МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Факультет физики и информационных технологий
Кафедра общей физики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

МОБИЛЬНОЕ УСТРОЙСТВО РАСПОЗНОВАНИЯ ОБЪЕКТОВ В ВИДЕООБРАЗЕ

ГГУ 1-39 03 02 112 ПЗ

Исполнитель:
Студент группы МС-32: _____ Лаптев И. А.

Научный руководитель:
Старший преподаватель
Кафедры общей физики _____ Ковалёв А. А.

Гомель 2023

РЕФЕРАТ

Курсовой проект: 36 страниц, 3 главы, 29 рисунков, 6 источников.

RASPBERRY PI, LCD, I2C, PYTHON, HOG ДЕСКРИПТОР, АЛГОРИТМЫ РАСПОЗНАВАНИЯ ОБЪЕКТОВ, CSI-КАМЕРА.

Объектом и предметом исследования является разработка мобильного устройства для распознавания объектов в видеообразе.

Цель работы: после получения практических навыков в разработке мобильных устройств, создать мобильное устройство способное распознавать объекты в видеообразе и реализовать возможность вывод информации о распознанных объектах.

В ходе данного курсового проекта решились следующие **задачи**:

- выполнен анализ предметной области.
- обоснована разработка мобильного устройства распознавания объектов в видеообразе.
- разработана структурная электрическая схема.
- разработана принципиальная электрическая схема.
- разработаны модель и алгоритм функционирования мобильного устройства распознавания объектов в видеообразе.
- оформлена пояснительная записка.

Актуальность темы курсовой обусловлена рядом факторов. Распознавание объектов в реальном времени является важной задачей в области компьютерного зрения и машинного обучения. Быстродействие и точность алгоритмов распознавания объектов на мобильных устройствах может значительно расширить возможности мобильных приложений, например, в области автоматизации рабочих процессов и обработки изображений.

Во время работы над курсовым проектом были исследованы различные алгоритмы распознавания объектов, выбраны язык написания этих алгоритмов и платформа для реализации проекта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1. Анализ предметной области 5
1.1 Общая информация 5
1.1.1 Способ передачи данных на дисплей 5
1.2 Выбор аппаратной платформы 7
2. Разработка электрических схем 12
2.1 Разработка структурной схемы 12
2.1.1 Обоснование базовых блоков структурной электрической схемы . 12
2.1.2 Обоснование связей структурной электрической схемы 15
2.2 Разработка структурной схемы 16
2.2.1 Обоснование выбора САПР для разработки принципиальной электрической схемы 16
2.2.2 Описание используемых элементов 19
3. Разработка алгоритма функционирования распознавания объектов в видеообразе 22
3.1. Выбор алгоритма функционирования распознавания объектов в видеообразе 22
3.2. Описание алгоритма функционирования распознавания объектов в видеообразе 25
3.3. Выбор IDE и языка программирования для написания алгоритма 26
3.4 Описание функций алгоритма 28
Заключение 35
Список используемых источников 36

ВВЕДЕНИЕ

Современные мобильные устройства, такие как смартфоны и планшеты, обладают высокой вычислительной мощностью и расширенными возможностями обработки изображений. Они стали неотъемлемой частью нашей повседневной жизни, и мы все больше полагаемся на них для выполнения различных задач. В связи с этим возникает потребность в разработке и применении инновационных технологий, которые могут улучшить наши возможности и обеспечить более удобный и эффективный пользовательский опыт.

Одной из актуальных проблем, стоящих перед мобильными устройствами, является распознавание объектов в видеообразе. Распознавание объектов — это процесс автоматического определения и классификации объектов на изображении или в видеопотоке. Эта технология находит широкое применение в различных областях, включая компьютерное зрение, машинное обучение, робототехнику, автоматизацию и даже в повседневных задачах, таких как распознавание лиц или идентификация объектов.

Одной из основных проблем, с которой сталкиваются мобильные устройства при распознавании объектов в видеообразе, является ограниченность вычислительных ресурсов и ограниченный объем памяти. Мобильные устройства имеют ограниченные возможности по сравнению с настольными компьютерами или серверами, что ограничивает использование сложных и ресурсоемких алгоритмов распознавания объектов. Это требует разработки и оптимизации алгоритмов, которые могут эффективно работать на мобильных платформах и при этом обеспечивать высокую точность распознавания.

Еще одной проблемой является обработка видеопотока в режиме реального времени. Мобильные устройства должны способны обрабатывать видеопоток с высокой скоростью, чтобы обеспечивать непрерывное и плавное распознавание объектов.

1. Анализ предметной области

1.1Общая информация

1.1.1 Способ передачи данных на дисплей

В качестве способа передачи данных на LCD дисплей был выбран протокол I2C, в силу своей простаты реализации и доступности.

Протокол I2C был разработан более 30 лет назад, и в настоящее время имеет широкое распространение под различными модификациями и названиями в электронных устройствах различного назначения.

I2C – является протоколом синхронной связи, что говорит нам о том, что обмен данными происходит по общему для всех связанных устройств сигналу синхронизации. За генерацию такого сигнала отвечает, так называемое Master-устройство. Устройства, которые принимают этот сигнал, называются – Slave. У каждого ведомого (slave) устройства, имеется свой уникальный номер, в то время как ведущий, такого номера не имеет.

Также шина I2C состоит из двух линий: SDA (линия данных) и SCL (линия тактирования). Инициатором обмена всегда выступает master, обмен данными между двумя ведомыми невозможен. На одной двухпроводной шине может быть до 127 устройств.

Рассмотрим некоторые моменты работы протокола I2C. Обращение мастера начинается с падения уровня на шине данных SDA, что является стартовым сигналом для ведомых. Повышения SDA является стом-сигналом. Всё что находится между этими двумя сигналами называется “сообщением”, то есть в этот момент и происходит передача данных.

Рис. 1.1.1. Визуализация работы I2C протокола

Рассмотрим структуру сообщения I2C.

Рис. 1.1.2. Структура I2C сообщения

- Что такое “страт” и “стоп” мы рассмотрели выше.
- Кадр адреса состоит из 7-10 бит, этот размер зависит от разновидности протокола, это и есть уникальный адрес ведомого устройства, к которому обращается ведущий.
- После адреса идёт бит чтения или записи, который показывает, что именно мы хотим – передать или получить данные.
- ACK/NACK – биты подтверждения или не подтверждения приёма информации каждого кадра. Этот бит является единственным, который генерируется в линии SCL ведомым устройством. ACK необходим только для сокращения времени передачи данных. Он обрывает передачу, когда принимающее устройство перестаёт отвечать.

Преимущества интерфейса I2C:

- Необходим лишь один микроконтроллер для управления набором устройств;
- Используется всего два проводника для подключения многих устройств;
- Возможна одновременная работа нескольких ведущих на одной шине;
- Стандарт предусматривает “горячее” подключение и отключение устройств в процессе работы системы;
- В микросхемы, реализующие интерфейс встроен фильтр, который подавляет всплески и тем самым обеспечивает целостность данных.

Недостатки интерфейса I2C:

- Ограничение на ёмкость линии – 400 пФ;
- Несмотря на простоту протокола, программирование контроллера I2C затруднено из-за большого количества возможных нештатных ситуаций на шине;
- Трудность локализации неисправности;

1.2 Выбор аппаратной платформы

При выборе аппаратных платформ для реализации мобильного приложения для распознавания объектов в видеообразе необходимо учитывать ряд факторов, таких как производительность, энергопотребление, доступность необходимых библиотек и средств разработки.

Среди платформ, наиболее распространенных для мобильных устройств, можно выделить смартфон, Arduino и Raspberry PI. Они имеют сходную функциональность и возможности, но различаются по аппаратным характеристикам и экосистеме разработки.

Для реализации приложения на смартфоне можно использовать такие платформы, как Qualcomm Snapdragon, Apple A16 Bionic, MediaTek, Kirin от Huawei и другие. Они обеспечивают высокую производительность и доступны для большинства производителей мобильных устройств. Для разработки на Android также доступны библиотеки и средства разработки, такие как Android Studio, Xcode и OpenCV.

Рис. 1.2.1. Смартфон

Плюсы использования смартфона:

- Практичность: смартфон всегда под рукой и может быть использован в любом месте и в любое время;
- Высокая мобильность: смартфон является портативным устройством, что позволяет использовать его в различных условиях.
- Наличие камеры: смартфоны обычно имеют высококачественные камеры, которые могут быть использованы для сбора видеоданных.

- Высокая производительность: современные смартфоны имеют мощные процессоры и графические усилители, что позволяет обрабатывать видеообразы в режиме реального времени;

Минусы использования смартфона:

- Ограниченные возможности хранения: смартфоны имеют ограниченный объём хранения данных;
- Ограниченные возможности управления камерой: управление камерой смартфона ограниченно, что может ограничивать возможности манипулирования изображением;
- Высокая стоимость: при покупке смартфона, в учёт идёт большое количество модулей, которые не нужны для реализации моего проекта;

Arduino: плата Arduino может быть использована для управления датчиками, но она имеет ограниченные возможности для обработки изображений и видео, что делает ее неидеальным выбором для проекта по распознаванию объектов в видеообразе. Однако, Arduino может использоваться в качестве компонента в сочетании с более мощными устройствами, такими как компьютеры или мобильные устройства.

Рис. 1.2.2. Arduino UNO

Плюсы использования Arduino для распознавания объектов в видеообразе:

- Низкая стоимость: Arduino является довольно доступной аппаратной платформой, которая может быть использована для создания систем распознавания объектов в видеообразе.

• Низкое энергопотребление: Arduino потребляет меньше энергии, чем многие другие аппаратные платформы, что делает его хорошим выбором для систем с ограниченным бюджетом и с низким потреблением энергии;

• Гибкость: Arduino можно легко настроить и расширить с помощью дополнительных модулей и датчиков, что делает его гибким и удобным для создания индивидуальных систем распознавания объектов;

Минусы использования Arduino для распознавания объектов в видеообразе:

•Ограниченные возможности – Arduino не имеет достаточно мощных процессоров и видеокарт, чтобы обрабатывать сложные алгоритмы распознавания объектов в реальном времени;

•Ограниченные возможности хранения – Arduino имеет ограниченный объем хранения, что может быть недостаточным для работы со сложными алгоритмами распознавания объектов в видеообразе;

•Ограниченные возможности связи – Arduino имеет ограниченные возможности для передачи и приёма данных, что может быть недостаточным для работы с большими объёмами видеоданных;

Такая платформа как Raspberry Pi может быть использованы для реализации приложения для распознавания объектов в видеообразе. Raspberry Pi поддерживает большое количество модулей, так как имеет GPIO пины, к которым можно подключить любые модули от Arduino, а также имеет отдельный порт, который предназначен для камеры. Raspberry Pi обеспечивает высокую производительность и доступен для большинства разработчиков. Для разработки на Raspberry Pi доступны библиотеки и средства разработки, такие как Python и OpenCV.

Рис. 1.2.2. Raspberry Pi

Плюсы использования Raspberry Pi для нашего проекта:

- Высокая производительность: Raspberry Pi обладает достаточно мощным процессором, который позволяет выполнять сложные операции по обработке видео в режиме реального времени;
- Гибкость: Raspberry Pi может быть настроен и настроен для выполнения множества задач, включая распознавание объектов в видеообразе. Кроме того, Raspberry Pi имеет различные порты ввода/вывода, которые могут быть использованы для подключения камеры или других устройств;
- Низкая стоимость: Raspberry Pi стоит гораздо меньше, чем большинство компьютеров или ноутбуков, при этом обладая достаточной производительностью для выполнения задач по обработке видео;
- Широкое сообщество разработчиков: Raspberry Pi имеет огромное сообщество разработчиков и пользователей, которые создают и делятся своими проектами и кодом, что может значительно упростить процесс создания приложения для распознавания объектов в видео;

Минусы использования Raspberry Pi для нашего проекта:

- Ограниченные возможности по обработке видео: хотя Raspberry Pi обладает достаточно мощным процессором, он все же имеет ограниченные возможности по обработке видео, особенно в высоком разрешении и при использовании сложных алгоритмов обработки;
- Ограниченные возможности по хранению данных: Raspberry Pi имеет ограниченные возможности по хранению данных, что может быть проблемой при работе с большим объемом видеоданных;
- Необходимость настройки и наладки: Raspberry Pi требует настройки и наладки перед использованием, что может занять некоторое время и потребовать определенных знаний и навыков в области программирования и работы с аппаратным обеспечением;

Естественно, мы могли бы использовать компьютер – который также является одной из возможных платформ для реализации проекта по распознаванию объектов в видеообразе. Компьютеры, особенно с мощными процессорами и графическими картами, могут обрабатывать видеопотоки с высоким разрешением и обеспечить быструю обработку кадров. Однако, использование компьютера ограничивает мобильность проекта, поскольку он требует постоянного подключения к питанию и монитору.

Рассмотрим наглядную таблицу для сравнения аппаратных платформ.

Таблица 1.2.1. Сравнение аппаратных платформ

Аппаратная платформа	Arduino	Смартфон	Raspberry Pi	Компьютер
Энергопотребление	Низкое	Среднее	Среднее	Высокая
Мощность	Низкая	Средняя	Средняя	Высокая
Гибкость настройки	Ограниченная	Средняя	Высокая	Высокая
Удобство использования	Низкое	Среднее	Среднее	Высокое
Стоимость	Низкая	Высокая	Средняя	Высокая

Надо учесть, что при выборе платформы для реализации приложения для распознавания объектов в видеообразе необходимо учитывать специфические требования проекта, а также доступность необходимых библиотек и средств разработки.

Исходя из всего вышеперечисленного, Raspberry Pi является лучшей платформой для реализации проекта по распознаванию объектов в видеообразе. Это связано с тем, что устройство обладает достаточной мощностью, поддерживает большое количество различных библиотек и API, и обладает низким энергопотреблением по сравнению с обычным компьютером.

Кроме того, Raspberry Pi имеет удобный интерфейс для подключения различных устройств и датчиков, что позволяет легко интегрировать его в систему распознавания объектов.

Таким образом, использование Raspberry Pi позволит создать надежную и эффективную систему распознавания объектов в видеообразе.

2. Разработка электрических схем

Структурная электрическая схема представляет собой графическое изображение электрических компонентов и их связей в системе. Она помогает понять, как устройства взаимодействуют друг с другом и как сигналы передаются между ними.

2.1 Разработка структурной схемы

2.1.1 Обоснование базовых блоков структурной электрической схемы

Рассмотрим каждый из блоков, которые используется в структурной электрической схеме, приведённой в Приложении А.

1. Raspberry Pi 3 Model B – это мини-компьютер, основанный на процессоре ARM Cortex-A53 с тактовой частотой 1,2ГГц и 1 ГБ оперативной памяти. В данном проекте он используется в качестве вычислительной платформы для обработки видеопотока и выполнения алгоритмов компьютерного зрения. Преимущество использования данного мини-компьютера заключается в его достаточной вычислительной мощности, наличии необходимых портов для подключения дополнительного оборудования и возможности использования широкого спектра программного обеспечения для обработки видео.

Рис. 2.1.1. Raspberry Pi 3 Model B

RPi работает на операционной системе Linux. В RPI доступны порты USB, Ethernet, HDMI, аудиовыход, GPIO и другие. Также в RPI Model B есть модули Wi-Fi и Bluetooth.

Одним из главных преимуществ Raspberry Pi является его открытая архитектура, что означает, что любой может создавать свои собственные приложения, модифицировать существующие, создавать свои собственные

модули и расширения. Это также позволяет Raspberry Pi работать с множеством различных операционных систем, включая Raspbian (основанную на Debian), Ubuntu, Windows 10 IoT Core и многие другие.

Raspberry Pi 3 Model B предлагает большую гибкость и универсальность для создания различных проектов, таких как управление роботами, автоматизация домашнего хозяйства, создание интернет-приложений, видео-стриминг и многое другое. Он также имеет небольшой размер и низкую стоимость, что делает его доступным для широкой аудитории

2. Raspberry Pi Camera V 2.1 – это камера высокого разрешения, разработанная специально для использования с Raspberry Pi. Камера имеет разрешение 8 мегапикселей и способна записывать видео в разрешении 1080p при 30 кадрах в секунду. Кроме того, камера имеет встроенный модуль автофокусировки, что позволяет получать чёткие изображения на разных расстояниях. Камера подключается через специальный CSI разъем и не требует дополнительного питания, так как получает его от самой платы Raspberry Pi. Камера имеет компактный размер и легко устанавливается на Raspberry Pi, что делает её идеальным выбором для проекта по распознаванию объектов в видеообразе на Raspberry Pi.

Рис. 2.1.2. Raspberry Pi Camera V2.1

3. LCD1602A – это жидкокристаллический дисплей (LCD), который используется для вывода информации. Он имеет две строки по 16 символов в каждой и используется для отображения результатов распознавания объектов. Дисплей имеет встроенный контроллер HD44780, который управляет отображением символов на дисплее. Этот дисплей позволяет отображать текст, числа, символы и пользовательские элементы.

Рис. 2.1.3. Дисплей LCD1602A

4. I2C коннектор – это модуль I2C, который подключается к GPIO-контактам на Raspberry Pi, через два провода, которые передают информацию на дисплей, который подключается к модулю I2C. Также этот модуль позволяет легко управлять дисплеем через Python-скрипт, что упрощает вывод информации на экран.

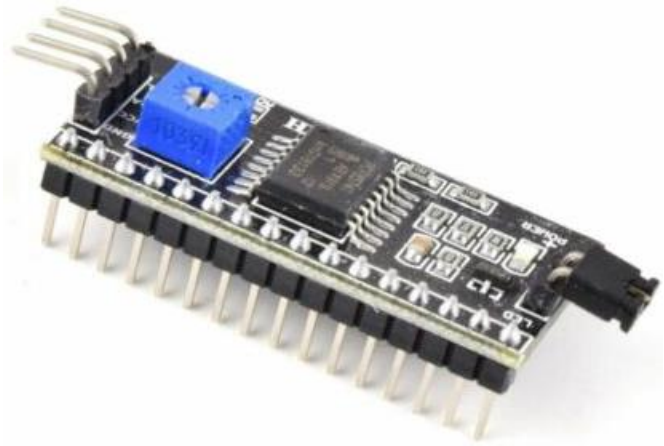


Рис. 2.1.4. I2C коннектор PCF8574

PCF8574 - это расширитель портов I/O, который может управлять до 8-ми цифровыми устройствами, используя только 2 провода I2C (Serial Clock и Data). Он предоставляет дополнительные 8 бит ввода/вывода (I/O) на вашем контроллере. Как правило, он используется в качестве интерфейса между микроконтроллером и LCD-дисплеем, таким как LCD1602A, чтобы управлять выводом на экран.

PCF8574 имеет адрес, который можно программно настроить с помощью пинов A0, A1 и A2. Это позволяет подключить до 8-ми устройств PCF8574 на одну линию I2C. Каждый вывод может быть настроен на вход или выход, и каждый вывод можно установить в состояние высокого или низкого уровня логической единицы.

Использование PCF8574 позволяет сократить количество используемых пинов на микроконтроллере для управления внешними устройствами, что

особенно важно в проектах, где есть ограничение на количество доступных пинов.

При использовании LCD-дисплея в связке с I2C коннектором используется пара проводов для передачи данных SDA и SCL, а также 2 провода для земли и питания соответственно.

2.1.2 Обоснование связей структурной электрической схемы

Для проекта по распознаванию объектов в видеообразе была разработана структурная электрическая схема, в которой использованы различные компоненты, такие как Raspberry Pi 3 Model B, Raspberry Pi Camera V2.1, LCD1602A и PCF8574.

В этой схеме Raspberry Pi 3 Model B является главным устройством, которое выполняет все вычисления и управляет другими компонентами. К Raspberry Pi 3 Model B подключена Raspberry Pi Camera V2.1 через CSI шлейф. Raspberry Pi Camera V2.1 используется для получения видео сигнала, который затем обрабатывается алгоритмами распознавания объектов.

LCD1602A и PCF8574 используются для отображения информации о результате распознавания объектов на экране. PCF8574 является интерфейсом между Raspberry Pi 3 Model B и LCD1602A, обеспечивая передачу данных и управляющих сигналов.

Для подключения к RPi дисплея, используются GPIO-пины. GPIO (General Purpose Input/Output) - это универсальные входы/выходы, которые могут быть использованы для взаимодействия с различными электронными компонентами. Это означает, что вы можете использовать GPIO для управления светодиодами, реле, датчиками и другими электронными устройствами, а также для чтения сигналов с кнопок и других устройств ввода.

LCD1602A идёт спаянный с I2C коннектором, то позволяет упростить подключение к RPI. I2C коннектор общается с RPI при помощи протокола I2C. Raspberry Pi имеет встроенную поддержку данного протокола и настройка соединения через GPIO довольно проста. Практически всю работу на себя берё библиотека smbus.

Таким образом, все компоненты в структурной электрической схеме связаны между собой для эффективной обработки видео сигнала, распознавания объектов и отображения результатов на экране.

Сама схема приведена в приложении А.

2.2 Разработка структурной схемы

2.2.1 Обоснование выбора САПР для разработки принципиальной электрической схемы

Рассмотрим две САПР, которые позволяют разрабатывать принципиальные электрические схемы различных устройств:

KiCAD - это бесплатная и открытая программа для разработки электронных схем и печатных плат (ПП). Она состоит из нескольких инструментов, каждый из которых может использоваться для создания, редактирования и проверки электронных схем и ПП. Некоторые из основных инструментов KiCAD:

- Eeschema: Инструмент для создания и редактирования электронных схем. Он позволяет создавать символы, связывать их между собой и проводить проверку целостности схемы.
- Cvpcb: Инструмент для привязки компонентов электронной схемы к футпринтам на ПП. Это позволяет создавать схему в Eeschema и автоматически генерировать соответствующую печатную плату в Pcbnew.
- Pcbnew: Инструмент для создания и редактирования ПП. Он позволяет размещать компоненты, проводить трассировку, оптимизировать расположение и создавать файлы герберов (для производства ПП).
- GerbView: Инструмент для просмотра файлов герберов, созданных в Pcbnew.
- Bitmap2Component: Инструмент для создания компонентов из изображений в формате BMP или PNG.
- Footprint Editor: Инструмент для создания и редактирования футпринтов компонентов на ПП.

KiCAD также поддерживает многоязычный интерфейс и имеет множество сообществ пользователей и разработчиков, которые активно поддерживают и улучшают эту программу. Кроме того, KiCAD является кроссплатформенным и работает на Windows, Linux и MacOS.

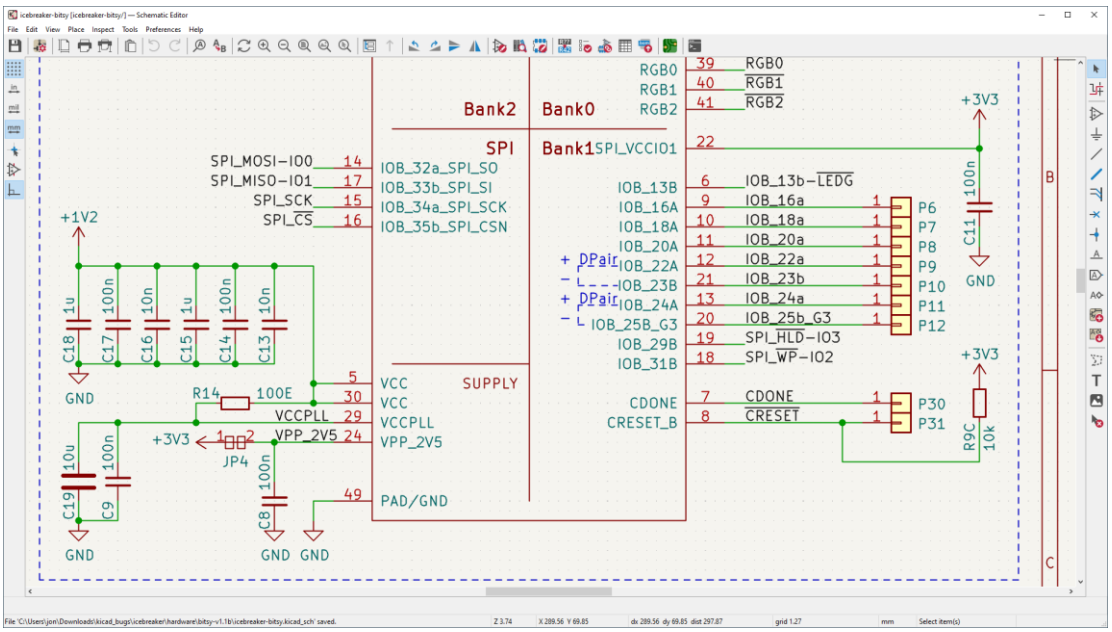


Рис. 2.2.1. Интерфейс САПР KiCAD

Eagle PCB Design — это профессиональный инструмент для проектирования печатных плат, который позволяет создавать схемы, размещать компоненты и трассировать соединения. Программа обладает удобным и интуитивно понятным интерфейсом, который позволяет легко и быстро освоить ее функциональность.

Основные функции Eagle PCB Design включают в себя:

- Создание схемы: программа позволяет создавать схемы электрических схем с использованием широкого спектра компонентов, таких как резисторы, конденсаторы, индуктивности, микросхемы и транзисторы.
- Размещение компонентов: Eagle PCB Design позволяет размещать компоненты на плате, используя автоматическую и ручную трассировку соединений.
- Трассировка соединений: программа позволяет проводить трассировку соединений на плате, включая ручную и автоматическую трассировку.
- Проверка правил проектирования: Eagle PCB Design проверяет правила проектирования на этапе проектирования платы, чтобы убедиться, что все соединения выполнены правильно.
- Экспорт и импорт данных: программа позволяет экспортировать данные в различные форматы, такие как Gerber и DXF, а также импортировать данные из других программ для дальнейшей работы над проектом.
- Создание библиотек компонентов: Eagle PCB Design позволяет создавать пользовательские библиотеки компонентов, что обеспечивает возможность повторного использования компонентов в других проектах.

- Совместная работа: программа поддерживает совместную работу над проектами, что позволяет нескольким пользователям работать над одним проектом одновременно.
- Работа с многослойными платами: Eagle PCB Design позволяет работать с многослойными платами, что обеспечивает возможность создания более сложных проектов.

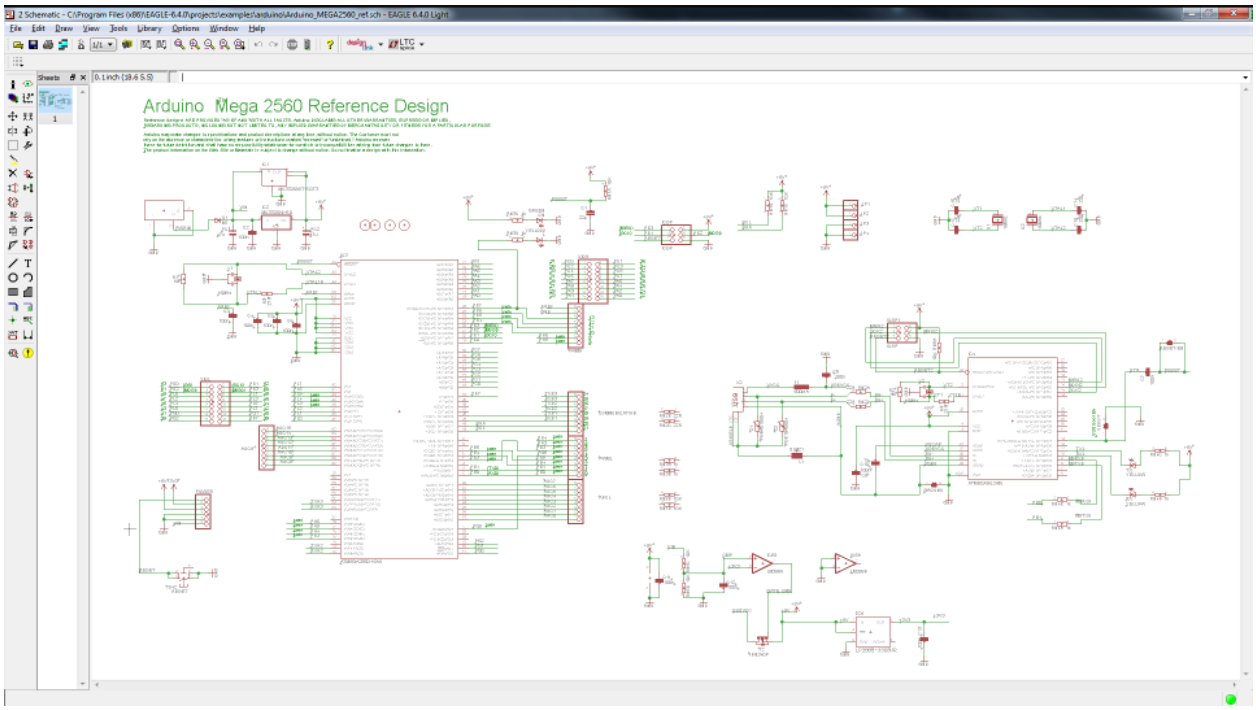


Рис. 2.2.2. Интерфейс САПР Eagle PCB Design

Обе программы предоставляют возможности для создания схем, размещения компонентов на печатной плате и проведения трассировки. Однако, KiCAD имеет несколько преимуществ перед Eagle PCB Design, которые делают его более привлекательным выбором:

- Бесплатность: KiCAD является бесплатным и открытым исходным кодом программным обеспечением, в то время как Eagle PCB Design требует покупки лицензии для полной функциональности.
- Большее сообщество: KiCAD имеет большое и активное сообщество пользователей, которые постоянно вносят улучшения и обновления в программу. Следовательно, у KiCAD больше доступных библиотек компонентов и решений для проблем, возникающих в процессе проектирования.

- Поддержка мультиплатформенности: KiCAD поддерживает Windows, Mac и Linux, тогда как Eagle PCB Design может быть запущен только на Windows и Mac.
- Неограниченные возможности для пользовательского интерфейса: KiCAD позволяет пользователям настраивать пользовательский интерфейс в соответствии с их потребностями и предоставляет более широкие возможности для изменения интерфейса, чем Eagle PCB Design.
- Открытый исходный код: KiCAD является открытым исходным кодом, что позволяет пользователям редактировать и изменять программу в соответствии с их потребностями.

В целом, KiCAD предоставляет больше гибкости, функциональности и свободы для пользователей. Благодаря бесплатности, более широкому сообществу и открытому исходному коду, KiCAD является предпочтительным выбором для моего курсового проекта.

2.2.2 Описание используемых элементов

Компонент RaspberryPi-CM3+ — это высокопроизводительный вычислительный модуль, разработанный компанией Raspberry Pi Foundation на базе микроконтроллера Broadcom BCM2837B0. Он является одним из самых мощных модулей в серии Raspberry Pi и используется в различных проектах для создания компактных устройств с высокой вычислительной мощностью и низким энергопотреблением.

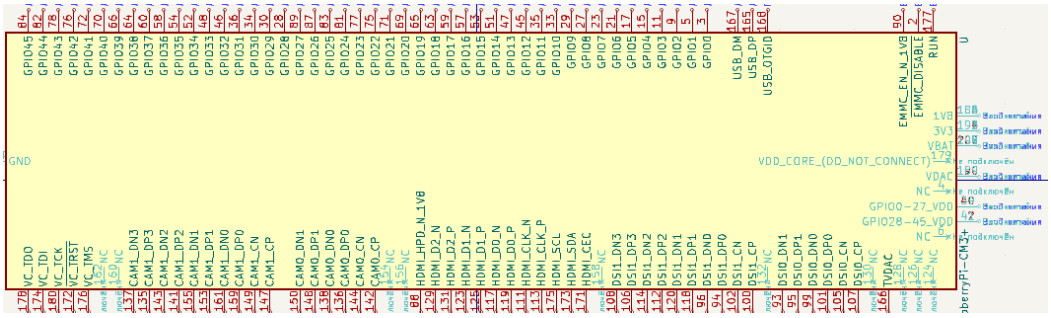


Рис. 2.2.3. Компонент RaspberryPi-CM3+

В KiCad также доступна библиотека компонентов для RaspberryPi-CM3+, которая содержит символы, модели и образцы печатных плат. В этой библиотеке представлены различные варианты RaspberryPi-CM3+, в том числе с разным объемом оперативной памяти и конфигурацией выводов.

В рамках данного проекта, компонент RaspberryPi-CM3+ отлично подходит для создания принципиальной электрической схемы логического

анализатора, так как он полностью соответствует отладочной плате Raspberry Pi Compute Module 3+, которая будет использоваться при дальнейшей сборке реального устройства.

Следующий компонент, который мы рассмотрим, представляет собой I2C коннектор, который можно использовать для подключения периферийных устройств, таких как светодиоды, кнопки, дисплеи и т.д. Компонент PCF8574 из библиотеки Interface_Expansion KiCad — это 8-битный расширитель портов I/O. Он обладает несколькими пинами, которые включают в себя сериальную линию данных SDA и линию тактирования SCL, используемые для передачи данных между микроконтроллером и расширителем портов соответственно. Кроме того, компонент имеет адресные входы A0-A2, которые используются для выбора адреса расширителя портов в системе, и 8 портов ввода-вывода P0-P7, которые можно настроить как входы или выходы с помощью регистров управления. Общая функция компонента заключается в расширении количества портов ввода-вывода, что делает его полезным для управления различными периферийными устройствами.

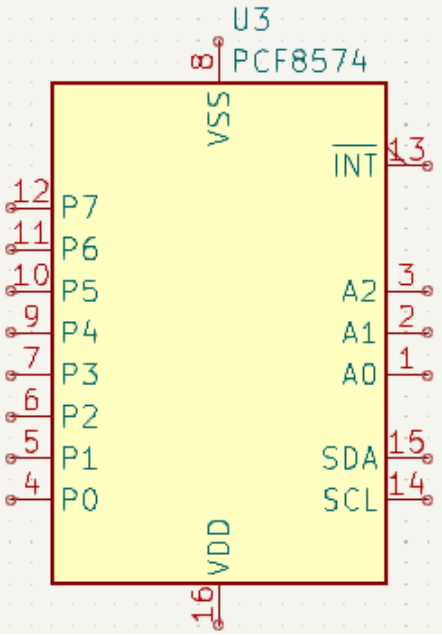


Рис. 2.2.4. Компонент PCF8574

Компонент WC1602A из библиотеки Display_Caracter может быть использован для отображения информации на принципиальной электрической схеме LCD-дисплея. Этот символьный ЖК-дисплей имеет две строки и 16 столбцов, а его контрастность может быть регулирована через переменный резистор, управляемый микроконтроллером.

WC1602A обладает следующими основными характеристиками:

• Размер экрана: 2 строки, 16 символов в каждой

• Интерфейс: параллельный
• Напряжение питания: 5 В
• Угол обзора: 6 часов

Кроме того, WC1602A использует параллельный интерфейс для передачи информации на дисплей, требующий от 6 до 11 пинов микроконтроллера. Компонент также поддерживает обратную связь через сигнал Busy (занятости), который сообщает о готовности дисплея принимать новые данные. Это позволяет микроконтроллеру ожидать освобождения дисплея, прежде чем отправлять новые данные.

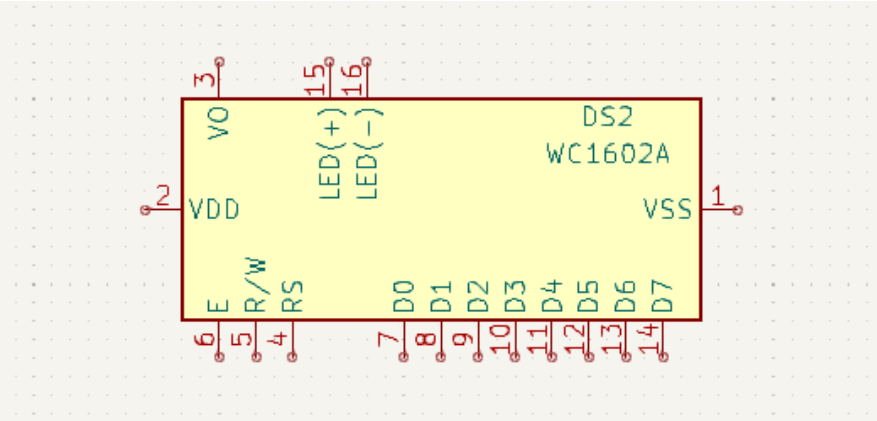


Рис. 2.2.5. Компонент WC1602A

И последний компонент, который был мне необходим для создания принципиальной электрической схемы это – RPi Camera V2.1. Но в стандартном пакете данного компонента не было.

Поэтому мной было принято решение создать собственный компонент, но так как место на чертеже ограниченно, а связь модуля камеры с CSI портом весьма объёмна, я решил абстрагироваться от модуля камеры. В следствии чего в качестве модуля камеры в данной схеме используется CSI_connector, который можно считать за модуль камеры.

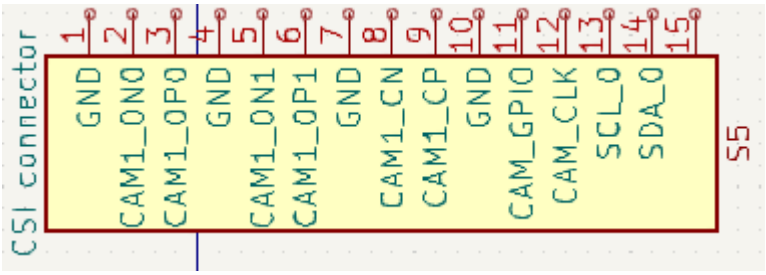


Рис. 2.2.6. Компонент CSI_connector

Сама схема приведена в приложении Б.

3. Разработка алгоритма функционирования распознавания объектов в видеообразе

3.1. Выбор алгоритма функционирования распознавания объектов в видеообразе

С помощью современных технологий видеоанализа можно быстро и удобно обнаруживать объекты в режиме реального времени и осуществлять поиск в архиве по различным критериям. Поисковый критерий может быть основан на размере объекта, его области присутствия в кадре или на приметах человека, включая его лицо. Чем уникальнее и оригинальнее критерий, тем более точные результаты можно получить.

Наиболее точные результаты достигаются при использовании поиска по образцу или по лицам. В данном разделе я выберу алгоритм, позволяющий обнаруживать и распознавать лица людей в кадре, для моего курсового проекта.

Метод Виолы-Джонса – один из способов распознавать образы. Его суть заключается в нахождении контуров объекта и исследовании свойств этого контура. Этот метод использует признаки Хаара, которые придумал венгерский математик Альфред Хаар.

Признаки, также известные как маски, представляют собой геометрические фигуры с черно-белым узором, которые используются для поиска контуров объектов на изображении. Эти фигуры помогают выделить определенные признаки на изображении, такие как очертания лица, линии бровей, носа или рта, что позволяет алгоритмам компьютерного зрения более точно определять объекты на изображении. Например, при поиске лица на изображении используются маски, которые соответствуют особенностям лица, таким как форма глаз, носа и рта, что помогает определить положение и размеры лица на изображении.

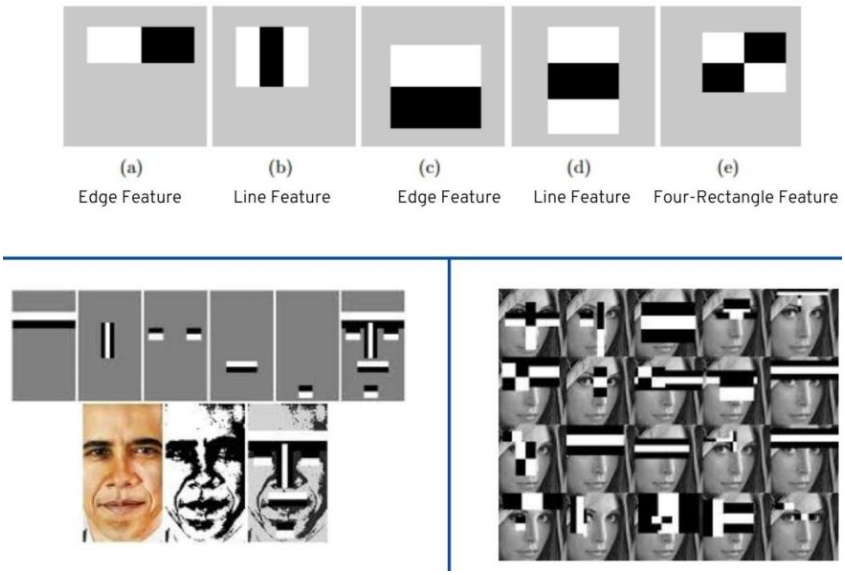


Рис. 3.1.1. Поиск лиц при помощи масок

В алгоритме Виолы-Джонса маски накладываются на разные части кадра, а программа определяет, может ли в них находиться объект. Работает это так:

1. Классификатор (алгоритм, который будет искать объект в кадре) обучают на фотографиях лиц и получают пороговое значение — его превышение будет сигнализировать о том, что в кадре есть лицо;
2. В классификатор загружают изображение, на котором будут искать лицо;
3. Классификатор накладывает на него маски и отдельно складывает яркость пикселей, попавших в белую часть маски, и яркость пикселей, попавших в черную часть маски. Потом из первого значения он вычитает второе;
4. Результат сравнивается с пороговой величиной. Если результат меньше пороговой величины, значит, в части кадра нет лица, и алгоритм заканчивает свою работу. А если больше, он переходит к следующей части кадра;

Реализация обнаружения лиц с использованием каскадов Хаара существует в библиотеке OpenCV. Хоть применение каскадов и является довольно медленной операцией, но в противовес она имеет крайне высокую точность. В этой курсовой работе я буду использовать каскады, только для обнаружения лиц, а распознавание будет проходить при помощи следующего алгоритма.

HOG является одним из наиболее эффективных методов извлечения признаков для распознавания объектов на изображениях. Он основан на подсчете градиентов яркости в каждом пикселе изображения и формировании гистограммы направлений градиентов. Эта информация позволяет выделить контуры и текстуры объектов на изображении.

Использование дескрипторов функции, таких как HOG, позволяет существенно упростить изображение, сохраняя при этом основные черты объектов. HOG может быть применен для распознавания объектов на изображениях различной сложности, включая лица, автомобили, пешеходов и т.д. Разработанный Навнитом Далалом и Биллом Триггсом HOG стал одним из самых широко используемых методов при обработке изображений для задач распознавания объектов.

Дескрипторная гистограмма направленных градиентов использует теорию, которая основана на том, что градиенты интенсивности и направления краев в изображении определяют форму и внешний вид объектов на изображении. По сути, гистограмма направленных градиентов извлекает информацию о том, как интенсивность изменяется в разных направлениях на изображении, что позволяет выделить края и углы, которые содержат больше деталей формы объекта, чем плоские области.

Дескриптор гистограммы направленных градиентов извлекает гистограммы градиентных направлений из изображения, разбивая его на маленькие ячейки. Затем эти гистограммы объединяются в блоки для создания окончательного дескриптора. Такой подход позволяет получить компактное и универсальное представление объектов на изображении, которое может быть использовано в задачах распознавания и классификации объектов.

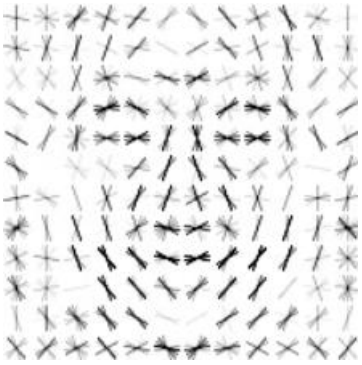


Рис. 3.1.2. HOG лица

В этой курсовой работе HOG дескрипторы я буду использовать для сравнения лиц с базой уже известных, ранее добавленных в базу изображений.

3.2. Описание алгоритма функционирования распознавания объектов в видеообразе

Для ускорения работы нашей программы мной было предпринято решение разбить всё программу на несколько потоков, так как обнаружение и распознавание объектов является ресурсоемкой операцией.

Всего в моей программе работает 3 потока:

1. Поток, отвечающий за считывание данных с камеры и обнаружению лиц;

На этом потоке лежит задачи по считыванию данных с CSI камеры и нахождении количества лиц. Я решил возложить на этот поток задачу по обнаружение лиц чтобы повысить производительность программы, так как операция по обнаружению лиц намного менее ресурсоемкая, чем операция по распознаванию, которая активируется, только при нахождении лиц.

2. Поток, отвечающий за расшифровку и сравнение данных, полученных с камеры;

В этом потоке происходит основная операция, данной курсовой работы, распознавание объектов в видеообразе. Когда первый поток обнаруживает лицо, запускается процесс сравнивания HOG дескрипторов обнаруженного лица и дескриптора из ранее обнаруженных и записанных лиц. После чего поток говорит имя распознанного лица, а в случае, если лицо не распознано выводит “Unknown”.

3. Поток, отвечающий за вывод полученных после расшифровки данных на дисплей и запись данных в файл;

Этот поток, проверяет появились ли новые распознанные лица, и в случае успеха выводит данные на дисплей в формате “data: D.M time: H:M”. Кроме того, этот поток, проверяет изменился ли текущий программный час, и в случае его изменения записывает все данные, которые выводились на дисплей в файл.

3.3. Выбор IDE и языка программирования для написания алгоритма

RPi является мини-компьютером из-за этого его можно программировать на любом языке. Но в RPi есть предустановленные языки, которые имеют большую пользовательскую базу библиотек и поддержку при решении возникающих проблем. Рассмотрим и выберем для создания проекта по распознаванию лиц один из предустановленных языков:

- Python - тот язык является довольно простым высокоуровневым языком программирования, который поддерживает большое количество библиотек, написанных как на Python, так и на других языках для управления различными модулями и компонентами. Этот язык является интерпретируемым, что говорит нам о не столь большой скорости работы, но для написания кода на разных платформах, это убирает необходимость компилировать код для каждой платформы. Факт того, что он является интерпретируемым языком, позволяет облегчить разработку, так как можно писать код на ПК и после этого, сразу запускать написанный или исправленный код на RPi избегая процесс компиляции.
- C/C++ - это низкоуровневые компилируемые языки программирования, которые дают максимальный контроль над процессом и высокую скорость работы. Из-за своей низкоуровневой системы разработка алгоритма на нём является довольно сложным процессом. C++ имеет также большое количество библиотек для работы с электронными компонентами. В силу своей сложности и компилируемой природы мне он не подошел для реализации моего проекта.
- Scratch – является простым визуальным языком программирования, который позволяет молодым людям создавать игры и анимации. Большим плюсом является простота этого языка. Но для создания эффективного алгоритма по распознаванию объектов он не подходит из-за своей низкой эффективности, относительно малого количества библиотек и самым важным недостатком для меня является тот факт, что он является визуальным языком, что не удобно для создания большого и алгоритмически сложного проекта.

Исходя из всего описанного для реализации проекта по распознаванию объектов в видеообразе я выбрал Python в силу того, что он является интерпретируемым и имеет большое количество библиотек для работы с электрическими компонентами.

Так как RPi является мини компьютером нет необходимости писать код в специализированном IDE, как для Arduino. Для написания алгоритма на Python существует большое количество IDE, рассмотрим некоторые из них:

- PyCharm – кроссплатформенная IDE, работающая на Windows, Linux и MacOS. Имеет как бесплатную, так и платную версии. Из плюсов: имеет большое количество встроенных функций, большое пользовательское комьюнити и возможность установки фреймворков, плагинов и дополнений. Из минус: медленный запуск, большой вес.
- Spyder – IDE, разработанная для Data Science. В неё встроено множество инструментов для чтения, анализа, обработки данных и реализации проектов, связанных с искусственным интеллектом и машинным обучением.
- IDLE – стандартный программный комплекс, идущий в комплекте с Python. Эта IDE идеальна для постижения азов Python, но для написания серьезных проектов ее функционала недостаточно.
- Thonny – относительно молодая IDE, которая подходит для обучения. Она загружается в комплекте с последней версией Python и сразу готова к работе. У Thonny есть ряд уникальных и полезных для начинающих программистов функций, а также она даёт подсказки при написании кода. Из плюсов: она предустановлена на RPi, не требует дополнительного скачивания Python, упрощенная установка опций и плагинов и небольшой вес. Из недостатков: ограниченные возможности, возможности возникновения ошибок, которые потребуют исправления со стороны разработчика.

В моём проекте я буду использовать сразу две IDE. Одну для написания основного кода на ПК, так как писать код постоянно на RPi не так удобно, как на уже привычном и более производительном ПК. И вторую я буду использовать для редактирования на RPi кода, написанного на ПК.

В качестве IDE для ПК я выбрал PyCharm. Ключевыми факторами при выборе для меня стали: наличие режима отладки, удобство установки библиотек и большое пользовательское комьюнити.

Для редактирования кода на RPi я выбрал Thonny, так как это IDE является предустановленным, занимает мало места на жёстком диске, не требует больших вычислительных ресурсов. Для редактирования кода функционала Thonny более чем достаточно.

3.4 Описание функций алгоритма

Весь проект состоит из двух Python скриптов, они же будут называться алгоритмами. Первый скрипт нужен для того, чтобы создавать файл с известными HOG дескрипторами. Второй скрипт несёт основную функцию алгоритма, распознает и записывает данные.

Разберем первый скрипт по созданию файла с известными HOG дескрипторами:

Начало алгоритма состоит из импортирования библиотек:

```
1  from imutils import paths
2  import face_recognition
3  import pickle
4  import cv2
5  import os
```

Рис. 3.4.1. Импортирование используемых библиотек

Imutils — это удобная библиотека для обработки изображений, которая упрощает многие задачи, связанные с работой с изображениями, такие как изменение размера, поворот, обрезка и т.д.

Face_recognition — это библиотека для распознавания лиц, которая использует алгоритмы глубокого обучения. Она позволяет обнаруживать лица на изображении и определять, какие люди известны, а также позволяет создавать и управлять базой данных изображений лиц.

Pickle — это модуль Python, который позволяет сохранять объекты Python в файлы и загружать их обратно.

Cv2 — это библиотека OpenCV для работы с изображениями и видео. Она включает в себя множество функций для обработки изображений, также может использоваться для обработки видео, включая захват видео с камеры и запись видео на диск.

Os — это модуль Python для работы с операционной системой. Он включает в себя функции для работы с файлами и директориями, изменения текущей директории, запуска других программ и т.д.

После импортирования библиотек идёт создание переменных, которые нам необходимы для реализации алгоритма. В этом алгоритме создаются переменная, в которой хранятся пути к папкам с названием людей, внутри этих папок хранятся фото этих людей, для создания HOG дескрипторов, переменная

в которой будут храниться HOG дескрипторы и переменная с именами людей, чьи дескрипторы были вычислены.

```
8   # в директории Images хранятся папки со всеми изображениями
9   imagePaths = list(paths.list_images('Images'))
10  knownEncodings = []
11  knownNames = []
```

Рис. 3.4.2. Создание переменных

Далее у нас идёт часть, которая проходит по каждой папке из директории Images и создаёт HOG каждого лица, после чего добавляет распознанные дескрипторы и имена в листы.

```
12  # перебираем все папки с изображениями
13  for (i, imagePath) in enumerate(imagePaths):
14      # извлекаем имя человека из названия папки
15      name = imagePath.split(os.path.sep)[-2]
16      # загружаем изображение и конвертируем его из BGR (OpenCV ordering)
17      # в dlib ordering (RGB)
18      image = cv2.imread(imagePath)
19      rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
20      # используем библиотеку Face_recognition для обнаружения лиц
21      boxes = face_recognition.face_locations(rgb, model='hog')
22      # вычисляем эмбеддинги для каждого лица
23      encodings = face_recognition.face_encodings(rgb, boxes)
24      # loop over the encodings
25      for encoding in encodings:
26          knownEncodings.append(encoding)
27          knownNames.append(name)
```

Рис. 3.4.3. Вычисление HOG дескрипторов

И завершающим этапом этого алгоритма является создание файла под названием face_enc с высчитанными для сравнения дескрипторами в формате “encodings: name”.

```
28  # сохраним эмбеддинги вместе с их именами в формате словаря
29  data = {"encodings": knownEncodings, "names": knownNames}
30  # для сохранения данных в файл используем метод pickle
31  f = open("face_enc", "wb")
32  f.write(pickle.dumps(data))
33  f.close()
```

Рис. 3.4.4 Сохранение HOG дескрипторов

Блок-схема первого алгоритма, которая наглядно показывает его работу приведена ниже:

Рис. 3.4.5 Блок-схема первого алгоритма

Второй алгоритм отвечает за безостановочную обработку входящих кадров с камеры и сравнении их с уже известными HOG дескрипторами. Некоторые функции этого алгоритма являются довольно объемными, поэтому я лишь опишу работу некоторых функций.

Начинается этот алгоритм точно также как и прошлый с импортирования библиотек.

```
1  import threading
2  import face_recognition
3  import pickle
4  import cv2
5  import os
6  from picamera.array import PiRGBArray
7  from picamera import PiCamera
8  from RPLCD import *
9  from RPLCD.i2c import CharLCD
10 from datetime import datetime
```

Рис. 3.4.6 Импортирование библиотек для второго алгоритма

Описание таких библиотек как pickle, cv2, os и face_recognition не имеет смысла, так как они уже описаны выше. Опишу библиотеки, которые ещё не встречались нам:

Threading — это модуль Python, который позволяет создавать и управлять потоками выполнения. Он используется для параллельного выполнения кода, что может улучшить производительность приложения.

Picamera — это библиотека Python для управления камерой Raspberry Pi. Она предоставляет API для съемки фотографий и записи видео с RPI-камеры.

RPLCD — это библиотека Python для работы с LCD-дисплеями, которые могут быть подключены к Raspberry Pi. Она предоставляет интерфейс для управления символьными LCD-дисплеями, которые могут использоваться для вывода текстовой информации.

Datetime — это модуль Python для работы с датами и временем. Он предоставляет классы и функции для работы с датами и временем, например, для получения текущего времени или вычисления разницы между двумя датами.

После этого идёт создание переменных, которые будут использоваться при работе алгоритма.

```
# find path of xml file containing haarcascade file
cascPathface = os.path.dirname(cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
# load the harcaascade in the cascade classifier
faceCascade = cv2.CascadeClassifier(cascPathface)
# load the known faces and embeddings saved in last file
data = pickle.loads(open('face_enc', "rb").read())
video_capture = cv2.VideoCapture(0)
ret, frame = video_capture.read()
#Создание объекта для подключения к дисплею
lcd = CharLCD('PCF8574', 0x27)
faces = []
names = []
framebuffer = ['', '',]
```

Рис. 3.4.7 Создание глобальных переменных для второго алгоритма

Здесь создаются переменные для работы алгоритма. “cascPathface” и “faceCascade” нужны для загрузки и работы с классификатором каскадов. После этого считывается файл “face_enc” из первого алгоритма. Далее создаются переменные для захвата камеры, работы с lcd-дисплеем и инициализируются пустые переменные и буфер кадров.

После этого у нас идёт объявление функций, скажу пару слов о некоторых из них.

Первыми идут функции для вывода текста на lcd-дисплей.

```
def write_to_lcd(lcd, framebuffer, num_cols):
    """Write the framebuffer out to the specified LCD."""
    lcd.home()
    for row in framebuffer:
        lcd.write_string(row.ljust(num_cols)[:num_cols])
        lcd.write_string('\r\n')


def output_data(text):
    global framebuffer
    global lcd

    lcd.cursor_pos = (1, 0)

    if len(text)<16:
        lcd.write_string(text)
    for i in range(len(text) - 16 + 1):
        framebuffer[1] = text[i:i+16]
        write_to_lcd(lcd, framebuffer, 16)
    sleep(0.2)
```

Рис. 3.4.8 Объявление функций работы с LCD-дисплеем

Далее идёт функция, которая считывает кадры с камеры и вычисляет количество лиц изображённых, на текущем кадре. Такая функция, мной была названа ‘streamig”.

```
52  def streaming():
53      global frame
54      global faces
55      global names
56      global rawCapture
57
58      ret, frame = video_capture.read()
59
60      #ret, frame = video_capture.read()
61      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
62
63      faces = faceCascade.detectMultiScale(gray,
64                                           scaleFactor=1.1,
65                                           minNeighbors=5,
66                                           minSize=(60, 60),
67                                           flags=cv2.CASCADE_SCALE_IMAGE)
68
```

Рис. 3.4.9 Объявление функции streaming

После этого идёт функция, которая в бесконечном цикле обрабатывает считанные кадры, и если в кадре была обнаружено лицо, тогда идёт процесс создания дескрипторов. После чего происходит сравнение, с уже известными HOG дескрипторами. В случае успеха добавляет имя распознанного лица в пустой лист “names”, если лицо было обнаружено, но распознать его не удалось, ему присваивается имя “Unknown”.

Последняя функция отвечает за вывод данных в удобном для понимания формате: “data: DAY.MOUNTH time: HOUR:MINUTE”. После этого функция смотрит изменился ли текущий час, и если он изменился, то происходит процесс выгрузки данных, данные о всех тех, кто были распознаны как известные и не известные лица сохраняются в файл. Это сделано для освобождения программных ресурсов.

После этого идёт создание и запуск потоков, которые будут работать параллельно с основной программой.

```
155 # создаем объекты потоков
156 encod_thread = threading.Thread(target=recognition, daemon=True)
157 encod_thread.start()
158 write_thread = threading.Thread(target=file_writer, daemon=True)
159 write_thread.start()
```

Рис. 3.4.9 Объявление функции streaming

В самом конце программы запускается бесконечный цикл по считыванию кадров и обнаружению на них лиц. В этом бесконечном цикле запускается функция “streaming”.

```
162  while True:
163      streaming()
```

Рис. 3.4.10 Запуск бесконечного цикла вызова функции “streaming”

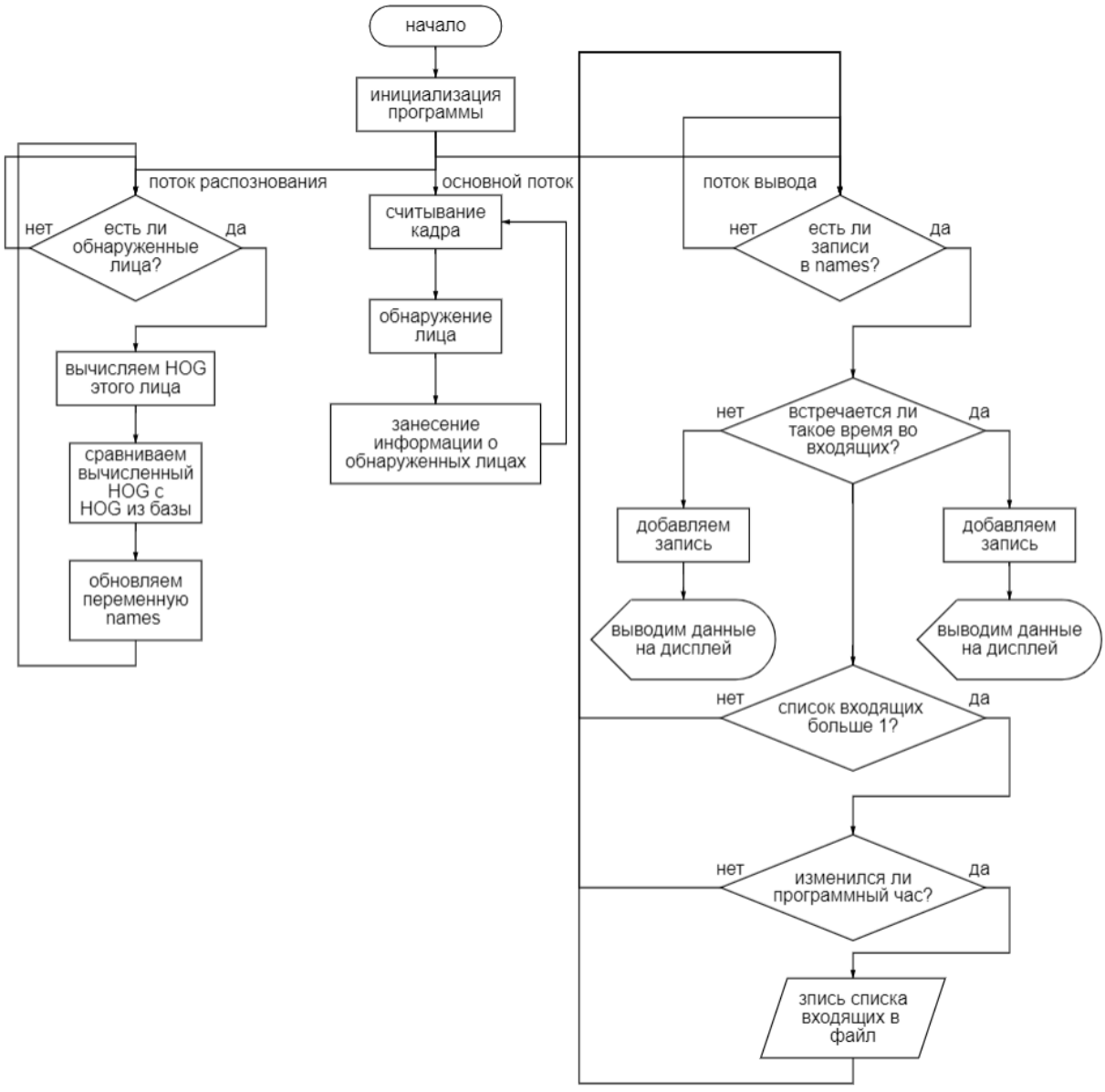


Рис. 3.4.11 Блок-схема второго алгоритма

Заключение

В ходе разработки курсового проекта было проанализирована предметная область. Изучены алгоритмы распознавания объектов, их структура и принцип работы. Также, рассмотрели область их применения. Проведен анализ микроконтроллеров и выбран наиболее оптимальный для проекта.

Во второй главе рассматривались различные САПР для проектирования принципиальной схемы. Была изучена САПР KiCad, с помощью которой была разработана принципиальная схема. Разработана структурная схема устройства в Visio, а также описаны необходимые модули, находящиеся в библиотеке KiCad.

В третьей главе была выбрана IDE для разработки, выбран алгоритм, составлена блок-схема работы устройства, а также описаны некоторые участки кода программы.

Таким образом, в результате выполнения курсового проекта было разработано мобильное устройство распознавания объектов в видеообразе, позволяющий распознавать и обнаруживать лица на кадрах считаных с видео.

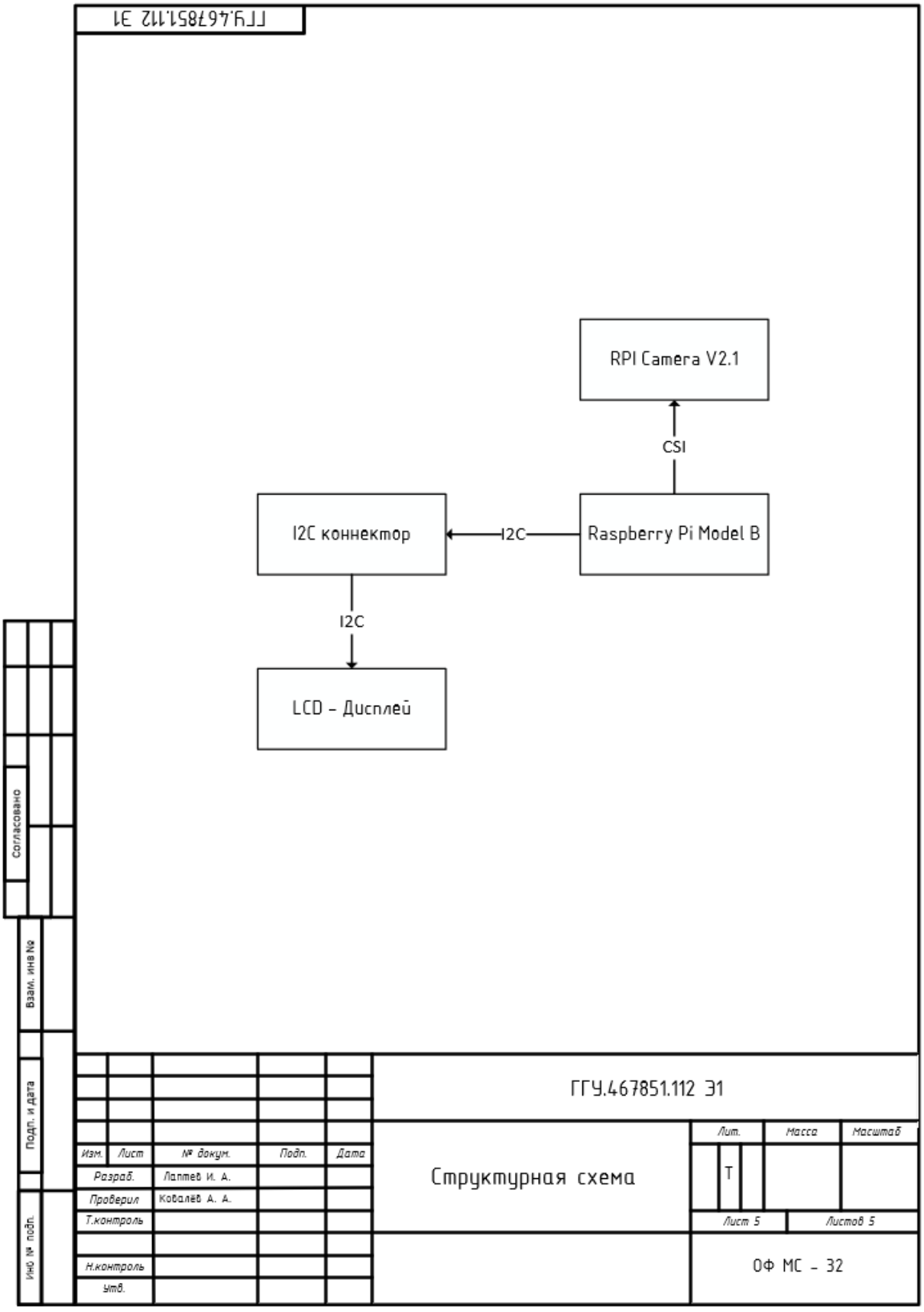
В ходе разработки курсового проекта был опубликован весь необходимый материал (приложения, схемы, программный код) в репозитории на GitHub [6].

Список используемых источников

1. I2C [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/I%C2%B2C. Дата доступа: 25.04.2023.
2. RPi [Электронный ресурс] // Официальная документация по RPi. – URL: https://www.raspberrypi.com/documentation/. Дата доступа: 30.04.2023
3. Python IDE [Электронный ресурс] // Интернет портал. – URL: https://www.hocktraining.com/blog/gde-pisat-kod-na-python. – Дата доступа: 02.05.2023
4. Алгоритм распознавания лиц [Электронный ресурс] // Интернет-портал. – URL: https://blog.skillfactory.ru/kak-napisat-programmu-dlya-raspoznavaniya-licz-2/. Дата доступа: 04.05.2023
5. Работа с LCD-дисплеем [Электронный ресурс] // Интернет-портал. – URL: https://microkontroller.ru/raspberry-pi-projects/podklyuchenie-k-plate-raspberry-pi-4-zhk-displeya-20x4/. Дата доступа: 07.05.2023
6. Листинг кода [Электронный ресурс] // Github – хостинг IT-проектов URL: https://github.com/triteraton/face_recognition

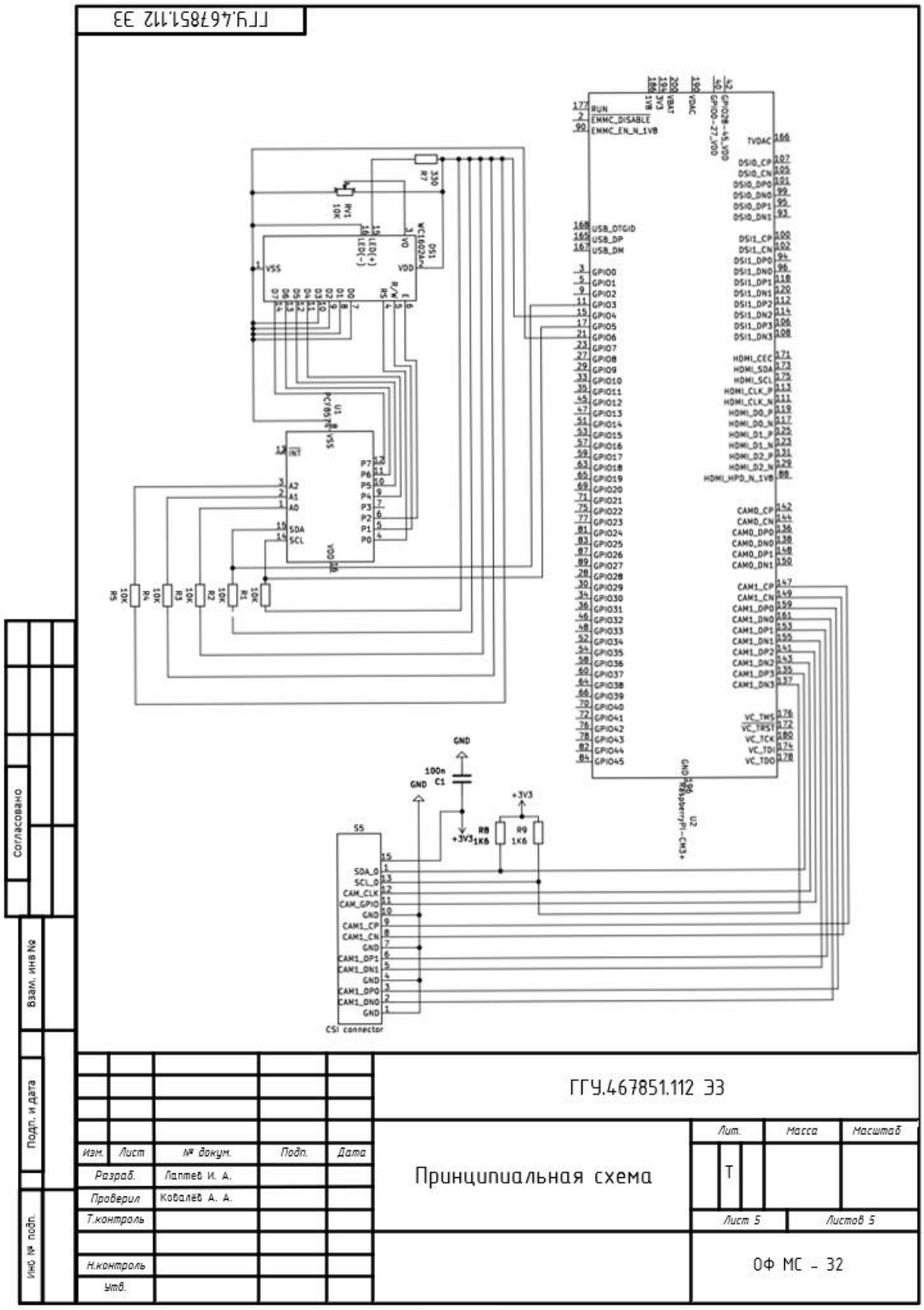
Приложение А

Структурная электрическая схема логического анализатора.



Приложение Б

Принципиальная электрическая схема логического анализатора.



Приложение В

Код алгоритма по созданию базы HOG дескрипторов.

```
from imutils import paths
import face_recognition
import pickle
import cv2
import os


# в директории Images хранятся папки со всеми изображениями
imagePaths = list(paths.list_images('Images'))
knownEncodings = []
knownNames = []
# перебираем все папки с изображениями
for (i, imagePath) in enumerate(imagePaths):
    # извлекаем имя человека из названия папки
    name = imagePath.split(os.path.sep)[-2]
    # загружаем изображение и конвертируем его из BGR (OpenCV ordering)
    # в dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # используем библиотеку Face_recognition для обнаружения лиц
    boxes = face_recognition.face_locations(rgb, model='hog')
    # вычисляем эмбеддинги для каждого лица
    encodings = face_recognition.face_encodings(rgb, boxes)
    # loop over the encodings
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
# сохраним эмбеддинги вместе с их именами в формате словаря
data = {"encodings": knownEncodings, "names": knownNames}
# для сохранения данных в файл используем метод pickle
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

Приложение Г

Код алгоритма для распознавания объектов в видеообразе.

```
import threading
import face_recognition
import pickle
import cv2
import os
from picamera.array import PiRGBArray
from picamera import PiCamera
from RPLCD import *
from RPLCD.i2c import CharLCD
from datetime import datetime


cascPathface = os.path.dirname(cv2.__file__) +
"/data/haarcascade_frontalface_alt2.xml"
faceCascade = cv2.CascadeClassifier(cascPathface)
data = pickle.loads(open('face_enc', "rb").read())

print("Program started")
video_capture = cv2.VideoCapture(0)
ret, frame = video_capture.read()
lcd = CharLCD('PCF8574', 0x27)
faces = []
names = []
framebuffer = ['', '',]


def write_to_lcd(lcd, framebuffer, num_cols):
    lcd.home()
    for row in framebuffer:
        lcd.write_string(row.ljust(num_cols)[:num_cols])
        lcd.write_string('\r\n')


def output_data(text):
    global framebuffer
    global lcd

    lcd.cursor_pos = (1, 0)

    if len(text)<16:
        lcd.write_string(text)
    for i in range(len(text) - 16 + 1):
        framebuffer[1] = text[i:i+16]
        write_to_lcd(lcd, framebuffer, 16)
    sleep(0.2)
    print(text)
```

```
def streaming():
    global frame
    global faces
    global names
    global rawCapture

    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.1,
                                         minNeighbors=5,
                                         minSize=(60, 60),
                                         flags=cv2.CASCADE_SCALE_IMAGE)


def recognition():
    global frame
    global faces

    while True:
        if not len(faces):
            continue
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        encodings = face_recognition.face_encodings(rgb)
        global names
        names = []
        for encoding in encodings:
            matches = face_recognition.compare_faces(data["encodings"],
encoding)
            name = "Unknown"
            if True in matches:
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1
                name = max(counts, key=counts.get)
            names.append(name)




def file_writer():
    global names
    enter = {(0, 0, 0, 0): ['start']}
    current_hour = datetime.now()

    while True:
        current_time = datetime.now()
        time = (current_time.day, current_time.month, current_time.hour,
current_time.minute)
        file_time = "data: " + str(current_time.day) + "." +
str(current_time.month) + " time: " + str(current_time.hour) + ":" +
str(current_time.minute)
```

```
        if len(names) > 0:
            for name in names:
                if time in enter:
                    if name not in enter[time]:
                        enter[time].append(name)
                        temp_str = file_time + " - " + name
                        output_data(temp_str)
                    else:
                        break
                else:
                    enter[time] = [name]
                    temp_str = file_time + " - " + name
                    output_data(temp_str)
        names = []

        if len(enter) > 1:
            if current_time.hour != current_hour.hour:
                file_name = str(current_hour.year) + "_" +
str(current_hour.month) + "_" + str(current_hour.day) + "_" +
str(current_hour.hour) + "_" + str(0) + ".txt"
                with open("log\\" + file_name, 'w') as f:
                    index = 1
                    for key in list(enter.keys())[1:]:
                        key_str = str(key[0]) + "." + str(key[1]) + " " +
str(key[2]) + ":" + str(key[3])
                        for j in enter[key]:
                            output_str = "№" + str(index) + " - data: " +
key_str + " Name :" + str(j) + "\n"
                            f.write(output_str)
                            index += 1
                current_hour = datetime.now()
                enter = {(0, 0, 0, 0): ['start']}


encod_thread = threading.Thread(target=recognition, daemon=True)
encod_thread.start()
write_thread = threading.Thread(target=file_writer, daemon=True)
write_thread.start()


while True:
    streaming()
```