предотвращение столкновений техники при работе и, как следствие, безопасное передвижение специализированной техники по строительной площадке. В качестве интерфейса программной части системы выступает приложение, которое позволяет оператору наиболее точно отслеживать происходящую ситуацию в режиме реального времени за счет обзора камер в 360 градусов. На рисунке 3 представлен графический пользовательский интерфейс для упрощения работы оператора системы.

Рисунок 3 – Интерфейс системы «Komatsu Intelligent Machine Control»

John Deere ActiveCollision Avoidance System

Система [5] использует передовые датчики и алгоритмы машинного обучения. Искусственный интеллект обеспечивает более точное распознавание объектов, а также адаптацию системы к различным условиям и типам препятствий. В случае возможного столкновения система может автоматически вмешиваться с управлением или торможением. Интерфейс программной части системы представлен на рисунке 4.

Рисунок 4 – Интерфейс системы «John Deere ActiveCollision Avoidance»

По итогам обзора аналогов был проведен сравнительный анализ описанных систем. В качестве общих выводов можно отметить, что каждая из систем использует сочетание определенных датчиков и искусственного интеллекта.

Таким образом, система «SICK Safety Collision Prevention» использует лазерные и радарные датчики, что дает ей возможность быстро реагировать и обнаруживать препятствия с высокой точностью в широком диапазоне погодных условий.

Система «Caterpillar Cat Detect Safety» в качестве датчиков использует камеры и радары, что помогает эффективно реализовать предупреждение пользователя о наличии препятствия и автоматическое торможение.

Система «Komatsu Intelligent Machine Control» использует датчики GPS, камеры и устройство, сканирующее пространство с помощью лазерных лучей, что дает возможность определения точного местоположения в режиме реального времени, автоматического управления и автоматического регулирования траектории и скорости движения.

Также стоит отметить, что система «John Deere ActiveCollision Avoidance» отличается автоматическим вмешательством, благодаря использованию искусственного интеллекта.

В результате анализа различных систем автономного предотвращения столкновений для крупногабаритной техники видно, что каждая из систем увеличивает эффективность использования техники благодаря использованию датчиков и искусственного интеллекта.

1.2. Обзор существующих технологий

В качестве основных существующих технологий реализации подобных систем, можно выделить сенсоры и датчики для контроля ситуации; компьютерное зрение для обработки информации, поступающей с датчиков; технологии связи, обеспечивающие обмен данными; операционные си-

стемы для встраиваемых устройств, необходимых для обеспечения автономности; а также многозадачность и параллельные вычисления для обеспечения высокой скорости работы.

Зачастую, в роли минимально необходимых датчиков выступают камеры. Так как при использовании алгоритмов компьютерного зрения они играют важнейшую роль в распознавании объектов. Однако наиболее качественный результат определения и детекции объектов достигается при использовании камер совместно с лидаром. Сам по себе лидар является радаром, основанным на свете, при помощи которого система узнает яркость и дальность цели.

Для обнаружения препятствий, анализа окружающей среды и принятия решений на основе визуальной информации. Необходимо использование компьютерного зрения. На данный момент наиболее эффективным средством реализации данной цели является библиотека OpenCV, которая представляет наиболее развитую библиотеку для работы с изображениями и видеоматериалами.

Машинное обучение и искусственный интеллект применяются для обработки данных, принятия решений и адаптации системы к изменяющимся условиям. Эти технологии улучшают точность обнаружения и классификации объектов, а также способствуют эффективной адаптации к различным сценариям. В данном контексте, в качестве наиболее проверенных технологий можно выделить предобученную нейронную сеть YOLO [6], основным принципом которой является наиболее эффективное по сочетанию скорости и качества определение объектов.

Беспроводные технологии связи обеспечивают эффективное взаимодействие между компонентами системы, включая обмен данными с внешними источниками. Операционные системы для встраиваемых устройств гарантируют стабильную работу системы при ограниченных вычислительных ресурсах, обеспечивая оптимальное функционирование.

Использование мобильных платформ, таких как Jetson Nano [7] и Raspberry Pi, добавляет высокую производительность и компактность в систему, что становится ключевым фактором в контексте автономных технических устройств.

Выводы по первой главе

Таким образом можно выделить три ключевые технологии: язык программирования Python, CAN-шина для обмена данными и YOLO в качестве нейросети.

Использование языка программирования Python в проекте предоставляет удобство и гибкость в разработке программных компонентов. Python позволяет легко интегрировать различные технологии, такие как компьютерное зрение, машинное обучение и взаимодействие с CAN-шиной.

CAN-шина, в свою очередь, выступает важным компонентом для обеспечения обмена информацией между камерами и узлом управления машины. Поэтому она играет ключевую роль в обеспечении согласованной работы системы.

Нейросеть YOLO обеспечивает точное обнаружение и классификацию объектов в режиме реального времени. Ее высокая эффективность существенно улучшает качество работы системы автономного предотвращения столкновения.

Таким образом, совместное воздействие данных средств реализации создает современное и эффективное решение и обеспечивает высокую точность, быструю передачу данных и адаптивность к различным сценариям движения.

2. ПРОЕКТИРОВАНИЕ

В данном разделе описано проектирование разрабатываемых компонентов. Основные функциональные и нефункциональные требования были выявлены на основе анализа предметной области. Также в данном разделе представлена архитектура системы.

2.1. Требования к системе

Требования к приложению

В ходе обзора существующих решений была выявлена необходимость в разработке приложения с интерфейсом для настройки параметров системы, обеспечивающего возможность их свободного редактирования пользователем. Исходя из этого, были выявлены следующие функциональные требования.

1. Приложение должно обеспечивать подключение к удаленному устройству по сети.
2. Приложение должно отображать видеопоток с удаленного устройства в режиме реального времени.
3. Приложение должно обеспечивать возможность настройки таких параметров как размер области определения объекта, выбор определяемых объектов.
4. Приложение должно обеспечивать обработку ошибок, возникающих в результате подключения, передачи данных и обработки видеопотока.

Также были выявлены следующие нефункциональные требования.

1. Приложение должно иметь удобный и интуитивно понятный интерфейс.
2. Приложение должно быть реализовано при помощи языка программирования Python.

Требования к компоненту обмена данных

Также в ходе обзора была выявлена необходимость реализации эффективного обмена данными между компонентами системы. Для этого было

принято решение использовать CAN-шину. При разработке программного компонента обозначены следующие требования.

1. Обеспечение эффективного обмена данными в режиме реального времени между камерами и узлом управления.
2. Обеспечение обработки данных с минимальными задержками.
3. Должна быть предусмотрена обработка ошибок, связанных с некорректными данными.

Требования к нейронной сети

Аналогично, в ходе анализа требований была выявлена необходимость использования алгоритмов машинного зрения, для распознавания объектов, находящихся в поле зрения камеры. При разработке компонента нейронной сети обозначены следующие требования.

1. Нейронная сеть должна обеспечивать определение объектов с минимальной задержкой.
2. Нейронная сеть должна обеспечивать эффективное определение объектов.
3. Нейронная сеть должна обеспечивать детекцию всех объектов, попадающих в зону видимости.
4. Нейронная сеть должна сохранять показатели эффективности при изменении погодных условий.

2.2. Архитектура системы

Архитектура всей системы представлена на рисунке 5, разрабатываемые программные компоненты обозначены пунктиром. В роли связующего компонента выступает приложение для настройки параметров. Пользователь сначала настраивает необходимые ему параметры подключения и детекции. После чего нажимает на кнопку «Подключиться», тем самым активируя процесс передачи данных. Далее, на вход системы поступает непрерывный видеопоток с IP камеры. Каждый кадр этого потока поступает в

нейронную сеть для детекции и классификации объектов. После чего, обработанный кадр отображается на экране пользователя.

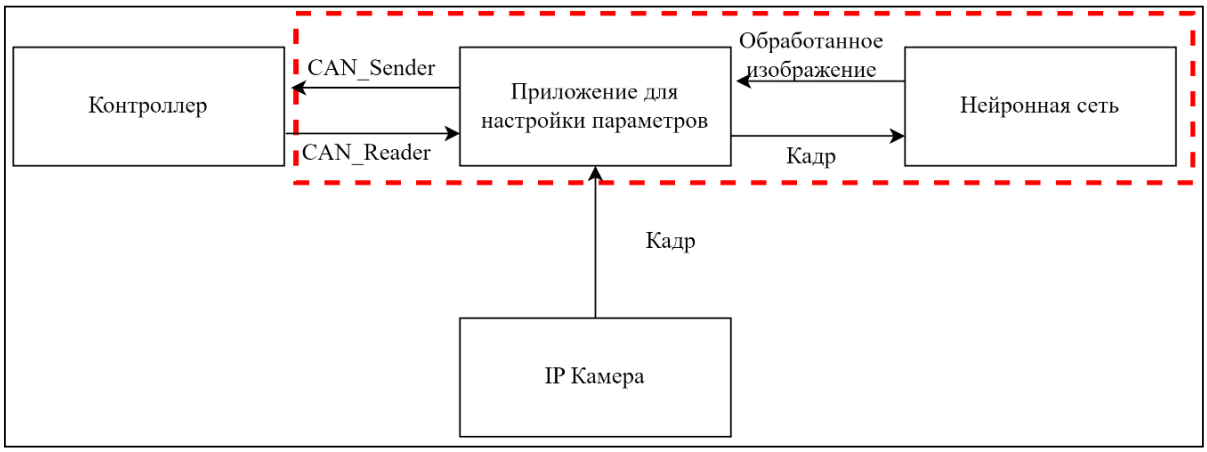

Рисунок 5 – Архитектура системы

Параллельно с этим происходит обмен данными с контроллером при помощи CAN шины, эти данные включают в себя информации о наличии объекта в области движения. Если в сообщении, поступающем на контроллер указано наличие объектов в зоне движения, то происходит остановка машины.

2.3. Проектирование приложения для настройки параметров

Согласно описанным выше требованиям, была разработана диаграмма вариантов использования графического интерфейса, представленная на рисунке 6. Актер «Пользователь» имеет доступ ко всему функционалу, он может подключиться к системе, используя изначально заданные параметры, либо после их изменения. В качестве изменения параметров подразумеваются действия по изменению ширины и высоты зоны, при попадании детектируемых объектов в которую происходит торможение, а также по изменению объектов детекции. Также пользователь может получить справку, в которой будет отражен смысл параметров и необходимость их изменения в зависимости от ситуации.

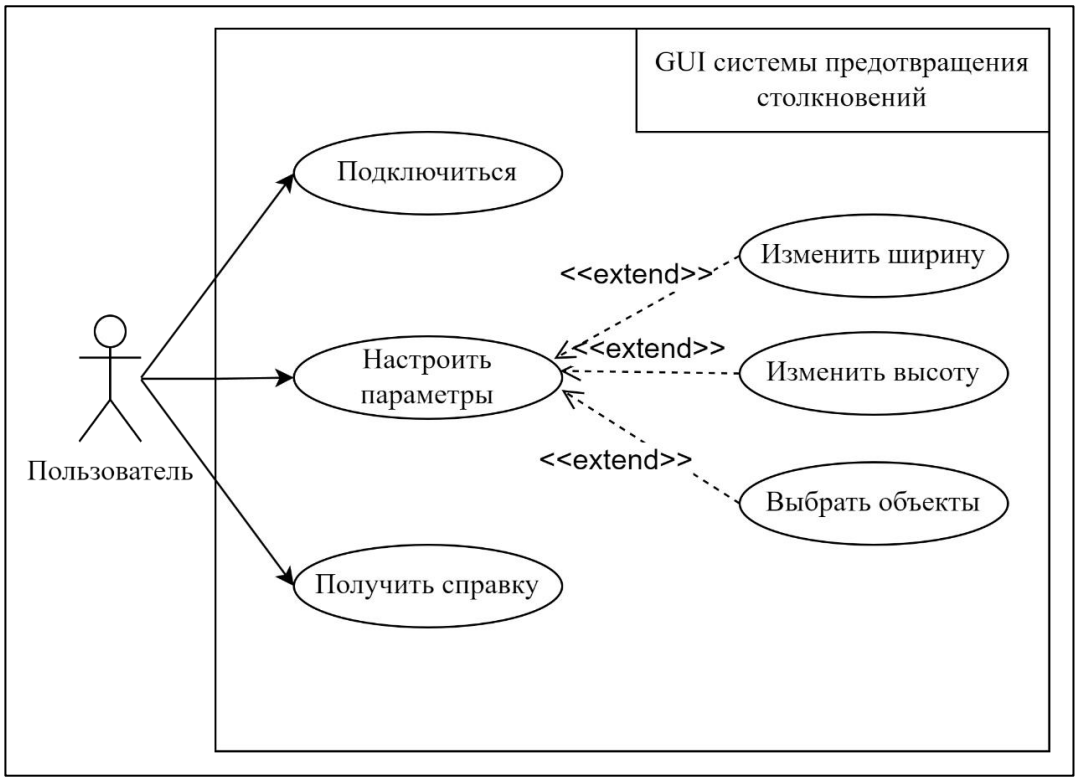

Рисунок 6 – Варианты использования приложения системы

На основе диаграммы вариантов использования, а также функциональных требований к компоненту приложения настройки параметров системы, была составлена диаграмма необходимых для эффективной работы компонентов, она представлена на рисунке 7.

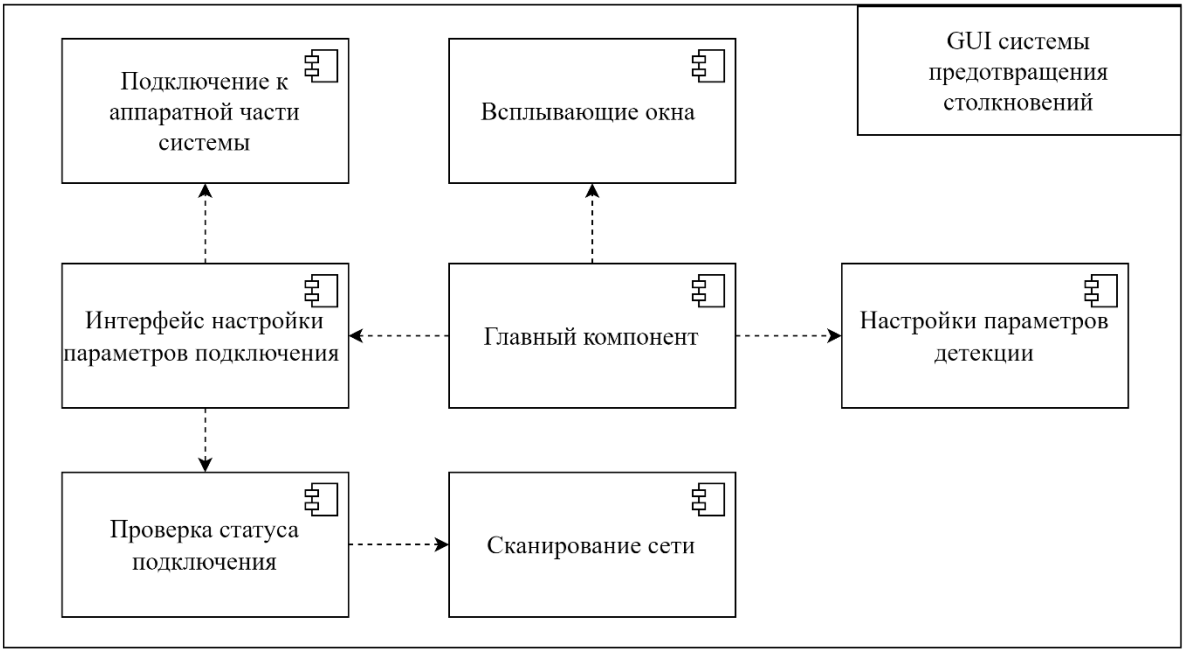

Рисунок 7 – Компоненты приложения для настройки параметров системы

Приложение для настройки параметров, которое является интерфейсом системы предотвращения столкновений, состоит из следующих компонентов:

1) «главный компонент» – основной модуль, в котором реализована логика программного компонента;

2) «интерфейс настройки параметров подключения» – модуль, полностью отвечающий за настройку параметров подключения;

3) «подключение к аппаратной части системы» – модуль, отвечающий за проверку корректности введенных параметров подключения, в случае успешности проверки, компонент связывается с аппаратной частью и начинается воспроизведение видеопотока;

4) «проверка статуса подключения» – модуль, отвечающий за отображение флага подключения;

5) «сканирование сети» – модуль, отвечающий за проверку наличия устройств в сети;

6) «интерфейс настройки параметров детекции» – модуль, отвечающий за изменение размера области движения и объектов детекции;

7) «всплывающие окна» – модуль, отвечающий за обработку ошибок.

На основе требований к компоненту был спроектирован макет интерфейса, который представлен на рисунке 8.

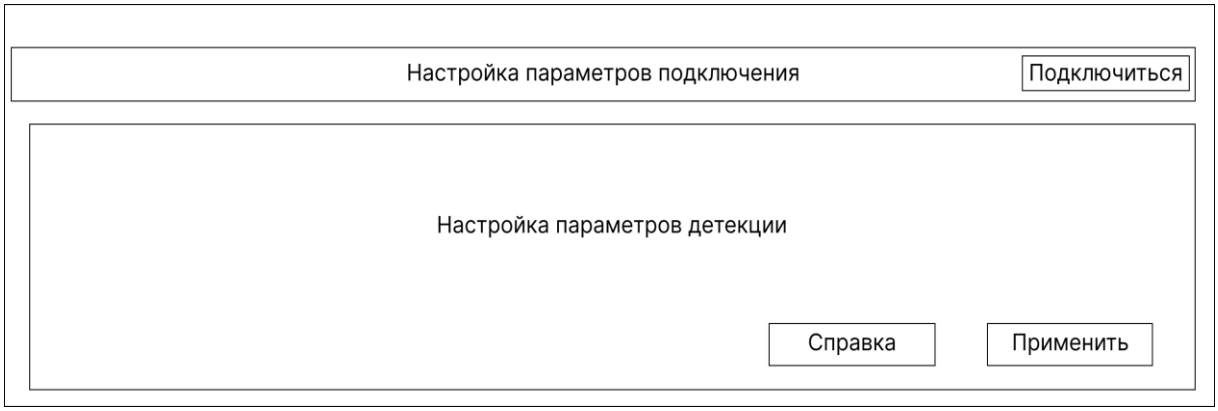

Рисунок 8 – Макет приложения

В данном макете предусмотрено окно настройки параметров подключения с кнопкой «Подключиться», после нажатия на которую открывается окно с видеопотоком для отображения работы программы.

Также в окне «Настройка параметров детекции» будет реализована возможность редактирования размера области реагирования, объектов детекции и необходимой точности. При нажатии на кнопку «Применить», будет происходить обновление настроек, при нажатии на кнопку «Справка» будет открываться окно с инструкцией по корректной настройке параметров.

Также для отображения ошибок при работе приложения предусмотрены всплывающие окна с текстом ошибки. Их интерфейс представлен на рисунке 9.

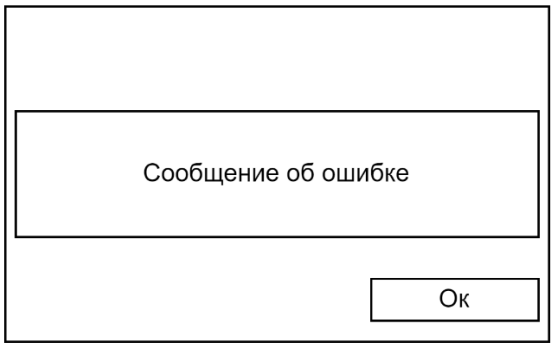

Рисунок 9 – Макет сообщения об ошибке

2.4. Проектирование компонента обмена данных

В процессе разработки программных компонентов для системы предотвращения столкновений крупногабаритной техники была выявлена необходимость в реализации эффективного обмена данными между компонентами системы [8]. Следовательно, были рассмотрены различные варианты реализации такого обмена.

Использование CAN шины в качестве основы компонента обмена данными дает возможность наиболее эффективной работы системы. Так как в протоколе CAN реализован механизм обнаружения ошибок, который в слу-

чае необходимости передает в сеть связанных устройств флаг ошибки, который аварийно завершает передачу текущего сообщения, после этого данное сообщение сбрасывается [9]. Таким образом в системе достигается непротиворечивость данных.

Так как данный компонент обеспечивает обмен информацией между камерами и узлом управления машиной, что играет одну из ключевых ролей в согласованной работе системы. было принято решение использовать CAN шину по причинам наиболее надежного и устойчивого метода обмена данными [10], более того, данная технология предназначена для использования в автомобильных условиях, что означает возможность работы в экстремальных температурных условиях и при воздействии вибраций [11].

2.5. Проектирование компонента нейронной сети

Согласно описанным выше требованиям к наличию и реализации нейронной сети, необходимо подобрать архитектуру для наиболее эффективной и быстрой детекции всех объектов, попадающих в поле зрения камеры. Для этого каждый кадр видеопотока должен проходить через сеть наименьшее количество раз.

Алгоритм YOLO (You Only Look Once) представляет собой инструмент для детекции объектов в реальном времени благодаря своей архитектуре, которая позволяет осуществлять детекцию и классификацию объектов на изображении за один проход через сеть [12], то есть параллельно. Такой эффект достигается благодаря разбиению изображения, либо каждого из кадров видеопотока на квадратную сетку. В каждой ячейке сетки YOLO делает несколько предсказаний наличия центра объекта и его границ. Кроме того, она оценивает процент достоверности этих предсказаний.

Параллельно с этим нейросеть предсказывает вероятность принадлежности объекта к нескольким различным классам для каждой из ячеек сетки, составляя карту вероятностей классов.

Далее предсказания рамок и классов объединяются для получения окончательного результата классификации и детекции объектов. Для удаления избыточных рамок используется метод не максимального подавления, который оставляет только рамки с наибольшей вероятностью достоверности [13]. Иллюстрация вышеописанной работы алгоритма представлена на рисунке 10.

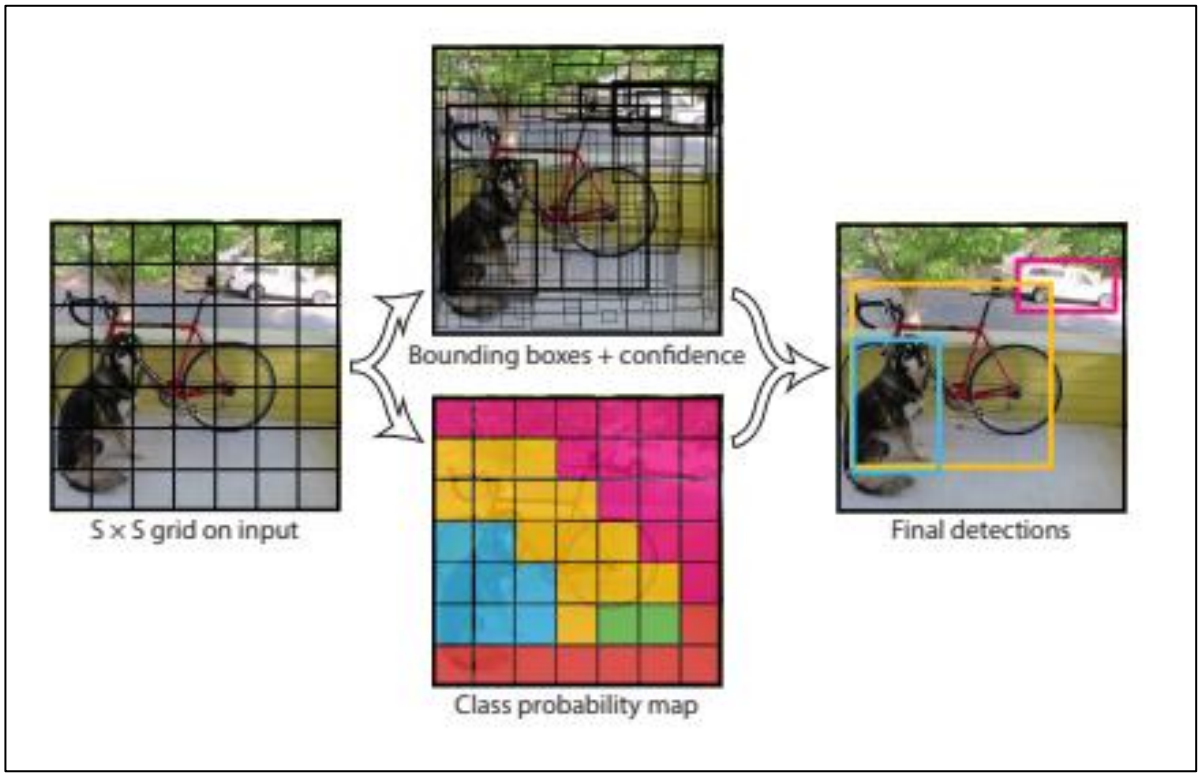

Рисунок 10 – Принцип работы модели YOLO

Также стоит отметить, что YOLO является моделью предобученной на крайне больших объемах данных, что позволяет не обучать модель с нуля, а дообучить ее для специфики конкретной задачи при помощи добавления пользовательского датасета с необходимыми изображениями. Таким образом использование YOLO является наиболее эффективным способом параллельного решения задач детекции и классификации в режиме реального времени.

Выводы по второй главе

Таким образом были определены требования, необходимые для разработки каждого из объявленных компонентов системы.

Для графического пользовательского интерфейса были разработаны и описаны диаграммы вариантов использования и компонентов, а также спроектированы макеты как самого приложения настройки параметров системы, так и всплывающего сообщения об ошибке. При проектировании учитывались требования, определенные в начале этапа.

При проектировании компонента обмена данных были описаны преимущества и возможности при использовании протокола CAN, в числе которых установка наиболее устойчивого и надежного соединения между компонентами системы, которое достигается за счет встроенного протокола обнаружения ошибок, аварийно заканчивающего передачу текущего сообщения при их возникновении. Стоит отметить, что благодаря данному протоколу повышается скорость передачи данных.

При проектировании нейронной сети был описан принцип работы модели YOLO, а также выявлено явное преимущество ее использования в виде быстрого и эффективного решения задач детекции и классификации объектов.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства реализации

Для реализации программных компонентов системы предотвращения для крупногабаритной техники был выбран язык программирования Python 3.7 [14]. Для написания кода и отладки использовалась среда разработки PyCharm 2023.3.3 (Community Edition) [15], обучение нейросети проводилось в среде разработки Google Collab [16], данные для обучения были взяты с сайта Roboflow [17].

В процессе разработки были использованы следующие программные продукты и библиотеки.

Customtkinter 5.2.2 [18]

Библиотека Python, которая предоставляет кастомные виджеты для Tkinter, делая интерфейсы более современными и удобными.

CTkMessagebox 2.5 [19]

Расширение для библиотеки Customtkinter, которое предоставляет удобную реализацию стилизованных сообщений, соответственно была использована при выведении всплывающих сообщений ошибки на экран пользователя.

Scapy 2.5.0 [20]

Библиотека Python для создания, отправки, захвата и анализа сетевых пакетов.

Clearml 1.16.1 [21]

Платформа для управления жизненным циклом машинного обучения, которая включает отслеживание экспериментов.

Ultralytics 8.2.28 [22]

Библиотека Python, разработанная создателями модели YOLO? предоставляющая инструменты для построения, обучения и развертывания моделей глубокого обучения.

3.2. Реализация приложения

Реализация пользовательского интерфейса

При реализации использовалась библиотека «Custom TKinter». С ее помощью был реализован интерфейс приложения на основе разработанного ранее макета. Интерфейс представлен на рисунке 11.

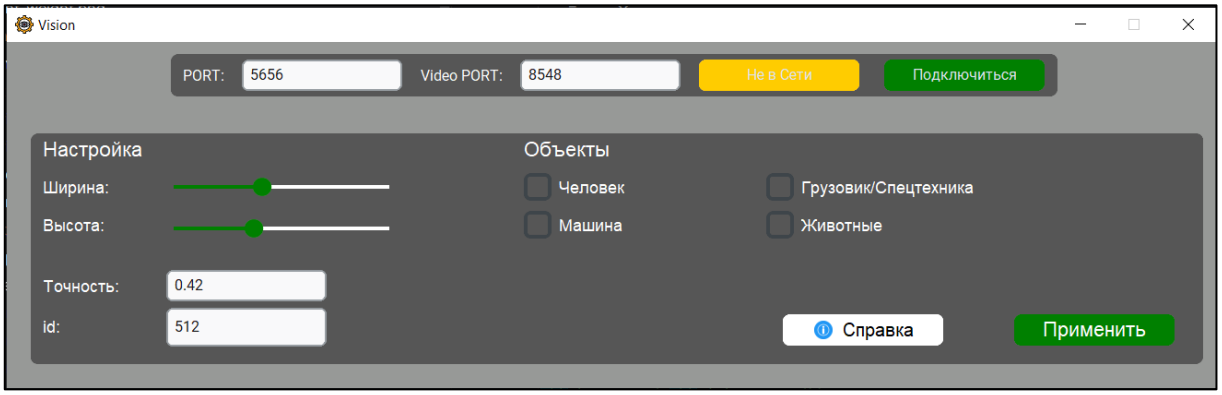

Рисунок 11 – Интерфейс разработанного приложения

В качестве параметров настройки подключения были добавлены возможности настройки порта для принятия сообщений и фактический номер порта компьютера пользователя, данные настройки заданы по умолчанию, но подлежат изменению в случае необходимости. Также в данном блоке реализована кнопка для подключения и проверка статуса подключения, которая меняет цвет и статус «Не в сети» на статус «В сети» после нажатия кнопки «Подключиться».

В качестве параметров для настройки детекции предусмотрен выбор объектов детекции, осуществляемый при помощи отметки пользователем необходимых позиций. Также предусмотрена настройка ширины и высоты области, при попадании в которую машина останавливается. Данная настройка реализована при помощи перемещения ползунка. При перемещении ползунка влево параметр уменьшается, а при перемещении вправо увеличивается. Также добавлена настройка точности определения объектов при увеличении которой качество определения объектов увеличивается, и настройка id сообщения, поступающего в CAN шину.

Реализация обработки ошибок

Для корректной работы приложения была предусмотрена обработка ошибок посредством всплывающих окон, содержащих сообщения об ошибке или предупреждения о невозможности корректной работы системы. Соответственно сообщения разделены на две категории: предупреждения и ошибки. Интерфейс предупреждения представлен на рисунке 12.

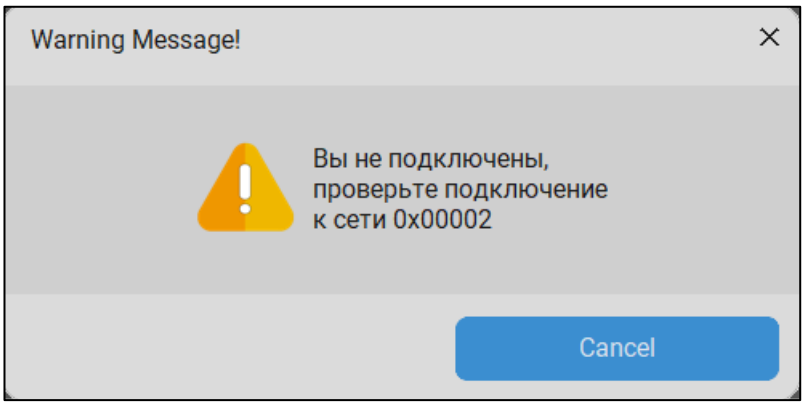

Рисунок 12 – Интерфейс всплывающего окна предупреждения

На данный момент предусмотрено четыре ситуации для появления окна предупреждения.

1. «Вы не подключены, проверьте подключение к сети» – предупреждение, появляющееся при отсутствии доступа к сети.
2. «В сети нет устройств» – предупреждение, возникающее при отсутствии устройств в сети.
3. «Введен недопустимый порт пользователя» – предупреждение, возникающее при указании номера порта, вне диапазона от 0 до 65535.
4. «Введен недопустимый порт видео» – предупреждение, возникающее при указании номера порта, не входящего в диапазон от 0 до 65535.

Также были предусмотрены ситуации, при которых появляется всплывающее окно ошибки

1. «Сообщение не отправлено» – ошибка, которая возникает при попытке применить настройки детекции без подключения к аппаратному модулю системы.

2. «При подключении произошла ошибка, подождите и попробуйте снова» – ошибка, возникающая если аппаратный модуль не успел начать свою работу.

Интерфейс окна ошибки представлен на рисунке 13.

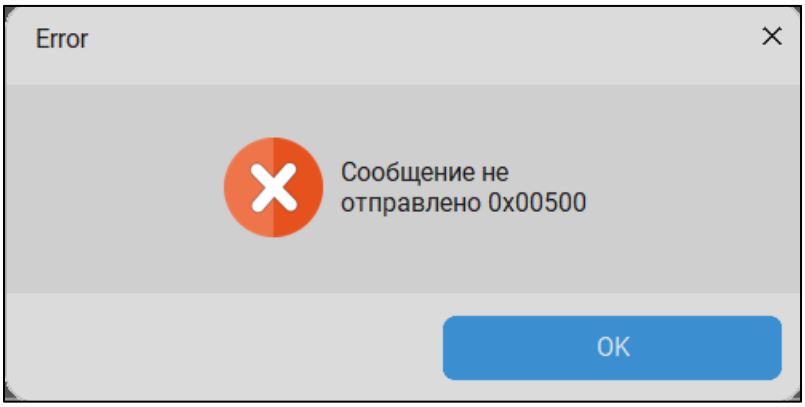

Рисунок 13 – Интерфейс окна ошибки

Реализация модуля подключения к аппаратной части системы

При реализации модуля в начале происходит проверка состояния флага подключения, при условии подключения устройстве вызывается метод `kill_jetson` для отключения программной части системы. Иначе начинается процесс подключения, прежде всего из пользовательского интерфейса извлекаются значения портов. После чего происходит проверка их корректности, при неправильности данных выводятся всплывающие окна ошибки.

Если данные корректны, то происходит настройка адреса для подключения. Затем обновляется файл конфигурации, флаг подключения устанавливается в состояние `True`.

После чего вызывается метод проверки соединения `self.call` если соединение установлено, создается объект `Run_Videoplayer_IMG`, отправляется сообщение с данными для корректного запуска видеопотока. И происходит непосредственный запуск. Также обновляется статус кнопки подключения и запускается мониторинг соединения при помощи метода

self.thread_cheker. Иначе вызывается метод kill_jetson. Код модуля представлен в листинге 1.

Листинг 1 – Код модуля подключения к аппаратной части системы

```
def connect_to_jetson(self):
    global flag_connect,base_ip,Target_Host,main_gateway
    if flag_connect:
        self.kill_jetson()
    else:
        try:
            if base_ip != None:
                user_ip = base_ip
                user_port = int(self.port_our_entry.get())
                user_video_port = int(self.port_video_entry.get())
                if user_port>65535 or user_port==user_video_port or
user_port <0:
                    messageboxes.show_warning("Введен недопустимый порт
пользователя 0x00200")
                elif user_video_port > 65535 or  user_video_port <0:
                    messageboxes.show_warning("Введен недопустимый порт ви-
део 0x00201")
                else:
                    target_ip = Target_Host
                    target_port = 7474
                    self.a = Connect_Module()
                    self.a.set_iPort_address(user_ip, user_port)
                    self.a.set_adrress((target_ip, target_port))
                    self.a.create_socket_data()
                    update_file({"ip": main_gateway, "port_data":
user_port, "port_video": user_video_port})
                    flag_connect = True
                    self.connect_button_txt = 'Отключиться'
                    self.back_conncet = 'red'
                    check_2 = self.call()
                    if check_2:
                        self.b = Run_Videoplayer_IMG()
                        self.b.set_iPort_stream(user_ip, user_video_port)
                        self.a.send_data({'video': 0, 'ip': base_ip,
'port': user_video_port, 'can': True, 'id':0x200 }, 1)
                        self.b.run_stream()
                        self.button_connection_status = CTkButton(mas-
ter=self.up_bar, text=self.connect_button_txt, fg_color=self.back_conncet,
command=self.connect_to_jetson).grid(row=0, column=11, padx=10, pady=5,
sticky="w")
                        self.update()
```

Реализация модуля сканирования сети

В качестве модуля сканирования был реализован метод для поиска всех устройств, находящихся в локальной сети и установления соединения с доступными устройствами. Данный метод был реализован для упрощения использования приложения оператором. Код метода scan представлен в листинге 2.

Листинг 2 – Код метода сканирования сети

```
def scan(self,target_ip):
    socket_scan = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    socket_scan.bind((HOST,PORT))
    arp = ARP(pdst=target_ip+"/24")
    ether = Ether(dst="ff:ff:ff:ff:ff:ff")
    packet = ether/arp
    result = srp(packet, timeout=3, verbose=0)[0]
    ip_pull = []
    for sent, received in result:
        ip_pull.append(received.psrc)
    for i in range(len(ip_pull)):
        if ip_pull[i]!=target_ip:
            give = 'give'
            socket_scan.sendto(give.encode(),(ip_pull[i],7474))
            try:
                wait_connect = select.select([socket_scan],[],[],5)
                data, addr = wait_connect[0][0].recvfrom(2048)
                if data != None:
                    ty ='ty'
                    socket_scan.sendto(ty.encode(),(ip_pull[i],7474))
                    self.ADDR_CONNNET = (ip_pull[i],7474)
                    self.TARGET_HOST = target_ip
                    break
```

Данный метод принимает параметр `target_ip`, который указывает IP-адрес целевого устройства. Внутри метода `scan` создается UDP сокет `socket_scan`, который связывается с локальным хостом и портом. Затем с помощью библиотек `scapy` создается и отправляется ARP-запрос для сканирования сети. ARP-запрос создается с помощью классов `ARP` и `Ether`.

Полученные ответы на ARP-запросы обрабатываются, и IP-адреса доступных для подключения устройств сохраняются в список `ip_pull`. Далее выводится список всех доступных устройств, после чего выполняется проверка каждого найденного IP-адреса. Если IP-адрес не совпадает с целевым `target_ip`, сокет отправляет сообщение `give` на обнаруженное устройство.

Сокет ждет ответа устройства в течение пяти секунд с помощью функции `select`. При получении ответа за это время, выводятся данные устройства, и сокет отправляет ответное сообщение `ty`. Затем атрибуты `ADDR_CONNNET` и `TARGET_HOST` обновляются, а цикл сканирования прерывается.

Реализация модуля проверки статуса подключения

Также был реализован модуль проверки статуса подключения для удобства пользователей. Данный модуль использует вышеописанную функцию сканирования, при помощи которой настраиваются параметры и происходит обновление файла конфигурации. После чего метод обновляет цвет и текст кнопки в зависимости от статуса подключения. Код обновления представлен в листинге 3.

Листинг 3 – Код для обновления статуса подключения

```
if cheker_status:
    self.check_connect_button_txt = 'В Сети'
    self.back_conncet_check = 'green'
else:
    self.check_connect_button_txt = 'Не в Сети'
    self.back_conncet_check = yellow_back
self.button_check_connection_status = CTkButton(
    master=self.up_bar,
    text=self.check_connect_button_txt,
    fg_color=self.back_conncet_check,
    command=self.check_network).grid(
        row=0,
        column=10,
        padx=10,
        pady=5,
        sticky="w")
```

3.3. Реализация компонента обмена данных

Для обеспечения максимальной отзывчивости системы, что дает возможность наиболее качественного контроля ситуации в режиме реального времени и минимизации задержек было принято решение реализовать каждый из классов в отдельном потоке. Также внутри каждого из классов реализована функция проверки контрольной суммы [23]. Метод `crc8` используется для обеспечения целостности передачи данных, Он позволяет проверить не произошло ли искажение данных в процессе передачи. Данная проверка является стандартной при использовании CAN протокола.

Класс `Can_Reader` предназначен для непрерывного мониторинга can порта на предмет поступающих данных. Функция `start_write_msg` в данном классе отвечает за обработку поступающих сообщений и запускает процесс анализа этих сообщений. В данном методе реализована проверка на

наличие данных в сообщении. Если сообщение является пустым, то вызывается метод класса re, отвечающий за сброс всех параметров объекта класса для подготовки к приему нового сообщения. Если же поступающее сообщение не пустое и в начале присутствует знак «$», означающий начало сообщения, флаг `writer` устанавливается в состояние `True`, что указывает на начало приема данных. После начала приема данных, они добавляются в список в формате шестнадцатеричных строк. Далее происходит анализ первого элемента в списке данных для определения максимальной длины сообщения. Если длина списка данных достигает максимальной длины, то производится вычисление `crc` и сравнение его со значением, указанным в сообщении. При совпадении этих значений вызывается метод `check_can_list`, отвечающий за анализ типа сообщения. После этого вызывается метод re, для подготовки к приему нового сообщения.

Также в данном классе реализован метод run, который отвечает за бесконечный цикл чтения, поступающих посылок. Данный цикл может завершиться только при установке флага `stop_ALL` в состояние `True`.

В классе `Can_Sender` происходит формирование и отправка сообщений. За формирование сообщений отвечает метод `pack_format`, в который передаются данные для отправки. В нем изначально инициализируется пустой список, который будет содержать пустое сообщение. Далее происходит проверка длины данных, если она превышает 8 байт или равна нулю, выводится сообщение об ошибке. После прохождения проверки происходит сбор данных, подсчет контрольной суммы, которая также добавляется в сообщение. Далее в начало сообщения добавляется начальный байт, после чего сообщение считается сформированным.

Также в классе `Can_Sender` описан метод `run`, который реализует основной алгоритм отправки данных по CAN-шине. В нем реализован бесконечный цикл, который проверяет наличие объекта в зоне движения машины, при наличии таковой информации метод формирует сообщение с кодом `0xFD` и отправляет его. После чего флаг наличия объекта устанавливается в

состояние `False`. Если объект не обнаружен, то при формировании сообщения добавляется код `0xFC`, после чего происходит отправка сообщения. Цикл завершается только при условии достижения состояния `True` флагом `stop_ALL`. Данная проверка также включена в реализацию цикла.

Реализация метода `run` класса `Can_Sender` представлена в листинге 4.

Листинг 4 – Реализация метода `run` класса `Can_Sender`

```
def run(self):
    global sending_special_info, id_msg_can, serial_port
    try:
        print('Start CAN')
        while True:
            if sending_special_info:
                msg_usb_to_can = self.pack_format(
                    id_msg_can,False,
                    [0xFD,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF])
                serial_port.write(msg_usb_to_can)
                sending_special_info = False
                time.sleep(0.5)
            else:
                msg_usb_to_can = self.pack_format(
                    id_msg_can,False,
                    [0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF])
                serial_port.write(msg_usb_to_can)
                time.sleep(0.5)
            if stop_ALL == True:
                print('video error')
                break
    except:
         print('drop')
```

Алгоритм формирования и отправки посылок представлена на рисунке 14. В данной диаграмме алгоритма отражено формирование нового сообщения, проверка обнаружения объекта, на основе которой в итоговое сообщение добавляются данные о наличии или отсутствии детектируемого объекта в зоне движения крупногабаритной техники, вычисление контрольной суммы сообщения, являющееся необходимой проверкой для обеспечения эффективного обмена данными, отправка сообщения и проверка получения флага остановки, при положительном результате которой алгоритм завершает работу, а иначе зацикливается и начинает формирование нового сообщения с такими же шагами.

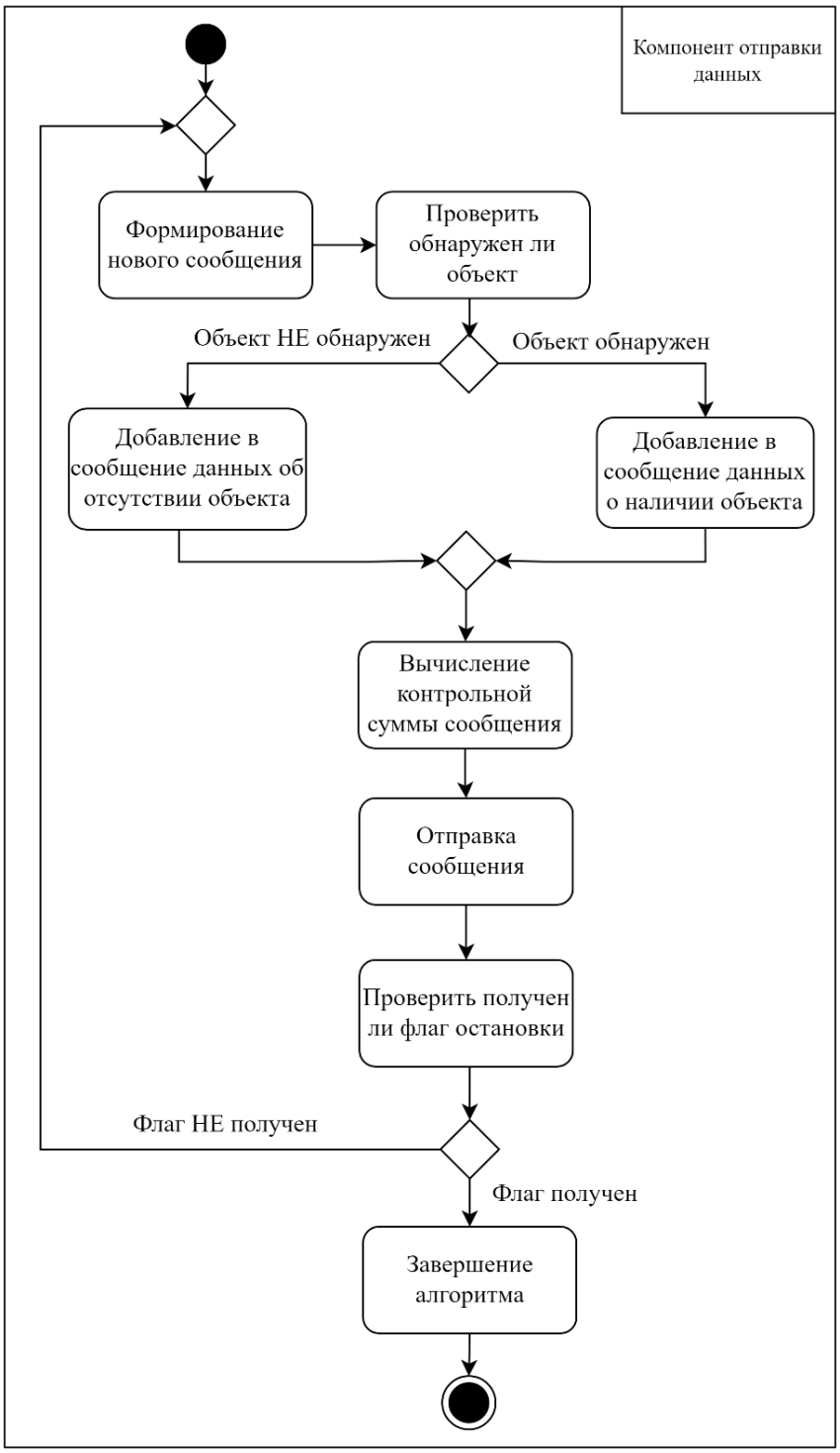

Рисунок 14 – Алгоритм отправки посылок

3.4. Реализация нейронной сети

На данный момент компания Ultralistics выпустила новую модель YOLOv8 [24], которая в сравнении с другими версиями обеспечивает наилучшие результаты. Сравнение моделей представлено на рисунке 15.

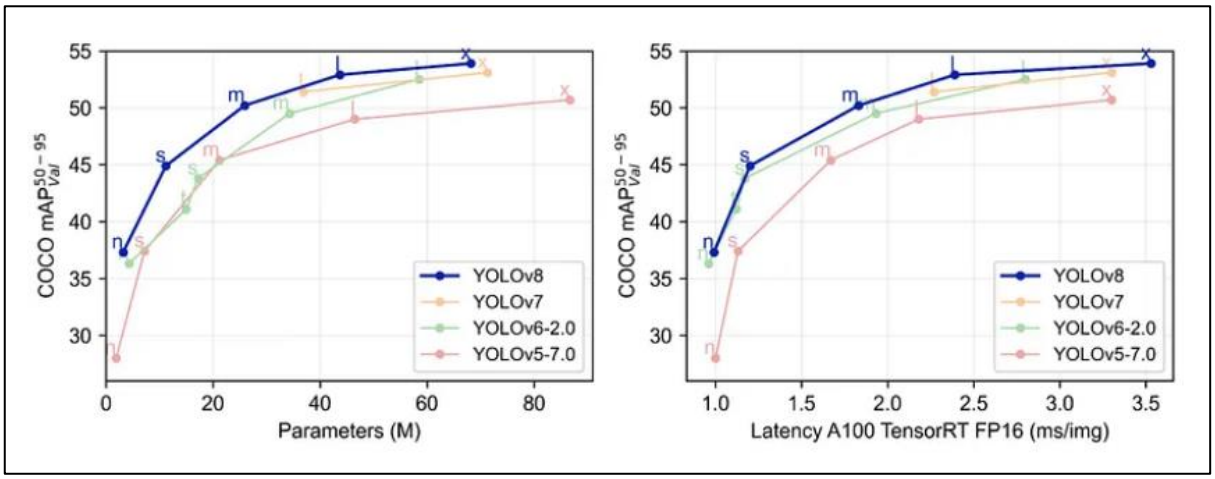

Рисунок 15 – Сравнение моделей YOLO

При сравнении производительности различных моделей YOLO, на одинаковом наборе данных COCO сравнивались разные показатели. Как можно видеть по графикам точность восьмой версии является наивысшей.

Исходя из этого, было принято решение использовать именно восьмую версию для нейросети. Датасет был взят с сайта roboflow [25]. Так как было принято решение дообучить нейросеть распознавать стоп-знаки [26], датасет состоит из 3048 изображений стоп-знака в различных ракурсах. Он изначально разделен на тренировочную, тестовую и валидационную выборки. Тренировочная выборка содержит 2552 изображения, тестовая 213, а валидационная 283. Также стоит отметить, что каждая из картинок дополнена рамкой, обозначающей границы объекта – аннотацией [27]. Пример используемых изображений представлен на рисунке 16.

Рисунок 16 – Пример изображений датасета

Обучение происходило в среде Google Collab. Код обучения представлен в листинге 5.

Листинг 5 – Обучение нейронной сети

```
model = YOLO('yolov8m.pt')
results = model.train(
    data='/content/data.yaml',
    imgsz=1280,
    epochs=50,
    batch=8,
    name='yolov8m_v8_50e')
```

По результатам обучения были построены графики, представленные на рисунке 17.

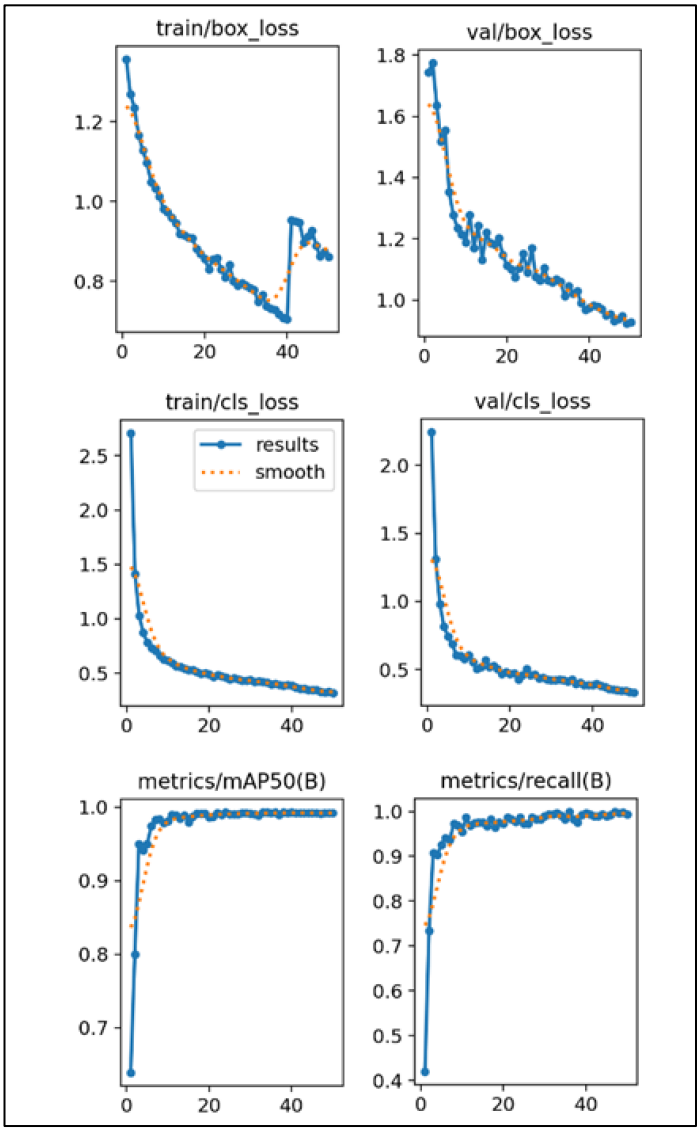

Рисунок 17 – Результаты обучения нейронной сети

Таким образом можно заметить скачки на тренировочной выборке, но не на валидационной. С течением времени потери уменьшаются, а метрики увеличиваются, что дает возможность понять, что нейросеть действительно обучается. Стоит отметить, что итоговая точность согласно метрике precision – 0,993. А средняя скорость распознания 12,751 ms.

Выводы по третьей главе

Таким образом в ходе реализации были разработаны необходимые компоненты системы с учетом, выявленных ранее, требований.

Так, был реализован интуитивно понятный графический пользовательский интерфейс, обеспечивающий подключение к аппаратной части системы по сети. В данном интерфейсе были реализованы блоки настройки параметров подключения и детекции. Также была реализована обработка ошибок при помощи всплывающих окон, а также реализованы методы для упрощения процесса использования приложения.

Компонент обмена информацией, реализованный на основе протокола CAN, обеспечивает эффективный обмен данными, с минимальными задержками и обработкой ошибок, при которой исключается обработка поврежденных посылок.

Также была обучена нейронная сеть, которая обеспечивает высокую точность и скорость детекции.

4. ТЕСТИРОВАНИЕ

4.1. Тестирование приложения

Для тестирования приложения было проведено функциональное [28] тестирование основных функций. Результаты тестирования представлены в таблице 1.

Таблица 1 – Функциональное тестирование приложения

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Проверка работы кнопки «Подключиться»	1. Проверить заполненность полей «Port» и «video port» 2. Нажать на кнопку подключиться	Открывается окно с видеопотоком.	Да
2	Проверка корректности данных поля «Port»	1. Заполнить поле Port числом, не входящим в диапазон от 0 до 65535 2 Нажать на кнопку подключиться	Появляется всплывающее окно с ошибкой «Введен недопустимый порт пользователя»	Да
3	Проверка корректности данных поля «Video port»	1. Заполнить поле Port числом, не входящим в диапазон от 0 до 65535 2 Нажать на кнопку подключиться	Появляется всплывающее окно с ошибкой «Введен недопустимый порт видео»	Да
4	Проверка корректной работы статуса подключения «Не в сети»	1. Проверить статус без подключения к аппаратной части	Плашка имеет желтый цвет и статус «Не в сети»	Да
5	Проверка корректной работы плашки статус подключения «В сети»	1. Подключиться к аппаратной части системы 2. Проверить статус подключения	Плашка имеет зеленый цвет и статус «В сети»	Да
6	Проверка отображения видеопотока	1. Подключиться к аппаратной части системы	Открывается окно с видеопотоком	Да
7	Проверка увеличения ширины области остановки при сдвиге ползунка настройки вправо	1. Подключиться к аппаратной части системы 2. Увеличить ширину в настройках 3. Нажать кнопку «применить»	Ширина области остановки увеличивается	Да

Продолжение таблицы 1

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
8	Проверка уменьшения ширины области остановки при сдвиге ползунка настройки влево	1. Подключиться к аппаратной части системы 2. Уменьшить ширину в настройках 3. Нажать кнопку «применить»	Ширина области остановки уменьшается	Да
9	Проверка увеличения высоты области остановки при сдвиге ползунка настройки вправо	1. Подключиться к аппаратной части системы 2. Увеличить высоту в настройках 3. Нажать кнопку «применить»	Высота области остановки увеличивается	Да
10	Проверка детекции объекта «Человек»	1. Подключиться к аппаратной части системы 2. Отметить в поле объекты «Человек» 3. Нажать кнопку «применить»	Объект «Человек» распознается на видеопотоке	Да
11	Проверка детекции объекта «Машина»	1. Подключиться к аппаратной части системы 2. Отметить в поле объекты «Машина» 3. Нажать кнопку «применить»	Объект «Машина» распознается на видеопотоке	Да
12	Проверка детекции объекта «Грузовик»	1. Подключиться к аппаратной части системы 2. Отметить в поле объекты «Грузовик» 3. Нажать кнопку «применить»	Объект «Грузовик» распознается на видеопотоке	Да
13	Проверка детекции объекта «Животные»	1. Подключиться к аппаратной части системы 2. Отметить в поле объекты «Животные» 3. Нажать кнопку «применить»	Объект «Животные» распознается на видеопотоке	Да
14	Обработка ошибки «Нет подключения к сети»	1. Нажать на кнопку «Подключиться» без подключения к сети	Появляется всплывающее окно с ошибкой «Вы не подключены, проверьте подключение к сети»	Да
15	Обработка ошибки «В сети нет устройств»	1. Нажать на кнопку «Подключиться» без подключения аппаратной части системы к сети	Появляется всплывающее окно с ошибкой «В сети нет устройств»	Да

Окончание таблицы 1

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
16	Обработка ошибки «Сообщение не отправлено»	1. Нажать на кнопку «Применить» до нажатия кнопки «Подключиться»	Появляется всплывающее окно с ошибкой «Сообщение не отправлено»	Да
17	Обработка ошибки «Введен недопустимый порт пользователя»	1. Ввести номер порта пользователя вне диапазона от 0 до 65535 2. Нажать на кнопку «Подключиться»	Появляется всплывающее окно с ошибкой «Введен недопустимый порт пользователя»	Да
18	Обработка ошибки «Введен недопустимы порт видео»	1. Ввести номер порта видео вне диапазона от 0 до 65535 2. Нажать на кнопку «Подключиться»	Появляется всплывающее окно с ошибкой «Введен недопустимый порт видео»	Да

4.2. Тестирование компонента обмена данными

Для проведения тестирования был разработан набор ручных тестов, покрывающий реализованный функционал. Тестирование проводилось посредством имитации приема и отправки посылок. Протокол тестирования представлен в таблице 2.

Таблица 2 – Функциональное тестирование компонента обмена данными

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Обработка корректных данных	Отправка корректных данных в формате JSON	Данные записались в соответствующие переменные	Да
2	Отправка сообщения при наличии объекта перед машиной	Установить флаг sending_special_info в состояние True	Отправка корректного сообщения по CAN шине	Да
3	Отправка сообщения при отсутствии объекта перед машиной	Установить флаг sending_special_info в состояние False	Отправка корректного сообщения по CAN шине	Да
4	Корректное завершение алгоритма	Установить флаг stop_ALL в состояние True	Алгоритм завершает работу	Да

4.3. Тестирование нейронной сети

Для тестирования корректной работы нейронной сети были разработаны ручные тесты, которые покрывают необходимый функционал в виде корректного определения объектов при различных окружающих условиях.

Тестирование проводилось посредством загрузки видеопотоков [29] в нейронную сеть. Протокол тестирования представлен в таблице 3.

Таблица 3 – Функциональное тестирование нейронной сети

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Проверка корректности работы при результатах, приближенных к идеальным	Загрузить в нейронную сеть видео с условиями, приближенными к идеальным	Нейронная сеть распознает стоп-знак	Да
2	Проверка корректности работы в условиях сумерков	Загрузить в нейронную сеть видео, снятое в сумерки	Нейронная сеть распознает стоп-знак	Да
3	Проверка корректности работы при нахождении в тени	Загрузить в нейронную сеть видео со знаком, находящимся в тени	Нейронная сеть распознает стоп-знак	Да
4	Проверка корректности работы в ночное время	Загрузить в нейронную сеть видео, снятое в ночное время	Нейронная сеть распознает стоп-знак	Да

Выводы по четвертой главе

Таким образом в ходе тестирования были разработаны и проведены функциональные тесты для каждого из компонентов. Для компонента приложения были проведены 18 тестов, покрывающие основной функционал. Для компонента обмена данными проводилось 4 теста. Также, для нейронной сети было проведено 4 теста. Все разработанные тесты прошли успешно.

ЗАКЛЮЧЕНИЕ

В рамках данной работы были разработаны программные компоненты системы предотвращения столкновения для крупногабаритной техники, при этом были решены нижеперечисленные задачи:

1) выполнен анализ предметной области и проведен обзор аналогов;
2) спроектированы компоненты системы;
3) реализованы компоненты;
4) проведено тестирование.

В настоящий момент продолжается улучшение и доработка системы, в частности происходит внедрение нейросети, способной классифицировать и детектировать большее количество объектов, чем та, которая используется в данный момент. При этом и классификация, и детекция проходит без потери скорости и эффективности.

Также в качестве улучшения ведется работа над добавлением обзора в 360 градусов для оператора данного программно-аппаратного комплекса, что позволит обеспечить больший контроль над ситуацией, происходящей не только спереди, но и вокруг крупногабаритной техники. На данный момент выявлено несколько возможных стратегий реализации данной задачи, проводится их сравнительный анализ.

ЛИТЕРАТУРА

1. Amato F., Nardone R., Santone A. CAN-Bus AttackDetection With Deep Learning. // Transactions on Intelligent Transportation Systems, IEEE 2021. – 5081–5090 pp.

2. SICK Safety Collision Prevention. [Электронный ресурс] URL: https://www.sick.com/ag/en/catalog/archive/s3000-anti-collision/c/g329351 (дата обращения: 25.02.2024 г.).

3. Caterpillar Cat Detect Safety System. [Электронный ресурс] URL: https://www.cat.com/en_US/support/operations/technology/product-link-owners-manuals/cat-object-detection-manuals.html (дата обращения: 25.02.2024 г.).

4. Komatsu Intelligent Machine Control. [Электронный ресурс] URL: https://www.komatsu.eu/en/komatsu-intelligent-machine-control (дата обращения: 25.02.2024 г.).

5. John Deere ActiveCollision Avoidance System. [Электронный ресурс] URL: https://www.deere.com/en/technology-products/precision-ag-technology/guidance/auto-trac/ (дата обращения: 25.02.2024 г.).

6. Kumar P., Narasimha Swamy S., Purohit G., Raju K. S., Real-time, yolo-based intelligent surveillance and monitoring system using jetson tx2. // Data Analytics and Management Proceedings of ICDAM, Springer 2021. – 461– 471 pp.

7. Jetson Software Documentation. [Электронный ресурс] URL: https://docs.nvidia.com/jetson/tation (дата обращения: 29.01.2024 г.).

8. Donmez T. C. M. Anomaly Detection in Vehicular CAN Bus Using-Message Identifier Sequences. // IEEE Access, 2021. – 105–108 pp.

9. Marchetti M., Stabili D. Anomaly detection of can bus messagesthrough analysis of id sequences. // IEEE Intelligent VehiclesSymposium (IV), 2017. – 1577–1583 pp.

10. Taylor A., Japkowicz N., Leblanc S. Frequency-based anomaly detection for the automotive can bus. // 2015 World Congress on IndustrialControl Systems Security (WCICSS), 2015. – 45–49 pp.

11. Müter M., Asaj N. Entropy-based anomaly detection for in-vehiclenetworks. // 2011 IEEE Intelligent Vehicles Symposium (IV), 2011. – 1110– 1115 pp.

12. Дьяченко А.А., Гущина О.М. Анализ этапов развития одноступенчатых детекторов объектов на основе YOLO. // Информационные технологии в моделировании и управлении: подходы, методы, решения, 2022. – С. 427–435.

13. Redmon J., Divvala S., Girshik R. You Only Look Once: Unified, Real-Time Object Detection.//IEEE Conference on Computer Vision and Pattern Recognition Conference, 2016. – 45–49 pp.

14. Python 3.7. [Электронный ресурс]. URL: https://www.python.org/downloads/release/python-373/ (дата обращения: 15.03.2024 г.).

15. Pycharm. [Электронный ресурс]. URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 15.03.2024 г.).

16. Google Collab Documentation. [Электронный ресурс]. URL: https://cloud.google.com/colab/docs (дата обращения: 15.03.2024 г.).

17. Roboflow Documentation. [Электронный ресурс]. URL: https://help.roboflow.com/ (дата обращения: 15.03.2024 г.).

18. Customtkinter Documentation. [Электронный ресурс]. URL: https://customtkinter.tomschimansky.com/ (дата обращения: 15.03.2024 г.).

19. CTkMessagebox Documentation. [Электронный ресурс]. URL: https://libraries.io/pypi/CTkMessagebox (дата обращения: 15.03.2024 г.).

20. Scapy Documentation. [Электронный ресурс]. URL: https://www.packetlevel.ch/html/scapy/docs/scapy.pdf (дата обращения: 15.03.2024 г.).

21. Clearml Documentation. [Электронный ресурс]. URL: https://clear.ml/docs/latest/docs/ (дата обращения: 15.03.2024 г.).

22. Ultralytics Documentation. [Электронный ресурс]. URL: https://docs.ultralytics.com/hub/models/ (дата обращения: 15.03.2024 г.).

23. Gao X., Huang D., Chen Y., Jin W., Luo Y. The design of a distributed control system based on CAN bus. // IEEE International Conference on Mechatronics and Automation, 2017. – 1118-1122 pp.

24. YOLOv8 Documentation. [Электронный ресурс]. URL: https://yolov8.org/yolov8-documentation/ (дата обращения: 29.05.2024 г.).

25. Dataset Stop_sign. [Электронный ресурс]. URL: https://universe.roboflow.com/wewilldoit/stopsign-hx5qv/dataset/2# (дата обращения: 15.03.2024 г.).

26. Сабиров А.И., Катасёв А.С., Дагаева М.В. Нейросетевая модель распознавания знаков дорожного движения в интеллектуальных транспортных системах. // Компьютерные исследования и моделирование. 2021. – №2 – С. 429-435.

27. Никитин Д.В., Тараненко И.С., Катаев А.В. Детектирование дорожных знаков на основе нейросетевой модели YOLO. // Инженерный вестник Дона, 2023. – №. 7 (103). – С. 91–99.

28. Функциональное тестирование [Электронный ресурс]. URL: https://unetway.com/tutorial/funkcionalnoe-testirovanie (дата обращения: 29.05.2024 г.).

29. Аксютина М.С., Гончарук С.Е. Распознавание объектов в видеопотоке при помощи алгоритма YOLO и технологии openface // Научно-техническое творчество аспирантов и студентов, 2018. – С. 208–211.