

**«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)**

Направление	09.03.01 – Информатика и вычислительная техника
Профиль	Организация и программирование интеллектуальных систем
Факультет	КТИ
Кафедра	ВТ
<i>К защите допустить</i>	
И.о. зав. каф. ВТ д.т.н. проф.	Ю.А. Шичкина

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

**Тема: ОТСЛЕЖИВАНИЕ КЛЮЧЕВЫХ ХАРАКТЕРИСТИК ДЛЯ
УСТОЙЧИВОЙ РАБОТЫ УЗВ БЕЗ СОТРУДНИКОВ НА БАЗЕ
МИКРОКОНТРОЛЛЕРА И СИСТЕМЫ ДАТЧИКОВ**

Студент		<hr/>	М. Д. Семенов
		<i>подпись</i>	
Руководитель	д.т.н. проф.	<hr/>	А. И. Водяхо
		<i>подпись</i>	
Консультанты		<hr/>	О. Ю. Потерпеев
		<i>подпись</i>	
	д.э.н., проф.	<hr/>	О.А. Цуканова
		<i>подпись</i>	
		<hr/>	М.Н. Гречухин
		<i>подпись</i>	

Санкт-Петербург
2025

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

И.о заведующего кафедрой ВТ д.т.н. проф.

_____ Ю. А. Шичкина

«__» _____ 2025 г.

Студент М. Д. Семенов

Группа 1308

Тема работы: Отслеживание ключевых характеристик для устойчивой работы УЗВ без сотрудников на базе микроконтроллера и системы датчиков.

Место выполнения ВКР: ООО "УК Митридат"

Технические требования: создать программу для устойчивой работы УЗВ без сотрудников.

Содержание ВКР: Анализ предметной области, проектирование и создание системы контроля УЗВ. Тестирование полученного устройства.

Перечень отчетных данных: пояснительная записка, иллюстративный материал в виде презентации MS PowerPoint.

Дополнительный раздел: Оценка экономической эффективности инновационного проекта

Дата выдачи задания

Дата представления ВКР к защите

«__» _____ 20__ г.

«__» _____ 20__ г.

Студент

М. Д. Семенов

Руководитель д.т.н., проф.

А. И. Водяхо

Консультант

О. Ю. Потерпеев

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

И.о. заведующего. кафедрой ВТ д.т.н. проф.

_____ Ю.А. Шичкина

« ____ » _____ 20 ____ г.

Студент М. Д. Семенов

Группа 1308

Тема работы: Отслеживание ключевых характеристик для устойчивой работы УЗВ без сотрудников на базе микроконтроллера и системы датчиков.

№ п/п	Наименование работ	Срок выполнения
1	Обзор материалов по теме работы	01.04 – 07.04
2	Исследования возможностей программного комплекса	8.04 – 12.04
3	Проектирование устройства, подключение компонент и программирование устройства.	13.04 – 20.05
4	Оформление дополнительного раздела	20.05 – 30.05
5	Оформление пояснительной записки	31.05 – 09.06
6	Оформление иллюстративного материала	10.06 – 17.06
7	Предварительное рассмотрение работы	18.06

Студент

М. Д. Семенов

Руководитель д.т.н., проф

А. И. Водяхо

Консультант

О. Ю. Потерпеев

РЕФЕРАТ

Пояснительная записка содержит: 86 страниц, 64 рисунка, 14 таблиц, 12 источников.

Объектом исследования является система контроля УЗВ на базе микроконтроллера и системы датчиков.

Цель работы – разработка и исследование системы автоматического контроля за параметрами УЗВ, обеспечивающей устойчивую работу без постоянного контроля персонала.

Выпускная квалификационная работа включает в себя пять разделов, включая дополнительный раздел, а также список использованных источников.

В первом разделе реализован анализ требований к системе, обоснование оптимальной архитектуры и технический аспект разрабатываемой системы.

Второй раздел включает в себя проектирование системы и подбор необходимых компонент для дальнейшего подключения устройства.

В третьем разделе приведена практическая реализация проекта, которая включает реализацию подключения компонент к системе и алгоритмы работы устройства.

Четвёртый раздел посвящён результатам проектирования системы, в разделе показаны шаги работы устройства и визуализирован результат.

Дополнительный раздел рассматривает экономическую эффективность реализации готового устройства системы контроля УЗВ.

ABSTRACT

The explanatory note contains: 86 pages, 64 figures, 14 tables, 12 sources.

The object of the study is an ultrasound monitoring system based on a microcontroller and a sensor system.

The purpose of the work is to develop and study a system for automatic monitoring of ultrasonic parameters, which ensures stable operation without constant monitoring of personnel.

The final qualifying work includes five sections, including an additional section, as well as a list of sources used.

The first section provides an analysis of the system requirements, justification of the optimal architecture and the technical aspect of the system being developed.

The second section includes designing the system and selecting the necessary components for further connection of the device.

The third section provides a practical implementation of the project, which includes the implementation of connecting components to the system and algorithms for the operation of the device.

The fourth section is devoted to the results of the system design, the section shows the steps of the device and visualization.

An additional section examines the cost-effectiveness of the implementation of a ready-made ultrasound control system device.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	8
ВВЕДЕНИЕ	10
1 Анализ предметной области и методов реализации	12
1.1 Требования	12
1.1.1 Бизнес-требования	12
1.1.2 Пользовательские требования	12
1.1.3 Системные требования	12
1.1.4 Функциональные требования	12
1.1.5 Нефункциональные требования	13
1.1.6 Требования к продукту	13
1.2 Обоснования выбора типа архитектуры разрабатываемого приложения	13
1.3 Анализ существующих решений	15
1.4 Обоснование выбора средств реализации проекта	15
2 Проектирование системы контроля УЗВ	17
2.1 Описание установки и выбор компонент	17
2.1.1 Модель УЗВ	17
2.1.2 Принцип работы УЗВ	18
2.1.3 Установка и подбор датчиков	19
2.2 Описание устройства	22
2.2.1 Общая схема принципа работы	22
2.2.2 Исследование и анализ микроконтроллеров	24
2.2.2 Выбор и характеристики микроконтроллера	30
2.2.3 Выбор камеры видеонаблюдения, дисплеев и sd-card адаптера	32
3 Разработка системы контроля УЗВ	35
3.1 Реализация подключения устройств	35
3.1.1 Интеграция датчиков в схему	35
3.1.2 Работа с sd-картой, дисплеем и камерой	41
3.1.3 Подключение ESP32-CAM	45
3.1.4 Общая схема подключённых устройств	46
3.2 Конфигурация среды разработки	47
3.3 Описание структуры для хранения данных	52
3.4 Работа с Wi-Fi и протоколом smtp	52

3.5 Сохранение и обработка полученных данных	55
3.6 Обработка и визуализация данных	57
3.7 Фрагменты функций	59
4 Тесты.....	67
5 Обоснование экономической эффективности инновационного проекта.....	73
5.1 Расчёт сметы затрат на производство	73
5.1.1 Расчёт затрат на основные и вспомогательные материалы, покупные изделия и полуфабрикаты	73
5.1.2 Расчёт затрат на топливо и энергию для технологических целей.....	74
5.1.3 Расчёт затрат на заработную плату персонала	75
5.1.4 Страховые взносы во внебюджетные фонды	77
5.1.5 Расчёт амортизационных отчислений	77
5.1.6 Расчёт затрат на контрагентские работы, командировки и прочие затраты	78
5.1.7 Расчёт накладных расходов.....	78
5.2 Определение аналогов объекта разработки, определение товарного типа товара-разработки	80
5.3 Определение цены объекта разработки. Анализ безубыточности производства. Определение экономической эффективности статистическими методами	81
5.3.1 Метод расчёта цены учётом пользовательского эффекта	81
5.3.2 Метод анализа коридора цен.....	82
5.3.3 Анализ безубыточности производства.....	82
5.3.4 Определение экономической эффективности продукта статическими методами	83
ЗАКЛЮЧЕНИЕ.....	85
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	86

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В данной пояснительной записке применяются следующие термины с соответствующими определениями:

ESP32 – микроконтроллер, используемый в проекте;

ESP32-CAM – компактный модуль на основе микроконтроллера ESP32 с встроенной камерой OV2640. Предназначен для создания изображений;

ESP-IDF – это официальная среда разработки от Espressif для программирования микроконтроллеров ESP32, ESP32-S2/S3/C3 и других.

IoT – интернет вещей, подключение устройств к сети для обмена данными.

OVP – окислительно-восстановительный потенциал воды;

RS485 – это стандарт последовательной передачи данных, широко используемый в промышленной автоматизации, системах управления и других сферах, где требуется надежная связь на большие расстояния;

SD-карта – устройство для хранения данных, в проекте используется для хранения файлов json и изображений с камеры;

SMTP – стандартный протокол для отправки электронной почты через интернет. Он определяет правила передачи писем между почтовыми серверами и клиентами.

SQL СУБД – реляционные СУБД, где данные хранятся в таблицах с чёткой структурой;

TDS – общее количество растворённых твёрдых веществ в воде;

TFT-дисплей – компактный цветной экран, в проекте предназначен для вывода информации и параметров датчиков на экран;

TTL-RS485 переходник – устройство, которое преобразует сигнал UART в сигнал RS485;

UART – последовательный интерфейс для асинхронной передачи данных между устройствами;

АСК – автоматизированная система контроля;

БД – база данных – система, для хранения управления и быстрого доступа к большим объёмам структурированной информации;

Датчик – устройство, которое преобразует физическую величину в электрический сигнал, который можно обработать микроконтроллером;

Интерфейс – способ обмена данными между устройствами, программами или пользователем и системой;

МК – микроконтроллер – компактная вычислительная система на одном кристалле, предназначенная для управления электронными устройствами. Объединяет процессор, память и периферийные модули, что позволяет ему работать автономно;

ПЛК ОВЕН – это линейка программируемых логических контроллеров (ПЛК) от российской компании "ОВЕН", специализирующейся на средствах автоматизации. Эти устройства широко используются в промышленности, ЖКХ, энергетике и других сферах для управления технологическими процессами;

СК – система контроля;

СУБД – система управления базами данных;

ТЗ – техническое задание;

УЗВ – устройство замкнутого водоснабжения;

ВВЕДЕНИЕ

В современном мире большое внимание уделяется рыбным деликатесам. Многие компании строят специальные рыбные фермы для выращивания и разведения гидробионтов. Для работы таких ферм используются современные аквакультурные технологии, такие как установки замкнутого водоснабжения. Данные установки требуют постоянного контроля различных параметров, которые в совокупности определяют качество воды. При помощи автоматизации мониторинга важных характеристик УЗВ можно обеспечить надёжную работу системы без постоянного наблюдения сотрудников

УЗВ – это технологическая система, в которой вода циркулирует по замкнутому контуру. Эта система предназначена для создания наилучших условий, чтобы выращивать рыбу в промышленных объёмах на рыбных фермах. В зависимости от предприятия, такая УЗВ может включать в себя различные технологические процессы, такие как: подача воды в бассейны, контроль уровня воды, механическая фильтрация, биологическая очистка, удаление нитратов, обеззараживание, регулирование температуры, возврат воды в бассейны.

Система контроля УЗВ – это набор аппаратно-программных средств, предназначенный для автоматизированного мониторинга и управления параметрами воды в УЗВ. Основная задача СК – контроль за соблюдением благоприятных условий для выращивания рыбы. СК УЗВ измеряет ключевые показатели воды, сохраняет полученные показатели для дальнейшего анализа и оповещает о критических ситуациях.

Система контроля состоит из аппаратной и программной частей. В аппаратную часть входят сами устройства, которые получают параметры: датчики, контроллеры, исполнительные устройства, устройства хранения и отображения параметров. Программная же часть состоит из мобильного приложения или БД, где хранится и анализируется история изменений параметров.

В отличие от ручного контроля воды, автоматизированная система обладает более высокой точностью, мгновенной реакцией на угрозы, минимальными трудозатратами при длительном использовании и минимальным риском, связанным с ошибками. Такие СК предназначены для оптимизации работы, в первую очередь, руководства, технологов и операторов, и являются залогом успешного повышения эффективности производительности рыбных ферм. Используя этот инструмент, сотрудникам становится значительно легче выполнять рутинные операции, связанные с получением и ведением данных о состоянии рыбной фермы, а смертность рыбы и затраты на электроэнергию снижаются на 15–20%.

Рассмотрев предметную область, можно сделать вывод о том, что разработка системы контроля УЗВ рыбной фермы является актуальной задачей и неотъемлемым условием конкурентоспособности в современных реалиях работы рыбных ферм.

1 Анализ предметной области и методов реализации

Для того, чтобы проанализировать предметную область, для начала рассмотрим требования, которым должна удовлетворять созданная система.

1.1 Требования

В качестве основных требований к системе будут бизнес-требования, пользовательские, системные, функциональные, нефункциональные, а также требования к продукту.

1.1.1 Бизнес-требования

- 1) уменьшение потерь рыбы из-за аварийных ситуаций,
- 2) повышение рентабельности за счёт автоматизации,
- 3) контроль расхода корма и электроэнергии.

1.1.2 Пользовательские требования

- 1) возможность видеть текущие параметры воды,
- 2) возможность анализировать полученные данные качества воды,
- 3) обеспечение настройки пороговых значений для срабатывания аварийных сигналов.

1.1.3 Системные требования

- 1) для реализации проекта должен быть использован язык C с платформой ПО для микроконтроллеров Espressif, язык python для занесения данных в БД, а база данных SQLite;
- 2) доступ к глобальной сети интернет.

1.1.4 Функциональные требования

- 1) контроль водных потоков в трубах,

- 2) контроль уровня воды в баках,
- 3) контроль параметров воды,
- 4) получение, хранение и визуализация характеристик системы.

1.1.5 Нефункциональные требования

- 1) частота опроса датчиков – не реже 1 раза в 60 минут,
- 2) поддержка подключения до 100 датчиков на 1 МК,
- 3) возможность увеличения количества бассейнов.

1.1.6 Требования к продукту

- 1) стабильная и быстрая работа приложения,
- 2) быстрая замена сломавшихся компонент или включение новых устройств в систему.

1.2 Обоснования выбора типа архитектуры разрабатываемого приложения

Перед тем как определить методы и технологии для реализации системы контроля, необходимо выбрать тип архитектуры разрабатываемого приложения. Систему контроля УЗВ рыбной фермы можно представить как многоуровневую архитектуру с событийно-ориентированным подходом.

Многоуровневая система состоит из следующих уровней: уровень сенсоров (датчики и микроконтроллер), уровень хранения (sd-карта), уровень передачи (Wi-Fi, SMTP), уровень сервера (парсинг писем и сохранение в БД) и представлена на рисунке 1.

К преимуществам такого подхода относятся разделение ответственности, легкость модификации и замены компонент, а также удобство тестирования.

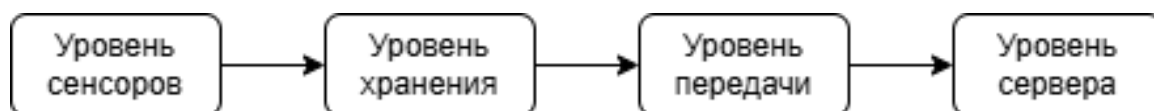


Рисунок 1 – Многоуровневая система

Событийно-ориентированный подход позволяет работать в режиме прерываний, например с помощью таймера или изменению данных с датчика. Данные могут записываться на уровень хранения и отправляться на сервер по событию (получение значений), по расписанию (раз в час/в день) или при накоплении N записей. Такой подход позволяет экономить энергию, так как микроконтроллер может “спать” между событиями.

У каждого типа архитектуры приложений есть свои преимущества и недостатки. Однако для реализации СК УЗВ рыбной фермы, многоуровневая архитектура представляет собой оптимальное решение по следующим причинам:

- минимизация зависимостей от клиентских устройств, так как данные собираются и обрабатываются на микроконтроллере, а не на клиентских ПК, а обновление ПО касается только микроконтроллера и серверной части,
- отсутствие проблемы совместимости версий, из-за того, что логика работы сосредоточена на стороне микроконтроллера и сервера;
- упрощённое администрирование, потому что нет необходимости в установке ПО на каждое приложение,
- доступность данных из любой точки мира, так как данные передаются по Wi-Fi/HTTP/SMTP,
- кросс-платформенность – микроконтроллер работает независимо от клиентских устройств, данные могут отображаться в браузере, мобильном приложении или API,
- низкие требования к клиентским устройствам в связи с тем, что основная нагрузка ложится на микроконтроллер и сервер, а не на ПК,

- Гибкость взаимодействия с внешними API – данные, полученные с микроконтроллера можно передавать в БД, облачные хранилища и системы аналитики.

1.3 Анализ существующих решений

СК УЗВ уже используются в аквакультуре. Есть крупные компании, которые занимаются реализацией таких СК. Например, компании AKVA Group или Pentair Aquaculture. Разрабатываемые данными компаниями АСК могут быть сделаны под конкретное ТЗ покупателя. Однако, данные системы реализуются на базе промышленных контроллеров, из-за чего обладают высокой стоимостью, которая начинается от 1000000 рублей. Многие УЗВ рыбных ферм являются небольшими, из-за чего такие расходы на создание УЗВ могут не окупиться и рыбная ферма обанкротится. Также эти компании обладают закрытым ПО, из-за чего в случае необходимости внесения изменений в систему, необходимо обращаться в компанию.

В связи с этими недостатками, было принято решение рассмотреть альтернативную реализацию, основанную на программировании микроконтроллеров, таких как Arduino, ESP32, ESP8266 и ПЛК ОВЕН.

Реализованные на микроконтроллерах установки, несмотря на необходимость программирования микроконтроллеров для промышленного использования, обладают рядом преимуществ, а именно: низкой стоимостью и гибкостью настройки, что является важными плюсами, так как при изменении УЗВ, скорректировать СК не составит больших проблем.

1.4 Обоснование выбора средств реализации проекта

Реализация программной части устройства может быть реализована на языке C++, MicroPython или же C. Для выбора языка проведём сравнение. Для СК УЗВ являются критичными:

- надёжность,

- производительность,
- доступ к железу.

Основываясь, на эти характеристики, выберем дальнейший язык программирования. MicroPython обладает ограниченным набором команд при программировании.

При выборе между C++ и C, остановимся на выборе языка C, так как C будет компилироваться в более компактный машинный код, что важно, так как МК обладают ограниченной Flash-памятью, в C проще работать с регистрами и памятью через указатели, а также большинство SDK изначально были написаны на C.

Для извлечения данных из письма и записи в БД будем использовать язык Python, который является оптимальным по ряду причин: простота разработки, наличие библиотек для работы с почтой и БД, поддержка JSON и современных форматов данных.

При выборе СУБД, остановимся на SQLite, так как данная СУБД не требует дополнительных настроек, обеспечивает удобную работу и высокую скорость операций.

2 Проектирование системы контроля УЗВ

Для проектирования системы контроля УЗВ необходимо сначала рассмотреть описание установки и выбрать компоненты.

2.1 Описание установки и выбор компонент

Рассмотрим реальную модель УЗВ рыбной фермы, для которой необходимо создать систему контроля.

2.1.1 Модель УЗВ

Система УЗВ состоит из одного маленького бассейна, двух больших бассейнов, основного и дополнительного механических фильтров, биофильтра, озонатора, сумматора и бочки долива.

Описание каждого компонента представлено в таблице 1, а полностью спроектированная схема УЗВ была создана в программе КОМПАС 3D и представлена на рисунке 2.

Таблица 1 – Описание установки УЗВ

№	Название компонента	Описание	Сокращение (Рисунок 2)
1	маленький бассейн	Используется для разведения мальков	МБ
2	большой бассейн 1	Основной резервуар для выращивания рыбы	ББ1
3	большой бассейн 2	Основной резервуар для выращивания рыбы	ББ2
4	Основной механический фильтр	Фильтр грубой очистки, основная задача – удаление крупных твёрдых частиц, таких как остатки корма, взвеси, экскрименты.	ОМФ
5	Дополнительный механический фильтр	Фильтр тонкой очистки. Удаляет более мелкие частицы, которые не были удалены в МФ1	ДМФ
6	Биофильтр	Устройство, которое обеспечивает биологическую очистку. Биофильтр преобразует аммиак в нитриты и нитраты при помощи бактерий	БФ
7	Озонатор	Служит для дезинфекции воды, уничтожает органические загрязнения, микробы	ОЗ

Продолжение таблицы 1

№	Название компонента	Описание	Сокращение (Рисунок 2)
8	Сумматор	Смешивает потоки только что поступившей из доливной бочки воды и воды из механических фильтров. Насыщает воду кислородом, нужен для дыхания рыбы.	СМ
9	Бочка долива	Компенсирует потери воды и поддерживает стабильный уровень	БКД
10	Основная магистраль	Подача воды в систему	ОМ

Отметим компоненты на рисунке.

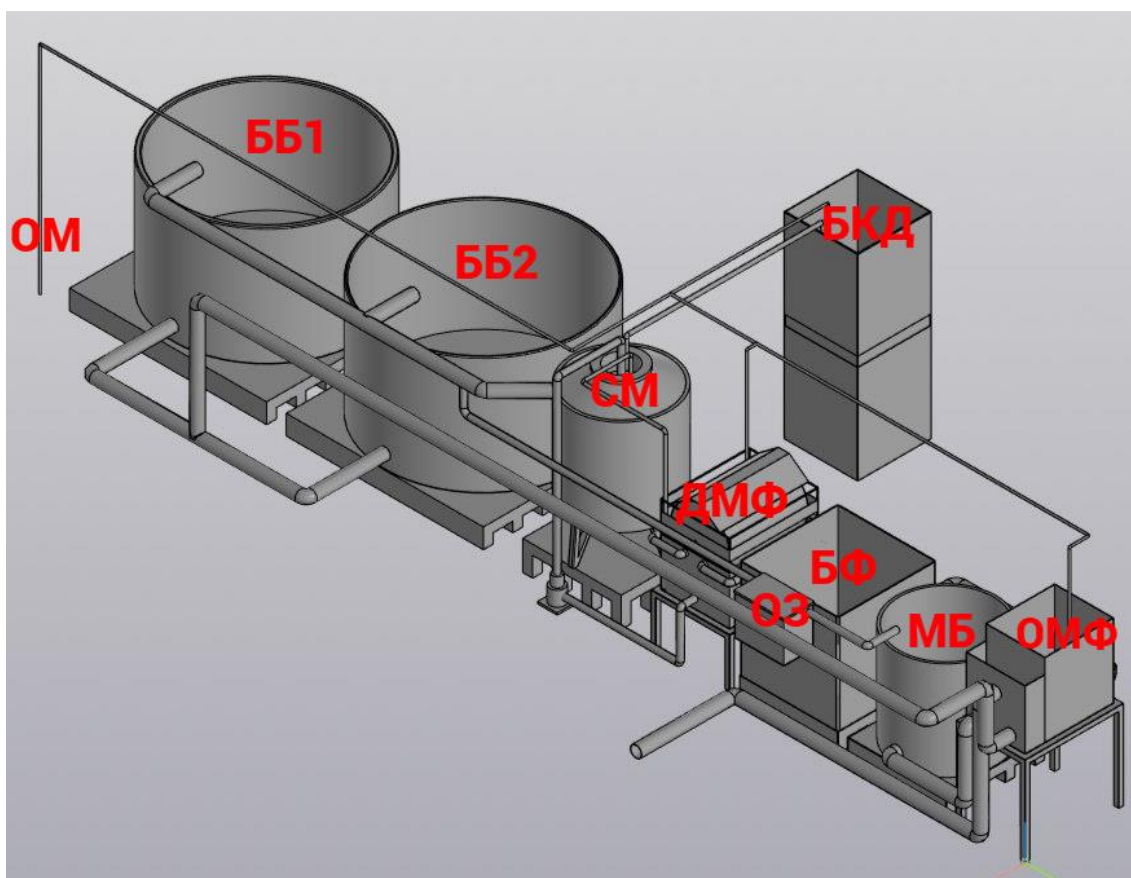


Рисунок 2 – Спроектированная схема УЗВ

2.1.2 Принцип работы УЗВ

Рассмотрим работу УЗВ рыбной фермы подробнее. Для этого проследим движение воды в системе.

Изначально вода подаётся в систему при помощи основной магистрали, откуда вода поступает в БКД, ОМФ и ДМФ. Из ОМФ вода поступает в БФ, откуда переливается в ДМФ. Из ДМФ потоки воды поступают в СМ, куда в

случае недостатка воды поступает вода из БКД. После вода поступает в ББ1, ББ2 и в ОЗ.С озонатора вода сливается в МБ. После этого вода заканчивается цикл обработки воды и потоки со всех баков поступают в ОМФ для следующей очистки воды.

Схематично принцип работы УЗВ представлен на рисунке 3.

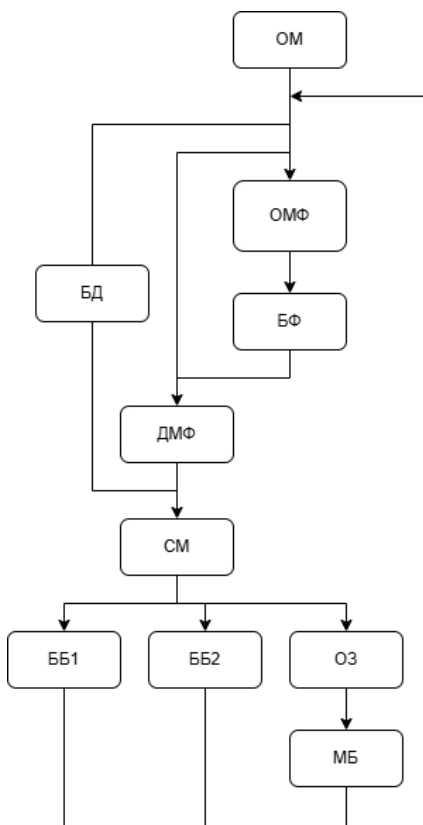


Рисунок 3 – Принцип работы УЗВ

2.1.3 Установка и подбор датчиков

Для автономной работы УЗВ диспетчер должен контролировать поток в трубах, иметь доступ к данным уровня воды в баках, а также получать параметрические значения о составе воды в баках, где находится рыба.

Важными характеристиками состава воды являются: температура, кислотность, мутность, количество растворённого в воде кислорода, окислительно-восстановительный потенциал и электропроводность.

Уровень воды в баках необходимо контролировать для того, чтобы не произошли переливы или бак не оказался пустым. Соответственно, в каждом

баке, где измеряется уровень, необходимы верхний и нижний датчики уровня воды или поплавковые выключатели.

Поток воды необходимо контролировать на критически важных магистралях, чтобы при необходимости регулировать объёмы циркуляции воды. Для этого будем использовать расходомеры. На магистралях, где не важен напор, но важно наличие потока, устанавливаются датчики проверки потока. Выбранные датчики для контроля перечисленных параметров в системе представлены в таблице 2.

Таблица 2 – Выбор датчиков

№	Тип датчика	Количество датчиков	Фото датчика представлено на рисунках	Обозначение
1	Латунный расходомер воды	5	4	FLL
2	Пластиковый расходомер воды	1	4	FLP
3	Латунный датчик потока	1	4	FLC
4	Поплавковый выключатель Aideepen PP Float Switch 52MM	6	4	WLA
5	Угловой датчик уровня воды Aideepen	1	4	WLB
6	Переключатель уровня жидкости для воды	1	4	WLR
7	Интеллектуальный ультразвуковой расходомер	1	4	IFL
8	Датчик кислотности	1	5	PHT
9	Датчик электропроводности	1	5	TDS
10	Датчик окислительно-восстановительного потенциала	1	5	OVP
11	Датчик мутности	3	5	MUT
12	Датчик кислорода	1	5	O2
13	Датчик уровня воды C1	8	4	FLN

Изображения датчиков, указанных в таблице 3 представлены на рисунке 4 и рисунке 5.



Рисунок 4 – Изображение непараметрических датчиков

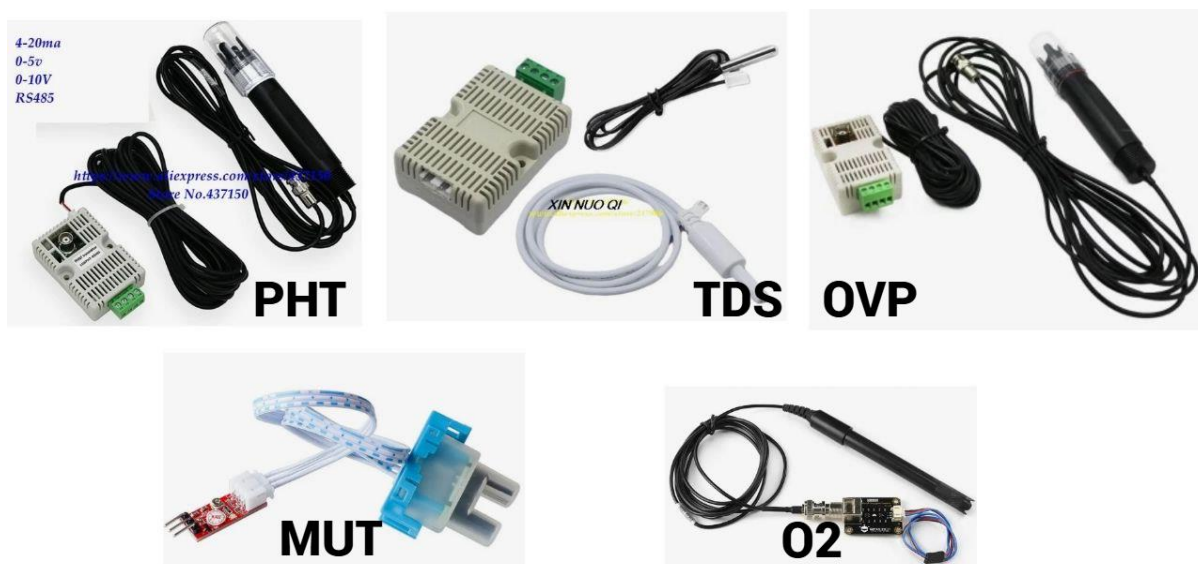


Рисунок 5 – Изображение параметрических датчиков

Для определения нужного количества датчиков воспользуемся спроектированной схемой рыбной фермы, на которой отметим места, где необходимо получать данные о параметрах системы. Спроектированная схема с отмеченными необходимыми датчиками представлена на рисунке 6.

Продолжение таблицы 3

№	Комплектуемый	Функция
3	TFT-дисплей	Визуализация данных с SD-карты
4	Дисплей диспетчера	Визуализация данных с SD-карты, управляется диспетчером
5	Камера	Визуальный контроль системы УЗВ рыбной фермы
6	SD-карта	Хранение показаний с датчиков и снимков с камеры.
7	Wi-Fi роутер	Wi-Fi для отправки данных с SD-карты, управляемой микроконтроллером, на ПК диспетчера
8	ПК диспетчера	Получение и анализ данных

Система получает данные с датчиков каждые 5 минут, которые управляются при помощи микроконтроллера, и сохраняет эти параметры на sd-карту, которая подключена к микроконтроллеру.

После чего данные с файла, расположенного на sd-карте отображаются на tft-дисплее и дисплее диспетчера, а также отправляются на ПК диспетчера по Wi-Fi раз в час.

Для визуального наблюдения за работой УЗВ рыбной фермы в систему включена камера, которая делает снимки и также сохраняет их на sd-карту и отправляет на ПК диспетчера.

Для наглядности изобразим схему проектируемого устройства на рисунке 7, указав только типы датчиков, и алгоритм работы устройства на рисунке 8.

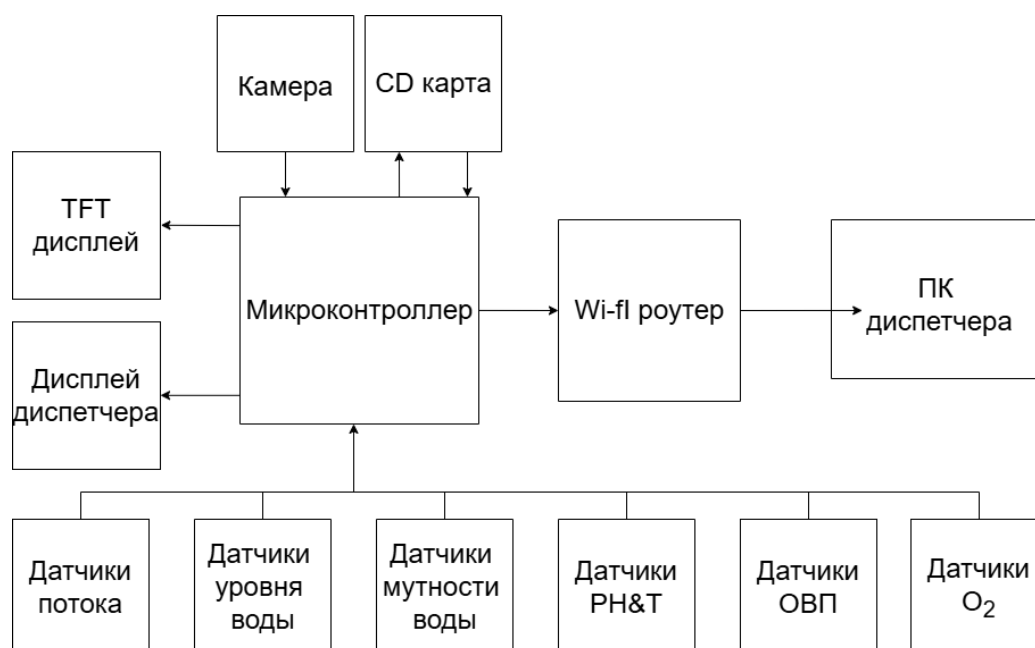


Рисунок 7 – Схема проектируемого устройства



Рисунок 8 – Алгоритм работы устройства

2.2.2 Исследование и анализ микроконтроллеров

Выберем микроконтроллер для дальнейшей работы. Распространёнными моделями являются платы Arduino, ESP8266, ESP32 и Овен ПЛК110.

Проведём сравнительный анализ плат Arduino, ESP8266, ESP32 и Овен ПЛК110 и выберем плату для дальнейшего использования:

Сравнение платы Arduino и ESP32

ESP32 — это недорогая плата для разработки с поддержкой Wi-Fi и Bluetooth. Эти платы играют важную роль в разработке проектов на основе Интернета вещей благодаря встроенной беспроводной технологии. Рабочее напряжение этой платы составляет от 2,2 до 6 В благодаря встроенному регулятору, обеспечивающему постоянное напряжение и выходной ток более 500 мА. Плата для разработки ESP32 оснащена двухъядерным процессором, который может работать независимо друг от друга, и 4 МБ флэш-памяти, что делает эту плату очень быстрой.

Эта плата разработки в основном предназначена для экономичных, энергоэффективных и простых приложений на базе Интернета вещей благодаря встроенному Wi-Fi и Bluetooth. Она построена на базе двухъядерного процессора Tensilica Xtensa с частотой 160 МГц, SRAM — 520 КБ, 34 вывода GPIO и т. д. Эту плату можно легко запрограммировать с помощью Arduino IDE, MicroPython, Lua, ESP-IDF, JavaScript и т. д. ESP32 работает в широком диапазоне температур от -40°C до 125°C.

Arduino

Плата Arduino — это плата для разработки с открытым исходным кодом, которая используется для создания встраиваемых систем, носимых устройств, электронных гаджетов, проектов в сфере Интернета вещей и робототехники.

Эти платы помогают разрабатывать различные электронные проекты, а также устройства, просто подключая различные датчики и двигатели. Плата Arduino включает 8-битный микроконтроллер ATmega328P с 0–13 цифровыми выводами. Эти контакты используются как для цифрового ввода, так и для цифрового вывода, где цифровой ввод предназначен для считывания данных с устройства, а цифровой вывод — для отправки данных с Arduino на устройство.

Рабочее напряжение этой платы составляет 5 В, а потребление тока — от 45 до 80 мА. В режиме глубокого сна плата потребляет 35 мА. Эту плату можно запитать от USB-разъёма, 9-вольтовой батареи, компьютера или блока

питания. Рабочая частота платы Arduino составляет 16 МГц, то есть каждую секунду выполняется 16 миллионов инструкций.

Сравнение ESP32 и Arduino представлено в таблице 4.

Таблица 4 – Сравнение ESP32 и Arduino

	ESP32	Arduino
Описание	ESP32 — это серия маломощных и недорогих SoC (систем на кристалле) с поддержкой Bluetooth и Wi-Fi в двух режимах.	Плата Arduino - это плата разработки с открытым исходным кодом.
Процессор	Tensilica Xtensa LX6	8-битный микроконтроллер ATmega328P
Рабочее напряжение	3,3 В	5 В
Оперативная память	520 КБ	2 КБ
Ethernet	встроенный порт Wi-Fi и Ethernet	подключение к Интернету Ethernet shield
Bluetooth	Есть	Нет
Используемое программное обеспечение	Python, micro python, C & C ++.	C / C ++
Операционные системы	Raspbian и Ubuntu	не использует никакой операционной системы
Рабочая частота	240 МГц	16 МГц
Флэш-память	16 МБ	Обычно 4 МБ, но может варьироваться
Контакты аналогового входа	18	6
Контакты GPIO	48	20 контактов GPIO, и 6 из них обеспечивают выходную мощность ШИМ
Типы плат	ESP32 DEV KIT DOIT, DevKitC, PICO, EYE, Thing, CAM, 32s	Arduino Uno, Mega, Nano, Leonardo
Сильные стороны	Высокая вычислительная мощность и возможности подключения	Простота и поддержка сообщества

Плата разработки ESP32 подходит для сложных проектов, требующих более высоких вычислительных возможностей и подключения к сети, таких как приложения для умного дома, промышленная автоматизация и устройства Интернета вещей. Простота платы Arduino позволяет использовать её в образовательных целях, для любительских проектов и художественных инсталляций.

Сравнение платы ESP8266 и ESP32

ESP8266 — это недорогой микрочип Wi-Fi со встроенным стеком TCP/IP и возможностями микроконтроллера, выпущенный в 2014 году компанией Espressif Systems в Шанхае, Китай.

ESP32, представленный в 2016 году, является преемником ESP8266 и предлагает несколько улучшений по сравнению с ним. К ним относятся более быстрый процессор, более быстрый Wi-Fi, Bluetooth-подключение, больше контактов GPIO и несколько других функций. Что еще более важно, он оснащен двухъядерным микропроцессором Tensilica Xtensa LX6, что значительно расширяет его вычислительные возможности.

Сравнение ESP32 и ESP8266 представлено в таблице 5.

Таблица 5 – Сравнение ESP32 и ESP8266

	ESP32	ESP8266
Описание	ESP32 — это серия маломощных и недорогих SoC (систем на кристалле) с поддержкой Bluetooth и Wi-Fi в двух режимах	ESP8266 — более старый, но популярный микроконтроллер с поддержкой Wi-Fi. Часто используется для простых IoT-проектов
Процессор	Tensilica Xtensa LX6	Tensilica Xtensa L106
Рабочее напряжение	3,3 В	3,3 В
Оперативная память	520 КБ	160 КБ
Wi-Fi	встроенный порт Wi-Fi и Ethernet	Поддержка Wi-Fi
Bluetooth	Есть	Не поддерживает
Используемое программное обеспечение	Python, micro python, C & C++	Python, micro python, C & C++
Операционные системы	Raspbian и Ubuntu	Чаще используется без ОС
Рабочая частота	240 МГц	160 МГц
Флэш-память	16 МБ	4 МБ
Контакты аналогового входа	18	1
Контакты GPIO	36	17
Типы плат	ESP32 DEV KIT DOIT, DevKitC, PICO, EYE, Thing, CAM, 32s	NodeMCU, Wemos D1 Mini, ESP-01, ESP-12E

Продолжение таблицы 5

	ESP32	ESP8266
Сильные стороны	Высокая вычислительная мощность и возможности подключения	Низкая стоимость. Простота в использовании для базовых IoT-проектов. Хорошая поддержка сообществом и документация
Цена	От 300 до 1500 рублей в зависимости от модификации и комплектации	От 200 до 800 рублей

ESP32 лучше подходит для сложных проектов, где требуется высокая производительность, поддержка Bluetooth и больше ресурсов.

ESP8266 – это бюджетное решение для простых IoT-проектов с минимальными требованиями.

Сравнение платы ESP32 и Овен ПЛК110

Овен ПЛК110 – промышленный программируемый логический контроллер, предназначенный для автоматизации производственных процессов. Отличается высокой надежностью и специализированным функционалом.

Сравнение ESP32 и Овен ПЛК110 представлено в таблице 6.

Таблица 6 – Сравнение ESP32 и Овен ПЛК110

	ESP32	Овен ПЛК110
Описание	ESP32 — это серия маломощных и недорогих SoC (систем на кристалле) с поддержкой Bluetooth и Wi-Fi в двух режимах	Однокристальный микроконтроллер (например, на базе ARM Cortex-M), оптимизированный для промышленных задач
Процессор	Tensilica Xtensa LX6	Однокристальный микроконтроллер (например, на базе ARM Cortex-M), оптимизированный для промышленных задач
Рабочее напряжение	3,3 В	24 В
Оперативная память	520 КБ	от 64 КБ до нескольких МБ
Wi-Fi	встроенный порт Wi-Fi и Ethernet	Отсутствует в базовой комплектации

Продолжение таблицы 6

	ESP32	Овен ПЛК110
Bluetooth	Есть	Отсутствует в базовой комплектации
Используемое программное обеспечение	Python, micro python, C & C ++	Программируется на языках МЭК 61131-3 (LD, FBD, ST, IL, SFC) в среде CODESYS или специализированном ПО Овен
Операционные системы	Raspbian и Ubuntu	Специализированная ОС для ПЛК
Рабочая частота	240 МГц	от 72 МГц до 200 МГц
Флэш-память	16 МБ	От 1 МБ до 16 МБ
Контакты аналогового входа	18	Обычно 4-8 аналоговых входов (16-битный АЦП, высокая точность)
Контакты GPIO	36	Обычно 8-16 цифровых входов/выходов
Типы плат	ESP32 DEV KIT DOIT, DevKitC, PICO, EYE, Thing, CAM, 32s	Моноблочные или модульные ПЛК (например, ПЛК110, ПЛК63)
Сильные стороны	Высокая вычислительная мощность и возможности подключения Низкая стоимость Высокая производительность для IoT-устройств	Высокая надежность и устойчивость к промышленным условиям Интеграция с промышленными сетями Простота программирования для инженеров-автоматизаторов Поддержка промышленных стандартов (МЭК 61131-3)
Цена	От 300 до 1500 рублей в зависимости от модификации и комплектации	От 15 000 до 50 000 рублей в зависимости от конфигурации

ESP32 лучше подходит для IoT-проектов, умного дома и прототипирования.

Овен ПЛК110 ориентирован на промышленную автоматизацию, где важны надежность и соответствие стандартам.

У ESP32 меньше рабочее напряжение, следовательно плата более экономична, чем Овен ПЛК110. У Овен ПЛК110 меньшее количество контактов, однако Овен ПЛК110 более прост для программирования. ESP32 значительно дешевле. И в нашем проекте Овен ПЛК110 не подходит, так как у него отсутствует Wi-Fi.

Вывод:

Плата ESP32 лучше всего подходит для дальнейшей работы, так как обладает важными преимуществами перед ESP8266, Arduino и Овен ПЛК110, так как имеет большее количество аналоговых входов и контактов GPIO, это нужно для нашего количества датчиков. Также ESP32 имеет большую оперативную и флэш-память, поддерживает Bluetooth и Wi-Fi, то есть обладает всем необходимым нам функционалом, поэтому остановимся на выборе этой платы.

2.2.2 Выбор и характеристики микроконтроллера

Исходя из результатов анализа сравнения микроконтроллеров, остановимся на выборе микроконтроллера семейства ESP32. Из семейства плат ESP32 рассмотрим плату ESP32-WROOM-32D, представленный на рисунке 9.

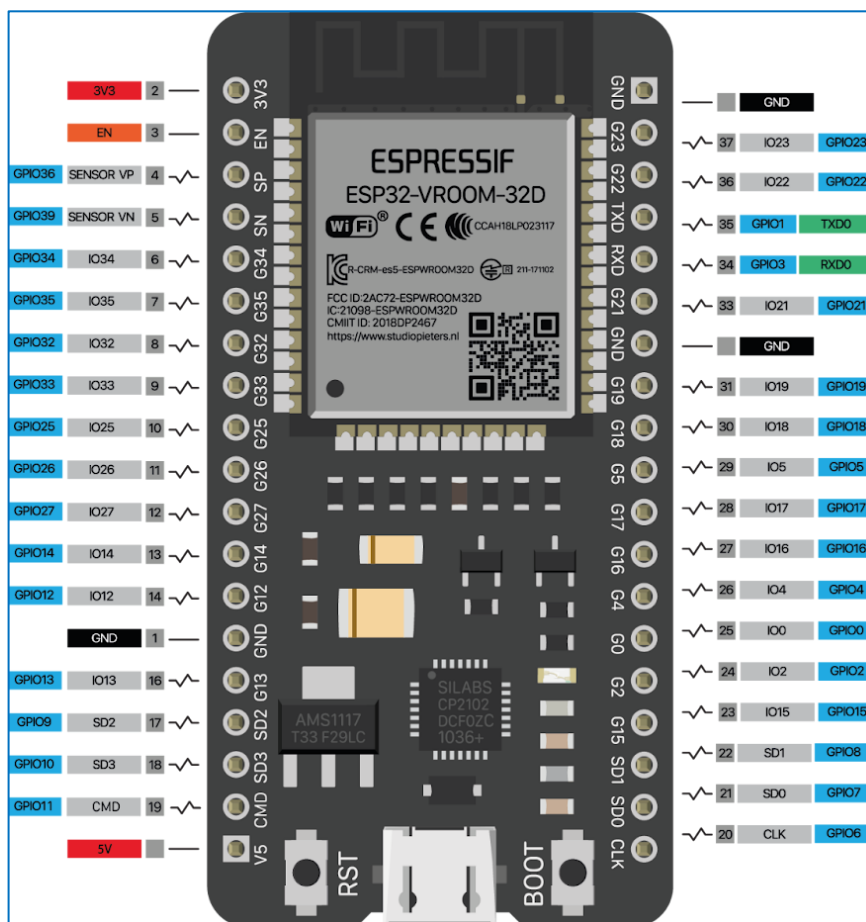


Рисунок 9 – Микроконтроллер ESP32-WROOM-32D

Рассмотрим какие функции у каждого пина платы ESP32-WROOM-32D, чтобы в дальнейшем подключать к нужным пинам другие устройства. Для этого воспользуемся таблицей 7.

Таблица 7 – ESP32-WROOM-32D

Назначение пинов	Номера пинов	Описание
Пины для входа	GPIO34, GPIO35, GPIO36, GPIO39	Эти контакты не могут быть использованы как выходы
SPI-flash	GPIO6 - GPIO11	Позволяют передавать данные и команды, а также контролировать специальные функции
Ёмкостные сенсорные датчики	GPIO0, GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO15, GPIO27, GPIO32, GPIO33	Обнаружение прикосновений и передачи данных в систему
Аналого-цифровой преобразователь	GPIO0, GPIO2, GPIO12, GPIO13, GPIO14, GPIO15, GPIO25, GPIO26, GPIO27, GPIO32, GPIO33, GPIO34, GPIO35, GPIO36, GPIO37, GPIO38, GPIO39	Измерение аналоговых сигналов
Цифро-аналоговый преобразователь	GPIO25, GPIO26	Нужны для подключения устройства к внешним устройствам. ЦАП преобразует цифровой сигнал в аналоговый, и пины нужны для передачи сигнала.
I2C	GPIO21, GPIO22	Связь между устройствами по протоколу I2C
VSPI	GPIO5, GPIO18, GPIO19, GPIO23	Реализация интерфейса SPI
HSPI	GPIO12, GPIO13, GPIO14, GPIO15	Реализация интерфейса SPI
UART0/UART1/UART2	GPIO1, GPIO3 / GPIO9, GPIO10 / GPIO16, GPIO17	Передача и приём данных между устройствами

Важными для нас являются наличие большого количества пинов аналого-цифрового преобразователя, две разновидности интерфейса SPI: VSPI и HSPI, наличие UART0/UART2 и пинов для работы с датчиками по интерфейсу I2C. Также у ESP32-WROOM-32D есть пины, предназначенные только на вход, соответственно мы сможем их использовать для получения данных. Именно исходя из этих характеристик была выбрана плата ESP32-WROOM-32D с 38 пинами, так как именно этот микроконтроллер обладает всеми необходимыми функциями и является оптимальным микроконтроллеру по соотношению цена/качество.

2.2.3 Выбор камеры видеонаблюдения, дисплеев и sd-card адаптера

Остановимся на выборе камеры для контроля за УЗВ рыбной фермы. В нашем случае достаточно, чтобы камера фиксировала на снимках корректную работу системы. Для этого нам не нужна камера с очень хорошим разрешением и высокими параметрами. Поэтому сделаем упор на простоту подключения устройства и цену товара. Специально для esp32 была создана как раз такая плата ESP32-CAM, представлена на рисунке 10. Она обладает невысокой ценой и достаточно для выполнения наших задач, остановимся на выборе этой платы.



Рисунок 10 – ESP32-CAM

Далее подберём дисплеи, на которые будут отображаться данные. В нашем устройстве присутствуют 2 дисплея: TFT-дисплей и дисплей диспетчера.

Дисплей диспетчера необходим непосредственно для работы диспетчера, поэтому он должен обладать размером экрана, не меньшим чем мобильный телефон. Определим, что для комфортной работы диспетчера

необходим дисплей с размером экрана не менее 7 дюймов, в то время как с обычным TFT-дисплеем диспетчер не работает и он нужен только для отображения информации, поэтому его размеры могут быть небольшими. Определим размеры для маленького обычного TFT-дисплея 1.8 дюймов. Для подключения дисплеев к микроконтроллеру необходима шина SPI. Исходя из этих характеристик выберем 2 дисплея. TFT-дисплей представлен на рисунке 11, а дисплей диспетчера на рисунке 12.



Рисунок 11 –TFT-дисплей



Рисунок 12 – Дисплей диспетчера

Также для монтирования sd-карты к микроконтроллеру необходим специальный sd-card адаптер. Для использования подойдёт самый простой sd-card адаптер, представленный на рисунке 13.

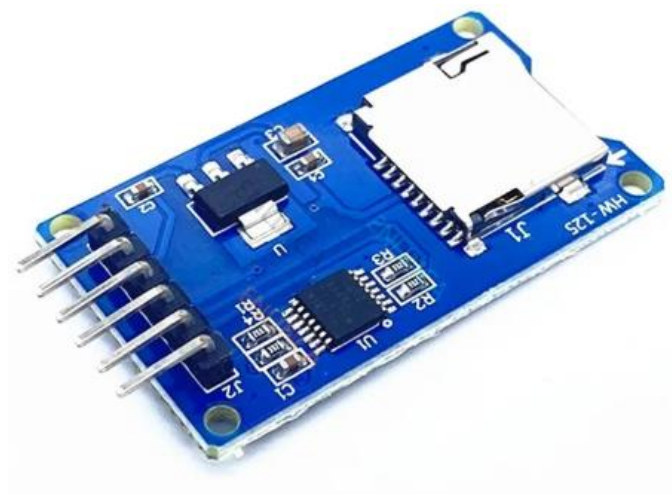


Рисунок 13 – sd-card адаптер

3 Разработка системы контроля УЗВ

Подобрав необходимые компоненты для создания устройства, переходим к разработке СК УЗВ.

3.1 Реализация подключения устройств

В данном разделе рассмотрим аппаратное подключение компонент к микроконтроллеру.

3.1.1 Интеграция датчиков в схему

Для дальнейшего проектирования системы будем использовать по одному датчику каждого типа, для уменьшения загромождённости схем. Рассмотрим принцип работы датчиков и опишем подключение датчиков к микроконтроллеру.

Для начала рассмотрим датчики FLC, WLA, WLB. Они имеют 2 контакта: VCC и сигнальный. Эти датчики являются самыми простыми по своему устройству. Принцип работы этих датчиков состоит в следующем: при достижении водой уровня, на котором расположен датчик (или при протекании по трубе через место, где расположен датчик, для датчика FLC) происходит срабатывание выключателя, это вызывает замыкание контактов. Соответственно при понижении воды (прекращении потока) контакт размыкается. При опросе датчика на микроконтроллер поступает либо “0”, либо “1”.

Далее рассмотрим аналоговые датчики мутности (MUT) и кислорода (O2). Они состоят из 3 контактов: VCC, GND и сигнального.

Принцип работы датчика мутности состоит в следующем: датчик состоит из двух частей – передатчика и приёмника. Принцип работы состоит в том, что передатчик излучает свет, а приёмник улавливает свет, пройденный через жидкость. В зависимости от наличия взвесей в воде приёмник улавливает разное количество испускаемого передатчиком света.

Датчик кислорода работает по следующему принципу: щуп датчика реагирует с молекулами кислорода, из-за этого происходит изменение электрических свойств, которое фиксируется датчиком и преобразуется в электрический сигнал, который считывается щупом.

Данные датчики измеряют сигнал и подают его на ножку микроконтроллера в качестве выходного сигнала. Каждому значению напряжения ставится в соответствие значение измеряемой величины, которое настраивается программно. Схема подключения датчиков, для наглядности на рисунке приведены по 1 примеру для каждого рассмотренного выше типа датчиков, представлена на рисунке 14.

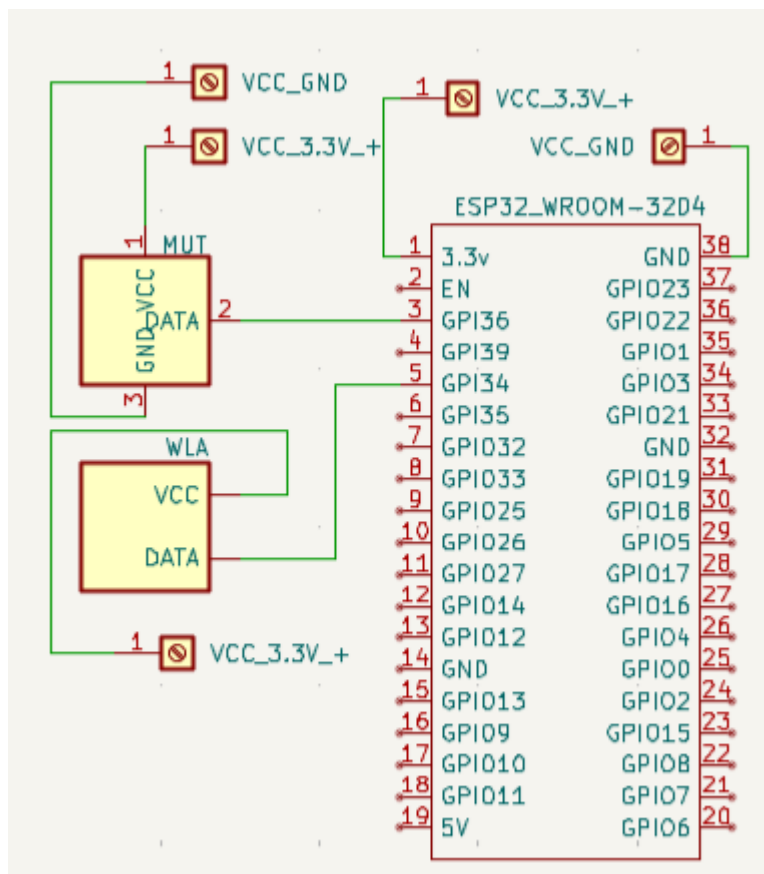


Рисунок 14 – Схема подключения датчиков (1)

Указанные в нашей модели датчики PHT, TDS и OVP работают при помощи UART. Поэтому для дальнейшего рассмотрения подключения данных датчиков и рассмотрения их принципа работы необходимо сначала рассмотреть понятие универсального асинхронного приёмопередатчика (UART).

UART является протоколом последовательной передачи данных, который позволяет меняться информацией микроконтроллеру и датчику без общего тактового сигнала.

UART – полнодуплексный интерфейс, то есть приемник и передатчик могут работать одновременно, независимо друг от друга. За каждым из них закреплена ножка микроконтроллера. Порт приемника обозначают RX, передатчика – TX.

Устройства соединяются противоположными портами друг к другу: $RX \rightarrow TX, TX \rightarrow RX$.

Пакет передачи данных UART, представленный на рисунке 15, состоит из старт-бита (значение 0), самих данных и стоп-бита (значение 1 или 2):

0	1	2	3	4	5	6	7	8	9
Старт-бит	Данные							Стоп-бит	

Рисунок 15 – пакет UART

Подключённым устройствам необходимо заранее установить скорость передачи данных и формат кадра. Обычно это скорость в диапазоне от 9600 до 115200 бит в секунду. Формат кадра определяется числом стоп-битов, а также числом бит данных и назначения 9 бита. К главным достоинствам такой передачи данных относится простота и отсутствие тактового сигнала.

Рассмотрим пример передачи символа по UART:

Передадим по UART символ 'А':

Код символа 'А' в ASCII равен 65. Представим символ в двоичном виде $65_{10} = 01000001_2$. Тогда UART отправит пакет 0010000011. Где в начале идёт старт-бит, а в конце стоп-бит.

Теперь же рассмотрим промышленный интерфейс для дальней связи RS-485.

RS-485 – это стандарт физического уровня, определяющий электрические характеристики и способ передачи данных по линии связи.

Принципом работы интерфейса RS-485 является дифференциальная передача данных. В интерфейсе RS-485 один из проводов в паре передает изначальный сигнал, а второй – его инверсную копию. Сигнал кодируется разностью потенциалов первого и второго проводов. Изобразим схематически сигнал на рисунке 16.

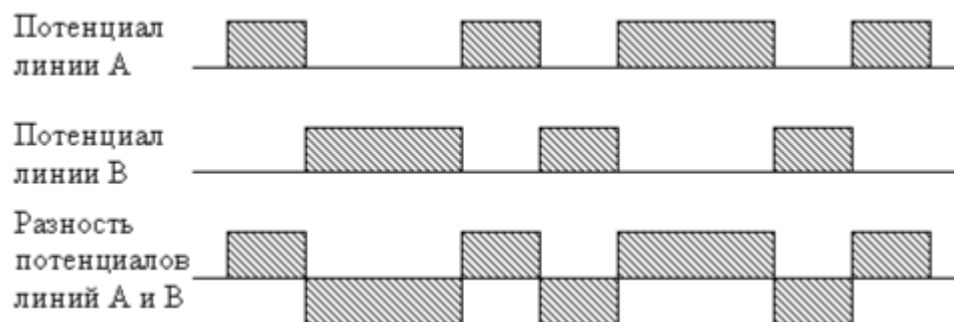


Рисунок 16 – Интерфейс RS-485

Главными преимуществами RS-485 являются:

- Дальность передачи. Данные могут передаваться на расстояние до 1200 метров;
- Устойчивость к помехам. Из-за дифференциальной передачи RS-485 не реагирует на электромагнитные помехи;
- Поддержка большого числа устройств. Возможность подключить до 32 устройств на одной шине;
- Экономическая эффективность. RS-485 совместим с недорогим оборудованием.

Аппаратной реализацией интерфейса являются микросхемы с дифференциальными входами/выходами к датчикам и цифровыми UART портами микроконтроллера TTL-RS485. На рисунке 17 представим микросхему TTL-RS485.

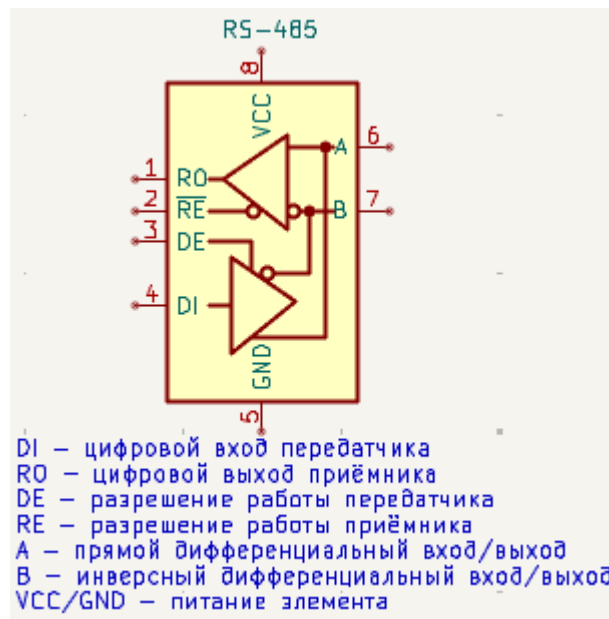


Рисунок 17 – TTL-RS485

Цифровой выход приёмника подключаем к порту приёмника (RX) на микроконтроллере, а цифровой вход передатчика – к порту передатчика (TX) на микроконтроллере. Пины DE и \overline{RE} являются управляющими пинами. Когда $DE=1$, происходит передача данных на линию A/B. При $RE=1$ микросхема принимает данные с линии A/B. При работе RS-485 только одно устройство может передавать данные в один момент времени. Пины DE и \overline{RE} дают возможность переключают режимы приёма/передачи и работать только с одним активным устройством. Пины DE и \overline{RE} можно объединить и подключить к одному управляющему пину на микроконтроллере. Для активации передачи на этот пин подают “1”, а для приёма – “0”. Схема подключения микроконтроллера к TTL-RS485 представлена на рисунке 18.

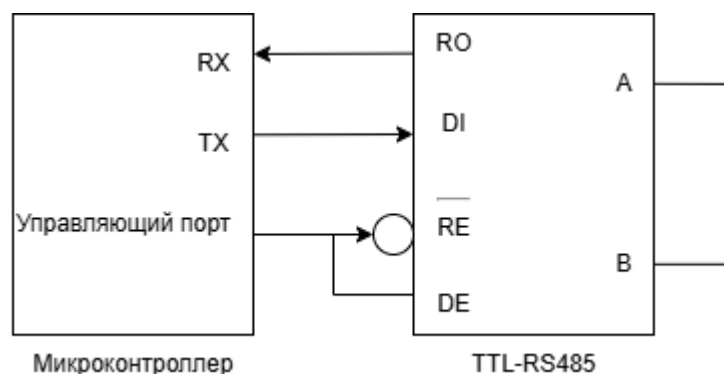


Рисунок 18 – Схема подключения микроконтроллера к TTL-RS485

Логика управления представлена в таблице 8.

Таблица 8 – Логика управления TTL-RS485

Режим	DE	\overline{RE}	Состояние микросхемы
Передача	1	0	Данные с DI от UART идут на линию A/B
Приём	0	1	Данные с линии A/B идут на RO в UART
Отключение	0	0	Выключено

Рассмотрев схему подключения микроконтроллера к TTL-RS485-переходнику детально рассмотрим принцип работы датчиков TDS, OVP и PHТ. Сами датчики подключаются одноимёнными шинами к шинам А и В.

Датчик TDS состоит из щупа сенсора в пластиковом цилиндре с двумя электродами на конце. Между электродами возникает сопротивление, которое фиксируется и обрабатывается платой управления.

Датчик PH: щуп опускается в воду, датчик считывает разницу потенциалов между выводами щупа и преобразует полученные данные в отклонение водородного показателя жидкости от нейтрального значения

Датчик OVP состоит из двух электродов: индикаторного и электрода сравнения. В растворе электроны взаимодействуют с электродами и создают между ними напряжения, которое считывается датчиком.

Эти датчики управляются через TTL-RS485 переходник управляющим пином на микроконтроллере. Теперь, когда мы рассмотрели принципы работы этих датчиков, изобразим схему подключения датчиков TDS, PHТ и OVP к микроконтроллеру ESP32-WROOM-32D на рисунке 19.

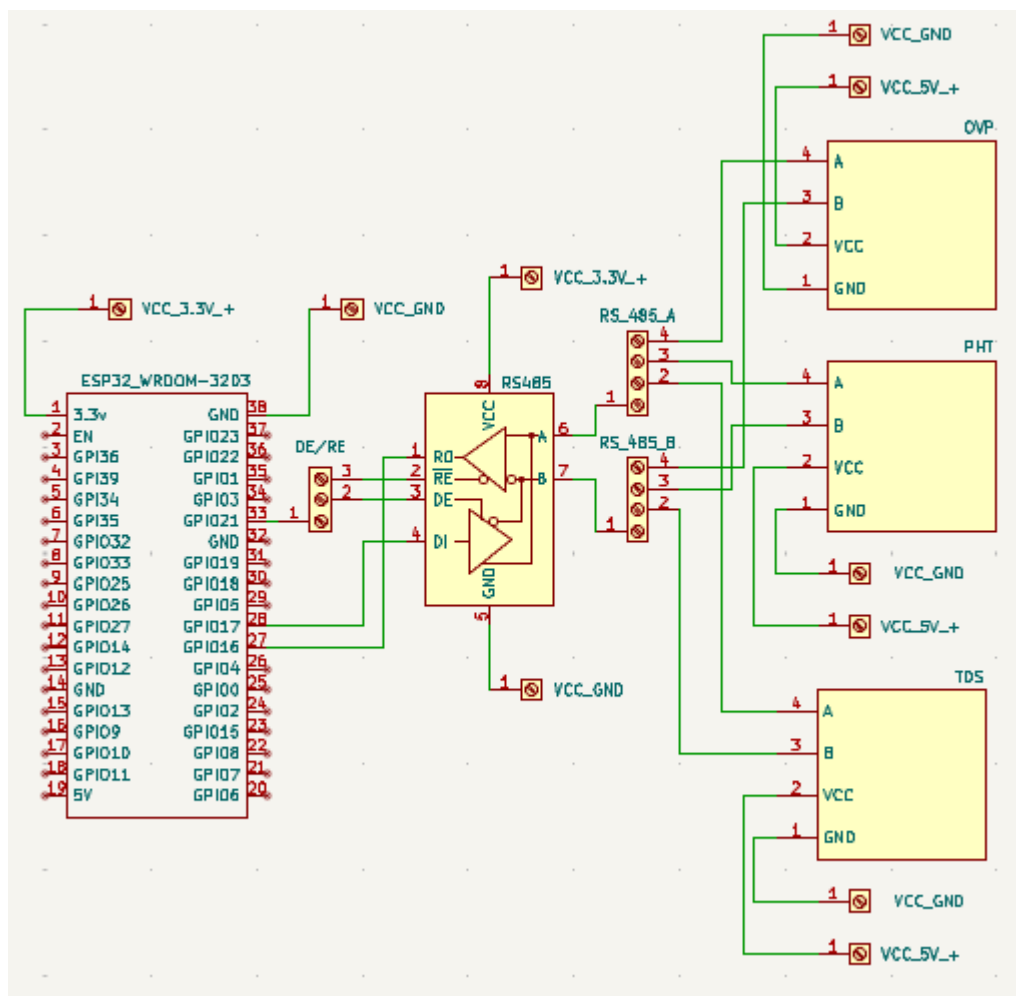


Рисунок 19 – Подключение датчиков TDS,PHT и OVP

3.1.2 Работа с sd-картой, дисплеем и камерой

Для подключения SD-карты и TFT-дисплея был использован единый SPI-интерфейс. Рассмотрим подробнее данный интерфейс:

SPI (Serial Peripheral Interface) - синхронный последовательный интерфейс, который служит для обмена данными между микроконтроллером и периферийными устройствами, в нашем случае sd-картой и TFT-дисплеем.

SPI использует архитектуру “Master – Slave”, где ведущее устройство Master управляет обменом данными с ведомыми Slave.

Для работы SPI интерфейса необходимо минимум 4 сигнальные линии:

- сигнал SCK является сигналом синхронизации и нужен для синхронизации при передаче данных;
- сигнал CS используется для выбора ведомого устройства;

- сигнал MOSI – сигнал, по которому происходит передача данных от управляющего устройства к ведомому;
- сигнал MISO – сигнал, передающий данные от ведомого устройства к управляющему.

Для работы дисплеев используются также дополнительные сигналы:

- сигнал AO – сигнал, который служит для разделения команд и данных;
- сигнал RESET – аппаратный сброс устройства.

Принцип работы SPI представлен на рисунке 20.



Рисунок 20 – Принцип работы SPI

При подключении нескольких ведомых устройств при подключении Slave устройств к Master используются шины SCK, MOSI и MISO, однако для каждого устройства необходимо использовать отдельный CS-пин. Рассмотрим подключение нескольких устройств на рисунке 21.

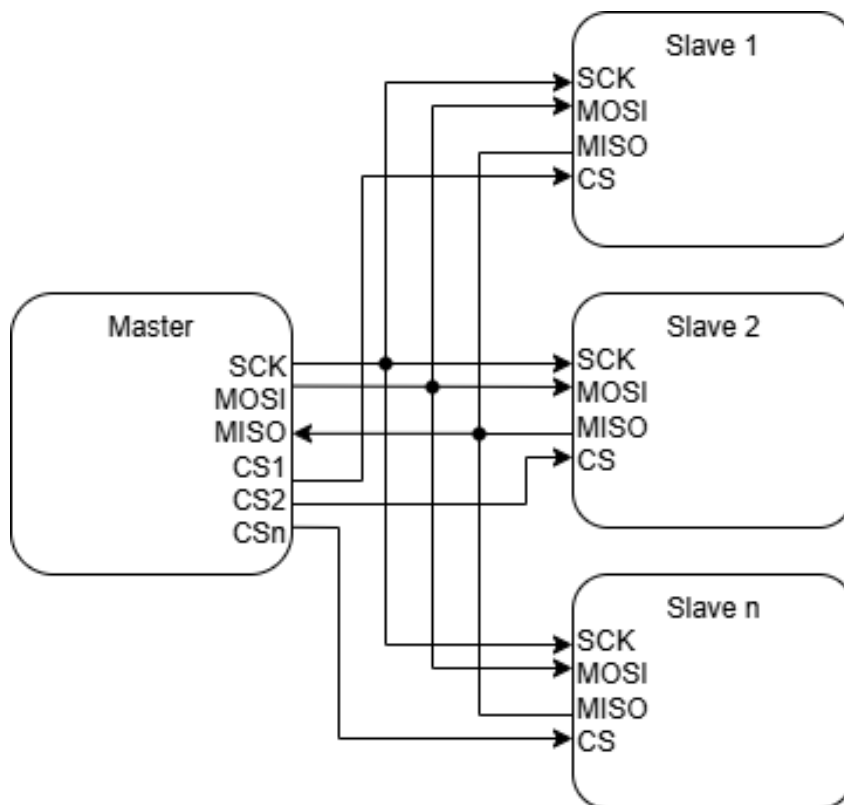


Рисунок 21 – Подключение нескольких устройств

Режим SPI поддерживает 4 режима работы, которые определяются комбинацией:

- CPOL – уровень SCK в ожидании:
 - CPOL = 0 – LOW;
 - CPOL = 1 – HIGH.
- CPHA – момент защёлкивания данных:
 - CPHA = 0 – данные считываются по фронту SCK;
 - CPHA = 1 – данные считываются по срезу SCK.

В таблице 9 представлены режимы работы SPI:

Таблица 9 – Режимы работы SPI

Режим	CPOL	CPHA	Описание
0	0	0	Данные считываются по возрастающему фронту (SCK из LOW в HIGH)
1	0	1	Данные считываются по спадающему фронту (SCK из HIGH в LOW)
2	1	0	Данные считываются по спадающему фронту (SCK из HIGH в LOW)
3	1	1	Данные считываются по возрастающему фронту (SCK из LOW в HIGH)

sd-карта в нашем устройстве использует режим 0, а TFT-дисплей – режим 3.

После того, как мы рассмотрели интерфейс SPI, подключим аппаратно sd-карту и TFT-дисплей. При подключении важно понимать, что устройства не могут работать одновременно и в один момент времени может быть активен только один CS пин. Поэтому, изображённая на рисунке 22 схема работы с sd-картой и дисплеем в нашем устройстве будет следующая:

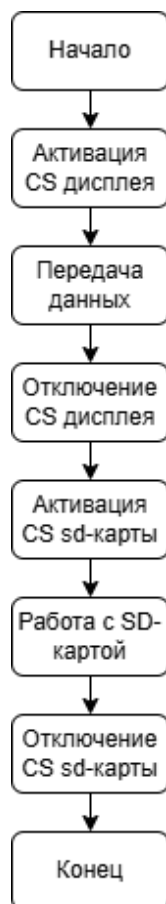


Рисунок 22 – Схема работы с sd-картой и TFT-дисплеем

Так как подключаются несколько устройств, то сигналы MISO, MOSI и SCK будут общими, а сигнал CS у каждого устройства будет свой. Но так как на дисплей будет передаваться информация для вывода, а с дисплея информация на микроконтроллер не передаётся, то пин MISO не будет использован, и схема подключения примет вид, представленный на рисунке 23.

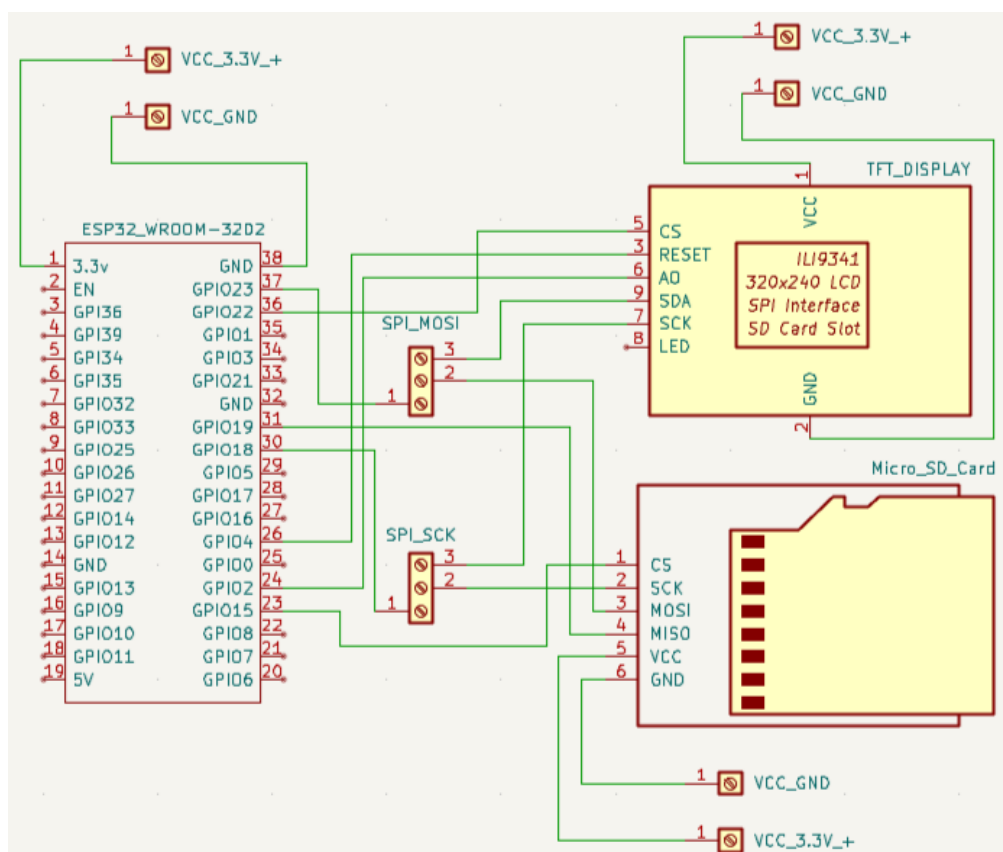


Рисунок 23 – Схема подключения sd-карты и TFT-дисплея

3.1.3 Подключение ESP32-CAM

Камеру ESP32-CAM подключим при помощи UART. Однако ранее мы уже использовали UART2 для подключения датчиков через RS485. Поэтому нужно использовать другой UART. Так как ESP32-WROOM-32D обладает UART0-UART2, то неиспользованными остаются UART0 и UART1, но UART0 по умолчанию используется для загрузки. Исходя из этого, подключим ESP32-CAM к UART1 на рисунке 24.

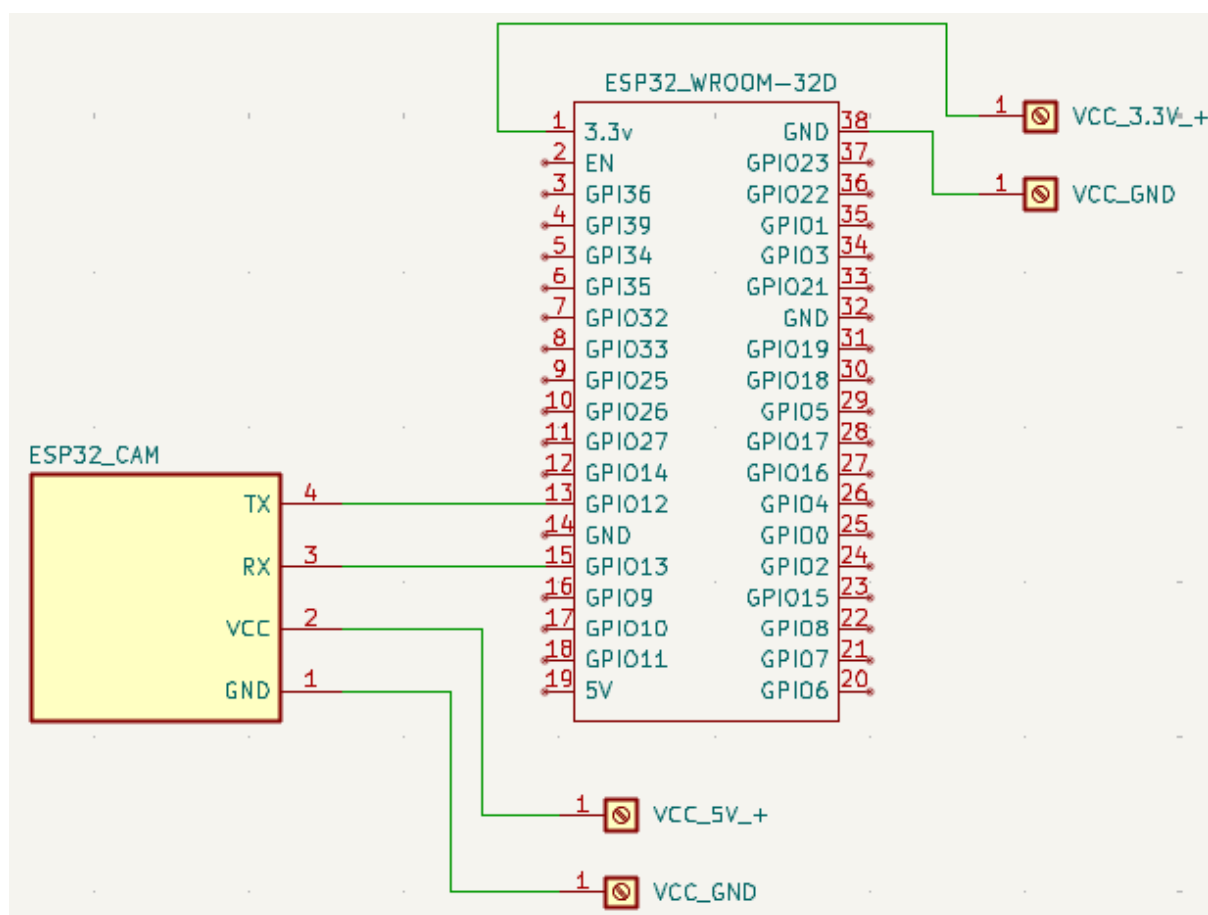


Рисунок 24 – Схема подключения ESP32-CAM

3.1.4 Общая схема подключённых устройств

После того, как были рассмотрены подключения отдельных компонент схемы к микроконтроллеру ESP32-WROOM-32D, объединим все комплектующие в общую схему на рисунке 25 для дальнейшего сбора и программирования устройства. Также, чтобы не подавать каждый раз отдельно напряжение на каждый элемент устройства, спроектируем шины для 3.3 и 5 вольт, куда будем подключать компоненты и подавать напряжение.

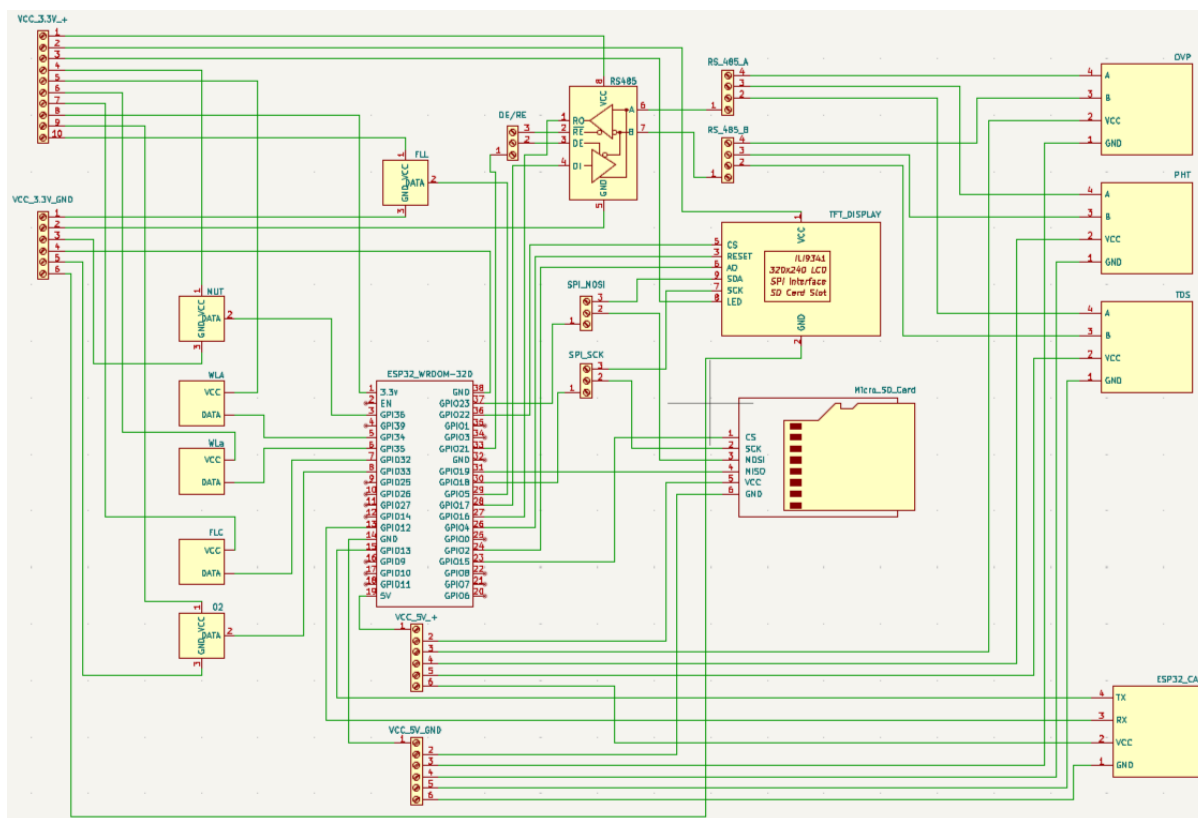


Рисунок 25 – Общая схема устройства

3.2 Конфигурация среды разработки

Перед тем, как начать программирование разработанного устройства, необходимо сконфигурировать и настроить среду разработки. В качестве редактора кода будем использовать Visual Studio Code - мощную среду разработки с поддержкой плагинов. Для программирования микроконтроллеров ESP32/ESP8266 на языке СИ нам понадобится фреймворк ESP-IDF. На рисунках 26 и 27 представлен данный фреймворк и проверена его установка.

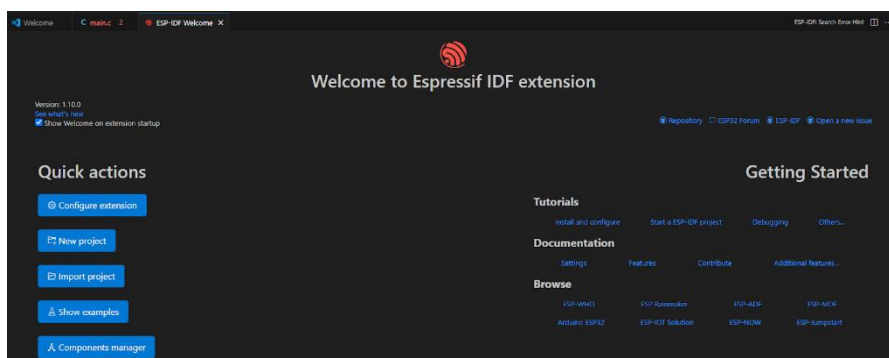


Рисунок 26 – ESP-IDF

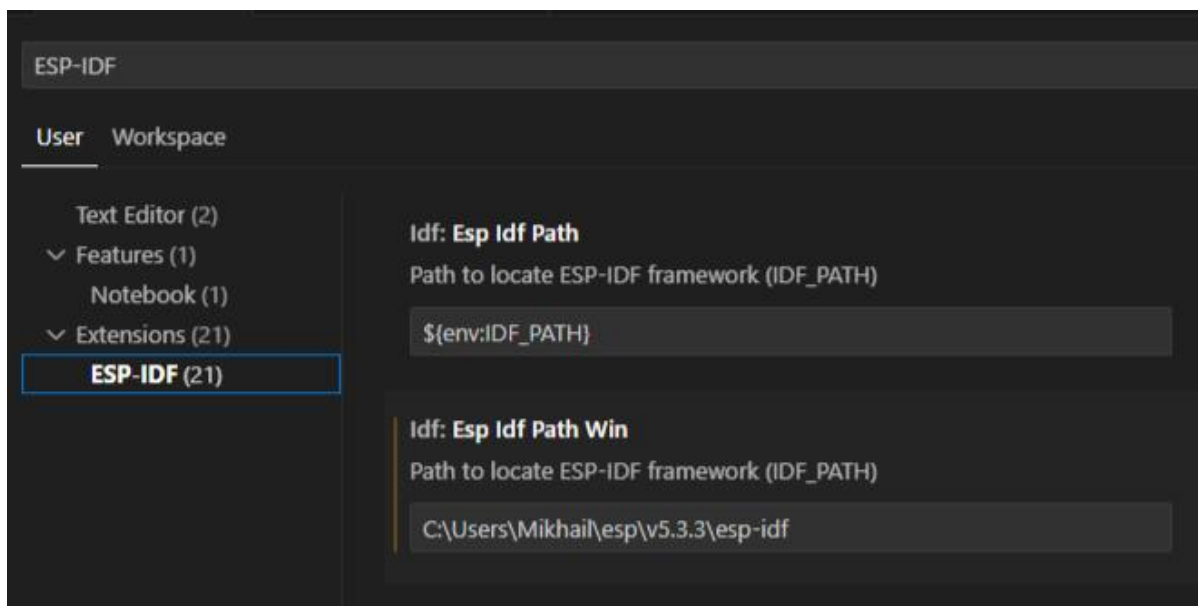


Рисунок 27 – Проверка установки ESP-IDF

Для компиляции и прошивки необходим набор инструментов Toolchain для ESP-IDF. Подтвердим его установку при помощи рисунка 28.

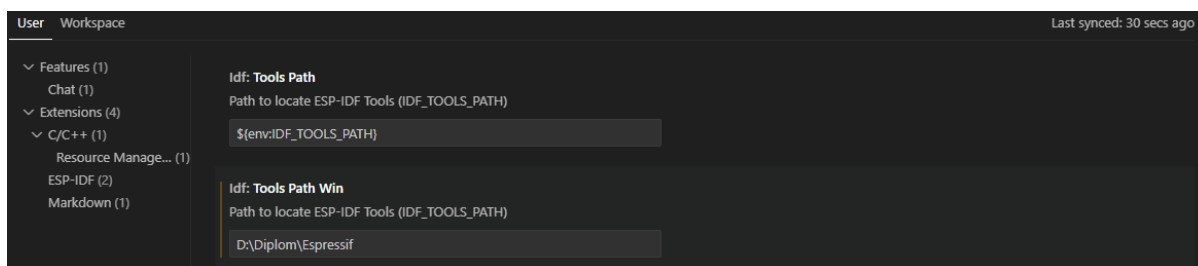


Рисунок 28 – Проверка установки Toolchain

Для программирования системы УЗВ, был создан проект со структурой, изображённой на рисунке 29.

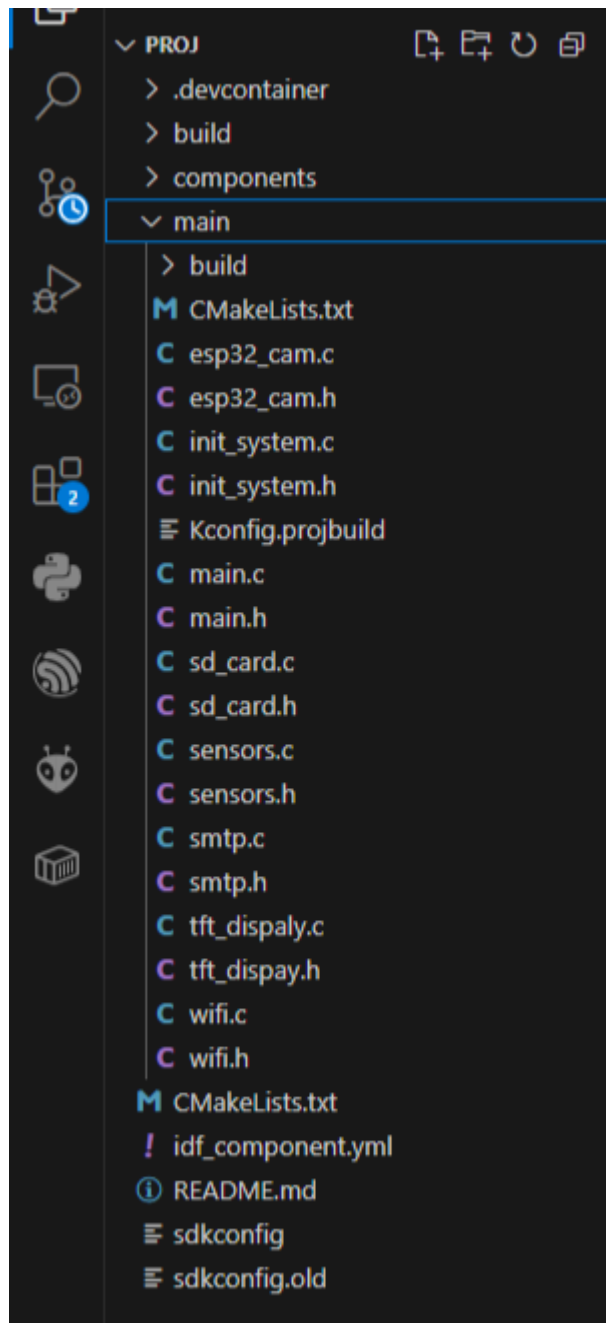
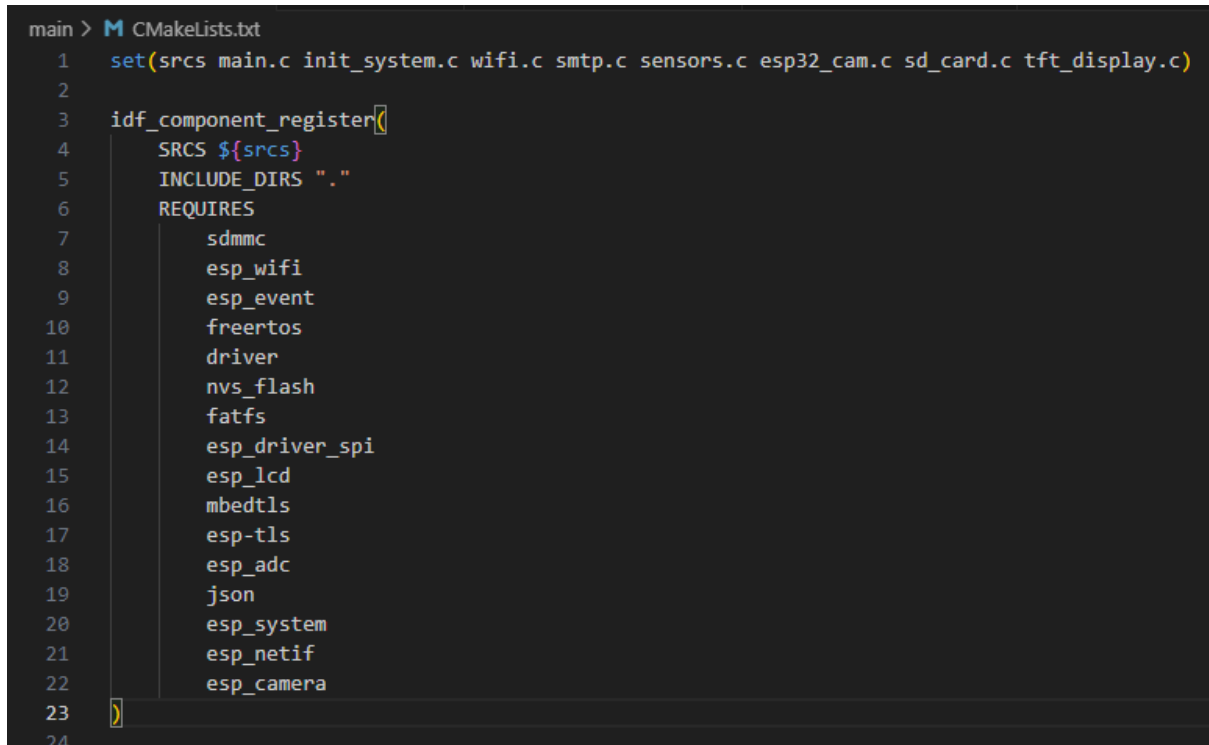


Рисунок 29 – Структура программируемого устройства

В папке `main` содержатся папка `build`, в которой находятся все файлы, которые появляются после сборки проекта, файлы конфигурации созданного проекта: `sdconfig` и `sdkconfig.old`.

Основными же файлами являются файл `CMakeLists.txt`, в котором указываются все зависимости проекта, а также исходные и заголовочные файлы программы работы с камерой, инициализации системы, основного кода программы, подключения к датчикам, отправка данных, подключение к дисплею и wi-fi.

Отдельно рассмотрим содержимое файла CMakeLists.txt на рисунке 30.



```
main > M CMakeLists.txt
1  set(srcs main.c init_system.c wifi.c smtp.c sensors.c esp32_cam.c sd_card.c tft_display.c)
2
3  idf_component_register(
4      SRCS ${srcs}
5      INCLUDE_DIRS "."
6      REQUIRES
7          sdmmc
8          esp_wifi
9          esp_event
10         freertos
11         driver
12         nvs_flash
13         fatfs
14         esp_driver_spi
15         esp_lcd
16         mbedtls
17         esp-tls
18         esp_adc
19         json
20         esp_system
21         esp_netif
22         esp_camera
23  )
24
```

Рисунок 30– Содержимое файла CMakeLists.txt

В переменной `${srcs}` указаны все исходные файлы, которые содержатся в исходном проекте и были описаны ранее.

`INCLUDE_DIRS` – это параметр, который указывает директории, где находятся заголовочные файлы, нужные для компиляции исходных файлов. “.” означает, что заголовочные файлы хранятся в той же директории, что и файл CMakeLists.txt.

`REQUIRES` – это компоненты ESP-IDF, необходимые для сборки проекта:

- `freertos` – обеспечивает управление задачами, служит для распараллеливания программы. В программе для считывания значений с разных типов датчиков, а также вывод их на дисплей и сохранение данных на sd-карту происходит параллельно. У каждой задачи устанавливается приоритет,
- `driver` – обеспечивает подключение GPIO, SPI, UART, ADC. Используется для взаимодействия устройств,
- `esp_system` – используется для управления питанием и системными событиями,

- esp_netif – обеспечивает работу Wi-Fi,
- nvs_flash – служит для хранения настроек Wi-Fi и калибровочных данных датчиков,
- esp_event – создаёт обработчики событий,
- sdmmc – является драйвером для работы с sd-картами и нужен для записи и чтения данных на карту памяти,
- fatfs – отвечает за поддержку системы FAT32,
- esp_wifi – создаёт подключение устройства к роутеру,
- mbedtls и esp-tls используются для передачи данных на почту,
- esp_adc – нужен для чтения значений с аналоговых датчиков,
- esp_lcd – драйвер для дисплеев. Необходим для вывода информации на экран,
- esp_driver_spi – драйвер SPI, используется для связи с дисплеем и sd-картой,
- esp_camera – отвечает за работу камеры, создание фото и передачу изображений,
- json – парсинг и создание JSON-файлов.

Также рассмотрим сборку программы на рисунке 31.

.rodata	215508	5.14		
.appdesc	256	0.01		
IRAM	111795	85.29	19277	131072
.text	110767	84.51		
.vectors	1027	0.78		
DRAM	35872	19.85	144864	180736
.bss	18928	10.47		
.data	16944	9.38		
RTC SLOW	24	0.29	8168	8192
.rtc_slow_reserved	24	0.29		
RTC FAST	4	0.05	8188	8192
.force_fast	4	0.05		
Total image size: 1068344 bytes (.bin may be padded larger)				

Рисунок 31 – Сборка программы

3.3 Описание структуры для хранения данных

Для хранения и передачи данных с датчиков используются файлы формата JSON. Данный формат был выбран из-за своего удобства для анализа и хранения данных, а также данный формат файлов легко применяется для дальнейшего использования данных и интеграции с другими системами. Этот формат файлов использует пары ключ-значение, что делает его легкочитаемым и более компактным, чем XML. Структура файла json представлена в таблице 10.

Таблица 10 – Структура файла json

Ключ	Тип данных
“Время >>>”	string
“Датчики уровня воды”	array
“Датчик потока”	number
“Датчик наличия потока”	string
“TDS датчик”	number
“OVP датчик”	number
“РН датчик”	number
“Температура воды”	number
“Датчик мутности”	number
“Датчик кислорода”	number

3.4 Работа с Wi-Fi и протоколом smtp

Ранее были рассмотрены алгоритмы подключения устройств к микроконтроллеру. Теперь опишем подключение самого микроконтроллера к Wi-Fi и передачу данных на почту оператора.

Алгоритм подключения к Wi-Fi представлен на рисунке 32.



Рисунок 32 – Алгоритм подключения к Wi-Fi

Также при подключении к wi-fi, использовалась обработка событий. В случае неудачного подключения вызывалась повторная попытка подключиться к сети. Обработка событий в функции `event_handler()` представлена на рисунке 33.



Рисунок 33 – Обработка событий в функции `event_hadler()`

После удачного подключения к Wi-Fi, можно передавать сообщения на почту. Для передачи данных будем использовать протокол SMTP. Он предназначен для отправления и обработки писем. Принцип работы SMTP представлен на рисунке 34. После установления соединения отправителя с сервером отправителя, устанавливается соединение с сервером получателя и в случае доступа получателя, сервер получателя принимает данные.



Рисунок 34 – Принцип работы SMTP

Для обеспечения безопасности соединения использовалось TLS-шифрование. Принцип работы TLS заключается в шифровании информации на двух этапах: при установлении подключения, когда клиент и сервер проверяют подлинность и договариваются о правилах шифрования, после чего начинается обмен данными. Данные разбиваются на пакеты и шифруются отдельно, но имеют общий ключ.

На рисунке 35 представлен алгоритм передачи данных по smtp с использованием tls, который реализован в файле smtp.c.



Рисунок 35 – Алгоритм передачи данных по smtp с использованием tls

После отправления данных программа погружается “в сон” и следующий шаг цикла (считывание данных с датчиков, сохранение данных, подключение и отправка данных по почте) выполняется через указанное время.

3.5 Сохранение и обработка полученных данных

Приём и сохранение данных на стороне получателя были реализованы при помощи языка python в VS_CODE в файле data_save.py. Данные сохранялись в базу данных SQLite esp32_data.db и в папку Data_esp32. Были импортированы модули для работы с электронной почтой, базами данных SQLite и json-файлами, логированием, временем. Они представлены на рисунке 36.

```
import imaplib
import email
import json
import sqlite3
import os
import time
import schedule
from datetime import datetime
import logging
import winreg
import sys
from email.utils import parsedate_to_datetime
```

Рисунок 36 – Импортированные модули

Принцип работы программы представлен на рисунке 37. Программа создаёт БД и папку, куда сохраняет данные, как только получает письмо от заданного пользователя. Также есть файл, куда записываются вся работа с почтой, и если сообщение прочитано некорректно, то туда происходит запись ошибки

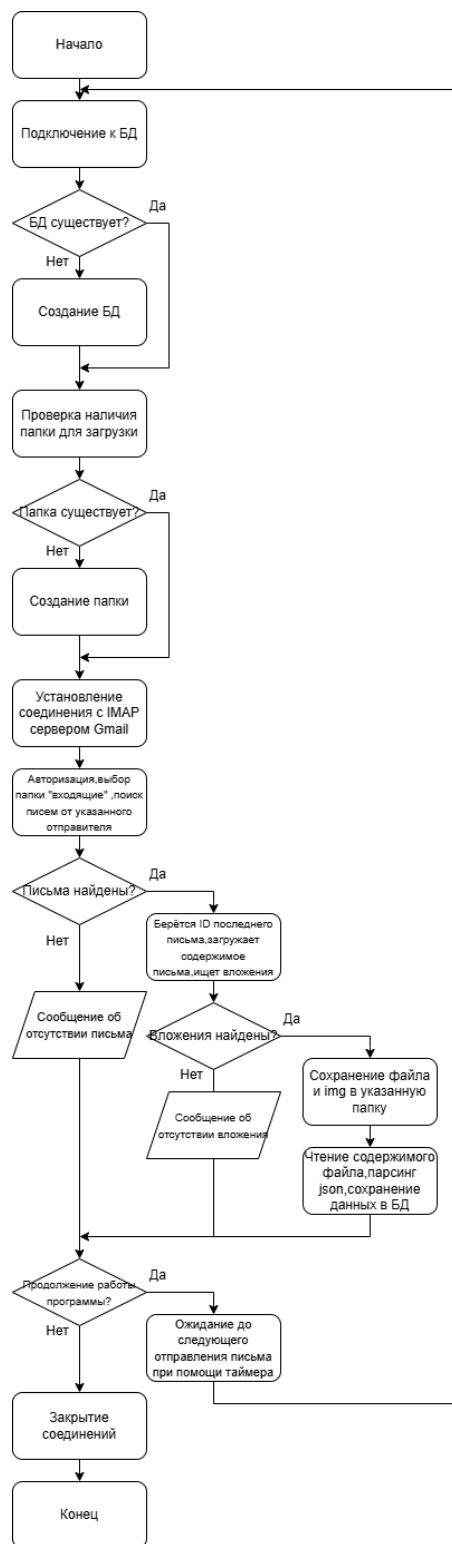


Рисунок 37 – Принцип работы программы data_save.py

3.6 Обработка и визуализация данных

Также была написана программа для визуализации данных из БД. Программа должна выводить измерения за период, указанный пользователем, строить графики и показывать, спроектированную систему в Компас 3D и просмотр фото. У пользователя должна быть возможность получать данные за конкретную дату, период дат и сравнивать графики за определённые даты.

Для визуализации работы программы представим диаграмму классов на рисунке 38 и диаграмму последовательностей программы на рисунке 39.

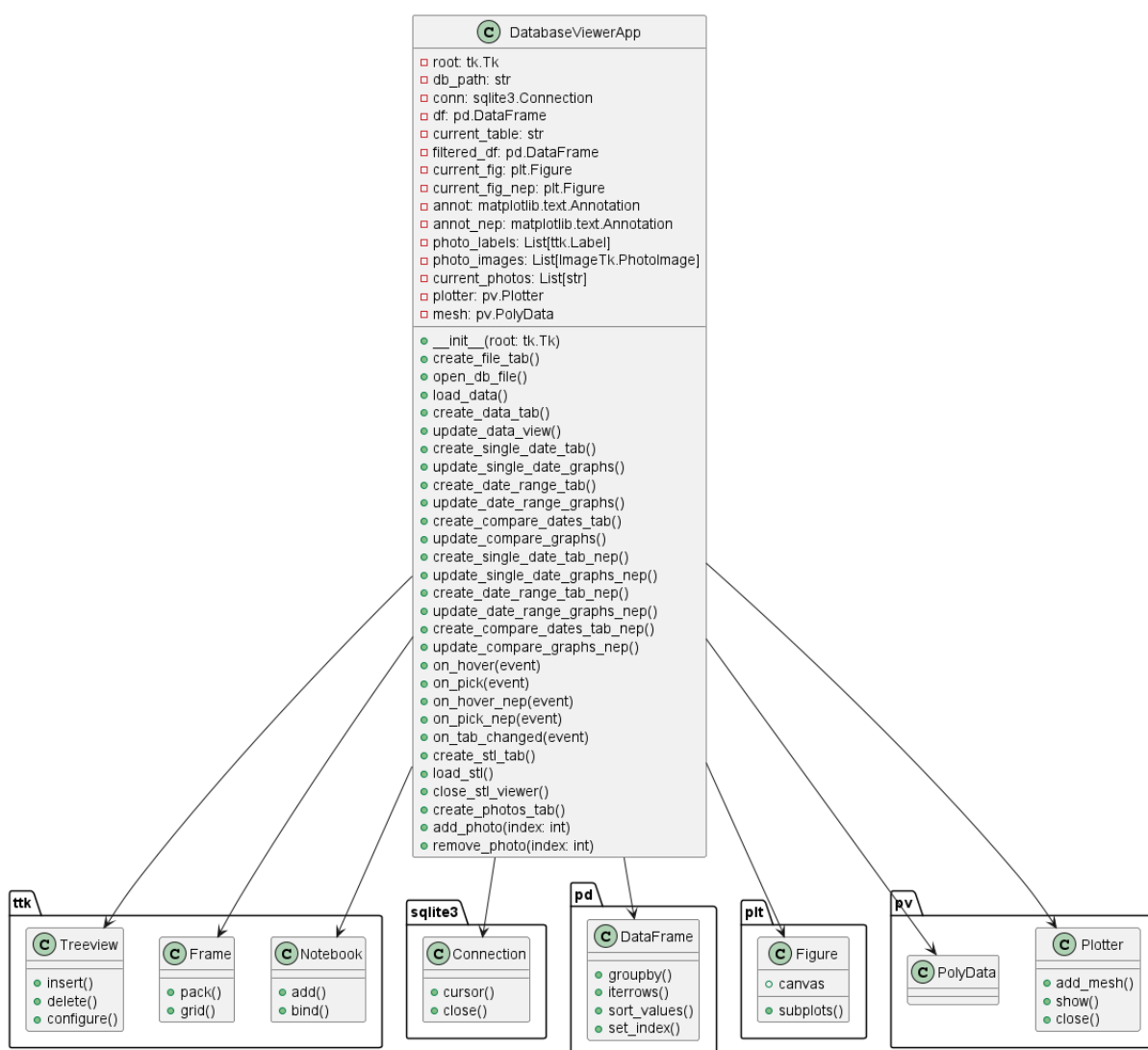


Рисунок 38 – Диаграмма классов

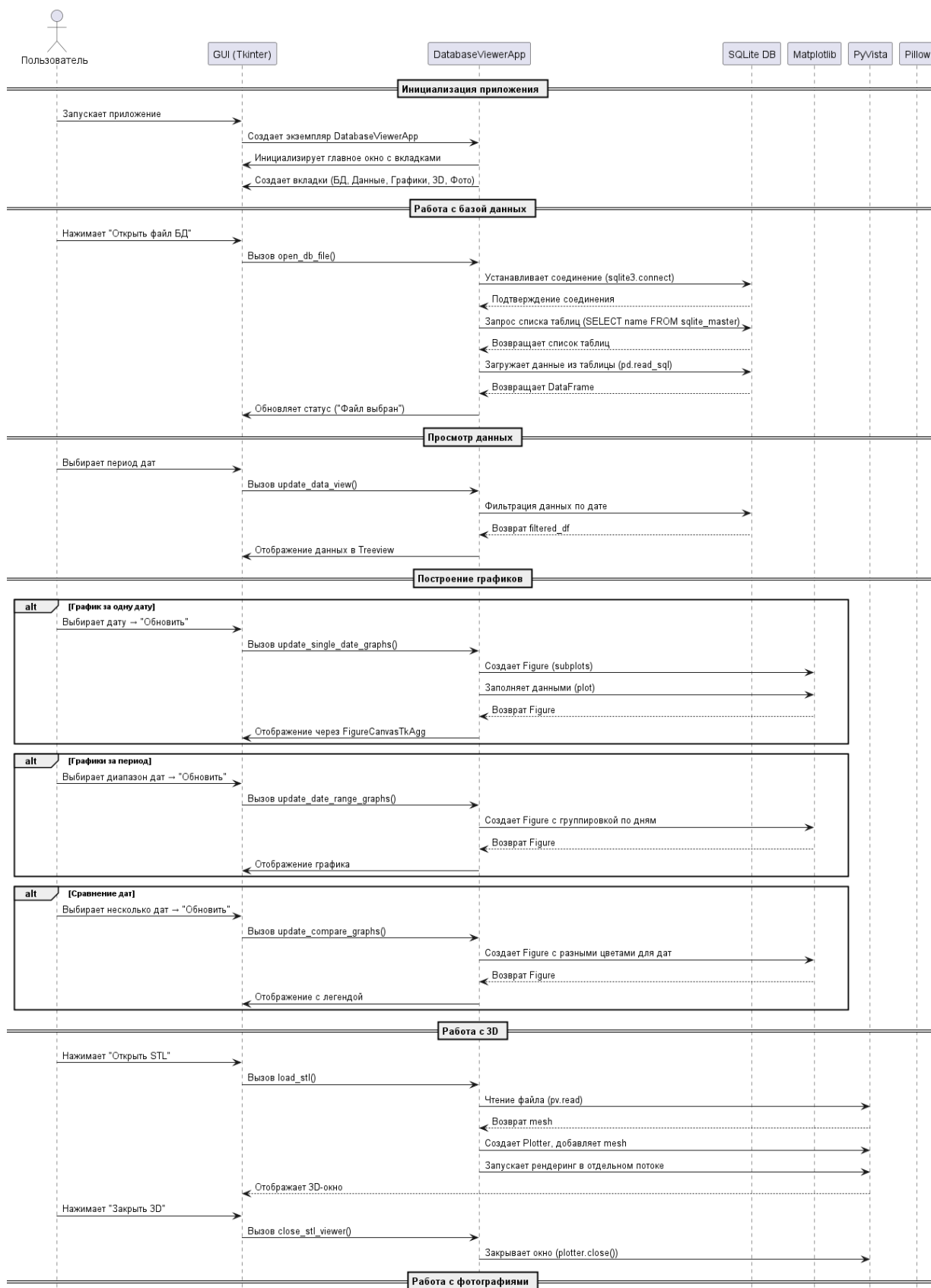


Рисунок 39 – Диаграмма последовательностей программы

3.7 Фрагменты функций

Рассмотрим фрагменты кода, которые выполняют определённые функции. На рисунках 40–45 рассмотрим вспомогательную функцию инициализации, функции инициализации и sd-карты и TFT-дисплея, и функции конвертации сырых значений аналоговых датчиков в реальные.

```
1416 void init_spi_bus() {  
1417     if (spi_initialized) return;  
1418  
1419     spi_bus_config_t buscfg = {  
1420         .mosi_io_num = PIN_NUM_MOSI,  
1421         .miso_io_num = PIN_NUM_MISO,  
1422         .sclk_io_num = PIN_NUM_CLK,  
1423         .quadwp_io_num = -1,  
1424         .quadhd_io_num = -1,  
1425         .max_transfer_sz = 4096,  
1426         .flags = SPICOMMON_BUSFLAG_MASTER  
1427     };  
1428  
1429     ESP_ERROR_CHECK(spi_bus_initialize(HSPI_HOST, &buscfg, SPI_DMA_CH_AUTO));  
1430     spi_initialized = true;  
1431 }
```

Рисунок 40 – Функция init_spi_bus()

```
516 bool init_sd_card() {  
517     if (sd_card_initialized) return true;  
518  
519     init_spi_bus();  
520  
521     sdmmc_host_t host = SDSPI_HOST_DEFAULT();  
522     host.slot = HSPI_HOST;  
523     host.max_freq_khz = 400;  
524  
525     gpio_set_pull_mode(PIN_NUM_CS, GPIO_PULLUP_ONLY);  
526     gpio_set_direction(PIN_NUM_CS, GPIO_MODE_OUTPUT);  
527     gpio_set_level(PIN_NUM_CS, 1);  
528     gpio_set_pull_mode(PIN_NUM_MISO, GPIO_PULLUP_ONLY);  
529  
530     sdspi_device_config_t slot_config = SDSPI_DEVICE_CONFIG_DEFAULT();  
531     slot_config.host_id = HSPI_HOST;  
532     slot_config.gpio_cs = PIN_NUM_CS;  
533  
534     esp_vfs_fat_mount_config_t mount_config = {  
535         .format_if_mount_failed = false,  
536         .max_files = 5,  
537         .allocation_unit_size = 16 * 1024  
538     };  
539  
540     esp_err_t ret = esp_vfs_fat_sdspi_mount("/sdcard", &host, &slot_config,  
541         &mount_config, &card);  
542     if (ret != ESP_OK) {  
543         ESP_LOGE(TAG_INIT, "Failed to mount SD card: %s", esp_err_to_name(ret));  
544         return false;  
545     }  
546  
547     sd_card_initialized = true;  
548     return true;  
549 }
```

Рисунок 41 – Функция init_sd_card()

```

473
474 bool TFT_init() {
475     init_spi_bus();
476     // Настройка GPIO
477     gpio_set_direction(TFT_DC, GPIO_MODE_OUTPUT);
478     gpio_set_direction(TFT_CS, GPIO_MODE_OUTPUT);
479     gpio_set_level(TFT_CS, 1);
480     gpio_set_direction(TFT_RST, GPIO_MODE_OUTPUT);
481     // Сброс дисплея
482     gpio_set_level(TFT_RST, 0);
483     vTaskDelay(100 / portTICK_PERIOD_MS);
484     gpio_set_level(TFT_RST, 1);
485     vTaskDelay(100 / portTICK_PERIOD_MS);
486     // Конфигурация SPI устройства
487     spi_device_interface_config_t tft_config = {
488         .clock_speed_hz = 5 * 1000 * 1000,
489         .mode = 0,
490         .spics_io_num = TFT_CS,
491         .queue_size = 7,
492         .pre_cb = NULL,
493         .post_cb = NULL,
494     };
495
496     if (spi_bus_add_device(HSPI_HOST, &tft_config, &tft_spi) != ESP_OK) {
497         return false;
498     }
499     // Инициализация дисплея
500     send_cmd(0x01);
501     vTaskDelay(150 / portTICK_PERIOD_MS);
502     send_cmd(0x11);
503     vTaskDelay(255 / portTICK_PERIOD_MS);
504     send_cmd(0x3A);
505     send_data(0x05);
506     send_cmd(0x29);
507     return true;
508 }

```

Рисунок 42 – Функция TFT_init()

```

850 static float convert_mutnost(uint32_t voltage) {
851     return 0.4 * voltage - 50.0;
852 }
853
854 static float convert_oxygen(uint32_t voltage) {
855     float V_zero = 0.0;
856     float V_sat = 2.0;
857     float DO_span = 20.0;
858
859     float DO = (voltage - V_zero) * (DO_span / (V_sat - V_zero));
860     return DO;
861 }
862
863 static float convert_bhthdi(uint32_t voltage) {
864     return (voltage - 500.0) * 0.1;
865 }

```

Рисунок 43 – Функции преобразования сырых значений

```

681 uint16_t modbus_crc(uint8_t *buf, int len) {
682     uint16_t crc = 0xFFFF;
683     for (int pos = 0; pos < len; pos++) {
684         crc ^= (uint16_t)buf[pos];
685         for (int i = 8; i != 0; i--) {
686             if ((crc & 0x0001) != 0) {
687                 crc >>= 1;
688                 crc ^= 0xA001;
689             } else {
690                 crc >>= 1;
691             }
692         }
693     }
694     return crc;
695 }
696

```

Рисунок 44 – Функция вычисления CRC для сообщения MODBUS

```

697 ✓ bool send_modbus_request(uint8_t addr, uint8_t function, uint16_t reg_addr, uint16_t reg_count) {
698     uint8_t request[8];
699     request[0] = addr;
700     request[1] = function;
701     request[2] = (reg_addr >> 8) & 0xFF;
702     request[3] = reg_addr & 0xFF;
703     request[4] = (reg_count >> 8) & 0xFF;
704     request[5] = reg_count & 0xFF;
705
706     uint16_t crc = modbus_crc(request, 6);
707     request[6] = crc & 0xFF;
708     request[7] = (crc >> 8) & 0xFF;
709
710     gpio_set_level(RTS_PIN, 1);
711 ✓   if (uart_write_bytes(UART_PORT_NUM, request, 8) < 0) {
712       gpio_set_level(RTS_PIN, 0);
713       return false;
714   }
715
716 ✓   if (uart_wait_tx_done(UART_PORT_NUM, pdMS_TO_TICKS(100)) != ESP_OK) {
717       gpio_set_level(RTS_PIN, 0);
718       return false;
719   }
720
721     gpio_set_level(RTS_PIN, 0);
722     return true;
723 }

```

Рисунок 45 – Функция формирования и отправки MODBUS-запроса

```

725 bool read_modbus_response(uint8_t addr, uint16_t *value, int expected_bytes) {
726     uint8_t response[BUF_SIZE];
727     int len = uart_read_bytes(UART_PORT_NUM, response, BUF_SIZE, pdMS_TO_TICKS(RD_TIMEOUT_MS));
728
729     if (len < expected_bytes) {
730         ESP_LOGE(TAG_1, "Нет ответа от датчика или ответ слишком короткий");
731         return false;
732     }
733
734     uint16_t crc = modbus_crc(response, len - 2);
735     if ((response[len-2] != (crc & 0xFF)) || (response[len-1] != (crc >> 8))) {
736         ESP_LOGE(TAG_1, "Ошибка CRC");
737         return false;
738     }
739
740     if (response[0] != addr || (response[1] & 0x80)) {
741         ESP_LOGE(TAG_1, "Некорректный ответ");
742         return false;
743     }
744
745     *value = (response[3] << 8) | response[4];
746     return true;
747 }

```

Рисунок 46 – Функция получения ответа MODBUS

Далее рассмотрим код сохранения данных в БД на рисунках 47–48.

```
142 def save_to_db(data, photo_path=None):
143     """Сохранение нормализованных данных в SQLite"""
144     try:
145         normalized = normalize_json_data(data)
146         if not normalized:
147             return False
148
149         conn = sqlite3.connect(DB_PATH)
150         cursor = conn.cursor()
151
152         cursor.execute('''
153             INSERT OR IGNORE INTO sensor_data (
154                 timestamp, water_level_sensors, flow_sensor, flow_presence,
155                 tds_sensor, ovp_sensor, ph_sensor, water_temp,
156                 turbidity_sensor, oxygen_sensor, photo_path
157             ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
158         ''', (
159             normalized["Время >>> "],
160             json.dumps(normalized["Датчики уровня воды"]),
161             normalized["Датчик потока"],
162             normalized["Датчик наличия потока"].lower(),
163             normalized["TDS датчик"],
164             normalized["OVP датчик"],
165             normalized["PH датчик"],
166             normalized["Температура воды"],
167             normalized["Датчик мутности"],
168             normalized["Датчик кислорода"],
169             photo_path
170         ))
171
172         conn.commit()
173         logging.info(f"Данные сохранены: {normalized['Время >>> ']}")
174         return True
175     except sqlite3.IntegrityError:
```

Рисунок 47 – Функция сохранения данных в БД(1)

```
171
172         conn.commit()
173         logging.info(f"Данные сохранены: {normalized['Время >>> ']}")
174         return True
175     except sqlite3.IntegrityError:
176         logging.warning(f"Данные за {normalized['Время >>> ']} уже существуют в БД")
177         return True
178     except Exception as e:
179         logging.error(f"Ошибка при сохранении в БД: {e}")
180         return False
181     finally:
182         if conn:
183             conn.close()
```

Рисунок 48 – Функция сохранения данных в БД(2)

На рисунках 49–51 представим код инициализации окна программы.

```
17 def __init__(self, root):
18     self.root = root
19     self.root.title("Система контроля УЗВ")
20     self.root.geometry("900x600")
21
22     self.db_path = None
23     self.conn = None
24     self.df = None
25     self.current_table = None
26     self.filtered_df = None
27     self.current_fig = None
28     self.current_fig_nep = None
29     self.annot = None
30     self.annot_nep = None
31
32     # Создаем вкладки
33     self.notebook = ttk.Notebook(root)
34     self.notebook.pack(fill='both', expand=True)
35
36     # Вкладка для выбора файла
37     self.file_tab = ttk.Frame(self.notebook)
38     self.notebook.add(self.file_tab, text="База данных")
39     self.create_file_tab()
40
41     # Вкладка данных
42     self.data_tab = ttk.Frame(self.notebook)
43     self.notebook.add(self.data_tab, text="Данные")
44
45     # Вкладка графиков
46     self.graphs_tab = ttk.Notebook(self.notebook)
47     self.notebook.add(self.graphs_tab, text="Параметрические датчики")
48
```

Рисунок 49 – Функция инициализации окна программы (1)


```

44
45     # Вкладка графиков
46     self.graphs_tab = ttk.Notebook(self.notebook)
47     self.notebook.add(self.graphs_tab, text="Параметрические датчики")
48
49     # Вкладка непараметрических графиков
50     self.graphs_tab_nep = ttk.Notebook(self.notebook)
51     self.notebook.add(self.graphs_tab_nep, text="Непараметрические датчики")
52
53     # Вкладка для просмотра STL файлов
54     self.stl_tab = ttk.Frame(self.notebook)
55     self.notebook.add(self.stl_tab, text="3D УЗВ")
56     self.create_stl_tab()
57
58     # Вкладка для просмотра фотографий
59     self.photos_tab = ttk.Frame(self.notebook)
60     self.notebook.add(self.photos_tab, text="Фото")
61     self.create_photos_tab()
62
63     # Подразделы вкладки графиков
64     self.single_date_frame = ttk.Frame(self.graphs_tab)
65     self.date_range_frame = ttk.Frame(self.graphs_tab)
66     self.compare_dates_frame = ttk.Frame(self.graphs_tab)
67
68     self.graphs_tab.add(self.single_date_frame, text="За дату")
69     self.graphs_tab.add(self.date_range_frame, text="За период")
70     self.graphs_tab.add(self.compare_dates_frame, text="Сравнение дат")
71
72     # Подразделы вкладки непараметрических датчиков
73     self.single_date_frame_nep = ttk.Frame(self.graphs_tab_nep)
74     self.date_range_frame_nep = ttk.Frame(self.graphs_tab_nep)
75     self.compare_dates_frame_nep = ttk.Frame(self.graphs_tab_nep)
76

```

Рисунок 50 – Функция инициализации окна программы (2)

```

66         self.compare_dates_frame = ttk.Frame(self.graphs_tab)
67
68         self.graphs_tab.add(self.single_date_frame, text="За дату")
69         self.graphs_tab.add(self.date_range_frame, text="За период")
70         self.graphs_tab.add(self.compare_dates_frame, text="Сравнение дат")
71
72         # Подразделы вкладки непараметрических датчиков
73         self.single_date_frame_nep = ttk.Frame(self.graphs_tab_nep)
74         self.date_range_frame_nep = ttk.Frame(self.graphs_tab_nep)
75         self.compare_dates_frame_nep = ttk.Frame(self.graphs_tab_nep)
76
77         self.graphs_tab_nep.add(self.single_date_frame_nep, text="За дату")
78         self.graphs_tab_nep.add(self.date_range_frame_nep, text="За период")
79         self.graphs_tab_nep.add(self.compare_dates_frame_nep, text="Сравнение дат")
80
81         # Биндим событие переключения вкладок
82         self.notebook.bind("<<NotebookTabChanged>>", self.on_tab_changed)
83
84         # Переменные для управления 3D окном
85         self.plotter = None
86         self.mesh = None
87
88         # Переменные для управления фотографиями
89         self.photo_labels = []
90         self.photo_images = []
91         self.current_photos = []
92
93

```

Рисунок 51 – Функция инициализации окна программы (3)

После того, как были рассмотрены аппаратное подключение устройств, рассмотрено программирование устройства и принципы работы программы на стороне отправителя и получателя, перейдём к проверке созданного устройства.

4 Тесты

Готовое устройство должно пройти тест основных функций. Проверим работу системы.

У некоторых датчиков есть светодиод. По зажжённому светодиоду убедимся, что датчики работают на рисунках 52 и 53. В качестве проверки работы дисплея, выведем на экран значения полученные с датчиков.



Рисунок 52 – Визуализация подключения датчиков

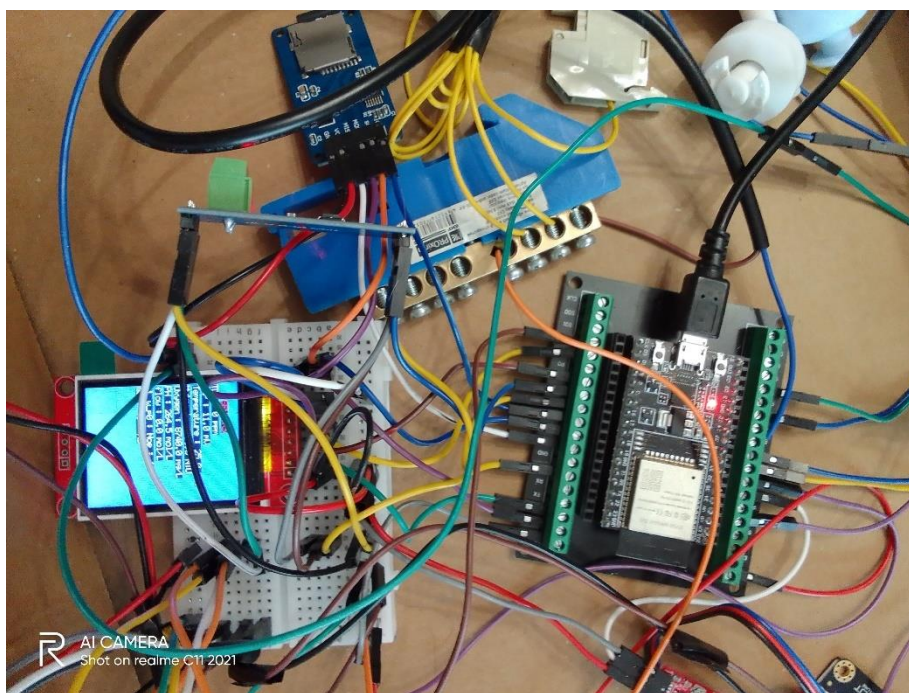
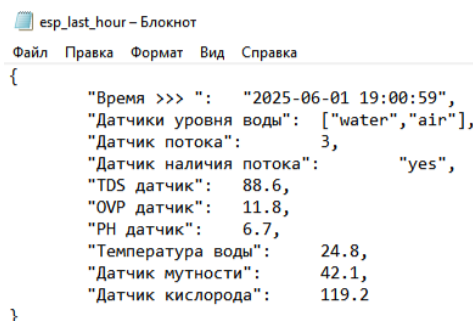


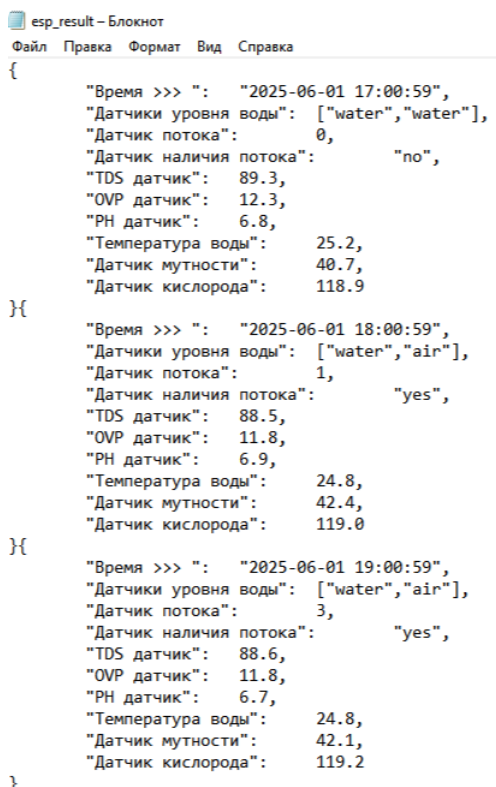
Рисунок 53 – Визуализация подключения sd-карты и TFT-дисплея

Для подтверждения работы sd-карты сверим данные, записанные в файл, с результатами, полученными на экране дисплея. На рисунках 54 и 55 проверим результаты, сохранённые в файл за последний час и результаты, полученные за всё время работы системы.



```
esp_last_hour - Блокнот
Файл  Правка  Формат  Вид  Справка
{
    "Время >>> ": "2025-06-01 19:00:59",
    "Датчики уровня воды": ["water", "air"],
    "Датчик потока": 3,
    "Датчик наличия потока": "yes",
    "TDS датчик": 88.6,
    "OVP датчик": 11.8,
    "PH датчик": 6.7,
    "Температура воды": 24.8,
    "Датчик мутности": 42.1,
    "Датчик кислорода": 119.2
}
```

Рисунок 54 – Сохранение в файл за последний час



```
esp_result - Блокнот
Файл  Правка  Формат  Вид  Справка
[
    {
        "Время >>> ": "2025-06-01 17:00:59",
        "Датчики уровня воды": ["water", "water"],
        "Датчик потока": 0,
        "Датчик наличия потока": "no",
        "TDS датчик": 89.3,
        "OVP датчик": 12.3,
        "PH датчик": 6.8,
        "Температура воды": 25.2,
        "Датчик мутности": 40.7,
        "Датчик кислорода": 118.9
    },
    {
        "Время >>> ": "2025-06-01 18:00:59",
        "Датчики уровня воды": ["water", "air"],
        "Датчик потока": 1,
        "Датчик наличия потока": "yes",
        "TDS датчик": 88.5,
        "OVP датчик": 11.8,
        "PH датчик": 6.9,
        "Температура воды": 24.8,
        "Датчик мутности": 42.4,
        "Датчик кислорода": 119.0
    },
    {
        "Время >>> ": "2025-06-01 19:00:59",
        "Датчики уровня воды": ["water", "air"],
        "Датчик потока": 3,
        "Датчик наличия потока": "yes",
        "TDS датчик": 88.6,
        "OVP датчик": 11.8,
        "PH датчик": 6.7,
        "Температура воды": 24.8,
        "Датчик мутности": 42.1,
        "Датчик кислорода": 119.2
    }
]
```

Рисунок 55 – Сохранение в файл за всё время

Чтобы убедиться, что камера делает снимки, откроем на sd-карте папку с фото и посмотрим на наличие фотографии в папке. Результат представим на рисунках 56 и 57.



Рисунок 56 – Фото с ESP32-CAM

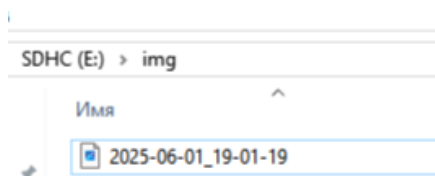


Рисунок 57 – Фото с на sd-карте

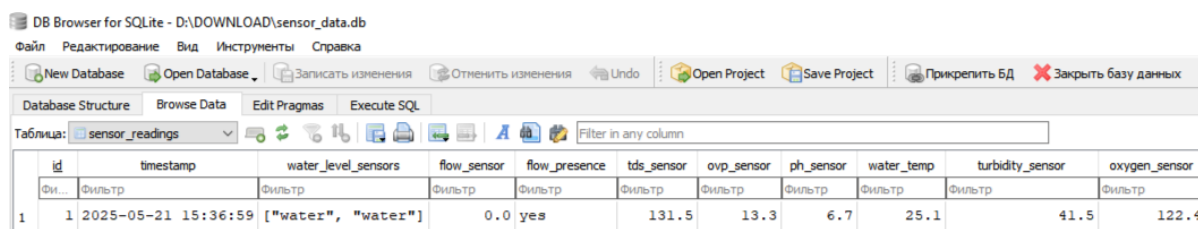
Так как программа должна указывать в данных время считывания показаний с датчиков, то необходимо проверить удачную синхронизацию времени. Проверим это в сохранённых файлах. Видим, что данные в файлах и терминале корректны.

Проверим отправку сообщений на почту. Сообщение отправляется, а файлы успешно сохраняются на ПК. Проверим это при помощи логов на рисунке 58.

```
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:30:28
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:31:46
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:33:02
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:33:32
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:34:02
INFO - Обработано письмо: Test from ESP32
INFO - Данные сохранены: 2025-06-13 18:34:32
```

Рисунок 58 – Проверка сохранения данных

В случае отсутствия базы данных на компьютере пользователя создаётся новая база данных, куда записываются полученные результаты. На рисунке 59 представлено создание БД.



DB Browser for SQLite - D:\DOWNLOAD\sensor_data.db

Файл Редактирование Вид Инструменты Справка

New Database Open Database Записать изменения Отменить изменения Undo Open Project Save Project Прикрепить БД Закрывать базу данных

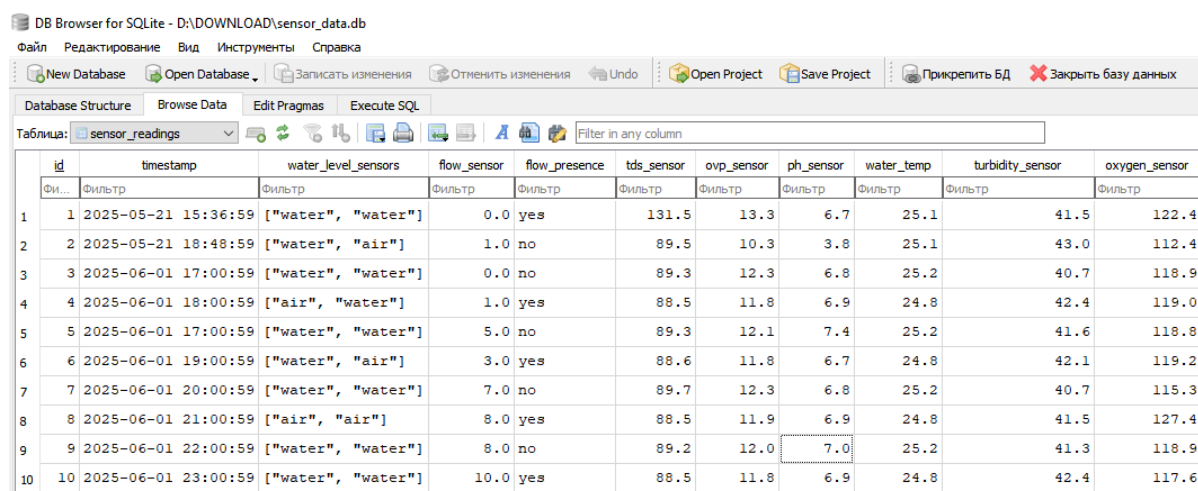
Database Structure Browse Data Edit Pragma Execute SQL

Таблица: sensor_readings Filter in any column

	id	timestamp	water_level_sensors	flow_sensor	flow_presence	tds_sensor	ovp_sensor	ph_sensor	water_temp	turbidity_sensor	oxygen_sensor
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	2025-05-21 15:36:59	["water", "water"]	0.0	yes	131.5	13.3	6.7	25.1	41.5	122.4

Рисунок 59 – Создание БД

Если база данных существует, то данные записываются в конец. Проверим результат на рисунке 60, записав в базу данных значения, полученные после того, как в базе данных уже были какие-то значения.



DB Browser for SQLite - D:\DOWNLOAD\sensor_data.db

Файл Редактирование Вид Инструменты Справка

New Database Open Database Записать изменения Отменить изменения Undo Open Project Save Project Прикрепить БД Закрывать базу данных

Database Structure Browse Data Edit Pragma Execute SQL

Таблица: sensor_readings Filter in any column

	id	timestamp	water_level_sensors	flow_sensor	flow_presence	tds_sensor	ovp_sensor	ph_sensor	water_temp	turbidity_sensor	oxygen_sensor
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	2025-05-21 15:36:59	["water", "water"]	0.0	yes	131.5	13.3	6.7	25.1	41.5	122.4
2	2	2025-05-21 18:48:59	["water", "air"]	1.0	no	89.5	10.3	3.8	25.1	43.0	112.4
3	3	2025-06-01 17:00:59	["water", "water"]	0.0	no	89.3	12.3	6.8	25.2	40.7	118.9
4	4	2025-06-01 18:00:59	["air", "water"]	1.0	yes	88.5	11.8	6.9	24.8	42.4	119.0
5	5	2025-06-01 17:00:59	["water", "water"]	5.0	no	89.3	12.1	7.4	25.2	41.6	118.8
6	6	2025-06-01 19:00:59	["water", "air"]	3.0	yes	88.6	11.8	6.7	24.8	42.1	119.2
7	7	2025-06-01 20:00:59	["water", "water"]	7.0	no	89.7	12.3	6.8	25.2	40.7	115.3
8	8	2025-06-01 21:00:59	["air", "air"]	8.0	yes	88.5	11.9	6.9	24.8	41.5	127.4
9	9	2025-06-01 22:00:59	["water", "water"]	8.0	no	89.2	12.0	7.0	25.2	41.3	118.9
10	10	2025-06-01 23:00:59	["water", "water"]	10.0	yes	88.5	11.8	6.9	24.8	42.4	117.6

Рисунок 60 – Создание БД

Программа должна работать циклично. Проверим на удачное получение результатов при повторном считывании данных с датчиков. Видим, что данные удачно считываются и сохраняются в БД.

Вывод данных из БД и построение графиков представлены на рисунках 61-64. На рисунке 61 представим данные БД, на рисунке 62-63 представим графики данных на рисунке 64 представлена, спроектированная в компасе 3d модель.

Система контроля УЗВ													
База данных: Данные: Параметрические датчики: Непараметрические датчики: 3D УЗВ: Фото													
Выборите период:													
<div> <div> <div>2025-06-10</div> <div>Пн</div> <div>2025-06-15</div> <div>Обновить данные</div> </div> </div>													
date	time	id	water_level_sensors	flow_sensor	flow_presence	tds_sensor	ovp_sensor	ph_sensor	water_temp	turbidity_sensor	oxygen_sensor		
2025-06-10	19:00:59	1	["water", "air"]	3.0	yes	138.8	88.6	7.1	24.8	15.3	2.4		
2025-06-10	20:00:59	2	["water", "air"]	3.0	yes	138.1	88.5	7.0	24.8	16.1	2.4		
2025-06-10	21:00:59	3	["air", "air"]	3.2	no	137.9	88.2	7.1	24.7	16.2	2.5		
2025-06-11	12:00:30	4	["air", "water"]	7.0	yes	137.4	88.2	7.1	18.8	17.1	3.1		
2025-06-11	13:00:31	5	["water", "air"]	4.5	yes	148.1	88.3	7.1	19.4	17.2	3.2		
2025-06-11	14:00:31	6	["water", "air"]	4.5	yes	149.4	89.1	7.2	23.2	17.2	3.2		
2025-06-13	15:00:31	7	["water", "air"]	4.6	yes	150.1	89.2	7.2	23.3	17.1	3.1		
2025-06-13	16:00:32	8	["water", "air"]	4.4	yes	147.2	89.3	7.2	23.2	17.2	3.2		
2025-06-13	17:00:32	9	["water", "air"]	4.0	no	148.9	89.6	7.2	23.1	17.2	3.3		
2025-06-13	18:00:32	10	["water", "air"]	3.7	no	148.3	102.8	7.2	23.0	17.0	3.3		
2025-06-14	11:47:13	11	["water", "water"]	3.4	yes	237.6	103.1	7.1	17.6	16.5	3.2		
2025-06-14	12:47:13	12	["water", "water"]	3.7	yes	239.0	103.7	7.1	17.9	16.4	3.3		
2025-06-14	13:47:13	13	["water", "air"]	3.6	yes	238.4	105.8	7.1	18.3	16.2	3.2		
2025-06-14	14:47:14	14	["water", "air"]	3.9	yes	241.1	104.6	7.1	18.9	15.9	3.1		
2025-06-14	15:47:14	15	["water", "air"]	4.1	yes	235.6	104.5	7.1	20.1	16.0	3.1		
2025-06-14	16:47:14	16	["air", "air"]	4.2	yes	238.2	108.2	7.1	20.2	15.9	3.1		
2025-06-14	17:47:14	17	["air", "air"]	4.2	no	240.8	107.7	7.1	20.4	16.0	3.0		
2025-06-14	18:47:14	18	["air", "air"]	4.1	no	237.3	107.1	7.1	19.9	15.9	3.7		
2025-06-15	12:09:23	19	["water", "air"]	5.5	yes	212.3	106.9	7.4	17.7	15.8	3.6		
2025-06-15	13:09:23	20	["water", "air"]	5.6	yes	213.7	108.4	7.4	18.6	15.8	3.7		
2025-06-15	14:09:23	21	["water", "air"]	4.8	yes	211.6	108.5	7.4	19.4	15.6	3.6		
2025-06-15	15:09:23	22	["water", "air"]	5.0	yes	214.2	108.2	7.4	20.1	15.5	3.5		
2025-06-15	16:09:23	23	["water", "air"]	5.2	yes	215.8	107.9	7.4	20.7	15.4	3.5		
2025-06-15	17:09:24	24	["water", "air"]	5.1	yes	212.2	107.4	7.4	21.4	15.4	3.4		
2025-06-15	18:09:24	25	["water", "air"]	5.0	no	214.2	107.8	7.3	21.1	15.2	3.4		
2025-06-15	19:09:24	26	["water", "air"]	5.0	no	213.5	108.0	7.4	20.8	15.1	3.3		

Рисунок 61 – Данные БД

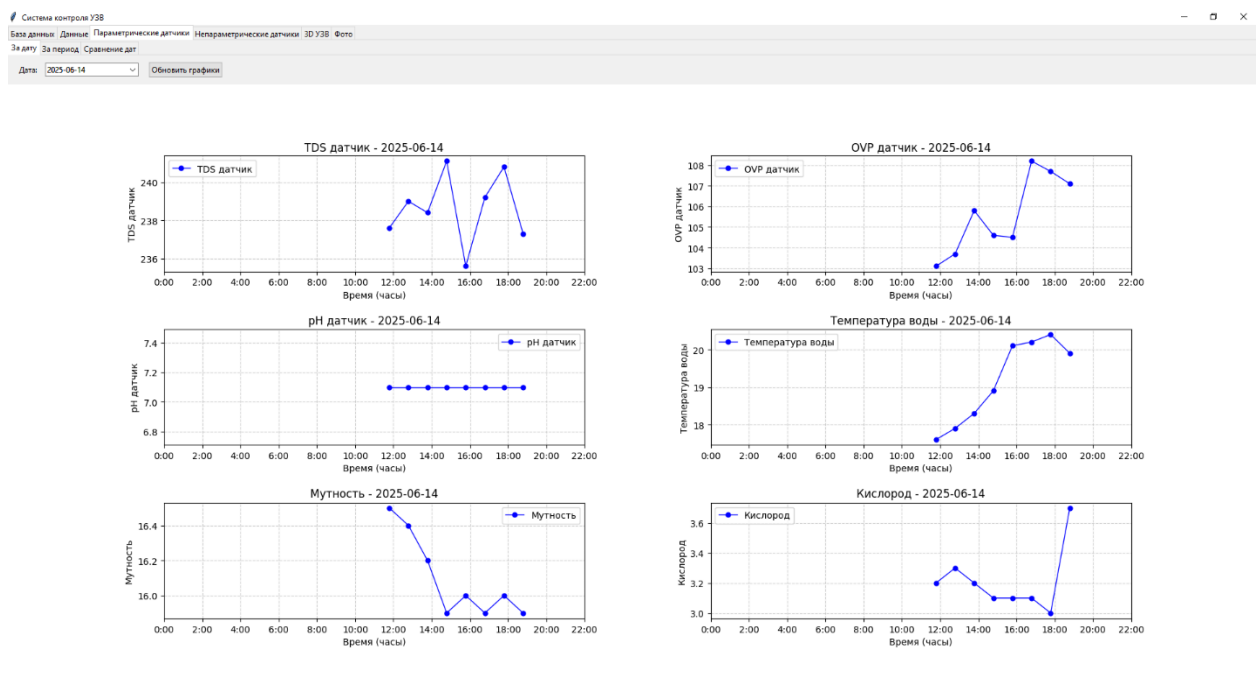


Рисунок 62 – Графики параметрических датчиков за определённую дату

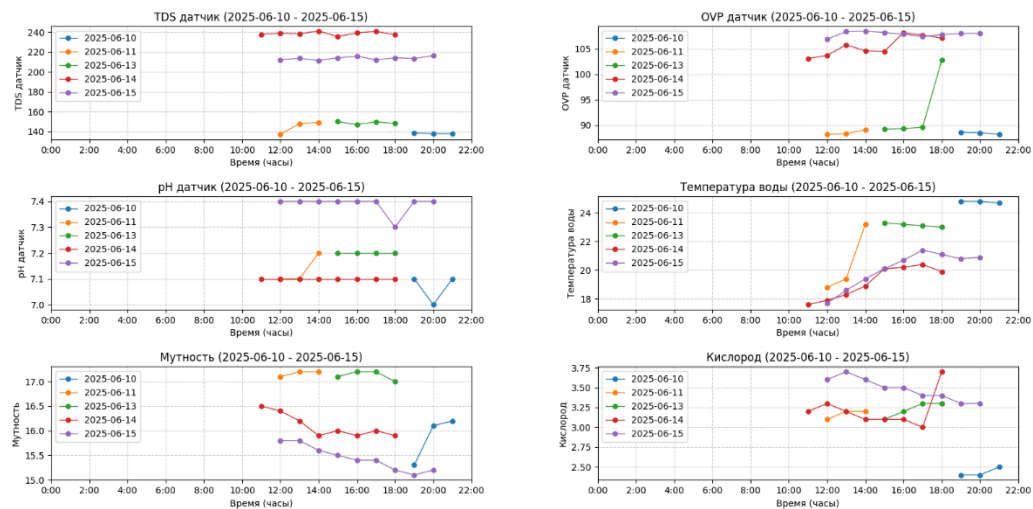
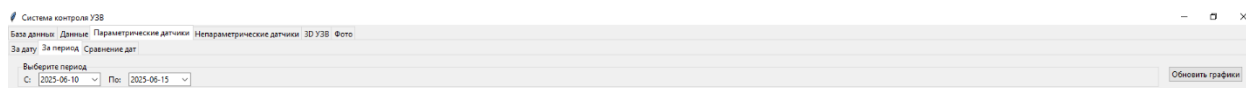


Рисунок 63 – Графики параметрических датчиков за период

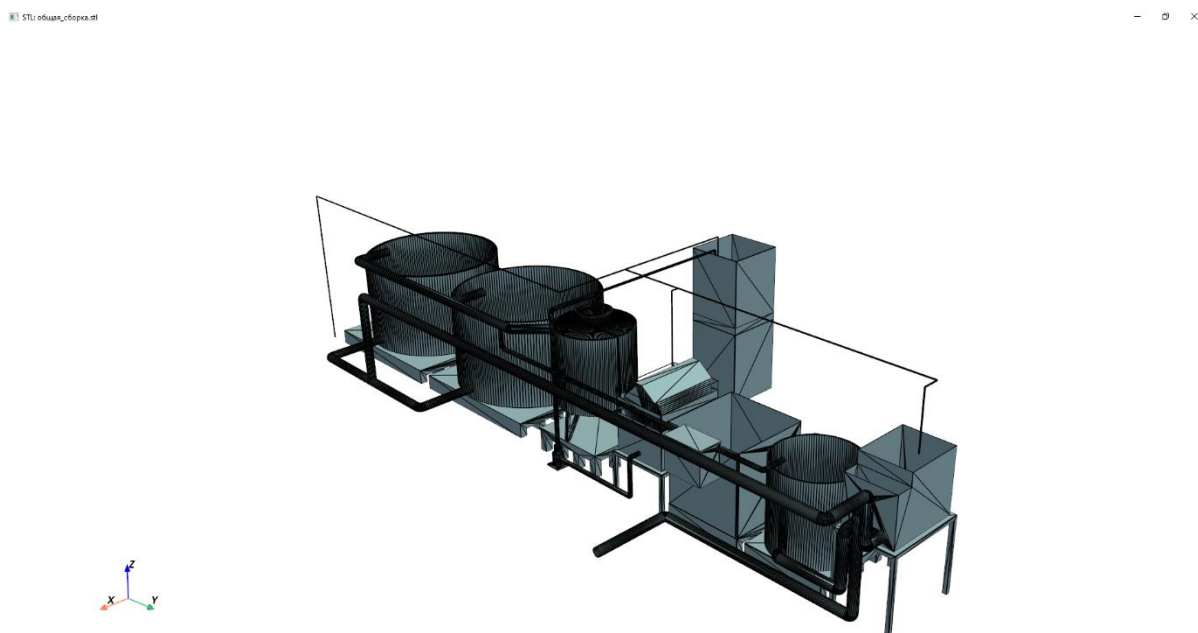


Рисунок 64 – Схема УЗВ

5 Обоснование экономической эффективности инновационного проекта

Экономическую эффективность проекта будем рассматривать с точки зрения возможностей возврата инвестиций, а также доходности при дальнейшей эксплуатации.

Для оценки эффективности произведём расчёт затрат на производство, определим аналоги и товарный тип СК УЗВ. После чего определим цену разработки СК УЗВ, проведём анализ безубыточности и выясним экономическую эффективность при помощи статистических методов. И в заключение рассчитаем динамические показатели эффективности и составим финансовый план проекта.

5.1 Расчёт сметы затрат на производство

В сметную стоимость затрат будут входить затраты на материалы, затраты на энергию, зарплата персоналу, взносы в фонды, амортизационные отчисления, накладные расходы и прочие расходы.

5.1.1 Расчёт затрат на основные и вспомогательные материалы, покупные изделия и полуфабрикаты

Данные затраты представим в таблице 11, для расчёта умножив нормы расхода на цену ресурсов, прибавим к полученному значению транспортно-заготовительные расходы и вычтем стоимость возвратных отходов. Так как данные товары заказывают оптом и с доставкой с маркетплейса, то итоговая цена с учетом транспортно-заготовительных затрат будет включать доставку 1000 рублей.

Таблица 11 – Материалы

№ п./п	Наименования материалов, покупных изделий и полуфабрикатов	Ед. Измерения	Цена единицы (руб.)	Кол-во (ед. измер.)	Сумма (руб.)	Итого затрат с учётом трансп.-загот. расх.(руб.)
1	Датчики уровня воды	шт	100	2	200	
2	Датчик потока воды	шт	500	1	500	
3	Датчик наличия потока	шт	400	1	400	
4	TDS датчик	шт	2300	1	2300	
5	OVP датчик	шт	5600	1	5600	
6	РН датчик	шт	5600	1	5600	
7	Датчик мутности	шт	1200	1	1200	
8	Датчик кислорода	шт	17000	1	17000	
9	Микроконтроллер ESP32	шт	500	1	500	
10	sd-карта	шт	200	1	200	
11	sd-card адаптер	шт	100	1	100	
12	TFT-дисплей	шт	500	1	500	
13	ESP32-CAM	шт	2000	1	2000	
14	Регуляторы напряжений	шт	200	3	600	
15	Провода соединительные	уп(20шт)	50	3	150	
16	Клеммы	уп(10шт)	600	3	1800	
	ИТОГО				38200	39200

Итоговая стоимость затрат, включая транспортно-заготовительных затраты, составила 39200 рублей.

5.1.2 Расчёт затрат на топливо и энергию для технологических целей

Затраты на электроэнергию определим по формуле (1).

$$Z_{эл\ эн} = \sum_{i=1}^n W_i T_i C K_w, \quad (1)$$

где n – количество используемого оборудования, W_i – мощность оборудования, T_i – время работы, C – стоимость одного кВт электроэнергии, K_w – количество используемого оборудования.

Для создания системы использовались ноутбуки программиста и руководителя проекта, рассчитаем затраты на электроэнергию за 53 рабочих дня каждого ноутбука, исходя из нормы рабочих дней равной 8 часов. Так как

ноутбуки одинаковые, то и мощность у них одинаковая, значит формула (1) примет вид, соответствующий формуле (2):

$$Z_{\text{эл эн}} = \sum_{i=1}^n W_i T_i CK_W = W_{\text{ноутбуков}} T_{\text{ноутбука}} CK_W, \quad (2)$$

Рассчитаем мощность ноутбуков в формуле (3) и время работы в формуле (4).

$$W_{\text{ноутбуков}} = N W_{i \text{ ноутбука}} = 2 \times 0,15 = 0,3 \text{ кВт} \quad (3)$$

$$T_{\text{ноутбука}} = \frac{T_{\text{ноутбуков}}}{2} = \frac{106 \text{ дней}}{2} = 53 \text{ дней} = 53 \times 8 \text{ часов} = 424 \text{ часа} \quad (4)$$

Стоимость одного кВт электроэнергии в Санкт-Петербурге составляет 6,19 рублей при использовании 0 – 6400 кВт × Ч. Поэтому С составит $6,19 \frac{\text{руб}}{\text{кВт} \times \text{Ч}}$.

Рассчитаем коэффициент использования мощности в формуле (5). Программы, используемые в проекте, могут быть открыты одновременно, что повышает коэффициент. Поэтому примем его равным 75%.

$$K_w = 75\% = 0,75 \quad (5)$$

Теперь подставим значения в формулу (2) и получим результат в формуле (6):

$$Z_{\text{эл эн}} = 0,3 \text{ кВт} \times 424 \text{ ч} \times 6,19 \left(\frac{\text{руб}}{\text{кВт} \times \text{ч}} \right) \times 0,75 = 591 \text{ рубль} \quad (6)$$

5.1.3 Расчёт затрат на заработную плату персонала

Основную заработную плату будем рассчитывать по повременной расценке и результаты расчёта представим в таблице 12.

Таблица 12 – Заработная плата персоналу

п/п	Наименование этапа разработки	Исполнитель этапа разработки	Кол-во чел.	Длительность этапа (рабочие дни)	Зарплата, руб/м	Затраты на этап
1	Анализ технического задания	Руководитель проекта	1	3	200000	20000
		Программист	1	3	100000	10000
		Всего для этапа 1	2	6		30000

Продолжение таблицы 12

п/п	Наименование этапа разработки	Исполнитель этапа разработки	Кол-во чел.	Длительность этапа (рабочие дни)	Зарплата, руб/м	Затраты на этап
2	Моделирование макета УЗВ	Программист	1	5	100000	17000
		Всего для этапа 2	1	5	100000	17000
3	Проектирование СК УЗВ	Руководитель проекта	1	10	200000	67000
		Программист	1	5	100000	17000
		Всего для этапа 3	2	15		84000
4	Программирование устройства	Программист	1	30	100000	100000
		Всего для этапа 4	1	30		100000
5	Тестирование работы устройства	Руководитель проекта	1	40	200000	266000
		Программист	1	10	100000	33000
		Всего для этапа 5	2	50		299000
	Итого			106		530000

При выполнении задачи за 106 рабочих дней, общие затраты сотрудникам на заработную плату составят 530000 рублей. Дополнительная заработная плата составит 10% от основной и равна 53000 рублей.

Затраты на заработную плату персонала составят 583000 рублей.

5.1.4 Страховые взносы во внебюджетные фонды

Компания является субъектом среднего бизнеса, но не является IT-компанией и не финансируется из федерального бюджета, поэтому с 1 января 2025 года выплаты составляют: 30% с выплат в пределах 1,5МРОТ и 15% выплат свыше 1,5МРОТ до достижения предельной базы. МРОТ в Санкт-Петербурге с 1 января 2025 года составляет 28750 рублей. Установим, что сотрудники работали 2 месяца. Значит МРОТ каждого из них составит 57500 рублей, тогда выплаты в пределах 1,5МРОТ составят 86250 рублей. Из предыдущего пункта рассчитаем зарплату сотрудников: 353000 рублей для руководителя проекта, 177000 для программиста. Тогда суммарная выплата страховых взносов рассчитывается в формуле (8) составит:

$$\begin{aligned} 2 \times 86250 \times 0,3 + (353000 - 86250) \times 0,15 + (177000 - 86350) \times 0,15 = \\ = 105360 \text{ рублей} \end{aligned} \quad (7)$$

5.1.5 Расчёт амортизационных отчислений

Расчёт амортизационных отчислений представлен в таблице 13. Норму амортизации будем высчитывать по формуле (8).

$$\text{Норма амортизации} = \frac{100\%}{\text{Срок использования (N месяцев)}} \quad (8)$$

К фондам отнесём вычислительное оборудование и компьютерную технику. Срок использования вычислительного оборудования составит 5 лет, а компьютерной техники - 10 лет.

К нематериальным активам отнесём программное обеспечение и разработанную базу данных. Их срок использования составляет 10 лет. Так как нематериальные активы не превышают 100000 рублей, то амортизационные отчисления по нематериальным активам учитываться не будут.

Таблица 13 – Амортизационные отчисления

Наименование оборудования	Цена единицы оборудования	Количество оборудования	Норма амортизации	Годовая сумма амортизационных отчислений	Итого амортизация по оборудованию
1.Основные фонды					
Вычислительная техника	39200	1	20%	7840	7840
Компьютерная техника	50000	2	10%	5000	10000
1.Нематериальные активы					
Нематериальные активы отсутствуют					
Итого	89200				20840

Итого, без учёта нематериальных активов, амортизационные взносы составят 17840 рублей.

5.1.6 Расчёт затрат на контрагентские работы, командировки и прочие затраты

Затрат на контрагентские работы и командировки не предусмотрено, но в качестве прочих затрат выделим дополнительные консультации руководителя проекта программисту (студенту). Затраты на одну консультацию составляют 1000 рублей. Было проведено 5 консультаций. Данные расходы составят 5000 рублей.

К прочим затратам также относится аренда помещения, которая составит 15000 рублей. Тогда итоговые затраты составят 20000 рублей.

5.1.7 Расчёт накладных расходов

Накладные расходы составят 70% от зарплаты 530000 рублей и будут равны 371000 рублей.

Определим общую смету и подытожим затраты на производство в таблице 14.

Таблица 14 – Общие затраты на производство

№	Статьи расходов	Затраты (руб.)	Доля
1	Затраты на основные и вспомогательные материалы, покупные изделия и полуфабрикаты	39200	4%
2	Затраты на топливо и энергию для технологических целей	591	<1%
3	Затраты на заработную плату персонала	583000	50%
4	Страховые взносы во внебюджетные фонды	105360	10%
5	Амортизационные отчисления	17849	1%
6	Затраты на контрагентские работы, командировки и прочие выплаты	20000	2%
7	Накладные расходы	371000	33%
	Итого	1137000	100%

Данные расходы нужны при создании СК, далее данное устройство продаётся в промышленных объёмах. Теперь рассчитаем затраты при изготовлении устройства в промышленных объёмах.

Пусть компания производит 300 устройств в год. Тогда затраты на основные и вспомогательные материалы, покупные изделия и полуфабрикаты составят 11760000 рублей.

Для реализации цепочки изготовления нужны 2 человека: консультант, инженер, зарплата которых составляет 60000 рублей в месяц. Тогда их общая зарплата за год составит 1440000 рублей.

Страховые взносы за год будут рассчитаны по формуле (9).

$$28750 \times 12 \times 2 \times 0.3 + (2160000 - 28750 \times 12 \times 2) \times 0.15 =$$

$$= 319500 \frac{\text{руб}}{\text{год}} \quad (7)$$

Амортизационные отчисления состоят только из 3х компьютеров и составляют 15000 рублей в год.

Аренда помещения составляет 120000 рублей в год

Накладные расходы будут равны 70% и составляют рублей в год 1008000.

5.2 Определение аналогов объекта разработки, определение товарного типа товара-разработки

Экономическая значимость объекта разработки состоит из потребительских выгод и выгод для производителя.

Потребительские выгоды:

- автоматизация контроля параметров воды повысит точность измерений параметров воды и ускорит рост рыбы,
- удалённый контроль УЗВ через Wi-Fi позволяет быстро реагировать на изменения в УЗВ рыбной фермы, избегая потери.

Выгоды для производителя:

- снижение себестоимости из-за использования недорогих компонент,
- преимущество перед конкурентами за счёт интеграции IoT,
- расширение рынка сбыта, так как сейчас активно идёт развитие рыбных хозяйств, из-за чего увеличивается количество потенциальных потребителей.

Проведём анализ аналогов и сравним созданное устройство с продукцией других компаний. Выделим несколько компаний – конкурентов.

Голландская компания AKVA Group – компания обладает широким функционалом, но продукция дорогая и избыточная для малых ферм.

Американская Pentair Aquaculture – высокая цена продукции, закрытое ПО.

Британская компания Seneye – отсутствует поддержка ESP32 и компания предоставляет ограниченный набор датчиков.

К особенностям разработанного устройства можно отнести:

- ценовую доступность: сама система программируется на недорогом, но функциональном микроконтроллере,
- функциональность: есть возможность подключать любые датчики, а также увеличивать количество датчиков при необходимости,

- открытое ПО: возможность пользователю вносить изменения в код при необходимости,
- локализация: поддержка русского языка.

Коммерческая цель СК УЗВ состоит в продаже аквакультурным предприятиям. При этом модель монетизации может являться розничной и оптовой торговлей, поддержкой и проектированием СК УЗВ.

5.3 Определение цены объекта разработки. Анализ безубыточности производства. Определение экономической эффективности статистическими методами

Для определения цены СК УЗВ рассмотрим несколько методов. Рассмотрим их.

5.3.1 Метод расчёта цены учётом пользовательского эффекта

Данный метод основан на выгоде, которую потребитель может получить, если будет использовать данный продукт, а цена товара рассчитывается по формуле (10).

$$P_{\text{новая}} = P_{\text{аналога}} + \Delta_{\text{потребителя}} \times K_{\text{торм}}, \quad (10)$$

где $P_{\text{новая}}$ – цена нового товара, $P_{\text{аналога}}$ – цена аналогичного товара, средняя цена аналогов 70000 рублей, $\Delta_{\text{потребителя}}$ эффект потребителя от замены старого товара на новый, составляет 6000 рублей в год, $K_{\text{торм}}$ – коэффициент торможения при переключении на новый товар, так как продукт новый, то коэффициент торможения возьмём 0,3. Подставим данные в формулу (10) и в формуле (11) получим результат.

$$P_{\text{новая}} = 70000 + 6000 \times 0,3 = 71800 \text{ рублей} \quad (11)$$

Отличительной особенностью устройства является меньшее энергопотребление, так как система в неактивное время находится в режиме “сна”.

5.3.2 Метод анализа коридора цен

При использовании данного метода рассматриваются верхнее и нижнее значения цен у товаров конкурентов, а цена назначается с учетом характеристик товара. Клиентам важно высокое качество и доступная цена, поэтому будем ориентироваться на среднюю цену, при хорошем качестве. Цена одной установки не должна превышать 80000 рублей.

Определение нормативной цены рассчитаем по формуле (12):

$$C_{\text{норм}} = C_n \times \left(1 + \frac{P_n}{100}\right), \quad (12)$$

где C_n – себестоимость изделия, P_n – нормативный уровень рентабельности.

Найдём себестоимость в формуле (12) одной СК УЗВ, при условиях, которые были описаны выше.

$$C = \frac{11760000 + 1440000 + 319500 + 15000 + 120000 + 1008000}{300} = 48875 \text{ рублей} \quad (13)$$

Уровень рентабельности примем равным 30% и рассчитаем по формуле (14):

$$C_{\text{норм}} = 48875 \times 1,3 = 63540 \text{ рублей} \quad (14)$$

Получаем, что цена товара не может быть меньше, чем 48875 рублей, чтобы товар продавался не дешевле, чем стоит его производство. Для желаемой рентабельности цена товара должна составлять 63540 рублей, цена является приемлемой и может достигать 71800.

5.3.3 Анализ безубыточности производства

Для проведения анализа по формуле (14) производится расчёт объёма производства, который обеспечивает безубыточную работы организации, то есть при таком объёме компания не будет получать прибыль, но будет полностью покрывать свои расходы на производство:

$$D_{\text{кр}} = \frac{\text{ПЗ}}{\text{Ц}_{\text{норм}} - \text{ПРЗ}}, \quad (14)$$

где $D_{\text{кр}}$ – критический объём товара, $\text{Ц}_{\text{норм}}$ – нормативная цена товара, ПЗ – постоянные затраты на производство, ПРЗ – переменные затраты на единицу товара. Подставим значения в формулу (14) и в формуле (15) найдём необходимое количество устройств.

$$D_{\text{кр}} = \frac{1440000 + 319500 + 15000 + 120000 + 1008000}{63540 - 48875} = 198 \text{ устройств в год} \quad (15)$$

Также анализ безубыточности позволяет рассчитать показатель запаса прочности – риск получения убытков по формуле (16). Для стабильного получения прибыли необходимо иметь высокий запас прочности.

$$K = \frac{Q - X}{Q} \times 100\%, \quad (16)$$

где K – запас прочности, Q – объём планируемой продукции, X – объём безубыточной реализации продукта. Подставим значения в формулу (16) и вычислим результат в формуле (17).

$$K = \frac{300 - 198}{300} \times 100\% = 34\% \quad (16)$$

5.3.4 Определение экономической эффективности продукта статическими методами

Найдём прибыль проекта по следующим формулам (17).

$$\text{Прибыль} = \text{Выручка} - \text{Затраты} \quad (17)$$

Рассчитаем прибыль без затрат на производство в формуле (18) и с затратами в формуле (19).

$$\begin{aligned} \text{Прибыль} &= 300 \times 63240 - 11760000 - 1440000 - 319500 - 15000 - 120000 - \\ &\quad - 1008000 = 4325000 \text{ рублей} \end{aligned} \quad (18)$$

$$\text{Прибыль с созданием} = 4325000 - 1137000 = 3188000 \text{ рублей} \quad (19)$$

Для определения показателя эффекта определим разность между результатом и затратами по формуле (20).

$$\mathcal{E} = P - Z = 3188000 \text{ рублей} \quad (20)$$

Показатель эффективности находят при помощи отношения результата к затратам. Найдём его по формуле (21).

$$\mathcal{E} = \frac{P}{Z} = \frac{63540 \times 300}{11760000 + 1440000 + 319500 + 120000 + 15000 + 1008000} = 1,2 \quad (21)$$

Экономия, которая возникает в связи с использованием нового товара по сравнению с предыдущим, называется экономией по статьям затрат, но в проекте она отсутствует, так как товар только появился.

Рассчитаем срок окупаемости нашего проекта – срок, за который окупятся первоначальные траты по формуле (22).

$$T_{\text{ок}} = \frac{C_p}{P_p} = \frac{1137000}{4325000} = 3 \text{ месяца} \quad (22)$$

где, C_p – сметная стоимость разработки, P_p – средний планируемый доход за период.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были рассмотрена система контроля УЗВ, принципы работы УЗВ рыбной фермы, а также создано устройство, при помощи которого можно отслеживать ключевые характеристики УЗВ.

В программе КОМПАС 3D была создана модель рыбной фермы, при помощи которой удалось выявить необходимые для отслеживания характеристики воды.

Также в программе KICAD было визуализировано аппаратное подключение компонент системы к микроконтроллеру ESP32-WROOM-32D.

После этого в среде разработки VS CODE на языке СИ было реализовано программное подключение устройств, а также работа устройства с wi-fi и передача данных по протоколу smtp.

При помощи языка программирования python, и баз данных SQLite было реализовано извлечение данных и сохранение данных на ПК диспетчера и в базу данных.

При проверке работоспособности системы, созданная установка прошла все , необходимые для контроля УЗВ, тесты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Vedat Ozan Oner. Developing IoT Projects with ESP32: Unlock the full Potential of ESP32 in IoT development to create production-grade smart devices. – Gamburg. 2023. – 217 с.
2. Furuto Kimiko. Mastering esp32 practical projects with Arduino ide programming. – Osaka. 2024. – 251 с.
3. Полное руководство от Espressif: API, примеры, конфигурация [Электронный ресурс] – URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/> (дата обращения 01.04.2025)
4. Детальное описание функций (Wi-Fi, Bluetooth, GPIO, SPI) [Электронный ресурс] – URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/> (дата обращения 11.04.2025)
5. Официальные видео Espressif [Электронный ресурс] – URL: <https://www.youtube.com/c/EspressifSystems> (дата обращения 13.04.2025)
6. Форум ESP32 [Электронный ресурс] – URL: <https://esp32.com/> (дата обращения 14.04.2025)
7. Сторонние библиотеки [Электронный ресурс] – URL: <https://github.com/topics/esp-idf-component> (дата обращения 18.04.2025)
8. Alex Sweigart. Automate the Boring Stuff with Python. – London. 2019. – 184 с.
9. Eric Matthes. Python Crash Course. Seattle. – 2021. – 159 с.
10. Работа с почтой [Электронный ресурс] – URL: <https://realpython.com/python-send-email/#using-the-gmail-api> (дата обращения 21.04.2025)
11. GitHub рипозиторий с примерами [Электронный ресурс] – URL: <https://github.com/sqlalchemy/sqlalchemy/wiki/> (дата обращения 03.05.2025)