ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В условиях роста цифровизации и распространения концепций умного дома и умных городов безопасность жилых и коммерческих помещений становится одной из ключевых задач. Традиционные механические замки и электронные системы доступа, основанные на ключах, картах или PIN-кодах, всё чаще воспринимаются как устаревшие и уязвимые. Они подвержены рискам потери, кражи или подделки, что создаёт угрозу для сохранности имущества и конфиденциальности пользователей. Биометрические технологии, такие как: распознавание лиц, предлагают принципиально иной уровень защиты, поскольку уникальные физиологические характеристики человека сложнее воспроизвести или передать третьим лицам. Это делает их перспективным решением для обеспечения безопасности в эпоху, когда киберфизические системы интегрируются в повседневную жизнь [1].

Распознавание лиц исключает физический контакт с замком, что повышает удобство и снижает риск передачи инфекций, особенно в общественных и медицинских учреждениях. Доверие к биометрии растёт благодаря её применению в смартфонах, банковских сервисах и госпрограммах [2-3].

Прогресс в области искусственного интеллекта и машинного обучения, включая TensorFlow Lite и OpenCV, позволяет реализовать точные и автономные системы на маломощных устройствах [4-6]. Это делает возможным создание умных замков, которые не зависят от постоянного подключения к интернету.

Тема данной дипломной работы: «Разработка системы распознавания лиц для умного электронного замка на базе Raspberry Pi».

Разработка умного замка на базе Raspberry Pi актуальна для безопасности, интернета вещей и биоаутентификации. Проект сочетает коммерческий потенциал, научно-техническое развитие и социальное значение, отвечая требованиям цифровой трансформации. Использование плат Raspberry Pi для систем распознавания лиц в умных замках обусловлено их доступностью, компактностью и энергоэффективностью. Эти устройства обеспечивают достаточную вычислительную мощность для работы нейронных сетей при низком энергопотреблении, что критично для автономных решений.

Цель работы — разработать систему распознавания лиц для умного электронного замка на базе Raspberry Pi.

Для достижения цели работы необходимо выполнить следующие задачи:

– исследовать существующие методы распознавания лиц;

– проанализировать возможность применения одноплатных компьютеров Raspberry Pi в системах распознавания лиц;

– спроектировать архитектуру системы и алгоритм взаимодействия компонентов системы распознавания лиц;

– осуществить выбор аппаратно-программных компонентов и реализовать систему распознавания лиц;

– протестировать систему и оценить её эффективность.

Объектом исследования в данной работе является системы распознавания лиц.

Предметом исследования является разработка системы распознавания лиц для умного электронного замка на базе Raspberry Pi.

В данной работе были использованы следующие методы научного исследования: теоретический анализ специализированной литературы по теме исследования, сравнительный метод, методы анализа и синтеза, системный подход, а также программирование, моделирование и проектирование схем соединения элементов электрической цепи.

Структура дипломной работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, двух глав, заключения, списка использованных источников и приложений.

Введение раскрывает актуальность выбранной темы, определены объект, предмет, цель, задачи и методы исследования.

В первой главе представлен анализ теоретических аспектов, связанных с разработкой программно-аппаратных средств распознавания лиц. Рассматриваются основные принципы распознавания лиц при помощи алгоритмов машинного обучения и нейросетей. Проводится обоснование выбора одноплатного компьютера Raspberry Pi для реализации дипломной работы.

Вторая глава посвящена практической реализации и разработке системы распознавания лиц на базе Raspberry Pi. Описывается процесс разработки программного обеспечения, включая написание алгоритмов для распознавания лиц, аутентификации пользователей, разработки системы администрирования. Приводится описание этапов сборки и настройки прототипа, включая подключение электронных компонентов, настройку программных утилит и зависимостей, а также тестирование системы.

В заключении подводятся итоги проведенного исследования.

Общий объем работы составляет 61 страниц текста компьютерного набора, в том числе 19 рисунков, 1 таблица. Список использованных источников насчитывает 38 наименований. Количество приложений 6.

ГЛАВА 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ

1.1 Анализ существующих разработок в области систем распознавания лиц

Современные системы безопасности всё чаще интегрируют биометрические технологии, обеспечивая не только удобство, но и повышенную защиту от несанкционированного доступа. Среди биометрических методов распознавание лиц выделяется благодаря своей бесконтактности, удобству и точности работы за счёт алгоритмов машинного обучения. Умные электронные замки, оснащенные системой распознавания лиц, становятся ключевым элементом концепции «умного дома», заменяя традиционные механические замки и пароли. Однако разработка эффективных решений для устройств с ограниченными вычислительными ресурсами, таких как Raspberry Pi, требует тщательного анализа методов оптимизации, аппаратных возможностей и программных инструментов.

При написании данной дипломной работы были использованы научная и учебно-методическая литература, статьи в периодических изданиях Российской Федерации.

Теоретической базой исследования существующих методов распознавания лиц послужили работы: Кирсанова К.О., Евстропова Д.А., Годовникова Е.А., Абдурахимова Н.А., Сметанина С.А., Клемешова В.П., Гаврилина М.С. и Никифорова Д.А., Никифорова М.Б., Волкова Д.А., Евстраткина К.С.

Основными источниками, раскрывающими тему устройства умных электронных замков, являются работы Кирсанова К.О., Евстропова Д.А., Годовникова Е.А. [7-9]. В данных источниках подробно рассмотрено понятие умного электронного замка, особенности работы аппаратной и программной части, а также методы электронного открывания замка. Так в работе Кирсанова К.О. «Разработка системы бесключевого доступа» была разработана система на базе микроконтроллера ESP32, отпирающая электромеханический замок при помощи беспроводного подключения Bluetooth [7]. В работе рассматривается вариант подключения цифровых пинов микроконтроллера непосредственно к отпирающему механизму умного электронного замка.

Для лучшего понимания устройства умных электронных замков обратимся к работе Евстропова Д.А. «Устройство, принцип действия и механизм секрета отдельных видов электронных замков» [8]. В данной работе подробно описан принцип работы электронного биометрического врезного замка, осуществляющего допуск по отпечатку пальца. Как правило, отпирающий механизм умных электронных замков представляет из себя электромеханический привод, приводимый в движение при помощи управляющего сигнала напряжением 3.3 или 5 вольт, в зависимости от логического уровня работы привода.

В ходе анализа научных работ, связанных с разработкой систем распознавания лиц, нами были рассмотрены работы Абдурахимова Н.А., Сметанина С.А., Клемешова В.П., Гаврилина М.С. и Никифорова Д.А. [5, 10-13]. В работе Абдурахимова Н.А. «Проектирование системы распознавания лиц» приведена система, основанная на алгоритме Виолы-Джонса, которая базируется на методе построения каскадного классификатора, каждый уровень которого имеет большее количество проверяемых параметров [10]. Данный способ позволяет очень быстро детектировать лица на изображении, не имея большого количества вычислительных ресурсов. Однако данный подход менее эффективен в сложных условиях, например: при плохом освещении, поворотах головы или частичном закрытии лица.

Сметанин С.А. в своей работе «Разработка системы потокового распознавания лиц» использовал библиотеку компьютерного зрения OpenCV Toolkit, способную в реальном времени детектировать лиц на изображении и выдавать вектор лицевых признаков в 64-мерном пространстве [11]. Благодаря этому подходу автор смог использовать эффективный способ поиска похожих лицевых признаков при помощи алгоритма K-мерных деревьев.

Гаврилин М.С. в работе «Модель машинного обучения при распознавании лиц» приводит пример использования алгоритмов машинного обучения для разработки систем распознавания лиц [5]. Использование нейросетей позволяет добиться высочайшей точности распознавания, но при этом Гаврилин описывает следующие проблемы: проблематичность поиска данных для обучения, сложность проектирования и обучения нейронной сети, необходимость использования высокопроизводительного оборудования для работы системы. Автор явно указывает на сложность поиска качественных и разнообразных данных для обучения нейросетей, что может приводить к переобучению или низкой обобщающей способности модели.

Подробнее всего описан метод применения алгоритмов машинного обучения для систем распознавания лиц в работе Никифорова М.Б. «Исследование степени устойчивости лицевых точек к различным положениям головы в задачах распознавания лиц с использованием библиотеки dlib» [14]. В своей работе автор применял модель свёрточной нейронной сети, предоставляемую библиотекой Dlib. Данная нейронная сеть способна строить маску на основе лицевых признаков. Автор применял данную нейронную сеть при различной постановке кадра, экспозиции и расположении лица в кадре. Автор делает вывод, что: «устойчивость при различных положениях лица наблюдается у точек, имеющих наименьший процент средней ошибки. Поскольку именно эти точки неизменно верно определяли закреплённые за ними структуры лица. Наиболее устойчивыми областями лица являются: глаза, нос, рот и брови. Наименее устойчивой - челюсть». Автор отмечает, что даже при

самой нестандартной постановке кадра, нейронная сеть способна детектировать лицо на изображении и строить лицевую маску.

Проанализировав научные работы других авторов, нами был сделан вывод, что важным аспектом разработки системы распознавания лиц является применение инструментов области компьютерного зрения. Евстраткин К.С. в своей работе «Opencv: варианты использования компьютерного зрения» рассматривает сценарии использования библиотеки Opencv для нормализации изображения и его дальнейшего использования нейросетевыми моделями [15].

Волков Д.А. в своей работе «Обзор программных средств обнаружения объектов с использованием микрокомпьютера Raspberry Pi» описал алгоритм работы клиент-серверной системы с использованием одноплатного компьютера Raspberry Pi [16]. Волков провел сравнительный анализ программных средств, необходимых для обнаружения объектов на изображении, полученного с камеры, подключенной к последовательной шине USB. При этом, в приведенном примере, Raspberry Pi производит лишь первичное обнаружение и выделение объекта на изображении, для его последующей отправки и обработки на высоконагруженном сервере при помощи HTTP запроса.

Нами были изучены работы, связанные с разработкой систем распознавания лиц. После проведенного анализа нами были сделаны следующие выводы:

– алгоритмические методы детекции лиц на изображении имеют большую эффективность, и, при этом, минимально используются вычислительные ресурсы устройства. Несмотря на это, подобные методы проигрывают нейросетевым методам распознавания лиц в сложных условиях съёмки (избыточная или недостаточная освещенность, наличие цифрового шума, нестандартный ракурс съемки и положения лица в кадре);

– создание свёрточной нейронной сети для распознавания лиц с нуля привносит большое количество сложностей, связанных с проектированием и тестированием модели, а также со сбором данных для обучения нейронной сети.

В связи с этим лучше использовать уже существующие эффективные модели для данных задач;

– нейросетевые методы распознавания лиц используют большое количество вычислительных ресурсов устройства, из-за чего стоит рассмотреть сценарий проектирования системы, в которой задачи, связанные с работой нейросетей, могут быть вынесены на удаленный высоконагруженный сервер;

– для использования инструментов компьютерного зрения следует использовать библиотеку OpenCV, так как она имеет большой инструментарий, а также имеет подробную техническую документацию, что может пригодиться в ходе работы.

Для достижения наилучших показателей системы, в ходе написания данной дипломной работы, следует учитывать преимущества и недостатки приведенных работ.

1.2 Исследование применения нейронных сетей и компьютерного зрения для систем распознавания лиц

Для выполнения настоящей дипломной работы нам необходимо было изучить методы применяемые в системах распознавания лиц. Анализ соответствующей литературы показал, что в подобных системах используются алгоритмы машинного обучения и компьютерного зрения.

Машинное обучение, как подмножество искусственного интеллекта, фокусируется на разработке алгоритмов, которые улучшают свои показатели на основе данных без явного алгоритма [17]. Нейронные сети играют ключевую роль в глубоком обучении, где используются многослойные нейронные сети для решения сложных задач. Нейронные сети — это инструменты машинного обучения, которые представляют из себя математические модели, моделирующие работу человеческого мозга, состоящие из слоев нейронов и соединенные между собой. Каждый нейрон принимает входные сигналы, применяет взвешенную сумму и передает результат через функцию активации.

На сегодняшний день наиболее актуальными являются многослойные полносвязные нейронные сети [18]. Подобные сети имеют помимо входного и выходного слоя, определённое количество «скрытых» слоёв. В случае стандартных полносвязных нейронных сетей, подобные слои находятся между входным и выходным слоем. Из-за невозможности повлиять на начальное значение в таких нейронах, слои назвали скрытыми. На сегодняшний день многослойные нейронные сети способны решать практически любую задачу, связанную с обнаружением корреляции входных и выходных данных.

Математически многослойную нейронную сеть можно представить следующей формулой [19]:

$$N\_OUT = f\left(\sum_{i_n} w_{i_n j_n n} \times \cdots \times f\left(\sum_{i_2} w_{i_2 j_2 2} \times f\left(\sum_{i_1} w_{i_1 j_1 1} x_{i_1 j_1 1}\right)\right)\right), \quad (1)$$

где N_OUT - результат работы многослойной нейронной сети;

$f()$ - функция активации нейрона;

i - номер входных данных;

j - номер нейрона в слое;

n - номер слоя;

w - весовой коэффициент;

x - входной сигнал.

Однако, из-за возможности добавления неограниченного количества слоёв и нейронов, возникают случаи переобучения нейронной сети. Данное состояние характерно для задач с большим количеством входных и выходных нейронов, чрезмерно большими или ненормализованными входными данными, а также неочевидной корреляцией между входными и выходными данными. Подобная проблема может быть решена следующими способами [20]:

– нормализация входных данных: входные данные, представленные числовым форматом, не должны быть чрезмерно большими. Максимальный и минимальный порог входных данных может изменяться в зависимости от задачи;

– разработка и тестирование иных архитектур нейронных сетей с меньшим количеством скрытых слоёв и нейронов, иных функций активации и методов вычисления ошибки;

– использование прореживания (dropout). Так как переобучение напрямую связано с количеством нейронов, самостоятельно не оказывающих существенное влияние на итоговое выходное значение, то возникает потребность отключить подобные нейроны при обучении нейронной сети и её последующей эксплуатации.

Для работы с визуальной информацией чаще всего используют свёрточные нейронные сети. Сверточные нейронные сети — это специализированный тип

нейронных сетей, разработанный для обработки данных со сеточной структурой, таких как изображения. Их архитектура включает несколько ключевых компонентов: свёрточные слои, слои субдискретизации, а также полносвязные слои.

Сверточные слои применяют матрицы к входному изображению, создавая карты признаков. Например, ранние слои могут выявлять края, а более глубокие — сложные паттерны, такие как глаза или нос. Фильтр анализирует изображение участками, вычисляя свертку, а затем применяется функция активации для введения нелинейности. Функции активации позволяют нейронной сети находить сложные закономерности в данных, преобразуя сигналы в каждом нейроне так, чтобы сеть могла более эффективно решать задачи нахождения закономерностей. Слои субдискретизации предназначены для понижения размерности изображения. Из блока выбирается значение при помощи заданного алгоритма (к примеру, нахождение максимального, минимального или среднего значения в блоке), при этом остальные значения блока отбрасываются. Это уменьшает размер данных, снижает вычислительную нагрузку и делает сеть устойчивой к цифровому шуму. После нескольких сверточных слоев и слоёв дискретизации данные преобразуются в вектор и передаются в полносвязные слои, где каждая нейронная связь подключена ко всем нейронам предыдущего слоя.

Нейронные сети, особенно свёрточные, эффективно обрабатывают изображения, представляя их в формате матрицы пикселей. Свёрточные нейронные сети извлекают иерархические признаки: начальные слои выявляют низкоуровневые элементы, такие как края объектов и текстуры, а глубокие слои — высокоуровневые концепции, такие как лицо в целом, отдельные особые признаки. Схематичный алгоритм работы свёрточной нейронной сети изображен на рисунке 1.

Для распознавания лиц свёрточные нейронные сети обучаются так, чтобы генерировать эмбеддинги, где лица одного человека близки в пространстве признаков, а разных — далеки. Это достигается с помощью функций потерь,

таких как triplet loss, где для каждого якорного изображения выбирается положительное (то же лицо) и отрицательное (другое лицо), и сеть обучается минимизировать расстояние между якорным и положительным, увеличивая его с отрицательным. Такая особенность выходных данных позволяет гибко работать с поиском и идентификацией людей имея лишь небольшой вектор признаков.

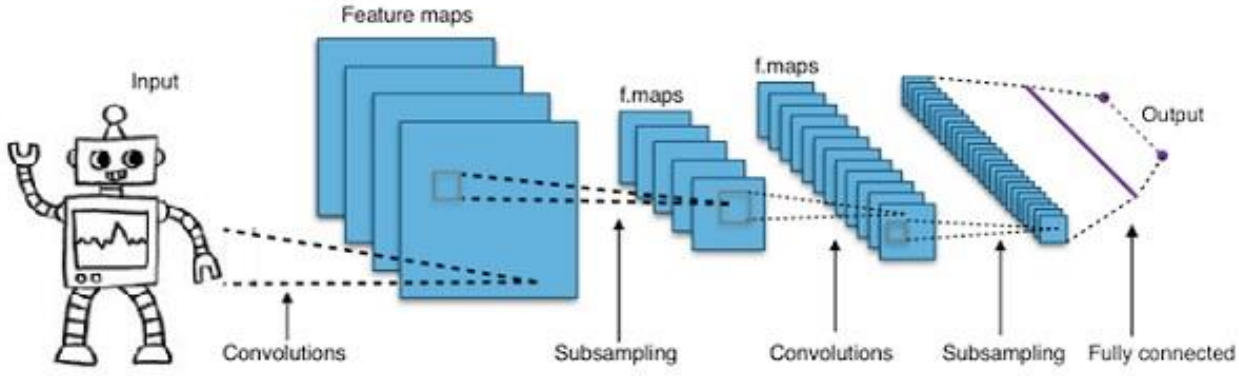

Рисунок 1 - Принцип действия свёрточной нейронной сети*

*Источник: [21]

На сегодняшний день существует большое количество уже спроектированных и обученных нейросетевых моделей, предназначенных для решения широкого спектра задач, включая задачи детекции и распознавания лиц. Данные модели были обучены и протестированы на большом количестве данных из-за чего имеют высокую точность работы. В контексте разработки систем распознавания лиц выделяют нейросетевые модели YOLO, ResNet и FaceNet.

YOLO (You Only Look Once) - это модель нейронной сети для обнаружения и сегментации объектов, представленная в 2015 году [22]. С 2015 года модель YOLO получила множество обновлений и версий, и на данный момент самая актуальная версия модели – YOLO11. YOLO – это свёрточная нейронная сеть, изначально обученная для детектирования 80 классов объектов на изображении. В данные 80 классов входили наиболее распространённые объекты, такие как: человек, транспортные средства, животные, еда, предметы быта и электронные приборы. Для обнаружения объектов YOLO делит полученное изображение по

сетке, получая таким образом множество клеток. Для каждой клетки описываются такие параметры, как ширина и высота, центр клетки, а также задаются вероятности принадлежности объекта в клетке к заданным классам при помощи свёрточной нейронной сети. За счёт того, что свёрточная нейронная сеть работает со множеством небольших участков одного большого изображения, архитектура нейронной сети имеет небольшое количество слоев для понижения размерности, и из-за этого работает быстрее. Благодаря своей общедоступности и хорошо написанной документации, YOLO можно удобно дополнительно обучить на своих данных. Таким образом можно исключить для обнаружения стандартные 80 классов объектов и добавить для обнаружения новый тип объектов. Таким новым классом для обнаружения могут стать человеческие лица. И, с точки зрения разработки системы распознавания лиц, можно обратиться к публичной модели YOLOv11-face, которую обучили на большом количестве изображений человеческих лиц. Данная модель показывает точность детектирования порядка 88%. Данный показатель является хорошим, учитывая тот факт, что YOLO не специализируется на распознавании лиц на изображении.

ResNet, (Residual Neural Network), также представляет собой глубокую свёрточную нейронную сеть, которая была представлена в 2015 году командой исследователей из Microsoft [23]. ResNet была представлена в ответ на проблему, возникающую при обучении больших глубоких нейронных сетей, известную как проблема исчезновения градиента. Эта проблема проявляется в том, что при увеличении количества слоёв в сети точность на обучающем и тестовом наборах данных перестаёт улучшаться и даже может ухудшаться. Ключевым новшеством ResNet является использование соединений быстрого доступа, представляющим собой разницу между входом и желаемым выходом. Это достигается за счёт добавления прямых связей, которые пропускают один или несколько слоёв, позволяя градиенту перенаправляться в обход через эти соединения во время обратного распространения ошибки. Пример работы соединения быстрого доступа изображен на рисунке 2. Существует несколько версий моделей ResNet. Они отличаются количеством слоев нейронной сети: 34, 50, 101 и 152. ResNet

изначально была разработана для задачи классификации изображений, но её применение распространилось на другие области компьютерного зрения, такие как обнаружение объектов и сегментация. Например, ResNet часто используется как базовая модель в более сложных архитектурах. Для задач, связанных с разработкой систем распознавания лиц, ResNet возможно использовать как нейросетевую модель для получения вектора лицевых признаков. В зависимости от архитектуры модели ResNet, в качестве выходных данных можно получить вектор размерностью 512, 1000 и 2048. Однако такая размерность является избыточно большой в системах распознавания лиц, и для использования ResNet в таких системах, необходимо модифицировать модель, добавив дополнительные слои понижения размерности.

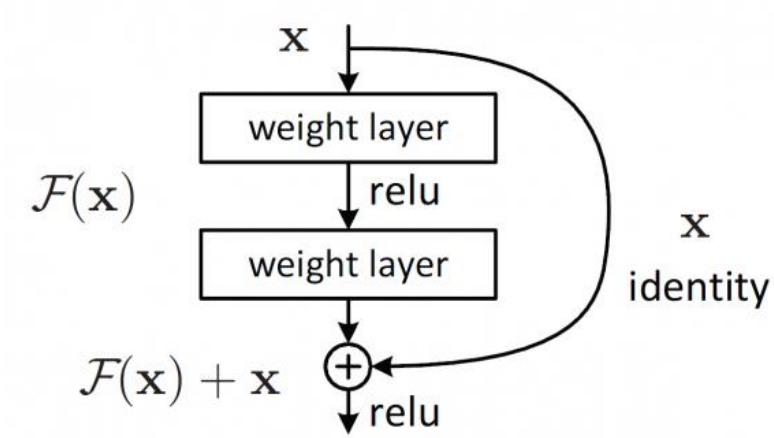

Рисунок 2 – пример соединения быстрого доступа в многослойной нейронной сети ResNet*

*Источник: [23]

FaceNet, также представленная в 2015 году исследователями из Google, является глубокой свёрточной нейронной сетью, разработанной для задач распознавания лиц [24]. Основным достижением FaceNet стало создание компактных векторов лицевых признаков в 128-мерном евклидовом пространстве, где расстояния между векторами напрямую соответствуют степени схожести лиц. Ключевая инновация модели заключается в использовании триплетной функции потерь (triplet loss), которая оптимизирует

обучение, минимизируя расстояние между векторами лицевых признаков одного человека и увеличивая расстояние между векторами разных людей. Это позволяет FaceNet эффективно решать задачи определения личности по изображению. FaceNet достигла рекордной точности 99.63% на датасете Labeled Faces in the Wild, превзойдя многие современные подходы. В контексте разработки систем детектирования и распознавания лиц FaceNet играет ключевую роль как модель для извлечения вектора лицевых признаков. Обычно она используется в связке с алгоритмами детектирования лиц, которые сначала выделяют лица на изображении, после чего FaceNet преобразует их в 128-мерные векторы. Эти вектора затем могут быть использованы для классификации или кластеризации с помощью дополнительных алгоритмов. Кроме того, FaceNet может быть оптимизирована для работы на мобильных устройствах за счёт упрощения архитектуры, что делает её универсальной для различных сценариев применения.

Однако, все приведенные ранее нейросетевые модели значительно нагружают систему, из-за чего их использование в режиме реального времени в системах детектирования и распознавания лиц являет не оптимальным. Использование только нейросетевых подходов может привести к зависаниям системы, повышенному электропотреблению, чрезмерному тепловыделению и быстрому износу аппаратных компонентов системы. По этим причинам, современные системы часто комбинируют применение алгоритмов компьютерного зрения и нейронных сетей. Например, для детектирования лиц на изображении может использоваться высокоэффективный подход с использованием инструментов компьютерного зрения, а для распознавания использоваться свёрточная нейронная сеть. Данный подход обеспечивает надежность и точность системы в условиях работы в режиме реального времени, что критично для систем с низкой производительностью.

Компьютерное зрение — это область, занимающаяся разработкой методов для интерпретации визуальной информации. В распознавании лиц компьютерное зрение предоставляет инструменты для обработки изображений, таких как

фильтрация, пороговая обработка и морфологические операции для улучшения качества данных. Но, помимо этого, существует ряд инструментов области компьютерного зрения, предназначенных для детектирования лиц на изображении. Наиболее распространенные алгоритмы это: каскадный классификатор Хаара и алгоритм HOG.

Алгоритм каскадов Хаара, разработанный Полом Виолой и Майклом Джонсом в 2001 году, представляет собой метод машинного обучения, ориентированный на быстрое обнаружение объектов, особенно лиц, в изображениях и видеопотоках [25]. Этот подход стал важным инструментом в компьютерном зрении благодаря своей способности обеспечивать высокую скорость обработки что делает его подходящим для реального времени. Основой алгоритма являются признаки Хаара — простые прямоугольные паттерны, которые захватывают различия в интенсивности пикселей между смежными областями изображения. Пример поэтапного наложения использования признаков Хаара изображен на рисунке 3. Эти признаки вычисляются эффективно с использованием интегральных изображений, что позволяет алгоритму оценивать тысячи признаков за короткое время, минимизируя вычислительные затраты. В контексте разработки систем обнаружения и распознавания лиц алгоритм каскадов Хаара играет критическую роль на этапе обнаружения лица на изображении. Он используется для локализации лиц в визуальных данных, предоставляя ограничивающие прямоугольники вокруг обнаруженных лиц. Следует отметить, что алгоритм хоть и способен значительно эффективно детектировать лица на изображении, в сравнении с нейросетевыми моделями, алгоритм каскадов Хаара менее точен, чем нейросетевые аналоги. Тем не менее, его ценность сохраняется, особенно в сценариях с ограниченными вычислительными ресурсами, таких как встроенные системы или устройства с низкой производительностью.

Алгоритм HOG (Гистограмма ориентированных градиентов) – это метод описания изображений, который применяется для обнаружения объектов [26]. Алгоритм работает, разделяя изображение на небольшие соединенные области,

называемые ячейками. Для каждого пикселя в ячейке вычисляются градиенты, отражающие изменения интенсивности и ориентации. Затем создаются гистограммы, которые подсчитывают частоту ориентаций градиентов в каждой ячейке. Этот процесс позволяет захватить распределение краев и текстур, что особенно полезно для выделения характерных особенностей лица, таких как контуры глаз, носа и рта. В системах обнаружения лиц алгоритм HOG часто используется в сочетании с классификатором, который определяет вероятность нахождения лица в каждой ячейке. Процесс включает предварительную обработку изображения, разделение на скользящие окна и извлечение HOG-признаков из каждого окна. Эти признаки затем классифицируются для определения, содержит ли окно лицо. За счёт особенностей работы, для того чтобы использовать алгоритм HOG в системах распознавания лиц, необходимо его в значительной степени доработать, добавив классификатор для получения оценки нахождения лица на изображении. Также HOG в значительной мере зависим от качества получаемого изображения. В случае, если в изображении будет присутствовать цифровой шум, будет избыточно или недостаточно экспонированный кадр, все эти факторы могут негативно повлиять на работу алгоритма.

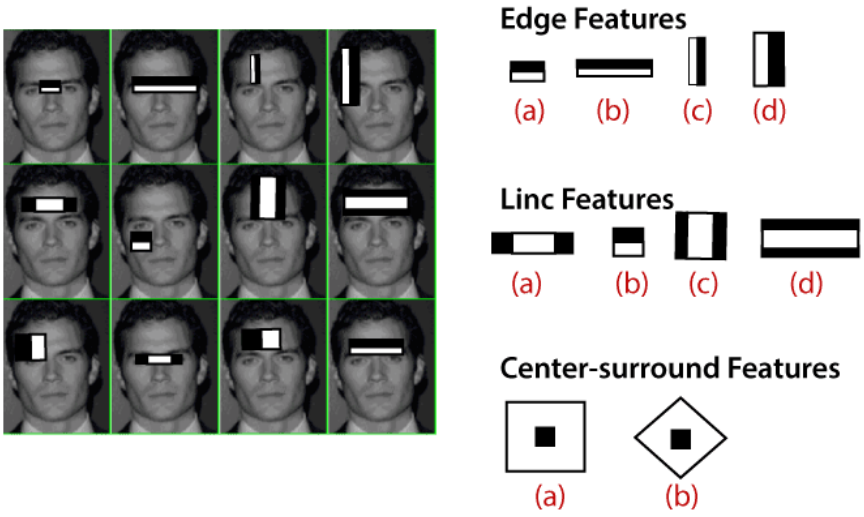

Рисунок 3 – поэтапное использование признаков Хаара в каскадном классификаторе системы распознавания лиц*

*Источник: [25]

1.3 Анализ одноплатных компьютеров и сравнение моделей одноплатных компьютеров Raspberry Pi

Для выполнения настоящей дипломной работы нам необходимо разобрать такое понятие, как одноплатный компьютер, и определить все преимущества и недостатки одноплатных компьютеров Raspberry Pi.

Одноплатные компьютеры представляют собой компактные устройства, объединяющие на одной печатной плате все ключевые компоненты компьютера: процессор, память, хранилище и интерфейсы ввода/вывода. Эти устройства приобрели значительную популярность благодаря своей универсальности, доступности и возможности использования в широком спектре приложений - от образовательных проектов до промышленных систем автоматизации и “Интернета вещей”. В последние годы особый интерес к одноплатным компьютерам проявляется в контексте разработки специализированных систем, таких как системы распознавания лиц, где требуется сочетание вычислительной мощности, компактности и энергоэффективности.

Одноплатные компьютеры обладают рядом преимуществ, которые делают их привлекательными для различных приложений. Во-первых, их компактный размер позволяет легко интегрировать их в ограниченные по пространству среды, такие как встроенные системы или устройства “Интернета вещей”. Это сочетается с низким энергопотреблением, что делает одноплатные компьютеры идеальными для автономных или энергоэффективных решений. Во-вторых, одноплатные компьютеры проще в установке и настройке, поскольку все необходимые компоненты уже интегрированы, что снижает сложность сборки и конфигурации системы.

Еще одно значительное преимущество — их стоимость. Одноплатные компьютеры обычно значительно дешевле полноразмерных компьютеров, что делает их доступными для энтузиастов, студентов и небольших предприятий. При этом они способны запускать полноценные операционные системы, такие как Linux, что расширяет их функциональность. Например, исследования показывают, что одноплатные компьютеры, такие как Raspberry Pi, часто используются для обучения программированию и разработки устройств “Интернета вещей” [27]. Одноплатный компьютер Raspberry Pi 4 B изображен на рисунке 4.

Рисунок 4 - Одноплатный компьютер Raspberry Pi 4 B*

*Источник: выполнено автором

Однако у одноплатных компьютеров есть и недостатки. Основной из них — отсутствие модульности. Поскольку все компоненты находятся на одной плате, пользователи не могут заменить отдельные части, такие как процессор или оперативная память, что ограничивает возможности улучшения системы при необходимости. Кроме того, хотя одноплатные компьютеры достаточно мощны для своих размеров, они не могут конкурировать по производительности с полноразмерными компьютерами. Это может ограничить их применение в задачах, требующих интенсивных вычислений.

При сравнении с полноразмерными компьютерами одноплатные компьютеры выигрывают в компактности, энергоэффективности и стоимости. Полноразмерные компьютеры, такие как настольные персональные компьютеры или ноутбуки, предлагают большую гибкость в плане улучшения и кастомизации, позволяя заменять отдельные компоненты, такие как процессор, графическая карта или оперативная память. Они также обеспечивают более высокую производительность, что необходимо для задач, таких как игровые приложения, редактирование видео или сложные симуляции. Однако их размер, энергопотребление и стоимость делают их менее подходящими для портативных или встроенных приложений.

Реализация систем идентификации личности посредством анализа лицевых биометрических данных предполагает комплексную интеграцию вычислительных ресурсов, коммуникационных интерфейсов и эксплуатационной устойчивости. Подобные системы, функционально ориентированные на обработку визуальной информации, получаемой посредством видеокамер, инициируют алгоритмический анализ изображений с целью локализации и верификации лицевых образов, что обусловливает необходимость последующей архивации или трансляции полученных данных.

При выборе одноплатного компьютера, предназначенного для интеграции в структуру рассматриваемой системы, целесообразно учитывать ряд критически важных факторов. Во-первых, необходимо обеспечить достаточную вычислительную мощность. Поскольку алгоритмы распознавания лиц, особенно в условиях обработки информации в режиме реального времени, отличаются высокой вычислительной сложностью, что подразумевает использование многоядерного процессора с адекватной тактовой частотой. Во-вторых, достаточный объем оперативной памяти (ОЗУ) является обязательным условием для корректной работы программного обеспечения, реализующего функции распознавания лиц, а также для поддержки сопутствующих приложений. В-третьих, необходимо обеспечить достаточную подключаемость, то есть одноплатный компьютер должен обладать необходимыми портами для

сопряжения с устройствами визуального ввода (например, USB или MIPI), а также возможностями для организации хранения данных и сетевого взаимодействия. В-четвертых, наличие адекватной программной поддержки, включающей доступность специализированных библиотек и фреймворков, таких как OpenCV, а также наличие детализированной документации, существенно упрощает процессы разработки и отладки программного обеспечения.

Учитывая эти требования, серия одноплатных компьютеров Raspberry Pi выделяется как популярный выбор благодаря балансу производительности, подключаемости и поддержки сообщества. Однако другие одноплатные компьютеры, такие как Orange Pi, Banana Pi и ASUS Tinker Board, также предлагают конкурентоспособные характеристики. Raspberry Pi выделяется благодаря огромному сообществу, обширной документации и множеству готовых проектов, что упрощает разработку и внедрение.

Для выбора подходящей модели Raspberry Pi для системы распознавания лиц рассмотрим ключевые характеристики популярных моделей. Сравнение параметров наиболее популярных моделей Raspberry Pi приведено в таблице 1.

Таблица 1 - Сравнительная таблица самых популярных моделей одноплатных компьютеров Raspberry Pi

Параметр	Raspberry Pi 2 B	Raspberry Pi 3 B	Raspberry Pi 4 B	Raspberry Pi 5 B
Процессор	900 МГц, 4-ядерный ARM Cortex-A7	1,2 ГГц, 4-ядерный ARM Cortex-A53	1,5 ГГц, 4-ядерный ARM Cortex-A72	2,4 ГГц, 4-ядерный ARM Cortex-A76
ОЗУ	1 ГБ LPDDR2		1/2/4/8 ГБ LPDDR4	2/4/8 ГБ LPDDR4X
Интерфейсы подключения	4x USB 2.0, HDMI, 3,5 мм аудио, Ethernet (100 Мбит/с)		2x USB 3.0, 2x USB 2.0, 2x Micro-HDMI, 3,5 мм аудио, Gigabit Ethernet	
Энергопотребление (среднее)	~3,5 Вт	~4 Вт	~6 Вт	~8 Вт
Физические размеры	85,6 x 56,5 x 17 мм			

Из таблицы видно, что Raspberry Pi 5 предлагает наивысшую производительность, но для многих приложений, включая распознавание лиц, Raspberry Pi 4 B обеспечивает достаточный уровень производительности при более низкой стоимости.

Учитывая требования к системе распознавания лиц, Raspberry Pi 4 Model B выделяется как наиболее подходящий вариант. Он предлагает оптимальное сочетание производительности, памяти и подключаемости. Четырехъядерный процессор ARM Cortex-A72 с тактовой частотой 1,5 ГГц и до 8 ГБ RAM позволяют эффективно обрабатывать алгоритмы распознавания лиц. Два порта USB 3.0 обеспечивают быстрое подключение камер или хранилищ, а гигабитный Ethernet гарантирует стабильное сетевое соединение, что может быть необходимо для интеграции с облачными сервисами.

Кроме того, Raspberry Pi 4 B имеет большое и активное сообщество, что упрощает настройку и отладку системы. Обширные руководства и готовые проекты по распознаванию лиц на основе OpenCV и других библиотек доступны в интернете. По сравнению с Raspberry Pi 5, который дороже и избыточен для большинства задач распознавания лиц, Pi 4 B предлагает лучший баланс цены и производительности. Его компактность также делает его удобным для размещения в различных условиях.

Таким образом, в первой главе данной дипломной работы нами был проведен литературный обзор научных и учебно-методических работ, статей в периодических изданиях Российской Федерации. Был проведен анализ существующих работ, связанных с разработкой систем распознавания лиц. Было подробно описано применение и принцип действия искусственных нейронных сетей и принцип действия свёрточных нейронных сетей. Был проведен анализ инструментов компьютерного зрения для распознавания лиц. В конце главы была поднята тема одноплатных компьютеров, их преимуществ и недостатков, и был проведен сравнительный анализ самых популярных моделей одноплатных компьютеров Raspberry Pi.

ГЛАВА 2 РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ ДЛЯ УМНОГО ЭЛЕКТРОННОГО ЗАМКА НА БАЗЕ RASPBERRY PI

2.1 Проектирование архитектуры и алгоритма взаимодействия компонентов системы распознавания лиц

Нами было проведено исследование теоретических основ, а также был проведен анализ существующих разработок, необходимых для создания системы распознавания лиц. Нами были проанализированы достоинства и недостатки подобных систем и были сделаны выводы, что разработки других авторов имеют ряд недостатков, таких как:

– низкая точность распознавания лиц в сложных условиях съемки из-за отсутствия нейросетевых средств распознавания;

– ограничения, связанные с числом активных лиц в базе системы;

– отсутствие возможности использовать единую базу данных для нескольких электронных замков.

Однако стоит выделить и явные достоинства их работ, такие как:

– совместное использование не нейросетевых методов для детекции лиц на изображении и нейросетевых методов для распознавания лиц. Данный подход минимизирует нагрузку системы;

– использование нейронных сетей, предоставляющих многомерный вектор вероятностных признаков, для их последующего хранения и использования в сравнении похожих лиц;

– проектирование микросервисной архитектуры системы для возможности распределения нагрузки на несколько независимых друг от друга приложений.

Исходя из проведенного анализа достоинств и недостатков, было решено разработать систему, ключевой часть которой является клиент-серверное приложение, в котором сервер предоставляет функционал администрирования системы распознавания лиц, а также предоставляет функционал сравнения полученного изображения лица с лицами, находящимися в базе данных. Клиентом в данной системе выступает устройство, которое при помощи камеры детектирует лица людей на полученном изображении, и, в случае положительного распознавания лица, данное устройство способно подать сигнал на отпирание механизма умного электронного замка. В рамках выполнения настоящей дипломной работы необходимо в первую очередь рассматривать сценарий использования программы системы распознавания лиц, при которой и серверная, и клиентская часть будут функционировать на одноплатном компьютере Raspberry Pi 4 B. Схема взаимодействия элементов системы изображена на рисунке 5.

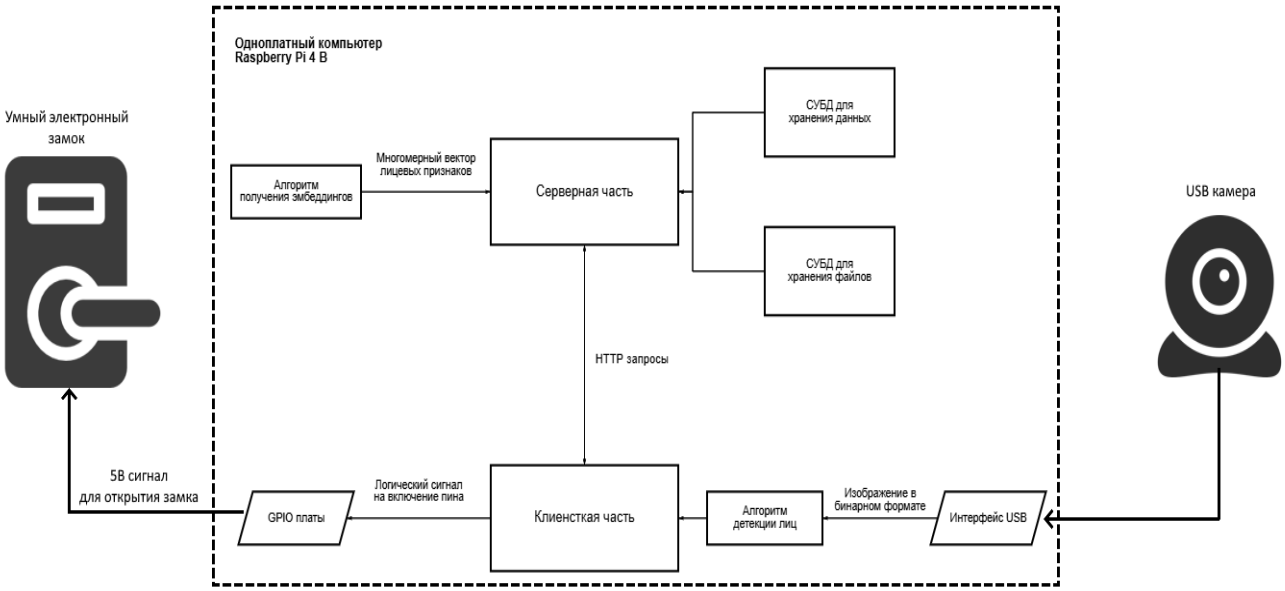

Рисунок 5 - Схема взаимодействия логических и физических элементов системы*

*Источник: выполнено автором

2.2 Подбор необходимых программных и аппаратных средств для проектирования системы распознавания лиц

В ходе реализации системы распознавания лиц для умного электронного замка нам необходимо было совершить подбор устройства умного электронного замка. В качестве подобного устройства нами был выбран электронный замок производства лаборатории «Фаблаб» Крымского Федерального Университета имени В.И. Вернадского. Данный замок имеет функционал доступа при помощи RFID-карт [30-31]. Для связи карты с замком используется специализированный программатор. Для питания замка используются батареи типа «АА». Данные батареи питают электронную плату управления замка, а также используются для питания электромеханического привода замка. На случай утери RFID-карты или при обесточивании замка его возможно открыть при помощи физического ключа. Электронные компоненты замка изображены на рисунке 6.

В ходе анализа принципа работы умного электронного замка производства лаборатории «Фаблаб» нами было определено, что для отпирания механизма замка нам необходимо подать логический сигнал 5 вольт на электромеханический привод. В данном случае подключение происходит параллельно управляющей плате замка, за счёт чего мы сохраним первоначальный функционал работы умного электронного замка. Для подключения замка к одноплатному компьютеру Raspberry Pi 4 B необходимо создать замкнутую цепь, подключенную к цифровому и заземляющему пину платы. Управление цифровым пином будет производиться в клиентской части клиент-серверного приложения системы.

Рисунок 6 – Электронные компоненты умного электронного замка*

*Источник: выполнено автором

Помимо умного электронного замка и одноплатного компьютера Raspberry Pi, для разработки системы нам также необходима малоразмерная камера для обнаружения лиц подходящих к замку людей. Для разработки системы нами была выбрана USB камера Logitech WebCam C170, данная камера представлена на рисунке 7. Камера способна записывать видео размером 640 на 480 пикселей со скоростью 30 кадров в секунду. Модель была выбрана как оптимальная для данной системы, так как большее разрешение видео является избыточным и будет лишь дополнительно нагружать систему, минимально влияя на точность распознавания лиц.

Для разработки клиент-серверной системы необходимо разработать API (набор правил и протоколов, позволяющих программным приложениям взаимодействовать друг с другом). API определяет методы и форматы данных для запросов и ответов, обеспечивая интеграцию различных систем. В разработке веб-API часто используется для обмена данными между клиентом и сервером [30].

Рисунок 7 – малоразмерная USB камера Logitech WebCam C170 *

*Источник: выполнено автором

В качестве протокола общения в данном случае используется HTTP. Протокол имеет набор методов, необходимых для доступа к ресурсам сервера. Выделяют 4 метода, которые используют чаще других:

– метод GET - запрос на получение ресурса. Запросы с использованием этого метода могут только извлекать данные;

– метод POST - используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния на сервере;

– метод PUT - полностью изменяет информацию о ресурсе;

– метод DELETE - удаляет указанный ресурс.

Для удобства последующей установки и запуска приложения следует использовать систему контейнеризации Docker, благодаря которой можно очень удобно запускать приложения с автоматической установкой всех зависимостей, настройкой сетевого взаимодействия и распределением системных ресурсов. К примеру, Docker может быть использован для запуска СУБД MongoDB, что обеспечит изоляцию базы данных от остальной системы и упростит процесс его установки и развертки в системе. Это особенно важно для систем, где ресурсы ограничены, а стабильность критически важна, как в случае с распознаванием лиц в реальном времени.

Для подробной проработки каждой части системы необходимо детально разобрать серверную и клиентскую часть системы распознавания лиц отдельно.

Для разработки серверной части предлагается использоваться язык программирования Python версии 3.9, что обеспечит высокую производительность приложения и упростит процесс разработки. Серверная часть системы распознавания лиц должна включать в себя следующее компоненты: веб-фреймворк, база данных, инструментарий компьютерного зрения, алгоритм распознавания лиц, включающий в себя алгоритм получения вектора лицевых признаков и нахождения похожих векторов.

В качестве основы серверной части системы планируется использовать FastAPI, современный и быстрый веб-фреймворк для создания API, основанный на стандартных типовых подсказках Python [31]. FastAPI обеспечивает автоматическую генерацию документации и высокую скорость выполнения, что важно для работы в режиме реального времени. Для повышения надежности и удобства разработки веб-приложения нами была использована библиотека Pydantic для валидации данных. Pydantic позволяет определять модели данных с типовыми подсказками и автоматически проверять их соответствие, что снижает вероятность возникновения ошибок. В качестве базы данных нами была выбрана документно-ориентированная СУБД MongoDB за её гибкость, скорость и возможность хранения больших файлов с помощью системы GridFS [32]. GridFS разбивает любые файлы на чанки размером до 255 КБ и хранит их как отдельные документы, что позволяет эффективно работать с файлами любого формата, в том числе и с изображениями. Для взаимодействия с базой данных используется PyMongo, официальный драйвер Python для MongoDB.

Для разработки системы распознавания лиц планируется использовать библиотеку Dlib, которая предоставляет предобученную модель на основе модифицированной нейронной сети ResNet [33]. Эта модель обучена на большом наборе данных лиц и извлекает уникальные лицевые признаки, выдавая на выходе вектор в 128 мерном пространстве. Выбор Dlib обусловлен её высокой точностью и простой интеграцией. Для задач, связанных с обработкой изображений, таких как изменение размера или цветового пространства, используется библиотека компьютерного зрения OpenCV [15].

Клиентскую часть, так же, как и серверную, планируется реализовать на языке программирования Python. Клиентская часть, в отличие от серверной, включает в себя не только программные, но и аппаратные компоненты, такие как камера и подключенный по GPIO электронный замок. Для работы клиентской части системы и взаимодействия с аппаратными средствами необходимы следующие компоненты: функционал для получения и обработки изображения с камеры, включая первичное детектирование лиц, функционал отображения визуальной информации, компонент для отправки и обработки HTTP запросов, интерфейс работы с GPIO платы.

OpenCV используется для захвата видеопотока с камеры, подключенной по интерфейсу USB. Камера передает данные в бинарном формате, из-за чего OpenCV приходится интерпретировать подобные данные в своем собственном формате. В процессе интерпретации данная библиотека способна производить первичную обработку изображения, которая включает в себя определение цветового пространства, изменение яркости изображения и его масштабирование. Библиотека OpenCV также предоставляет готовые модели и алгоритмы для детектирования лиц на изображении. Нами планируется использовать алгоритм каскадного классификатора Хаара, позволяющий эффективно выделять области с лицами, без использования требовательных систем на основе свёрточных нейронных сетей. В качестве результата работы классификатора Хаара будет получена область кадра с лицом человека. Данный кадр в последующем будет отправлен на серверную часть для распознавания. OpenCV также обеспечивает отображение видеопотока с наложением информации об обнаруженных лицах, что позволяет пользователю видеть процесс распознавания [15].

Модуль requests будет использоваться для отправки HTTP запросов с изображением на сервер и получения результатов распознавания. Это упрощает взаимодействие между клиентом и сервером, обеспечивая надёжную передачу данных. Клиентская часть также должна быть способна подавать электрический сигнал на цифровые пины для отпирания механизма электронного замка.

Одноплатные компьютеры Raspberry Pi предоставляют функционал взаимодействия с GPIO (интерфейс ввода/вывода общего назначения) [34]. Данные пины представляют из себя металлические ножки, у которых есть либо постоянный функционал, такой как: вывод 3.3 или 5 вольт, пины заземления, либо функционал цифрового входа и выхода (GPIO). Пины GPIO изображены на рисунке 8. Для удобной работы с данными пинами существует модуль для языка программирования Python под названием “RPi.GPIO”, позволяющий внутри Python программы включать и выключать сигнал на определенном пине платы. Следует отметить, что цифровые пины GPIO работают на 5 вольтовой логике, что идеально подходит для взаимодействия с электромеханическим приводом умного электронного замка.

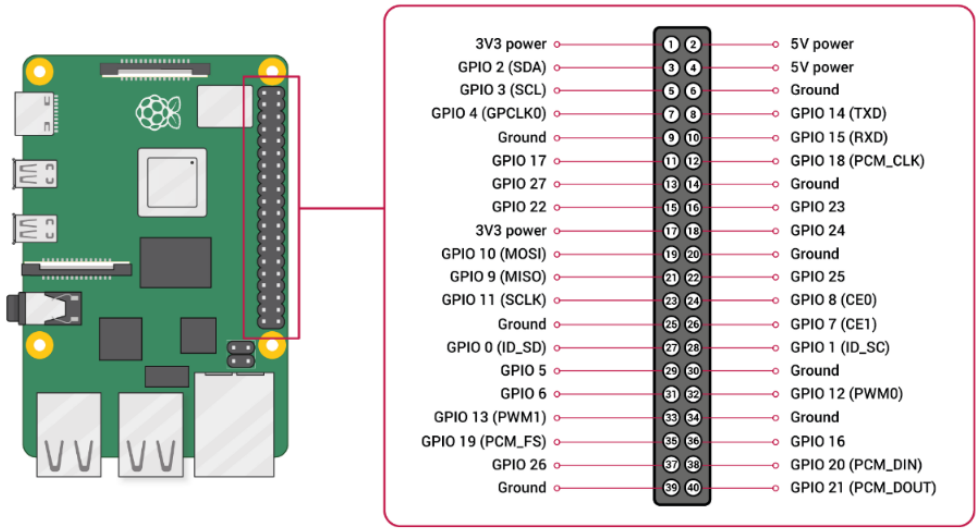

Рисунок 8 - Нумерация и назначение пинов платы Raspberry Pi 4 B*

*Источник: [34]

2.3 Разработка программы серверной части системы распознавания лиц на базе Raspberry Pi

Проработав план разработки системы распознавания лиц на база одноплатного компьютера Raspberry Pi, мы приступили к написанию программы серверной части системы. Разработка производилась на ПК с установленной ОС Ubuntu 24.04 LTS, в среде разработки PyCharm. Алгоритмы установки зависимостей для работы системы в ОС Ubuntu и системы, установленной на одноплатном компьютере Raspberry Pi - идентичны.

Для обеспечения работы приложения, реализованного на языке программирования Python, требуется установка интерпретатора Python. Установка Python версии 3.9 включает обновление системных пакетов, добавление стороннего репозитория (если требуемая версия отсутствует в стандартных репозиториях), установку интерпретатора Python и проверку версии для подтверждения успешной установки. Использование инструмента apt обеспечивает управление зависимостями, а репозиторий deadsnakes предоставляет доступ к различным версиям Python. После установки рекомендуется настроить виртуальное окружение для изоляции проектов.

Процесс начинается с обновления списка пакетов и установки необходимых зависимостей. Затем добавляется репозиторий deadsnakes/ppa, содержащий Python 3.9. После обновления кэша пакетов выполняется установка интерпретатора Python 3.9 и утилиты pip для управления пакетами. Список выполненных bash команд для установки интерпретатора Python версии 3.9 изображен на рисунке 9.

Для удобства установки и запуска СУБД MongoDB нами была установлена система контейнеризации Docker. Docker — это система контейнеризации, позволяющая запускать приложения в изолированных средах. Установка Docker включает добавление официального репозитория Docker, установку пакета docker-ce и настройку прав доступа для текущего пользователя. Запуск контейнера MongoDB требует загрузки образа mongodb из хранилища образов Docker Hub и конфигурации параметров, таких как имя контейнера, порты, переменные окружения для аутентификации и фоновый режим.

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install -y python3.9 python3.9-venv python3.9-dev python3-pip
```

Рисунок 9 - Список использованных команд для установки Python 3.9*

*Источник: составлено автором на основе [35]

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o \
    /usr/share/keyrings/docker-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
    https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install -y docker-ce
sudo usermod -aG docker $USER
sudo systemctl enable docker
sudo systemctl start docker
docker run --name mongodb -d -p 27017:27017 \
    -e MONGO_INITDB_ROOT_USERNAME=user \
    -e MONGO_INITDB_ROOT_PASSWORD=pass \
    mongodb/mongodb-community-server
```

Рисунок 10 - Список использованных команд для установки Docker и запуска СУБД Mongodb*

*Источник: составлено автором на основе [36]

Процесс установки Docker начинается с обновления пакетов и установки зависимостей для работы с HTTPS-репозиториями. Затем добавляется GPG-ключ и репозиторий Docker, после чего устанавливается docker-ce. Пользователь

добавляется в группу docker для выполнения команд без sudo. Для запуска MongoDB используется команда docker run, которая задаёт имя контейнера, порт (27017), переменные окружения для имени пользователя и пароля, а также образ MongoDB. Список использованных команд для установки Docker и запуска образа СУБД MongoDB отображен на рисунке 6.

После установки всех необходимых программ и зависимостей мы перешли к написанию программной части системы распознавания лиц. Начали мы с серверной части. Серверная часть состоит из 4 ключевых элементов:

1. Веб сервер, написанный с использованием веб-фреймворка FastAPI, обрабатывающий входные HTTP запросы. Исходный код представлен в приложении А;
2. Инструменты валидации входных и выходных HTTP запросов. Исходный код представлен в приложении Б;
3. Инструментарий взаимодействия серверной части с СУБД MongoDB при помощи библиотеки Pymongo. Исходный код представлен в приложении В;
4. Набор инструментов распознавания лиц с использованием библиотеки face_recognition. Исходный код представлен в приложении Г.

Серверное приложение, реализованное в файле server.py, представляет собой программную основу системы распознавания лиц, разработанную с использованием фреймворка FastAPI и языка Python. Оно обеспечивает асинхронную обработку HTTP-запросов для выполнения двух ключевых функций: регистрации новых лиц в базе данных MongoDB и поиска соответствий лиц на основе загруженных изображений. Приложение интегрирует обработку изображений с помощью библиотеки OpenCV и извлечение вектора лицевых признаков посредством специализированного модуля распознавания лиц, предположительно основанного на библиотеке face_recognition.

Первая функция, реализованная через эндпоинт «/add_person», предназначена для добавления данных о новом человеке. Код данной функции приведён на рисунке 11. На вход принимаются JSON-данные, содержащие имя пользователя, и файл изображения. После валидации MIME-типа изображения

оно декодируется в массив с использованием OpenCV. Затем извлекается кодировка лица, представляющая собой числовой вектор, описывающий уникальные характеристики лица. При отсутствии лица на изображении возвращается ошибка. Успешно извлечённая кодировка, имя пользователя, имя файла и само изображение сохраняются в базе данных MongoDB. В случае успешного выполнения возвращается подтверждение операции.

```python
@app.post('/add_person')  IgorVolochay
async def add_person(json_data: str = Form(...),
                     img_data: UploadFile = File(...),
                     face_recognition: FaceRecognition = Depends(lambda: face_recognition_worker),
                     mongo: MongoWorker = Depends(lambda: mongo_worker)) -> BaseResponse:
    json_validate = AddPersonJSON(**json.loads(json_data))
    if img_data.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException( status_code: 400, detail="Invalid file type")

    file_data = await img_data.read()
    face_img = cv2.imdecode(np.fromstring(file_data, np.uint8), cv2.IMREAD_COLOR)
    encodings = face_recognition.get_encodings(face_img)
    if not encodings:
        raise HTTPException( status_code: 422, detail="Can't find human face")

    result = mongo.add_person(json_validate.username, encodings[0], img_data.filename, face_img)
    if result.error:
        raise HTTPException( status_code: 422, detail=str(result))
    return BaseResponse(result="Success")
```

Рисунок 11 - Исходный код функции add_person*

*Источник: выполнено автором

Вторая функция, доступная через эндпоинт «/find_by_img», выполняет поиск лица по загруженному изображению. Код данной функции приведён на рисунке 12. Аналогично процессу добавления изображение подвергается проверке формата и декодированию. Извлечённая кодировка лица передаётся в модуль сравнения, который, предположительно, вычисляет метрику схожести (например, евклидово расстояние) между входной кодировкой и кодировками, хранящимися в базе данных. Результат сравнения возвращается клиенту, хотя формат результата (например, идентификатор пользователя или числовое значение расстояния) зависит от реализации метода сравнения.

```python
@app.post('/find_by_img')  IgorVolochay
async def find_by_img(img_data: UploadFile = File(...),
                      face_recognition: FaceRecognition = Depends(lambda: face_recognition_worker),
                      mongo: MongoWorker = Depends(lambda: mongo_worker)) -> BaseResponse:
    if img_data.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException( status_code: 400, detail="Invalid file type")

    file_data = await img_data.read()
    face_img = cv2.imdecode(np.fromstring(file_data, np.uint8), cv2.IMREAD_COLOR)
    encodings = face_recognition.get_encodings(face_img)
    if not encodings:
        raise HTTPException( status_code: 422, detail="Can't find human face")
    res = face_recognition_worker.face_distance(encodings)
    print(res)
    return BaseResponse(result=res)
```

Рисунок 12 - Исходный код функции find_by_img*

*Источник: выполнено автором

Сервер запускается с использованием ASGI-сервера Uvicorn на хосте 0.0.0.0 и порту 5000, обеспечивая доступность для клиентов внутри локальной сети. Для корректной работы серверной части системы было принято решение внедрить систему строгой типизации выходных и выходных HTTP запросов. Файлы base_schemas.py и api_schemas.py содержат определения Pydantic-моделей, которые используются для структурирования и валидации данных в серверной части системы распознавания лиц. Эти модели обеспечивают строгую типизацию, проверку входных и выходных данных, а также единообразное представление структур данных, что упрощает взаимодействие между компонентами системы и API. В base_schemas.py описана модель для представления данных о лице в базе данных, тогда как api_schemas.py содержит модели для обработки запросов и ответов API. Модель Person в base_schemas.py формирует основу для хранения данных о лицах в MongoDB, включая все необходимые метаданные и кодировки с поддержкой сложных типов данных. В api_schemas.py модели AddPersonJSON и BaseResponse управляют входными данными и ответами API, гарантируя их корректность и единообразие, а константа ALLOWED_MIME_TYPES ограничивает форматы загружаемых изображений для повышения безопасности. Эти компоненты совместно

обеспечивают надежную работу системы, минимизируя ошибки и упрощая интеграцию между сервером, базой данных и клиентами.

Зависимости, такие как взаимодействие с MongoDB и обработка кодировок лиц, инкапсулированы в модулях MongoWorker и FaceRecognition, что способствует модульности архитектуры.

Класс MongoWorker, реализованный в файле mongo_worker.py, представляет собой компонент серверной части системы распознавания лиц, обеспечивающий взаимодействие с базой данных MongoDB. Он управляет хранением и извлечением данных о лицах, включая их кодировки, метаданные и изображения, а также предоставляет вспомогательные функции для генерации уникальных идентификаторов и работы с файловой системой GridFS.

Метод инициализации класса MongoWorker отвечает за настройку соединения с базой данных MongoDB и подготовку необходимых ресурсов для работы с данными. Он устанавливает связь с СУБД MongoDB, выбирает целевую базу данных и инициализирует объекты для взаимодействия с коллекциями и файловой системой GridFS. При инициализации происходит загрузка конфигурационных параметров из системного окружения, таких как адрес хоста, порт, имя пользователя и пароль, что обеспечивает безопасное управление учетными данными. Затем создается клиент MongoDB, который подключается к серверу с использованием указанных параметров. Выбирается база данных с именем "Face_recognition", в которой хранятся все данные системы. Далее инициализируются ссылки на коллекции: "persons" для хранения информации о лицах и "counters" для управления счетчиками уникальных идентификаторов. Также создается объект GridFS для работы с файлами, такими как изображения лиц, что позволяет эффективно хранить и извлекать их из базы данных.

Класс MongoWorker включает в себя следующие методы:

– count_persons - метод предназначен для определения общего количества записей о лицах, хранящихся в коллекции "persons". Он возвращает целое число, отражающее текущее число документов в этой коллекции;

– get_and_update_counter обеспечивает получение и увеличение значения счетчика, используемого для генерации уникальных идентификаторов. Он принимает имя счетчика и возвращает его текущее значение после инкрементирования;

– add_person отвечает за добавление новой записи о лице в базу данных, включая имя, вектор лицевых признаков, название изображения и само изображение. Он принимает параметры, описывающие лицо, и возвращает объект ответа, содержащий информацию об успехе или ошибке операции. В случае успешной записи, в базе данный создается документ с записью. Пример такой записи изображен на рисунке 13;

– save_image_to_gridfs предназначен для сохранения изображения лица в файловой системе GridFS. Он принимает на вход изображения в бинарном формате и имя файла, возвращая идентификатор сохраненного файла;

– get_all_encodings извлекает кодировки всех лиц из коллекции "persons" и возвращает их в виде словаря, где ключами являются идентификаторы лиц, а значениями — вектор лицевых признаков;

– retrieve_image_from_gridfs извлекает изображение из GridFS по его идентификатору и возвращает его в виде массива, пригодного для обработки.

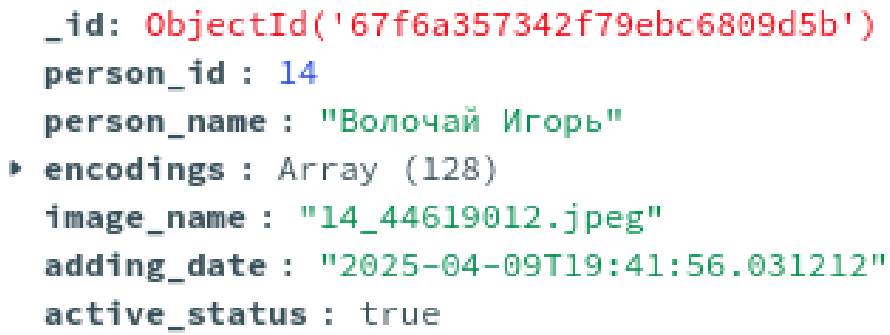

Рисунок 13 - Пример записи в базе данных MongoDB*

*Источник: выполнено автором

Класс FaceRecognition, реализованный в файле face_rec_worker.py, является центральным компонентом системы распознавания лиц, отвечающим за

обработку изображений, генерацию вектора лицевых признаков и их сравнение для идентификации. Он интегрируется с библиотекой face_recognition для извлечения лицевых признаков и использует MongoDB через класс MongoWorker для доступа к сохраненным данным. Класс предоставляет методы для инициализации, извлечения кодировок, нормализации данных и вычисления расстояний между кодировками, применяя метрику KL-дивергенции для точной идентификации.

Метод инициализации класса FaceRecognition создает экземпляр, готовый к обработке изображений и сравнению кодировок, устанавливая связь с базой данных и загружая кэшированные данные. Он создает объект MongoWorker для взаимодействия с MongoDB и запрашивает все кодировки лиц из базы данных, сохраняя их в атрибут face_encodings_in_cache в виде словаря, где ключами являются идентификаторы лиц, а значениями — соответствующие кодировки. Этот кэш используется для ускорения операций сравнения, минимизируя обращения к базе данных во время идентификации. Метод обеспечивает подготовку системы к выполнению задач распознавания, предоставляя доступ к актуальным данным и инфраструктуре для обработки изображений, что делает его основой для всех последующих операций класса.

Метод get_encodings отвечает за генерацию кодировок лица из переданного изображения, используя библиотеку face_recognition для анализа и возвращая их в виде списка числовых векторов. Он принимает изображение в формате, совместимом с OpenCV, и применяет функцию face_encodings из библиотеки, которая обнаруживает лица на изображении и вычисляет 128-мерные векторы, представляющие уникальные характеристики каждого лица. Если на изображении обнаружено одно или несколько лиц, метод возвращает список кодировок; в случае отсутствия лиц возвращается пустой список. Эта функция является ключевой для подготовки данных к последующему сравнению, обеспечивая преобразование визуальной информации в числовой формат, пригодный для математического анализа, и служит основой для операций идентификации в системе.

Метод softmax выполняет нормализацию массива числовых значений, преобразуя их в вероятностное распределение, что необходимо для последующего вычисления KL-дивергенции при сравнении кодировок. Он принимает числовой массив, вычисляет экспоненты каждого элемента, суммирует их и делит каждую экспоненту на полученную сумму, обеспечивая, чтобы итоговые значения находились в диапазоне от 0 до 1, и их сумма равнялась единице. Этот процесс стабилизирует входные данные, устраняя влияние масштаба и делая их пригодными для сравнения в терминах вероятностных распределений. Метод используется как вспомогательная функция в процессе вычисления расстояний между кодировками, повышая точность идентификации за счет приведения данных к единому формату, что особенно важно при работе с метрикой KL-дивергенции.

Метод kl_divergence вычисляет дивергенцию Кульбака-Лейблера между двумя встраиваемых лиц для сравнения их сходства, принимая кодировки из входного изображения и кэшированные кодировки из базы данных, и возвращая массив расстояний [37]. Он сначала проверяет, не является ли входной массив кодировок пустым, возвращая пустой массив в таком случае, чтобы избежать ошибок. Затем кодировки нормализуются с помощью функции softmax, после чего выполняется дополнительная нормализация для обеспечения суммирования значений до единицы. Чтобы предотвратить деление на ноль, к значениям добавляется малое число (эпсилон). Далее вычисляется KL-дивергенция как сумма произведений нормализованных кодировок на логарифм отношения этих кодировок, что позволяет количественно оценить различия между распределениями. Этот метод используется в процессе идентификации для определения степени сходства между лицами, что обеспечивает высокую точность сравнения кодировок.

Метод face_distance выполняет идентификацию лица, сравнивая кодировки из входного изображения с кэшированными кодировками из базы данных, и возвращает идентификатор наиболее подходящего лица или None, если совпадение не найдено. Он начинается с проверки актуальности кэша. А если

количество лиц в базе данных превышает размер кэша, кэш обновляется путем повторной загрузки всех кодировок. Затем для каждого лица в кэше вычисляется KL-дивергенция между входными кодировками и кэшированными данными, формируя словарь расстояний. Метод ищет минимальное расстояние и проверяет, не превышает ли оно пороговое значение (0.001), чтобы исключить ложные совпадения. Если минимальное расстояние удовлетворяет порогу, возвращается идентификатор соответствующего лица. В ином случае возвращается None. В сценарии обработки нескольких кодировок (например, при наличии нескольких лиц на изображении) метод выбирает наилучшее совпадение для каждого лица, обеспечивая гибкость и устойчивость при анализе сложных изображений. Эта функция является ядром процесса распознавания, обеспечивая точную и надежную идентификацию.

Серверная часть системы распознавания лиц, реализованная в предоставленных файлах (server.py, face_rec_worker.py, mongo_worker.py, api_schemas.py, base_schemas.py), представляет собой приложение, разработанное для обработки запросов на добавление новых лиц и их идентификацию на основе изображений через RESTful API. Для запуска серверной части необходимо установить сторонние модули через систему управления пакетами pip. Необходимый набор библиотек и модулей с указанием версий приведен в приложении Д. Такой же список зависимостей находится в текстовом документе requirements.txt, и для установки всех зависимостей необходимо выполнить команду «pip install -r requirements.txt».

Затем необходимо создать переменные окружения для подключения к СУБД MongoDB. Нужно прописать такие переменные окружения как: MONGO_HOST, MONGO_PORT, MONGO_USER, MONGO_PASS. Переменные должны совпадать с теми, что были использованы при запуске СУБД MongoDB через систему контейнеризации Docker. После всего можно запустить серверную часть при помощи команды «python3 server.py».

2.4 Разработка программы клиентской части системы распознавания лиц на базе Raspberry Pi

Для системы была выполнена разработка программы клиентской части системы распознавания лиц на базе Raspberry Pi.

Клиентская часть системы распознавания лиц, реализованная в файле client.py, представляет собой приложение на языке программирования Python, которое взаимодействует с серверной частью (описанной ранее) для распознавания лиц в реальном времени с использованием видеопотока с камеры. Основная цель клиентской части заключается в захвате кадров с камеры, обнаружении лиц, отправке изображений лиц на сервер для идентификации и управлении доступом умного электронного замка на основе результатов распознавания. Система использует OpenCV для обработки видеопотока и обнаружения лиц, библиотеку requests для взаимодействия с серверным API, а также многопоточность для асинхронной обработки запросов и управления дверным механизмом. Клиентская часть интегрируется с сервером через HTTP-запросы, отправляя изображения лиц на эндпоинт /find_by_img и получая идентификаторы распознанных лиц. Она также визуализирует результаты, отображая обнаруженные лица в реальном времени.

Клиентская часть организована вокруг четырех основных классов, каждый из которых выполняет специализированные функции. Класс DoorWorker управляет состоянием GPIO пинов, подключенных к механизму отпирания электронного замка. Класс HttpWorker отвечает за отправку HTTP-запросов на сервер для идентификации лиц, используя блокировку для предотвращения одновременных запросов. Класс RecognitionWorker выполняет обнаружение лиц

на кадрах видеопотока, полученного с камеры, подключенной по интерфейсу USB. Первичная детекция лиц происходит с использованием каскадного классификатора Хаара из стандартного набора инструментов OpenCV. Класс VisualizeWorker визуализирует кадры, рисуя рамки вокруг обнаруженных лиц. Красной рамкой помечаются все обнаруженные лица на изображении, зеленой - те лица, что положительно прошли процесс распознавания на серверной части. Пример визуализации распознавания лиц клиентской частью изображен на рисунке 14. Исходный код программы приведён в приложении Е.

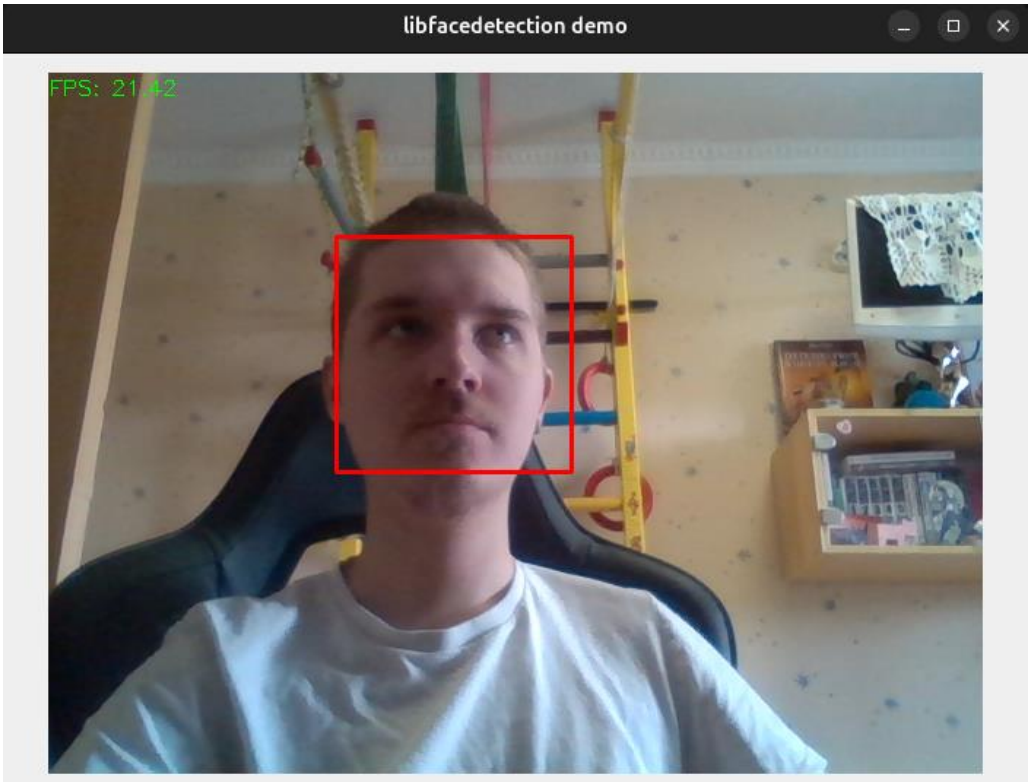

Рисунок 14 - Обнаружение лица в режиме реального времени при помощи каскадного классификатора Хаара*

*Источник: выполнено автором

Клиентская часть выполняет следующие ключевые задачи: она захватывает видеопоток с камеры и обнаруживает лица в каждом кадре с помощью каскадного классификатора Haar. Обнаруженные лица обрезаются, масштабируются и отправляются на сервер через POST-запросы к конечной точке /find_by_img. Сервер возвращает идентификатор распознанного лица, который добавляется в

white_list с информацией о позиции лица в кадре и счетчиком распознаваний. Если лицо распознается несколько раз (не менее трех), оно добавляется в permit_list, что инициирует открытие двери. Визуализация кадров включает отображение рамок вокруг лиц. Дверной механизм активируется в отдельном потоке, производя открытие электронного замка на 5 секунд, после чего списки очищаются, и процесс повторяется. Многопоточная архитектура позволяет параллельно обрабатывать запросы к серверу и управлять дверью, обеспечивая отзывчивость системы.

Многопоточность реализована через стандартную библиотеку python threading. Данная библиотека позволяет отправлять запросы к серверу и управлять дверью без блокировки основного цикла обработки кадров. Блокировка (threading.Lock) в HttpWorker предотвращает одновременные HTTP-запросы, снижая нагрузку на сервер. Белый список (white_list) и список разрешений (permit_list) реализованы как глобальные структуры данных, что упрощает управление состоянием, но требует осторожности при многопоточной обработке. Визуализация выполняется в полноэкранном режиме, что подходит для демонстрационных или стационарных систем контроля доступа.

Отпирание механизма умного электронного замка производится при помощи библиотеки RPi.GPIO. В методе DoorWorker производится инициализация работы с GPIO платы Raspberry Pi и перевод пина под номером 16 в режим GPIO.OUT. Это значит, что на уровне программы можем указать, будет ли данный пин выдавать логическую единицу или ноль. При необходимости открыть замок подается логическая единица, а для того, чтобы запереть замок - логический ноль. Логический уровень, на котором работают GPIO Raspberry Pi, равняется 5 вольтам. Данного напряжение достаточно для работы с замком без использования модулей повышения или понижения напряжения.

2.5 Настройка и тестирование системы распознавания лиц

После того как нами было разработано программное обеспечение для системы распознавания лиц, необходимо было перейти к настройке аппаратной части системы и тестированию её работоспособности. Начать недо с настройки одноплатного компьютера Raspberry Pi 4 B.

Настройка одноплатного компьютера Raspberry Pi 4 B представляет собой многоэтапный процесс, включающий подготовку носителя образа операционной системы, установку операционной системы и интеграцию периферийных устройств. Первым шагом является выбор и подготовка Micro SD карты, которая служит основным хранилищем для Raspberry Pi 4 B. Согласно официальной документации Raspberry Pi [33], для полной версии Raspberry Pi OS с графическим интерфейсом рекомендуется использовать карту объемом не менее 32 ГБ. Нами была использована Micro SD карта объемом 32 Гб от безымянного производителя, с заявленной скоростью записи свыше 10 мегабайт в секунду.

Далее необходимо установить на данный носитель образ операционной системы. Операционная система Raspberry Pi OS, основанная на дистрибутиве Debian, была выбрана для данного проекта из-за ее адаптации под архитектуру Raspberry Pi и наличия графического интерфейса, необходимого для визуального взаимодействия. Образ ОС загружается с официального сайта Raspberry Pi, где доступны различные варианты, включая полную версию с рабочим столом и легковесную версию без него [38]. Нами была установлена полная версия ОС.

Процесс записи образа на Micro SD карту был выполнен с использованием приложения "Диски" на операционной системе Ubuntu Linux 24 LTS. Хотя существуют специализированные инструменты, такие как Raspberry Pi Imager,

которые упрощают процесс и минимизируют ошибки, приложение "Диски" предоставляет стандартный интерфейс для управления дисками, позволяя выбрать образ и целевое устройство. Важно отметить, что перед записью необходимо убедиться, что выбрана именно Micro SD карта, чтобы избежать случайного перезаписи данных на других накопителях. Процесс установки образа на диск изображен на рисунке 15. Результат установки образа на Micro SD карту с автоматически созданной разметкой дискового пространства изображен на рисунке 16.

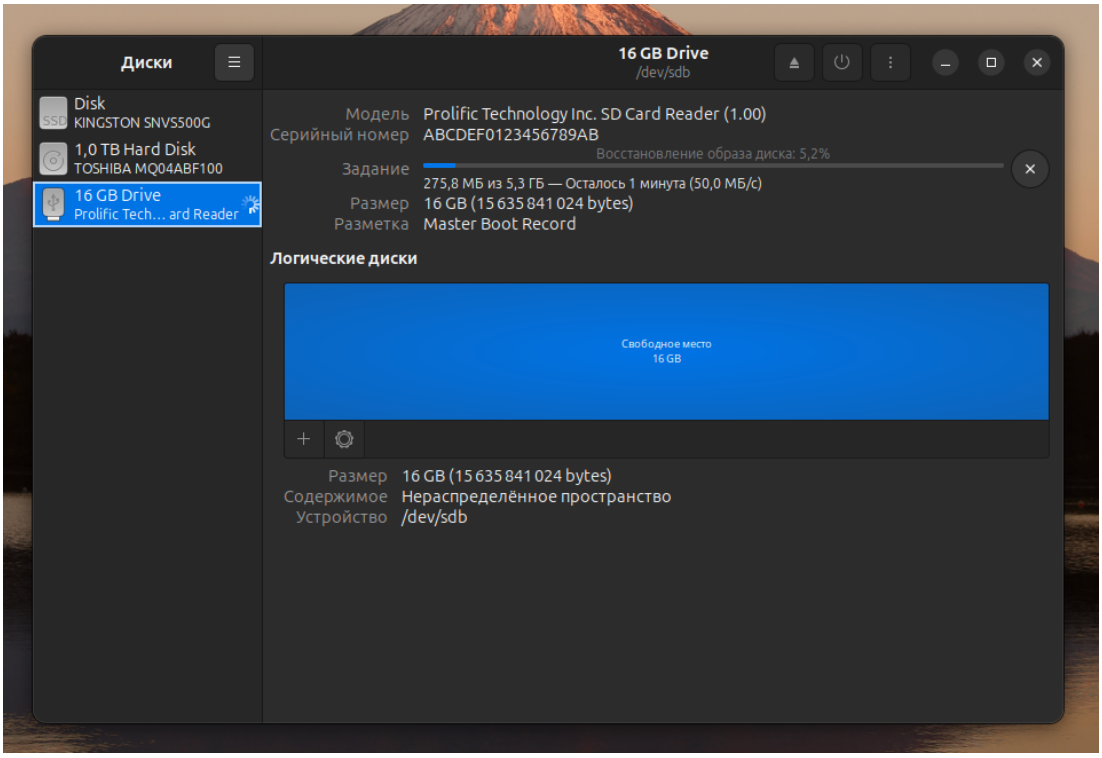

Рисунок 15 - Процесс установки образа Raspberry Pi OS на Micro SD карту*

*Источник: выполнено автором

После подготовки носителя следующим этапом является сборка аппаратной части, включая подключение периферийных устройств. Для получения визуальной информации и взаимодействия с системой был использован монитор с интерфейсом подключения HDMI. Одноплатный компьютер Raspberry Pi 4 B имеет интерфейс подключения Micro HDMI, поэтому

во время работы необходимо было пользоваться переходником для подключения к полноформатному HDMI на мониторе.

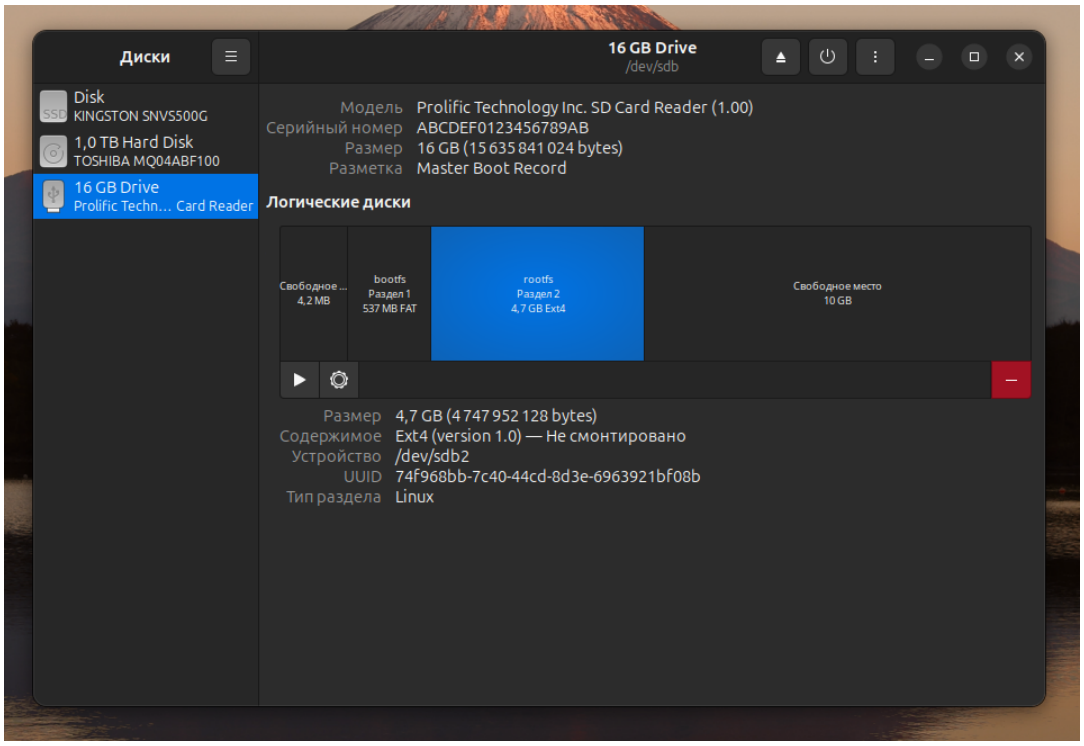

Рисунок 16 - Резульат установки образа Raspberry Pi OS на Micro SD карту*

*Источник: выполнено автором

При первом запуске системы будет предложено провести первичную настройку системы, которая включает в себя: выбор языка системы и языка ввода, выбор временной зоны, наименование пользователя и пароль, настройка подключения к сети Интернет при помощи сетевого подключение через Ethernet или при помощи Wi-Fi.

После первичной настройки системы запускается рабочий стол. Операционная система Rapsberry Pi OS основана на графическом окружении XFCE, которая имеет минималистичный дизайн, не требующий большого количества вычислительных ресурсов. Рабочий стол Raspberry Pi OS с окружением XFCE изображен на рисунке 17. После этого необходимо обновить список репозиториев пакетного менеджера apt при помощи команды apt update и произвести обновление всех системных утилит при помощи команды apt upgrade.

Рисунок 17 - Рабочий стол Raspberry Pi OS*

*Источник: выполнено автором

Далее необходимо произвести установку интерпретатора Python версии 3.9, всех необходимых зависимостей для работы серверной и клиентской части, а также необходимо произвести установку системы контейнеризации docker и произвести запуск образа СУБД MongoDB. Подробное описание процесса установки всех зависимостей для работы системы было приведено в пункте 2.2 настоящей дипломной работы. В ходе установки зависимостей не выявлено никаких изменений в порядке выполнения загрузки, связанных с изменением платформы и операционной системы.

После установки всех зависимостей нами было запущена серверная часть системы распознавания лиц. В ходе работы программы не было выявлено никаких ошибок. Приложение сервера успешно запустилось, штатно работало, принимая HTTP запросы на 5000 сетевом порту. Серверная часть успешно подключилась к СУБД MongoDB, запущенной локально при помощи системы контейнеризации Docker на стандартном сетевом порте 27017.

При помощи предустановленного в системе Raspberry Pi OS браузера FireFox было произведено подключение к системе OpenAPI, по адресу 127.0.0.1:5000/docs. На открывшейся странице нами было произведено взаимодействие с API системы распознавания лиц. Было добавлено лицо человека в базу данных системы при помощи эндпоинта /add_person. В ответ мы получили сообщение, информирующее об успешном добавлении человека в базу системы. Тем самым мы протестировали работоспособность системы веб-сервера FastAPI, работоспособность системы распознавания лиц с использованием Dlib, удостоверились в способности системы взаимодействовать с СУБД MongoDB.

После тестирования работоспособности серверной части нам необходимо было произвести запуск клиентской части системы для проверки работоспособности и анализа эффективности. Перед запуском программной части необходимо произвести подключение камеры по интерфейсу USB. Нами было произведено подключение камеры Logitech Webcam C170. Данная камера производит запись видео в формате 640x480 с частотой 30 кадров в секунду. Разрешения 640x480 пикселей является достаточным для работы системы распознавания лиц, а за счёт небольшого разрешения системе необходимо будет затрачивать меньшее количество вычислительный ресурсов для детекции лиц каждого кадра.

Нами было произведено также подключение GPIO пинов к механизму умного электронного замка. В качестве данного устройства выступал замок производства Лаборатории “Фаблаб” Крымского Федерального Университета имени В.И. Вернадского. Данный замок имеет функционал программатора доступа при помощи RFID-карт [28-29]. Умный электронный замок производства лаборатории “Фаблаб” изображен на рисунке 18. Подключение представляло из себя замкнутую цепь, состоящую из управляющего пина и заземления. Подключение производилось к пину под номером 16 в качестве управляющего и к пину GND в качестве заземления.

Рисунок 18 - Умный электронный замок производства лаборатории «Фаблаб»*

*Источник: выполнено автором

Так как нами было произведено подключение лишь к управляющему механизму замка, для работы всех систем замка, включая работу электромеханического привода запирающего механизма, нам необходимо было запитать весь замок от двух батареек типа “AAA”. После проверки наличия питания замка мы могли приступить к тестированию клиентской части системы распознавания лиц.

Нами был произведен запуск клиентской части системы распознавания лиц для умного электронного замка на базе одноплатного компьютера Raspberry Pi 4B. Параллельно с работой клиентской части на одноплатном компьютере была запущена программа серверной части системы. Во время одновременной работы система показала свою работоспособность, детектируя лица людей на этапе программы клиента и успешно распознавая лицо, находя совпадение с записями в базе данных. На рисунке 19 изображена работа системы, при которой человек в кадре положительно распознается системой и производится отпирание замка.

Нами так же была произведена проверка системы на ложные срабатывания. В ходе данной проверки нами была удалена запись о человеке из базы данных

MongoDB, и была заменена множеством других записей в размере 100 случайных лиц разного пола, возраста и национальности. При такой конфигурации системы был произведен запуск клиентской части и анализ системы на ложные срабатывания. В результате эксперимента система ни разу не совершила ложного срабатывания. Было зафиксировано 53 попытки распознавания лица с разным освещением, ракурсом и позицией лица в кадре. За все попытки ни разу не было найдено ложных совпадений со 100 записями случайных лиц в базе данных системы.

Рисунок 19 - Проверка работоспособности системы распознавания лиц*

*Источник: выполнено автором

Таким образом во второй главе данной дипломной работы нами была разработана система распознавания лиц для умного электронного замка на базе Raspberry Pi. Нами был разработан план проектирования системы и подбор программных и аппаратных компонентов. Было разработано программное обеспечение системы, представляющее из себя клиент-серверное приложение. В конце нами была произведена установка и настройка операционной системы для одноплатного компьютера Raspberry Pi 4 B и произведено тестирование работоспособности системы в различных условиях.

ЗАКЛЮЧЕНИЕ

Данная дипломная работа посвящена разработке системы распознавания лиц для умного электронного замка на базе Raspberry Pi. Актуальность исследования обусловлена ростом цифровизации и необходимостью повышения безопасности в умных домах и городах. Традиционные системы доступа не предоставляют необходимого уровня безопасности и удобства, тогда как биометрические системы распознавания лиц обеспечивает надежную защиту благодаря уникальности характеристик.

Целью работы являлась разработка системы распознавания лиц для умного электронного замка на базе Raspberry Pi.

Для достижения цели дипломной работы нами были выполнены все поставленные задачи. Исследованы существующие методы распознавания лиц. Спроектирована архитектура системы и подобраны аппаратно-программные компоненты. Разработан алгоритм распознавания лиц. Реализована модель управления электронным замком и выполнена интеграция компонентов системы. Проведено тестирование системы и оценена её эффективность.

Было проведено исследование научных и учебно-методических работ, статей в периодических изданиях Российской Федерации. Проведен анализ существующих работ, связанных с разработкой систем распознавания лиц, указаны достоинства и недостатки описанных работ. Исходя из анализа приведенных работ сделан вывод, что для разработки систем распознавания лиц следует использовать свёрточные нейронные сети и алгоритмы компьютерного зрения. Исходя из этого нами было подробно описано применение и принцип

действия искусственных нейронных сетей и свёрточных нейронных сетей в частности.

Для разработки систем распознавания лиц возможно применение предварительно обученных нейросетевых моделей, таких как ResNet, FaceNet и Yolo. Проведенный анализ этих моделей показал, что для обеспечения высокой надежности предпочтительно использовать модифицированную сверточную нейронную сеть ResNet в сочетании с эффективным алгоритмом детекции лиц, основанным на методах компьютерного зрения. С целью выбора оптимального алгоритма был выполнен анализ инструментов детектирования лиц на изображении, включая каскадный классификатор Хаара и алгоритм гистограммы направленных градиентов. Алгоритм гистограммы направленных градиентов демонстрирует высокую эффективность, но его производительность существенно зависит от экспозиции изображения и расположения лиц в кадре. Каскадный классификатор Хаара, напротив, менее чувствителен к качеству входного изображения.

Было исследовано использование одноплатных компьютеров, их преимуществ и недостатков, и был проведен сравнительный анализ самых популярных моделей одноплатных компьютеров Raspberry Pi моделей 2B, 3B, 4B и 5B. В результате сравнительного анализа для разработки системы распознавания лиц была выбрана модель Raspberry Pi 4 B, обладающая достаточными вычислительными ресурсами для оптимальной работы системы распознавания лиц в режиме реального времени.

В работе разработана система распознавания лиц для умного электронного замка на базе Raspberry Pi. Архитектура включает одноплатный компьютер Raspberry Pi 4 B, умный электронный замок производства лаборатории «Фаблаб» и компактную USB-камеру Logitech Webcam C170. Для реализации создано клиент-серверное приложение на языке Python с использованием веб-фреймворка FastAPI. Система детектирует лица на изображениях и сравнивает их с данными в документоориентированной базе MongoDB. Анализ существующих решений показал, что комбинированный подход, при котором

инструменты компьютерного зрения используются для обнаружения лиц и свёрточная нейронная сеть для извлечения лицевых признаков – повышает эффективность. Данный подход снизил вычислительную нагрузку, обеспечив работу системы на оборудовании Raspberry Pi 4 B. Для детектирования лиц применён каскадный классификатор Хаара, а для получения вектора признаков – модифицированная модель ResNet.

Было проведено тестирование работоспособности системы в различных условиях. Для проверки работоспособности системы распознавания лиц для умного электронного замка на базе одноплатного компьютера Raspberry Pi, нами был произведён поэтапный запуск компонентов системы. Сперва был произведен запуск базы данных и серверной части системы. Убедившись, что серверная часть системы функционирует в штатном режиме, был произведен запуск клиентской части. Она производила эффективную детекцию лиц на изображении, которые были получены с малоразмерной камеры, подключенной по интерфейсу USB. Детекция производилась при помощи алгоритма каскадного классификатора Хаара. Во время одновременной работы клиентской и серверной части, система показала свою работоспособность, детектируя лица людей на этапе программы клиента. Серверная часть успешно распознаёт лица, находя совпадение с записями в базе данных.

По результатам тестирования разработанная система распознавания лиц для умного электронного замка на базе Raspberry Pi показала свою надежность и эффективность, и может быть использована в качестве основной или дополнительной меры обеспечения удобства и безопасности доступа.

Таким образом цель работы была достигнута, все поставленные задачи были решены успешно.

В перспективе разработанная система может быть улучшена за счёт использования систем построения глубины изображения. Данное улучшение может в значительной степени повлиять на защищенность системы, исключая срабатывания системы на плоские изображения человеческих лиц.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Василенко, К. А. Информационная безопасность: технология распознования лиц / К. А. Василенко // Материалы Региональной научно-практической конференции студентов, аспирантов и молодых учёных по естественным наукам, Владивосток, 15–30 апреля 2019 года. – Владивосток: Дальневосточный федеральный университет, 2019. – С. 81-83.

2 Васина, М. В. Биометрические технологии в банковской системе обслуживания / М. В. Васина, Е. Р. Богатенко // Вестник Тульского филиала Финуниверситета. – 2020. – № 1. – С. 279-281.

3 Джаксыбекова, Г. Н. Особенности и перспективы биометрических мер безопасности в операциях с платежными банковскими картами shape * MERGEFORMAT / Г. Н. Джаксыбекова, И. Пулатов // Евразийский союз ученых. – 2020. – № 4-6(73). – С. 47-55.

4 Грянников, Д. А. Применение искусственного интеллекта для распознования лиц на базе opencv / Д. А. Грянников // Проблемы и перспективы развития России: Молодежный взгляд в будущее : Сборник научных статей 6-й Всероссийской научной конференции. В 3-х томах, Курск, 19–20 октября 2023 года / Редколлегия: А.А. Горохов (отв. редактор). Том 2. – Курск: Закрытое акционерное общество "Университетская книга", 2023. – С. 386-389.

5 Гаврилин, М. С. Модель машинного обучения при распозновании лиц / М. С. Гаврилин, М. В. Земсков, К. Н. Немыгин // ИССЛЕДОВАНИЕ ПУТЕЙ РАЗВИТИЯ НАУЧНО-ТЕХНИЧЕСКОГО ПОТЕНЦИАЛА ОБЩЕСТВА в СТРАТЕГИЧЕСКОМ ПЕРИОДЕ : сборник статей по итогам Международной научно-практической конференции, Ижевск, 24 ноября 2022 года. – Стерлитамак:

Общество с ограниченной ответственностью "Агентство международных исследований", 2022. – С. 83-85.

6 Шишков, И. А. Распознование эмоций лица с помощью нейронной сети / И. А. Шишков, Е. В. Шишова // Ресурсам области - эффективное использование : Сборник материалов XX Ежегодной научной конференции студентов Технологического университета, Москва, 30 июня 2020 года. – Москва: Общество с ограниченной ответственностью "Научный консультант", 2020. – С. 116-121.

7 Кирсанов, К. О. Разработка системы бесключевого доступа / К. О. Кирсанов, Ю. А. Заргарян // Исследования и творческие проекты для развития и освоения проблемных и прибрежно-шельфовых зон юга России : Сборник трудов XIII Всероссийской Школы-семинара, молодых ученых, аспирантов, студентов и школьников, Геленджик, 18–20 мая 2022 года. – Ростов-на-Дону –Таганрог: Южный федеральный университет, 2022. – С. 386-389.

8 Евстропов, Д. А. Устройство, принцип действия и механизм секрета отдельных видов электронных замков / Д. А. Евстропов, А. В. Кондаков, А. Г. Монин // Актуальные вопросы экспертной деятельности в современных условиях : Материалы Международной научно-практической конференции, Санкт-Петербург, 19–20 октября 2023 года. – Санкт-Петербург: Федеральное государственное казенное образовательное учреждение высшего образования «Санкт-Петербургская академия Следственного Комитета Российской Федерации», 2024. – С. 45-49.

9 Технология управления доступом в помещение через web-приложение по сети интернет / Е. А. Годовников, О. А. Петухова, В. М. Татьянкин // Вестник Югорского государственного университета. – 2020. – № 3(58). – С. 61-69.

10 Абдурахимов, Н. А. Проектирование системы распознавания лиц / Н. А. Абдурахимов // Ломоносовские научные чтения студентов, аспирантов и молодых учёных – 2018 : Материалы конференции, Архангельск, 26 марта – 30 2021 года / сост. Ю.С. Кузнецова; Северный федеральный университет им. М.В. Ломоносова. – Архангельск: ИД САФУ, 2018. – С. 27-28.

11 Сметанин, С. А. Разработка системы потокового распознавания лиц / С. А. Сметанин, М. А. Новопашин // Межвузовская научно-техническая конференция студентов, аспирантов и молодых специалистов им. Е.В. Арменского : Материалы конференции, Москва, 17–29 февраля 2016 года / Московский институт электроники и математики Национального исследовательского университета «Высшая школа экономики». – Москва: Московский институт электроники и математики НИУ ВШЭ, 2020. – С. 51.

12 Клемешов, В. П. Система распознавания лиц средствами языка Python / В. П. Клемешов // Интернаука. – 2023. – № 2-1(272). – С. 15-16.

13 Никифоров, Д. А. Как устроено распознавание лиц / Д. А. Никифоров // XXVI Туполевские чтения (школа молодых ученых) : Материалы Международной молодёжной научной конференции. Сборник докладов, Казань, 09–10 ноября 2023 года. – Казань: ИП Сагиев А.Р., 2023. – С. 2510-2513.

14 Никифоров, М. Б. Исследование степени устойчивости лицевых точек к различным положениям головы в задачах распознавания лиц с использованием библиотеки Dlib / М. Б. Никифоров, Е. Р. Муратов, А. С. Епифанов // Методы и средства обработки и хранения информации : Межвузовский сборник научных трудов. – Рязань : Рязанский государственный радиотехнический университет, 2021. – С. 102-105.

15 Евстраткин, К. С. OPENCV: варианты использования компьютерного зрения / К. С. Евстраткин, А. Р. Султанова, А. В. Ерпелев // Цифровые технологии: наука, образование, инновации : Материалы III Международного научного Форума профессорско-преподавательского состава и молодых ученых, Москва, 09 ноября 2020 года / Под редакцией А.В. Олейник, А.А. Зеленского. – Москва: Московский государственный технологический университет "СТАНКИН", 2021. – С. 28-31.

16 Волков, Д. А. Обзор программных средств обнаружения объектов с использованием микрокомпьютера Raspberry Pi / Д. А. Волков // Решетневские чтения : Материалы XXIII Международной научно-практической конференции, посвященной памяти генерального конструктора ракетно-космических систем

академика М.Ф. Решетнева. В 2-х частях, Красноярск, 11–15 ноября 2019 года / Под общей редакцией Ю.Ю. Логинова. Том Часть 2. – Красноярск: Федеральное государственное бюджетное образовательное учреждение высшего образования "Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева", 2019. – С. 338-339.

17 Машинное обучение: общие принципы и концепции // Хабр URL: https://habr.com/ru/articles/862704/ (дата обращения: 15.03.2025).

18 Константинова, Д. С., Николаева И. В. Применение многослойных нейронных сетей для распознавания образов //Информационное общество: современное состояние и перспективы развития. – 2020. – С. 150-153.

19 Пырнова, О. А., Зарипова Р. С. Методы и проблемы переобучения многослойной нейронной сети //Информационные технологии в строительных, социальных и экономических системах. – 2020. – №. 2. – С. 101-102.

20 Персептрон. Многослойный персептрон. // Apsheronsk.bozo.ru URL: http://apsheronsk.bozo.ru/Neural/Lec2.htm (дата обращения: 12.05.2025).

21 Глубокие Нейронные Сети // CoderLessons.com URL: https://coderlessons.com/tutorials/python-technologies/python-deep-learning/glubokie-neironnye-seti (дата обращения: 27.04.2025).

22 Ultralytics YOLO Docs // ultralytics URL: https://docs.ultralytics.com/ru/ (дата обращения: 07.05.2025).

23 ResNet (34, 50, 101): «остаточные» CNN для классификации изображений // Neurohive URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/ (дата обращения: 07.05.2025).

24 FaceNet: Универсальный эмбеддинг для распознавания и кластеризации лиц // Хабр URL: https://habr.com/ru/articles/731020/ (дата обращения: 07.05.2025).

25 Амеличев, Г. Э. Распознавание лиц с использованием каскадов Хаара / Г. Э. Амеличев, В. С. Панина, Ю. С. Белов // E-Scio. – 2020. – № 8(47). – С. 221-228.

26 Гистограмма направленных градиентов / А. С. Карнаухов, Ю. А. Панков, Р. С. Гаврюшин [и др.] // Состояние и перспективы развития современной науки

по направлению «Техническое зрение и распознавание образов» : сборник статей II Всероссийской научно-технической конференции, Анапа, 22 октября 2020 года / Военный инновационный технополис "ЭРА". Том 2. – Анапа: Федеральное государственное автономное учреждение "Военный инновационный технополис "ЭРА", 2020. – С. 25-27.

27 Understanding Single Board Computers (SBCs) : Their Functions and Applications // Voltrium Systems URL: https://www.voltrium.com.sg/en/understanding-single-board-computers-sbcs-their-functions-and-applications/ (дата обращения: 27.03.2025).

28 Свидетельство о государственной регистрации программы для ЭВМ № 2023617743 Российская Федерация. СКУД для работы с электронным замком : № 2022686428 : заявл. 27.12.2022 : опубл. 13.04.2023 / В. В. Овчаренко, Д. В. Павленко ; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Крымский федеральный университет имени В.И. Вернадского».

29 Свидетельство о государственной регистрации программы для ЭВМ № 2023680683 Российская Федерация. Программа для микроконтроллера ESP32-C3 для управления BLE/Wi-Fi ретранслятором электронного замка : № 2023669488 : заявл. 21.09.2023 : опубл. 04.10.2023 / В. В. Овчаренко ; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Крымский федеральный университет имени В.И. Вернадского».

30 Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API // Tproger URL: https://tproger.ru/articles/web-api (дата обращения: 29.04.2025).

31 FastAPI // FastAPI URL: https://fastapi.tiangolo.com/ (дата обращения: 29.04.2025).

32 NoSQL база данных MongoDB: работа на Python (pymongo) // Блог проекта Директ-ПРО URL: http://directprobi.ru/blogs/mongo-python-nosql-baza-dannyh-pymongo-api-database-mongodb-subd/ (дата обращения: 29.04.2025).

33 Face detection with dlib (HOG and CNN) // pyimagesearch URL: https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/ (дата обращения: 29.04.2025).

34 Raspberry Pi hardware // Raspberry Pi URL: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (дата обращения: 29.04.2025).

35 Установка Python 3.9 и виртуального окружения (VENV) // FirstVDS URL: https://firstvds.ru/technology/ustanovka-python-39-i-virtualnogo-okruzheniya-venv (дата обращения: 29.04.2025).

36 Установка и Настройка MongoDB в Docker // Server Space URL: https://serverspace.ru/support/help/ustanovka-i-nastrojka-mongodb-v-docker (дата обращения: 29.04.2025).

37 Сайпиддинов, Ш. Экспоненциальные оценки сумм многомерных случайных величин / Ш. Сайпиддинов, Р. К. Бахрамов // Мировая наука. – 2024. – № 12(93). – С. 107-110. – EDN GQZDEX.

38 Operating system images // Raspberry Pi URL: https://www.raspberrypi.com/software/operating-systems/ (дата обращения: 15.04.2025).

ПРИЛОЖЕНИЯ

Приложение А
Исходный код веб-сервера FastAPI

```
import json
import cv2
import uvicorn, asyncio
import numpy as np
from fastapi import FastAPI, File, UploadFile, Form, HTTPException, Depends
from schemas.api_schemas import *
from workers.mongo_worker import MongoWorker
from workers.face_rec_worker import FaceRecognition

app = FastAPI()
mongo_worker = MongoWorker()
face_recognition_worker = FaceRecognition()

@app.post('/add_person')async def add_person(json_data: str = Form(...),
                img_data: UploadFile = File(...),
                face_recognition: FaceRecognition = Depends(lambda:
face_recognition_worker),
                mongo: MongoWorker = Depends(lambda: mongo_worker)) ->
BaseResponse:
   json_validate = AddPersonJSON(**json.loads(json_data))
```

```
    if img_data.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException(400, detail="Invalid file type")
    file_data = await img_data.read()
    face_img = cv2.imdecode(np.fromstring(file_data, np.uint8),
cv2.IMREAD_COLOR)
    encodings = face_recognition.get_encodings(face_img)
    if not encodings:
        raise HTTPException(422, detail="Can't find human face")
    result = mongo.add_person(json_validate.username, encodings[0],
img_data.filename, face_img)
    if result.error:
        raise HTTPException(422, detail=str(result))
    return BaseResponse(result="Success")


@app.post('/find_by_img')
async def find_by_img(img_data: UploadFile = File(...),
                      face_recognition: FaceRecognition = Depends(lambda:
face_recognition_worker),
                      mongo: MongoWorker = Depends(lambda: mongo_worker)) ->
BaseResponse:
    if img_data.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException(400, detail="Invalid file type")
    file_data = await img_data.read()
    face_img = cv2.imdecode(np.fromstring(file_data, np.uint8),
cv2.IMREAD_COLOR)
    encodings = face_recognition.get_encodings(face_img)
    if not encodings:
        raise HTTPException(422, detail="Can't find human face")
    res = face_recognition_worker.face_distance(encodings)
    print(res)
```

```
    return BaseResponse(result=res)
async def main():
    config = uvicorn.Config("server:app", host="0.0.0.0", port=5000,
log_level="info")
    server = uvicorn.Server(config)
    await server.serve()

if __name__ == "__main__":
    asyncio.run(main())
```

Приложение Б

Исходный код системы валидации HTTP запросов

```
from typing import Any
from pydantic import BaseModel
ALLOWED_MIME_TYPES = {"image/png", "image/jpeg"}
class AddPersonJSON(BaseModel): username: str
class BaseResponse(BaseModel): result: Any error: bool = False
from pydantic import BaseModel, NonNegativeInt, ConfigDict from typing import
Any
from datetime import datetime
from numpy import ndarray, dtype
class Person(BaseModel):
    person_id: NonNegativeInt = 0 person_name: str
    encodings: list[float]
    image_name: str
    adding_date: str = datetime.now().isoformat()
    active_status: bool = True
    model_config = ConfigDict(
        arbitrary_types_allowed=True,
    )
```

Приложение В

Исходный код системы взаимодействия с СУБД MongoDB

```
import os
import gridfs
import numpy as np
import cv2
from pymongo import MongoClient
from dotenv import load_dotenv
from schemas.base_schemas import *
from schemas.api_schemas import *
class MongoWorker():
    def __init__(self):
        load_dotenv()
        self.client = MongoClient(host=os.getenv('MONGO_HOST'),
                                  port=int(os.getenv('MONGO_PORT')),
                                  username=os.getenv('MONGO_USER'),
                                  password=os.getenv('MONGO_PASS'))
        self.db = self.client["Face_recognition"]
        self.persons = self.db["persons"]
        self.counters = self.db["counters"]
        self.gfs = gridfs.GridFS(self.db)
    def count_persons(self):
        docs_count = self.persons.count_documents({})
        return docs_count
    def get_and_update_counter(self, counter_name: str) -> int:
```

```
        counter = self.counters.find_one_and_update(
            {"counter_name": counter_name},
            {"$inc": {"counter": 1}},
            upsert=True,
            return_document=True)
        return counter["counter"]
    def add_person(self, person_name:str, encodings:ndarray[Any, dtype],
image_name:str, face_img:np.ndarray) -> BaseResponse:
        try:
            new_person_id = self.get_and_update_counter("persons")
            new_person = Person(
                person_id = new_person_id,
                person_name = person_name,
                encodings = encodings.tolist(),
                image_name = f"{new_person_id}_{image_name}")
            file_id = self.save_image_to_gridfs(face_img, image_name)
            mongo_result = self.persons.insert_one(new_person.model_dump())
            return BaseResponse(result={"file_id": file_id, "mongo_result":
mongo_result})
        except Exception as exception:
            print(exception)
            return BaseResponse(result=exception, error=True)
    def save_image_to_gridfs(self, image_array, filename):
        success, encoded_image = cv2.imencode('.png', image_array)
        if not success:
            raise ValueError("Не удалось закодировать изображение")
        file_id = self.gfs.put(encoded_image.tobytes(), filename=filename)
        return file_id
    def get_all_encodings(self):
        result = {}
```

```
        projection = {"person_id": 1, "encodings": 1, "_id": 0}
        for document in self.persons.find({}, projection):
            entry = {document["person_id"]: document["encodings"]}
            result.update(entry)
        return result
    def retrieve_image_from_gridfs(self, file_id):
        try:
            gridfs_file = self.gfs.get(file_id)
            image_bytes = gridfs_file.read()
            image_array = cv2.imdecode(np.frombuffer(image_bytes, np.uint8),
cv2.IMREAD_COLOR)
            if image_array is None:
                raise ValueError("Не удалось декодировать изображение")
            return image_array
        except gridfs.errors.NoFile:
            raise FileNotFoundError(f"Файл с ID {file_id} не найден в GridFS")
```

Приложение Г

Исходный код системы распознавания лиц

```
import face_recognition
import numpy as np
import cv2
from workers.mongo_worker import MongoWorker
class FaceRecognition():
    def __init__(self):
        self.mongo_worker = MongoWorker()
        self.face_encodings_in_cache = self.mongo_worker.get_all_encodings()
    def get_encodings(self, face_image):
        return face_recognition.face_encodings(face_image)
    def softmax(self, values):
        exp_values = np.exp(values)
        exp_values_sum = np.sum(exp_values)
        return exp_values / exp_values_sum
    def kl_divergence(self, face_encodings, face_to_compare):
        if len(face_encodings) == 0:
            return np.empty((0))
        face_encodings = np.asarray(self.softmax(face_encodings))
        face_to_compare = np.asarray(self.softmax(face_to_compare))
        face_encodings = face_encodings / face_encodings.sum(axis=1,
keepdims=True)
        face_to_compare = face_to_compare / face_to_compare.sum()
        epsilon = 1e-10
```

```
        face_encodings = np.clip(face_encodings, epsilon, 1)
        face_to_compare = np.clip(face_to_compare, epsilon, 1)
        return np.sum(face_encodings * np.log(face_encodings / face_to_compare),
axis=1)
    def face_distance(self, face_encodings:list[float]):
        result = {}
        if self.mongo_worker.count_persons() > len(self.face_encodings_in_cache):
            self.face_encodings_in_cache = self.mongo_worker.get_all_encodings()
        for person_id in self.face_encodings_in_cache:
            result.update({person_id: self.kl_divergence(face_encodings,
self.face_encodings_in_cache[person_id])})
        print(result.values())
        try:
            minimal = min(result.values())
            print(minimal)
            if minimal > 0.001:
                return None
            return [key for key, val in result.items() if val == minimal][0]
        except:
            best_in_frame = dict()
            for key, elements in result.items():
                best_in_frame.update({key:min(elements)})
            minimal = min(best_in_frame.values())
            print(minimal)
            if minimal > 0.001:
                return None
            return [key for key, val in best_in_frame.items() if val == minimal][0]
```

Приложение Д

Список необходимых библиотек и модулей Python для работы системы

pymongo==4.11.3

fastapi==0.115.11

uvicorn==0.34.0

opencv-python==4.11.0.86

face-recognition==1.3.0

python-multipart==0.0.20

Приложение Е

Исходный код клиентской части системы

```
import requests
import time
import threading
import RPi.GPIO as GPIO
import cv2 as cv
import numpy as np
class DoorWorker():
    def __init__(self):
        self.door_status = False
    def open_door(self):
        if not self.door_status:
            self.door_status = True
            print("OPEN DOOR")
            GPIO.setmode(GPIO.BOARD)
            GPIO.setup(16, GPIO.OUT)
            GPIO.output(16, 1)
            time.sleep(5)
            GPIO.output(16, 0)
            white_list.clear()
            permit_list.clear()
            self.door_status = False
class HttpWorker():
    def __init__(self):
```

```
        self.last_answer = None
        self.search_filter = threading.Lock()
    def search_face(self, position, img_data):
        if not self.search_filter.locked():
            with self.search_filter:
                URL = ADDRESS + "/find_by_img"
                headers = {'accept': 'application/json'}
                files = {'img_data': ('photo.jpg', img_data["image"], 'image/jpeg')}
                answer = requests.post(URL, headers=headers, files=files).json()
                if answer["result"]:
                    if answer["result"] in white_list:
                        white_list[answer["result"]] = \
                            {"position": position,
                             "counter": white_list[answer["result"]]["counter"] + 1}
                    else:
                        white_list[answer["result"]] = \
                            {"position": position, "counter": 1}
                print(white_list)
                self.last_answer = answer
                time.sleep(0.1)
class RecognitionWorker():
    def __init__(self, cap):
        self.face_cascade = cv.CascadeClassifier(cv.data.haarcascades +
'haarcascade_frontalface_default.xml')
    def check_frame(self, frame):
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
        return faces
class VisualizeWorker():
```

```
    def drawer(self, image, faces, door_status, permit_list, fps=None):
        output = image.copy()
        if fps:
            cv.putText(output, 'FPS: {:.2f}'.format(fps), (0, 15),
cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0))
        white_position = list()
        if permit_list:
            for person in permit_list:
                try:
                    white_position.append(white_list[person]["position"])
                except:
                    pass
        crop_img = list()
        for position, face in enumerate(faces):
            coords = faces[position].astype(np.int32)
            for i in range(len(coords)):
                if coords[i] < 0:
                    coords[i] = 0
            if door_status and position in white_position:
                color = (0, 255, 0)
            else:
                color = (0, 0, 255)
            cv.rectangle(output, (coords[0], coords[1]), (coords[0] + coords[2], coords[1]
+ coords[3]), color, 2)
            crop_img.append(output[coords[1]:coords[1] + coords[3],
coords[0]:coords[0] + coords[2]])
        return output, crop_img
def resize(img):
    height_size = 640
    scale = height_size / img.shape[0]
```

```
    width = int(img.shape[1] * scale)
    height = int(img.shape[0] * scale)
    return cv.resize(img, (width, height))
def permit(check_list):
    permit_ids = list()
    if len(check_list) != 0:
        for person in check_list:
            if check_list[person]["counter"] >= 3:
                permit_ids.append(person)
        return permit_ids
    else:
        return
if __name__ == '__main__':
    ADDRESS = "http://127.0.0.1:5000"
    device_id = 0
    white_list = dict()
    door_close = True
    cap = cv.VideoCapture(device_id)
    tm = cv.TickMeter()
    cv.namedWindow('libfacedetection demo', cv.WINDOW_NORMAL)
    cv.setWindowProperty('libfacedetection demo', cv.WND_PROP_FULLSCREEN,
cv.WINDOW_FULLSCREEN)
    visualize = VisualizeWorker()
    recognition = RecognitionWorker(cap=cap)
    request = HttpWorker()
    door = DoorWorker()
    while cv.waitKey(1) < 0:
        has_frame, frame = cap.read()
        if not has_frame:
            print('No frames grabbed!')
```

```
        permit_list = permit(white_list)
        tm.start()
        faces = recognition.check_frame(frame)
        tm.stop()
        if faces is not None:
            frame, crop_imgs = visualize.drawer(frame, faces, door.door_status,
permit_list, fps=tm.getFPS())
            for position, img in enumerate(crop_imgs):
                resize_frame = resize(img)
                _, im_buf_arr = cv.imencode(".jpg", resize_frame)
                byte_im = im_buf_arr.tobytes()
                threading.Thread(target=request.search_face, args=(position, {'image':
byte_im},)).start()
        if permit_list:
            threading.Thread(target=door.open_door).start()
        cv.imshow('libfacedetection demo', frame)
        tm.reset()
    print("END")
```