МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «**Московский технический университет связи и информатики**» (**МТУСИ**)

Кафедра «Информационные системы и сетевые технологии»

ОТЧЁТ

По учебной практике технологической

По теме: **«Настраиваемая, трехдиапазонная, звуковая сигнализация в зависимости от веса объекта»**

Выполнил студент группы УБСТ2304:

Паокин А. И.

Проверил: Кандзюба Е.В.

Москва 2025

Содержание

Введение 3
1. Техника безопасности 4
2. Проектирование прототипа устройства 6
2.1 Описание предметной области 6
2.2 Извлечение именованных сущностей 8
2.3 Создание диаграммы «Сущность-связь» 10
2.4 Определение составных частей системы 11
2.5 Выбор симуляторов для аппаратно-программных средств семейства Arduino 12
2.5.1 Инсталляция на устройство 14
3. Разработка прототипа устройства 17
3.1 Выбор электронных компонентов, необходимых для функционирования прототипа устройства и составление схемы 17
3.2 Создание базы данных 20
3.3 Проектирование и реализация графического интерфейса 23
Заключение 27

Введение

В современном мире автоматизация процессов контроля и управления играет ключевую роль в повышении эффективности и безопасности различных систем. Одной из таких систем является контроль веса транспортных средств перед проездом по мостам, что позволяет предотвратить перегрузку конструкций и обеспечить их долговечность. В данном отчёте представлена разработка прототипа настраиваемой трёхдиапазонной звуковой сигнализации, которая реагирует на вес объекта и обеспечивает оперативное оповещение о возможных перегрузках.

Целью учебной практики стало проектирование и реализация системы, включающей аппаратную часть на базе микроконтроллера ESP32, базу данных для учёта взвешиваний и графический интерфейс для взаимодействия с пользователями. В ходе работы были рассмотрены вопросы техники безопасности при эксплуатации оборудования, проведено проектирование сущностей базы данных, выбор электронных компонентов и симуляторов для отладки, а также реализован прототип устройства с интеграцией в облачное хранилище Firebase.

Данный отчёт отражает основные этапы работы, начиная с анализа предметной области и заканчивая созданием полноценного решения, сочетающего аппаратные и программные компоненты. Результаты проекта могут быть применены в реальных условиях для мониторинга весовых параметров транспортных средств и предотвращения перегрузки мостовых сооружений.

1. Техника безопасности

При эксплуатации персонального компьютера на работника могут оказывать действие следующие опасные и вредные производственные факторы:

- повышенный уровень электромагнитных излучений;
- повышенный уровень статического электричества;
- пониженная ионизация воздуха;
- статические физические перегрузки;
- перенапряжение зрительных анализаторов.

Требования безопасности перед началом работы

Подготовить рабочее место. Отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране. Проверить правильность подключения оборудования к электросети. Проверить исправность проводов питания и отсутствие оголенных участков проводов. Убедиться в наличии заземления системного блока, монитора и защитного экрана. Протереть антистатической салфеткой поверхность экрана монитора и защитного экрана.

Проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение "мыши" на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

Требования безопасности во время работы

Работнику при работе на ПК запрещается:

- прикасаться к задней панели системного блока (процессора) при включенном питании;

• переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;

• допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств;

• производить самостоятельное вскрытие и ремонт оборудования;

• работать на компьютере при снятых кожухах;

• отключать оборудование от электросети и выдергивать электровилку, держась за шнур.

Продолжительность непрерывной работы с компьютером без регламентированного перерыва не должна превышать 2-х часов.

Во время регламентированных перерывов с целью снижения нервно - эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления выполнять комплексы упражнений.

Требования безопасности по окончании работы

Отключить питание компьютера. Привести в порядок рабочее место. Выполнить упражнения для глаз и пальцев рук на расслабление.

2. Проектирование прототипа устройства

2.1 Описание предметной области

Система учета весов перед мостами для взвешивания транспорта представляет собой комплексное решение, предназначенное для автоматизации процессов контроля веса грузовых транспортных средств перед их проездом по мостовым сооружениям. Основная цель системы - предотвращение перегрузки мостовых конструкций путем точного измерения массы транспортных средств и оперативного реагирования на случаи превышения допустимых норм.

Работа системы начинается с момента въезда транспортного средства на весовую платформу, где происходит автоматическое или операторское измерение весовых параметров. Система фиксирует несколько ключевых показателей: полную массу автомобиля с грузом (брутто), массу пустого транспортного средства (тара). Эти данные сразу же сопоставляются с установленными нормативными значениями для конкретного мостового перехода. В случае обнаружения потенциально опасной перегрузки система может инициировать различные сценарии реагирования - от простого предупредительного сообщения до автоматической блокировки проезда через шлагбаум или отправки уведомления ответственным лицам.

Важной составляющей системы является модуль учета клиентов и транспортных средств, который позволяет хранить и оперативно использовать информацию о регулярных перевозчиках. База данных содержит не только основные реквизиты организаций и индивидуальных предпринимателей, но и историю их предыдущих взвешиваний, что значительно ускоряет процесс обработки повторных заездов. Для каждого зарегистрированного в системе клиента доступна полная статистика по количеству проездов, средним весовым показателям и случаям перегруза.

Управление весовым оборудованием представляет собой еще один критически важный аспект системы. Каждая весовая установка имеет цифровой паспорт с указанием технических характеристик, включая максимальную грузоподъемность, даты последней поверки и калибровки, а также точного местоположения. Система автоматически отслеживает необходимость проведения очередных метрологических проверок и формирует соответствующие напоминания для технического персонала.

Техническое обслуживание весового оборудования регистрируется в системе с детализацией всех выполненных работ: от плановых поверок и калибровок до аварийных ремонтов. Каждая запись содержит исчерпывающую информацию о дате проведения работ, конкретном типе обслуживания, ответственном техническом специалисте и перечне выполненных операций. Эта информация не только обеспечивает прозрачность технического обслуживания, но и служит основой для анализа надежности оборудования.

Процедура взвешивания в системе максимально оптимизирована, оператор говорит завести на весы автомобиль, после взвевания и показания результатов, если звуковая сигнализация не предупреждает о перевесе, то данные отправляться в базу данных откуда попадают в пользовательский интерфейс для учёта взвешиваний.

В систему взвешивания перед мостами интегрирован специализированный модуль на базе Arduino, выполняющий функцию звукового оповещения о перегрузе. Этот аппаратный компонент представляет собой компактное устройство, подключенное непосредственно к весовым датчикам и работающее в реальном времени.

Принцип работы модуля основан на непрерывном мониторинге поступающих с тензодатчиков сигналов. Микроконтроллер Arduino обрабатывает эти данные, сравнивая текущие показания веса с заранее запрограммированными пороговыми значениями, установленными для конкретного мостового перехода. В момент обнаружения превышения

допустимой массы модуль активирует звуковую сигнализацию - громкий прерывистый или постоянный сигнал, хорошо различимый даже в условиях шумной окружающей среды.

Звуковое оповещение реализовано через динамик, подключенный к цифровому выходу Arduino. Длительность и характер звукового сигнала (непрерывный/прерывистый) программно адаптированы под конкретные условия эксплуатации. Модуль предусматривает возможность ручной настройки порогов предельной нагрузки через панель устройства.

Разграничение прав доступа реализовано через систему ролей, где каждый пользователь имеет строго определенный набор полномочий. Операторы отвечают за непосредственное проведение взвешиваний, технические специалисты - за обслуживание оборудования, менеджеры - за анализ статистики и формирование отчетов, а администраторы - за общее управление системой и настройку ее параметров.

Аналитический модуль системы предоставляет широкие возможности для генерации различных отчетов: от стандартных ведомостей по количеству взвешиваний за определенный период до сложных аналитических выборок, показывающих динамику нагрузок, статистику перегрузов или загруженность отдельных весовых пунктов. Эти данные позволяют не только контролировать текущую ситуацию, но и прогнозировать нагрузку на мостовые сооружения в перспективе.

2.2 Извлечение именованных сущностей

Система учета весов перед мостами включает следующие сущности: информация о весовом оборудовании, данные клиентов-перевозчиков, учетные записи сотрудников, результаты взвешиваний, а также записи о техническом обслуживании. Для обработки данных с датчиков используется структура, которая содержит информацию о весе, событиях и временных метках. Эти

данные могут преобразовываться в записи взвешиваний или технического обслуживания.

Ключевые особенности системы: связи между сущностями обеспечивают целостность данных, типы данных подобраны для точного учета (например, десятичные числа для веса), а также предусмотрена гибкость — структуру данных с датчиков можно расширить для интеграции с дополнительными сенсорами.

Выделенные сущности представлены в таблице 2.2.1.

Таблица 2.2.1 – Выделенные сущности из предметной области

Сущность	Наименование	Тип данных	Описание
Scales	ScaleID	int (PK)	Уникальный идентификатор весов.
	Model	string	Модель весов (например, "Весы мостовые 30т").
	SerialNumber	string (Unique)	Серийный номер оборудования.
	MaxCapacity	decimal	Максимальная грузоподъемность в кг.
	InstallationDate	DateTime?	Дата установки весов.
	LastCalibrationDate	DateTime?	Дата последней поверки/калибровки.
	Location	string	Место установки (например, "Склад №1").
	IsActive	bool	Флаг активности (включены/выключены).
Clients	ClientID	int (PK)	Уникальный ID клиента.
	Name	string	Название организации или ФИО перевозчика.
	Phone	string	Контактный телефон.
	Email	string	Электронная почта.
Users	UserID	int (PK)	Уникальный ID пользователя.
	Username	string (Unique)	Логин для входа в систему.
	Password	string	Зашифрованный пароль.
	FullName	string	Полное имя пользователя.
	Role	string	Роль (Admin, Operator, Technician, Manager).
	IsActive	bool	Флаг активности учетной записи.
Weighings	WeighingID	int (PK)	Уникальный ID взвешивания.
	ScaleID	int (FK Scales)	Ссылка на весы, на которых проводилось взвешивание.
	ClientID	int? (FK Clients)	Ссылка на клиента (если известен).
	VehicleNumber	string	Номер транспортного средства.
	GrossWeight	decimal	Полный вес ТС с грузом (брутто).
	TareWeight	decimal	Вес пустого ТС (тара).
	WeighingDateTime	DateTime	Дата и время взвешивания.
	OperatorID	int (FK Users)	ID оператора, проводившего взвешивание.

Продолжение таблицы 2.2.1

Maintenance	Notes	string	Дополнительные примечания (тип груза и т.д.).
	MaintenanceID	int (PK)	Уникальный ID записи о техническом обслуживании.
	ScaleID	int (FK Scales)	Ссылка на весы, которые обслуживались.
	MaintenanceDate	DateTime	Дата проведения ТО/ремонта.
	MaintenanceType	string	Тип работ (Поверка, Ремонт, Техобслуживание).
	TechnicianID	int (FK Users)	ID техника, выполнявшего работы.
	Description	string	Подробное описание выполненных работ.
SensorData	Event	string?	Тип события (например, "weight_measurement").
	Range1	int	Первый диапазон измерения (для калибровки).
	Range2	int	Второй диапазон измерения.
	Timestamp	int	Временная метка в Unix-формате.
	Weight	int	Текущий вес с датчика (в кг или г).

2.3 Создание диаграммы «Сущность-связь»

Диаграмма сущность-связь, разработанная в Visio 2016, представляет собой модель системы учета весового контроля перед мостами. Она включает ключевые компоненты, такие как данные пользователей, сведения о весовом оборудовании, информацию о клиентах-перевозчиках, результаты взвешиваний и записи технического обслуживания.

Модель обеспечивает целостность данных за счет четких связей между сущностями. Пользователи системы имеют различные роли и уровни доступа, что позволяет разграничивать их функции. Весовое оборудование характеризуется техническими параметрами, включая максимальную нагрузку и даты калибровки, что важно для точного учета. Клиенты, зарегистрированные в системе, связаны с процессами взвешивания, что упрощает контроль за перевозками. Каждое взвешивание фиксирует вес транспортного средства, время операции и ответственного оператора, обеспечивая прозрачность данных. Техническое обслуживание весов регистрируется с указанием типа работ, даты и исполнителя, что способствует поддержанию оборудования в рабочем состоянии.

Схема служит основой для проектирования базы данных и дальнейшего развития системы, обеспечивая удобство управления информацией и автоматизацию ключевых процессов. Диаграмма «Сущность связь» изображена на рисунке 2.3.1.

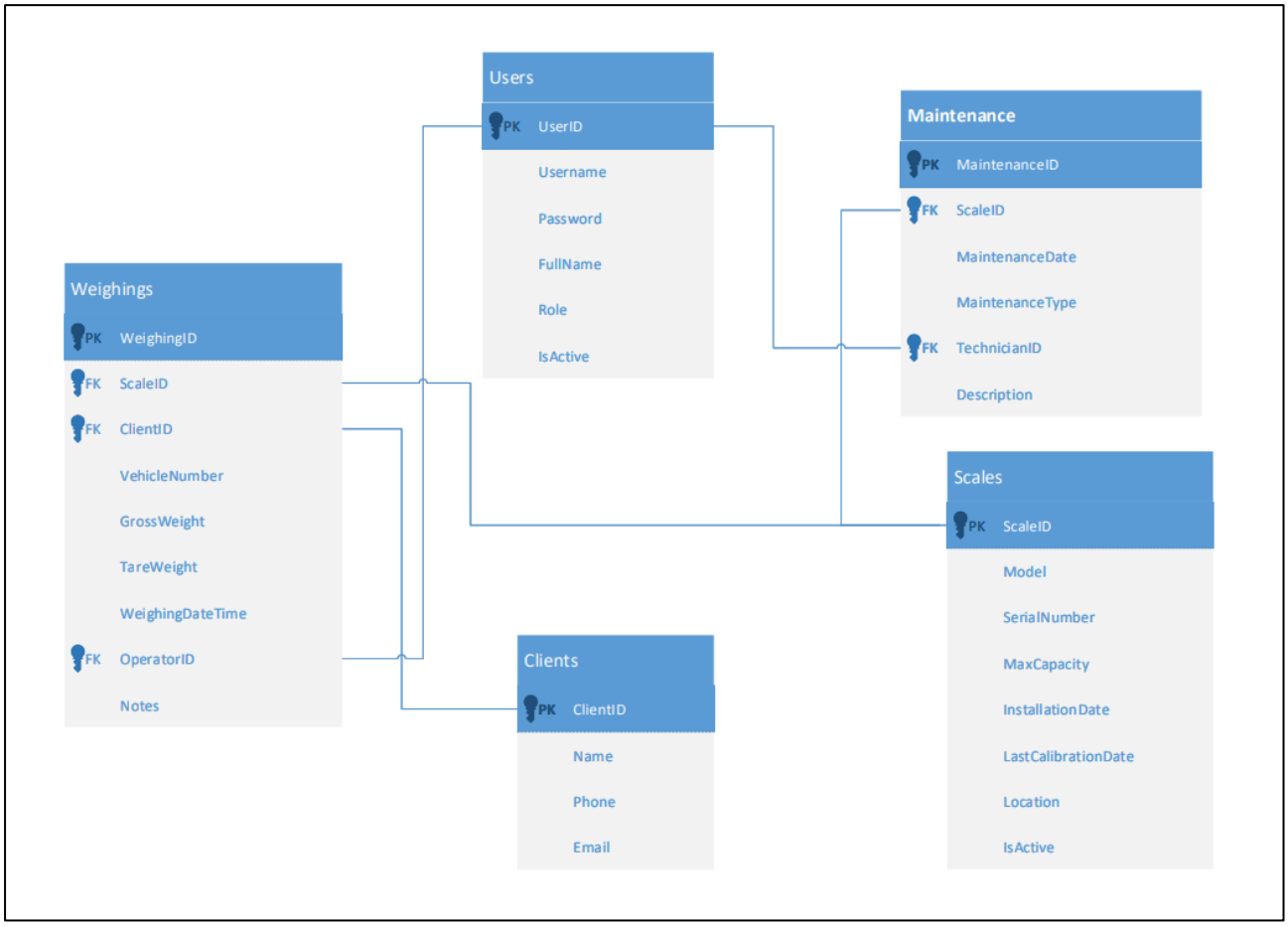

Рисунок 2.3.1 – Диаграмма «Сущность связь»

2.4 Определение составных частей системы

Разработка прототипа макета настраиваемой трёхдиапазонной звуковой сигнализации веса требует тщательного подбора электронных компонентов, обеспечивающих точность измерений, гибкость управления и наглядность отображения данных.

Основой системы является микроконтроллер ESP32, который объединяет все модули в единую схему. Для измерения веса используется потенциометр, имитирующий работу тензодатчика, а звуковая индикация реализована с помощью зуммера, подающего сигналы разной тональности в зависимости от

заданных диапазонов. Основой системы является микроконтроллер ESP32 DevKit C v4, выбранный благодаря своей универсальности и наличию встроенного модуля Wi-Fi. Это позволяет не только обрабатывать данные локально, но и отправлять их в облако для дальнейшего анализа. Для имитации датчика веса используется потенциометр, подключённый к аналоговому входу ESP32. В реальных условиях его можно заменить на тензодатчик, что делает систему масштабируемой.

Управление настройками системы осуществляется с помощью четырёх кнопок. Кнопки позволяют регулировать границы диапазонов, кнопка. Для визуализации информации используется OLED-дисплей SSD1306, который отображает текущий вес, границы диапазонов и режим работы.

Звуковая сигнализация реализована с помощью зуммера, который генерирует тоны разной частоты в зависимости от текущего диапазона веса. Это обеспечивает мгновенную обратную связь с пользователем. Переключение между режимами настройки и измерения осуществляется с помощью двух слайд-переключателей, что делает интерфейс интуитивно понятным.

Данная система демонстрирует эффективное сочетание аппаратных и программных средств для создания гибкого и функционального решения. Использование ESP32 обеспечивает не только локальную работу, но и возможность интеграции в более крупные системы мониторинга. Проект может быть доработан путём замены потенциометра на реальный датчик веса, что расширит его применение в промышленных и бытовых сценариях.

2.5 Выбор симуляторов для аппаратно-программных средств семейства Arduino

При выборе симулятора для работы с Arduino важно учитывать несколько ключевых факторов в частности: поддержку различных плат, функциональность, удобство интерфейса, доступность и наличие документации. Ниже рассмотрим несколько популярных средств.

SimulIDE

SimulIDE предлагает поддержку Arduino на базе AVR, а также других микроконтроллеров. Главное преимущество — возможность оффлайн-работы, что полезно при отсутствии стабильного интернета. Однако интерфейс программы выглядит устаревшим, а документация не всегда подробна. Кроме того, отсутствует облачное сохранение проектов, что усложняет совместную работу.

Wokwi

Wokwi выделяется среди онлайн-симуляторов благодаря своей универсальности. Он поддерживает не только классические платы вроде Arduino Uno и Mega, но и более современные решения, такие как ESP32. Симулятор предлагает удобную отладку с подсветкой ошибок, логированием и визуализацией работы схемы. Интерфейс интуитивно понятен, а возможность делиться проектами упрощает командную работу. Единственный минус — зависимость от интернета, но для большинства пользователей это не критично.

UnoArduSim

Этот симулятор ориентирован исключительно на Arduino Uno и давно не обновлялся. Его простота может быть полезна для самых базовых задач, но отсутствие поддержки других плат и ограниченный функционал делают его непрактичным для серьезной работы.

PicsimLab

PicsimLab поддерживает не только Arduino, но и микроконтроллеры PIC, что может быть полезно в некоторых сценариях. Однако интерфейс программы сложен для новичков, а скорость эмуляции оставляет желать лучшего. Недостаток документации также усложняет освоение.

Tinkercad Circuits

Tinkercad Circuits от Autodesk — хороший вариант для начинающих благодаря простому визуальному редактору схем. Однако симулятор работает

медленно и не поддерживает многие продвинутые библиотеки, что ограничивает его применение в сложных проектах.

Итог

Наиболее сбалансированным вариантом является Wokwi. Он сочетает в себе широкую поддержку плат, удобный интерфейс, мощные инструменты отладки и возможность совместной работы. Если требуется оффлайн-решение, можно рассмотреть SimulIDE, а для самых простых задач подойдет Tinkercad Circuits. Однако для профессионального использования и обучения Wokwi остается лучшим выбором.

2.5.1 Инсталляция на устройство

Для удобной работы с симулятором Wokwi в Visual Studio Code при использовании PlatformIO потребуется выполнить несколько шагов. Этот подход позволяет разрабатывать, тестировать и отлаживать код для Arduino прямо в виртуальной среде без необходимости подключения реального устройства.

Описание инсталляции

Первым делом необходимо установить Visual Studio Code — это основной редактор кода, в котором будет вестись работа. Далее в расширениях VSCode нужно найти и установить PlatformIO IDE. Это мощный инструмент для разработки под различные микроконтроллеры, включая платы Arduino. PlatformIO предоставляет все необходимые средства для компиляции и загрузки кода, а также богатый набор библиотек.

Для интеграции Wokwi потребуется установить Node.js, так как симулятор использует собственный интерфейс командной строки (CLI), работающий на этой платформе. После установки Node.js можно переходить к установке Wokwi CLI через встроенный в VSCode терминал. Этот инструмент позволит запускать симуляцию прямо из среды разработки.

В PlatformIO создается новый проект с выбором нужной платы Arduino, например, Uno или Nano. После инициализации проекта в его корневой папке

необходимо создать конфигурационный файл для Wokwi. В этом файле указывается версия симулятора, путь к скомпилированному прошивочному файлу и, при необходимости, перечень виртуальных компонентов, которые должны быть в схеме.

После написания кода его нужно скомпилировать через PlatformIO. Затем, используя ранее установленный Wokwi CLI, можно запустить симуляцию. Симулятор откроется в браузере, где будет виртуальная копия выбранной платы Arduino со всеми указанными в конфигурации компонентами. В этом режиме можно наблюдать за выполнением программы, проверять состояние пинов, читать данные с виртуальных датчиков и взаимодействовать с другими элементами схемы.

Основное преимущество этой связки — возможность быстрой проверки идей без необходимости возиться с реальными компонентами. Особенно это полезно на этапе обучения или при разработке сложных проектов, где важно сначала проверить логику работы. Кроме того, симулятор позволяет легко демонстрировать проекты другим людям — достаточно просто поделиться ссылкой.

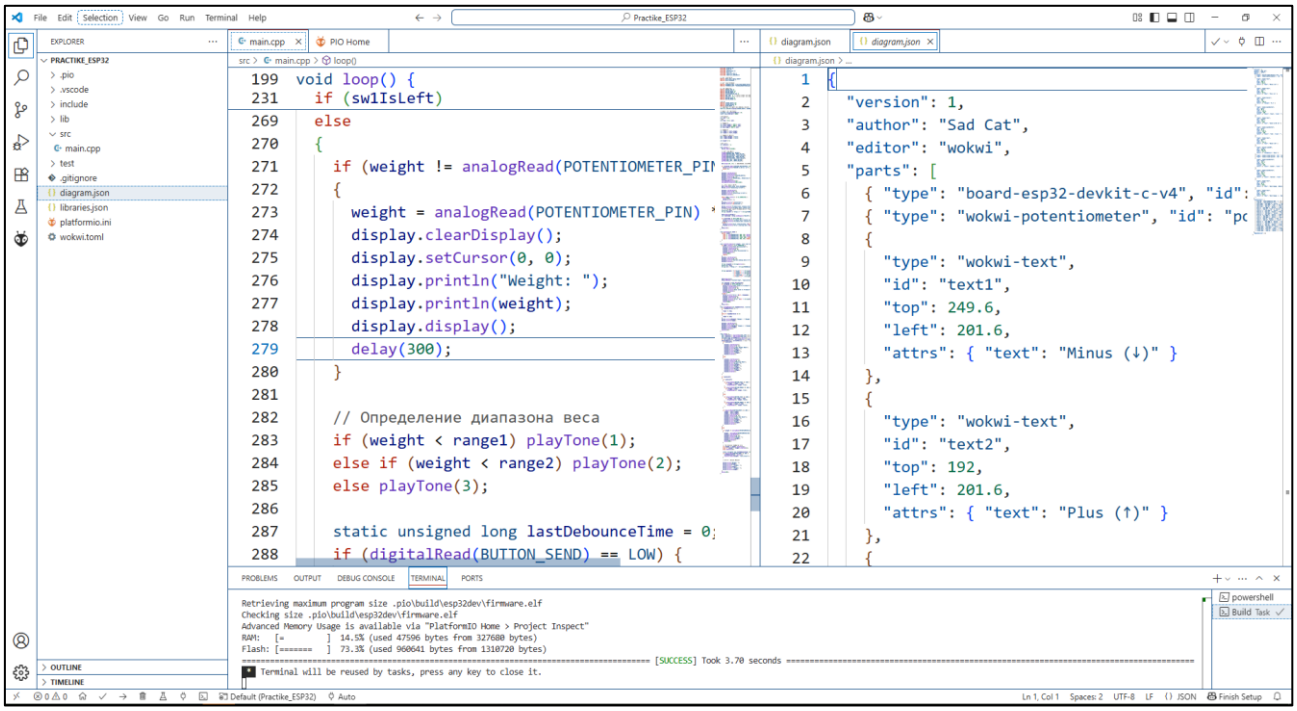

Рисунок 2.5.1 – Среда разработки Visual Studio Code

Интеграция Wokwi с Visual Studio Code через PlatformIO создает удобную среду для разработки проектов на Arduino. Такой подход экономит время на сборку физических прототипов и позволяет сосредоточиться на программировании и отладке логики работы устройства. Всё, что нужно — это немного настроить окружение, после чего можно сразу приступать к созданию и тестированию виртуальных устройств.

На рисунке 2.5.1 изображена среда разработки Visual Studio Code установленными расширениями и подключенными акантами для разработки в Wokwi.

3. Разработка прототипа устройства

3.1 Выбор электронных компонентов, необходимых для функционирования прототипа устройства и составление схемы

Основным управляющим модулем выбран микроконтроллер ESP32 DevKit C v4. Его преимущества включают встроенные Wi-Fi и Bluetooth, что позволяет отправлять данные в базу данных или управлять системой удаленно, а также большое количество GPIO-пинов для подключения периферии. Альтернативой мог бы стать Arduino Uno, но он менее функционален, или STM32, который мощнее, но сложнее в программировании. Вся схема элементов изображена на рисунке 3.1.1.

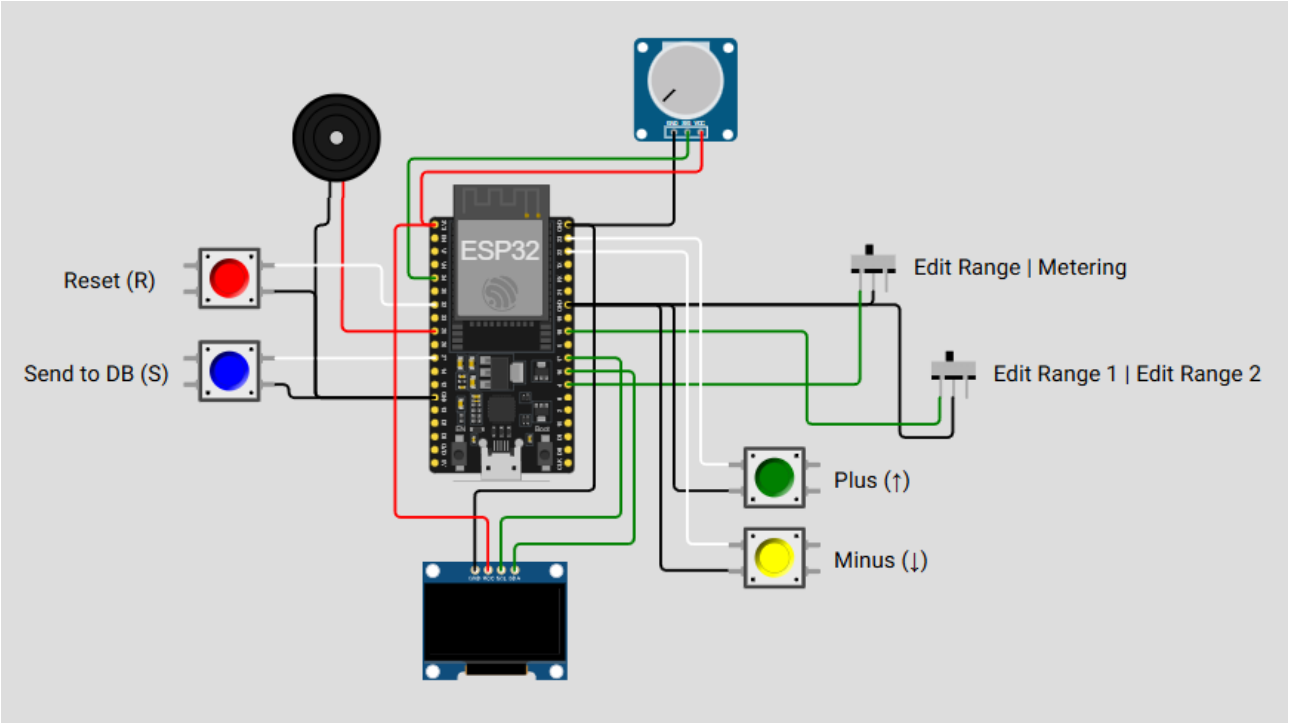

Рисунок 3.1.1 – Схема подключения элементов к микроконтроллеру ESP32

Потенциометр в схеме служит для регулировки чувствительности или пороговых значений веса. Он выбран из-за простоты использования и возможности аналогового считывания, что удобно для настройки в реальном времени. В качестве альтернативы можно было бы использовать цифровой

энкодер, но он сложнее в подключении и обработке сигналов. Зуммер выполняет функцию звукового оповещения при достижении заданного веса. Пассивный зуммер удобен тем, что управляется через ШИМ (PWM), что позволяет гибко настраивать тон и частоту звука. Динамик или пьезоэлемент могли бы стать альтернативами, но первый требует усилителя, а второй обычно тише.

Кнопки в схеме выполняют различные функции: синяя кнопка отправляет данные в базу данных, красная сбрасывает систему, а зеленая и желтая используются для увеличения и уменьшения значений. Тактические кнопки выбраны из-за их надежности и простоты использования. Сенсорные кнопки могли бы стать альтернативой, но они менее надежны в некоторых условиях эксплуатации. OLED-дисплей отображает текущий вес, пороговые значения и статус системы. Его преимущества — высокая контрастность, низкое энергопотребление и интерфейс I2C, который экономит пины микроконтроллера. LCD-дисплей дешевле, но требует больше энергии и места.

Переключатели в схеме позволяют выбирать режимы работы: один переключатель отвечает за переход между редактированием диапазона и измерением, а второй — за выбор между диапазонами 1 и 2. Слайд-переключатели удобны благодаря фиксированным положениям, что делает их надежными для выбора режимов. Альтернативой могли бы быть DIP-переключатели, но они менее удобны для частого использования. Текстовые метки на схеме служат для наглядности, подписывая функции кнопок и переключателей. Они не требуют подключения к электрической части, но значительно упрощают понимание схемы.

Соединения в схеме организованы таким образом, чтобы обеспечить стабильную работу всех компонентов. Потенциометр подключен к аналоговому входу ESP32 (пин 34), кнопки — к цифровым входам с подтяжкой к земле, зуммер управляется через ШИМ (пин 25), а OLED-дисплей использует интерфейс I2C (пины 16 и 17). Переключатели подключены к цифровым пинам 4 и 18. В целом, схема хорошо продумана и обеспечивает гибкость для дальнейшего расширения,

например, добавления новых датчиков или функций. Все элементы выбраны с учетом баланса между функциональностью, простотой и стоимостью, что делает систему удобной для настройки и использования. Схема изображённая на рисунке 3.1.1 создана с помощью симуляторf электроники, для моделирования работы с микроконтроллерными платами Wokwi.

Описание программного кода

Код реализует систему звуковой сигнализации на базе микроконтроллера ESP32, которая реагирует на вес объекта, измеряемый через потенциометр. В начале программы выполняется инициализация всех компонентов: настраиваются пины для подключения зуммера, кнопок управления, потенциометра и OLED-дисплея, а также устанавливается соединение с Wi-Fi сетью "Wokwi-GUEST" и сервисом Firebase для хранения данных. Основная логика работы системы построена вокруг двух настраиваемых диапазонов веса (range1 и range2) и максимального значения веса (max_range). Пользователь может изменять границы этих диапазонов с помощью кнопок "+" и "-", причем кнопка "Reset" позволяет вернуть исходные значения. Переключатели SW1 и SW2 используются для выбора режима работы: настройки диапазонов или непосредственного измерения веса.

Потенциометр, подключенный к аналоговому входу, имитирует датчик веса. В зависимости от текущего значения веса система активирует разные звуковые сигналы через зуммер: если вес ниже первого диапазона, воспроизводится короткий звук частотой 600 Гц; если вес попадает в первый диапазон, звучит двойной сигнал 800 Гц; а если вес превышает второй диапазон, включается продолжительный сигнал 1000 Гц. Кнопка "Send" позволяет отправить текущие данные (вес, границы диапазонов и метку времени) в базу данных Firebase в формате JSON с уникальным идентификатором. Интерфейс системы включает OLED-дисплей, который отображает текущий вес, установленные диапазоны и режим работы, а также серийный монитор для вывода отладочной информации.

Таким образом, код обеспечивает гибкую настройку параметров сигнализации, визуализацию данных и возможность их сохранения в облачном хранилище, что делает систему удобной для мониторинга веса объектов со звуковым оповещением.

На рисунке 3.1.1 изображена часть кода, отправляющего Json – документ в FireBase Realtime Database.

```
String uniqueId = String(millis());
HTTPClient http;
String url = "https://" + String(FIREBASE_HOST) + "/sensorData/idScales"
+String(idScales)+"/"+String(uniqueId)+".json?auth=" + String(FIREBASE_AUTH);


String payload = "{\"weight\":" + String(weight) +
                 ",\"range1\":" + String(range1) +
                 ",\"range2\":" + String(range2) +
                 ",\"timestamp\":" + String(uniqueId) +
                 ",\"event\":\"" + String(eventType) + "\"}";

http.begin(url);
http.addHeader("Content-Type", "application/json");
```

Рисунок 3.1.2 – Часть кода отправка Json – документа в FireBase

3.2 Создание базы данных

База данных (БД) разработана для автоматизации учета взвешиваний, управления клиентами, весовым оборудованием и техническим обслуживанием. Система позволяет фиксировать операции взвешивания, хранить историю обслуживания весов, а также управлять пользователями с разными ролями (администратор, оператор, техник). Выбор MS SQL Server обусловлен его надежностью, производительностью и совместимостью с корпоративными решениями, что особенно важно для промышленных систем, где критична точность данных и бесперебойная работа.

Выбор MS SQL Server

MS SQL Server был выбран в качестве платформы для данной базы данных по нескольким ключевым причинам. Во-первых, это высокая надежность и

поддержка транзакций, что гарантирует целостность данных при операциях взвешивания и технического обслуживания. Во-вторых, MS SQL Server обеспечивает высокую производительность даже при работе с большими объемами данных, что важно для систем, где ежедневно фиксируются десятки или сотни взвешиваний. В-третьих, встроенные механизмы безопасности, включая аутентификацию и авторизацию, позволяют гибко управлять доступом пользователей к данным. Кроме того, MS SQL Server легко интегрируется с другими корпоративными системами, такими как ERP или CRM, и поддерживает мощные инструменты аналитики, включая Power BI, что упрощает генерацию отчетов и анализ данных.

Создание базы данных

База данных BridgeScales представляет собой систему управления и учета операций взвешивания на мостовых и железнодорожных весах. Она предназначена для автоматизации работы предприятий, занимающихся взвешиванием грузов, и включает в себя несколько взаимосвязанных таблиц,

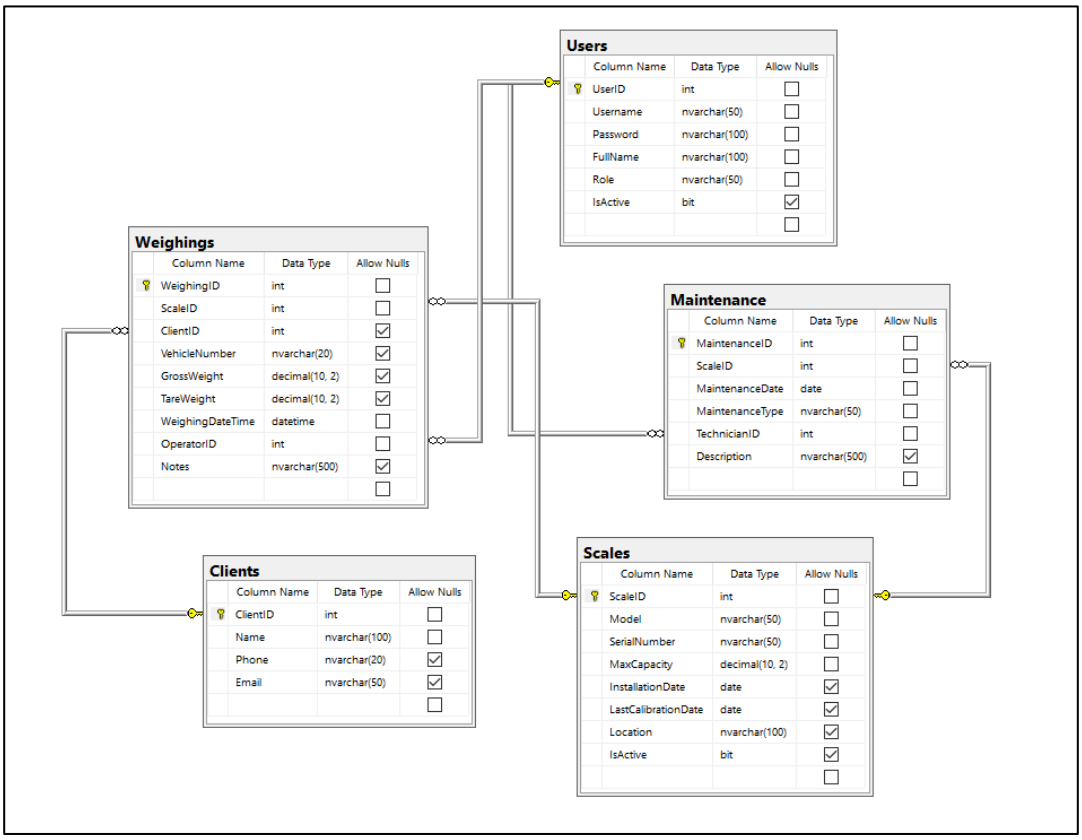

Рисунок 3.2.1 – Схема данных БД BridgeScales

обеспечивающих полный контроль над процессами. Реализация схемы данных представлена на рисунке 3.2.1.

Основной таблицей является Scales, которая хранит информацию о весовом оборудовании. Здесь фиксируются такие параметры, как модель весов, серийный номер, максимальная грузоподъемность, дата установки, дата последней поверки, место расположения и статус активности.

Другая важная таблица — Clients, содержащая данные о клиентах, пользующихся услугами взвешивания. В ней хранятся названия организаций, контактные телефоны и адреса электронной почты.

Для учета сотрудников, работающих с системой, предназначена таблица Users. В ней содержатся учетные записи с логинами, хэшированными паролями, ФИО, ролями (администратор, оператор, техник, менеджер) и статусом активности.

Все операции взвешивания фиксируются в таблице Weighings. Каждая запись включает идентификатор весов, клиента (если он известен), номер транспортного средства, вес брутто и тару, дату и время взвешивания, идентификатор оператора и примечания.

Техническое обслуживание весов регистрируется в таблице Maintenance. Здесь указываются идентификатор весов, дата обслуживания, тип работ (поверка, ремонт и т.д.), идентификатор техника и описание выполненных работ.

Связи между таблицами обеспечивают целостность данных. Взвешивания связаны с весами, клиентами и операторами, а записи обслуживания — с весами и техниками. Дополнительные ограничения гарантируют уникальность серийных номеров весов и логинов пользователей, а также автоматическую фиксацию даты и времени взвешивания.

База данных уже содержит тестовые данные, включая пять клиентов, пять весов, пять пользователей и несколько записей о взвешиваниях и техническом обслуживании. Это позволяет сразу начать работу с системой и протестировать её функциональность. В целом, BridgeScales предоставляет комплексное

решение для учета и управления процессами взвешивания, обслуживания оборудования и работы с клиентами.

3.3 Проектирование и реализация графического интерфейса

Для реализации части проекта, связанного с полноценной работой с БД и выводом в пользовательский интерфейс информации выбраны следующие средства: язык программирования C#, среда разработки Visual Studio и фреймворк WPF (Windows Presentation Foundation). C# – это современный объектно-ориентированный язык, обеспечивающий высокую производительность и удобство разработки. Visual Studio предоставляет мощные инструменты для написания, отладки и тестирования кода. WPF позволяет создавать интерактивные и визуально привлекательные пользовательские интерфейсы с поддержкой XAML, стилей и анимаций. Выбор этих технологий обусловлен их надежностью, широкими возможностями и интеграцией в экосистему Microsoft, что делает их оптимальными для разработки десктоп-приложений.

Пользовательский интерфейс

В данном разделе описывается структура пользовательского интерфейса, реализованного с использованием XAML. Основные элементы включают главное окно (MainWindow.xaml), содержащее навигационную панель, область отображения данных и управляющие элементы. Дополнительные окна, такие как диалоги добавления и редактирования записей, используют Grid, StackPanel для организации содержимого. Стили и шаблоны вынесены в отдельные ресурсы для обеспечения единообразия интерфейса. Окно авторизации пользователей представлено на рисунке 3.2.1.

Код окон

Здесь рассматривается логика работы окон, написанная на C#. Основное окно (MainWindow.xaml.cs) отвечает за навигацию и загрузку данных. Вспомогательные окна (AddWindow.xaml.cs, EditWindow.xaml.cs) содержат

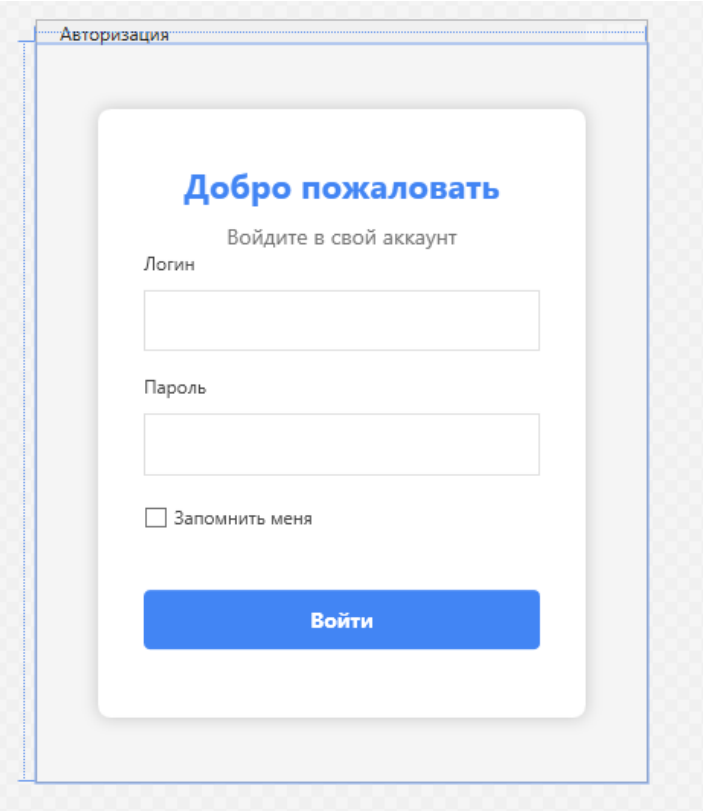

Рисунок 3.3.1 – Окно авторизации пользователей

обработчики событий для кнопок, валидацию ввода и передачу данных в основное приложение. Используются паттерны MVVM и привязка данных (DataBinding) для минимизации кода в code-behind. Код загрузки данных и обновления данных последнего взвешивания на весах представлен на рисунке 3.2.2.

```
Ссылок: 3
private Scale UpdateScaleData()
{
    var sensorService = new FirebaseSensorService();
    _scaling = (Scale)ScaleComboBox.SelectedItem;
    var lastSensorsData = sensorService.
    GetAllSensorDataById(_scaling.ScaleId).OrderByDescending(x=> x.timestamp).FirstOrDefault();
    if (lastSensorsData != null)
    {
        GrossWeightTextBox.Text = lastSensorsData.weight.ToString();
        _sensor = lastSensorsData;
    }
    return _scaling;
}
```

Рисунок 3.3.2 – Код загрузки данных весох

Классы Entity Framework

Работа с базой данных реализована через Entity Framework Core (EF Core). EF Core - это легковесная, кроссплатформенная ORM (Object-Relational Mapper)

для .NET, позволяющая работать с базами данных через объекты C#. Он упрощает доступ к данным, заменяя SQL-запросы на LINQ и автоматизируя маппинг сущностей на таблицы. EF Core поддерживает различные СУБД (SQL Server, PostgreSQL, SQLite и др.), миграции для управления схемой БД и работу с транзакциями. Основные сущности, созданные на основании БД (User, Weighing, Scale) представлены в виде классов с атрибутами, определяющими связи и ограничения в БД. BridgeScalesContext.cs настраивает подключение к MS SQL Server и управляет миграциями. Все необходимы классы для работы с базой данных находиться в папке ModelData она изображена на рисунке 3.3.3.

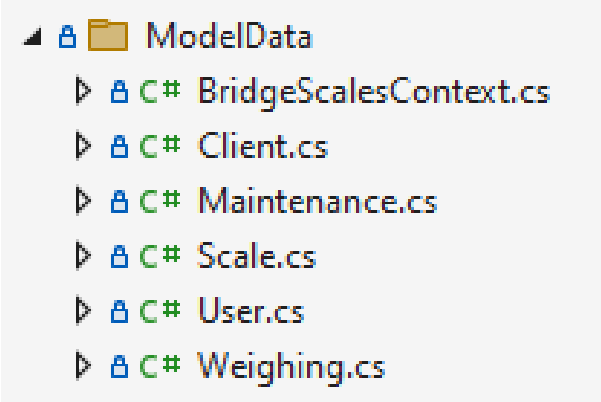

Рисунок 3.3.3 – Классы EF Core папки ModelData

Подключение к FireBase

Описывается интеграция с Firebase Realtime Database для хранения и синхронизации данных в облаке. Класс FirebaseService.cs содержит методы, загрузки и обновления записей. Используется библиотека System.Net.Http для работы с запросами. В проекте применяется JSON-сериализация для передачи объектов между клиентом и сервером.

Реализация кода выгрузки данных с FireBase Realtime Database представлена на рисунке 3.2.4.

Описание работы приложения

Работа приложения начинается с авторизации где операторы весов авторизуются и после попадают на основное окно «WindowWeighings» на котором отображаются все взвешивания. Далее через нажав на кнопку «Добавить» можно попасть на окно «WindowAddEditWeighing» в котором мы

автоматически получаем последнее возвещение, записанное на выбранных весах (с модуля Arduino) и потом можем создать запись возвышения определённого автомобиля.

```
Ссылок: 1
public List<SensorData> GetAllSensorDataById(int Id)
{
    try
    {
        string fullUrl = $"{_firebaseUrl}/{_databasePath}/idScales{Id}.json?auth={_firebaseSecret}";

        HttpResponseMessage response = _httpClient.GetAsync(fullUrl).Result;

        if (response.IsSuccessStatusCode)
        {
            string responseBody = response.Content.ReadAsStringAsync().Result;

            if (!string.IsNullOrEmpty(responseBody) && responseBody != "null")
            {
                using JsonDocument doc = JsonDocument.Parse(responseBody);
                JsonElement root = doc.RootElement;

                var sensorDataList = root.EnumerateObject()
                    .Select(prop => JsonSerializer.Deserialize<SensorData>(prop.Value.GetRawText()))
                    .ToList();

                return sensorDataList!;
            }
        }
```

Рисунок 3.3.4 – Код выгрузки данных с FireBase

Заключение

В ходе учебной практики был успешно разработан прототип настраиваемой трёхдиапазонной звуковой сигнализации, реагирующей на вес объекта. Проект объединил аппаратную часть на базе микроконтроллера ESP32, базу данных для учёта и анализа взвешиваний, а также удобный графический интерфейс для пользователей.

Основные задачи, такие как проектирование сущностей базы данных, выбор электронных компонентов, настройка симуляторов и реализация программного кода, были выполнены в полном объёме. Система демонстрирует высокую гибкость и масштабируемость, позволяя адаптироваться под различные условия эксплуатации. Интеграция с Firebase обеспечивает надёжное хранение данных и возможность их удалённого мониторинга.

Практическая значимость проекта заключается в его применении для контроля веса транспортных средств перед мостами, что способствует повышению безопасности и продлению срока службы инфраструктуры. Дальнейшее развитие системы может включать добавление новых датчиков, расширение функционала аналитики и улучшение пользовательского интерфейса.

Проведённая работа позволила закрепить теоретические знания и приобрести ценный практический опыт в области проектирования и реализации автоматизированных систем. Результаты проекта подтверждают его актуальность и потенциал для внедрения в реальных условиях.

Ссылка на GitHub проекта: https://github.com/Fiery-Paks?tab=repositories

Ссылка на Видео-Отчёт:

https://drive.google.com/file/d/1lH51KnVKd78eVz6V62IIwFOoSB7xlCBE/view?usp=sharing