МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Веб-сервер для встраиваемых систем с индикацией динамически изменяющихся данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.405 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

Автор работы,
студентка группы КЭ-405
_____ А.Д. Шарипова
«___»_____2024 г.

Нормоконтролёр,
ст. преподаватель каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра
студентке группы КЭ-405
Шариповой Арине Даниловне
обучающейся по направлению
09.03.01 «Информатика и вычислительная техника»

1.Тема работы: «Веб-сервер для встраиваемых систем с индикацией динамически изменяющихся данных» утверждена приказом по университету от «22» апреля 2024 г. № 764-13/12

2. Срок сдачи студентом законченной работы: 01 июня 2024 г.

3. Исходные данные к работе:

– стандарты программирования на языке программирования JavaScript;

– описание структур данных для обмена на языке C;

– список параметров для обмена с электронным блоком управления.

Технические данные микроконтроллера электронного блока управления:

– микроконтроллер должен иметь достаточную вычислительную мощность для обработки HTTP-запросов и динамического формирования веб-

страниц. Рекомендуется использовать микроконтроллер с тактовой частотой не менее 100 МГц;

– поддержка WiFi 802.11n 2.4 ГГц с максимальной скоростью 150 Мбит/с и Bluetooth 4.2 BLE;

– использование микроконтроллера с не менее 128 КБ Flash-памяти и 32 КБ оперативной памяти;

– поддержка различных типов датчиков, такие как температурные датчики (DS18B20, DHT11, DHT22), датчики влажности;

– различные интерфейсы ввода-вывода, такие как SPI, I2C, UART и другие, для подключения к различным устройствам и датчикам.

4. Перечень подлежащих разработке вопросов:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Разработка архитектуры встроенного веб-сервера.
3. Разработка программного кода.
4. Тестирование и отладка программного кода.

5. Дата выдачи задания: 1 декабря 2023 г.

Руководитель работы _____/ Д.В. Топольский /

Студентка _____/ А.Д. Шарипова /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы	28.02.2023	
Разработка архитектуры встроенного веб-сервера	03.03.2024	
Разработка программного кода	24.03.2024	
Тестирование и отладка программного кода	07.04.2024	
Компоновка текста работы и сдача на нормоконтроль	08.04.2024	
Подготовка презентации и доклада	09.04.2024	

Руководитель работы _____/ Д.В. Топольский /

Студентка _____/ А.Д. Шарипова /

Аннотация

А.Д. Шарипова. Веб-сервер для встраиваемых систем с индикацией динамически изменяющихся данных. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 52 с., 3 табл., 18 ил., библиогр. список – 27 наим.

Выпускная квалификационная работа посвящена разработке веб-сервера для встраиваемых систем с индикацией динамически изменяющихся данных.

В первой главе проведен аналитический обзор технологий для отображения динамически изменяющихся данных с помощью веб-интерфейса, были выбраны необходимые для разработки средства.

Во второй главе разработана архитектура для понимания работы встроенного веб-сервера, который взаимодействует с динамически изменяющимися данными.

Третья глава посвящена разработке веб-интерфейса для отображения данных в реальном времени с использованием конструктора веб-интерфейсов GyverPortal.

В четвертой главе проведено тестирование разработанного кода для создания веб-интерфейса с использованием библиотеки GyverPortal и приведены результаты тестирования.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире веб-технологии играют ключевую роль в повседневной жизни. Они постоянно развиваются и совершенствуются, чтобы обеспечить пользователям наиболее эффективный и удобный доступ к информации. В таких условиях разработка интерфейсов для динамического отображения данных через веб-интерфейс стала одной из ключевых задач, поскольку современное поколение гораздо лучше запоминает информацию с помощью зрительных образов. Визуальное отображение информации гораздо проще и удобнее для восприятия, чем текстовый формат.

Веб-серверы для встраиваемых систем позволяют отображать динамически обновляемые данные в реальном времени через интернет. Эти системы идеально подходят для управления физическими процессами, мониторинга и контроля оборудования, автоматизации зданий и многого другого. Веб-серверы встраиваемых систем компактны, энергоэффективны и могут работать в различных условиях окружающей среды.

Эти системы обычно используют HTTP-протокол для передачи данных, и их можно интегрировать с различными датчиками, инструментами и системами управления. Они также могут быть настроены для обработки различных типов данных, таких как текст, изображения, аудио и видео.

Одним из ключевых преимуществ веб-серверов для встраиваемых систем является их способность обрабатывать большое количество запросов без потери производительности. Это делает их идеальным выбором для приложений, требующих высокой степени масштабируемости и надежности.

В целом, веб-серверы для встраиваемых систем представляют собой мощное и гибкое решение для отображения динамических данных в реальном времени. Они идеально подходят для широкого спектра применений, от мониторинга промышленных процессов до управления домашними устройствами.

Актуальность данной ВКР обусловлена участием в совместном проекте ЮУрГУ и ООО «Уральский инжиниринговый центр» (УрИЦ), реализуемого в рамках Уральского межрегионального научно-образовательного центра мирового

уровня (УМНОЦ), в котором целью является полное импортозамещение конструкции следящего гидропривода, включая все компоненты, в том числе и программное обеспечение к 2025 году. Производство следящих гидроприводов будет развернуто на промплощадке индустриального партнера в городе Челябинске [1-2].

Целью же данной ВКР является разработка программного кода для отображения в реальном времени данных, поступающих от электронного блока управления, с помощью веб-интерфейса.

Задачи ВКР:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Разработка архитектуры встроенного веб-сервера.
3. Разработка программного кода для отображения в реальном времени данных, поступающих от электронного блока управления, с помощью веб-интерфейса.
4. Тестирование и отладка программного кода на микроконтроллере.

1. ОБЗОР ТЕХНОЛОГИЙ ДЛЯ ОТОБРАЖЕНИЯ ДИНАМИЧЕСКИ ИЗМЕНЯЮЩИХСЯ ДАННЫХ С ПОМОЩЬЮ ВЕБ-ИНТЕРФЕЙСА

1.1. Краткая история развития web-технологий

Эволюция интернета обычно делится на три этапа: Web 1.0, Web 2.0 и Web 3.0. Web 1.0 относится к технологиям 90-х и ранних 2000-х годов. Это был период, когда появились первые сайты с ограниченной информацией и функциональностью. Пользователи могли только получать информацию, но не взаимодействовать с ней.

Web 2.0 начался в конце 90-х и продолжается до сих пор. Этот этап характеризуется возможностью взаимодействия пользователей с информацией. Люди получили возможность создавать свой контент, общаться и делиться информацией через социальные сети, вики-сайты и другие платформы. Однако, с развитием Web 2.0 появились и проблемы, такие как онлайн-мошенничество, нарушение приватности и сложности с контролем качества контента [3].

Web 3.0 все еще остается концепцией, но ее идеи уже сейчас привлекают внимание. Одна из ключевых идей – децентрализация данных и повышение качества сервисов. Также предполагается использование искусственного интеллекта для улучшения пользовательского опыта и создания более персонализированной среды. Концепция Web 3.0 также отличается от предыдущих поколений тем, что пользователи будут играть более активную роль в формировании контента и контроле над своими данными. Это может привести к более открытому и прозрачному интернету, где каждый сможет участвовать в создании контента и обмене информацией.

Различия концепций представлены в таблице 1 [4].

Таблица 1 – Различия концепций web-технологий

Концепция Свойство	***Web 1.0***	***Web 2.0***	***Web 3.0***
Концепция	Веб только для чтения («the mostly read only web»)	Веб для бурного чтения-записи («the wildly read-write web»)	Портативный индивидуальный Веб («the portable personal web»)
Количество пользователей	45 миллионов глобальных пользователей (1996)	Больше 1 миллиарда глобальных пользователей (2006)	Еще больше
Ориентация	Ориентация на компании (focused on companies)	Ориентация на сообщества (focused on communities)	Ориентация на индивидуальностях (focused on the individual)
Структура данных	Домашние страницы (home pages)	Блоги (blogs)	Lifestreaming-функции (lifestream)
Концепция данных	Владение контентом (owning content)	Обмен контентом (sharing content)	Объединение динамического контента (*consolidating dynamic content*)
Управление знаниями	Britannica Online	Wikipedia	*Интернет*
Технологии	*HTML*, порталы	*XML*, RSS	Технологии «drag and drop» и mashups
Представление	Web формы	Веб приложения	Виджеты (widgets) и гаджеты (*gadgets*)
Классификация	Директории (иерархическое строение) (directories (*taxonomy*))	Тэги (практика совместной категоризации информации (ссылок, фото, видео клипов и т. п.)) (tagging («folksonomy»))	Поведение пользователей (большая сосредоточенность на предпочтениях отдельных лиц) (user behavior («me-onomy»))
Поиск	Netscape	Google	iGoogle, NetVibes
Стоимость рекламы	Просмотр страниц (pages views)	Цена за клик (cost per click)	Активность пользователей (user engagement)
Продвижение	Реклама (advertising)	«Из уст в уста» (word of mouth)	Advertainment

1.2. Обзор технологий

Веб-интерфейс представляет собой набор инструментов и средств, с помощью которых пользователь может взаимодействовать с веб-сайтом или другим программным обеспечением через браузер. Ключевое преимущество онлайн-инструментария заключается в возможности доступа к нему через интернет или виртуальную частную сеть (VPN) с использованием любого современного браузера [5].

Изначально веб-страницы в основном применялись для простого отображения информации [6]. Однако со временем они стали более функциональными и интерактивными, а их количество значительно возросло. В связи с этим возникла необходимость в разработке более привлекательных и

интересных веб-интерфейсов, способных пробудить интерес пользователей к компании или организации, представленной на сайте.

«Динамическая информация (средства отображения информации) – информация, отображаемая на экране средства отображения информации, меняющаяся во времени по содержанию и (или) но положению на экране» [7].

Разница между статическими и динамическими веб-страницами в таблице 2 [8].

Таблица 2 – Разница между статическими и динамическими веб-страницами

Статическая веб-страница	Динамическая веб-страница
На статических веб-страницах страницы останутся прежними, пока кто-то не изменит их вручную.	На динамических веб-страницах содержимое страниц отличается для разных посетителей.
Статические веб-страницы просты с точки зрения сложности.	Динамические веб-страницы сложны.
На статических веб-страницах информация редко меняется.	На динамической веб-странице информация часто меняется.
Загрузка статической веб-страницы занимает меньше времени, чем динамическая веб-страница.	Динамическая веб-страница занимает больше времени для загрузки.
В статических веб-страницах база данных не используется.	На динамических веб-страницах используется база данных.
Статические веб-страницы написаны на таких языках, как: HTML, JavaScript, CSS и т. д.	Динамические веб-страницы написаны на таких языках, как: CGI, AJAX, ASP, ASP.NET и т. д.
Статические веб-страницы не содержат никакой прикладной программы.	Динамические веб-страницы содержат прикладную программу для различных сервисов.
Статические веб-страницы требуют меньше работы и затрат при их разработке.	Динамичные веб-страницы требуют сравнительно больше работы и затрат при их разработке.

Разработка веб-сайта обычно включает две основные стадии: бэкенд-разработку и фронтенд-разработку [9]. Бэкенд-разработка сосредоточена на серверной части, включая работу с базами данных, обработку и хранение информации, управление пользовательскими данными и обеспечение безопасности. Фронтенд-разработка, в свою очередь, фокусируется на создании пользовательского интерфейса, отображении контента и функций, а также обеспечении интерактивности и отзывчивости веб-приложения.

В настоящее время основными технологиями для создания веб-интерфейсов являются: HTML, CSS и Java Script.

HTML является основной технологией для создания веб-страниц. С помощью HTML вы определяете структуру и содержимое вашего веб-интерфейса. HTML предоставляет множество тегов и атрибутов, которые позволяют создавать разные части веб-страницы, такие как заголовки, параграфы, изображения, ссылки и т.д.

CSS используется для задания стилей и внешнего вида веб-страниц. С помощью CSS можно изменить цвета, шрифты, размеры, расположение элементов и многое другое. CSS работает в сочетании с HTML, позволяя создавать привлекательный дизайн веб-интерфейса.

JavaScript – это язык программирования, который позволяет добавлять интерактивность и функциональность вашему веб-интерфейсу. С помощью JavaScript вы можете обрабатывать события, выполнять анимацию, валидацию форм, запросы на сервер и многое другое [10].

Технология DOM, завершающая этот ряд, является объединяющей, так как обеспечивает через JavaScript доступ ко всем элементам html-документа с возможностью динамического изменения этих элементов вплоть до удаления и создания новых, а также позволяет изменять стилевые свойства объектов, за которые отвечает технология CSS. Часто все эти технологии в объединенном варианте называют «Динамический HTML» [11].

PHP (Personal Homepage Tools) – еще один язык программирования, используемый для добавления интерактивных элементов на веб-страницы. Код

PHP встраивается в HTML-документ подобно подпрограмме – сценарий PHP вставляется в ту часть документа, где должен быть размещен интерактивный элемент. Язык PHP имеет мнемонику, основанную на синтаксисе языков PERL, Java и C, что облегчает его изучение. Технология PHP позволяет создавать счетчик посещений веб-страницы, считать статистику обращений к различным разделам веб-сайта, защищать доступ к HTML-документу паролем и многое другое. Однако, стоит отметить, что PHP поддерживается не всеми интернет-серверами [12].

Помимо этих трех основных технологий, в последние годы появилось множество дополнительных инструментов и фреймворков, которые значительно расширяют возможности разработки веб-приложений.

Например, существуют JavaScript-фреймворки, такие как React, Angular, Vue.js, которые позволяют создавать более сложные и динамичные пользовательские интерфейсы с использованием компонентного подхода. Они предоставляют мощные средства для управления состоянием приложения, маршрутизации, работы с HTTP-запросами и многое другое.

Кроме того, активно развиваются технологии, связанные с серверной частью веб-приложений. Здесь можно выделить такие языки, как Node.js (для создания серверной логики на JavaScript), Python (с фреймворками Django, Flask) и Ruby (с фреймворком Ruby on Rails). Эти технологии позволяют создавать полноценные веб-приложения с серверной логикой, базами данных, API и т.д.

1.3. Обзор инструментов

Создание качественного веб-интерфейса при помощи упомянутых технологий требует много времени и усилий. Именно поэтому появились фреймворки – инструменты, которые уже содержат эти технологии и позволяют автоматизировать многие повторяющиеся операции, значительно упрощая работу.

Фреймворк (от англ. “framework” – “каркас”, “структура”) – это набор инструментов, который уже имеет готовую структуру и правила для дальнейшего развития, существенно упрощая процесс разработки. Большинство фреймворков

основаны на декомпозиции проекта на отдельные слои (приложение, модули и так далее). Это позволяет расширять функциональность программы в соответствии с потребностями и использовать модифицированную версию вместе с исходным кодом фреймворка, или же подключать сторонние приложения. Эта гибкость является одним из основных преимуществ использования фреймворков [13].

Рассмотрим три наиболее популярных инструментов для создания современных веб-интерфейсов: React, Angular, Vue.js.

ReactJS - это мощная и популярная JavaScript-библиотека, разработанная Facebook в 2013 году. Она идеально подходит для создания сложных и динамичных веб-приложений. Ключевой особенностью React является концепция виртуального DOM (Document Object Model), который работает быстрее, чем обычный браузерный DOM, что значительно повышает производительность приложений.

React Native - это платформа для разработки кроссплатформенных мобильных приложений высокого качества как для iOS, так и для Android. Преимущества React Native включают в себя совместимость с различными модулями, готовые компоненты, однонаправленную потоковую модель и библиотеку Redux [14].

Многие известные проекты, такие как Pinterest, Facebook и Netflix, были разработаны с использованием React. Основными преимуществами этой библиотеки являются малый вес базы данных, простота использования, гарантированная неизменность связанных данных, постоянное улучшение открытой библиотеки благодаря открытому исходному коду, а также высокая гибкость и отзывчивость.

Однако, как и у любого инструмента, у React есть и некоторые недостатки. Например, отсутствие упорядоченной документации, большое количество решений для одних и тех же проблем, которое может сбить с толку разработчиков, а также относительно слабая кроссбраузерная поддержка [15].

В целом, React является мощным и востребованным инструментом для разработки современных веб- и мобильных приложений, который постоянно развивается и совершенствуется.

Angular. Angular - это мощный и всесторонний фреймворк с открытым исходным кодом, разработанный и поддерживаемый компанией Google. Он предназначен для создания и управления высокопроизводительными и масштабируемыми динамическими веб-приложениями [16].

Ключевой особенностью Angular является его модульная архитектура, которая позволяет разработчикам создавать приложения из независимых и повторно используемых компонентов. Это упрощает процесс разработки и делает код более читаемым и поддерживаемым. Фреймворк предлагает мощные инструменты для управления данными, маршрутизации, форм, тестирования и многого другого.

Одним из главных преимуществ Angular является его двустороннее связывание данных. Это означает, что изменения в пользовательском интерфейсе мгновенно отражаются в приложении, и наоборот. Фреймворк автоматически отслеживает события браузера, изменения модели и действия пользователя, обновляя соответствующие шаблоны без необходимости ручного управления DOM [17].

Другие ключевые преимущества Angular включают его интерфейс командной строки (CLI) для быстрого прототипирования, богатую экосистему библиотек и инструментов, а также обширное сообщество разработчиков, которое предоставляет множество ресурсов для обучения и решения проблем [18].

Несмотря на свои преимущества, Angular также имеет некоторые недостатки. Например, его сложный синтаксис и строгая структура могут ограничивать гибкость при проектировании. Кроме того, встроенные интерактивные элементы могут замедлять производительность страниц, а переход на новые версии может вызывать интеграционные проблемы.

В целом, Angular является мощным и всесторонним фреймворком, который идеально подходит для создания сложных и масштабируемых веб-приложений.

Его модульная архитектура, двустороннее связывание данных и богатая экосистема делают его отличным выбором для многих разработчиков.

Vue. Vue - это действительно прогрессивный JavaScript-фреймворк для создания пользовательских интерфейсов. Он отличается высокой степенью адаптивности и ориентирован на уровень представления (View). Одно из главных преимуществ Vue - его легкая интеграция с другими библиотеками, что позволяет использовать его для улучшения существующих проектов [19].

В отличие от React, Vue не требует такого глубокого понимания JavaScript, включая знание JSX и ES2015+, а также систем сборки. Это делает его более доступным для начинающих разработчиков [20]. При этом, в сравнении с Angular, Vue обладает немного большей производительностью [21].

Основные преимущества Vue [22]:

1. Декларативный подход к программированию, который делает код более легким для чтения и поддержки.
2. Небольшой размер файлов фреймворка.
3. Понятная и подробная документация.
4. Низкий порог входа и простота изучения.
5. Поддержка TypeScript.

Однако, у Vue есть и некоторые недостатки:

1. Относительно небольшое сообщество, так как фреймворк является относительно новым.
2. Недостаток плагинов. Экосистема Vue меньше, чем у других фреймворков, поэтому многие задачи приходится решать самим разработчикам.

В целом, Vue является мощным и гибким инструментом для создания пользовательских интерфейсов, который стоит рассмотреть при выборе JavaScript-фреймворка для своего проекта.

Помимо вышеназванных фреймворков был обнаружен замечательнаый конструктор веб-интерфейсов GyverPortal [23]. Gyverportal - это легковесная и

компактная библиотека для создания веб-приложений на JavaScript. Основные преимущества Gyverportal перед традиционными фреймворками:

1. Минимализм и простота. Gyverportal не перегружен множеством встроенных функций, а предоставляет только самые необходимые инструменты для разработки. Это позволяет создавать быстрые и эффективные приложения.
2. Высокая производительность. Благодаря отсутствию лишних абстракций и оптимизации кода, Gyverportal демонстрирует высокую производительность и низкое потребление ресурсов.
3. Гибкость и расширяемость. Библиотека имеет модульную структуру, что позволяет подключать только те компоненты, которые нужны для конкретного проекта. Разработчики могут легко расширять функциональность Gyverportal с помощью собственных модулей.
4. Небольшой размер. Размер ядра Gyverportal составляет всего около 10 Кб, что делает ее одной из самых компактных библиотек для веб-разработки.
5. Кроссбраузерность и совместимость. Gyverportal поддерживает все современные браузеры и устройства, обеспечивая единообразное поведение приложений.
6. Простота изучения. Благодаря лаконичному API и понятной документации, Gyverportal легко осваивается даже начинающими разработчиками.

Недостатки Gyverportal:

1. Ограниченная документация. Несмотря на простоту использования, документация по Gyverportal может быть недостаточной для новичков.
2. Небольшое сообщество. По сравнению с популярными фреймворками, таких как React или Angular, сообщество Gyverportal не так велико, что может затруднять поиск решений для нестандартных задач.
3. Малый функционал. Gyverportal, будучи более легковесной библиотекой, может не предоставлять некоторые возможности, которые есть в более крупных фреймворках.

4. Низкая популярность. Gyverportal не так широко известен и используем, как другие фреймворки, что может затруднять поиск специалистов, знакомых с этой библиотекой.

В целом, Gyverportal является хорошим выбором для быстрой и эффективной разработки веб-приложений, особенно для небольших и средних проектов. Однако для более сложных задач могут потребоваться более мощные и функциональные фреймворки.

1.4. Обзор вариантов архитектур микроконтроллеров для реализации веб-сервера.

Микроконтроллеры (МК) являются ключевым компонентом любой системы – от простых датчиков до сложных автоматизированных систем. Потребители предъявляют следующие основные требования к управляющим модулям приборов (микроконтроллеры): низкая стоимость, высокая надежность, высокая степень миниатюризации, малое энергопотребление и способность функционировать в жестких условиях. Достаточная производительность также необходима для выполнения всех необходимых функций. В отличие от универсальных компьютеров, управляющие контроллеры обычно не требуют высокой производительности или программной совместимости. В настоящее время доступно множество типов микроконтроллеров. Все эти устройства можно условно разделить на три основные категории: 8-битные МК для встроенных приложений, 16-битные и 32-битные микроконтроллеры и цифровые сигнальные процессоры.

x86. Архитектура x86 была создана в 1978 году и является разновидностью CISC (Complex Instruction Set Computing). Она предполагает наличие инструкций в процессоре для большинства задач, используется в настольных компьютерах, ноутбуках и других устройствах с высокой производительностью и не экономией энергии. Поскольку история развития этой архитектуры длиннее, чем ARM, она имеет больший набор команд, что делает ее очень сложной и продвинутой, позволяя выполнять множество сложных вычислений за короткий промежуток

времени. Виртуализация здесь лучше, есть разнообразные защитные методы. В ARM-процессоре набор команд пока меньше, но разница сокращается из-за уменьшения техпроцесса и развития производственных технологий.

ARM. ARM была создана в 1985 году компанией Acorn и относится к типу RISC. Этот подход предполагает, что процессор содержит минимальный набор команд, необходимых для работы. Инструкции здесь проще и короче. Целью разработчиков ARM было создание архитектуры, свободной от недостатков x86, и можно сказать, что они создали очень эффективную и недорогую архитектуру. ARM-процессоры стали популярны на рынке мобильных устройств не только из-за этого. Вот несколько дополнительных преимуществ:

1. Они дешевле в производстве и использовании.
2. Архитектуру ARM позволяют крупным поставщикам создавать свои собственные решения для разных ниш.
3. Она предлагает гибкие возможности настройки.

Кроме того, низкое энергопотребление и отсутствие перегрева делают эти процессоры идеальным выбором для серверных приложений, а также для маршрутизаторов и высокопроизводительных решений для хранения данных. Если говорить о недостатках, они заключаются в том, что из-за компактного размера и ориентации на автономность, данные обрабатываются медленнее и не так эффективно, как на x86. Также, если какая-то часть вашей ИТ-инфраструктуры основана на решениях x86, вам придется учитывать это при выборе между ARM и x86. Одни и те же программы не будут работать на обеих платформах без адаптации, для одной из них вам точно придется выполнить адаптацию [24].

Самые популярные архитектуры процессоров сегодня – x86 и ARM, но недавно у них появился новый конкурент – RISC-V. RISC-V – это открытая архитектура набора инструкций (ISA), базирующаяся на концепции компьютера с ограниченным набором инструкций (RISC). ISA представляет собой абстрактную модель компьютера, которая определяет инструкции, или основные операции с описаниями их использования. RISC-V состоит из 47 основных инструкций, и к

ним могут быть добавлены дополнительные расширения. Изначально существовали два типа архитектуры микропроцессора: RISC и CISC. В CISC используется один набор инструкций для выполнения нескольких низкоуровневых функций (загрузка, оценивание, сохранение данных). Таким образом, программа использует меньше специальных инструкций, но время выполнения одной инструкции увеличивается. Примером CISC-архитектуры может служить x86. RISC, напротив, использует небольшой набор кастомизированных инструкций, поэтому программе требуется больше инструкций, однако каждая инструкция выполняется быстрее.

RISC-V является открытой и свободной моделью архитектуры. Проектирование RISC-V основано на принципах проектирования продуктов с открытым исходным кодом. Модульная архитектура позволяет разработчикам использовать только необходимые модули для создания любых устройств, от компактных встраиваемых систем до мощных компьютеров. Нет необходимости платить лицензионные сборы за использование RISC-V в исследовательских и конструкторских работах, что позволяет любому энтузиасту внести свой вклад в улучшение микропроцессоров. Такая концепция способствует инновациям. Снижаются затраты на разработку и сокращается время вывода продукта на рынок, повторное использование открытого исходного кода обеспечивает быструю и эффективную разработку программного и аппаратного обеспечения. RISC-V предоставляет российским разработчикам оборудования возможность снизить зависимость от иностранных поставщиков и создавать технические решения в рамках стратегии импортозамещения [25].

ESP8266. ESP8266 – это Wi-Fi микросхема, которая включает в себя все необходимые компоненты для создания функционального Wi-Fi устройства. Она представляет собой недорогой микрочип с поддержкой Wi-Fi, который может быть подключен для беспроводного подключения к любому микроконтроллеру. ESP8266 имеет 16 контактов GPIO и различные периферийные устройства, такие как последовательный периферийный интерфейс, протокол для связи между интегральными схемами и аналого-цифровой преобразователь. Однако, у него нет

доступа к Интернету и сенсорным датчикам, а также нет криптографических датчиков или датчиков температуры.

ESP32. ESP32 является обновленной версией ESP8266 и оснащен 34 контактамиGPIO с двухъядерным процессором Xtensa на 160 МГц. Чип имеет 32-бит процессор с суперэнергоэффективным сопроцессором и множество входов/ выходов, включая аналого-цифровое преобразование. ESP32 предоставляет безопасную платформу для IoT. Он обеспечивает доступ к дистанционному контролю и датчику температуры. Чип предлагает 1024 битное шифрование загрузочного флэш-хранилища с OTP и 16 битным ШИМ. ESP32 оборудован десятью сенсорами.

Если выбирать между ESP32 и ESP8266, то лучше остановиться на ESP32 по следующим причинам:

1. ESP32 оснащена более быстрым процессором и большим объемом памяти, что позволяет разрабатывать значительно более масштабные проекты на одной микросхеме.
2. ESP32 обеспечивает надежную и современную систему безопасности, что является ее преимуществом.
3. Плата ESP32 обладает более надежными платой, прошивкой и периферийными устройствами. Вычислительные мощности также обеспечивают безопасные соединения на уровне сокетов и соответствуют всем требованиям мира IoT.
4. На плате ESP32 больше GPIO, что позволяет работать с более сложными и полезными проектами. Многие платы ESP32 идут с небольшими камерами, что является приятным бонусом.
5. В ESP32 имеются возможности, обеспечивающие лучшую безопасность и большее количество оперативной памяти для проектов, которых нет в ESP8266.

1.5. Среда разработки

Необходимо, чтобы среда разработки (Integrated development environment – IDE) поддерживала языки JavaScript и C/C++, поэтому рассмотрим три наиболее распространенных варианта и выберем наиболее подходящий:

Visual Studio Code (VS Code) - это бесплатный и кроссплатформенный редактор кода от Microsoft, который отлично подходит для работы с JavaScript, C/C++ и поддерживает множество расширений, включая инструменты для работы с Gyverportal.

Преимущества:

1. Официальная поддержка JavaScript, C/C++ и множества других языков программирования.
2. Огромное количество расширений, включая поддержку Gyverportal.
3. Встроенная отладка, интеграция с системами контроля версий, терминал и другие полезные инструменты.
4. Кроссплатформенность (доступен на Windows, macOS и Linux).

Недостатки:

1. Поддержка TypeScript.
2. Может быть ресурсоемким для старых или слабых машин.

1. Некоторые пользователи могут найти интерфейс сложным для освоения.

Sublime Text - еще один отличный выбор для работы с Gyverportal. Sublime Text известен своей производительностью, расширяемостью и удобным интерфейсом. Он предоставляет множество плагинов и пакетов, которые могут быть полезны при разработке с Gyverportal.

Преимущества:

1. Быстрый и легкий редактор кода.
2. Поддержка множества языков программирования, включая JavaScript и C/C++.
3. Широкий выбор расширений, в том числе для работы с Gyverportal.
4. Простой и интуитивно понятный интерфейс.

Недостатки:

1. Не имеет встроенной отладки, как VSCode.
2. Платная лицензия после пробного периода.
3. Ограниченная интеграция с системами контроля версий.

Atom - это бесплатный и открытый редактор кода, разработанный GitHub. Он имеет большое сообщество разработчиков, которые создают множество полезных пакетов и расширений, в том числе для работы с Gyverportal.

Преимущества:

1. Бесплатный и открытый исходный код.
2. Поддержка множества языков программирования, включая JavaScript и C/C++.
3. Большое сообщество разработчиков, создающих расширения, в том числе для Gyverportal.
4. Кроссплатформенность (доступен на Windows, macOS и Linux).

Недостатки:

1. Может быть более ресурсоемким, чем Sublime Text.
2. Некоторые пользователи могут найти интерфейс менее интуитивным, чем в VSCode или Sublime Text.

В дополнение к средам разработки, можно также рассмотреть такие варианты, как PlatformIO [26] или Arduino IDE [27], которые также хорошо подходят для работы с библиотеками, ориентированными на встраиваемые системы, как Gyverportal.

PlatformIO - это мощная кроссплатформенная среда разработки, которая предоставляет ряд преимуществ по сравнению с Arduino IDE:

1. Поддержка множества плат: PlatformIO поддерживает широкий спектр плат, включая Arduino, ESP, Raspberry Pi и многие другие. Это дает больше гибкости при выборе оборудования.
2. Интеграция с популярными IDE: PlatformIO может быть интегрирован с такими IDE, как Visual Studio Code, Atom и CLion, что упрощает разработку.
3. Улучшенное управление зависимостями: PlatformIO предоставляет более эффективное управление библиотеками и зависимостями, что упрощает сборку и развертывание проектов.
4. Расширенные возможности отладки: PlatformIO имеет более мощные инструменты для отладки, включая поддержку удаленной отладки и интеграцию с популярными отладчиками.
5. Кроссплатформенность: PlatformIO работает на Windows, macOS и Linux, что делает его более универсальным по сравнению с Arduino IDE, которая в основном ориентирована на Windows.

Arduino IDE (Integrated Development Environment) - это программное обеспечение, разработанное для написания, компиляции и загрузки кода на платы Arduino. Это бесплатная и открытая среда разработки, которая предоставляет простой и интуитивно понятный интерфейс для программирования Arduino-совместимых плат.

Преимущества:

1. Простота и удобство использования. Arduino IDE имеет интуитивно понятный интерфейс, который легко освоить даже новичкам.
2. Широкая поддержка аппаратных платформ Arduino. Arduino IDE отлично подходит для работы с различными платами Arduino.
3. Обширная база знаний и сообщество. Arduino имеет большое сообщество разработчиков, что облегчает поиск решений и помощь.
4. Встроенные библиотеки. Arduino IDE поставляется с множеством полезных библиотек, которые упрощают разработку.

Недостатки:

1. Ограниченность функциональности. Arduino IDE имеет ограниченный набор инструментов по сравнению с более мощными средами разработки.
2. Отсутствие поддержки сложных проектов. Для больших и сложных проектов Arduino IDE может оказаться недостаточно гибкой.
3. Ограниченная кроссплатформенность. Arduino IDE работает только на Windows, macOS и Linux, в то время как PlatformIO кроссплатформенный.

В целом, PlatformIO предоставляет более гибкую и мощную среду разработки для встраиваемых систем, что делает его предпочтительным выбором по сравнению с Arduino IDE, особенно для более сложных проектов. Однако Arduino IDE по-прежнему остается хорошим выбором для начинающих или простых проектов.

Вывод по разделу 1

В первой главе был проведен аналитический обзор литературы по технологиям, а также обзор необходимых инструментов и микроконтроллеров. Таким образом, для разработки были выбраны: конструктор веб-интерфейсов GyverPortal, микроконтроллер ESP32 и среда для разработки Visual Studio Code + PlatformIO.

2. РАЗРАБОТКА АРХИТЕКТУРЫ ВСТРОЕННОГО ВЕБ-СЕРВЕРА

Для понимания работы встроенного веб-сервера, который взаимодействует с динамически изменяющимися данными, была разработана следующая схема, представленная на рисунке 1.

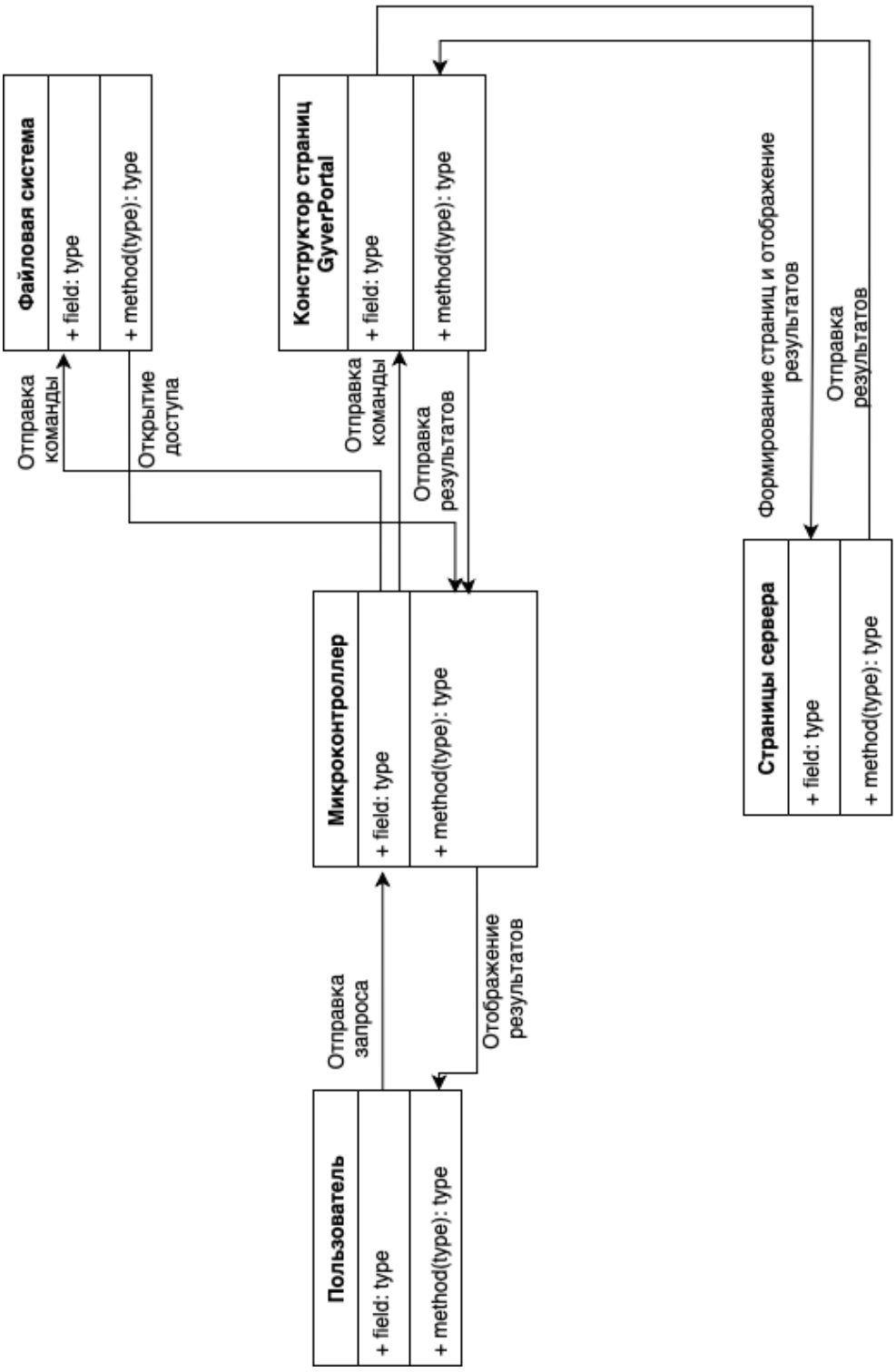

Рисунок 1 – Схема взаимодействия модулей программы

1. Пользователь – это тот, кто взаимодействует с веб-сервером, открывая страницы через браузер. Он может взаимодействовать с элементами на странице, отправлять данные на сервер и получать обновленную / измененную информацию.
2. Микроконтроллер – это устройство, на котором будет работать наш веб-сервер. Он обрабатывает запросы от пользователя и, по сути, управляет всеми компонентами системы.
3. Файловая система на микроконтроллере хранит все необходимые файлы для работы веб-сервера, включая JavaScript и другие ресурсы. Она обеспечивает доступ к этим файлам и позволяет микроконтроллеру загружать их при запросе.
4. Конструктор Web-страниц (Gyver Portal) – это инструмент, который позволяет создавать динамические веб-страницы без необходимости писать код. Он может использоваться для быстрого создания интерфейсов, отображения данных и управления контентом на веб-сервере.
5. Страницы сервера – это веб-страницы, которые отображаются пользователю при запросе. Они могут быть написаны вручную с использованием HTML, CSS и JavaScript, но в нашем случае был применен Gyver Portal. Страницы сервера могут содержать динамически изменяющиеся данные, которые микроконтроллер получает и отображает на страницах.

Вывод по разделу 2

В второй главе была создана схема взаимодействия программы, где микроконтроллер взаимодействует с файловой системой для доступа к ресурсам, с конструктором Web-страниц для создания интерфейсов и с страницами сервера для отображения контента клиентам. Все эти компоненты совместно позволяют создать функциональный и динамический веб-сервер.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

В второй главе была спроектирована архитектура программного обеспечения проекта. В данной главе необходимо разработать спроектированный алгоритм программы.

В файле main.cpp определяется режим работы (точка доступа (раздает WiFi) или рабочая станция (подключается к имеющейся точке доступа)), название сети и пароль WiFi, показанный на рисунке 2.

```
#define WIFI_MODE 1 /* 0 - STA, 1 - AP*/
#define AP_SSID  «TP-Link_B53F»
#define AP_PASS  «15700933»
```

Рисунок 2 – Название сети и пароль WiFi в листинге А.1 приложения А.

Подключаются библиотеки (заголовочные файлы) (рисунок 3).

```
void lcd_setup(void);
void lcd_tick(void);
void clock_setup();
void clock_loop();
void ff_setup();
void ff_loop();
```

Рисунок 3 – Подключение библиотеки в листинге А.1 приложения А.

Объявлены прототипы тестовых функций (рисунок 4).

```
void lcd_setup(void);
void lcd_tick(void);
void clock_setup();
void clock_loop();
void ff_setup();
void ff_loop();
```

Рисунок 4 – Объявление прототипов тестовых функций в листинге А.1 приложения А.

Создается объект для работы с веб-интерфейсом: «GyverPortal ui;».

Определены функции для доступа к данным контроллера по протоколу Modbus (рисунок 5).

```
int16_t ModbusGetReg(uint16_t addr) {

 return ECU_AccessRegister(addr, 0, 0);
}

void ModbusSetReg(uint16_t addr, int16_t val) {

 ECU_AccessRegister(addr, val, 1);
}
```

Рисунок 5 – Определены функции для доступа к данным контроллера по протоколу Modbus в листинге А.1 приложения А.

Далее реализованы две основные функции build() и action(), которые опишем чуть подробнее (рисунок 6). Функция build() вызывается объектом ui, когда к web-серверу подключен клиент и нужно сформировать для него html-страницу. В целом он содержит в себе функции, необходимые для настройки и создания внешнего вида веб-интерфейса. Сам веб-интерфейс является многостраничным, ниже показано содержание меню и вывод новой страницы для каждого пункта меню.

```
// ссылки меню
 GP.UI_LINK(«/», «Доступ, идентфикация»);
 GP.UI_LINK(«/status», «Мониторинг состояния»);
 GP.UI_LINK(«/inputs_settings», «Параметры входов-выходов»);
 GP.UI_LINK(«/control_settings», «Параметры регуляторов»);
 GP.UI_LINK(«/plots» , «Осциллограф»);
 GP.UI_LINK(«/stat» , «Статистика»);
 GP.UI_LINK(«/update», «Обновление прошивки»);
```

Рисунок 6 – Ссылки меню в листинге А.1 приложения А.

Этот код создает элементы интерфейса для отображения текущего времени и кнопку «Сохранить параметры». Также запускается основное окно интерфейса и имеется возможность делать таблицу отключаемой при ширине экрана меньше 700 пикселей (рисунок 7).

```
// контент
 GP.HR(GP_GRAY);
 GP.LABEL(«Текущее время:»);
 GP.BREAK();
 GP.LABEL(ui.getSystemTime().encode());
 GP.BUTTON(«btnSave», «Сохранить параметры», ««, GP_RED);
 // начать основное окно
 GP.UI_BODY();
 // основной интерфейс
 // позволяет «отключить» таблицу при ширине экрана меньше
700px
 GP.GRID_RESPONSIVE(600);
```

Рисунок 7 – Контент в листинге А.1 приложения А.

Дальше проверяется условие на соответствие текущего URI значению «/ control_settings» (рисунок 8). Если соответствует, то создается сетка (grid) с блоком (block tab) под названием «Параметры регуляторов» и «Задание положения». Внутри первого блока находятся три элемента M_BOX, каждый из которых содержит метку (label) и спиннер (spinner). Спиннеры используются для ввода числовых значений, и здесь они предназначены для настройки параметров регуляторов (Kp, Ki и Kd). Внутри второго блока находятся: переключатель «Задавать вручную», который позволяет пользователю выбрать, будет ли положение задаваться вручную, два элемента M_BOX, каждый из которых метку «Положение» и ползунок «pos»(используется для изменения значения положения), а также две кнопки, «◄» и «►», использующиеся для изменения положения вправо и влево соответственно.

```
if (ui.uri(«/control_settings»)) {
  // Настройка регуляторов
  M_GRID(
   M_BLOCK_TAB(
    «Параметры регуляторов»,
    M_BOX(GP.LABEL(«Kp»);GP.SPINNER(«kP»););
    M_BOX(GP.LABEL(«Ki»);GP.SPINNER(«kI»););
    M_BOX(GP.LABEL(«Kd»);GP.SPINNER(«kD»););
   );
       M_BLOCK_TAB( «Задание положения», GP.SWITCH(«Задавать
    вручную»); GP.BREAK();
       M_BOX(GP.LABEL(«Положение»);GP.SLIDER(«pos»); );
       M_BOX(GP.BUTTON(«posB», «◀»); GP.BUTTON(«posF», «▶»););
);
);
```

Рисунок 8 – Настройка регуляторов в листинге А.1 приложения А.

В этом фрагменте идет проверка условия на соответствие, текущего URI значению «/plots». Если соответствует, то создается сетка с блоком «Осциллограф» и 2D-графиком на темном фоне. Параметр «table» содержит в себе график, names – массив строк с названием графика, arr – двумерный массив, содержащий значения для графика (рисунок 9).

```
else if (ui.uri(«/plots»)) {
  //M_GRID(
  //M_BLOCK_TAB(
   // «Осциллограф»,
   GP.PLOT_DARK<2, PLOT_SIZE>(«table», names,arr);
   //GP.PLOT<2, PLOT_SIZE>(«table», names,arr);
  // );
  //);
 }
```

Рисунок 9 – Проверка условий на соответствие в листинге А.1 приложения А.

В этом фрагменте (рисунок 10) идет проверка условия на соответствие, текущего URI значению «/update». Если соответствует, то создается блок «Обновление прошивки» и вызываются GP.OTA_FIRMWARE(), GP.OTA_FILESYSTEM(), которые описывают интерфейс для обновления прошивки и файловой системы.

```
else if (ui.uri(«/update»)) {
  M_BLOCK_TAB(
   «Обновление прошивки»,
   GP.OTA_FIRMWARE();
   GP.OTA_FILESYSTEM();
  );
 }
```

Рисунок 10 – Проверка на соответствие в листинге А.1 приложения А.

В этом фрагменте создается сетка с блоками (рисунок 11), которые содержат разделы информации. В данном случае, есть три основных раздела:

1. УРОВЕНЬ ДОСТУПА: в этом разделе пользователь может указать уровень доступа и пароль.
2. ДАННЫЕ УСТРОЙСТВА: здесь пользователь может увидеть информацию о устройстве, такую как номер партии, серийный номер, аппаратная и программная версии.
3. ИДЕНТИФИКАЦИЯ: в этом разделе есть информация о идентификации устройства, такие как EMAC, EIP, и ад сети Modbus. Также есть кнопка «Изменить», которая, вероятно, позволит изменить эти данные. `GP.LABEL`, `GP.NUMBER`, `GP.PASS`, `GP.TEXT`, и `GP.BUTTON` используются для создания разных типов элементов интерфейса, таких как метки, поля для ввода чисел, поля для ввода паролей, текстовые поля, и кнопки соответственно.

```
else {
  M_GRID(
   M_BLOCK_TAB(
    «УРОВЕНЬ ДОСТУПА»,
    M_BOX(GP.LABEL(«Уровень доступа:»); GP.NUMBER(«AL»,»«,
ECU_Id.ACL););
    M_BOX(GP.LABEL(«Пароль доступа:»); GP.PASS(«PWD»););
    GP.BUTTON(«btnPWD», «Запросить доступ»);
   );
   M_BLOCK_TAB(
    «ДАННЫЕ УСТРОЙСТВА»,
    M_BOX(GP.LABEL(«№ партии:»); GP.NUMBER(«PN»,»«,
ECU_Id.BatchNo); );
    M_BOX(GP.LABEL(«№ серийный:»); GP.NUMBER(«SN»,»«,
(ECU_Id.SerialNo)); );
    M_BOX(GP.LABEL(«Аппаратная версия:»); GP.TEXT(«HW»,»«,
V_str(ECU_Id.HWver)); );
    M_BOX(GP.LABEL(«Версия ПО:»); GP.TEXT(«SW»,»«,
V_str(ECU_Id.SWver)); );
   );
   M_BLOCK_TAB(
    «ИДЕНТИФИКАЦИЯ»,
    M_BOX(GP.LABEL(«EMAC:»); GP.TEXT(«MAC»,»«,
MAC_str(ECU_Id.MAC)););
    M_BOX(GP.LABEL(«EIP:»); GP.TEXT(«IP»,»«,
IP_str(ECU_Id.IP)););
    M_BOX(GP.LABEL(«Адрес в сети Modbus:»); GP.NUMBER(«MA»,»«,
ECU_Id.ModbusAddr););
    GP.BUTTON(«btnID», «Изменить»);
   );
  );
 }
```

Рисунок 11 – Создание сетки с блоками в листинге А.1 приложения А.

Этот фрагмент кода создает блок с названием «Журнал событий» в интерфейсе пользователя. Внутри этой вкладки отображается текстовая область (area log), которая предназначена для отображения данных в реальном времени. Также завершаются создание панели управления и построение самой веб-страницы (рисунок 12).

```
M_BLOCK_TAB(
  «Журнал событий»,
  /*M_BOX(*/ GP.AREA_LOG(); /*);*/
 )
 GP.UI_END();  // завершить окно панели управления <<<<--------
-----
 GP.BUILD_END();
}
```

Рисунок 12 – Создание блока «Журнал событий» в листинге А.1 приложения А.

Функция void action() которая обрабатывает нажатия на кнопки в веб-интерфейсе. Если пользователь нажимает на элемент, то функция записывает имя элемента и его содержимое. Если же пользователь нажимает на элемент с именем «PWD», то функция записывает введённое пользователем значение в переменную buf. Затем это значение преобразуется в целое число и сохраняется в переменной PWD. Затем функция String(buf).toInt() преобразует это значение в целое число и сохраняет его в переменной PWD (рисунок 13).

```
void action() {
 if (ui.click()) {
  ui.log.println(String(«Click on «) + ui.clickName() + ' ' +
ui.getString());
  // text PWD
  if(ui.clickStr(«PWD», buf, sizeof(buf)-1)) {
   PWD = String(buf).toInt();
   //ui.log.println(String(«PWD=«) + ECU_Id.PWD);
  }
```

Рисунок 13 – Обработка нажатия на кнопки в листинге А.1 приложения А.

В этом фрагменте проверяется, если нажата кнопка с именем «PWD», то в переменную «PWD» сохраняется значение из текстового поля, которое отображается рядом с кнопкой (рисунок 14). Затем проверяется, совпадает ли значение `PWD` с одним из трех значений (2222, 3333, 4444). В зависимости от значения, устанавливается новый уровень доступа (`ECU_Id.ACL`). Таким образом, этот фрагмент кода обрабатывает нажатия на кнопку «PWD» и меняет уровень доступа в зависимости от введённого пароля.

```
// button PWD
if(ui.click(«btnPWD»)) {
 if(PWD == 2222) ECU_Id.ACL = 2; else
 if(PWD == 3333) ECU_Id.ACL = 3; else
 if(PWD == 4444) ECU_Id.ACL = 4;
 ui.log.println(String(«запрошен доступ с паролем «) + PWD);
}
```

Рисунок 14 – Обработка нажатия на кнопки в листинге А.1 приложения А.

Проверяется условие, если нажата кнопка Save, то идет сообщение о записи параметров в память устройства (рисунок 15). Кроме того, проверяется условие, если нажата кнопка ID, то идет сообщение о записи изменений параметров сетевой идентификации. Также есть условие об обнаружении изменений в интерфейсе пользователя, то есть если произошли какие-то изменения, то идет сообщение с информацией об обновлении значений в интерфейсе и функция ui.updateName() возвращает имя того поля, в котором произошли изменения. Проще говоря, идет отслеживание изменений/действий в веб-интерфейсе.

```
  // button Save
  if(ui.click(«btnSave»)) {
   ui.log.println(«запись параметров в память устройства...»);
  }
  // button ID
  // if(ui.click(«btnID»)) {
  //  ui.log.println(«изменены параметры сетевой
идентификации...»);
  // }
 }
 if(ui.update()) {
  ui.log.print(String(«Update of «) + ui.updateName());
 }
}
```

Рисунок 15 – Проверка нажатия Save в листинге А.1 приложения А.

Вывод по разделу 3

В третьей главе был разработан код для создания веб-интерфейса с использованием библиотеки GyverPortal. Веб-ннтерфейс позволяет выполнять различные операции, такие как мониторинг состояния, настройка параметров, просмотр графиков и обновление прошивки. Код демонстрирует использование "блоков" и сетки для структурирования и оформления веб-интерфейса.

4. ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО КОДА

В данной главе необходимо проверить работоспособность разработанного программного кода. Для определения соответствует ли результаты ожидаемым требованиям, был приведён ряд тестов. Перечень тестов приведён в таблице 3.

Таблица 3 – Перечень проведённых тестов

Ожидаемый результат	Полученный результат
Время обновления должно быть ≤ 1000 мс	Время обновления было от 800-900 мс
Веб-интерфейс должен быть рабочим и осуществлять переход между страницами	Все работает
Предполагается управление с 1-2 устройств (клиентов)	Возможность подключения до ~8 клиентов

Кроме того, на рисунках 16-18 продемонстрированы результаты работы выполненной программы.

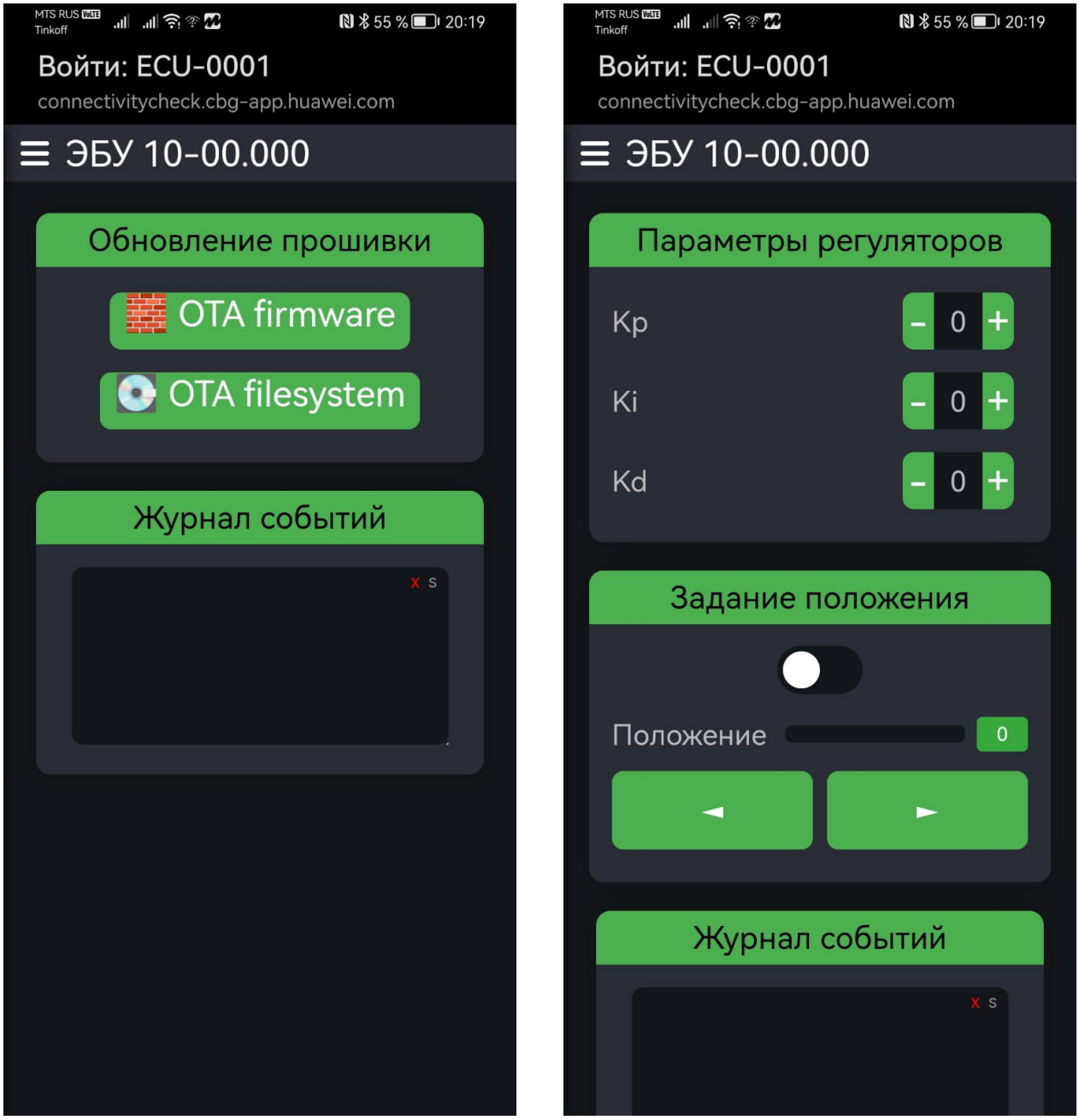

Рисунок 16 – Экранная форма выполнения программы

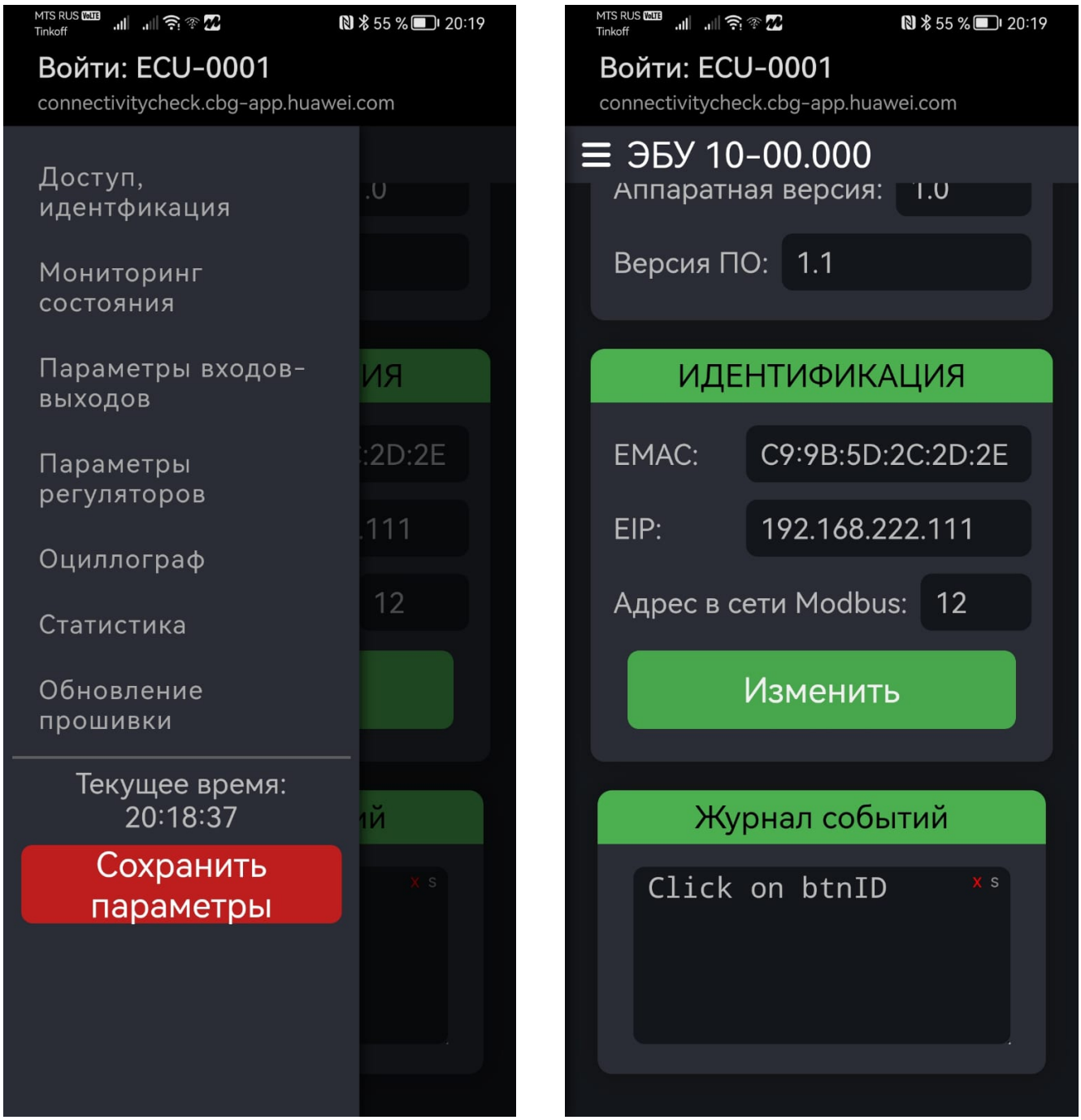

Рисунок 17 – Экранная форма выполнения программы

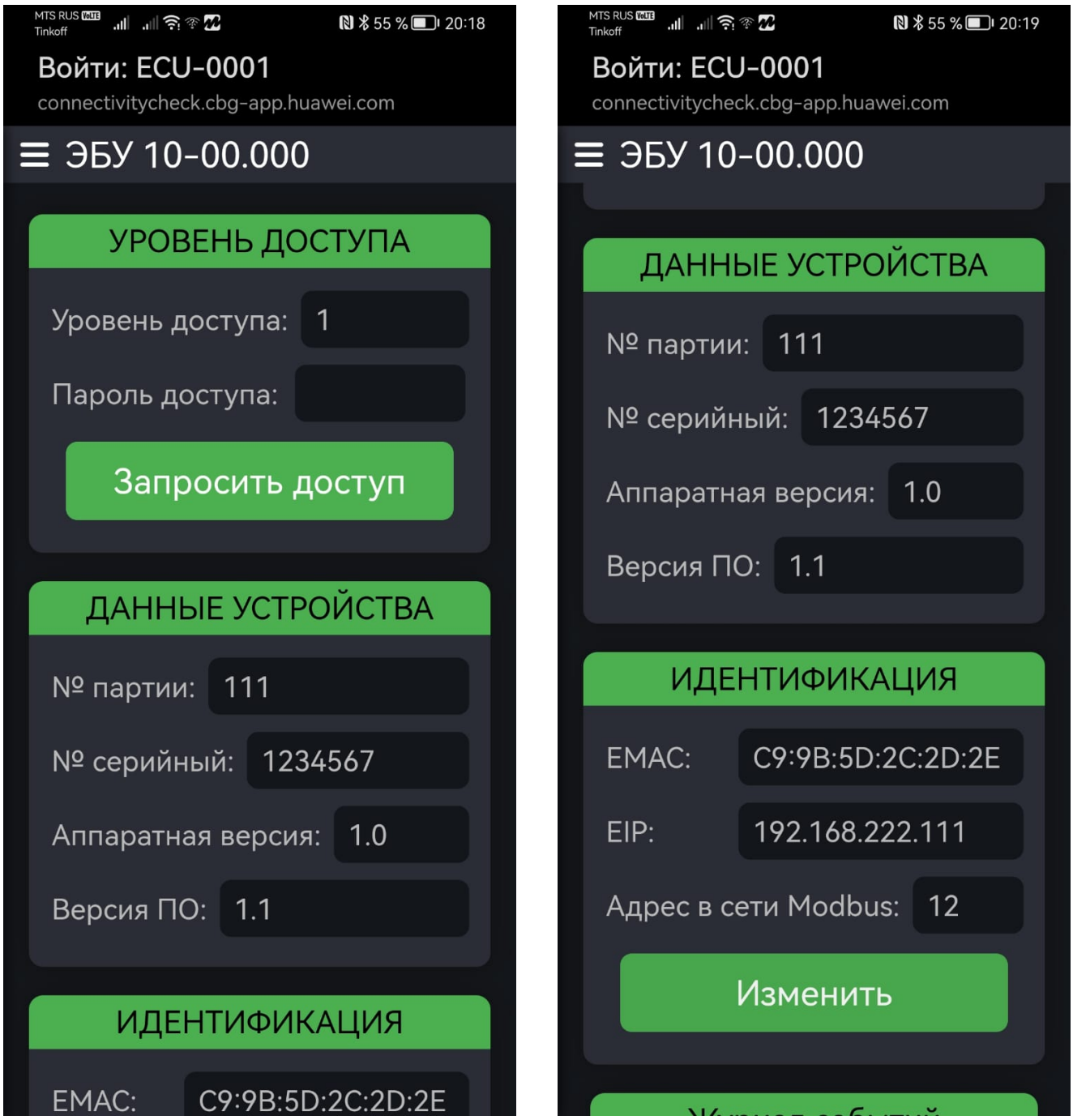

Рисунок 18 – Экранная форма выполнения программы

Вывод по разделу 4

В четвертой главе было проведено тестирование разработанного кода для создания веб-интерфейса с использованием библиотеки GyverPortal. Результаты тестирования, представленные в таблице 3 и на рисунках 16-18 показывают, что программа является рабочей и соответствует поставленным требованиям.

ЗАКЛЮЧЕНИЕ

В рамках ВКР был проведен аналитический обзор технологий для отображения динамически изменяющихся данных с помощью веб-интерфейса, были выбраны необходимые для разработки средства.

Была создана схема взаимодействия программы, в которой все компоненты совместно позволяют создать функциональный и динамический веб-сервер.

Опираясь на это, был разработан программный код для отображения данных в реальном времени с помощью веб-интерфейса.

Кроме того, была проведена проверка работоспособности разработанного кода в реальных условиях, в ходе которой был проведён ряд тестов для определения, соответствует ли результат некоторым заданным требованиям.

В результате выполнения всех задач достигнута цель ВКР, которая заключалась в разработке программного кода для отображения в реальном времени данных, поступающих от электронного блока управления, с помощью веб-интерфейса.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. В ЮУрГУ создают программно-аппаратный комплекс для управления следящим гидроприводом. – https://www.susu.ru/ru/news/2023/11/08/v-sozdayut-programmno-apparatnyy-kompleks-dlya-upravleniya-sledyashchim. Дата обращения: 25.02.2024.

2. Проект ЮУрГУ по производству следящих гидроприводов получил федеральную поддержку. – https://www.susu.ru/ru/news/2023/01/11/proekt-po-proizvodstvu-sledyashchih-gidroprivodov-poluchil-federalnuyu-podderzhku. Дата обращения: 25.02.2024.

3. Современные веб-интерфейсы. Основные технологии разработки: https://digital.tn.ru/articles/sovremennye-veb- interfeysy-osnovnye-tekhnologii-razrabotki/?ysclid=lqdpmpjmzc478254000. Дата обращения: 25.12.2023.

4. Киргизова Е. В., Рубцов А. В. WEB-ТЕХНОЛОГИИ: от теории к практике //Учебное пособие. – 2017. – №. 672. – С. 5-8. Дата обращения: 24.12.2023.

5. Левин Ю. Разработка веб-интерфейса для сервера уведомлений // International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 2. – С. 12-16. Дата обращения: 24.12.2023.

6. Скотт Б., Нейл Т. Проектирование веб-интерфейсов. – 2010. Дата обращения: 24.12.2023.

7. ГОСТ 27833-88: Средства отображения информации. Термины и определения: межгосударственный стандарт: изд. офиц. : утв. и введ. в действие Постановлением Государственного комитета СССР по стандартам от 23 сент. 1988 No 3240: введ. впервые : дата введ. 1990-01-01. – Москва : Стандартинформ, 1990. – С. 149-155. –Текст : непосредственный. Дата обращения: 24.12.2023.

8. Официальный сайт компании Geeks for Geeks . – https:// www.geeksforgeeks.org/difference-between-static-and-dynamic-web- pages/. Дата обращения: 24.12.2023.

9. Ершов Т. А. ОБЗОР СОВРЕМЕННЫХ ТЕХНОЛОГИЙ РАЗРАБОТКИ ДЛЯ WEB-ПРИЛОЖЕНИЙ //E-Scio. – 2023. – №. 8 (83). – С. 38-42. Дата обращения: 27.12.2023.

10. Официальный сайт компании initialcode. – https://initialcode.ru/ Дата обращения: 25.12.2023.

11. Диков, А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 / А. В. Диков. — 2-е изд., стер. — Санкт-Петербург : Лань, 2023. — 188 с. — ISBN 978-5-507-46740-2. — Текст : электронный // Лань : электронно-библиотечная система. – https://e.lanbook.com/book/318443. Дата обращения: 22.12.2023.

12. Зеленко О.В., Валеева Л.Р., Климанов С.Г. Обзор современных Web-технологий // Вестник Казанского технологического университета. 2015. №2. – https://cyberleninka.ru/article/n/obzor-sovremennyh-web-tehnologiy. Дата обращения: 27.12.2023.

13. Веб-фреймворки: введение для новичков. — 02.07.2018 / Пер. с англ. // Tproger —типичный программист. – http:// tproger.ru/translations/web-frameworks- how-to-get-started Дата обращения: 27.12.2023.

14. React. JavaScript-библиотека для создания пользовательских интерфейсов. – http://ru.reactjs.org. Дата обращения: 27.12.2023.

15. Стефанов, С. React.js. Быстрый старт = React: Up & Running / Пер. с англ. Н. Вильчинского. — Санкт- Петербург: Питер, 2017. — 304 с. — (Серия «Бестселлеры O`Reilly»). Дата обращения: 27.12.2023.

16. Рейсиг, Д. JavaScript. Профессиональные приемы программирования = Pro JavaScriptTM Techniques / Пер. с англ. Н. Вильчинского. — Санкт-Петербург: Питер, 2008. — 352 с. Дата обращения: 27.12.2023.

17. Angular. JavaScript-фреймворк // Habr. – http:// habr.com/ru/hub/angular . Дата обращения: 27.12.2023.

18. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. СОВРЕМЕННЫЕ ФРЕЙМВОРКИ ДЛЯ РАЗРАБОТКИ WEB- ПРИЛОЖЕНИЙ / / Интеллектуальные технологии на транспорте. 2020. №4 (24). . – https://

cyberleninka.ru/article/n/sovremennye-freymvorki-dlya-razrabotki-web-prilozheniy. Дата обращения: 27.12.2023.

19. Файн, Я. Angular и TypeScript. Сайтостроение для профессионалов = Angular 2 Development with TypeScript / Я. Файн, А. Моисеев; пер. с англ. Н. Вильчинского, Е. Зазнобы. — Санкт-Петербург: Питер, 2018. — 464 с. Дата обращения: 27.12.2023.

20. Vue.js — Introduction // Vue.js. – http://vuejs.org/ v2/guide/index.html. Дата обращения: 27.12.2023.

21. Vue.js — Comparison with Other Frameworks // Vue.js . – http://vuejs.org/ v2/guide/comparison.html. Дата обращения: 27.12.2023.

22. Results for js web frameworks benchmark — round 4. Table Report — 12.09.2016 // Stefan_Krause.blog(). – http:// stefankrause.net/js-frameworks-benchmark4/ webdriver-ts/table.html. Дата обращения: 27.12.2023.

23. GyverPortal. – https://github.com/GyverLibs/GyverPortal?tab=readme-ov-file#gyverportal. Дата обращения: 25.12.2023.

24. VAS Experts. – https://vasexperts.ru/blog/telekom/x86-arm/. Дата обращения: 25.12.2023.

25. AXIOM JDK. – https://axiomjdk.ru/announcements/2022/05/18/what-is-risc-v-and-when-is-the-java-port-coming/?ysclid=ltgy1muosz277766800. Дата обращения: 25.12.2023.

26. Platform.io. – https://platform.io/. Дата обращения: 25.12.2023.

27. Arduino IDE. – https://www.arduino.cc/en/software. Дата обращения: 25.12.2023.

ПРИЛОЖЕНИЯ

Приложение А

Листинг А.1 Исходный код main.cpp

```
// пример использования "блоков" и сетки для оформления

#define  WIFI_MODE  1 /* 0 - STA, 1 - AP*/
#define AP_SSID   "TP-Link_B53F"
#define AP_PASS   "15700933"

void st7565_setup();

#include <GyverPortal.h>
//#include <ECU_types.h>
#include <ECU_data.inc>

#include <Modbus.h>

//
GyverPortal ui;

#define PLOT_SIZE 10
int16_t arr[2][PLOT_SIZE];
const char *names[] = {"Input", "Output",};

int PWD = 0;

//////////////////////////////////////////////////////////////////////////////////
//////////////////
//
int16_t ModbusGetReg(uint16_t addr) {

  return ECU_AccessRegister(addr, 0, 0);
}

void  ModbusSetReg(uint16_t addr, int16_t val) {

  ECU_AccessRegister(addr, val, 1);
}
//////////////////////////////////////////////////////////////////////////////////
// Буфер для работы со строками
char buf[30];

// Формирование строки MAC
static char* MAC_str(Int16* MAC) {
   snprintf(buf, sizeof(buf) - 1, "%02X:%02X:%02X:%02X:%02X:%02X", MAC[0],
MAC[1], MAC[2], MAC[3], MAC[4], MAC[5]);
   return buf;
 }
// Формирование строки IP-адреса
static char* IP_str(Int16* IP) {
   snprintf(buf, sizeof(buf) - 1, "%d.%d.%d.%d", IP[0], IP[1], IP[2],
IP[3]);
   return buf;
 }
// Формирование строки версии
static char* V_str(int V) {
   snprintf(buf, sizeof(buf) - 1, "%d.%d", V >> 8, V & 0xFF);
   return buf;
```

```
        }

       ////////////////////////////////////////////////////////////////////////////
////////
       // Web-интрефейс
       void build() {
         GP.BUILD_BEGIN(GP_DARK);

         // Заголовок страницы в браузере
         GP.PAGE_TITLE("Связь с ЭБУ");
         // проверка связи
         GP.ONLINE_CHECK();

         // создать окно панели управления
         GP.UI_MENU("ЭБУ 10-00.000"/*, GP_RED*/);   // начать меню

         // ссылки меню
         GP.UI_LINK("/", "Доступ, идентфикация");
         GP.UI_LINK("/status", "Мониторинг состояния");
         GP.UI_LINK("/inputs_settings",  "Параметры входов-выходов");
         GP.UI_LINK("/control_settings", "Параметры регуляторов");
         GP.UI_LINK("/plots" , "Оциллограф");
         GP.UI_LINK("/stat"  , "Статистика");
         GP.UI_LINK("/update", "Обновление прошивки");

         // контент
         GP.HR(GP_GRAY);
         GP.LABEL("Текущее время:");
         GP.BREAK();
         GP.LABEL(ui.getSystemTime().encode());
         GP.BUTTON("btnSave", "Сохранить параметры", "", GP_RED);

         // начать основное окно
         GP.UI_BODY();

         // основной интерфейс
         // позволяет "отключить" таблицу при ширине экрана меньше 700px
         GP.GRID_RESPONSIVE(600);

         if (ui.uri("/control_settings")) {
          // Настройка регуляторов
           M_GRID(
             M_BLOCK_TAB(
               "Параметры регуляторов",
               M_BOX(GP.LABEL("Kp"); GP.SPINNER("kP"););
               M_BOX(GP.LABEL("Ki"); GP.SPINNER("kI"););
               M_BOX(GP.LABEL("Kd"); GP.SPINNER("kD"););
             );

             M_BLOCK_TAB(
               "Задание положения",
               GP.SWITCH("Задавать вруную"); GP.BREAK();
               M_BOX(GP.LABEL("Положение"); GP.SLIDER("pos"); );
               M_BOX(GP.BUTTON("posB", "◄"); GP.BUTTON("posF", "►"););
             );
           );

          } else if (ui.uri("/plots")) {

           //M_GRID(
           //M_BLOCK_TAB(
```

```
        // "Осциллограф",
             GP.PLOT_DARK<2, PLOT_SIZE>("table", names, arr);
             //GP.PLOT<2, PLOT_SIZE>("table", names, arr);
          // );
          //);

         } else if (ui.uri("/update")) {

           M_BLOCK_TAB(
             "Обновление прошивки",
             GP.OTA_FIRMWARE();
             GP.OTA_FILESYSTEM();
           );
         }  else {

           M_GRID(
             M_BLOCK_TAB(
               "УРОВЕНЬ ДОСТУПА",
               M_BOX(GP.LABEL("Уровень доступа:"); GP.NUMBER("AL","",
ECU_Id.ACL););
               M_BOX(GP.LABEL("Пароль доступа:");  GP.PASS("PWD"););
               GP.BUTTON("btnPWD", "Запросить доступ");
             );
             M_BLOCK_TAB(
               "ДАННЫЕ УСТРОЙСТВА",
               M_BOX(GP.LABEL("№ партии:"); GP.NUMBER("PN","",
ECU_Id.BatchNo); );
               M_BOX(GP.LABEL("№ серийный:"); GP.NUMBER("SN","",
(ECU_Id.SerialNo)); );
               M_BOX(GP.LABEL("Аппаратная версия:"); GP.TEXT("HW","",
V_str(ECU_Id.HWver)); );
               M_BOX(GP.LABEL("Версия ПО:"); GP.TEXT("SW","",
V_str(ECU_Id.SWver)); );
             );
             M_BLOCK_TAB(
               "ИДЕНТИФИКАЦИЯ",
               M_BOX(GP.LABEL("EMAC:"); GP.TEXT("MAC","", MAC_str(ECU_Id.MAC)););
               M_BOX(GP.LABEL("EIP:");  GP.TEXT("IP","", IP_str(ECU_Id.IP)););
               M_BOX(GP.LABEL("Адрес в сети Modbus:"); GP.NUMBER("MA","",
ECU_Id.ModbusAddr););
               GP.BUTTON("btnID", "Изменить");
             );
           );
         }

         M_BLOCK_TAB(
           "Журнал событий",
           /*M_BOX(*/ GP.AREA_LOG(); /*);*/
         )

         GP.UI_END();    // завершить окно панели управления <<<<-------------
         GP.BUILD_END();
       }

       ////////////////////////////////////////////////////////////////////////
////////
       // Обработка событий Web-интерфейса
       void action() {
         if (ui.click()) {
           ui.log.println(String("Click on ") + ui.clickName() + ' ' +
ui.getString());
           // text PWD
```

```
      if(ui.clickStr("PWD", buf, sizeof(buf)-1)) {
        PWD = String(buf).toInt();
        //ui.log.println(String("PWD=") + ECU_Id.PWD);
      }
      // button PWD
      if(ui.click("btnPWD")) {
        if(PWD == 2222) ECU_Id.ACL = 2; else
        if(PWD == 3333) ECU_Id.ACL = 3; else
        if(PWD == 4444) ECU_Id.ACL = 4;
        ui.log.println(String("запрошен доступ с паролем ") + PWD);
      }
      // button Save
      if(ui.click("btnSave")) {
        ui.log.println("запись параметров в память устройства...");
      }
      // button ID
      // if(ui.click("btnID")) {
      //   ui.log.println("изменены параметры сетевой идентификации...");
      // }
    }

    if(ui.update()) {
      ui.log.print(String("Update of ") + ui.updateName());
    }
  }

  ////////////////////////////////////////////////////////////////////////////
////////
  void setup() {
    Serial.setTimeout(5);
    Serial.begin(921600);
    while(!Serial); // wait for serial
    Serial.println("Starting AP...");
   #if WIFI_MODE == 0 // Рабочая станция
    WiFi.mode(WIFI_STA);
    WiFi.begin(AP_SSID, AP_PASS);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println(WiFi.localIP());
  #else // Точка доступа
    WiFi.mode(WIFI_AP);
    WiFi.softAP("ECU-0001");//, "12345678", 12, false);
    Serial.println("AP started");
  #endif

    ui.attachBuild(build);
    ui.attach(action);
    ui.start();
    ui.log.start(80);   // размер буфера
    //ui.enableOTA("admin", "pass");  // с паролем

    //st7565_setup();
  }
  //
==============================================================================
  void loop() {
    ui.tick();
    ModbusPoll();
  }
```

Приложение Б
Листинг Б.1 Данные для заполнения страниц

```
        // Определения структур и функций для обмена данными с ЭБУ

        /*
          Пример использования:
          char buf[1024];        // буфер может быть объявлен динамически
          ECU_ID_data* P = buf;  // указатель для чтения/записи структуры
ECU_ID_data
          ModbusReadMultiple(buf, 00001, sizeof(ECU_ID_data)); // Чтение блока в
буфер
          // Работа с блоком
          P->ModbusAddr = 12;
          printf("Версия аппаратной части: %d.%d", P->HWver >> 8, P->HWver &
0xFF);
        */

        #ifndef ECU_TYPES_H // защита от повтора
        #define ECU_TYPES_H

        #ifdef __cplusplus
        extern "C" {
        #endif

        // Определения базовых типов
        typedef signed int     Int32;
        typedef signed short   Int16;

        // Параметры идентификации, безопасный режим
        // Начальный адрес 00001
        typedef struct {
            Int16 ModbusAddr;   // Адрес на шине modbus
            Int16 IP[4];        //
            Int16 MAC[6];       //
            Int16 SRKey;        // 5555 - сохранение настроек, 1111 - сброс
настроек
            Int16 ACL;          // Уровень доступа
            Int16 L2Pwd;        // Пароль для 2-го уровня доступа
            Int16 L3Pwd;        // Пароль для 3-го уровня доступа
            Int16 L4Pwd;        // Пароль для 4-го уровня доступа
            Int16 BatchNo;      // Номер партии
            Int16 dummy1;        // Выравнеивание
            Int32 SerialNo;     // Серийный номер
            Int16 HWver;        // Версия аппаратной части
            Int16 SWver;        // Версия встроенного ПО
            Int16 dummy2;       // Резерв
            Int16 GuardTime;    // Время до перехода в безопасный режим, мс
            Int16 State;        // Текущее состояние 0...4
            Int16 ErrorCode;    // Код неисправности
            Int16 ErrorFlags;   // Флаги неисправностей
            Int16 GuardAction;  // Действие при переходе в безопасный режим
            float GuardPos;
        } ECU_ID_data;

        // Параметры управления
        // Начальный адрес 00038
        typedef struct {
            Int16 CmdSource;    // Интерфейс управления
            Int16 dummy;        // Выравнивание
            Int16 OperMode;     // Режим работы 0...4, 9999
```

```
        Int16 RefMode;      // Форма сигнала генератора задания
        float RefFreq;      // Частота сигнала генератора задания
        float HCposRef;     // Задание положения ГЦ
        float SpoolPosRef;  // Задание положения золотника
    } ECU_control_params;

    // Параметры регуляторов (определяются при разработке)
    // адреса 00048...00078
    typedef struct {
        Int16 P[30];//
    } ECU_reg_params;

    // Параметры текущего состояния
    // Начальный адрес 00080
    typedef struct {
        float Vdc;          // Напряжение питания
        float Temc;         // Температура обмотки ЭМП
        float Tecu;         // Температура контроллера ЭБУ
        float HCpos;        // Среднее положение штока гидроцилиндра
        float HCstr;        // Амплитуда колебаний (ход) штока гидроцилиндра
        float SPpos;        // Среднее положение золотника
        float SPstr;        // Амплитуда колебаний золотника
        float HCspd;        // Скорость штока гидроцилиндра
        float SPspd;        // Скорость золотника
        float Iemc;         // Ток в обмотке ЭМП
        float Vemc;         // Напряжение на якоре ЭМП
    } ECU_state_data;

    // Конфигурация задающего входа
    // Начальный адрес 00106 - вход задания положения штока ГЦ
    // Начальный адрес 00146 - вход задания положения золотника
    typedef struct {
        float Scale;        // Масштабный коэффициент
        Int16 Select;       // Выбор входа
        Int16 dummy1;       // Выравнивание
        float Ref;          // Текущее задание
        float Offset;       // Смещение (установка 0)
        float MaxRef;       // Верхний предел задания
        float MinRef;       // Нижний предел задания
        float RaiseTime;    // Время нарастания, мс (ramp)
        float FallTime;     // Время спада, мс (ramp)
        float FiltFreq;     // Частота среза фильтра, Гц
        float Vmin;         // Нижняя граница входа по напряжению
        float VminErr;      // Нижняя граница входа по напряжению для
индикации ошибки
        float Vmax;         // Верхняя граница входа по напряжению
        float VmaxErr;      // Верхняя граница входа по напряжению для
индикации ошибки
        float Imin;         // Нижняя граница входа по току
        float IminErr;      // Нижняя граница входа по току для индикации
ошибки
        float Imax;         // Верхняя граница входа по току
        float ImaxErr;      // Верхняя граница входа по току для индикации
ошибки
    } ECU_ref_control;

    // Конфигурация аналолгового выхода
    // Начальный адрес 00200 - выход 1
    // Начальный адрес 00210 - выход 2
    typedef struct {
        Int16 Mode;         // Режим и диапазон выходного сигнала
```

```
    Int16 dummy1;       // Выравнивание
    float Gain;         // Усиление
    float Offset;       // Смещение
    Int16 Select;       // Выбор сигнала
} ECU_output_control;

// Дитеринг (шум)
// Начальный адрес 00300
typedef struct {
    Int16 Type;
    Int16 dummy;
    float Freq;
    float Value;
    float Min;
    float Max;
} ECU_dither_control;

// Статистика
// Начальный адрес 01000 - текуий сеанс
// Начальный адрес 01500 - общая статистика
typedef struct {
    Int32 OnTime;       // Время работы
    Int32 Stroke;       // Общий ход поршня
    float Current;      // Средний ток ЭМП
    float Power;        // Средняя мощность
} ECU_statistics;

// RAW данные для калибровки
// Начальный адрес 02000
typedef struct {
    Int16 ADC1;
    Int16 HCpos;
    Int16 SpoolPos;
    Int16 Vdc;
    Int16 V12;
    Int16 Iemc;
    Int16 ADC7;
    Int16 Tmcu;
    Int16 DAC1;
    Int16 DAC2;
    Int16 HCposC;
} ECU_raw_data;

// Калибровочные данные
// Начальный адрес 3000
typedef struct {
    float K1;
    float C1;
    float K2;
    float C2;
    float K3;
    float C3;
    float K4;
    float C4;
    float K5;
    float C5;
    float K6;
    float C6;
    float K7;
    float C7;
    float K8;
```

```
        float C8;
    } ECU_calib_values;

    // Скрытые параметры управления
    // Начальный адрес 3300
    typedef struct {
        Int16 Param[100];
    } ECU_internal_params;

    // Отладочные данные
    // Начальный адрес 5000
    typedef struct {
        Int16 Reg[100];
    } ECU_debug_data;

    // Архив
    // Начальный адрес 20000
    typedef struct {
        Int16 Record[100];
    } ECU_archive_data;

    // Параметры записи осциллограмм
    // Начальный адрес 30000
    typedef struct {
        Int16 Channel1;     // Первая координата, выводимая в буфер
осциллограммы
        Int16 Channel2;     // Вторая координата, выводимая в буфер
осциллограммы
        Int16 Channel3;     // Третья координата, выводимая в буфер
осциллограммы
        Int16 Channel4;     // Четвертая координата, выводимая в буфер
осциллограммы
        Int16 SampleTime;   // Время между выборками , мкс
        Int16 BufferSize;   // Количество выборок в буфере
        Int16 PageToRead;   // Номер страницы для считывания из буфера
    } ECU_scope_control;

    // Начальные адреса
    enum {
        ECU_BASE_ID             = 1,    // Идентификация
        ECU_BASE_CONTROL        = 38,   // Управление
        ECU_BASE_REG_PARAM      = 48,   // Параметры регуляторов
        ECU_BASE_REF_HC         = 106,  // Вход положения ГЦ
        ECU_BASE_REF_SPOOL      = 146,  // Вход положения золотника
        ECU_BASE_OUT1           = 200,  // Аналоговый выход 1
        ECU_BASE_OUT2           = 210,  // Аналоговый выход 2
        ECU_BASE_DITHER         = 300,  // Дитеринг
        ECU_BASE_CURRENT_STAT   = 1000, // Текущая статистика
        ECU_BASE_GLOABL_STAT    = 1500, // Общая статистика
        ECU_BASE_RAW_DATA       = 2000, // Данные для выполнения калибровки
        ECU_BASE_CALIB_VALUES   = 3000, // Калибровочные коэффициенты
        ECU_BASE_INTERNAL       = 3300, // Скрытые параметры управления
        ECU_BASE_DEBUG          = 5000, // Отладочные данные
        ECU_BASE_ARCHIVE        = 20000,// Архив событий
        ECU_BASE_SCOPE_CONTROL  = 30000,// Параметры осциллограммы
        ECU_BASE_SCOPE_BUFFER   = 32768 // Буфер осциллограммы
    } ECU_BASE_ENUM; //

    #ifdef __cplusplus
    }
    #endif

    #endif // ECU_TYPES_H
```