Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Промышленная робототехника и электроника»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«19» 06 2026 г.

Декан ФЭУ

_____ А.С. Гудим
(подпись)

«19» 06 2026 г.

УТВЕРЖДАЮ

Ио. проректора по научной работе, д-р техн. наук. профессор

_____ А.В. Космынин
(подпись)

«19» 06 2026 г.

«Электрический замок с системой распознавания»

Комплект документации на управляющую программу для
автоматизированной/роботизированной системы

Руководитель СКБ _____ 18.06.2026 _____ С.И. Сухоруков
(подпись, дата)

Руководитель проекта _____ 15.06.2026 _____ Н.Н. Любушкина
(подпись, дата)

Комсомольск-на-Амуре 2026

Карточка проекта

Название	Электрический замок с системой распознавания
Тип проекта	Тип проекта: техническое творчество (инициативный).
Исполнители	Студент _____ Д.А. Цыган – 5ПЭб-1 Студент _____ П.А. Усов – 5ПЭб-1 Студент _____ А.А. Кузнецов – 5ПЭб-1
Срок реализации	01.10.2025 - 11.06.2026

Использованные материалы и компоненты

Наименование	Количество, шт.
Распределительная коробка	1
Мышь компьютерная	1
Клавиатура (мембранная)	1
Raspberry Pi 5 (8GB RAM)	1
Дверная ручка	1
Дверной замок	1
ДСП, $м^2$	1
Веб-камера	1
Сервопривод	1
Саморез (ГОСТ 1145-80)	6
Болт и гайка (ГОСТ 1491-80)	Попарно 4
Raspberry Pi 27W USB-C Power Supply	1
Стержень клеевого пистолета	2

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: Электрический замок с системой распознавания

Назначение: Обеспечение контролируемого доступа в помещение, путем биометрической верификации пользователя по лицу, исключая использование физических ключей, карт или PIN-кодов.

Область использования: Устройство может быть использовано: в частном доме/квартире на входных и внутренних дверях; в офисах для контроля доступа в кабинеты, серверные, учёта рабочего времени; в рамках умного дома за счёт интеграции со сценариями (свет, климат при распознавании).

Функциональное описание проекта: Электрический замок с системой распознавания (далее Face Lock) - это автономный электромеханический замок с распознаванием лица. Устройство сравнивает изображение с локальной базой пользователей, поддерживает роли администратора и гостевые доступы. Для защиты от обмана используется (детекция живости). При успешной верификации контроллер подает импульс на замок и отслеживает состояние двери. Замок работает от источника питания с напряжением 5 В. Все события фиксируются в журнале.

Техническое описание устройства:

Устройство (программно-аппаратный комплекс) состоит из электромеханического замка на основе сервопривода с подключенной к нему системой управления. Без применения (в обход) алгоритмов распознавания замок открывается по нажатию кнопки. С помощью системы технического зрения на базе USB-видеокамеры происходит распознавание лиц людей, желающих пройти в помещение. В случае успешного распознавания (доступ

разрешен) система открывает замок, подавая управляющий сигнал на сервопривод.

Требования:

Программно-аппаратный комплекс должен обеспечивать распознавание лиц с вероятностью не менее 95%. При успешном распознавании электромеханический замок должен открываться, при неуспешном – оставаться закрытым. Открытие/закрытие замка должен обеспечивать сервопривод.

План работ:

Наименование работ	Срок
Изучение базовых навыков (теория компьютерного зрения, Python, OpenCV) и постановка задач	10.2025
Подбор деталей и оборудования (Raspberry Pi 5, камера, сервопривод, коробка, замок, крепёж)	11.2025
Создание программного кода (модули сбора данных, обучения, распознавания) и сбор фотографий для датасета	01.2026
Распознавание лиц, отладка алгоритмов (LBPH / Fisherfaces), настройка точности и логирование	03.2026
Сборка макета и подключение компонентов (камера, сервопривод, питание) к Raspberry Pi	04.2026
Запись журнала событий (лог-файл face_log.txt), интеграция с сервоприводом и тестирование сценариев доступа	05.2026
Оформление документации (паспорт, пояснительная записка, схемы, фотографии) и сдача проекта	06.2026

Комментарии:

План разбит на 7 этапов общей длительностью 9 месяцев. Всё построено по принципу «от простого к сложному»: сначала разбираемся в теории и закупаем детали, потом пишем код и учим программу узнавать лица, затем подключаем замок и сервопривод. Отдельно вынесены записи журнала и оформление документов. На два самых трудоёмких этапа выделяется по 2 месяца, так как именно там возникает больше всего ошибок.

Перечень графического материала:

1. Структурная схема;

2. Внешний вид изделия;

3. Трехмерная модель изделия

Руководитель проекта 15.05.25 Н.Н. Любушкина

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

Управляющей программы для автоматизированной системы

«Электронный замок с системой распознавания»

Руководитель проекта _____ 15.06.26 Н.Н. Любушкина
(подпись, дата)

Исполнители проекта _____ 19.06.2026 Д.А. Цыган
(подпись, дата)

_____ 19.06.2026 П.А. Усов
(подпись, дата)

_____ 19.06.2026 А.А. Кузнецов
(подпись, дата)

Комсомольск-на-Амуре 2026

Содержание

1 Общие положения 8

1.1 Наименование изделия 8

1.2 Наименования документов, на основании которых ведется проектирование изделия 8

1.3 Перечень организаций, участвующих в разработке изделия 8

1.4 Сведения об использованных при проектировании нормативно-технических документах 9

2 Описание программы 10

2.1 Общие сведения 10

2.2 Функциональное назначение программы 10

2.3 Описание логической структуры 11

2.4 Используемые технические средства 12

2.5 Вызов и загрузка 13

2.6 Входные данные 13

2.7 Выходные данные 13

3 Руководство оператора 15

3.1 Назначение программы 15

3.2 Условия выполнения программы; 15

3.3 Выполнение программы 15

3.4 Сообщения оператору 16

ПРИЛОЖЕНИЕ А 18

ПРИЛОЖЕНИЕ Б 28

ПРИЛОЖЕНИЕ В 30

1 Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации управляющей программы «Электронный замок с системой распознавания» (далее «программа»).

Паспорт входит в комплект поставки изделия и программы. Перед началом использования внимательно изучите правила эксплуатации и порядок работы. Изделие постоянно совершенствуется с целью повышения надежности и улучшения условий эксплуатации, поэтому в конструкцию могут быть внесены изменения, не нашедшие отражения в данном документе.

1.1 Наименование изделия

Полное наименование изделия – «Электронный замок с системой распознавания».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Проектирование устройства «Электронный замок с системой распознавания» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком создания программы «Электронный замок с системой распознавания» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт, д. 17. Исполнителями работ по созданию программы «Электронный замок с системой распознавания» являются Конструкторы студенческого

конструкторского бюро (далее СКБ), студенты группы 5ПЭб-1,. Д.А. Цыган, П.А. Усов , А.А. Кузнецов.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 19.001-77. Единая система программной документации (ЕСПД). Общие положения.

ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77. ЕСПД. Стадии разработки.

ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78. ЕСПД. Описание программы.

ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

2 Описание программы

2.1 Общие сведения

При разработке программы использовалось несколько простых функций, чтобы код был понятным и его удобно было дорабатывать.

Программа поделена на четыре части (функции): первая отвечает за сбор фотографий, вторая - за обучение, третья - за узнавание лиц, четвёртая - за управление замком.

Классы и функции. Всё, что связано с камерой, распознаванием и экраном, вынесено в отдельные блоки. Это помогает не запутаться в коде и использовать уже написанное повторно.

Добавление без переустановки. Можно в любой момент добавить нового человека и заново обучить программу, без корректировки программы.

Готовые наработки. Для реализации алгоритмов взяты готовые алгоритмы из библиотеки OpenCV: один ищет лица на картинке, два других - узнают, чьё это лицо. Программа написана на языке программирования Python, поэтому запускается на Windows, Linux, Mac и на Raspberry Pi - везде, где есть Python и нужные библиотеки

2.2 Функциональное назначение программы

Программа предназначена для создания базы лиц доверенных лиц, обучения модели распознавания и последующей идентификации людей в реальном времени с выдачей управляющего сигнала на сервопривод (открытие замка) при успешной верификации.

Основными функциями программы являются: регистрация новых пользователей с захватом до двухсот эталонных изображений лица через веб-камеру; автоматическое обновление базы пользователей в файле users.json; обучение локальной бинарной модели LBPH (Local Binary Patterns Histograms) на собранном датасете; распознавание лиц в реальном времени с веб-камеры (или из видеофайла или потока); вычисление процента уверенности и принятие решения на основе заданного порога (более

пятнадцати процентов); управление сервоприводом через вывод GPIO 18 при успешном распознавании (два цикла поворота на девяносто градусов с возвратом); ведение подробного журнала событий в текстовые файлы (attendance_log.txt, unknown_faces_log.txt, персональные логи), CSV-файл attendance.csv, а также базу данных SQLite attendance.db; сохранение изображений распознанных лиц в папку recognized_faces/; формирование отчёта о посещениях за текущий день; обеспечение единого консольного меню для удобного запуска всех режимов через модуль run.py.

2.3 Описание логической структуры

Программа состоит из следующих исходных файлов и вспомогательных данных.

Модуль run.py является главным модулем, реализующим консольное меню для выбора действий: распознавание, создание датасета, обучение модели, отчёт и выход. Он запускает остальные модули как дочерние процессы через библиотеку subprocess.

Модуль dataset.py предназначен для сбора данных. Он захватывает видео с камеры, детектирует лица по каскаду Хаара, сохраняет область лица размером 200 на 200 пикселей в папку dataset/<имя_пользователя>/ и обновляет файл users.json.

Модуль trainer.py выполняет обучение модели. Он загружает изображения из папки dataset/, для каждого файла применяет предобработку (размытие по Гауссу и эквализацию гистограммы), повторно детектирует лицо, приводит изображение к размеру 200 на 200 пикселей. Затем обучает модель cv2.face.LBPHFaceRecognizer_create() и сохраняет её в файл trained_faces_lbph.yml.

Модуль recognition.py отвечает за распознавание и управление. Он выполняет идентификацию лиц в реальном времени или по запросу с аргументами командной строки (--image, --stream). Модуль управляет сервоприводом, логирует события в текстовые файлы, CSV и SQLite, а также

сохраняет фотографии узнанных лиц. Поддерживается аргумент --report для вывода дневной статистики посещений.

Файл haarcascade_frontalface_default.xml содержит обученный каскад Хаара для детекции лиц (поставляется вместе с библиотекой OpenCV). Файл users.json представляет собой базу пользователей в формате {"ID": "Имя"}. Файл trained_faces_lbph.yml является сохранённой обученной моделью LBPH. База данных attendance.db содержит таблицу visits со следующими полями: идентификатор, временная метка, дата, время, имя, уверенность, статус и путь к изображению.

Взаимосвязи модулей организованы следующим образом. run.py выступает диспетчером: пользователь выбирает пункт меню, и соответствующий модуль запускается через subprocess. dataset.py создаёт структуру папок и файлов, которые впоследствии используются trainer.py. trainer.py читает папки пользователей из dataset/, создаёт trained_faces_lbph.yml и обновляет users.json. recognition.py загружает trained_faces_lbph.yml, users.json и каскад Хаара; в процессе работы создаёт или дополняет файлы логов, базу данных и сохраняет изображения в папку recognized_faces/.

2.4 Используемые технические средства

Аппаратное обеспечение включает процессор с тактовой частотой не менее 1,5 гигагерц (рекомендуется 2,0 гигагерц для Raspberry Pi 5), оперативную память от двух гигабайт (рекомендуется четыре гигабайта), веб-камеру с поддержкой V4L2 (USB или встроенную), а также опционально сервопривод, подключаемый к выводу GPIO 18 (сигнальный провод), питанию 5 вольт и общему проводу GND.

Программное обеспечение включает операционную систему (Windows 10 или 11, Linux, включая Raspberry Pi OS, или macOS 10.15 и выше), интерпретатор Python версии 3.7 или выше, а также библиотеки Python,

устанавливаемые через pip: opencv-python версии не ниже 4.5, opencv-contrib-python (для модуля cv2.face), numpy, pillow и (для Raspberry Pi) RPi.GPIO.

2.5 Вызов и загрузка

Для вызова и загрузки программы необходимо выполнить следующие действия. Убедиться, что все файлы проекта (run.py, dataset.py, trainer.py, recognition.py,haarcascade_frontalface_default.xm) находятся в одной директории. Открыть терминал в этой директории. Установить зависимости командой pip install opencv-python opencv-contrib-python numpy pillow RPi.GPIO. Запустить программу командой python run.py. После запуска на экране появится главное меню с пунктами от одного до шести.

2.6 Входные данные

Входными данными программы являются: для сбора датасета – видеопоток с веб-камеры (автоматический захват); для распознавания через пункт 1 главного меню – видеопоток с веб-камеры; для распознавания через аргументы командной строки (при прямом вызове recognition.py) – параметр --image <путь_к_файлу> для статичного изображения, параметр --stream <URL_или_путь_к_видео> для видеопотока, параметр --report для вывода статистики без запуска камеры; для обучения – директория dataset/, содержащая подпапки с именами пользователей и файлами в форматах .jpg или .png.

2.7 Выходные данные

Выходными данными программы являются: при добавлении пользователя – набор изображений лица в формате .jpg в папке dataset/<имя>/; при обучении – файл trained_faces_lbph.yml и обновлённый файл users.json; при распознавании – видеоокно с прямоугольниками (зелёные для узнанных лиц, красные для неопознанных) и подписями с именем и процентом уверенности, запись события в файлы attendance_log.txt, unknown_faces_log.txt, attendance.csv и базу данных

SQLite attendance.db, сохранение вырезанного изображения лица узнанного пользователя в папку recognized_faces/ с именем вида <имя>_<время>_<уверенность>.jpg, а также движение сервопривода при успешной верификации; при запросе отчёта (аргумент --report или пункт 4 меню) – вывод в консоль количества посещений по каждому пользователю за текущую дату.

3 Руководство оператора

3.1 Назначение программы

Программа позволяет администратору добавлять новых пользователей в базу, обучать модель распознавания и в дальнейшем идентифицировать людей перед камерой. При успешном распознавании активируется сервопривод (имитация открытия замка), а все события фиксируются в журналах.

3.2 Условия выполнения программы;

Для корректной работы программы необходимо соблюдение следующих условий. Веб-камера должна быть исправна и доступна системе (не занята другим приложением). Для работы с сервоприводом требуется Raspberry Pi с подключённым питанием 5 вольт и общим проводом GND, сигнальный провод подключается к выводу GPIO 18. Файл haarcascade_frontalface_default.xml должен находиться в той же папке, что и скрипты программы. Рекомендуется обеспечить равномерное освещение лица пользователя как при сборе датасета, так и при распознавании.

3.3 Выполнение программы

Запуск программы осуществляется в терминале командой python run.py. После запуска на экране отображается главное меню следующего вид.

«СИСТЕМА РАСПОЗНАВАНИЯ ЛИЦ ДЛЯ RASPBERRY PI 5. ЕДИНЫЙ ЗАПУСКАТЕЛЬ. Пункт 1 – запустить распознавание лиц и сервопривод. Пункт 2 – создать датасет нового пользователя. Пункт 3 – обучить модель. Пункт 4 – показать отчёт за сегодня. Пункт 5 – перезапустить меню. Пункт 6 – выход.»

Для добавления нового пользователя необходимо выбрать пункт 2, затем ввести имя латиницей или кириллицей. Система автоматически сделает двести снимков лица. Пользователю рекомендуется поворачивать голову в небольших пределах для повышения качества датасета. По окончании процесса будет выведено сообщение об успешном сохранении изображений.

Для обучения модели после добавления хотя бы одного пользователя следует выбрать пункт 3. Программа соберёт все изображения из папки dataset/ и создаст файл обученной модели trained_faces_lbph.yml. Процесс обучения может занять от одной до двух минут в зависимости от количества изображений.

Для запуска распознавания необходимо выбрать пункт 1. Откроется окно видеопотока с камеры. При появлении лица в кадре система выводит на экран имя пользователя (или надпись «Неопознанный») и процент уверенности. При успешном распознавании с уверенностью более пятнадцати процентов сервопривод совершает два поворота на девяносто градусов с возвратом в исходное положение, имитируя открытие замка. Для выхода из режима распознавания необходимо нажать клавишу q на клавиатуре.

Для просмотра отчёта за сегодняшний день следует выбрать пункт 4. На экран будет выведено количество успешных распознаваний по каждому зарегистрированному пользователю за текущую дату.

Завершение работы программы осуществляется выбором пункта 6 в главном меню.

3.4 Сообщения оператору

Таблица 1 - Сообщение оператору

Сообщение	Значение	Действие оператора
Введите имя:	Запрос имени нового пользователя.	Ввести имя латиницей или кириллицей.
Samples: 34/200	Индикатор сбора эталонов.	Оставаться неподвижно перед камерой.
Файлы моделей не найдены...	Первый запуск или обучение не проводилось.	Согласиться на создание моделей (ввести да).
Не удалось открыть камеру.	Веб-камера не найдена или занята.	Проверить подключение камеры. Закрыть другие программы.исп. камеру

Продолжение таблица 2

Сообщение	Значение	Действие оператора
Неопознанный (23%)	Лицо обнаружено, но не найдено в базе или уверенность ниже порога.	Добавить этого человека в базу через пункт 1.
Ошибка: не найден файл модели...	Отсутствует trained_faces_lbph.yml.	Выполнить тренировку модели через пункт 2 главного меню.
Нажмите 'q' для выхода...	Программа ожидает команды на закрытие окна с видео.	Нажать клавишу q на клавиатуре.
Некорректный выбор!	В меню введено не число или число вне диапазона 1-6.	Ввести корректный номер пункта меню.

ПРИЛОЖЕНИЕ А

(обязательное)

Dataset_creator

```
import cv2
import os
import numpy as np
import time
import json
from datetime import datetime

def create_user_dataset():
    print("=" * 60)
    print("СОЗДАНИЕ ДАТАСЕТА ДЛЯ НОВОГО ПОЛЬЗОВАТЕЛЯ")
    print("=" * 60)

    while True:
        name = input("\nВведите имя нового пользователя: ").strip()
        if name:
            break
        print("Имя не может быть пустым!")

    dataset_dir = "dataset"
    user_dir = os.path.join(dataset_dir, name.replace(" ", "_"))

    os.makedirs(dataset_dir, exist_ok=True)

    if os.path.exists(user_dir):
        print(f" Папка {name} уже существует.")
        if input("Перезаписать? (y/n): ").lower() != 'y':
            return
    else:
        os.makedirs(user_dir)

    print(f"\n Папка: {user_dir}")
    print(" Камера запущена. Двигайтесь перед камерой.")
    print("Нажмите 'q' для завершения.\n")

    cap = cv2.VideoCapture(0, cv2.CAP_V4L2)
    if not cap.isOpened():
        cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print(" Не удалось открыть камеру!")
        return

    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

    count = 0
    max_samples = 200

    while count < max_samples:
        ret, frame = cap.read()
        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

        for (x, y, w, h) in faces:
            face_roi = gray[y:y + h, x:x + w]
```

```
            face_resized = cv2.resize(face_roi, (200, 200))

            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S_%f")[:-3]
            filename = os.path.join(user_dir, f"{name}_{count:03d}_{timestamp}.jpg")
            cv2.imwrite(filename, face_resized)
            count += 1

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, f"Sample: {count}/{max_samples}", (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
            break

        cv2.imshow("Сбор датасета - нажмите q", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

    print(f"\n Сохранено {count} изображений для {name}")
    update_users_json(name)

def update_users_json(new_name):
    users_file = "users.json"
    if os.path.exists(users_file):
        with open(users_file, "r", encoding="utf-8") as f:
            users = json.load(f)
    else:
        users = {}

    if users:
        max_id = max(int(k) for k in users.keys())
        new_id = max_id + 1
    else:
        new_id = 1

    users[str(new_id)] = new_name

    with open(users_file, "w", encoding="utf-8") as f:
        json.dump(users, f, ensure_ascii=False, indent=4)

    print(f" Пользователь '{new_name}' добавлен с ID = {new_id}")

def main():
    print("1. Создать датасет нового пользователя")
    print("2. Выход")
    choice = input("\nВыберите действие (1-2): ")
    if choice == "1":
        create_user_dataset()
    else:
        print("Выход.")

if __name__ == "__main__":
                                        main()
```

Face_recognition

```
import cv2
import numpy as np
import os
import datetime
import json
import argparse
import RPi.GPIO as GPIO
import time
import sqlite3

SERVO_PIN = 18
ATTENDANCE_DB = "attendance.db"
ATTENDANCE_FILE = "attendance_log.txt"

os.environ["OPENCV_VIDEOIO_PRIORITY_BACKEND"] = "1"

def init_db():
    conn = sqlite3.connect(ATTENDANCE_DB)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS visits (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT,
            date TEXT,
            time TEXT,
            name TEXT,
            confidence REAL,
            status TEXT,
            image_path TEXT
        )
    ''')
    conn.commit()
    conn.close()

def set_servo_angle(angle):
    duty = 12.0 - (angle / 54.0)
    pwm.ChangeDutyCycle(duty)
    time.sleep(2)
    pwm.ChangeDutyCycle(0)

def cleanup_servo():
    try:
        pwm.stop()
        GPIO.cleanup()
    except:
        pass
GPIO.setmode(GPIO.BCM)
GPIO.setup(SERVO_PIN, GPIO.OUT)
pwm = GPIO.PWM(SERVO_PIN, 50)
pwm.start(0)

USERS_FILE = "users.json"
TRAINED_FILES = {"LBPH": "trained_faces_lbph.yml", "Fisherfaces": "trained_faces_fisher.yml"}

last_logged_time = {}
log_cooldown = 60

def save_recognized_face(frame, face_coords, name, confidence):
    x, y, w, h = face_coords
    face_img = frame[y:y + h, x:x + w]
```

```
    recognized_dir = "recognized_faces"
    os.makedirs(recognized_dir, exist_ok=True)
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"{recognized_dir}/{name}_{timestamp}_{int(confidence)}.jpg"
    cv2.imwrite(filename, face_img)
    return filename

def log_attendance(name, confidence, status="recognized", frame=None, face_coords=None):
    now = datetime.datetime.now()
    timestamp = now.strftime("%Y-%m-%d %H:%M:%S")
    date = now.strftime("%Y-%m-%d")
    time_only = now.strftime("%H:%M:%S")

    image_path = None
    if frame is not None and face_coords is not None and status == "recognized":
        image_path = save_recognized_face(frame, face_coords, name, confidence)

    if status == "unknown":
        log_entry = f"{timestamp} | СТАТУС: НЕОПОЗНАННЫЙ | Уверенность: {confidence}%\n"
        with open(ATTENDANCE_FILE, "a", encoding="utf-8") as f:
            f.write(log_entry)
        with open("unknown_faces_log.txt", "a", encoding="utf-8") as f:
            f.write(log_entry)
        print(f"[{timestamp}] Неопознанный человек (уверенность: {confidence}%)")

        conn = sqlite3.connect(ATTENDANCE_DB)
        cursor = conn.cursor()
        cursor.execute('''INSERT INTO visits (timestamp, date, time, name, confidence, status, image_path)
                          VALUES (?, ?, ?, ?, ?, ?, ?)''',
                       (timestamp, date, time_only, "Unknown", confidence, status, image_path))
        conn.commit()
        conn.close()
        return

    current_time = time.time()
    if name in last_logged_time and current_time - last_logged_time[name] < log_cooldown:
        return
    last_logged_time[name] = current_time

    log_entry = f"{timestamp} | ИМЯ: {name} | Уверенность: {confidence}%\n"
    csv_entry = f"{date},{time_only},{name},{confidence}%,recognized\n"

    with open(ATTENDANCE_FILE, "a", encoding="utf-8") as f:
        f.write(log_entry)
        f.write("-" * 60 + "\n")

    csv_file = "attendance.csv"
    if not os.path.exists(csv_file):
        with open(csv_file, "w", encoding="utf-8") as f:
            f.write("Date,Time,Name,Confidence,Status\n")
    with open(csv_file, "a", encoding="utf-8") as f:
        f.write(csv_entry)

    user_log_dir = "user_logs"
    os.makedirs(user_log_dir, exist_ok=True)
    user_file = os.path.join(user_log_dir, f"{name}.txt")
    with open(user_file, "a", encoding="utf-8") as f:
        f.write(f"{timestamp} - Уверенность: {confidence}% | Фото: {image_path or 'нет'}\n")
        f.write("-" * 40 + "\n")

    print(f"[{timestamp}] ЗАПИСАНО: {name} (уверенность: {confidence}%)")
```

```
    conn = sqlite3.connect(ATTENDANCE_DB)
    cursor = conn.cursor()
    cursor.execute('''INSERT INTO visits (timestamp, date, time, name, confidence, status, image_path)
                      VALUES (?, ?, ?, ?, ?, ?, ?)''',
                   (timestamp, date, time_only, name, confidence, status, image_path))
    conn.commit()
    conn.close()

def get_today_stats():
    today = datetime.datetime.now().strftime("%Y-%m-%d")
    conn = sqlite3.connect(ATTENDANCE_DB)
    cursor = conn.cursor()
    cursor.execute("SELECT name, COUNT(*) FROM visits WHERE date=? AND status='recognized' GROUP BY
name", (today,))
    stats = dict(cursor.fetchall())
    conn.close()
    return stats

def print_daily_report():
    stats = get_today_stats()
    today = datetime.datetime.now().strftime("%Y-%m-%d")
    print("\n" + "=" * 50)
    print(f"ОТЧЕТ ЗА {today}")
    print("=" * 50)
    if not stats:
        print("Посещений не зарегистрировано")
    else:
        for name, count in stats.items():
            print(f"  • {name}: {count} раз(а)")
    print("=" * 50 + "\n")

def load_users():
    if not os.path.exists(USERS_FILE):
        return {}
    with open(USERS_FILE, "r", encoding="utf-8") as f:
        data = json.load(f)
    return {int(k): v for k, v in data.items()}

USER_NAMES = load_users()

def load_recognizer(algorithm):
    if algorithm == "LBPH":
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        model_file = TRAINED_FILES["LBPH"]
    elif algorithm == "Fisherfaces":
        recognizer = cv2.face.FisherFaceRecognizer_create()
        model_file = TRAINED_FILES["Fisherfaces"]
    else:
        raise ValueError("Неизвестный алгоритм")

    if os.path.exists(model_file):
        recognizer.read(model_file)
        print(f" Модель {model_file} загружена")
        return recognizer
    else:
        print(f" ✗ Модель {model_file} не найдена! Запустите обучение.")
        return None

def trigger_servo_action():
    print("Успешное распознавание! Активация сервопривода...")
    set_servo_angle(90)
    time.sleep(5)
```

```
    set_servo_angle(0)
def process_frame(frame, recognizer, algorithm, face_cascade):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

    for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        try:
            resized = cv2.resize(roi_gray, (200, 200))
            predicted_id, confidence = recognizer.predict(resized)
        except:
            continue

        if algorithm == "LBPH":
            confidence_text = max(0, min(100, int(100 - confidence)))
        else:
            confidence_text = max(0, min(100, int((1000 - confidence) / 10)))

        face_coords = (x, y, w, h)

        if predicted_id in USER_NAMES and confidence_text > 15:
            name = USER_NAMES[predicted_id]
            label = f"{name} ({confidence_text}%)"
            color = (0, 255, 0)
            log_attendance(name, confidence_text, "recognized", frame, face_coords)
            trigger_servo_action()
        else:
            label = f"Неопознанный ({confidence_text}%)"
            color = (0, 0, 255)
            log_attendance("Unknown", confidence_text, "unknown", frame, face_coords)

        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_COMPLEX, 0.8, color, 2)

    return frame

def main():
    init_db()

    parser = argparse.ArgumentParser(description="Face Recognition System")
    parser.add_argument("--report", action="store_true")
    parser.add_argument("--algorithm", choices=["LBPH", "Fisherfaces"], default="LBPH")
    args = parser.parse_args()

    if args.report:
        print_daily_report()
        cleanup_servo()
        return

    recognizer = load_recognizer(args.algorithm)
    if recognizer is None:
        cleanup_servo()
        return

    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

    print("Откpываем камеру...")
    cap = cv2.VideoCapture(0, cv2.CAP_V4L2)
    if not cap.isOpened():
        cap = cv2.VideoCapture(0)

    if not cap.isOpened():
```

```
        print("✗Не удалось открыть камеру!")
        cleanup_servo()
        return

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

    print("Камера запущена. Нажмите 'q' для выхода.")

    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            processed = process_frame(frame, recognizer, args.algorithm, face_cascade)
            cv2.imshow("Face Recognition - Raspberry Pi 5", processed)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        cap.release()
        cv2.destroyAllWindows()
        cleanup_servo()

if __name__ == "__main__":
    main()
```

Trainer

```
import cv2
import os
import numpy as np
import json
from datetime import datetime

def train_model():
    print("=" * 70)
    print("ОБУЧЕНИЕ МОДЕЛИ РАСПОЗНАВАНИЯ ЛИЦ")
    print("=" * 70)

    dataset_dir = "dataset"
    if not os.path.exists(dataset_dir):
        print("✗Папка 'dataset' не найдена! Сначала создайте датасеты.")
        return

    print("Поиск датасетов пользователей...")

    faces = []
    labels = []
    label_to_name = {}
    current_id = 0

    # Проходим по всем папкам пользователей
    for user_name in os.listdir(dataset_dir):
        user_dir = os.path.join(dataset_dir, user_name)
        if not os.path.isdir(user_dir):
            continue

        print(f"Обработка пользователя: {user_name}")
        count = 0

        for image_name in os.listdir(user_dir):
            if image_name.endswith(('.jpg', '.jpeg', '.png')):
                img_path = os.path.join(user_dir, image_name)
```

```
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

            if img is None:
                continue

            faces.append(img)
            labels.append(current_id)
            count += 1

        if count > 0:
            label_to_name[current_id] = user_name
            print(f"  → Добавлено {count} изображений")
            current_id += 1

    if len(faces) == 0:
        print("✗ Не найдено ни одного изображения для обучения!")
        return

    print(f"\nНайдено {len(faces)} изображений от {len(label_to_name)} пользователей")

    # Обучение LBPH модели (лучше всего работает на Raspberry Pi)
    print("\nНачало обучения модели LBPH...")
    recognizer = cv2.face.LBPHFaceRecognizer_create()

    try:
        recognizer.train(faces, np.array(labels))
        model_file = "trained_faces_lbph.yml"
        recognizer.write(model_file)

        print(f"Модель успешно обучена и сохранена в {model_file}")

        # Обновляем users.json
        users = {str(k): v for k, v in label_to_name.items()}
        with open("users.json", "w", encoding="utf-8") as f:
            json.dump(users, f, ensure_ascii=False, indent=4)

        print(f"Файл users.json обновлён ({len(users)} пользователей)")

        # Показываем статистику
        print("\n" + "=" * 50)
        print("✓ ОБУЧЕНИЕ ЗАВЕРШЕНО УСПЕШНО!")
        print("=" * 50)
        for id_, name in label_to_name.items():
            print(f"  • ID {id_}: {name}")
        print("=" * 50)

    except Exception as e:
        print(f"✗ Ошибка при обучении: {e}")

def main():
    print("1. Обучить модель (LBPH)")
    print("2. Выход")

    choice = input("\nВыберите действие (1-2): ").strip()

    if choice == "1":
        train_model()
        input("\nНажмите Enter для возврата в меню...")
    else:
        print("Выход.")
```

```
if __name__ == "__main__":
    main()
```

run

```
import os
import subprocess
import sys
import time

def clear_screen():
    os.system('cls' if os.name == 'nt' else 'clear')

def print_header():
    print("=" * 85)
    print(" СИСТЕМА РАСПОЗНАВАНИЯ ЛИЦ ДЛЯ RASPBERRY PI 5")
    print(" ЕДИНЫЙ ЗАПУСКАТОР")
    print("=" * 85)
    print()

def run_script(script_name, description):
    print(f"\n Запуск: {description}...")
    print("-" * 70)
    try:
        subprocess.run([sys.executable, script_name], check=True)
    except FileNotFoundError:
        print(f" ✗ Файл {script_name} не найден!")
        print("Убедитесь, что файл находится в той же папке.")
        time.sleep(2)
    except Exception as e:
        print(f" ✗ Ошибка: {e}")
        time.sleep(2)

def main():
    while True:
        clear_screen()
        print_header()

        print("1.  Запустить распознавание лиц + сервопривод")
        print("2. Создать датасет нового пользователя")
        print("3.  Обучить модель (trainer.py)")
        print("4.  Показать отчет за сегодня")
        print("5.  Перезапустить меню")
        print("6.  Выход")

        print("\n" + "-" * 85)
        choice = input("Выберите действие (1-6): ").strip()

        if choice == "1":
            run_script("recognition.py", "Распознавание лиц")

        elif choice == "2":
            run_script("dataset.py", "Создание датасета")

        elif choice == "3":
            run_script("trainer.py", "Обучение модели")

        elif choice == "4":
            print("\n Формирование отчета...")
            run_script("recognition.py", "Отчет")

        elif choice == "5":
            print("Перезапуск меню...")
```

```
            time.sleep(1)
            continue

        elif choice == "6":
            print("\n До свидания! Система завершена.")
            break

        else:
            print("✗Некорректный выбор! Введите число от 1 до 6.")
            time.sleep(1.5)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("\n\nПрограмма остановлена.")
    except Exception as e:
        print(f"Критическая ошибка: {e}")
    finally:
        print("Выход из системы.")
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Фото и чертежи

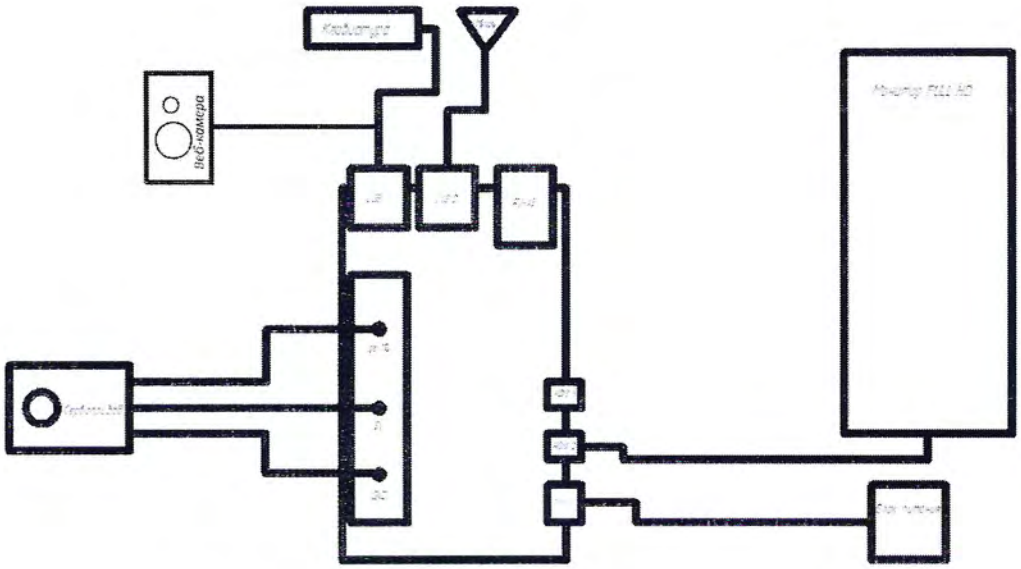

Рисунок 1 - схема подключения устройства

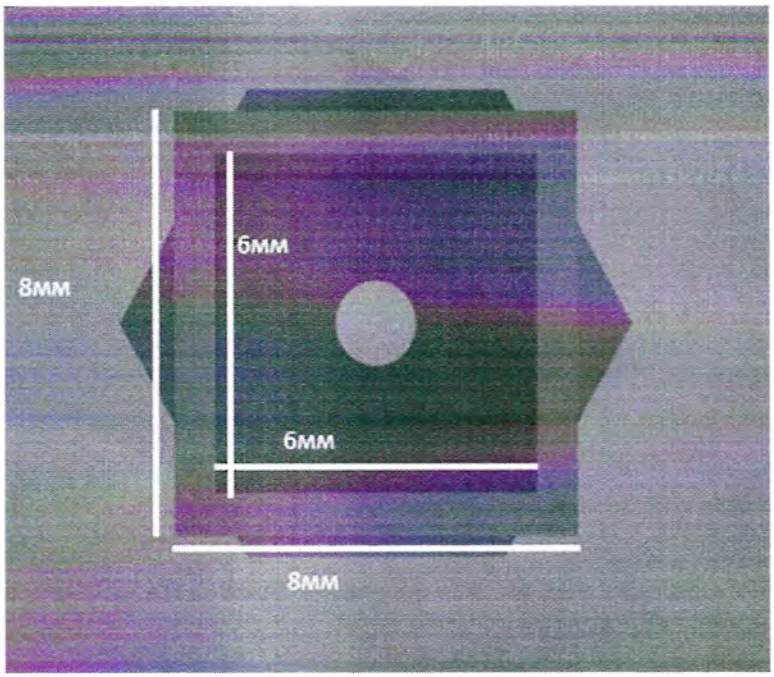

Рисунок 2 - Насадка на сервопривод (вид сверху)

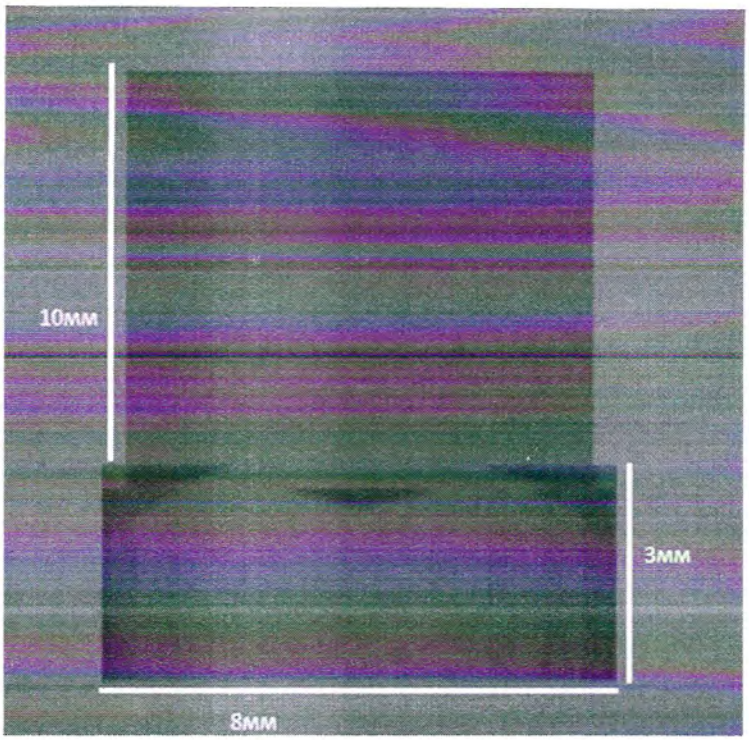

Рисунок 3 - Насадка на сервопривод (вид сбоку)

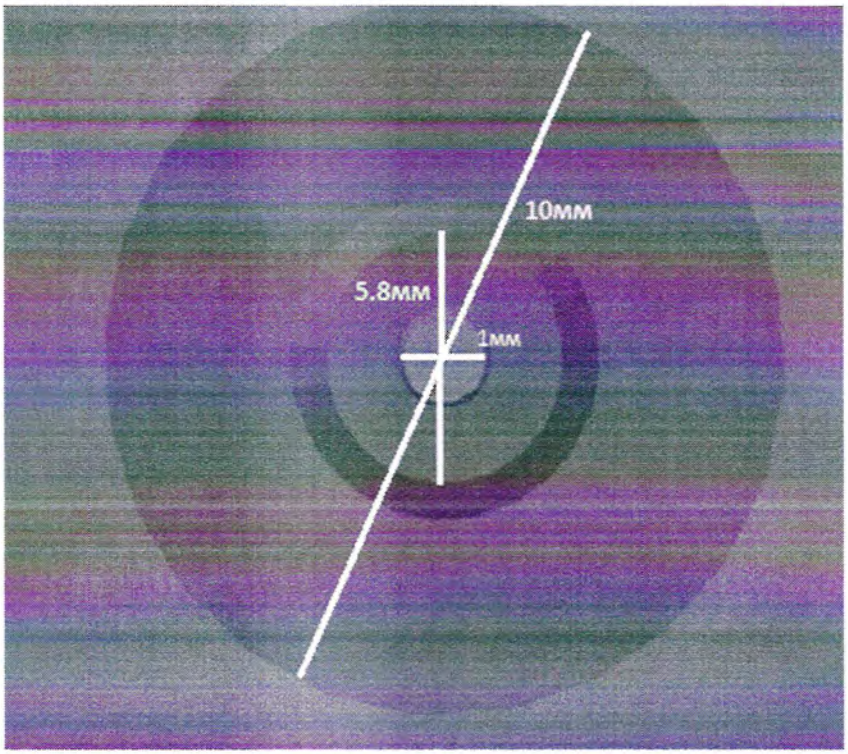

Рисунок 4 - Насадка на сервопривод (вид снизу)

ПРИЛОЖЕНИЕ В

(обязательное)

Результаты выполнения программы

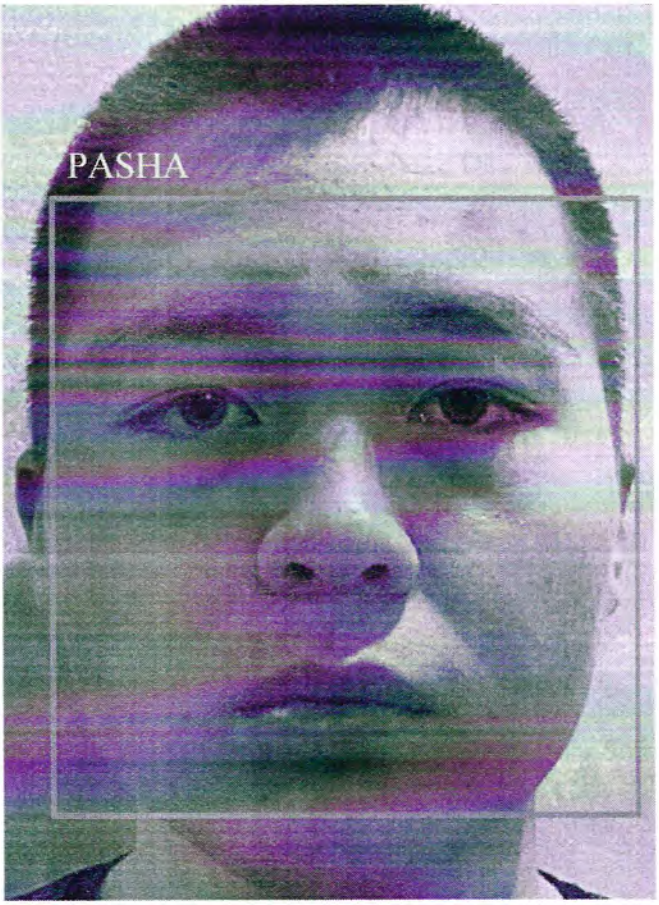

Рисунок 4 - Результат распознавания (1 лицо)

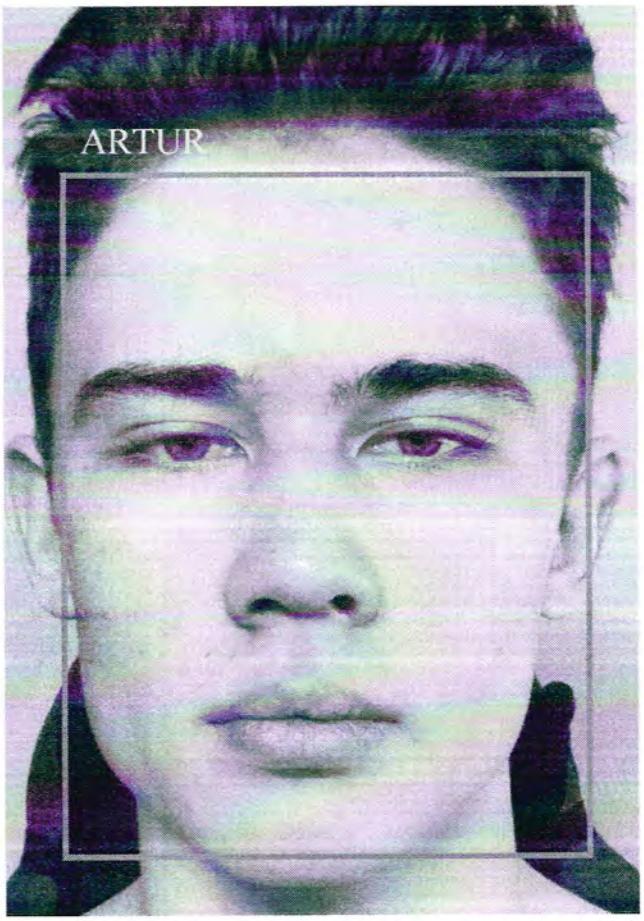

Рисунок 5 - Результат распознавания (2 лицо)

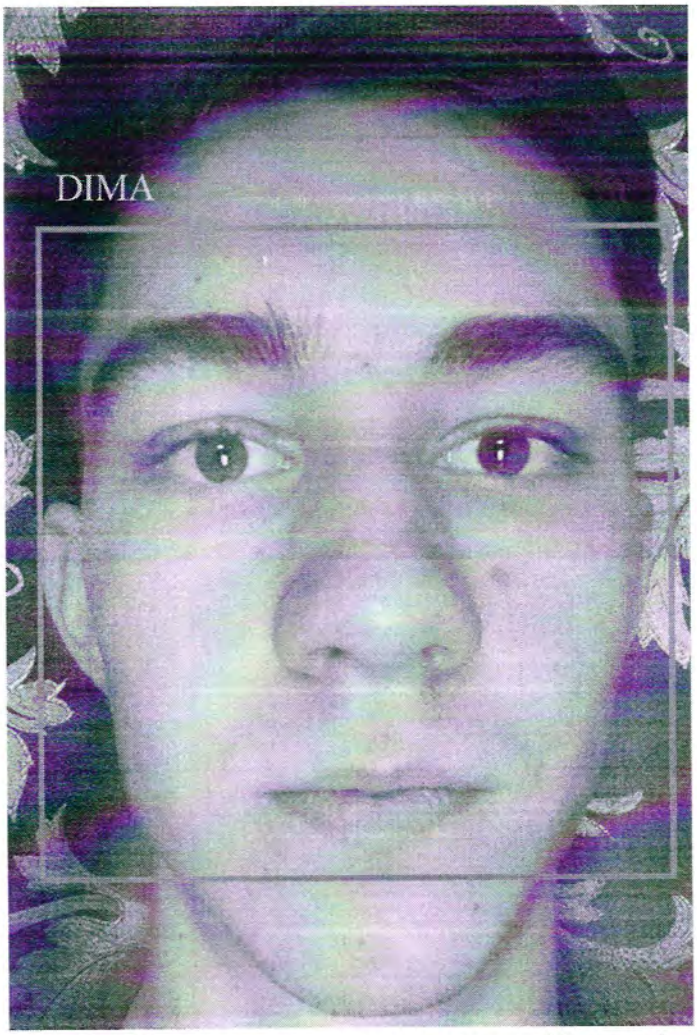

Рисунок 6 - Результат распознавания (3 лицо)

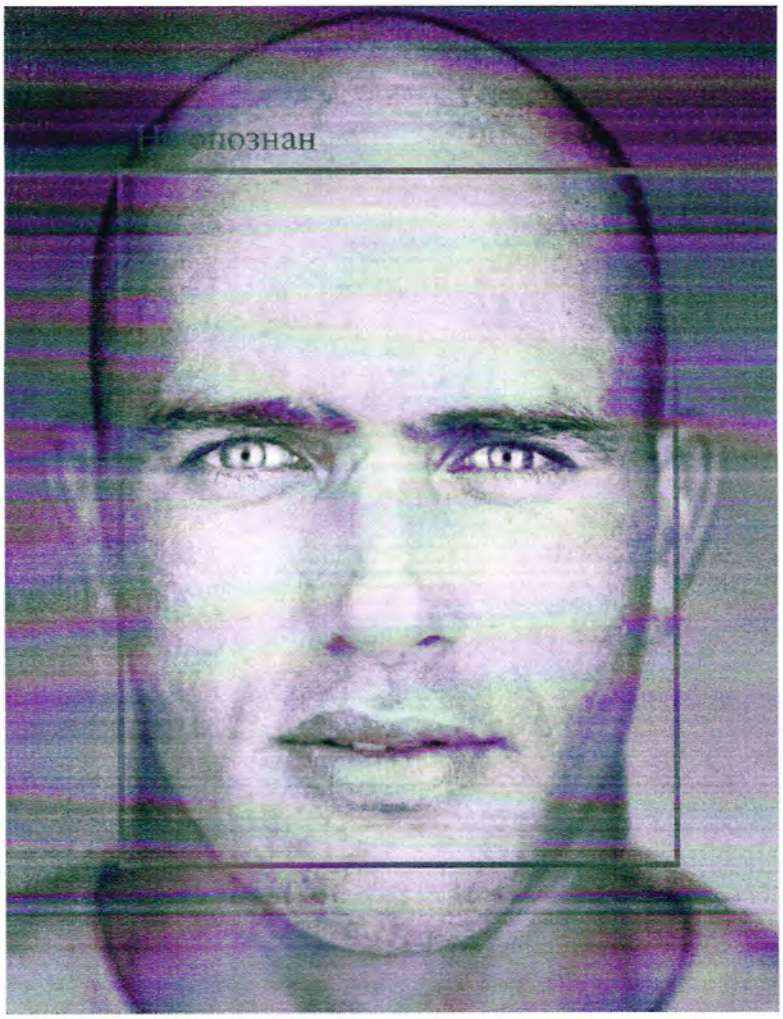

Рисунок 6 - Результат распознавания (4 лицо)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«16» 06 2026 г.

Декан ФЭУ

_____ А.С. Гудим
(подпись)

«16» 06 2026 г.

УТВЕРЖДАЮ

И.о. проректора по научной работе, д-р техн. наук, профессор

_____ А.В. Космынин
(подпись)

«16» 06 2026 г.

АКТ
о приемке в эксплуатацию проекта
«Разработка телемеханической системы управления манипулятором»

г. Комсомольск-на-Амуре «16» 06 2026 г.

Комиссия в составе представителей:

со стороны заказчика

- С.И. Сухоруков – руководитель СКБ,
- А.С. Гудим – декан ФЭУ

со стороны исполнителя

- Н.Н. Любушкина – руководителя проекта,
- Д.А. Цыган – группа 5ПЭб-1,
- П.А. Усов – группа 5ПЭб-1,
- А.А. Кузнецов – группа 5ПЭб-1,

составила акт о нижеследующем:

«Исполнитель» передает проект «Электронный замок с системой распознавания», в составе:

1. Механическое устройство замка в распаячной коробке;

2. Система распознавания лиц и система управления электронным замком;

3. Программное обеспечения для работы устройства на съемном носителе;

4. Паспорт устройства.

Руководитель проекта — 15.06.2026 (подпись, дата) — Н.Н. Любушкина

Исполнители проекта — 19.06.2026 (подпись, дата) — Д.А. Цыган

19.06.2026 (подпись, дата) — П.А. Усов

19.06.2026 (подпись, дата) — А.А. Кузнецов