Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

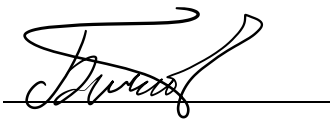
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

по направлению 9.03.04 Программная инженерия

тип программы академическая

профиль «Разработка программно-информационных систем»

Разработка системы дистанционного контроля параметров и управления электрооборудованием автомобиля с использованием микроконтроллера

Студент
Группа ПИ-16  Бычков А. А.

Руководитель ВКР
канд. техн. наук, доцент _____ Качановский Ю. П.

Консультант по
программному
обеспечению
канд. техн. наук, доцент _____ Ведищев В. В.

Нормоконтроль _____ Болдырихин О. В.

Заведующий кафедрой
канд. техн. наук, доцент _____ Алексеев В. А.

Липецк 2020

ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет ФАИ

Кафедра АСУ Заведующий кафедрой Алексеев В.А.

« » _____ 20 ___ г.

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студенту *Бычкову Алексею Андреевичу* группы *ПИ-16*

Направление (специальность) 9.03.04 «Программная инженерия»

1. *Тема Разработка системы дистанционного контроля параметров и управления электрооборудованием автомобиля с использованием микроконтроллера*

2. Цель и задачи работы *спроектировать систему и разработать приложение управляющие системой дистанционного контроля параметров и управления электрооборудованием автомобиля с использованием микроконтроллера поддерживающую текущую доступную реализацию на базе автомобиля ВАЗ-2110.*

3. Характеристика предметной области *дистанционное управление автомобилем*

4. Содержание расчетно-пояснительной записки *введение, постановка задачи, изучение и моделирование предметной области, разработка информационной базы, программно-аппаратная реализация решения задачи, результаты внедрения и использования системы. Достижение целей разработки, заключение (выводы), список источников*

5. Перечень графического материала *иллюстрации предметной области (скриншоты программ), диаграмма вариантов использования, схема функциональной структуры, диаграмма "сущность-связь", концептуальная модель базы данных, физическая структура базы данных, структура аппаратного обеспечения, диаграмма состояний,*

6. Срок сдачи ВКР руководителю *26.06.2020*

7. Консультанты по ВКР _____

8. Дата выдачи задания *10.02.2020*

9. Руководитель ВКР _____

кандидат технических наук, доцент Качановский Ю. П.

10. Задание принял к исполнению студент *Бычков А. А.*

Заведующий кафедрой АСУ Алексеев В.А. _____

Аннотация

С. 63, Ил. 21, Табл. 4, Прил. 1, Литература 9 назв.

В данной работе представлена проектирование системы и разработка приложения для управления системой дистанционного контроля параметров и управления электрооборудованием автомобиля с использованием микроконтроллера. Рассматриваемая система включает две части реализации: управление автомобиля с помощью микроконтроллера и веб-серверное приложение.

Разработанное решение состоит из нескольких компонентов: готового веб-приложения для управления системой, спроектированная и реализованная база данных, спроектированная архитектура и перечень используемых микроконтроллеров и физических модулей для включения в электроцепь автомобиля, так же рассмотрены протоколы общения между электронным блоком управления автомобиля и микроконтроллером.

Оглавление

Введение

В современном мире повсеместной автоматизации процессов и внедрения цифровых технологий, активно развивается область «Интернет вещей». В это понятие входят так называемые «умные» вещи - предметы, которые имеют свой собственный микроконтроллер, а также, опционально, ряд датчиков и непосредственное подключение к сети интернет. Благодаря такому набору надстроек, любой предмет быта может стать «умным».

Умные розетки, которые выключаются, когда хозяин квартиры покидает её. Умный электрочайник, который можно включить по пути домой с работы, чтобы сразу по приходу домой налить горячий чай. Умная лампочка, которая отслеживает количество света в комнате и включается, когда света становиться мало. Все эти вещи направленны в первую очередь на создание комфорта и удобства.

Мной была выбрана тема разработки системы дистанционного управления автомобилем. В эту систему должен входить дистанционный автозапуск двигателя, управление его некоторыми электроприборами, а также отслеживание некоторых параметров работы.

Дистанционный автозапуск двигателя, позволяет в холодное время года заранее завести машину, до выхода из дома, чтобы двигатель прогрелся до своих рабочих температур. Для более комфортабельного использования часть водителей также отмечают важность прекондиционирования салона автомобиля. Дистанционное управление отопителем салона автомобиля, позволяет прогреть салон автомобиля, при холодной погоде. Мониторинг и статистический анализ параметров работы двигателя автомобиля может дать необходимые для автовладельца данные о состоянии автомобиля. Например, это и другое создаёт комфорт и удобство пользования автомобилем.

1 Постановка задачи

1.1 Литературный и патентный обзор постановки подобных задач. Анализ стандартных средств и существующих способов решения задачи

В настоящее время на рынке существует много устройств дистанционного запуска двигателя с помощью радиопультов управления. Данный метод управления очень прост в реализации, не требует никаких дополнительных программных продуктов для управления системой. Наличие пультов управления упрощает систему и снижает входной порог пользователей, доводя его до абсолютно любого массового пользователя. Однако в современном мире данный метод начинает устаревать и ан его смену приходит дистанционное управление через смартфон.

Рынок устройств дистанционного управления автомобилем через смартфон относительно мал. Из крупных фирм здесь можно выделить несколько главенствующих фирм:

- Tesla;
- Kia Motors Corporation;
- Volkswagen Audi Gruppe.

Фирма «Tesla» стоит в списке на первом месте, не просто так. Данная фирма является ведущей в производстве электрокаров. Их автомобили стали первыми, для открытия и запуска которых не требуются ключи. Всё что нужно для их использования – смартфон и специальное приложение от фирмы «Tesla».

Инженеры «Tesla» проделали большую работу в сфере безопасности дистанционной передачи данных между смартфоном и автомобилем. Ведь со смартфона можно не только завести/заглушить машину, управлять её системой мультимедиа, но и открыть её. Учитывая высокую стоимость автомобилей этой фирмы, это делает их более привлекательными для авто угонщиков, обладающими навыками взлома программных продуктов и

беспроводных сетей.

Другие две фирмы, специализируются на выпуске классических автомобилей на ДВС, однако их достижения в создании систем дистанционного управления автомобиля немного уступают решениям от фирмы «Tesla». Для анализа решений данных фирм, с точки зрения потребителя, был использован веб-форум автовладельцев Drive2.ru [1].

Системы от KIA Motors позволяют не только выполнять автозапуск двигателя, системой открытия автомобиля, но также отслеживать ряд параметров автомобиля: температуры в салоне и на улице, оборотами двигателя, отслеживания маршрута автомобиля по GPS.

Система от VAG позволяет отслеживать параметры автомобиля, такие как: напряжение аккумулятора, уровень топлива, пробег, параметры автомобиля, техосмотр автомобиля и его VIN номер.

1.2 Объекты управления, информационные объекты и автоматизируемые процессы. Пользователи и внешние сущности

Объектом управления является автомобиль, в частности его электросеть и силовой агрегат (двигатель).

Автоматизируемые процессы:

- запуск двигателя;
- управление электрооборудованием автомобиля;
- сбор и обработка информации с датчиков и ЭБУ автомобиля;

Пользователи системы:

1. Основной пользователь - водители автомобилей:

Запуск/остановка двигателя, управления электроприборами автомобиля, просмотр параметров работы двигателя и статистические данные.

2. Администратор:

Регистрация новых пользователей и их автомобилей, редактирование

набором управляемого электрооборудования, изменение описания и наименования элементов системы.

Так же можно выделить автомобиль как внешнюю сущность, так как он является отдельной большой и сложной системой, к которой присоединяется разрабатываемая система. От автомобиля поступают данные в систему, и производиться контроль его некоторых функций, что позволяет его выделить как источник входной и приёмник выходной информации.

1.3 Цели разработки, функции системы, ограничения и критерии оценки результатов

1.3.1 Цели разработки

Целями создания системы являются:

- дистанционное управление электроприборами автомобиля;
- отслеживание ошибок работы двигателя для своевременного ремонта автомобиля и предотвращения больших затрат;
- формирование кратких сводок по основным параметрам работы автомобиля за определённый период;
- общее повышение комфорта при использовании автомобиля.

1.3.2 Функции системы

Система должна выполнять следующие функции:

- управление запуском двигателя автомобиля;
- управление отдельным электрооборудованием электросистемы автомобиля;
- статистический анализ данных, получаемых от электронного блока управления (далее ЭБУ) автомобиля;
- вывод параметров работы двигателя в реальном времени;
- вывод ошибок работы двигателя;
- администрирование системы и базы данных.

1.3.3 Критерии оценки эффективности

Для оценки результата реализации системы можно выделить следующие критерии:

- удобный и интуитивно-понятный интерфейс, простота в обращении;
- организация функционала достаточного для предоставления пользователю функций, описанных в пункте 1.3.2;
- сохранение работоспособности и восстановление системы, при временном отключении питания электроцепи автомобиля, или же при сбое на сервере, вызванном внешними факторами.

2 Изучение и моделирование предметной области

2.1 Выявление основных понятий и процессов, их свойств и закономерностей. Построение ER-диаграммы предметной области

2.1.1 Основные понятия

Модуль (микроконтроллер) – микросхема, предназначенная для управления электронными устройствами.

Электронный блок управления (ЭБУ) – общий термин для любых встраиваемых систем, которые управляют одним или несколькими электрическими системами или подсистемами в автомобиле.

Автоматический запуск двигателя автомобиля – процесс, управления силовым агрегатом транспортного средства на расстоянии. Основное предназначение этого механизма – предварительный прогрев мотора, что особенно актуально для холодного периода года.

Диалоговое кодирование – способ кодирования, требующий двухстороннего канала связи, заключающийся в том, что для выполнения команды, блок охраны генерирует случайное число и отправляет его обратно отправителю, отправитель по специальному алгоритму генерирует ответ и отправляет блоку. В свою очередь блок так же внутри себя преобразует число

по такому же алгоритму и сравнивает с пришедшим ответом. Только в том случае, если числа совпали, блок охраны выполняет команду.

2.1.2 Построение ER-диаграмм предметной области

Диаграмма вариантов использования представлена на рисунке 1.

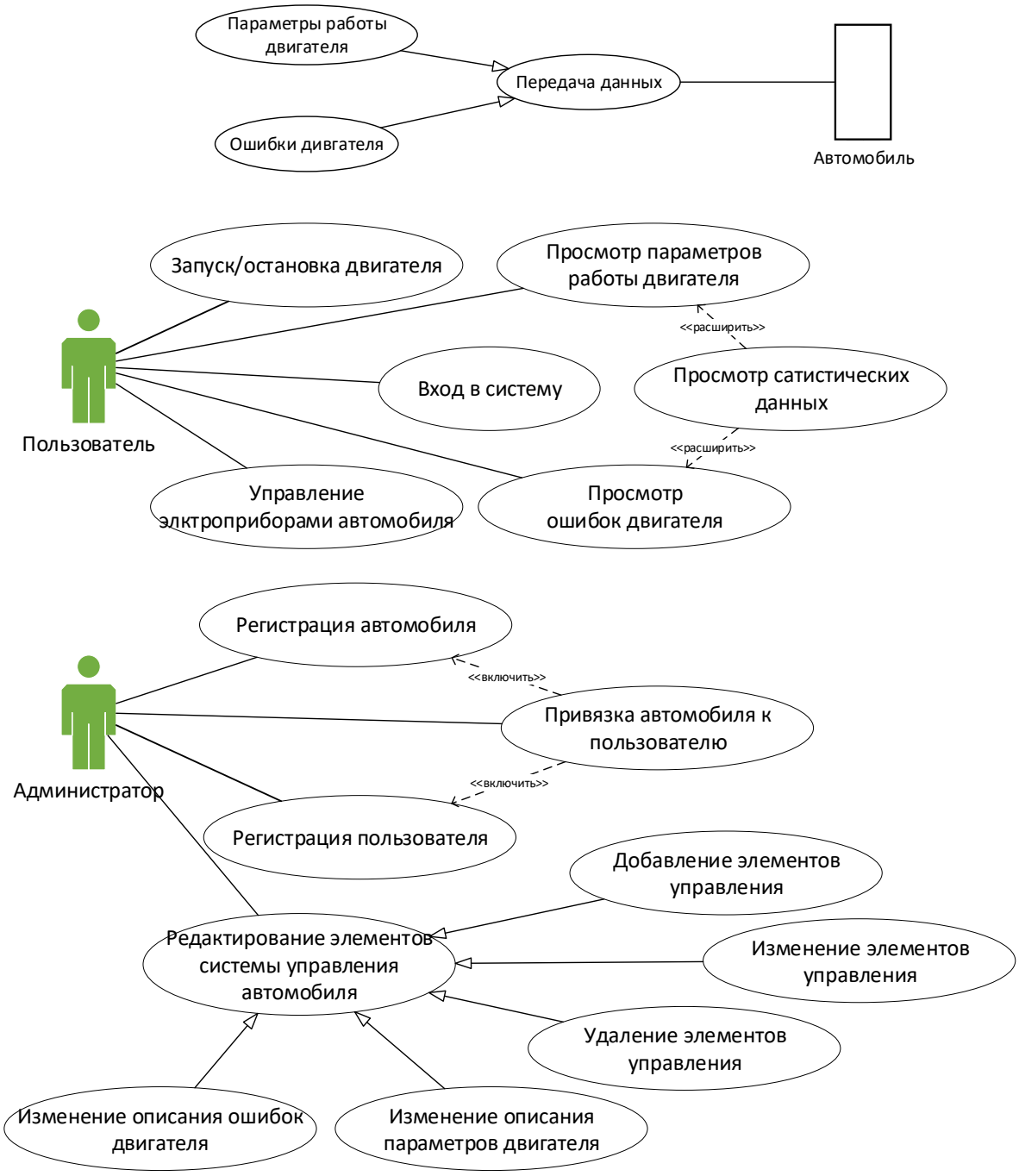


Рисунок 1 – Диаграмма вариантов использования

Ниже представлена спецификация диаграммы вариантов использования.

Пользователь: запускает двигатель автомобиля, просматривает параметры работы двигателя, статистические данные и ошибки, включает подключенные к системе управляемые элементы электрооборудования.

Администратор: регистрирует пользователя, регистрирует автомобиль и привязывает к пользователю. При необходимости регистрирует автомобиль или пользователя отдельно, если необходимо осуществить привязку к уже имеющемуся субъекту в системе. Добавляет, изменяет, удаляет элементы управления автомобиля (датчики, управляющие реле) и изменяет описание ошибок и параметров двигателя.

ER-диаграмма в нотации Чена представлена на рисунке 2.

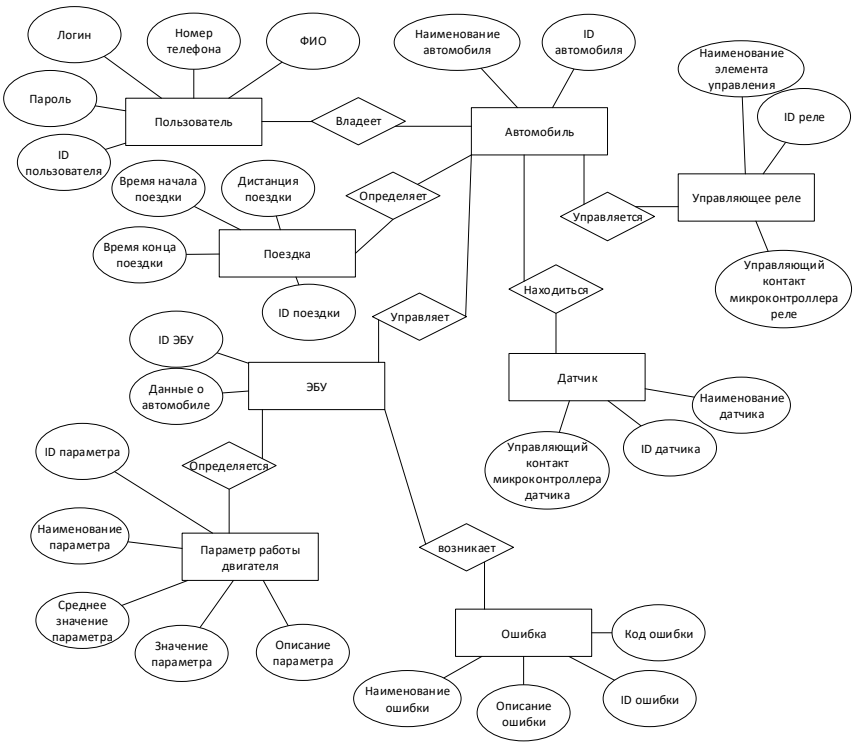


Рисунок 2 – Диаграмма атрибут-атрибут

2.2 Теоретическое изучение предметной области. Построение теоретических математических моделей

Выбранная мной область дистанционного контроля и мониторинга параметров электрооборудования автомобиля является очень востребованной

в современной жизни, а также в реалиях проживания в умеренном климате, где зима длиться около полугода.

Выбранная предметная область умных вещей, в первую очередь направлена на создание комфорта и удобства пользования, различными предметами повседневной жизни. Основной потребностью для водителей нашего региона является дистанционный запуск двигателя. Дистанционный автозапуск двигателя, позволяет в холодное время года заранее завести машину, до выхода из дома, чтобы двигатель прогрелся до своих рабочих температур. Для более комфортабельного использования часть водителей также отмечают важность предкондиционирования салона автомобиля. Дистанционное управление отопителем салона автомобиля, позволяет прогреть салон автомобиля, при холодной погоде. Мониторинг и статистический анализ параметров работы двигателя автомобиля может дать необходимые данные о состоянии авто. Это создаёт комфорт и удобство пользования автомобилем.

Исходя из выше сказанного, можно сделать вывод, что для создания дистанционного управления автомобилем необходимо подготовить реализацию следующего функционала:

- Управление запуском двигателя автомобиля;
- Управление электрооборудованием автомобиля;
- Статистический анализ данных получаемых от ЭБУ автомобиля;
- Вывод параметров работы двигателя в реальном времени.

Дистанционное управление достигается за счёт внедрения в электросистему автомобиля модулей дистанционного управления. Любой модуль дистанционного управления состоит из микроконтроллера, модуля беспроводной связи и ряда приспособлений, для включения в электросеть автомобиля.

2.2.1 Шифрование сигнала. Защита данных

Для защиты информации и обеспечения безопасности использования системы необходимо применять методы шифрования сигнала.

Для первого этапа защиты между автомобилем и сервером используется протокол HTTPS. Для шифрования данных в нём используется протокол SSL/TLS [2], который используется на уровень ниже, поверх HTTPS. Для лучшего понимания принципа работы рассмотрим упрощённый вариант функционирования SSL/TLS.

Первым этапом начала обмена шифрованными данными является так называемое «SSL рукопожатие». На этом этапе происходит аутентификация, сверка версий SSL, поддерживаемых алгоритмов шифрования, согласование сеансового ключа и обмен другими данными необходимыми для общения сервера с клиентом. Процесс согласования сеансовых ключей в случае RSA представлен на рисунке 3.

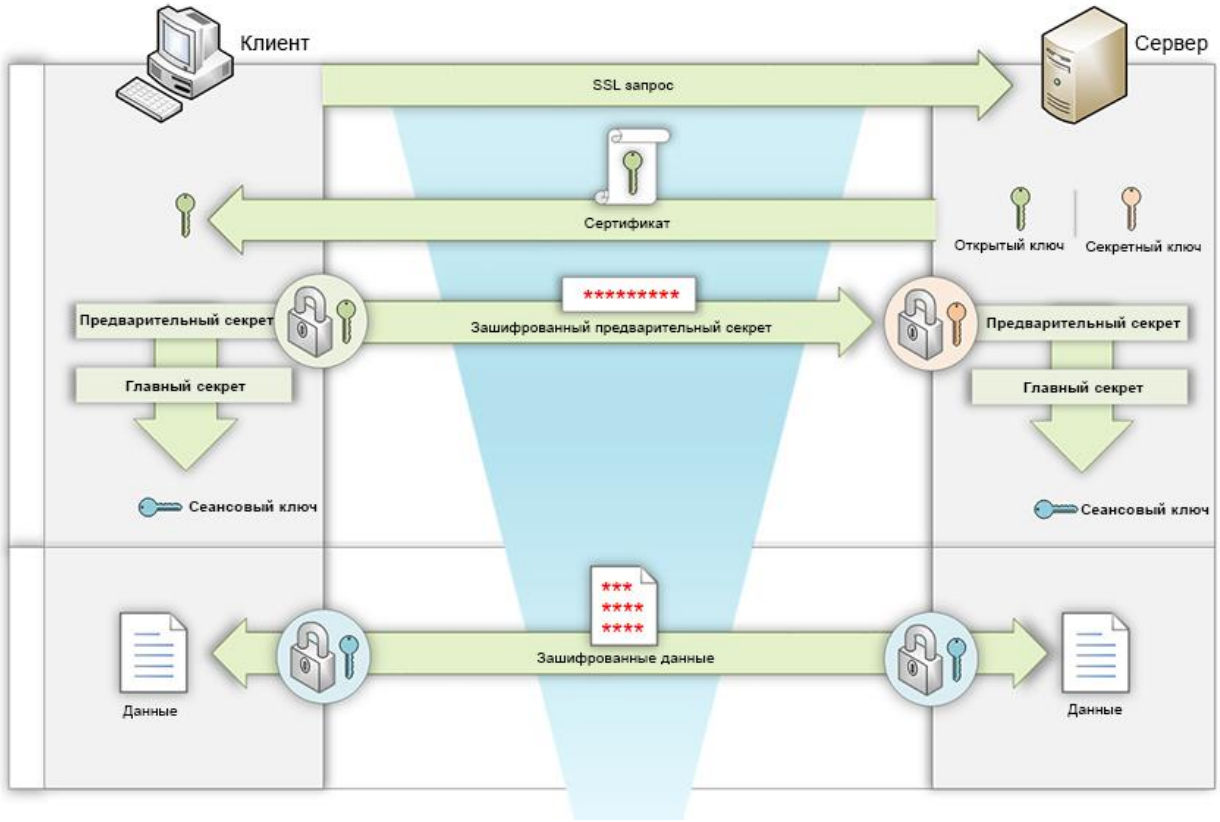


Рисунок 3 – Диаграмма согласования сеансовых ключей в случае RSA

В момент установки соединения клиент генерирует случайное число – предварительный секрет. Шифрует его открытым ключом, полученным в сертификате от сервера.

Отправляет в зашифрованном виде на сервер. Сервер расшифровывает предварительный секрет своим секретным ключом. Далее обе стороны, обладая одинаковым предварительным секретом, конвертируют его в главный и уже из него создают общий сеансовый ключ.

В ассиметричных алгоритмах шифрования (RSA) данные, зашифрованные открытым ключом, могут быть расшифрованы только секретным. При этом, открытый и секретный ключи должны быть связаны между собой определенным математическим образом, являются ключевой парой.

Основной уязвимостью данного подхода является то, что, используя алгоритм RSA, перехваченный трафик между сервером и клиентом всегда можно расшифровать, имея секретный ключ сервера. Дело в том, что в момент установки SSL/TLS-соединения клиент пересылает серверу зашифрованное значение предварительного секрета. Предварительный секрет дешифруется секретным ключом сервера, далее вычисляется сеансовый ключ, и данные расшифровываются полученным сеансовым ключом.

Так как существует риск перехвата и расшифровки передаваемого пакета по HTTPS трафику, следует предусмотреть дополнительное кодирование для передаваемых управляющих сигналов. Для этого было выбрано диалоговое кодирование [3].

Пример диалогового кодирования.

Пользователь посылает команду на запуск двигателя, сервер её обрабатывает и передаёт на микроконтроллер. Микроконтроллер принимает команду, но перед её выполнением посылает обратный сигнал на сервер со случайно сгенерированным числом, для проверки подлинности.

Рассмотрим следующий алгоритм: $X * A^2 + X^2 * B^3 - X^3 * C^4 + \frac{D}{X} = Y$, где A, B, C и D – это числа, которые записываются в систему на этапе производства. X – случайное число, которое генерирует микроконтроллер в момент получения команды. Y – число полученное в результате работы алгоритма по преобразованию числа X.

Сервер принимает сигнал, обрабатывает сигнал по заданному алгоритму и передаёт ответ обратно на микроконтроллер. Одновременно с ним, микроконтроллер пропускает своё случайно сгенерированное число через такой же алгоритм, как и на сервере, заложенный при производстве, и сверяет полученное число с ответом сервера. В том, и только в том случае, если оба числа совпадают, микроконтроллер выполнит команду сервера.

Криптостойкость данного алгоритма подтверждена временем и многие крупные компании по производству сигнализаций, в том числе и StarLine, проводят конкурс на поиск уязвимостей в системах, оснащённых таким алгоритмом кодирования. И на текущий момент времени данный алгоритм не взломан.

2.2.2 Построение математических моделей для формирования статистических данных

Одной из основных функций системы является формирование кратких сводок за определённый период основанных на математическом и статистическом анализе данных поступающих.

В таблице 1 приведены формулы, по которым производиться расчёт для некоторых основных параметров.

Таблица 1. Основные формулы расчёта статистических данных

Наименование	Формула	Расшифровка
Средний расход топлива в час за определённый	$\sum_{i=n}^{m} \frac{x_i}{m-n+1}$ (л/ч)	x_i – значение параметра «моментальный расход топлива в час»;

период		n – начало периода; m – конец периода.
Средний расход топлива на 100км за определённый период	$\sum_{i=n}^{m} \frac{x_i}{m-n+1}$ $(\frac{\text{л}}{100\text{км}})$	x_i – значение параметра «моментальный расход топлива на 100км»; n – начало периода; m – конец периода.
Средний расход топлива в час за поездку	$\sum_{i=n}^{m} \frac{x_i}{m-n+1}$ (л/ч)	x_i – значение параметра «моментальный расход топлива в час»; n – начало поездки; m – конец поездки.
Средний расход топлива на 100км за поездку	$\sum_{i=n}^{m} \frac{x_i}{m-n+1}$ $(\frac{\text{л}}{100\text{км}})$	x_i – значение параметра «моментальный расход топлива на 100км»; n – начало поездки; m – конец поездки.
Среднее время поездки за определённый период	$\sum_{i=n}^{m} \frac{t_{i-\text{кон}} - t_{i-\text{нач}}}{m-n}$ (ч)	$t_{i-\text{кон}}$ – значение параметра «время конца поездки»; $t_{i-\text{нач}}$ – значение параметра «время начала поездки»; n – начало периода; m – конец периода.
Среднее расстояние поездки за определённый период	$\sum_{i=n}^{m} \frac{S_i}{m-n+1}$ (км)	S_i – пробег за поездку; n – начало периода; m – конец периода.

Время прогрева двигателя до заданной температуры за поездку	$t_{\text{тек}} - t_{\text{нач}}$ (мин), если $t° \geq 85℃$	$t_{\text{тек}}$ – текущее время; $t_{\text{нач}}$ – время начала поездки; $t°$ – значение параметра «температура охлаждающей жидкости».
Средняя скорость за поездку	$\frac{S_i}{t_{i-\text{кон}} - t_{i-\text{нач}}}$ (км/ч)	$t_{i-\text{кон}}$ – значение параметра «время конца поездки»; $t_{i-\text{нач}}$ – значение параметра «время начала поездки»; S_i – пробег за поездку.
Пробег за определённый период	$\sum_{i=n}^{m} S_i$ (км)	S_i – пробег за поездку; n – начало периода; m – конец периода.
Израсходовано топлива за определённый период	$\sum_{i=n}^{m}(t_{i-\text{кон}} - t_{i-\text{нач}}) * X_i$ (л)	$t_{i-\text{кон}}$ – значение параметра «время конца поездки»; $t_{i-\text{нач}}$ – значение параметра «время начала поездки»; X – средний расход топлива за поездку; n – начало периода; m – конец периода.
Расстояние, пройденное за период опроса датчиков	$\frac{V_0 + V_{\text{тек}}}{2} * T$	V_0 – значение параметра «текущая скорость» за предыдущий опрос датчиков; $V_{\text{тек}}$ – значение параметра «текущая скорость» T – период опроса датчиков.

Расстояние, пройденное за поездку	$\sum_{i=n}^{m} \frac{V_{0\,i} + V_{\text{тек}\,i}}{2} * T_i$	V_0 – значение параметра «текущая скорость» за предыдущий опрос датчиков; $V_{\text{тек}}$ – значение параметра «текущая скорость» T – период опроса датчиков; n – начало поездки; m – конец поездки.

2.3 Экспериментальное изучение предметной области. Построение эмпирических математических моделей

Для вычисления среднего расхода топлива на 100км за поездку потребуется знать время начала и время конца поездку, а также все промежуточные значения моментального расхода топлива на 100км, которые брались с определённой периодичностью. Стоит отметь, что большое значение в подобных вычислениях средних значений дискретизированных величин играет период обновления данных – чем он больше, тем меньше точность вычислений.

Для сравнения возьмём два массива данных и посчитаем для них средний расход топлива:

$$A = [10.1, 15.3, 23.7, 25.1, 24.7, 21.6, 8.9, 6.7, 6.6, 6.8, 0]$$

$$B = [10.1, 24.4, 21.6, 6.6, 0]$$

Массив A описывает дискретизацию величины расхода топлива на 100км с периодом 2 секунды, на отрезке времени 20 секунд. Массив B описывает ту же величину, на том же отрезке времени, одна период дискретизации теперь 5 секунд. Рассчитаем необходимые нам значения, по формуле «Средний расход топлива на 100км за поездку» приведённой в таблице 1.

$$X_A = \frac{10.1 + 15.3 + 23.7 + 25.1 + 24.7 + 21.6 + 8.9 + 6.7 + 6.6 + 6.8 + 0}{11 - 0}$$

$$= 13.590 \ \frac{\text{л}}{100\text{км}}$$

$$X_B = \frac{10.1 + 24.4 + 21.6 + 6.6 + 0}{5 - 0} = 12.54 \ \frac{\text{л}}{100\text{км}}$$

Наиболее точным оказался результат, полученный из массива A. Он отличается от значения 13.7, рассчитанных бортовым компьютером с периодом дискретизации 1 секунда, на 0.11 единиц, что является приемлемым результатом.

Значения полученные из массива B в средней значимости отличаются от значений, полученных из массива A. Это говорит о том, что увеличение периода дискретизации негативно влияет на точность вычислений средних значений. Однако увеличение отрезка времени обработки данных нивелирует эту погрешность.

В ходе дополнительных вычислений и сравнений данных на больших отрезках времени показало, что оптимальной величиной для периода дискретизации была является величина 5 секунд. Этот период даёт приемлемое соотношение точности к общему числу запросов от модуля к серверу.

Эти выводы касаются и других формул из таблицы 1, использующих дискретизированные величины.

Посчитаем так же расстояние, пройдённое за поездку, по одноимённой формуле из таблицы 1.

$$\frac{0 + 30}{2} * \frac{1}{720} + \frac{30 + 70}{2} * \frac{1}{720} + \frac{70 + 50}{2} * \frac{1}{720} + \frac{50 + 25}{2} * \frac{1}{720} + \frac{25 + 10}{2}$$

$$* \frac{1}{720} + \frac{10 + 0}{2} * \frac{1}{720} = 0.257 \text{ км}$$

Для примера был взят разгон с места до 80 км/ч и плавное торможение до полной остановки. Результатом является 0.257 км, что отличается от

результата 0.275, посчитанного бортовым компьютером, на 0.018км, что является хорошим результатом.

Из вышеописанных вычислений можно сделать вывод о том, что предложенные формулы для расчёта статистических величин приемлемы для использования в разрабатываемой системе. Так же был подобран период дискретизации для дальнейшей разработки.

3 Разработка информационной базы для решения задачи.

3.1 Построение концептуальной и физической модели данных.

Концептуальная модель базы данных приведена на рисунке 4.

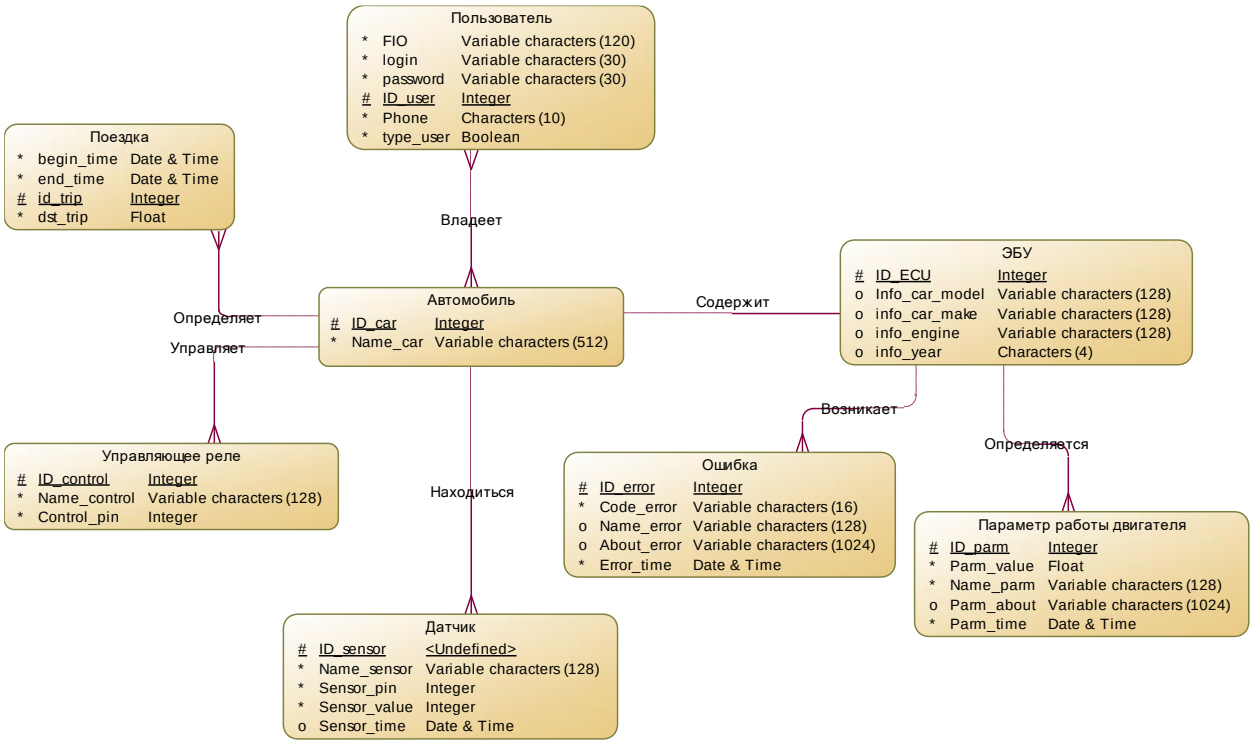


Рисунок 4 – Концептуальная модель БД

Сущность «Пользователь» описывает пользователей системы и хранит данные для аутентификации, их ФИО и номер телефона, а также тип учётной записи пользователя.

Сущность «Автомобиль» описывает автомобиль, который принадлежит пользователю и хранит в себе информацию о его наименовании, которое задал пользователь при регистрации.

Сущность «ЭБУ» описывает электронный блок управления автомобиля, в частности, данные, которые система считывает из него: модель автомобиля, марка автомобиля, модель двигателя, год выпуска.

Сущность «Ошибка» описывает ошибки считываемые из ЭБУ автомобиля и содержит в себе код, наименование, описание дату и время появления ошибки.

Сущность «Параметр работы двигателя» описывает конкретный параметр работы двигателя и содержит в себе наименование, значение и описание параметра, а также дату и время считывания.

Сущность «Управляющее реле» описывает реле управления электрооборудованием автомобиля и содержит в себе наименование управляемого электрооборудования, и управляющий пин на микроконтроллере.

Сущность «Датчик» описывает датчики, установленные в автомобиль, для контроля электрооборудования и содержит в себе наименование электрооборудования, контролируемого датчиком, принимающий сигнал пин на микроконтроллере, значение датчика в определённы момент времени, дата и время считывания.

Сущность «Поездка» описывает поездку на автомобиле, которая измеряется с момента запуска двигателя до момента его остановки, и содержит в себе время начала и время окончания поездки, пройденное расстояние за поездку.

Физическая модель базы данных представлена на рисунке 5.

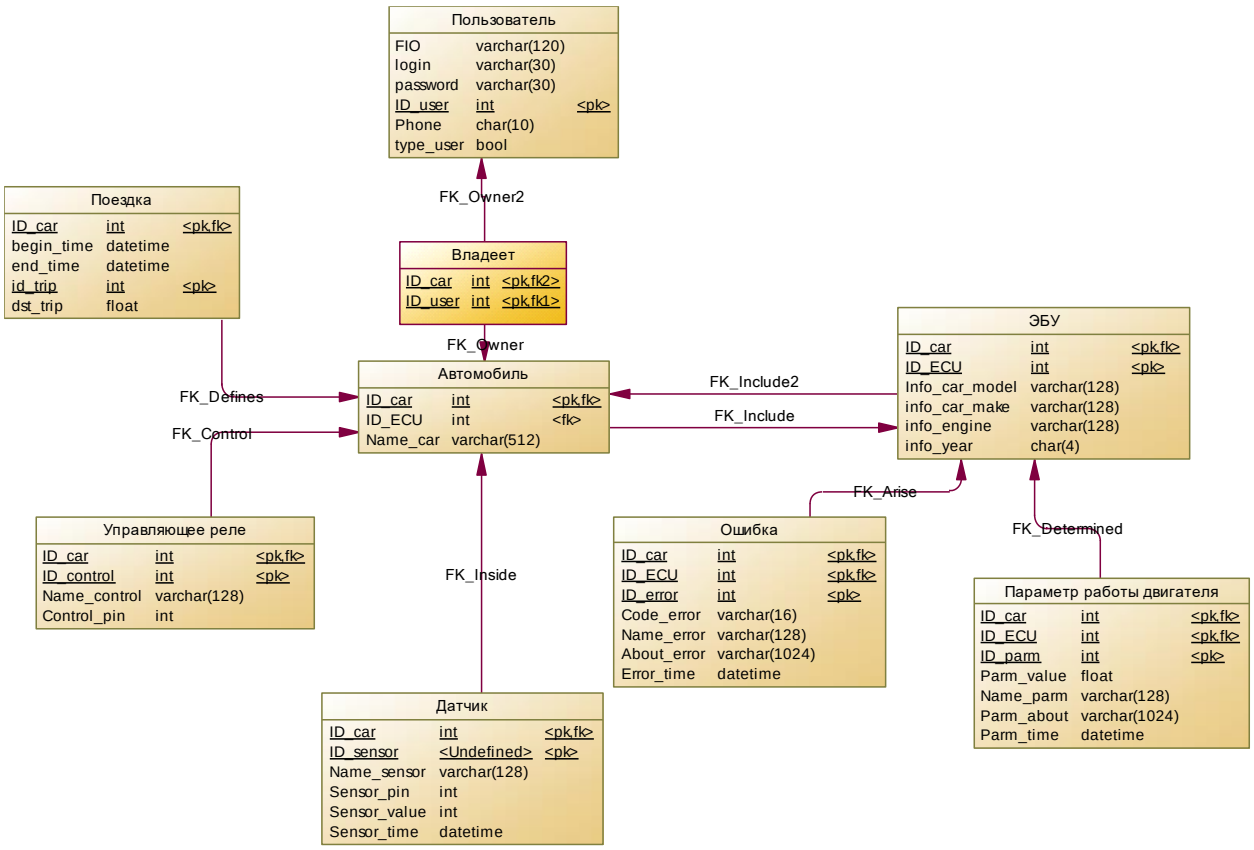


Рисунок 5 – Физическая модель БД

Спецификация физической модели базы данных приведена в приложении А.

3.2 Описание источников информации, входных сигналов и документов.

Основными источниками входных сигналов является ЭБУ автомобиля, состояние некоторых участков электроцепи автомобиля, а также команды пользователей системы, получаемых из веб-приложений.

Входная информация от ЭБУ автомобиля – это структурированный поток данных, получаемых через адаптер типа OBD-II, содержащий в себе информацию о всех параметрах работы двигателя, а также некоторую информацию о самом автомобиле, такую как: марка и модель автомобиля, год выпуска, модель двигателя, комплектация автомобиля.

Входная информация от некоторых участков электроцепи автомобиля –

это проверка состояния, путём отслеживания наличия/отсутствия/изменения напряжения на цепи датчиков отслеживания положения следующих устройств:

- ручка ручного тормоза;
- ручка коробки передач;
- электроцепь зажигания автомобиля;
- датчик температуры внутри салона.

Входная информация от пользователей системы заключается в запросах к серверу на выполнение конкретных действий, нацеленных как на управление автомобиля через систему, так и на управление самой системой и внесения в неё каких-либо изменений.

3.3 Описание выходной информации: сигналов, документов и видеокадров.

Выходная информация представлена в следующем виде:

- информация о параметрах работы двигателя автомобиля;
- Собранные статистические данные, хранящиеся в базе данных, в виде отчётов или краткой сводки за определённый период.
- Выходные сигналы представлены в следующем виде:
- сигналы управления электрооборудованием автомобиля, а также управления запуска/остановкой двигателя.

4 Программно-аппаратная реализация решения задачи:

4.1 Аппаратное обеспечение.

Перечень, характеристика, настройка и схема аппаратных средств.

Для реализации системы была платформа на базе автомобиля ВАЗ-2110 с инжекторным двигателем и электронной системой управления двигателем (далее ЭСУД) 2111–1411020-70 на базе ЭБУ Bosch M1.5.4 R83.

Для обращения к ЭБУ используется адаптер OBD-II на базе микроконтроллера ELM327 с интерфейсом подключения USB. ELM327 поддерживает большое количество ЭБУ и протоколов связи с ними. Простота в настройке и использовании, а также большой перечень поддерживаемых ЭБУ являются основными факторами выбора адаптера.

Для общения между ЭБУ и адаптером используется протокол KWP-2000, использующий стандарты ISO 14230 (Road Vehicles - Diagnostic Systems Keyword Protocol 2000), ISO 14229 (Road Vehicles - Diagnostic Systems Diagnostic Services Specification), SAE J1930 (E/E Systems Diagnostic Terms, Definitions, Abbreviations & Acronyms) и SAE J2012 (Diagnostic Trouble Codes) [4]. Для общения с OBD-II адаптером на базе чипа ELM327 используются AT-команды [5].

Для включения в электроцепь автомобиля были использованы следующие аппаратные средства:

- Arduino UNO R3 на базе чипа ATmega328;
- радио-модуль rf24L01+;
- GSM/GPRS SIM900R Shield;
- модуль-реле базирующийся на SONGLE SRD-05VDC.

Характеристики платы Arduino UNO R3 представлены в таблице 2.

Таблица 2. Характеристики платы Arduino UNO R3

Микроконтроллер	ATmega328
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В

Цифровые входы/выходы	14 (из них 6 могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	6
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода 3.3V	50 мА
Flash-память	32 КБ (ATmega328) из которых 0.5 КБ используются загрузчиком
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц

Для взаимодействия микроконтроллеров друг с другом используются радио-модули nRF24L01+. Беспроводной способ передачи данных между микроконтроллерами облегчает монтаж системы. Радио-модуль rf24L01+ подключается к плате Arduino UNO как показано на рисунке 9. Характеристики радио-модуля представлены в таблице 3.

Таблица 3. Характеристики радио-модуля rf24L01+

Микроконтроллер	rf24L01
Интерфейс обмена данными	SPI
Рабочее напряжение	3.3В
Напряжение питания	1,9В – 3,6В
Частота приёма и передачи	2,4 ГГц;
Количество каналов	128 с шагом 1МГц
Тип модуляции	GFSK
Скорость передачи данных	250kbps, 1Mbps и 2Mbps
Чувствительность приёмника	-82 dBm

Расстояние приёма/передачи данных	100м – прямая видимость; 30м – помещение
Коэффициент усиления антенны	2dBm
Диапазон рабочей температуры	-40℃…+85℃
Организация сети на одном канале	7 модулей (1 приёмник и 6 передатчиков)

Для передачи данных от автомобиля на сервер используется GSM/GPRS shield SIM900, со специально приобретённой SIM-картой. GSM/GPRS передача данных позволяет передавать данные практически из любой точки, не зависимо от каких-либо ещё дополнительных устройств передачи данных.

Характеристики платы GSM/GPRS SIM900R Shield представлены в таблице 4.

Таблица 4. Характеристики GSM/GPRS SIM900R Shield

Микроконтроллер	SIM900R
Рабочие диапазоны GSM	850/1800МГц
Рабочее напряжение	5В
Напряжение питания	7-12В
Частота приёма и передачи	2,4 ГГц;
Класс GPRS	2+
Класс GSM	B

Для включения в электроцепь автомобиля используется набор модулей реле для Ардуино, схема работы которого представлена на рисунке 4. Были выбраны модули с одноканальными электромагнитными реле номиналами 10 А при 250 и 125 В переменного тока и 10 А при 30 и 28 В постоянного тока. Управляющая группа контактов состоит из двух наборов по три и четыре контакта. Первый набор содержит контакты: GDN – заземления, Vcc – постоянное питание +5В, In1 и In2 – управляющие контакты. Второй набор

имеет контакты с перемычкой (джампер) между JD-Vcc и Vcc и открытый GDN. При использовании перемычки электромагнит реле напрямую получает питание напрямую от реле и при выходе реле из строя, может быть повреждён микроконтроллер. Во избежание подобной ситуации питание для модуля подключается от отдельного источника питания 5V на второй набор к контактам GDN и JD-Vcc.

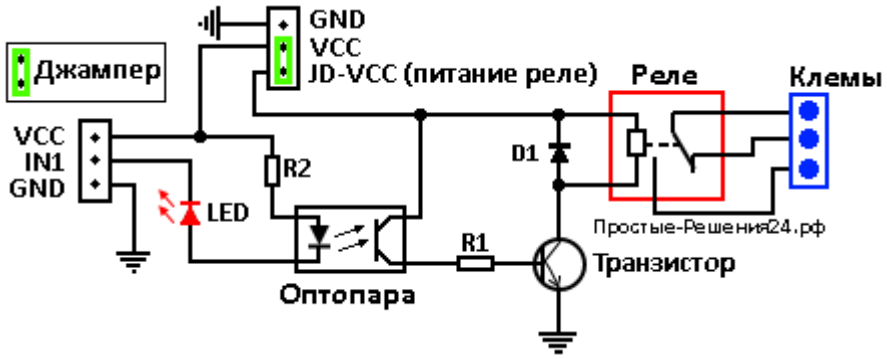


Рисунок 6 – электронная схема модуля реле для Ардуино

Выходной разъем высокого напряжения имеет 3 контакта: средний является общим контактом, один из двух других контактов предназначен для нормально разомкнутого соединения, а другой – для нормально замкнутого соединения.

При подаче сигнала «LOW» на контакт In1, соответствующее реле находиться в нормально-замкнутом состоянии, когда ключ замкнут на 2 контакт реле. При подаче сигнала «HIGH» на контакт In2, соответствующее реле переходит в нормально-разомкнутое состояние, когда ключ замкнут на 1 контакт реле.

Структурная схема системы представлена на рисунке 7.

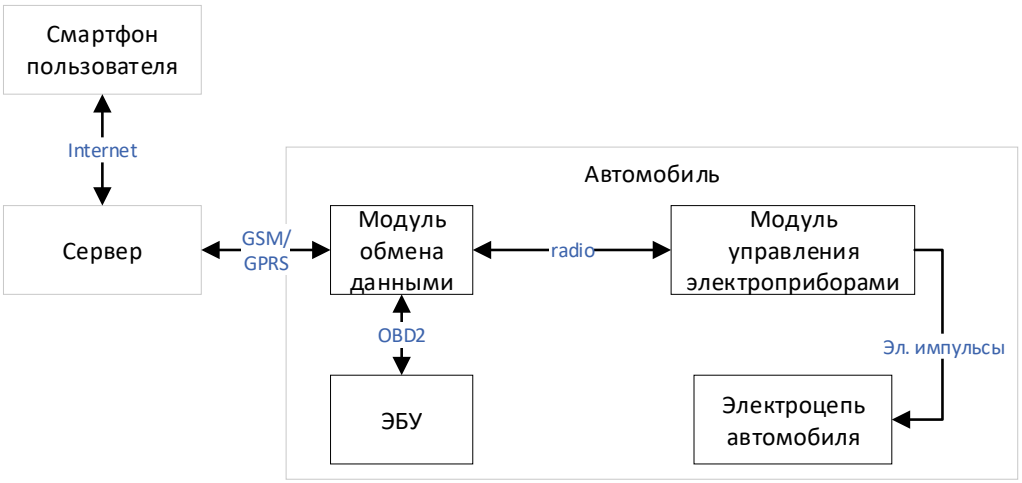


Рисунок 7 - структурная схема системы

Для установки сервера с минимальными требованиями подходит любой облачный хостинг.

Для обеспечения более стабильной работы и обеспечения отказоустойчивости рекомендуется использовать сервисы, предоставляющие серверное оборудование с RAID-массивами 1-го уровня и выше, а также с приемлемой производительностью аппаратной части (12 и более потоков CPU, 32Гб и более ОЗУ).

Для взаимодействия пользователю с системой необходимо иметь любое устройство поддерживающие веб-браузер и имеющие доступ к сети Internet.

4.2 Программное обеспечение

Перечень, характеристика и настройка общих программных средств: операционной системы, драйверов, инструментальных средств программирования, библиотек, СУБД, других дополнительных программных средств.

Для реализации проекта используются следующие средства программирования, СУБД и др.:

- PyCharm;
- SQlite;

- Git;
- Heroku;
- Arduino IDE;
- PowerDesigner.

PyCharm – это интегрированная среда разработки для языка программирования Python, предоставляющая средства для анализа кода, инструментарий для запуска юнит-тестов и графической отладки.

PyCharm был выбран для написания серверного приложения с использованием интерпретатора Pyhton 3.6. Конфигурация запуска/дебага приложения: модуль: “flask”, параметр запуска: “run”, переменные среды: “PYTHONUNBUFFERED=1;FLASK_APP=start.py;FLASK_DEBUG=1”.

SQLite – компактная встраиваемая СУБД, исходный код библиотеки которой является открытым.

Для SQLite термин «встраиваемый» характеризован тем, что движок данной СУБД не является отдельно работающим процессом (то есть не использует парадигму клиент-сервер), а который можно представить в виде библиотеки, которая подключается к программе и становится её составной частью. Для реализации такого подхода, используются вызовы функций (API) библиотеки SQLite для обмена данными. Результатом такого подхода является уменьшение накладных расходов, времени отклика и упрощение программы.

Для данной СУБД характерно депонирование базы данных (определения, таблицы, индексы и сами данные) в одном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Производить чтение данных из одной базы способны единовременно несколько процессов или потоков без каких-либо проблем. Осуществление

записи в базу возможно только в случае, когда нет других параллельно выполняющихся запросов. При попытке записи в этот, операция может закончиться отказом СУБД и возвращением кода ошибки. Существует также альтернативный вариант, когда попытка записи автоматически повторяется в течении заданного интервала времени.

Heroku — облачная PaaS-платформа, поддерживающая относительно небольшой ряд языков программирования. Является одной из первых облачных платформ, появилась в июне 2007 года и изначально поддерживала только язык программирования Ruby, однако на данный момент список поддерживаемых языков также включает в себя Java, Node.js, Scala, Clojure, Python, Go, Ruby и PHP, а на серверах Heroku используются операционные системы Debian или Ubuntu. Пример страницы работы с сервером приведён на рисунке 8.

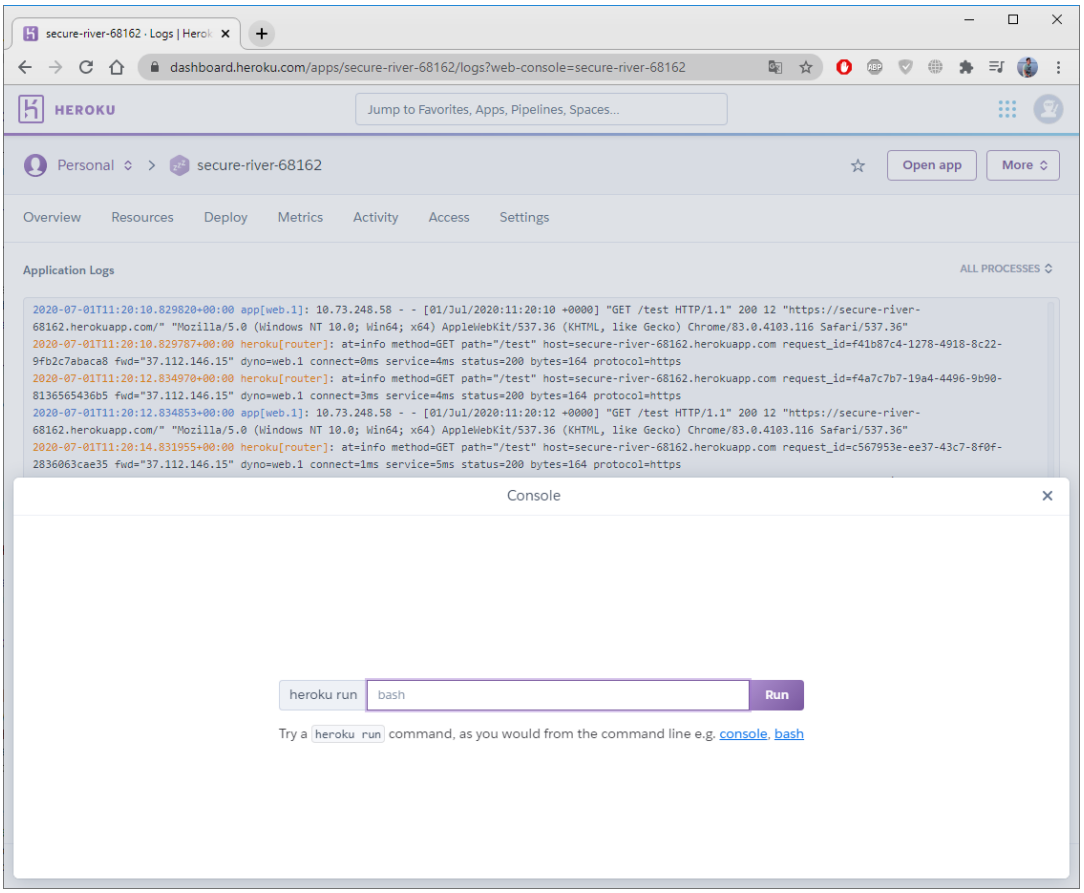


Рисунок 8 – скриншот примера работы с сервером в Heroku

Arduino IDE – интегрированная среда разработки (IDE) с открытом

кодом, позволяющее писать скетчи для плат Arduino или других, схожих с ними по конструкции, достаточно легко и обладающее встроенным функционалом компиляции кода и возможностью выгрузить его на микроконтроллер [6]. Данная IDE способна работать на Windows, MacOS и Linux. Пример работы с Arduino IDE показан на рисунке 9.

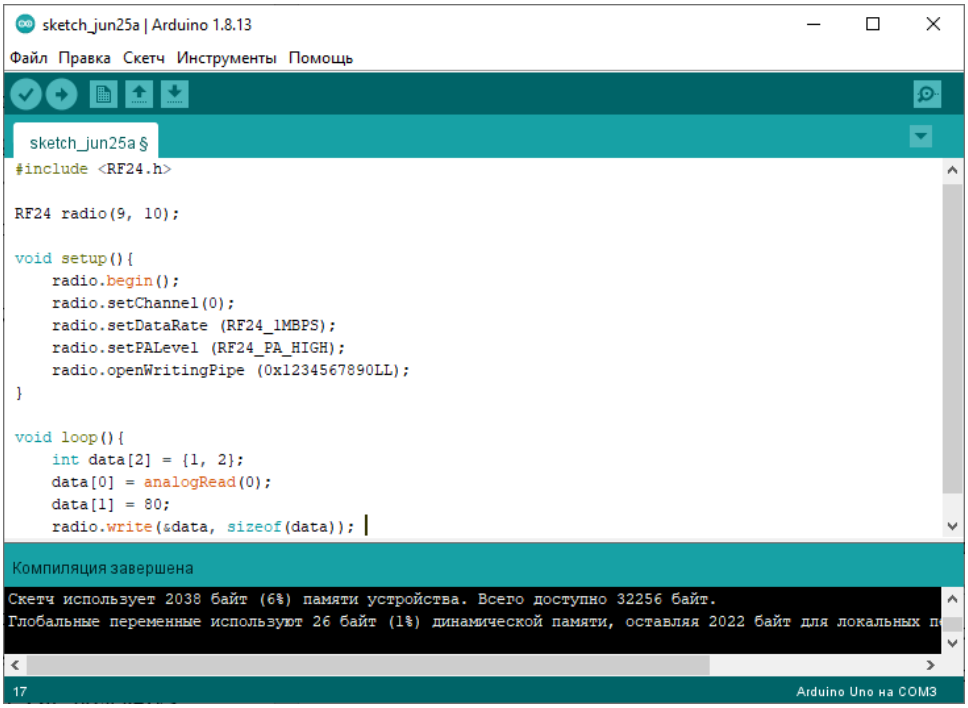


Рисунок 9 – скриншот работы в Arduino IDE

Arduino IDE поддерживает языки C и C++ с использованием специальных правил структурирования кода. Для запуска эскиза и основного цикла программы пользователю необходимо написать всего две базовые функции «void setup()» и «void loop()», которые будут скомпилированы и структурированы под заглушкой программы main() в исполняемую циклическую программу с помощью GNU toolchain (пакет программ, необходимых для компиляции и генерации выполняемого кода из исходных текстов), которая включена в дистрибутив IDE.

Данная среда использует программу avrdude способную преобразовать код в текстовый файл шестнадцатеричной кодировки, загружаемый в плату Arduino программой-загрузчиком. Для загрузки пользовательского кода в

качестве инструмента по умолчанию на официальные платы Arduino используется avrdude.

Sybase PowerDesigner – это средство для проектирования на UML-языке. Пример работы в PowerDesigner представлен на рисунке 10.

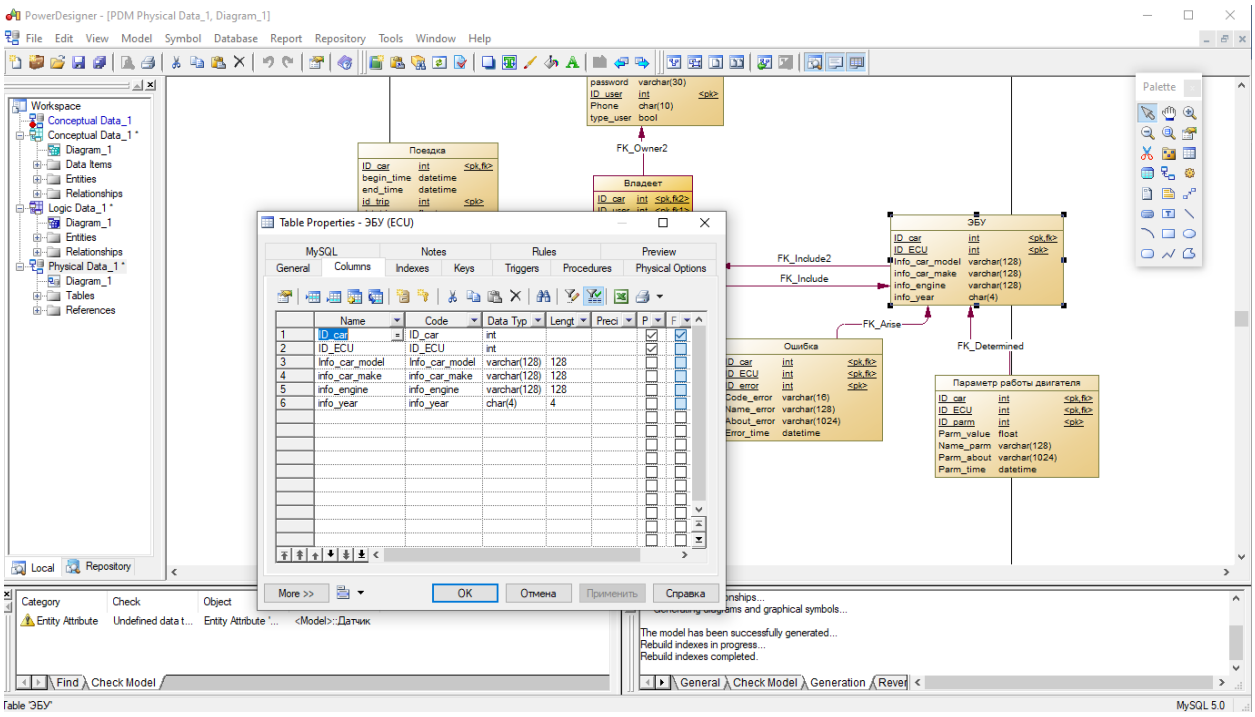


Рисунок 10 – скриншот рабочего пространства в PowerDesigner

PowerDesigner поддерживает следующие языки программирования (только для генерации кода): C#, C++, Java, PowerBuilder, VisualBasic. Так же доступно генерирование на XML и IDL. Возможно добавление собственных языков.

Поддерживаемые базы данных IBM DB2, Informix, Ingres, InterBase, Access, MS SQL, MySQL, Oracle, PostgeSQL, Sybase Anywhere, Enterprise.

Основные плюсы работы в PowerDesigner в:

- Обладает удобным и понятным интерфейсом;
- Высокий уровень реализации проектирования баз данных;
- Для коллективной разработки существует поддержка общего репозитория;
- Доступность использования большого количества визуальных

эффектов;

- Для генерации кода существует возможность создание новых и внесение изменений в старые шаблоны;
- Поддержка стандарта UML 2.0.

Git – это распределённая система управления версиями. Облегчает ведение контроль версий, а также совместную разработку. Так же используется для загрузки приложения на сервис Heroku [7].

4.3 Разработанные программные средства.

4.3.1 Описание использованных средств, подходов, методов, языков, библиотек

Для реализации системы был выбран функционально-ориентированный подход программирования. Серверное приложение состоит из набора функций, никак не связанных друг с другом, кроме обращения некоторых друг к другу.

Так же был применён паттерн MVC (model-view-control), предполагающий разделение приложения на уровни: представление, контроллер и модель, таким образом, что модифицирование каждого из уровней может осуществляться независимо [8]. Схема взаимодействия уровней представлена на рисунке 11.

Модель – представляет собой данные и управляется контроллером.

Контроллер – представляет собой набор функций, которые реагируют на действие пользователя и взаимодействуют с моделью, изменяя её.

Представление – верхний слой взаимодействия с пользователем. Реагирует на изменение в модели и передаёт команды на её изменение контролёру.

Также в рамках взаимодействия между веб-приложения, модулями и сервером была выбрана архитектура REST API. Эта архитектура заключается во взаимодействии путём вызова удалённой функции с помощью HTTP (или

HTTPS) запроса методами GET или POST.

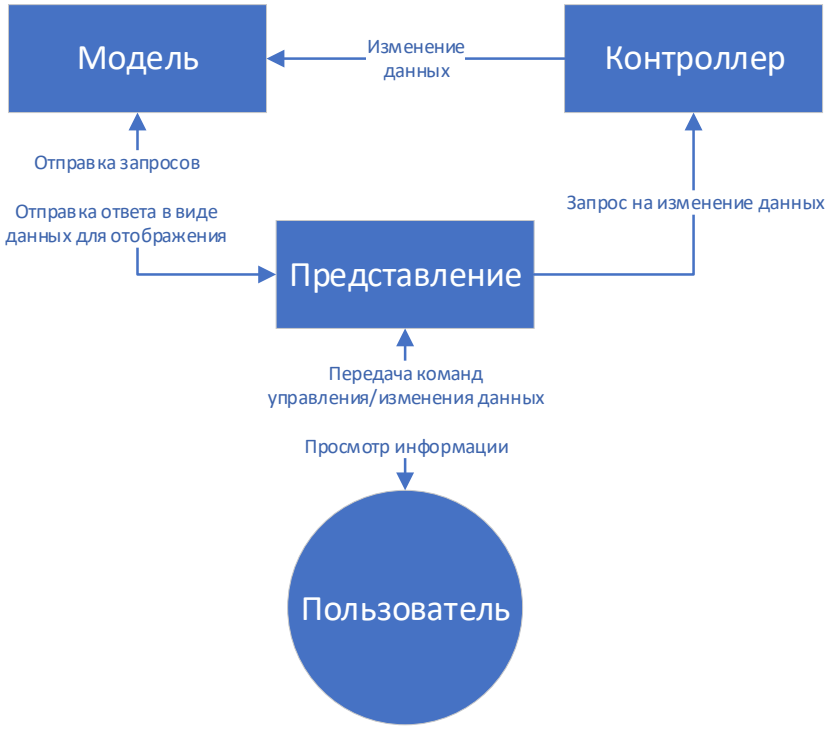


Рисунок 11 – схема взаимодействия уровней MVC

Для реализации серверной части и веб-приложения использовались следующее:

- Python – высокоуровневый язык программирования, отличающийся большой универсальностью и ориентированный на максимально простое понимание кода, со стороны разработчика;
- JavaScript – язык программирования, получивший большое распространение в проектировании веб-приложений. Используется для построения динамических страниц, а также написания серверных-приложений;
- Flask – фреймворк для создания веб-приложений. Использует инструментарий Werkzeug и Jinja2. Является микрофреймворком, то есть предоставляет минимальный набор базовых функций необходимых для создания приложения;
- jQuery – библиотека на языке JavaScript, облегчающая написание запросов на изменения представления. Помогает легко получить

доступ к любому элементу DOM, а также предоставляет инструментарий для использования ajax-запросов;

- AJAX – подход к построению динамических пользовательских интерфейсов веб-приложений, заключающийся в фоновом обращении браузера к веб-серверу.

Также для вёрстки страниц используются язык гипертекстовой разметки HTML и каскадные таблицы стилей CSS.

Также используются библиотеки:

- Werkzeug – обеспечивает шифрование данных. Используется для хеширования паролей;
- SQLAlchemy – обеспечивает подключение к БД и дополнительный уровень администрирования;
- Flask-Login – облегчает реализацию авторизации в системе;
- Gunicorn – используется для переноса приложения на хостинг в облачный сервис Heroku;
- Jinja2 – библиотека для создания шаблонов страниц веб-сайта.

Для реализации программной части модулей Arduino планируется использовать следующие библиотеки:

- softwareserial.h – библиотека для дополнительных программных портов UART;
- SIM900.h – библиотека для взаимодействия с контроллером SIM900R;
- AmperkaGPRS.h – библиотека для взаимодействия с GPRS/GSM Shield;
- rf24.h – библиотека для взаимодействия с модулями радиосвязи NRF24L01+;

ArduinoJson.h – библиотека добавляющая поддержку JSON-файлов.

4.3.2 Описание программы

4.3.2.1 Общие сведения

Веб-приложение «Система дистанционного контроля автомобиля» предназначена для взаимодействия пользователя с системой дистанционного контроля параметров и управления электрооборудованием автомобиля. Веб-приложение предоставляет основные органы управления пользователю: авторизация, управление электрооборудованием, в частности запуск двигателя, контроль параметров и ошибок работы двигателя.

Приложение реализовано с использованием языков программирования Python и JavaScript, а также языка гипертекстовой разметки HTML и каскадной таблицы стилей CSS. Более подробное описание, характеристика и настройка используемых технологий, приведено в пункте 4.2.

Для работы приложения необходим сервер, поддерживающий интерпретатор Python 3.6. Для запуска клиентского веб-приложения необходим веб-браузер, поддерживающий HTML5 и язык JavaScript. Для запуска клиентской части приложения необходимо иметь браузер, далее в нем вводится зарегистрированное доменное имя сайта. Вся работа с клиентской частью приложения ведется через веб-браузер. Для администрирования приложения необходимо подключиться к серверу и через командную консоль вводить непосредственно команды управления БД.

4.3.2.2 Функциональные назначения

Веб-приложение предназначено для управления пользователем системой дистанционного контроля параметров и управления электрооборудованием автомобиля. Для этого приложение предлагает элементы управления, а также вывод информации о автомобиле. Для решения этой задачи пользователь имеет следующие возможности:

- авторизация пользователя;
- регистрация пользователя;

- выход из системы;
- выбор автомобиля для управления;
- запуск/остановка двигателя;
- просмотр параметров работы двигателя;
- просмотр ошибок работы двигателя.

4.3.2.3 Описание логической структуры

На рисунке 12 представлена диаграмма состояний системы дистанционного контроля параметров и управления электрооборудованием автомобиля.

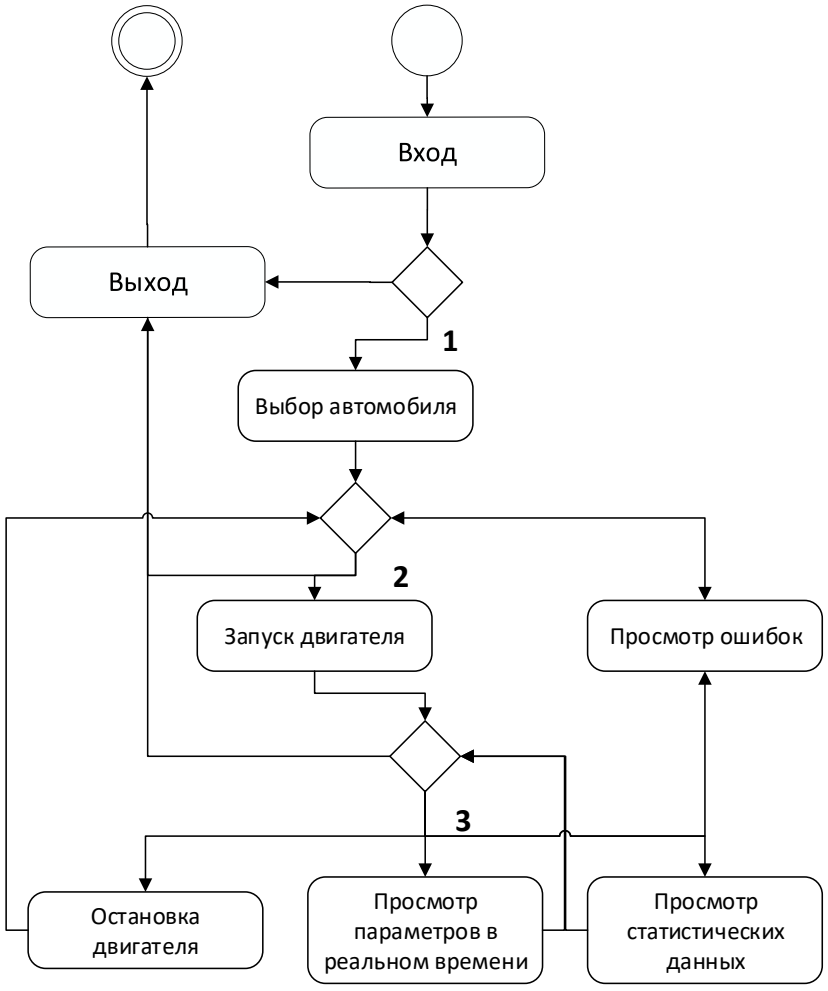


Рисунок 12 – диаграмма состояний

Последействия «вход пользователя» система переходит в состояние 1 из которого доступны действия выхода и выбор автомобиля. При выборе действия «выбор автомобиля», система предлагает пользователю список доступных ему автомобилей, и после выбора система переходит в состояние 2. В этом состоянии становиться доступен так же «запуск двигателя» и просмотр ошибок. После действия «запуск двигателя» «запуск двигателя» система переходит в состояние 3 из которого может вернуться в предыдущее состояние 2 посредством действия «остановка двигателя». В состоянии 3 доступны действия просмотра ошибок, просмотр параметров в реальном времени и статистических данных. Во всех состояниях системы пользователю доступен выход из системы.

Общий алгоритм программы

Основной процесс использования приложения состоит из следующих шагов:

1. Регистрация в приложении. На странице регистрации необходимо ввести уникальный логин и пароль к учетной записи, так же ввести своё ФИО и номер телефона для прохождения полной регистрации с администратором системы (для привязки автомобиля). Пароль необходимо вводить дважды.
2. Авторизация в приложении. На странице авторизации необходимо ввести ранее зарегистрированный логин и пароль.
3. После авторизации необходимо выбрать автомобиль, нажав кнопку «сменить автомобиль» и выбрав нужный автомобиль в открывшемся списке.
4. Для запуска двигателя нажать кнопку «Start».
5. Для просмотра параметров работы двигателя перейти на вкладку «параметры» в верхнем меню.
6. Для просмотра ошибок работы двигателя перейти на вкладку «ошибки» в верхнем меню.

7. Для остановки двигателя нажать кнопку «Stop» на главной странице приложения, перейдя на неё открыв вкладку «главная» в верхнем меню.
8. Для выхода из системы нажать кнопку «выход» на главной странице приложения, перейдя на неё открыв вкладку «главная» в верхнем меню.

Структура приложения

Общая структура приложения отображена на рисунке 13.

Рисунок 13 – структурная схема приложения

Структурно приложение состоит из БД, веб-серверного приложения и веб-страниц, представляющих интерфейс. То есть структура полностью описывается паттерном MVC, более подробное описание которого предоставлена в пункте 4.3.1.

При взаимодействии с базой данных используется подход ORM (объектно-реляционное-моделирование) [9]. Такой подход позволяет облегчить взаимодействие с базой данных. Обращение к БД выглядит в виде функций объекта «db.session» [10]. Нет необходимости продумывать SQL-запросы, чтобы максимально эффективно использовать реляционную БД. При получении корневой модели (таблицы), автоматически подтягиваются все

зависимые модели и добавляются во внутрь корневого. Таким образом взаимодействие с объектами БД происходит в виде взаимодействия похожего как взаимодействие с объектами похожего в ООП (объектно-ориентированном подходе).

Также плюсом использования ORM является простой переход от использования одного вида базы данных к другой. Для перехода от SQLite к более продвинутым базам данных, таких как MySQL или PostgreSQL, необходимо поменять всего пару строк кода, а настройках адаптера БД.

Вызовы удалённой процедуры происходят на основе архитектуры REST API. Его основные принципы приведены далее:

- уровневая структура приложения. REST API использует идентификатор ресурса для идентификации конкретного ресурса, участвующего во взаимодействии между компонентами.
- интерфейс взаимодействия клиента и имеется следующие ограничения:
  - ресурсы должны быть идентифицированы в запросе;
  - взаимодействие с ресурсом происходит через представление;
  - каждое сообщение должно содержать всю необходимую информацию для его обработки;
- в приложении выделятся сервер и клиент для разделения процессов работы с данными и их представлением клиенту/

4.3.2.4 Используемые технические средства

Серверная часть выполняется на ЭВМ типа IBM PC. Сервер сохраняет всё данные на жёсткий диск в БД, с помощью СУБД. Подробнее описано в пункте 4.1.

Работа клиента происходит в диалоговом режиме, используя экран и

устройство ввода для управления курсором, поэтому подойдёт любой устройство, поддерживающе веб-браузер.

4.3.2.5 Вызов и загрузка

Для запуска приложения, с точки зрения разработчика, необходимо разместить серверное приложение на хостинге.

Алгоритм размещения приложения на сервере с помощью облачного сервиса Heroku:

1. Создать аккаунт на Heroku и создать в личном кабинете проект для бедующего приложения.
2. Создать репозиторий git на компьютере в корневой папке проекта с сервером и создать необходимые файлы, которые указаны в документации Heroku для вашего проекта [6].
3. Выполнить установку Heroku CLI.
4. Выполнить вход в аккаунт Heroku через командную строку в корневой папке проекта.
5. Выполнить коммит и пуш на специальную ветку репозитория “heroku master”.

Данный алгоритм пригоден для серверов поддерживаемых Heroku, перечень которых представлен в их официальной документации.

По окончании алгоритма, приложение будет размещено на выделенном доменном адресе, который можно посмотреть в личном кабинете, в разделе проекта.

Повторив действия из приведённого выше алгоритма мы получаем сайт с URL-доступа: https://secure-river-68162.herokuapp.com/. Для проверки работоспособности был зарегистрирован пользователь с логином: “elena”, паролем: “4674” и ФИО: “Попова Елена Анатольевна”. Скриншот тестового входа в аккаунт приведён на рисунке 14. Скриншот успешного входа приведён на рисунке 15.

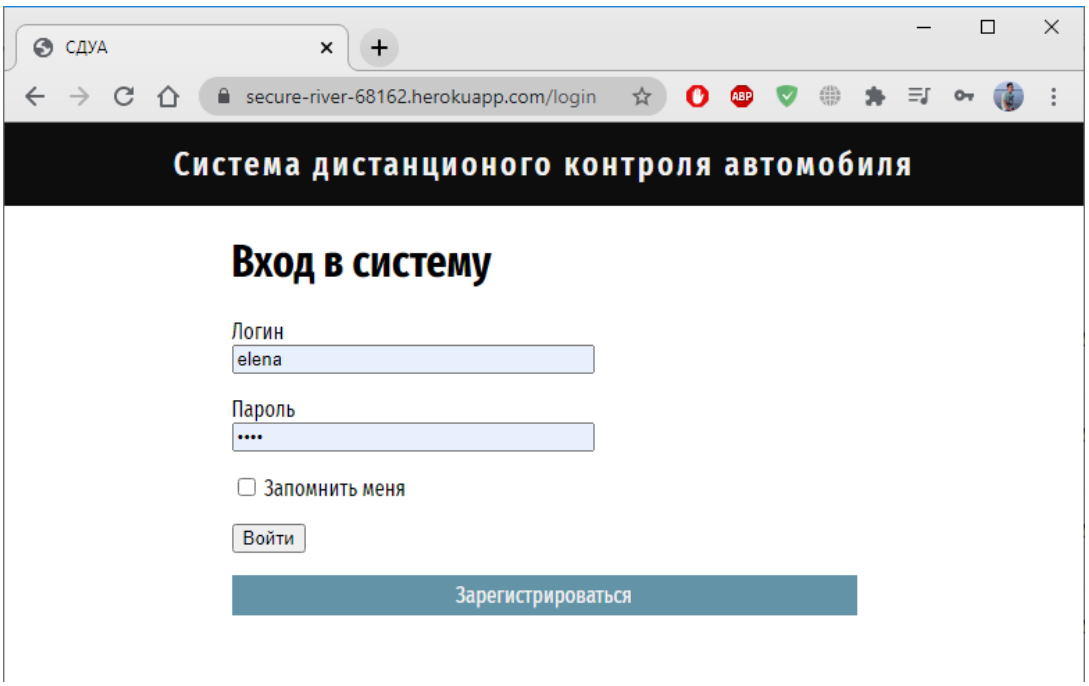


Рисунок 14 – скриншот тестового входа выгруженного приложения

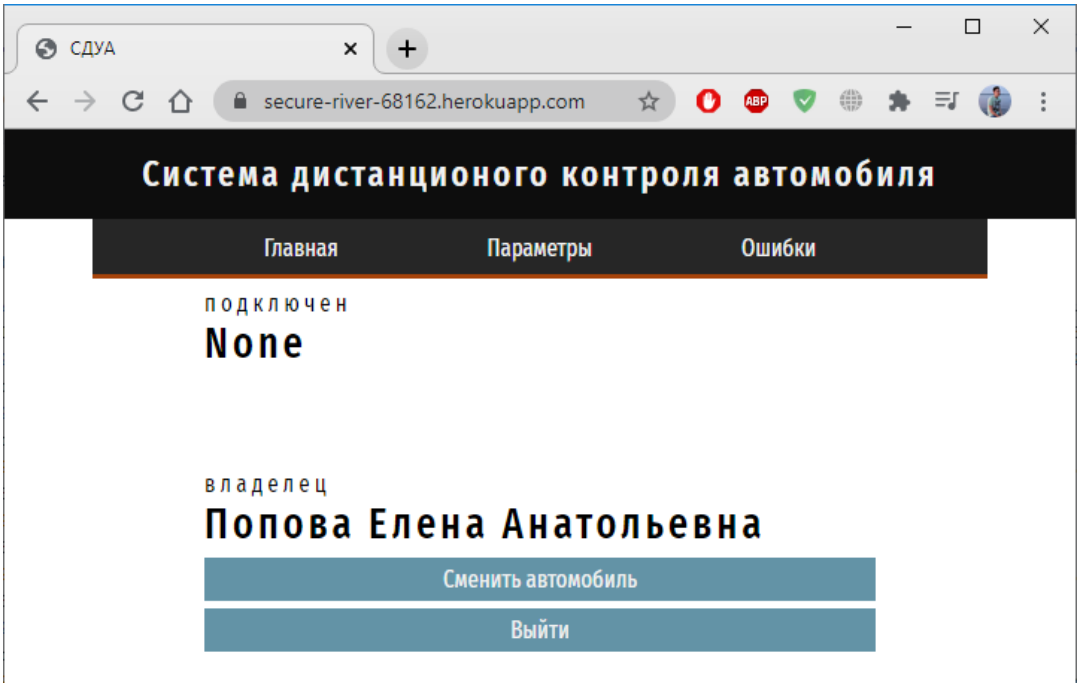


Рисунок 15 – скриншот успешного входа в систему

4.3.3 Входные данные

В приложение присутствуют команды пользователя, которые он вводит посредство графического интерфейса. Данные команды перечислены ниже:

- открытие страницы входа в систему;
- выполнение входа в систему;

- открытие страницы регистрации;
- выполнение регистрации пользователя в системе;
- открытие страницы со списком доступных автомобилей;
- выбор автомобиля для дальнейшего взаимодействия;
- запуск двигателя автомобиля, если двигателя не запущен;
- остановка двигателя, если двигатель запущен;
- открытие страницы просмотра параметров работы двигателя;
- открытие страницы просмотра ошибок работы двигателя;
- выход из системы.

Так же в приложении подразумевается ввод данных от пользователя при регистрации в системе:

- ввод логина;
- ввод пароля;
- повторный ввод пароля;
- ввод ФИО;
- ввод номера телефона пользователя.

И ввод данных при авторизации пользователя (входа в систему):

- ввод логина;
- ввод пароля.

Данные параметры добавляются в систему автоматически при взаимодействии пользователем с системой.

4.3.4 Выходные данные

Выходные данные приложения являются данные, предоставляемые пользователю, а именно:

- параметры работы двигателя в реальном времени;
- статистические параметры работы двигателя;
- ошибки работы двигателя;

- текущий выбранный автомобиль;
- ФИО авторизованного пользователя.

Так же в неявном виде пользователю предоставляется информация запущен ли двигатель автомобиля или нет (с помощью видоизменения изменения кнопки «Start/Stop») и выбран автомобиль или нет (с помощью скрытия кнопки «Start/Stop» и отображения «None» вместо наименования автомобиля).

4.3.5 Описание применения программы

4.3.5.1 Назначение программы

Приложение предназначено для повышения комфорта при использовании личного автотранспортного средства.

Для выполнения этой цели, в приложении реализован ряд функций, который был выявлен как необходимый в предпроектном исследовании, отображённый в пункте 1.

Приложение может занимать не более 105 Кб данных в ПЗУ, из расчёта кеширования страниц браузером.

Среднее время отклика обычной функции не более одной секунды. Среднее время отклика кнопки запуска/остановки двигателя составляет 4 секунды. Время обновления значений параметров работы двигателя около 6-12 секунд.

Для работы приложения необходимо иметь доступ к сети Интернет.

4.3.5.2 Условия применения

Минимальный состав технических средств для серверной части приложения.

В состав технических средств должен входить IBM-совместимый компьютер, включающий в себя:

- процессор с тактовой частотой, не менее ГГц – 1.300;
- оперативную память объемом, не менее 2Гб.

Минимальный состав программных средств.

Минимальный состав программных средств должен быть представлен облачным хостингом. Для реализации всех возможностей дополнительные программные средства не требуются. Вся настройка аппаратно-программной части указана в пункте 4.1 и 4.2.

Требования и условия организационного и технического характера.

Для взаимодействия клиента и микроконтроллера с сервером используется REST API архитектура, реализующая HTTP протокол обмена данными.

База данных встроена в серверное приложение и взаимодействие с ней происходит посредством вызова функций объекта «db.session».

Микроконтроллеры взаимодействуют друг с другом посредством передачи радиосигнала через модули NRF24L01+ на частоте 2401 Мг.

Организационно приложение предполагает многопользовательское использование. После покупки и установки физического модуля (набора микроконтроллеров для включения в электроцепь автомобиля), пользователь регистрируется в веб-приложении и обращается в службу технической поддержки (администрирования) системы, для того чтобы связать его аккаунт с его машиной.

К управлению одной машиной допускается несколько человек, однако только после удостоверения личности и подтверждения права управлять автомобилем.

Система нацелена на автовладельцев старых автомобилей, поддерживающих проток передачи данных OBD-II, а также новых автомобилей, для которых не предусмотрен подобный функционал.

4.3.5.3 Описание задачи

Задача системы состоит в передачи управляющей команды от пользователя, например, на запуск двигателя до машины и приёма ответа от машины и отображение пользователю результата.

Процесс запуска двигателя начинается с того, что пользователь авторизован и у него выбран нужный автомобиль. После нажатия кнопки «запустить двигатель» веб приложение выполняет HTTP запрос на сервер по адресу “…/start_engine”, где вызывается удалённая функция запуска двигателя, которая записывает в базу данных информацию, о том какую машину необходимо запустить.

В свою очередь, каждые 5 секунд модуль Arduino через GPRS/GSM Shield посылает HTTP GET запросы на веб сервер по адресу “…/something_for_me/<id>”, передавая ID машины, к которой она подключена. При очередном запросе сервер сравнивает ID из запроса от веб-приложения и ID от Arduino и при формировании ответа отправляет ей команду на запуск двигателя. Если модулю Arduino не удалось запустить двигатель, следующий HTTP GET запрос, так же придёт на адрес “…/something_for_me/<id>”, и сервер при сверке ID поймёт, что попытка запуска не удалась и вернёт отрицательный результат пользователю.

При успешной попытке запуска, Arduino отправляет HTTP GET запрос по адресу “…/Arduino/<id>/<parms>”, где добавляется параметр “parms” – содержащий в себе закодированную строку со значением параметров работы и ошибок двигателя. В таком случае сервер распознаёт, что запуск произошёл успешно и перерисовывает кнопку «Start» на «Stop».

При остановке двигателя операция повторяется в обратной последовательности.

4.3.5.6 Входные данные приложения

Входными данными приложения является данные пользователя

введённые при регистрации, а именно: логин, пароль, ФИО и номер телефона; данные пользователя введённые при авторизации: логин и пароль; а также управляющие команды введённые посредством взаимодействия с интерфейсом приложения: запуск/остановка двигателя, смена машины, открытие вкладок параметров и ошибок, кнопки входа, выхода и регистрации в системе.

4.3.5.7 Выходные данные приложения

Выходными данными является списки параметров работы двигателя и ошибок двигателя, а также ФИО пользователя, название текущего выбранного автомобиля, а также отображение его состояния заведён он или нет.

4.3.6 Описание результатов работы программы

В ходе работы было разработано приложение для управления системой дистанционного контроля и управления электрооборудованием автомобиля с использованием микроконтроллера.

В приложении реализованы следующие функции:

- авторизации;
- регистрации;
- запуск и остановка двигателя автомобиля;
- выбор используемого автомобиля пользователем;
- просмотр текущих параметров работы двигателя:
  - температура охлаждающей жидкости;
  - скорость вращения двигателя;
  - скорость автомобиля;
  - напряжение борт. сети;
  - часовой расход топлива;
  - путевой расход топлива;

- цикловой расход воздуха;
- массовый расход воздуха;
- положение дроссельной заслонки;
- положение регулятора холостого хода;
- коэффициент коррекции времени впрыска;
- угол опережения зажигания;
- признак выключения двигателя;
- признак холостого хода;
- средний расход топлива – статистический параметр.

- просмотр ошибок работы двигателя. Возможный перечень ошибок [11]:
  - P0120 – неисправность в электрической цепи датчика положения дроссельной заслонки;
  - P0501 – Неисправность датчика скорости автомобиля;
  - P0560 – напряжение системы (бортовой сети) – неисправность;
- получение данных от модулей и внесение их в БД.

Дистанционный запуск двигателя позволяет заранее заводить двигатель для прогрева, что увеличивает срок службы двигателя и уменьшает время необходимое на подготовку перед выездом.

Контроль параметров работы двигателя, сбор статистических данных повышает комфорт от использования автомобиля и позволяет отслеживать интересующие параметры в реальном времени. В совокупности с просмотром ошибок работы двигателя, это позволит также увеличить срок службы двигателя за счёт своевременного выявления неполадок и их устранения.

5 Результаты внедрения и использования системы. Достижение целей разработки.

В ходе работы была разработано веб-приложение, позволяющее управлять дистанционно автомобилем, отслеживать параметры работы двигателя в реальном времени и просматривать появляющиеся ошибки.

Приложение позволяет авторизовать зарегистрированного пользователя, и зарегистрировать нового. С помощью администрирования сервера через консоль, завершается процесс регистрации, созданием машины в БД или привязке к пользователю уже существующего автомобиля.

Для корректной проверки функционала был разработан эмулятор ЭБУ, который имитирует передачу данных от самого ЭБУ на микроконтроллер. Функции, задающие значения параметров выполнены в виде генераторов случайных чисел. Ошибки были прописаны заранее и внесены в базу данных.

На рисунке 16 представлена страница входа в систему.

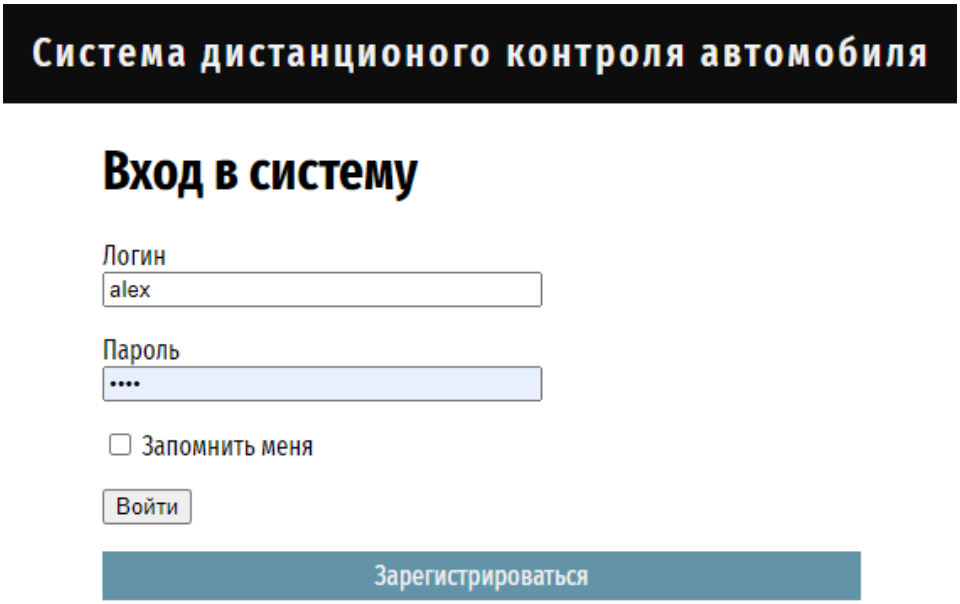


Рисунок 16 – скриншот страницы входа в систему

На рисунке 17 представлена страница регистрации в системе.

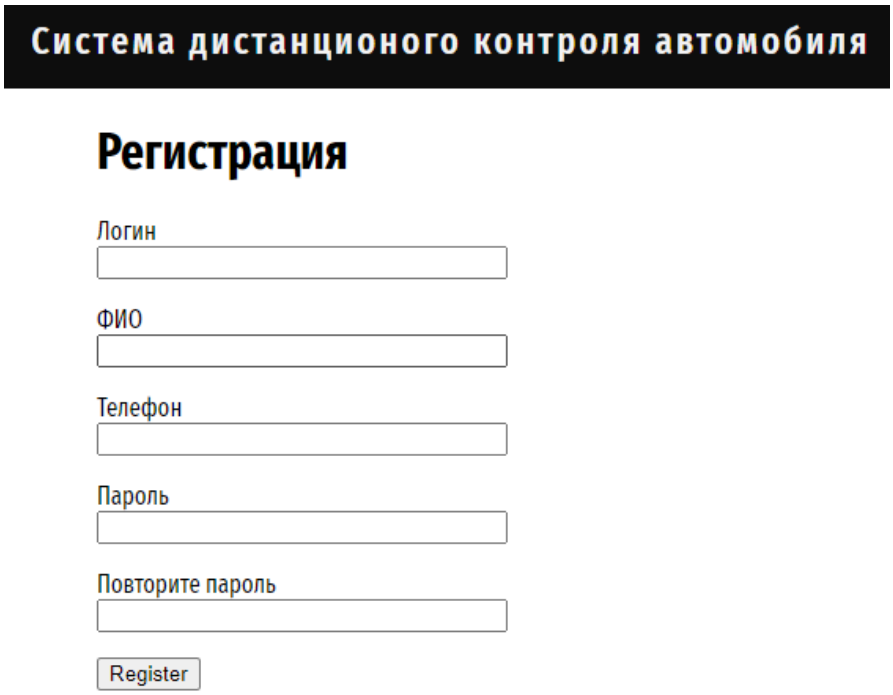


Рисунок 17 – скриншот страницы регистрации в системе

На главной странице, которая представлена на рисунке 18 отображается наименование автомобиля, к которому выполнено подключение в настоящий момент, ФИО текущего пользователя, а также расположены кнопки для запуска/остановки двигателя, смены автомобиля и выхода из системы.

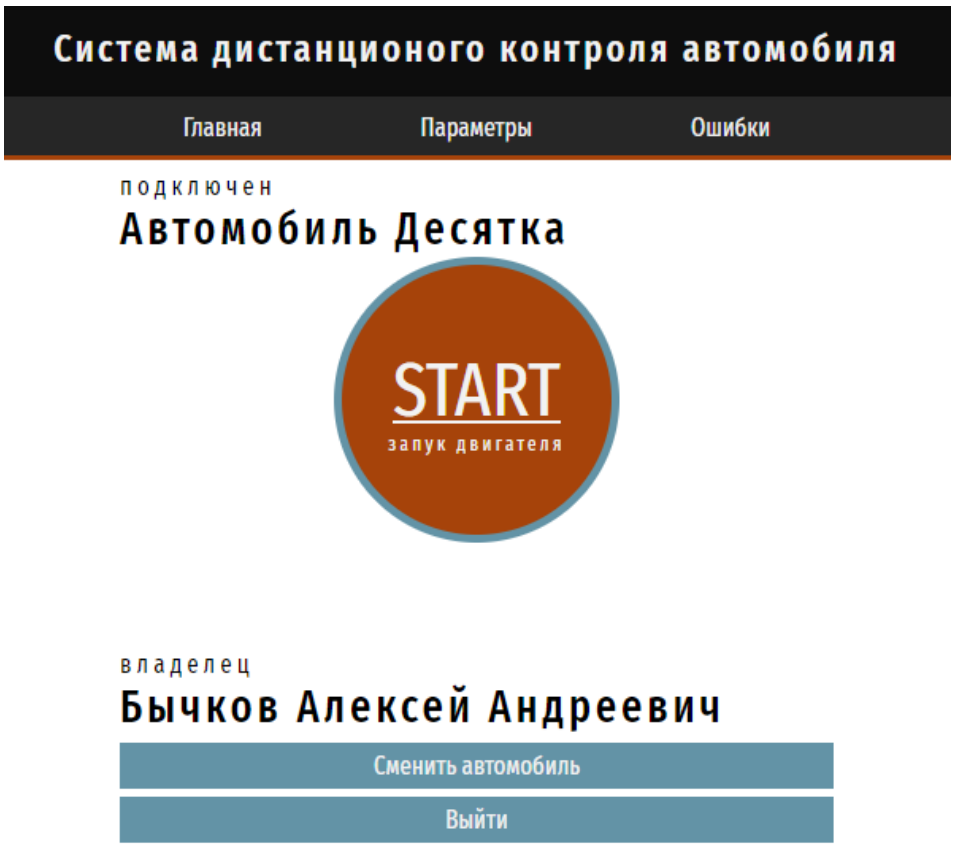


Рисунок 18 – главная страница

На странице сены машины пользователя, представленной на рисунке 19, отображается список автомобилей, принадлежащих пользователю, который можно выбрать, щёлкнув по нужному авто.

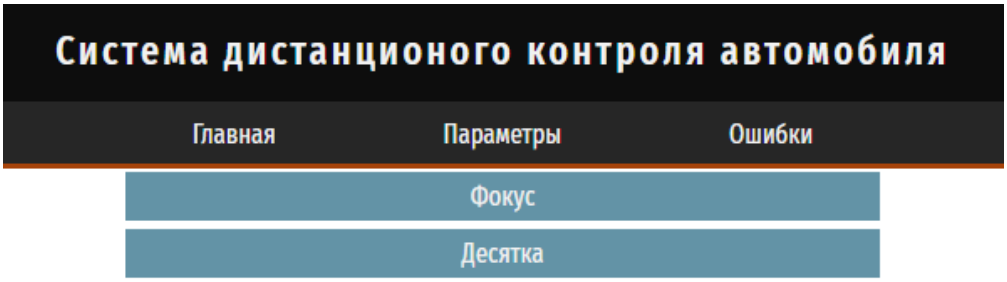


Рисунок 19 – страница выбора автомобиля

На странице «Параметры», представленной на рисунке 20, отображается набор параметров работы двигателя в реальном времени, получаемых от системы.

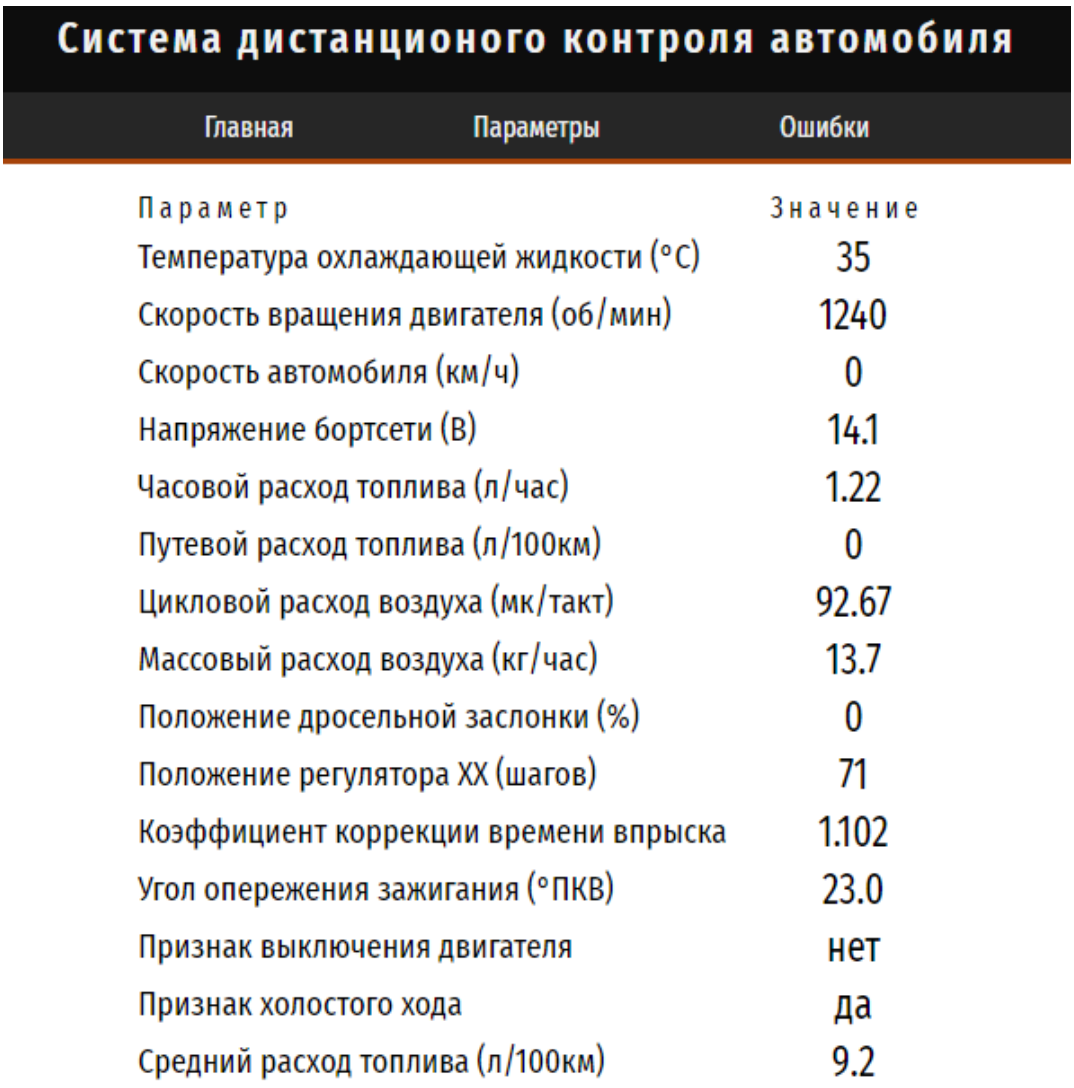


Рисунок 20 – страница «Параметры»

На странице «Ошибки», представленной на рисунке 21, отображаются все ошибки, которые были зафиксированы системой.

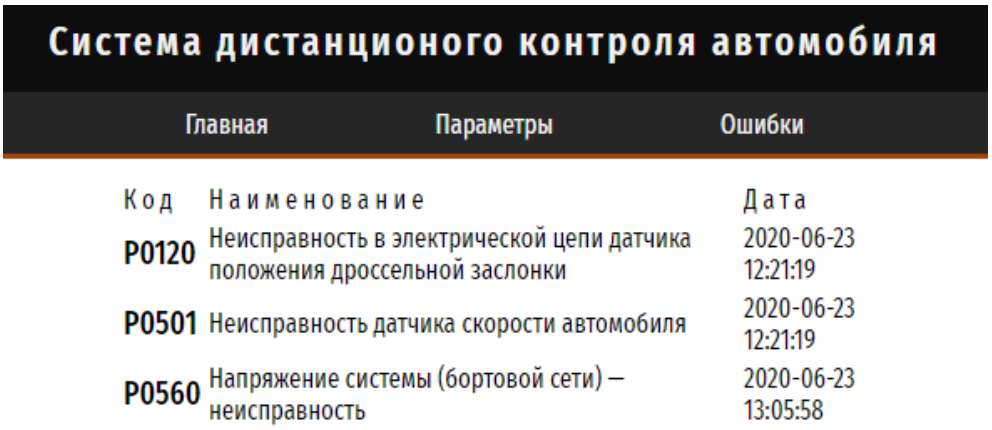


Рисунок 21 – страница «Ошибки»

В результате реализации были достигнуты следующие цели проекта:

1. Разработано веб-приложение, позволяющее выполнять дистанционный запуск двигателя.
2. Реализован функционал для отслеживания параметров работы двигателя в реальном времени, а также сбора данных для анализа и формирования статистических параметров.
3. Реализован функционал просмотра ошибок работы двигателя, получаемых от ЭБУ автомобиля.
4. Повышение комфорта при использовании автомобиля.

Заключение

В ходе работы была спроектирована архитектура системы и реализовано веб-приложение для управления ею с гибкой архитектурой на базе REST API и паттерна MVC, позволяющие изменять необходимые параметры в соответствии с возможными целями и ограничениями. Данное приложение реализует процесс дистанционного контроля параметров и управления электрооборудованием автомобиля.

В системе используются наработки в области мобильной передачи данных, взаимодействия с электрооборудованием и электроцепями, а также рассматриваются протоколы передачи данных для ЭБУ автомобиля. В свою очередь приложение предоставляет простой и удобный интерфейс для взаимодействия с системой, которая поможет увеличить комфортабельность использования старых автомобилей.

Система сконструирована так, что подразумевает дальнейшее развитее и наращивание мощностей для увеличения функционала и горизонтальной масштабируемости.

Приложение так же разработано с использованием архитектурных принципов, позволяющих развивать и изменять его без существенных затрат на работу с уже существующей базой. Также присутствует некоторая независимость от библиотек и фреймворков, что позволяет беспрепятственно их менять уже существующие, за исключением основных базовых библиотек.

Список источников

1. Сообщество автовладельцев [Электронный ресурс]: веб-форум / – Режим доступа: https://www.drive2.ru/communities/, свободный

2. Анализ SSL/TLS трафика в Wireshark / Блог компании Nexign / Хабр [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://habr.com/ru/company/billing/blog/261301/, свободный

3. Алгоритмы шифрования в автосигнализациях – Википедия [Электронный ресурс]: электронная энциклопедия /. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/Алгоритмы_шифрования_в_автосигнализациях, свободный

4. Keyword Protocol 2000: Спецификация канала связи с диагностическим оборудованием – Уровень обмена данными. – АО “АвтоВАЗ” Генеральный Департамент Развития Управление Проектирования Электроники и электрооборудования, 2000. – 42с.

5. ELM327 OBD-II to RS 232 interpreter [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://www.elmelectronics.com/wp-content/uploads/2017/01/ELM327DS.pdf, свободный

6. Arduino IDE – Википедия [Электронный ресурс]: электронная энциклопедия /. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/Arduino_IDE

7. Getting Started on Heroku with Python | Heroku Dev Center [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://devcenter.heroku.com/articles/getting-started-with-python, свободный

8. Знакомство с паттерном MVC (Model-View-Controller) [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://javarush.ru/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller, свободный

9. Что такое ORM [Электронный ресурс] /. – Электрон. текстовые

дан. – Режим доступа: https://myrusakov.ru/what-is-it-orm-php.html, свободный

10. Flask-SQLAlchemy – Документация [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://flask-sqlalchemy-russian.readthedocs.io/ru/latest/index.html, свободный

11. Расшифровка диагностических кодов протокола OBD-II [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://chiptuner.ru/content/obdcod/, свободный

Приложение А

Таблица 1. Список таблиц

№	Имя	Код
1	Пользователь	User_
2	Автомобиль	Automobile
3	Ошибка	Error_
4	ЭБУ	ECU
5	Параметр работы двигателя	Parm_engine
6	Датчик	Sensor
7	Управляющее реле	Relay
8	Владеет	Own

Таблица 2. Таблица «Пользователь»

Пользователь (User_)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ФИО	FIO	Variable characters (100)	No	No
Логин	Login	Variable characters (100)	Yes	No
Пароль	Pass	Variable characters (100)	Yes	No
ID Пользователя	ID_user	Integer	Yes	Yes
Номер телефона	Telephone	Characters (10)	No	No
Зависимости				
Имя		Код		
Владеет		Own_		

Таблица 3. Таблица «Автомобиль»

Автомобиль (Automobile)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID ЭБУ	ID_ECU	Integer	Yes	Yes
Наименование автомобиля	Name_auto	Variable characters (1024)	Yes	No
ID автомобиля	ID_auto	Integer	Yes	Yes
Зависимости				
Принадлежит		Belong		
Имеет		Have		
Передаёт данные		Tr_data		
Принимает данные		Rec_data		
Управляется		Operate		

Таблица 4. Таблица «Ошибка»

Ошибка (Error_)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID ошибки	ID ошибки	Integer	Yes	Yes
ID ЭБУ	ID ЭБУ	Integer	Yes	No
Код ошибки	Код ошибки	Variable characters (100)	Yes	No
Наименование ошибки	Наименование ошибки	Variable characters (1024)	No	No
Описание ошибки	Описание ошибки	Text	No	No
Зависимости				
Имя		Код		
Возникает		Arise		

Таблица 5. Таблица «ЭБУ»

ЭБУ (ECU)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID ЭБУ	ID ЭБУ	Integer	Yes	Yes
ID автомобиля	ID автомобиля	Integer	Yes	No
Данные о автомобиле	Data_about	Text	No	No
Зависимости				
Возникает		Arise		
Определяет		Determine		
Передаёт данные		Tr_data		
Принимает данные		Rec_data		

Таблица 6. Таблица «Параметр работы двигателя»

Параметр работы двигателя (Parm_engine)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID параметра	ID параметра	Integer	Yes	Yes
ID ЭБУ	ID ЭБУ	Integer	Yes	No
Значение параметра	Значение параметра	Float	Yes	No
Наименование параметра	Наименование параметра	Variable characters (100)	Yes	No
Описание параметра	Описание параметра	Text	No	No
Среднее значения параметра	Среднее значения параметра	Float	No	No
Зависимости				

Имя	Код
Определяет	Determine

Таблица 7. Таблица «Датчик»

Датчик (Sensor)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID датчика	ID датчика	Integer	Yes	Yes
ID ЭБУ	ID ЭБУ	Integer	Yes	No
ID автомобиля	ID автомобиля	Integer	Yes	No
Наименование датчика	Наименование датчика	Variable characters (1024)	Yes	No
Управляющий контакт микроконтроллера датчика	Управляющий контакт микроконтроллера датчика	Variable characters (13)	Yes	No
Зависимости				
Имя		Код		
Имеет		Have		

Таблица 8. Таблица «Управляющее реле»

Управляющее реле (Relay)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID реле	ID реле	Integer	Yes	Yes
ID ЭБУ	ID ЭБУ	Integer	Yes	No
ID автомобиля	ID автомобиля	Integer	Yes	No
Наименование элекмента управления	Наименование элекмента управления	Variable characters (150)	Yes	No
Управляющий контакт микроконтроллера реле	Управляющий контакт микроконтроллера	Variable characters (3)	Yes	No
Зависимости				
Имя		Код		
Управляется		Operate		

Таблица 9. Таблица «Владеет»

Владеет (Own)				
Столбцы				
Имя	Код	Тип данных	Обяз.	Перв.
ID Пользователя	ID Пользователя	Integer	Yes	Yes
ID ЭБУ	ID ЭБУ	Integer	Yes	Yes
ID автомобиля	ID автомобиля	Integer	Yes	Yes

Зависимости	
Имя	Код
Владеет	Own_
Принадлежит	Belong