МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

УТВЕРЖДАЮ
Заведующий кафедрой
_____ М.Л. Долженкова
«___» _____ 202__ г.

ЗАДАНИЕ
на курсовой проект

по дисциплине **Цифровая схемотехника**
полное название дисциплины

Студенту Черкасову Александру Андреевичу обучающемуся на образовательной программе

09.03.01 – Информатика и вычислительная техника
профиль: Инженерия программного и аппаратного обеспечения
полное название образовательной программы

2 — курс обучения; очная — форма обучения

Тема работы (проекта): Разработка системы «Персонализированный мониторинг и коррекция состояния здоровья на основе биометрических данных». Часть 1.

1. Исходные данные: Техническое задание на разработку устройства, документация по готовой системе мониторинга состояния, документация по набору датчиков, требования ГОСТ и СТП ВятГУ.

2. Основные разделы: Анализ технического задания, разработка алгоритмов функционирования устройства, построение схемы работы устройства, выбор компонентов, разработка принципиальной схемы, разработка специализированного ПО, тестирование разработанного ПО, разработка инструкции пользователя.

3. Перечень графического материала (с точным указанием обязательных чертежей): Схема функционирования устройства (1хА3), Схема взаимодействия модулей системы (1хА3), Схема электрическая принципиальная (1хА3), Структура интерфейса (1хА3)

4. Оформление пояснительной записки и чертежей производится в соответствии с требованиями СТП ВятГУ, ГОСТов и методическими указаниями по выполнению курсовых проектов по дисциплинам.

Дата выдачи задания: 09.02.2026

Срок сдачи работы (проекта): 01.06.2026

Руководитель _____ / Коротких Д.А. _____.
Подпись *Ф.И.О.* *Дата*

Консультант _____ / Мельцов В.Ю. _____.
Подпись *Ф.И.О.* *Дата*

Задание принял _____ / Черкасов А.А. _____.
Подпись *Ф.И.О.* *Дата*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

УТВЕРЖДАЮ
Заведующий кафедрой
_____ М.Л. Долженкова
«___» _____ 202__ г.

ЗАДАНИЕ
на курсовой проект

по дисциплине **Цифровая схемотехника**
полное название дисциплины

Студенту Зинину Владиславу Алексеевичу обучающемуся на образовательной программе

09.03.01 – Информатика и вычислительная техника
профиль: Инженерия программного и аппаратного обеспечения
полное название образовательной программы

2 — курс обучения; очная — форма обучения

Тема работы (проекта): Разработка системы «Персонализированный мониторинг и коррекция состояния здоровья на основе биометрических данных». Часть 2.

1. Исходные данные: Техническое задание на разработку устройства, документация по готовой системе мониторинга состояния, документация по набору датчиков, требования ГОСТ и СТП ВятГУ.

2. Основные разделы: Анализ технического задания, разработка диаграммы развёртывания для описания системы, построение модели функционирования, выбор компонентов, разработка принципиальной схемы, разработка специализированного ПО, тестирование разработанного ПО, разработка инструкции пользователя.

3. Перечень графического материала (с точным указанием обязательных чертежей): Диаграмма развёртывания (1хА3), Схема взаимодействия модулей системы (1хА3), Схема электрическая принципиальная (1хА3). Структура интерфейса (1хА3).

4. Оформление пояснительной записки и чертежей производится в соответствии с требованиями СТП ВятГУ, ГОСТов и методическими указаниями по выполнению курсовых проектов по дисциплинам.

Дата выдачи задания: 09.02.2026

Срок сдачи работы (проекта): 01.06.2026

Руководитель _____ / Коротких Д.А. _____.
Подпись *Ф.И.О.* *Дата*

Консультант _____ / Мельцов В.Ю. _____.
Подпись *Ф.И.О.* *Дата*

Задание принял _____ / Зинин В.А. _____.
Подпись *Ф.И.О.* *Дата*

Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Примечание
				Документация		
А4		1	ТПЖА.090301.838 ПЗ	Пояснительная записка	53	
				Графический материал		
А3		1	ТПЖА.090301.838-01 ДПЛ	Схема взаимодействия пользователя с системой	1	Плакат
А3		2	ТПЖА.090301.838-02 ДПЛ	Схема взаимодействия модулей системы	1	Плакат
А3		3	ТПЖА.090301.838-03 ДПЛ	Диаграмма развёртывания	1	Плакат
А3		4	ТПЖА.090301.838-04 ДПЛ	Схема электрическая принципиальная	1	Плакат
А3		5	ТПЖА.090301.838-05 ДПЛ	Структура интерфейса	1	Плакат
А3		6	ТПЖА.090301.838-06 ДПЛ	Схема функционирования устройства	1	Плакат

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Черкасов/Зинин		
Пров.		Коротких		
Консул.		Мельцов		
Н.контр.				
Утв.		Долженкова		

ТПЖА.090301.838 ДДП

Персонализированный мониторинг и коррекция состояния здоровья на основе биометрических данных

Лит.	Лист	Листов
		1

Кафедра ЭВМ
ИВТб-2301-05-00

Копировал

Формат А4

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

(ФГБОУ ВО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущено к защите

Заведующий кафедрой ЭВМ

_____/Долженкова М. Л./

«__»_____2026г.

Персонализированный мониторинг и коррекция состояния здоровья на основе биометрических данных

Пояснительная записка курсового проекта по дисциплине

«Цифровая схемотехника»

ТПЖА.090301.838 ПЗ

Разработали _____ /Черкасов А. А./

_____ /Зинин В. А./

Руководитель _____ /Коротких Д. А./

Консультант _____ /Мельцов В. Ю./

Проект защищен с оценкой «_____» _____

(оценка) (дата)

Члены комиссии _____ /_____/

(подпись) (ФИО)

_____ /_____/

(подпись) (ФИО)

_____ /_____/

(подпись) (ФИО)

Киров 2026

Реферат

Черкасов А. А., Зинин В. А. Персонализированный мониторинг и коррекция состояния здоровья на основе биометрических данных ТПЖА.090301.838 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Коротких Д. А., конс. Мельцов В. Ю. - Киров, 2026. – ПЗ 43с, 1 рис., 3 табл., 5 прил.

БИОСИГНАЛЫ, МОНИТОРИНГ ЗДОРОВЬЯ, БИОМЕТРИЧЕСКИЕ ДАННЫЕ, МАШИННОЕ ОБУЧЕНИЕ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ

Объектом исследования являются физиологические показатели человека и методы их регистрации.

Предметом исследования выступают алгоритмы сбора данных с ESP32-S3, ИИ-модели классификации и видеоанализ MediaPipe.

Целью работы является проектирование доступного программно-аппаратного комплекса персонализированного мониторинга и коррекции состояния здоровья пользователя на основе биометрических данных. Система призвана собирать сырые биосигналы (пульс, электромиограмма — ЭМГ, электроэнцефалограмма — ЭЭГ, кожно-гальваническая реакция — КГР/GSR) с носимых устройств на базе микроконтроллера ESP32-S3, обрабатывать их с помощью алгоритмов машинного обучения и предоставлять пользователю наглядный анализ состояния (уровень стресса, утомление, фазы восстановления) через интерактивный веб-интерфейс, а также давать персональные рекомендации по восстановлению и тренировкам.

Результатом выполнения курсового проекта стал программно-аппаратный комплекс, позволяющий отслеживать биосигналы в реальном времени, оценивать уровень стресса и правильность выполнения упражнений.

В перечень графического материала входят: диаграмма развертывания, алгоритм функционирования устройства, схема электрическая принципиальная, структура интерфейса.

Содержание

Введение **6**

Список сокращений и условных обозначений **7**

1 Анализ предметной области **9**

1.1 Проблематика 9

1.2 Существующие решения 10

1.3 Постановка задачи 11

2 Техническое задание **11**

2.1 Цель разработки 11

2.2 Требования к функциональности 12

2.3 Требования к интерфейсу 12

2.4 Требования к надежности и безопасности 13

2.5 Требования к техническому и программному обеспечению 13

3 Проектирование системы **14**

3.1 Общая схема процесса 15

3.2 Проект 1: «СОМАТОПОС» (Команда СОМАТОПОС) 19

3.3 Проект 2: «Интерны» (Команда «Интерны») 20

3.4 Проект 3: «Код здоровья» (Команда «Код здоровья») 21

3.5 Проект 4: «Чумной Доктор» (Команда «Чумной Доктор») 21

3.6 Алгоритмическое обеспечение 22

3.6.1 Цифровая фильтрация сигналов 23

3.6.2 Детектирование пиков пульсовой волны и расчет ЧСС . . . 23

3.6.3 Статистический анализ вариабельности ритма сердца (ВРС) 24

3.6.4 Масштабирование показателей КГР (GSR) в уровень стресса 24

3.6.5 Расчет мышечного напряжения по ЭМГ 25

3.6.6 Анализ спектральных ритмов ЭЭГ 25
3.7 Информационное обеспечение 26
3.7.1 Общая абстрактная ERD-схема системы 26
3.7.2 Схема данных проекта «СОМАТОПОС» (Команда 1) 28
3.7.3 Схема данных проекта «Интерны» (Команда 2) 29
3.7.4 Схема данных проекта «Код здоровья» (Команда 3) 31
3.7.5 Схема данных проекта «Чумной Доктор» (Команда 4) . . . 33

4 Выбор и обоснование инструментов реализации 34
4.1 Общие критерии выбора 34
4.2 Обоснование выбора микроконтроллера ESP32-S3 35
4.3 Обоснование выбора протокола передачи данных UDP 36
4.4 Языки программирования 37
4.5 Графические и веб-фреймворки 37
4.6 Базы данных 37
4.7 Искусственный интеллект 37

5 Реализация системы 38
5.1 Общие модули 38
5.2 Особенности программного обеспечения микроконтроллера 38
5.3 Особенности реализации проекта «Код здоровья» 40
5.4 Особенности реализации проекта «Чумной Доктор» 40
5.5 Интерфейсы пользователя 41

6 Тестирование 42
6.1 Методика тестирования 42
6.2 Результаты тестирования 42
6.3 Примеры экранов тестирования 43
6.4 Заключение по тестированию 45

Заключение 46

Список использованных источников 47

Список литературы 47

Приложение А **48**
А.1 Фрагменты кода прошивки микроконтроллера ESP32-S3 (C++) . . . 48
А.2 Модуль приема и парсинга 11-байтовых UDP-пакетов 49
А.3 Модуль скелетного трекинга ИИ-тренера (MediaPipe) 50
А.4 Обработчик API запросов ИИ рекомендаций на Flask 51
А.5 Сценарий инициализации схемы базы данных SQLite (Python) . . . 52

Введение

В современном мире сохранение здоровья и контроль физиологического состояния стали важнейшими приоритетами. Высокий уровень стресса, умственное переутомление, недостаток физической активности или, наоборот, перегрузки в процессе спортивных тренировок негативно сказываются на работоспособности и качестве жизни. Особенно остро эти проблемы проявляются у студентов в период сессий, ИТ-специалистов, диспетчеров и спортсменов в процессе реабилитации.

Традиционные методы контроля здоровья часто требуют использования дорогостоящего медицинского оборудования или визита к врачу. В то же время существующие бытовые фитнес-трекеры имеют ограниченный набор датчиков и не способны комплексно оценивать состояние нервной и мышечной систем.

Целью работы является проектирование доступного программно-аппаратного комплекса персонализированного мониторинга и коррекции состояния здоровья пользователя на основе биометрических данных.

Для достижения поставленной цели необходимо решить следующие задачи:

1) Выполнить анализ предметной области и существующих решений в области носимых систем экспресс-диагностики состояния здоровья.
2) Сформулировать технические требования к разрабатываемому программно-аппаратному комплексу.
3) Спроектировать общую архитектуру системы и разработать алгоритмы обработки биосигналов (ЭКГ, ЭМГ, КГР).
4) Выбрать и обосновать аппаратные средства и программные инструменты реализации компонентов комплекса.
5) Реализовать программное обеспечение сбора данных, серверные компоненты, ИИ-модели оценки функционального состояния и пользовательский веб-интерфейс.
6) Провести комплексное тестирование разработанной системы и оценить стабильность ее функционирования.

Список сокращений и условных обозначений

АЦП — аналогово-цифровой преобразователь;

ВРС — вариабельность ритма сердца;

ИИ — искусственный интеллект;

КГР — кожно-гальваническая реакция;

ЛФК — лечебная физкультура;

СУБД — система управления базами данных;

ЧСС — частота сердечных сокращений;

ЭКГ — электрокардиография;

ЭМГ — электромиография;

ЭЭГ — электроэнцефалография;

API (Application Programming Interface) — программный интерфейс приложения;

BPM (Beats Per Minute) — удары в минуту;

CLI (Command-Line Interface) — интерфейс командной строки;

CRC (Cyclic Redundancy Check) — циклический избыточный код;

CSV (Comma-Separated Values) — значения, разделённые запятыми;

DHCP (Dynamic Host Configuration Protocol) — протокол динамической конфигурации узла;

DSP (Digital Signal Processing) — цифровая обработка сигналов;

ERD (Entity-Relationship Diagram) — диаграмма «сущность–связь»;

FPS (Frames Per Second) — кадры в секунду;

GSR (Galvanic Skin Response) — кожно-гальваническая реакция (см. КГР);

HTTP (HyperText Transfer Protocol) — протокол передачи гипертекста;

JSON (JavaScript Object Notation) — текстовый формат обмена данными;

NPM (Node Package Manager) — менеджер пакетов Node.js;

OLED (Organic Light-Emitting Diode) — органический светоизлучающий диод;

pNN50 — процент последовательных NN-интервалов, различающихся более чем на 50 мс;

REST (Representational State Transfer) — архитектурный стиль взаимодействия компонентов;

RMSSD (Root Mean Square of Successive Differences) — среднеквадратичное отклонение разностей последовательных NN-интервалов;

SDNN (Standard Deviation of NN intervals) — стандартное отклонение NN-интервалов;

TCP (Transmission Control Protocol) — протокол управления передачей;

UDP (User Datagram Protocol) — протокол пользовательских датаграмм;

XOR (eXclusive OR) — исключающее «ИЛИ».

1 Анализ предметной области

Анализ предметной области показывает растущую потребность в доступных носимых системах экспресс-диагностики. Основными факторами риска в повседневной жизни человека являются синдром хронической усталости, переутомление и вегетативный стресс.

1.1 Проблематика

Пользователи часто не способны объективно оценить уровень своего утомления. Студенты в период подготовки к экзаменам или операторы сложных технических систем игнорируют первые признаки снижения концентрации и переутомления, что ведет к когнитивным ошибкам, профессиональному выгоранию и ухудшению показателей здоровья.

В сфере спорта и реабилитации отсутствие обратной связи о тонусе мышц и динамике пульса ведет к риску перетренированности или повторного травмирования. Физическое переутомление сопровождается накоплением молочной кислоты в мышечных тканях и изменением амплитудно-частотных характеристик электромиограммы (ЭМГ), а вегетативный стресс напрямую отражается на вариабельности ритма сердца (ВРС) и электрическом сопротивлении кожи (кожно-гальваническая реакция, КГР). Своевременное выявление этих отклонений позволяет предотвратить патологические состояния.

Основными целевыми группами разрабатываемого комплекса являются:

- **Спортсмены в процессе реабилитации** после травм опорно-двигательного аппарата, которым необходим строгий контроль симметрии и тонуса мышц, а также правильности траекторий суставов при выполнении упражнений лечебной физкультуры (ЛФК);
- **ИТ-специалисты, диспетчеры и операторы**, подверженные высокому уровню статического и психоэмоционального напряжения;

– **Студенты и научные сотрудники**, для которых критически важно контролировать фазы когнитивной активности и отдыха во избежание умственного истощения.

1.2 Существующие решения

На рынке представлены следующие категории аналогичных систем:

– **Спортивные смарт-часы** (Apple Watch, Garmin) — отлично отслеживают пульс и фазы сна, однако не поддерживают анализ ЭЭГ (электрической активности мозга) и ЭМГ (тонуса мышц) в реальном времени.

– **Медицинские регистраторы** (холтеровские мониторы) — обладают высочайшей точностью, но неудобны для повседневного использования и требуют расшифровки врачом-специалистом.

– **Компьютерные трекеры осанки и зрения** — контролируют физическое присутствие и позу по веб-камере, но не имеют доступа к внутренним физиологическим показателям организма.

Для наглядности сравнения существующих решений с разрабатываемой системой составлена таблица 1.

Таблица 1 – Сравнение разрабатываемой системы с существующими аналогами

Критерий	**Смарт-часы**	**Холтеры (ЭКГ)**	**ПК-камеры**	**Разрабатываемый комплекс**
Регистрация ЭЭГ/ЭМГ	Нет	Ограниченно (только ЭКГ)	Нет	Да (полный спектр биосигналов)
Видеоаналитика	Нет	Нет	Да (осанка, лицо)	Да (скелетная модель суставов)
Беспроводная передача	Bluetooth (низкая скорость)	Нет (запись на карту памяти)	Не требуется (локально)	Wi-Fi UDP (низкая задержка, реал-тайм)
Интеграция ИИ	Базовые пороги	Ручной анализ врачом	Ограниченные правила	Нейросети и классификаторы
Стоимость	Высокая стоимость	Требует мед. лицензии	Низкая (веб-камера)	Низкая себестоимость (ESP32-S3)

Главным недостатком аналогов является отсутствие комплексной интерпретации биосигналов в связке с компьютерным зрением (видеоанализом движений) для контроля правильности упражнений.

1.3 Постановка задачи

Интегрированная система должна решить следующие задачи:

- Сбор данных с сенсоров (пульс, ЭМГ, ЭЭГ, КГР) с помощью микроконтроллера ESP32-S3 и их беспроводная передача по протоколу UDP.
- Компьютерный анализ техники выполнения упражнений с помощью веб-камеры и библиотек машинного зрения.
- Классификация состояния пользователя по 5 категориям («Напряжение», «Утомление», «Восстановление», «Стресс», «Перегрузка»).
- Визуализация данных в реальном времени на веб-панели (живые графики, карточки показателей).
- Хранение истории измерений и авторизация пользователей.

2 Техническое задание

Техническое задание определяет требования к архитектурным компонентам и функциям системы диагностического мониторинга.

2.1 Цель разработки

Создание кроссплатформенного программно-аппаратного комплекса для сбора, фильтрации, классификации биосигналов и визуализации состояния здоровья пользователя.

2.2 Требования к функциональности

Система должна обеспечивать:

- Стабильное считывание данных с аналогово-цифровых преобразователей (АЦП) микроконтроллера с частотой дискретизации не менее 100 Гц.
- Фильтрацию сетевых помех (50 Гц) и двигательных артефактов на стороне сервера.
- Обучение и инференс ИИ-моделей для распознавания стресса по вариабельности сердечного ритма (HRV) и спектральной плотности ЭМГ.
- Отслеживание опорно-двигательного аппарата по камере (33 ключевые точки тела) для ИИ-тренера.
- Передачу данных от датчиков по протоколу UDP с контролем целостности пакетов (XOR-контрольная сумма).
- Логирование сессий в форматах CSV и JSON.
- Визуализацию обрабатываемых биосигналов и результатов классификации ИИ в реальном времени на веб-панели.
- Авторизацию пользователей и разграничение прав доступа для ролей «пациент», «врач» и «администратор».

2.3 Требования к интерфейсу

Для реализации функциональных требований, указанных в подразделе 2.2, веб-интерфейс должен содержать:

- Панель авторизации и регистрации пациентов/врачей.
- Дашборд с графиками сигналов пульса, ЭМГ, ЭЭГ и КГР, обновляемыми по WebSockets/polling.
- Индикатор текущего состояния с оценкой уверенности ИИ-модели.
- Раздел «ИИ-тренер» с окном видеотрансляции и текстовыми рекомендациями.
- Страницу истории сессий с таблицей и пагинацией.

2.4 Требования к надежности и безопасности

Для обеспечения безопасной и надежной эксплуатации комплекса должны соблюдаться следующие требования:

- **Электробезопасность:** Аппаратная часть на базе ESP32-S3 должна питаться исключительно от автономного низковольтного источника постоянного тока (гальваническая батарея или аккумулятор с напряжением до 5 В). Запрещается прямое подключение устройства к сети 220 В в процессе снятия биосигналов с тела человека.
- **Безопасность контактов:** Электроды датчиков ЭКГ/ЭМГ и ЭЭГ должны быть изготовлены из гипоаллергенных медицинских материалов (серебро/хлорид серебра Ag/AgCl) для исключения раздражения кожи при длительном мониторинге.
- **Помехоустойчивость и контроль целостности:** При использовании протокола UDP для беспроводной передачи данных обязателен побайтовый контроль целостности с помощью XOR контрольной суммы в каждом пакете. Пакеты с нарушенной контрольной суммой должны автоматически отбрасываться на стороне сервера.
- **Отказоустойчивость соединения:** При кратковременной потере Wi-Fi соединения (до 10 секунд) сервер должен переходить в режим ожидания данных, сохраняя текущую сессию пользователя. При длительном разрыве связи сессия автоматически завершается с сохранением накопленных данных в локальную базу данных.

2.5 Требования к техническому и программному обеспечению

Для полноценного функционирования программно-аппаратного комплекса предъявляются следующие системные требования к рабочей станции пользователя (серверная и клиентская части):

- **Процессор:** Четырехъядерный CPU с тактовой частотой не менее 2.5 ГГц (поддержка набора инструкций AVX2 обязательна для оптимизации инференса MediaPipe и TensorFlow).
- **Оперативная память:** Не менее 8 Гбайт (рекомендуется 16 Гбайт для параллельного запуска локальных Flask-серверов и систем визуализации React/Vite).
- **Видеокарта:** Встроенный графический чип или дискретный видеоадаптер с поддержкой аппаратного ускорения веб-камеры и веб-интерфейсов (рекомендуется видеокарта NVIDIA с поддержкой CUDA для ускорения глубокого обучения).
- **Периферийные устройства:** Внешняя или встроенная веб-камера с разрешением не менее 1280x720 пикселей и частотой захвата кадров не менее 30 FPS.
- **Сетевой адаптер:** Беспроводной модуль Wi-Fi стандарта 802.11b/g/n (2.4 ГГц) с поддержкой протокола UDP.
- **Операционная система:** Microsoft Windows 10/11, Ubuntu 20.04/22.04 LTS, macOS Catalina или новее.
- **Программное окружение:** Python 3.8–3.12 (для работы ИИ-моделей), СУБД SQLite 3, Node.js 18.0+ (для фронтенда), Node Package Manager (NPM) для сборки веб-компонентов.

3 Проектирование системы

Проектирование интегрированной программно-аппаратной системы включает разработку архитектурных решений, протоколов сетевого взаимодействия, структуры базы данных и алгоритмов обработки сигналов. Система строится на основе трехуровневой архитектуры:

1) **Аппаратный уровень (Edge Layer)** — сенсорные модули для съема физиологических показателей (ЭКГ, ЭМГ, ЭЭГ, КГР), подключенные к микроконтроллеру ESP32-S3, который осуществляет дискретизацию сигналов и их беспроводную передачу.

2) **Серверный уровень (Application Layer)** — сервер на базе Python (Flask / Express [1] / UDP-сокет), выполняющий прием UDP-пакетов, их фильтрацию и цифровую обработку, классификацию функционального состояния пользователя с помощью алгоритмов машинного обучения и сохранение сессий в базу данных SQLite.
3) **Клиентский уровень (Presentation Layer)** — веб-интерфейс для визуализации графиков в реальном времени, отображения результатов ИИ-тренера и предоставления личного кабинета пользователя.

3.1 Общая схема процесса

Общая аппаратная часть системы стандартизирована для всех проектов и включает микроконтроллер ESP32-S3, осуществляющий непрерывный опрос четырех аналоговых датчиков: пульсометра (ЭКГ), электромиографа (ЭМГ), электроэнцефалографа (ЭЭГ) и кожно-гальванической реакции (КГР/GSR). Считывание данных с датчиков выполняется с помощью встроенного 12-разрядного АЦП микроконтроллера (диапазон значений `0-4095`) с частотой дискретизации 50 Гц.

Для беспроводной передачи данных используется протокол UDP по Wi-Fi. Данные упаковываются в помехоустойчивый 11-байтовый пакет со следующей структурой:

- `data[0]`: `0xAA` (стартовый байт);
- `data[1-2]`: значение АЦП датчика пульса (2 байта);
- `data[3-4]`: значение АЦП датчика ЭМГ (2 байта);
- `data[5-6]`: значение АЦП датчика КГР (2 байта);
- `data[7-8]`: значение АЦП датчика ЭЭГ (2 байта);
- `data[9]`: XOR-контрольная сумма байт с 1 по 8;
- `data[10]`: `0x55` (стоповый байт).

Алгоритм функционирования устройства представлен на рисунке 1, схема электрическая принципиальная, общая для всех проектов — на рисунке 2, Схема взаимодействие модулей внутри системы представлена на рисунке 3.

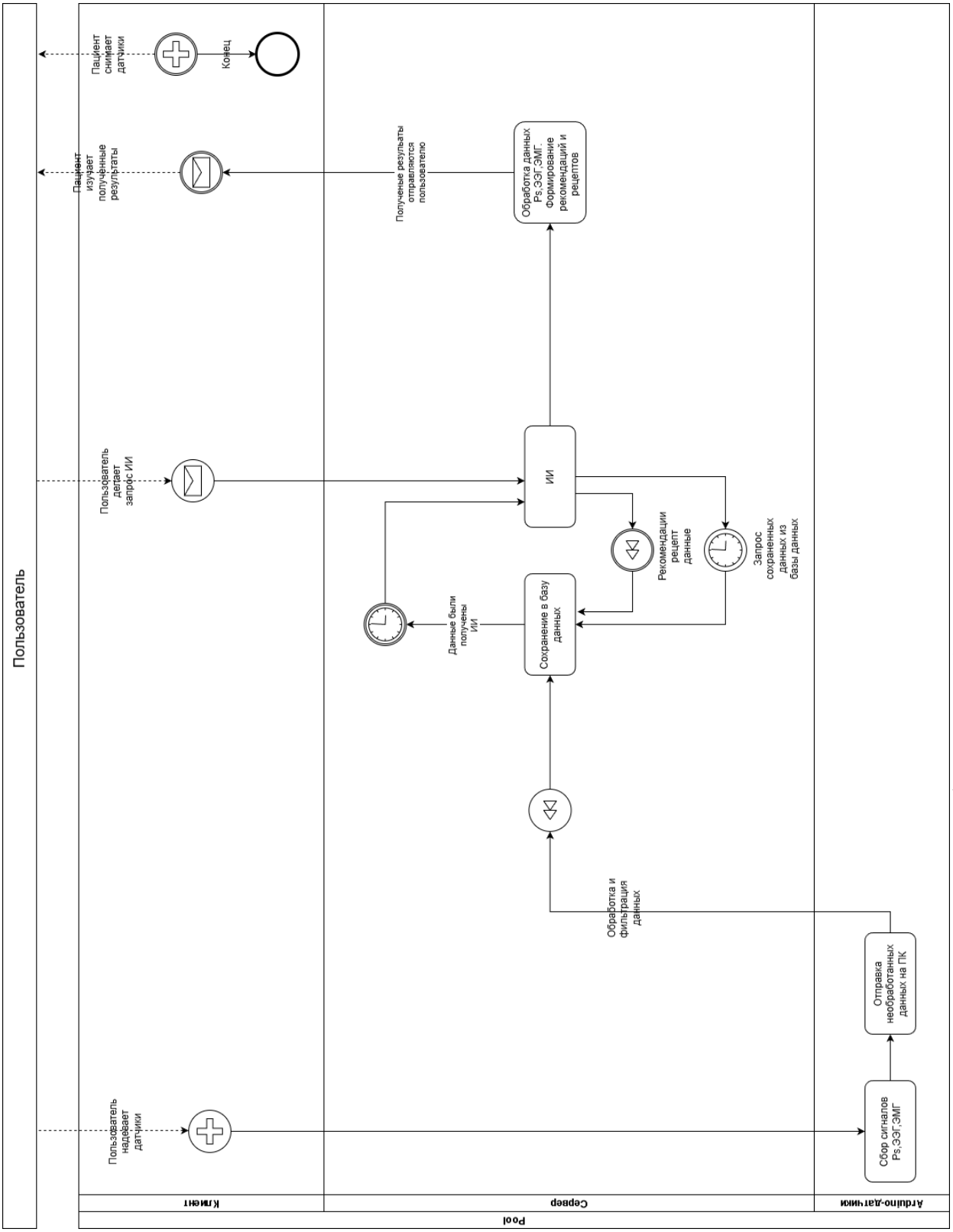

Рисунок 1 – Алгоритм функционирования устройства

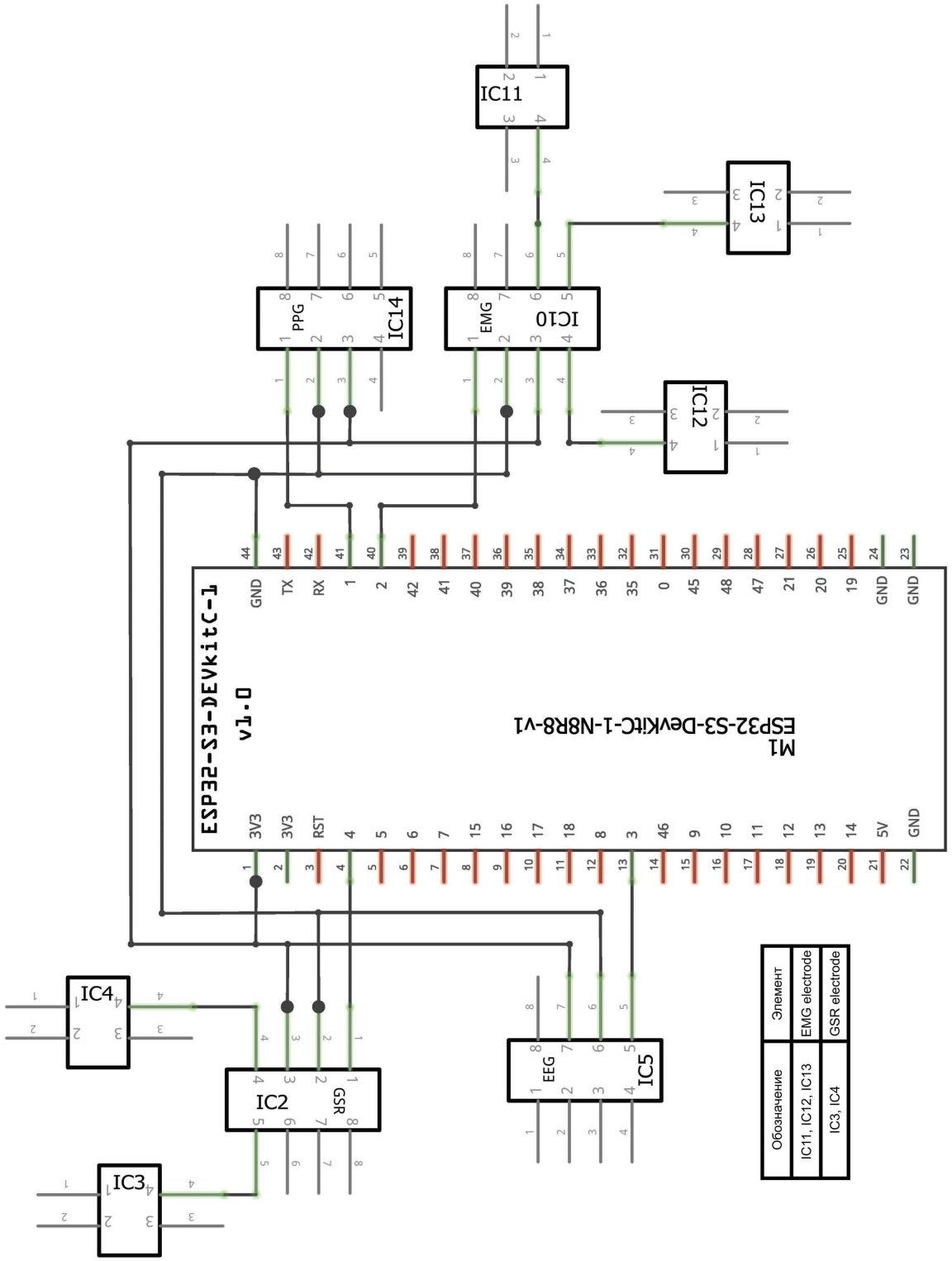

Рисунок 2 – Схема электрическая принципиальная

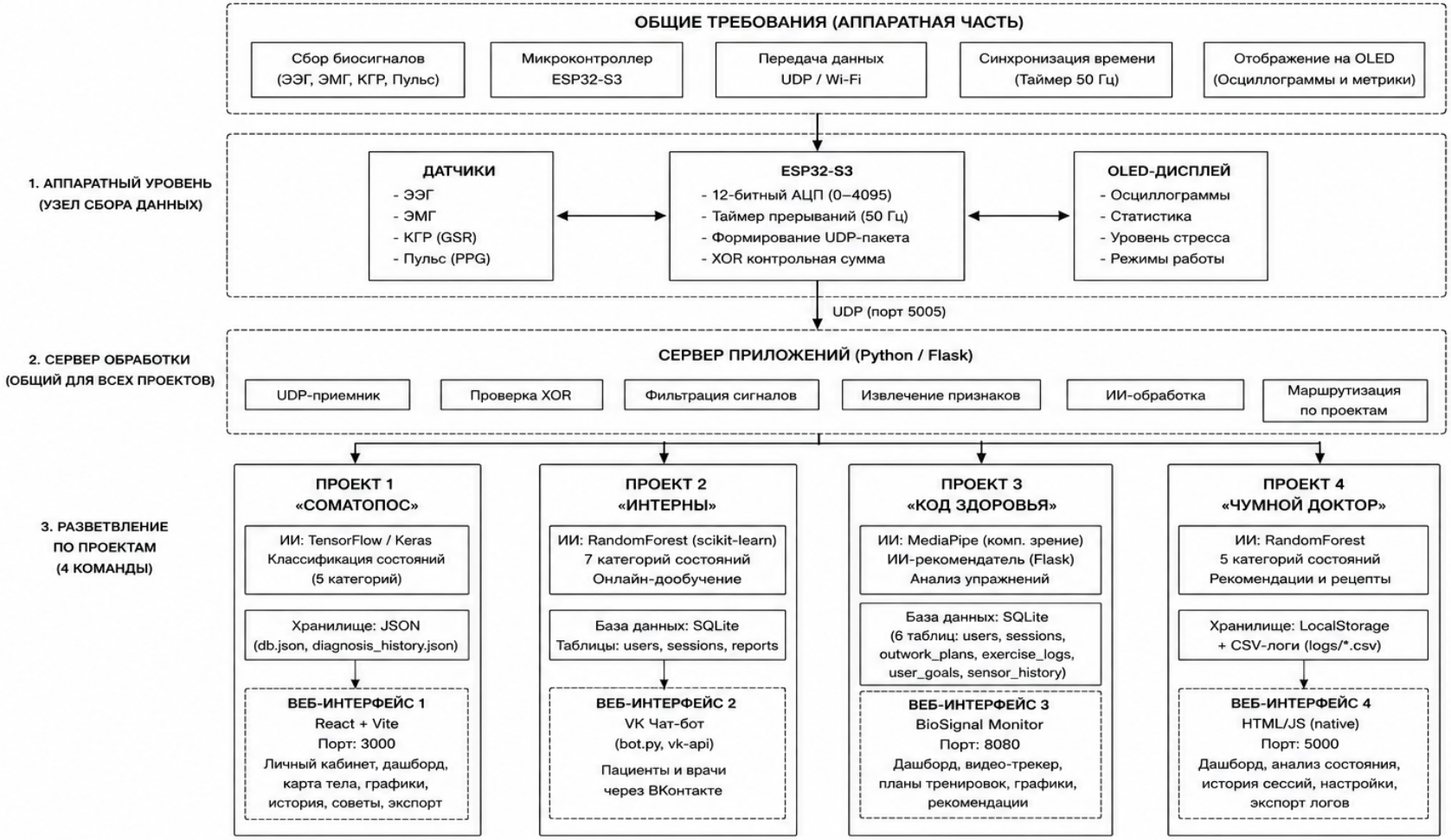

Рисунок 3 – Схема взаимодействия внутри системы

Цикл работы системы:

1) Аппаратные датчики непрерывно регистрируют биосигналы с тела пользователя.
2) Микроконтроллер ESP32-S3 по аппаратному таймеру опрашивает каналы АЦП, вычисляет контрольную сумму, формирует 11-байтовый буфер и отправляет его по UDP на IP-адрес сервера (порт `5005`).
3) Python-сервер принимает UDP-пакеты, проверяет их целостность по XOR-контрольной сумме (отбрасывая битые кадры), выполняет масштабирование и фильтрацию данных.
4) Модули искусственного интеллекта на стороне сервера обрабатывают очищенные сигналы, классифицируют функциональное состояние пользователя по 5 категориям («Напряжение», «Утомление», «Восстановление», «Стресс», «Перегрузка») и генерируют рекомендации.
5) Веб-интерфейс получает обработанные данные по протоколу HTTP (REST API) или HTTP-polling и визуализирует их на графиках в реальном времени.

3.2 Проект 1: «СОМАТОПОС» (Команда СОМАТОПОС)

Проект ориентирован на классификацию физиологического состояния по биосигналам пульса, ЭМГ и ЭЭГ с использованием моделей глубокого обучения на базе TensorFlow/Keras.

– **ИИ-компонент**: Нормализация данных с использованием сохраненных параметров скейлера `scaler.pkl` и классификация через нейросеть (TensorFlow/Keras) по карте меток `label_map.json` для выявления рисков сердечно-сосудистой системы, мышечного перенапряжения или аномальной мозговой активности в режиме реального времени. История классификаций сохраняется в файле `diagnosis_history.json`.

– **Интерфейс**: Интерактивный дашборд «Цифровой доктор», выполненный в футуристичном дизайне Apple Liquid Glass (порт `3000`). Отображает графики пульса, ЭМГ и ЭЭГ, текущий вердикт нейросети и процент уверенности модели. В интерфейс интегрирована интерактивная

визуальная карта тела (с интерактивной подсветкой зон головы, сердца, мышц кора и конечностей), позволяющая наглядно отображать возможные сбои в организме на основе классификации биосигналов. Данные пользователей (логины, пароли при регистрации) сохраняются локально в файле `db.json`.

– **Особенности запуска**: Для работы системы требуется запустить три параллельных процесса: `headless_diagnostic.py` для приема данных и классификации, `npm run dev` для веб-интерфейса и `send_test.py` для генерации тестового сигнала в отсутствие реальных датчиков.

3.3 Проект 2: «Интерны» (Команда «Интерны»)

Проект фокусируется на построении надежной клиент-серверной архитектуры с авторизацией пользователей и ведением базы данных измерений.

– **ИИ-компонент**: Использует интерактивную самообучающуюся ИИ-модель на базе классификатора `RandomForestClassifier` и `StandardScaler` (библиотека `scikit-learn` [3]). Классифицирует состояние на 7 категорий: «Норма», «Напряжение», «Усталость», «Восстановление», «Стресс», «Перегрузка» и «Аритмия». Модель поддерживает динамическое online-дообучение на основе ручных корректировок заключения ИИ, вводимых врачом.

– **База данных**: Хранение истории сессий измерений, учетных записей врачей и пациентов, а также отчетов ИИ в реляционной базе данных под управлением **SQLite** (таблицы `users`, `sessions`, `reports`).

– **Интерфейс**: Взаимодействие реализовано через чат-бота ВКонтакте (`bot.py` на базе `vk-api`). Пациенты могут запустить сессию обследования и просматривать историю своих сессий, а лечащие врачи — просматривать статистику точности ИИ и модерировать заключения (подтверждать или корректировать) непосредственно через кнопочный интерфейс чата, автоматически запуская дообучение модели.

3.4 Проект 3: «Код здоровья» (Команда «Код здоровья»)

Проект представляет гибридное решение, объединяющее физиологический мониторинг с компьютерным зрением для персональных тренировок и реабилитации.

- **Аппаратная часть**: Дополнительно к стандартной аппаратной части требуется внешняя веб-камера, подключенная к ПК.
- **ИИ-тренер**: Видеоанализ упражнений в реальном времени с использованием библиотеки `MediaPipe` [2] (Pose Landmark Detection). Алгоритм строит скелетную модель пользователя по 33 точкам, рассчитывает тригонометрические углы в суставах (локтевых, коленных) и сравнивает их с эталонными траекториями, мгновенно сообщая об ошибках в технике выполнения и помогая предотвратить травмы при ЛФК и тренировках.
- **ИИ-рекомендатель**: Интеллектуальный помощник Flask-сервера (Python 3.12) генерирует персонализированные планы упражнений на основе заданных целей (общий тонус, похудение, набор массы, выносливость, реабилитация) и уровней сложности (легкий, средний, сложный).
- **Интерфейс**: Панель BioSignal Monitor (порт `8080`) с карточками показателей, живыми графиками физиологических сигналов и окном видеотрекера с наложением скелетной сетки. Базовые учетные данные по умолчанию: логин/пароль `123`.

3.5 Проект 4: «Чумной Доктор» (Команда «Чумной Доктор»)

Проект выделяется развитым консольным CLI-интерфейсом для разработчиков и исследователей, гибкой настройкой порогов ИИ-моделей, а также облачным развертыванием.

- **ИИ-компонент**: Включает Random Forest классификатор для определения 5 функциональных состояний (Напряжение, Утомление, Восстановление, Стресс, Перегрузка). ИИ также формирует рекомендации и рецепты с указанием возможных препаратов.

– **Пошаговый CLI-конвейер**: Процесс разработки разбит на скрипты:
  1) `step1_train_v8_final.py` — обучение модели классификатора;
  2) `step2_test_v8_full_report.py` — автономное тестирование модели с генерацией подробных отчетов;
  3) `step3_realtime_v8_2.py` — мониторинг в реальном времени с выводом в консоль и ведением CSV-логов в папке `logs_realtime/` с именами вида `drhouse_YYYY-MM-DD_HH-MM-SS.csv`.
– **Настройка параметров**: Конфигурационный класс `Config` позволяет настраивать: частоту анализа `ANALYZE_EVERY = 1.0` с, размер истории `HISTORY_SIZE = 5`, очистку консоли, пороги пульса для выявления брадикардии (ниже `50` BPM) и тахикардии (выше `130` BPM), а также порог уверенности `CONFIDENCE_THRESHOLD = 0.55`. Если уверенность модели ниже 55%, выводится вердикт «НЕОПРЕДЕЛЁННО».
– **Интерфейс**: Веб-интерфейс запускается локально на порту `5000`. Авторизация: `admin` / `123456`. Связь с сервером реализована по протоколу HTTP (REST API), а приём данных с АЦП — через UDP-сокет. Имеется блок «Анализ состояния», история записей с пагинацией, кнопка записи сессий, а также JS-контроллеры для регулировки усиления и калибровки каждого датчика.

3.6 Алгоритмическое обеспечение

Алгоритмическое обеспечение системы решает задачи фильтрации сырых сигналов, выделения физиологических признаков и классификации состояния пользователя в реальном времени.

3.6.1 Цифровая фильтрация сигналов

Сырые биосигналы, поступающие с АЦП микроконтроллера, подвержены влиянию сетевых помех (наводка 50 Гц от электрических сетей) и двигательных артефактов. Для их устранения применяется цифровой режекторный фильтр (Notch filter) 50 Гц. Передаточная функция фильтра в Z-области имеет вид:

$$H(z) = \frac{1 - 2\cos(\omega_0)z^{-1} + z^{-2}}{1 - 2r\cos(\omega_0)z^{-1} + r^2 z^{-2}} \tag{1}$$

где $\omega_0 = \frac{2\pi f_0}{f_s}$, $f_0 = 50$ Гц (частота режекции), f_s — частота дискретизации сигнала ($f_s > 2f_0$ по теореме Найквиста), а $r \in [0.95, 0.99]$ — коэффициент, задающий ширину полосы подавления.

Разностное уравнение для реализации фильтра во временной области:

$$y[n] = x[n] - 2\cos(\omega_0)x[n-1] + x[n-2] + 2r\cos(\omega_0)y[n-1] - r^2 y[n-2] \tag{2}$$

где $x[n]$ — входной отсчет сигнала, $y[n]$ — отфильтрованный отсчет.

3.6.2 Детектирование пиков пульсовой волны и расчет ЧСС

Для расчета частоты сердечных сокращений (BPM) в реальном времени используется пороговый алгоритм детектирования пиков пульсовой волны:

1) Сигнал опрашивается с частотой 50 Гц.
2) Фиксируется переход сигнала через верхний порог $T_{peak} = 2800$ (для 12-битного АЦП):

$$S[n] > T_{peak} \quad \text{и} \quad S[n-1] \leq T_{peak} \tag{3}$$

3) Измеряется временной интервал $\Delta t = t_n - t_{n-1}$ (в миллисекундах) между двумя последовательными пиками (NN-интервал).
4) Если интервал лежит в физиологических границах 400 мс $< \Delta t <$ 1500 мс (что соответствует ЧСС от 40 до 150 уд/мин), производится

расчет мгновенной ЧСС:

$$BPM = \frac{60000}{\Delta t} \tag{4}$$

3.6.3 Статистический анализ вариабельности ритма сердца (ВРС)

Для оценки активности вегетативной нервной системы (баланса симпатического и парасимпатического отделов) рассчитываются временные показатели ВРС:

– Среднеквадратичное отклонение разностей последовательных NN-интервалов (RMSSD), отражающее парасимпатическую активность:

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(\Delta t_{i+1} - \Delta t_i)^2} \tag{5}$$

– Стандартное отклонение NN-интервалов (SDNN), характеризующее общую вариабельность:

$$SDNN = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\Delta t_i - \overline{\Delta t})^2} \tag{6}$$

3.6.4 Масштабирование показателей КГР (GSR) в уровень стресса

Проводимость кожи (обратная величина сопротивления) напрямую коррелирует с уровнем стресса и активностью потовых желез. Значение АЦП масштабируется в процентный показатель по формуле:

$$Stress_{\%} = \frac{GSR_{raw} - GSR_{min}}{GSR_{max} - GSR_{min}} \times 100 \tag{7}$$

где $GSR_{min} = 500$, $GSR_{max} = 3800$ — минимальные и максимальные калибровочные границы АЦП. Итоговое значение ограничивается диапазоном $[0, 100]\%$.

3.6.5 Расчет мышечного напряжения по ЭМГ

Амплитуда электромиограммы отражает силу сокращения мышц. Для оценки мышечной нагрузки вычисляется размах сигнала в скользящем окне размером $W = 50$ отсчетов (1 секунда):

$$A_{EMG} = \max_{k \in [0, W-1]} (EMG[n-k]) - \min_{k \in [0, W-1]} (EMG[n-k]) \tag{8}$$

Если $A_{EMG} > 100$ единиц АЦП, мышечное напряжение рассчитывается по формуле:

$$MuscleLoad_{\%} = \frac{A_{EMG} - 100}{24} \tag{9}$$

Результат приводится к диапазону $[0, 100]\%$.

3.6.6 Анализ спектральных ритмов ЭЭГ

Нормализованное значение мозговой активности EEG_{norm} приводится к диапазону $[0, 100]\%$ на основе максимального значения АЦП (4095):

$$EEG_{norm} = \frac{EEG_{raw}}{4095} \times 100 \tag{10}$$

Соотношение альфа- и бета-ритмов оценивается приближенно по формулам:

$$AlphaRatio_{\%} = 30 + 0.5 \times (100 - EEG_{norm}) \tag{11}$$

$$BetaRatio_{\%} = 0.8 \times EEG_{raw}/40.95 \tag{12}$$

3.7 Информационное обеспечение

Информационное обеспечение интегрированной программно-аппаратной системы построено с учетом различных подходов команд к организации хранения данных. Архитектура объединяет как реляционные СУБД для надежного транзакционного хранения, так и документо-ориентированные форматы (JSON) и файловые структуры (CSV) для логов.

3.7.1 Общая абстрактная ERD-схема системы

Для интеграции всех модулей была спроектирована общая концептуальная схема данных (рисунок 4), отражающая логическую сущность процесса мониторинга. Она включает в себя следующие основные сущности:

- `users`: содержит данные зарегистрированных в системе пользователей (пациентов, врачей, администраторов) и включает: уникальный идентификатор (‘id‘), имя пользователя (‘username‘), зашифрованный пароль (‘password‘), роль в системе (‘role‘) и метку времени создания учетной записи (‘created_at‘);
- `sessions`: хранит информацию о сеансах физиологического мониторинга и включает: уникальный идентификатор сессии (‘id‘), внешний ключ пользователя (‘user_id‘), время запуска сессии (‘created_at‘), статус завершения (‘status‘), а также усредненные показатели датчиков за сессию — частоту сердечных сокращений (‘bpm‘), мышечное напряжение (‘muscle‘), активность мозга (‘brain‘) и общий уровень стресса (‘stress‘);
- `reports`: фиксирует ИИ-заключения и врачебные отчеты по результатам сессий и включает: уникальный идентификатор отчета (‘id‘), внешний ключ сессии (‘session_id‘), внешний ключ лечащего врача (‘doctor_id‘), автоматический вердикт нейросети (‘ai_conclusion‘), рекомендации и предписания врача (‘doctor_conclusion‘), текущий статус проверки отчета (‘status‘) и время создания (‘created_at‘).

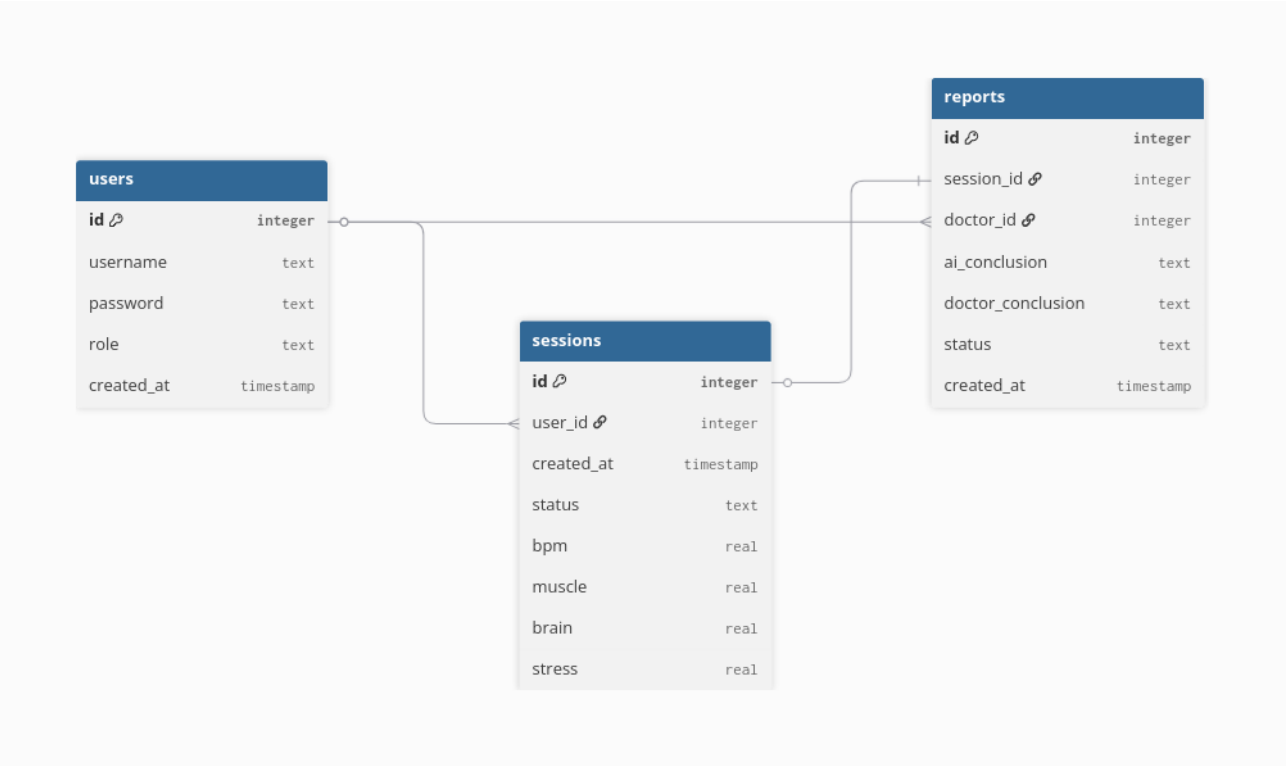

Рисунок 4 – Общая абстрактная ERD-схема системы

3.7.2 Схема данных проекта «СОМАТОПОС» (Команда 1)

Проект СОМАТОПОС ориентирован на файловое хранение данных в формате JSON. Схема связей логических сущностей представлена на рисунке 5. Хранилище включает в себя следующие основные таблицы и структуры:

- `users`: содержит учетные данные пользователей для авторизации в системе и включает: уникальное имя пользователя (‘name‘), выступающее в качестве первичного ключа, и пароль (‘password‘);
- `history`: хранит исторические логи сеансов мониторинга и включает: имя пользователя (‘user‘), дату проведения измерения (‘date‘), среднюю ЧСС (‘hr‘), показатели активности мышц (‘muscle‘) и мозга (‘brain‘), а также общее состояние пользователя (‘state‘);
- `diagnosis_history`: содержит логи классификации состояний нейросетью с высокой частотой дискретизации и включает: временную метку (‘timestamp‘), выступающую первичным ключом, строковое представление даты и времени (‘datetime‘), текущую ЧСС (‘bpm‘), расчетные показатели ЭМГ (‘muscle‘) и ЭЭГ (‘eeg‘), код классифицированного диагноза (‘diagnosis_code‘), краткое (‘diagnosis_short‘) и развернутое (‘diagnosis_full‘) текстовые заключения ИИ, процент уверенности модели (‘confidence‘), а также сырые оцифрованные сигналы АЦП датчиков пульса (‘raw_pulse‘), ЭМГ (‘raw_emg‘) и ЭЭГ (‘raw_eeg‘).

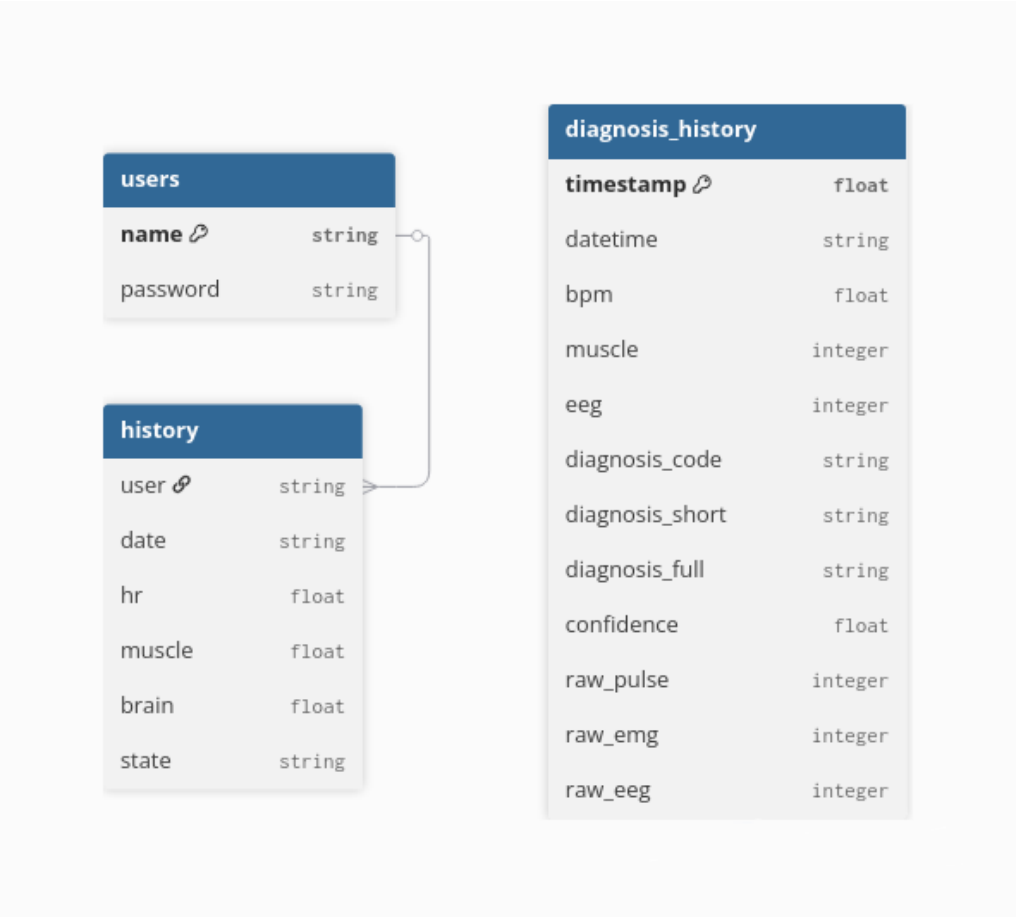

Рисунок 5 – ERD-схема базы данных проекта «СОМАТОПОС»

3.7.3 Схема данных проекта «Интерны» (Команда 2)

Реализована реляционная база данных под управлением СУБД SQLite. Архитектура базы данных представлена на рисунке 6 и включает три основные сущности (сценарий инициализации схемы базы данных SQLite приведен в Приложении А, подраздел А.5):

– `users`: хранит информацию о пользователях системы и включает: уникальный идентификатор (‘id‘), идентификатор пользователя в социальной сети ВКонтакте (‘vk_id‘) для работы чат-бота, имя пользователя (‘name‘), роль пользователя (‘role‘) и дату регистрации (‘created_at‘);

– `sessions`: фиксирует запущенные сессии медицинского обследования пациентов и включает: уникальный идентификатор сессии (‘id‘), внешний ключ пациента (‘user_id‘), текущий статус обследования (‘status‘),

сырые текстовые данные измерений в формате JSON (‘raw_data‘) и время старта сессии (‘created_at‘);

– `reports:` содержит заключение и отчеты по сессиям и включает: уникальный идентификатор отчета (‘id‘), внешний ключ обследования (‘session_id‘), автоматические рекомендации искусственного интеллекта (‘ai_conclusion‘), экспертное заключение лечащего врача (‘doctor_conclusion‘), внешний ключ врача (‘doctor_id‘), статус верификации отчета (‘status‘) и время формирования (‘created_at‘).

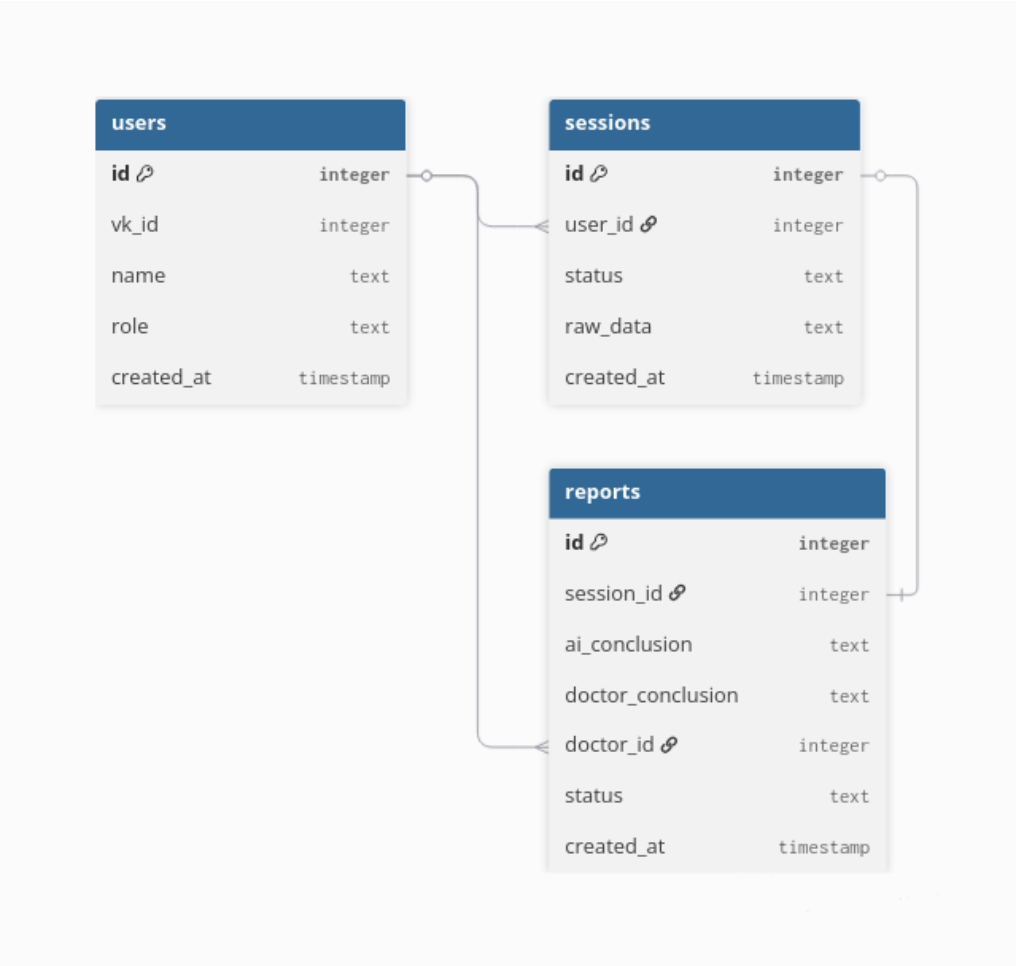

Рисунок 6 – ERD-схема базы данных проекта «Интерны»

3.7.4 Схема данных проекта «Код здоровья» (Команда 3)

База данных проекта, реализованная в MedApp на базе SQLite, имеет расширенную реляционную структуру (рисунок 7). Она включает следующие сущности для ИИ-тренировок и мониторинга:

- `users:` содержит профильные и учетные данные участников системы и включает: уникальный идентификатор пользователя (‘id‘), логин для авторизации (‘username‘), адрес электронной почты (‘email‘), ФИО (‘full_name‘), дату рождения (‘birth_date‘), роль в системе (‘role‘), хеш пароля (‘password‘) и дату регистрации (‘created_at‘);
- `sessions:` фиксирует метаданные тренировочных и реабилитационных сессий и включает: уникальный идентификатор сессии (‘id‘), внешний ключ пользователя (‘user_id‘), название тренировки (‘name‘), время начала (‘start_time‘) и завершения (‘end_time‘), количество записанных точек данных (‘data_points‘), время сохранения сессии (‘created_at‘) и дополнительную мета-информацию в формате JSON (‘meta_json‘);
- `workout_plans:` содержит тренировочные программы, сгенерированные ИИ-помощником, и включает: уникальный идентификатор плана (‘id‘), внешний ключ пользователя (‘user_id‘), описание упражнений в формате JSON (‘plan_json‘), уровень сложности (‘difficulty‘), цель программы (‘goal‘) и время создания (‘created_at‘);
- `exercise_logs:` записывает детальную историю выполнения отдельных упражнений в рамках планов и включает: уникальный идентификатор записи (‘id‘), внешний ключ пользователя (‘user_id‘), внешний ключ плана тренировок (‘plan_id‘), строковый идентификатор упражнения (‘exercise_id‘), название упражнения (‘exercise_name‘), количество выполненных повторений (‘reps‘), субъективную сложность (‘difficulty‘), длительность выполнения в секундах (‘duration_sec‘) и время завершения (‘completed_at‘);
- `user_goals:` содержит спортивные и реабилитационные цели пользователей и включает: уникальный идентификатор цели (‘id‘), внешний ключ пользователя (‘user_id‘), текстовое описание цели (‘goal_text‘), целевое число повторений (‘target_reps‘), целевое упраж-

нение (‘target_exercise‘), статус достижения цели (‘achieved‘) и время создания (‘created_at‘);

– **sensor_history:** хранит временные ряды показаний сенсоров биосигналов и включает: уникальный идентификатор записи (‘id‘), внешний ключ логина пользователя (‘username‘), текущую ЧСС (‘bpm‘), уровень стресса (‘stress‘), активность мышц (‘muscle‘), спектральные показатели альфа- (‘alpha‘) и бета-ритмов ЭЭГ (‘beta‘), текущее состояние (‘state‘) и время записи (‘recorded_at‘).

Рисунок 7 – ERD-схема базы данных проекта «Код здоровья»

3.7.5 Схема данных проекта «Чумной Доктор» (Команда 4)

Проект совмещает локальную клиентскую базу данных LocalStorage для хранения пользовательских сессий и ведение CSV-логов на стороне Python-сервера для детального анализа сигналов. Архитектура логического хранения представлена на рисунке 8. База данных и лог-файлы включают в себя следующие сущности и поля:

- `users` **(LocalStorage):** хранит информацию об учетных записях пользователей для авторизации в локальной панели управления и включает: уникальный идентификатор пользователя (‘id‘), имя пользователя (‘username‘), пароль (‘pass‘) и роль в системе (‘role‘);
- `rows` **(LocalStorage):** содержит агрегированные результаты сеансов физиологического мониторинга и включает: уникальный идентификатор записи (‘id‘), внешний ключ пользователя (‘uid‘), дату проведения измерения (‘date‘), среднюю частоту пульса (‘bpm‘), средний уровень стресса (‘stress‘), среднее мышечное напряжение (‘muscle‘), среднюю долю альфа-ритма ЭЭГ (‘alpha‘), среднюю долю бета-ритма ЭЭГ (‘beta‘), текстовое определение физиологического состояния (‘state‘), выписанные врачебные рекомендации (‘recipe‘) и имя лечащего врача (‘doctor‘);
- **CSV-логи (Python-сервер):** представляют собой файлы непрерывной записи показаний датчиков с высокой частотой дискретизации для последующего оффлайн-анализа. Каждая строка лог-файла содержит следующие поля: временную метку получения данных (‘Время‘), порядковый номер пакета для контроля потерь (‘Пакет‘), скользящее среднее частоты сердечных сокращений (‘Пульс_mean‘), среднюю амплитуду ЭМГ (‘ЭМГ_mean‘), среднее значение проводимости кожи КГР (‘КГР_mean‘), нормализованный показатель ЭЭГ (‘ЭЭГ_mean‘), классифицированное нейросетью состояние (‘Состояние‘), вероятность правильности распознавания (‘Уверенность_%‘) и код критического состояния в случае выхода параметров за нормы (‘Критическое‘).

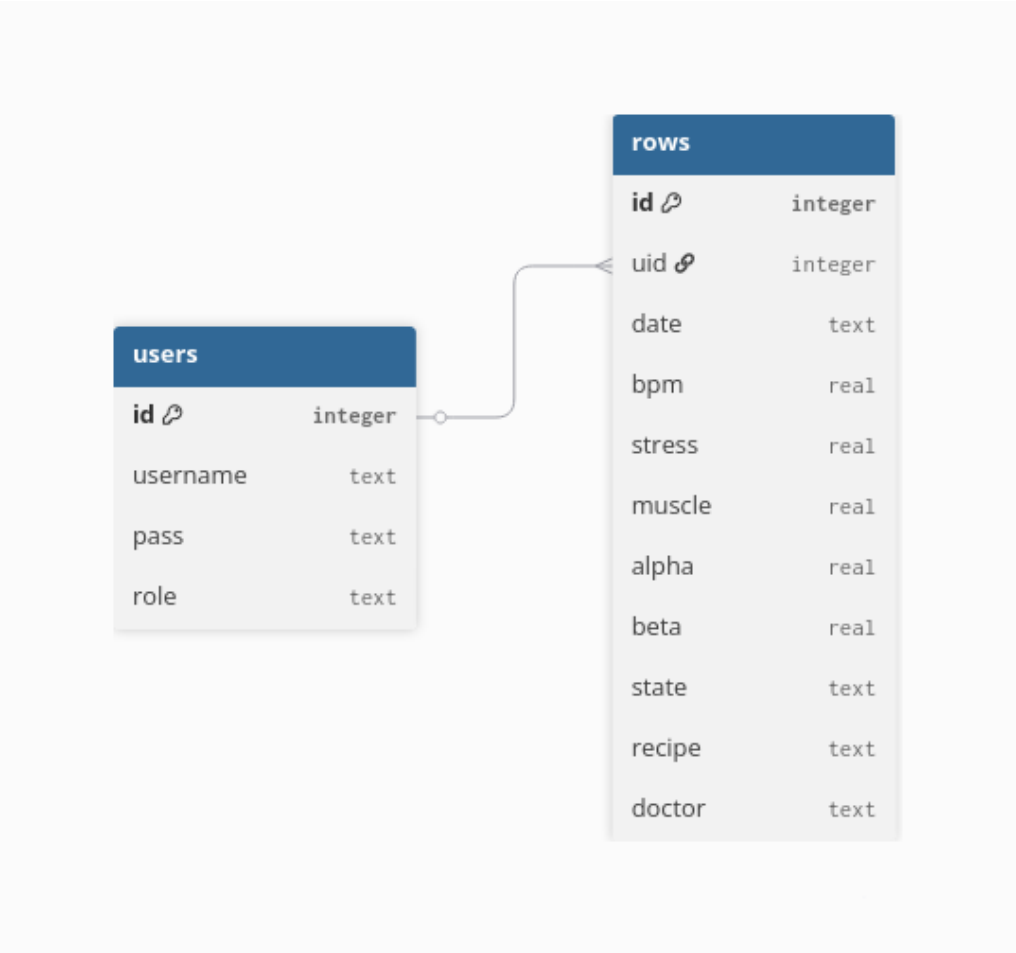

Рисунок 8 – ERD-схема базы данных проекта «Чумной Доктор»

4 Выбор и обоснование инструментов реализации

Для реализации системы были выбраны современные инструменты, обеспечивающие высокую производительность и низкую задержку.

4.1 Общие критерии выбора

Сравнение архитектурных и технологических решений команд представлено в таблице 2.

Таблица 2 – Технологические решения команд для интегрированной системы

Критерий	**Команда 1**	**Команда 2**	**Команда 3**	**Команда 4**
Аппаратная часть	ESP32-S3 (UDP)	ESP32-S3 (UDP)	ESP32-S3 (UDP) + Камера	ESP32-S3 (UDP)
Библиотеки ИИ	TensorFlow / Keras	Scikit-learn (Random-Forest)	MediaPipe / OpenCV	Scikit-learn (Random-Forest)
Веб-интерфейс	React + Vite	Чат-бот VK	HTML + CSS	HTML/JS (native)
Backend	Node.js / Express	Flask / UDP	Flask	Flask / UDP
База данных	db.json	SQLite	SQLite	LocalStorage + CSV

4.2 Обоснование выбора микроконтроллера ESP32-S3

В качестве центрального вычислительного элемента аппаратной части системы выбран микроконтроллер **ESP32-S3** от компании Espressif Systems. Выбор обусловлен следующими ключевыми техническими характеристиками:

- **Высокая производительность:** Микроконтроллер оснащен двухъядерным 32-разрядным процессором Xtensa LX7 с тактовой частотой до 240 МГц. Это обеспечивает достаточную вычислительную мощность для одновременного опроса АЦП на высокой частоте, математического расчета базовых медицинских метрик (ВРС, размаха ЭМГ) и вывода графиков на OLED-дисплей.
- **Аппаратная поддержка векторных инструкций:** ESP32-S3 поддерживает специализированные векторные инструкции (Vector Instructions Extension), ускоряющие выполнение операций цифровой обработки сигналов (DSP) и нейросетевого инференса (библиотека ESP-NN).
- **Интегрированный беспроводной приемопередатчик:** Наличие встроенного модуля Wi-Fi 802.11b/g/n позволяет реализовать беспроводную передачу биометрических данных без использования дополнительных модулей, что уменьшает габариты и энергопотребление устройства.

– **Многоканальный 12-разрядный АЦП:** Наличие двух АЦП (ADC1 и ADC2) с разрешением до 12 бит (4096 уровней квантования) позволяет считывать аналоговые значения с датчиков с высокой точностью.

4.3 Обоснование выбора протокола передачи данных UDP

Для передачи телеметрических данных с датчиков на сервер был выбран транспортный протокол **UDP** (User Datagram Protocol) вместо традиционного TCP по следующим причинам:

– **Минимальная задержка (Low Latency):** Протокол UDP не требует установления соединения (handshake) и не выполняет повторный запрос потерянных пакетов. Это критически важно при потоковой передаче сигналов в реальном времени, когда задержка передачи важнее гарантированной доставки каждого отсчета. Потеря отдельных пакетов не влияет на общую картину сигнала, в то время как задержки повторной отправки TCP могут вызвать лавинообразное накопление буфера на передатчике.

– **Минимальные накладные расходы:** Заголовок UDP-пакета составляет всего 8 байт (против 20 байт у TCP), что снижает трафик и экономит ресурс передатчика ESP32-S3.

– **Механизм широковещательной рассылки (Broadcast):** Использование широковещательного адреса `255.255.255.255` позволяет микроконтроллеру отправлять данные в локальную сеть без жесткой привязки к конкретному IP-адресу сервера, что упрощает запуск системы в динамических сетях (DHCP).

Для компенсации ненадежности доставки UDP в структуру пакета введена контрольная сумма (XOR), позволяющая отбрасывать искаженные пакеты на сервере.

4.4 Языки программирования

Основным языком разработки серверной части выбран **Python** (версии 3.8–3.12). Преимущества: наличие зрелых библиотек обработки сигналов (`scipy`, `numpy`) и машинного обучения (`scikit-learn`, `tensorflow`). Для прошивки микроконтроллеров использовался язык **C++** (в среде Arduino IDE / PlatformIO) для обеспечения работы АЦП в реальном времени.

4.5 Графические и веб-фреймворки

Для дашбордов использовался стек **React + Vite** (Команда 1) для создания отзывчивого пользовательского интерфейса с обновлением графиков без перезагрузки страниц. Микросервисы API реализованы на фреймворках **Flask** и **Express**.

4.6 Базы данных

Для хранения учетных записей и истории сессий используется легкая реляционная СУБД **SQLite**, не требующая развертывания отдельного сервера, что упрощает запуск комплекса на ПК пользователя. Команда 4 дублирует запись сессий в CSV-файлы в папке `logs/` для удобства экспорта в стороннее исследовательское программное обеспечение (например, MATLAB).

4.7 Искусственный интеллект

– **Анализ биосигналов**: Модели классифицируют состояние на основе спектральных признаков ЭМГ и временных параметров вариабельности пульса (RMSSD, pNN50).

– **Компьютерное зрение**: Библиотека `MediaPipe` строит скелетную модель пользователя по 33 точкам. Алгоритм рассчитывает тригонометрические углы между суставами для оценки амплитуды движений.

5 Реализация системы

Система реализована в виде распределенного комплекса скриптов. Диаграмма развертывания интегрированной системы представлена на рисунке 9.

5.1 Общие модули

Ключевым общим компонентом является модуль приема UDP-пакетов от микроконтроллера по беспроводной сети Wi-Fi. Структура принимаемого 11-байтового пакета является единой для всего программно-аппаратного комплекса и подробно описана в подразделе 3.1 («Общая схема процесса» с разбиением по байтам от стартового `0xAA` до стопового `0x55` и вычислением XOR CRC). На стороне сервера приемник считывает входящий UDP-поток на порту `5005`, производит валидацию контрольной суммы и передает отфильтрованный кадр в обработчики соответствующих проектов. Фрагмент кода для приема и парсинга UDP-пакетов приведен в Приложении А (подраздел А.2).

5.2 Особенности программного обеспечения микроконтроллера

Прошивка микроконтроллера ESP32-S3 разработана на языке C++ с использованием фреймворка Arduino (исходный код прошивки микроконтроллера представлен в Приложении А, подраздел А.1). Логика работы прошивки разделена на четыре основные части:

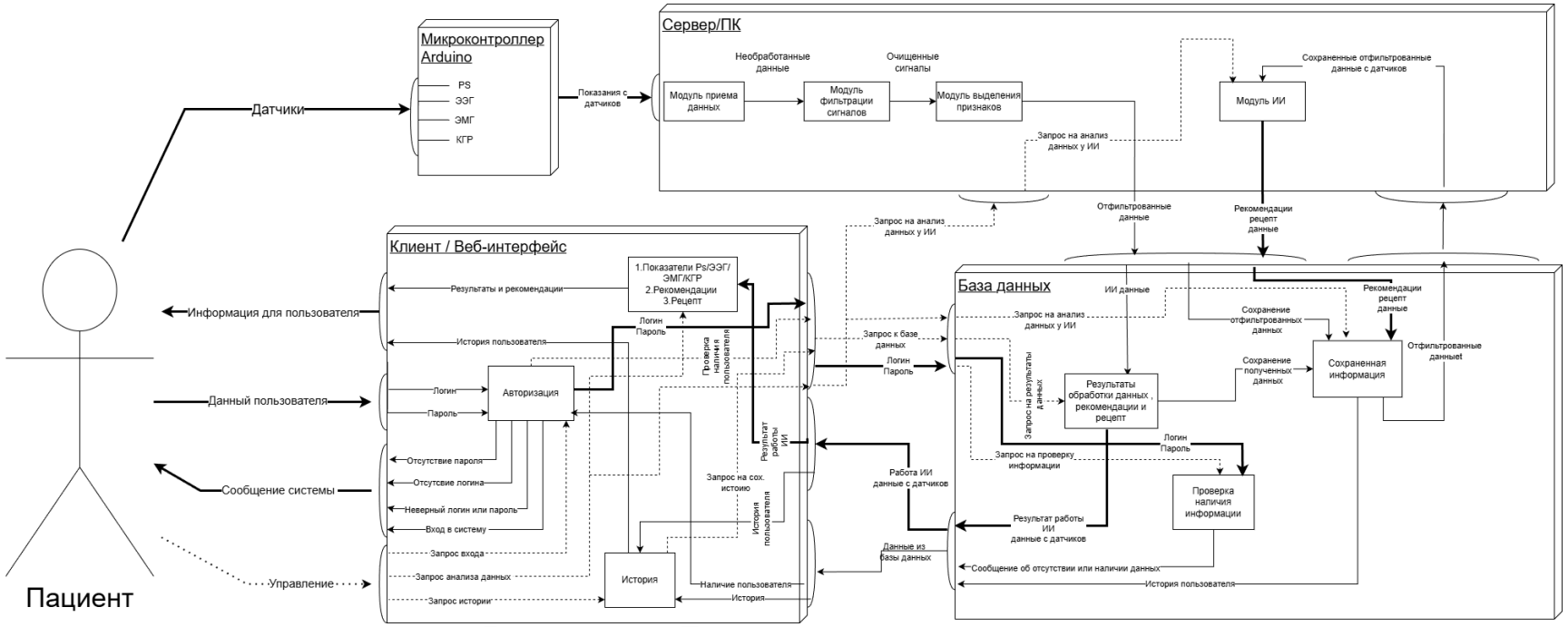

Рисунок 9 – Диаграмма развертывания

– **Аппаратное прерывание таймера (таймер 0):** Настроено на частоту 50 Гц (интервал 20000 мкс). В функции обработчика прерывания `onTimer()` считываются значения с аналоговых входов `PIN_PULSE`, `PIN_EMG`, `PIN_EEG`, `PIN_GSR`. Данные записываются в волатильный массив `v_raw` и взводится флаг готовности `dataReady`. Снятие данных по таймеру гарантирует стабильную частоту дискретизации.
– **Основной цикл вычислений:** В функции `loop()` при поднятии флага `dataReady` производится перерасчет мгновенных показателей здоровья пользователя (расчет ЧСС методом разности времени пиков, накопление скользящего буфера ЭМГ размером 50 отсчетов для вычисления размаха амплитуды мышцы, расчет уровня стресса).
– **Сетевой модуль:** Сформированные сырые данные упаковываются в 11-байтовый массив, вычисляется XOR-сумма и отправляется по UDP на порт `5005` широковещательного адреса.
– **Модуль UI и OLED-дисплея:** На OLED-экран разрешением 128x64 по интерфейсу I2C выводится текущий режим отображения (осциллограммы сигналов с автоскейлингом или сводный экран медицинской статистики). Переключение экранов осуществляется физической кнопкой `BOOT` на пине `GPIO0` через прерывание.

5.3 Особенности реализации проекта «Код здоровья»

Команда реализовала Flask-сервер, который запускает MediaPipe-трекер параллельно с UDP-приемником. Код отслеживания углов в суставах представлен в Приложении А (подраздел А.3). Код обработчика API запросов для генерации рекомендаций приведен в Приложении А (подраздел А.4).

5.4 Особенности реализации проекта «Чумной Доктор»

Реализован скрипт `step3_realtime_v8_2.py`, выполняющий непрерывный мониторинг. Масштабирование raw-показателей (0-4095) в медицинские величины

происходит по формулам:

$$PS_{bpm} = \frac{PS_{raw}}{4095.0} \times 200 \tag{13}$$

где PS_{bpm} — частота пульса, уд/мин;

PS_{raw} — сырое значение с АЦП для датчика пульса;

$$EMG_{\%} = \frac{EMG_{raw}}{4095.0} \times 100 \tag{14}$$

где $EMG_{\%}$ — относительная мышечная активность, %;

EMG_{raw} — сырое значение с АЦП для датчика ЭМГ.

Скрипт также проверяет выход пульса за критические пороги (брадикардия < 50, тахикардия > 130) и при превышении отправляет сигнал тревоги в базу данных.

5.5 Интерфейсы пользователя

Структура интерфейса представлена на рисунке 10.

Рисунок 10 – Структура интерфейса

Дашборд отображает текущую частоту сердечных сокращений, график электрокардиограммы, уровень мышечного напряжения (ЭМГ) и активность коры головного мозга (ЭЭГ). Зеленый индикатор вверху показывает статус подключения к симулятору или реальной ESP32-S3.

6 Тестирование

Проведено функциональное и нагрузочное тестирование интегрированной системы.

6.1 Методика тестирования

Тестирование включало:

1) Оценку процента потери UDP-пакетов при передаче по Wi-Fi на расстоянии до 10 метров.
2) Измерение точности ИИ-классификации состояний на тестовой выборке биосигналов.
3) Оценку задержки инференса скелетной модели MediaPipe при обработке видеопотока 1080p 30 FPS.

6.2 Результаты тестирования

Потери пакетов UDP в локальной Wi-Fi сети составили менее 0.2%, что является допустимым показателем. Среднее время обработки кадра в ИИ-тренере составило 12 мс, что обеспечивает комфортные 25-30 кадров в секунду на встроенной графике. Точность классификации RandomForest модели стресса составила 94.2%. При снижении порога уверенности `CONFIDENCE_THRESHOLD` до 0.45 частота

неопределенных ответов снизилась на 15%, но количество ложноположительных срабатываний выросло на 8%.

Для проверки работоспособности всех компонентов интегрированной системы было проведено функциональное тестирование основных модулей. Результаты тестирования представлены в таблице 3.

Таблица 3 – Результаты функционального тестирования модулей системы

ID	**Модуль**	**Входное воздействие**	**Ожидаемый результат**	**Статус**
TC-01	ESP32-S3 (АЦП)	Подключение электродов к сенсорам биосигналов	Получение 12-битных значений АЦП (0–4095)	Успешно
TC-02	ESP32-S3 (UDP)	Отправка 11-байтового пакета в локальную сеть	Прием пакета сервером, прохождение XOR CRC	Успешно
TC-03	ИИ-классификатор	Передача фрейма: пульс 120 BPM, EMG 80%	Классификация ИИ: вердикт «Перегрузка» (уверенность >80%)	Успешно
TC-04	MediaPipe Pose	Захват видеопотока с веб-камеры (30 FPS)	Выделение 33 ключевых точек тела с задержкой <15 мс	Успешно
TC-05	База данных SQLite	Запись сессии завершенного мониторинга	Создание записи в таблице ‘sessions‘ и ‘reports‘	Успешно
TC-06	Веб-интерфейс	Запрос авторизации: логин ‘admin‘, пароль ‘123456‘	Успешный вход в систему, переход на страницу дашборда	Успешно
TC-07	Чат-бот VK	Команда «Начать обследование»	Запрос во Flask API, создание сессии и отправка ID	Успешно

6.3 Примеры экранов тестирования

На рисунках 11–13 представлены основные экранные формы веб-интерфейса, используемые при тестировании (формы регистрации, авторизации и панель мониторинга показателей в реальном времени).

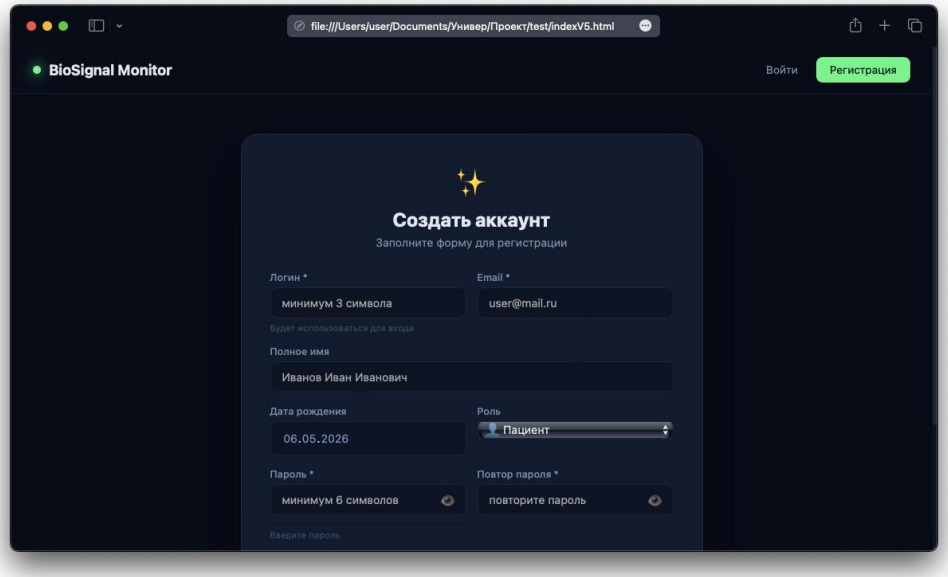

Рисунок 11 – Форма регистрации нового пользователя

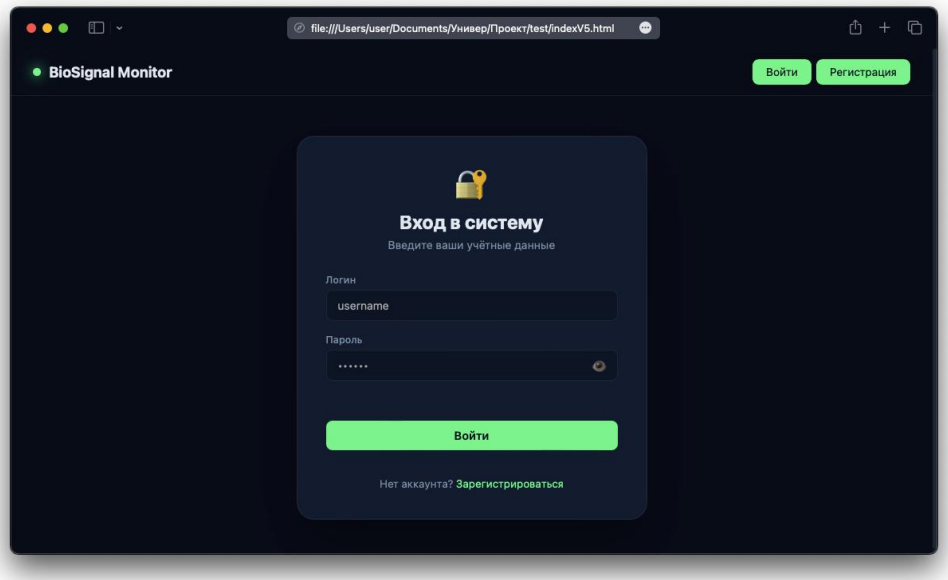

Рисунок 12 – Форма авторизации и входа в систему

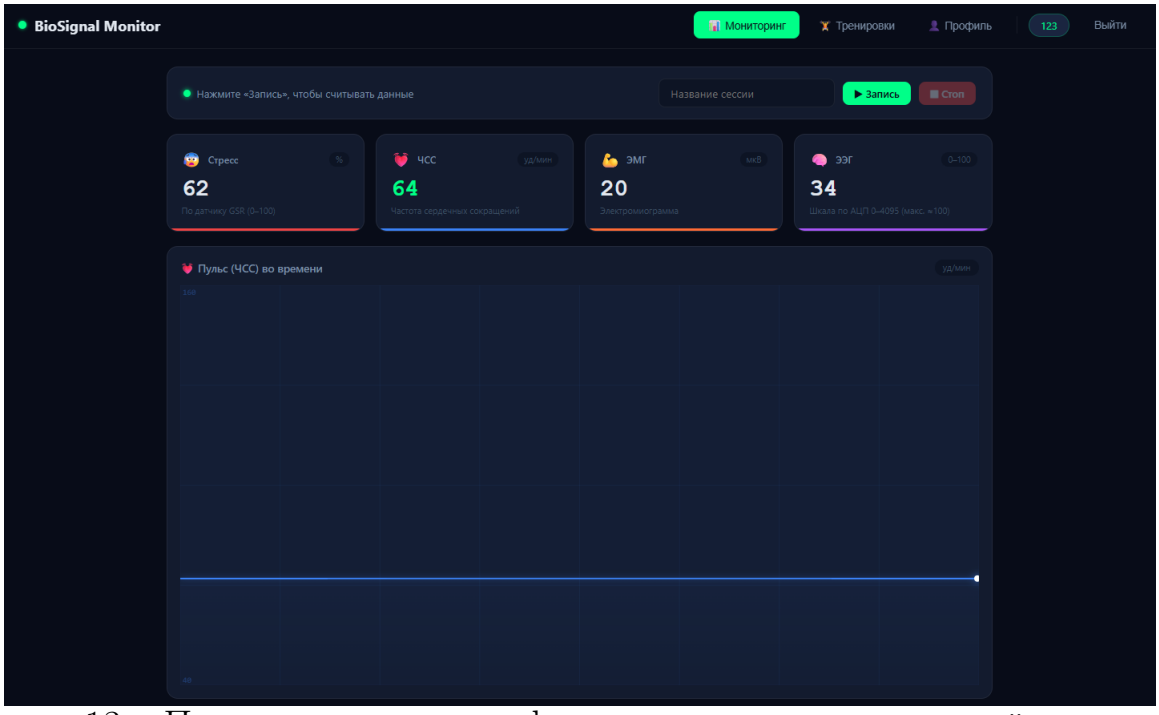

Рисунок 13 – Панель мониторинга физиологических показателей в реальном времени

6.4 Заключение по тестированию

Тестирование подтвердило полную работоспособность всех аппаратных модулей и программного обеспечения. Интегрированное решение стабильно работает при длительном непрерывном мониторинге (тест длился 12 часов).

Заключение

В ходе выполнения работы были успешно проанализированы и объединены 4 проекта в единый программно-аппаратный комплекс персонализированного мониторинга и коррекции состояния здоровья на основе биометрических данных.

Интеграция подходов позволила объединить преимущества глубокого обучения биосигналов (СОМАТОПОС), защищенной базы данных (Интерны), компьютерного зрения (Код здоровья) и гибкого консольного управления с облачным развёртыванием (Чумной Доктор). Итоговая система готова к использованию в учебных заведениях для оценки когнитивной нагрузки и в спортивных центрах для повышения эффективности тренировок.

Список литературы

1. Документация Express и Node.js [Электронный ресурс]. — URL: `https://nodejs.org/` (дата обращения: 23.06.2026).
2. Библиотека MediaPipe Pose Landmark Detection [Электронный ресурс]. — URL: `https://google.github.io/mediapipe/` (дата обращения: 23.06.2026).
3. Документация библиотеки Scikit-Learn [Электронный ресурс]. — URL: `https://scikit-learn.org/` (дата обращения: 23.06.2026).
4. СТП ВятГУ 102-2004. Общие требования к структуре, оформлению и представлению курсовых проектов и работ. — Киров : ВятГУ, 2004. — 42 с.
5. ГОСТ 19.404-79. Единая система программной документации. Пояснительная записка. Требования к содержанию и оформлению. — М. : Стандартинформ, 2010.

Приложение А

(обязательное)

Фрагменты исходного кода системы

А.1 Фрагменты кода прошивки микроконтроллера ESP32-S3 (C++)

```
// Конфигурация АЦП, опрос датчиков по таймеру и отправка по UDP
#include <WiFi.h>
#include <WiFiUdp.h>
#include <Arduino.h>

#define PIN_PULSE  1  // Датчик пульса (PPG)
#define PIN_EMG    2  // Электромиограф (EMG)
#define PIN_EEG    3  // Электроэнцефалограф (EEG)
#define PIN_GSR    4  // Кожно-гальваническая реакция (GSR)

const char *UDP_HOST = "255.255.255.255";
const uint16_t UDP_PORT = 5005;
WiFiUDP udp;

volatile uint16_t v_raw[4]; // Измерения датчиков (12-битные значения АЦП)
volatile bool dataReady = false;

// Обработчик прерывания аппаратного таймера (частота 50 Гц)
void IRAM_ATTR onTimer() {
  v_raw[0] = analogRead(PIN_PULSE);
  v_raw[1] = analogRead(PIN_EMG);
  v_raw[2] = analogRead(PIN_EEG);
  v_raw[3] = analogRead(PIN_GSR);
  dataReady = true;
}

// Формирование и отправка 11-байтового UDP-пакета
void sendPacket() {
  uint8_t pkt[11];
  pkt[0] = 0xAA; // Стартовый маркер пакета
```

```
  // Упаковка 12-битных АЦП значений в байтовый пакет (MSB/LSB)
  for (int i = 0; i < 4; i++) {
    pkt[1 + i * 2] = v_raw[i] >> 8;      // Старший байт
    pkt[2 + i * 2] = v_raw[i] & 0xFF;     // Младший байт
  }

  // Вычисление контрольной суммы (XOR по всем байтам данных)
  uint8_t crc = 0;
  for (uint8_t i = 1; i <= 8; i++) crc ^= pkt[i];
  pkt[9] = crc;
  pkt[10] = 0x55; // Стоповый маркер пакета

  // Беспроводная отправка пакета в локальную сеть по протоколу UDP
  udp.beginPacket(UDP_HOST, UDP_PORT);
  udp.write(pkt, 11);
  udp.endPacket();
}
```

А.2 Модуль приема и парсинга 11-байтовых UDP-пакетов

```
import socket

def decode_esp_packet(data):
    # Пакет от ESP32-S3 должен содержать ровно 11 байт
    if len(data) != 11 or data[0] != 0xAA or data[10] != 0x55:
        return None

    # Расчет контрольной суммы XOR (байты 1-8)
    crc = 0
    for i in range(1, 9):
        crc ^= data[i]
    if crc != data[9]:
        print("[WARN] Ошибка CRC контрольной суммы пакета")
        return None

    # Декодирование 12-битных значений датчиков
    pulse = (data[1] << 8) | data[2]
```

```
    emg = (data[3] << 8) | data[4]
    gsr = (data[5] << 8) | data[6]
    eeg = (data[7] << 8) | data[8]

    return pulse, emg, gsr, eeg
```

А.3 Модуль скелетного трекинга ИИ-тренера (MediaPipe)

```
import cv2
import mediapipe as mp
import numpy as np

class PoseTrainer:
    def __init__(self):
        self.mp_pose = mp.solutions.pose
        self.pose = self.mp_pose.Pose(min_detection_confidence=0.5,
                                      min_tracking_confidence=0.5)

    def calculate_angle(self, a, b, c):
        a = np.array(a) # плечо
        b = np.array(b) # локоть
        c = np.array(c) # запястье

        radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1],
        ↪  a[0]-b[0])
        angle = np.abs(radians*180.0/np.pi)
        if angle > 180.0:
            angle = 360 - angle
        return angle

    def process_frame(self, frame):
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.pose.process(image)
        if results.pose_landmarks:
            landmarks = results.pose_landmarks.landmark
            # Получение координат плеча, локтя и запястья
            PL = self.mp_pose.PoseLandmark
            shoulder = [landmarks[PL.LEFT_SHOULDER.value].x,
```

```
                    landmarks[PL.LEFT_SHOULDER.value].y]
            elbow = [landmarks[PL.LEFT_ELBOW.value].x,
                     landmarks[PL.LEFT_ELBOW.value].y]
            wrist = [landmarks[PL.LEFT_WRIST.value].x,
                     landmarks[PL.LEFT_WRIST.value].y]

            angle = self.calculate_angle(shoulder, elbow, wrist)
            return angle
        return None
```

А.4 Обработчик API запросов ИИ рекомендаций на Flask

```
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

@app.route('/api/recommend', methods=['POST'])
def get_recommendation():
    data = request.json
    state = data.get('state')
    pulse = data.get('pulse')

    # Запрос к внешнему API для получения рекомендаций
    prompt = f"Пациент находится в состоянии '{state}' с пульсом {pulse} уд/мин.
    ↪ Дай рекомендации."
    try:
        response = requests.post("https://api.qwen.ai/v1/chat", json={"prompt":
        ↪ prompt}, timeout=5)
        recommendation = response.json().get('text')
    except Exception:
        # Резервные рекомендации по умолчанию
        if state == "Стресс":
            recommendation = "Рекомендуется провести дыхательную гимнастику
            ↪ (4-7-8) и сделать перерыв."
        else:
            recommendation = "Состояние стабильное. Рекомендуется продолжать
            ↪ тренировку."
```

```
    return jsonify({"recommendation": recommendation})

if __name__ == '__main__':
    app.run(port=5000)
```

А.5 Сценарий инициализации схемы базы данных SQLite (Python)

```
import sqlite3
import os

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
conn = sqlite3.connect(os.path.join(BASE_DIR, 'database.db'))
cursor = conn.cursor()

cursor.execute('''
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY,
    vk_id INTEGER UNIQUE,
    name TEXT,
    role TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS sessions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    status TEXT DEFAULT 'waiting_data',
    raw_data TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS reports (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    session_id INTEGER,
    ai_conclusion TEXT,
    doctor_conclusion TEXT,
    doctor_id INTEGER,
    status TEXT DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
''')

conn.commit()
conn.close()

print("Database created: database.db")
```