МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

Курсовая работа на тему

Симулятор диагностического ПО для автомобиля

по дисциплине

Функциональное и логическое программирование

РУКОВОДИТЕЛЬ:

Суркова А.С.

СТУДЕНТ:

Малинок С.М.

гр. 21-ПО

Работа защищена « »

С оценкой

Нижний Новгород 2024

Оглавление

[Введение 2](#_Toc167559372)

[Необходимая справочная информация по используемым узлам 3](#_Toc167559373)

[Особенности автомобильной диагностики 15](#_Toc167559374)

[Особенности функционального программирования 17](#_Toc167559375)

[Выбор языка 18](#_Toc167559376)

[Описание переменных, функций 19](#_Toc167559377)

[Диаграмма работы программы 20](#_Toc167559378)

[Диаграмма работы программы с пользователем 21](#_Toc167559379)

[Используемая библиотека для графического интерфейса 22](#_Toc167559380)

[Используемая литература 23](#_Toc167559381)

[Приложение 24](#_Toc167559382)

# Введение

Современные автомобильные системы становятся все более сложными и высокотехнологичными, что приводит к увеличению требований к диагностике и обслуживанию автомобилей. Разработка программного обеспечения для симуляторов автодиагноста является важным шагом в подготовке квалифицированных специалистов, способных эффективно выявлять и устранять неисправности в сложных автомобильных системах.

Основная цель данной курсовой работы - разработка программного обеспечения для симуляции неисправностей автомобиля, который позволит обучать и тренировать будущих специалистов по автомобильной диагностике. Это программное обеспечение должно предоставлять пользователю возможность проводить диагностику различных систем автомобиля, выявлять и анализировать ошибки, а также разрабатывать эффективные стратегии их устранения.

В работе рассматриваются неисправности связанные с такими элементами двигателя как: дроссельная заслонка, ДПРВ (Датчик положения распределительного вала), ДПКВ (Датчик положения коленчатого вала), иммобилайзер и абс, а так же присутствует возможность настройки этих систем

Данная курсовая работа направлена на создание инструмента, который будет полезен как для учебных заведений, готовящих специалистов в области автосервиса, так и для самих специалистов, стремящихся улучшить свои навыки диагностики современных автомобильных систем.

# Необходимая справочная информация по используемым узлам

**Дроссельная заслонка** (также известная как дроссельный клапан) является важным компонентом системы впуска воздуха в двигателе внутреннего сгорания. Она регулирует количество воздуха, поступающего в двигатель, и, соответственно, влияет на его мощность и скорость. Основные аспекты работы и устройства дроссельной заслонки включают следующие элементы:

Основные функции

Регулировка воздушного потока: Дроссельная заслонка управляет объемом воздуха, поступающего в цилиндры двигателя. Это осуществляется за счет изменения угла открытия заслонки, что влияет на количество воздуха, смешиваемого с топливом для образования горючей смеси.

Управление оборотами двигателя: Положение дроссельной заслонки напрямую связано с положением педали акселератора. Когда водитель нажимает на педаль, заслонка открывается, увеличивая подачу воздуха и повышая обороты двигателя.

Контроль за уровнем выбросов: Современные системы управления двигателем используют данные с датчиков положения дроссельной заслонки для оптимизации состава топливовоздушной смеси, что помогает снизить выбросы вредных веществ.

Конструкция и компоненты

Корпус дроссельной заслонки: Это корпус, в котором размещена заслонка. Обычно он изготавливается из алюминия или пластика и соединен с впускным коллектором двигателя.

Заслонка: Это подвижный элемент (диск или пластина), который вращается вокруг своей оси, регулируя проходное сечение для воздушного потока.

Датчик положения дроссельной заслонки (TPS): Этот датчик измеряет угол открытия заслонки и передает данные в блок управления двигателем (ECU). На основании этих данных ECU регулирует подачу топлива и зажигание.

Электропривод: В современных автомобилях дроссельная заслонка часто оснащена электроприводом, который управляется электронным блоком. Такая система называется "электронная дроссельная заслонка" или "drive-by-wire".

Пружина возврата: В механических системах заслонка снабжена пружиной, возвращающей её в закрытое положение при отпускании педали акселератора.

Принцип работы

Когда водитель нажимает на педаль акселератора, дроссельная заслонка открывается, позволяя большему количеству воздуха поступать в двигатель. В традиционных механических системах заслонка соединена с педалью акселератора тросом или рычагами. В системах drive-by-wire датчики и электроприводы обеспечивают электронное управление заслонкой.

Электронный блок управления (ECU) использует данные от датчика положения дроссельной заслонки и других датчиков для точного регулирования подачи топлива и угла опережения зажигания, обеспечивая оптимальную работу двигателя при различных режимах эксплуатации.

Обслуживание и проблемы

Загрязнение: Со временем дроссельная заслонка может покрываться нагаром и загрязнениями, что может приводить к нестабильной работе двигателя на холостом ходу и снижению отклика на педаль акселератора. Регулярная чистка помогает поддерживать её в рабочем состоянии.

Неисправности датчика TPS: Проблемы с датчиком положения дроссельной заслонки могут вызывать ошибки в работе двигателя, такие как рывки, остановки или снижение мощности. В таком случае необходимо заменить датчик.

Проблемы с электроприводом: В электронных системах неисправности электропривода могут приводить к неправильной работе заслонки. Диагностика и ремонт требуют специальных инструментов и оборудования.

Дроссельная заслонка является ключевым элементом системы управления двигателем, влияющим на его производительность, экономичность и уровень выбросов. Ее правильная работа и своевременное обслуживание играют важную роль в обеспечении надежной и эффективной эксплуатации автомобиля.

**Датчик положения распределительного вала** (также известный как датчик фаз, CMP — Camshaft Position Sensor) является важным компонентом системы управления двигателем внутреннего сгорания. Он отслеживает положение распределительного вала, обеспечивая точное управление моментом впрыска топлива и углом опережения зажигания. Основные аспекты работы и особенности датчика положения распределительного вала включают следующие элементы:

Основные функции

Определение положения распределительного вала: Датчик фиксирует угловое положение распределительного вала, передавая эту информацию в блок управления двигателем (ECU).

Синхронизация работы двигателя: Датчик позволяет ECU синхронизировать работу системы впрыска топлива и системы зажигания с фазами работы цилиндров, обеспечивая эффективное сгорание топливной смеси.

Контроль фаз газораспределения: В двигателях с изменяемыми фазами газораспределения (VVT) датчик положения распределительного вала играет ключевую роль в управлении моментом открытия и закрытия клапанов.

Конструкция и компоненты

Корпус датчика: Обычно изготавливается из пластика или металла, защищает внутренние компоненты датчика от внешних воздействий.

Магнитный или оптический элемент: Большинство датчиков используют магнитные или оптические элементы для обнаружения движения и положения распределительного вала. Магнитные датчики генерируют сигнал на основе изменений магнитного поля, а оптические — на основе прерывания светового луча.

Электронный блок: Внутри датчика находится электронная схема, которая преобразует сигнал от магнитного или оптического элемента в электрические импульсы, передаваемые в ECU.

Соединительные провода и разъемы: Обеспечивают электрическое соединение датчика с блоком управления двигателем.

Принцип работы

Датчик положения распределительного вала устанавливается рядом с распределительным валом и фиксирует его вращение. В зависимости от конструкции датчика, он может работать следующим образом:

Магнитные датчики: В магнитном датчике используется постоянный магнит и катушка. Когда зубья или пазы на шестерне распределительного вала проходят мимо датчика, это вызывает изменения в магнитном поле, которые катушка преобразует в электрические сигналы. Эти сигналы передаются в ECU.

Оптические датчики: В оптическом датчике используется светодиод и фотодетектор. Когда зубья или прорези на шестерне распределительного вала прерывают световой луч, это фиксируется фотодетектором, который преобразует прерывания в электрические сигналы, передаваемые в ECU.

Эти сигналы позволяют блоку управления двигателем точно определить положение распределительного вала и синхронизировать работу системы впрыска топлива и зажигания.

Обслуживание и проблемы

Износ и загрязнение: Со временем датчик может загрязняться маслом, грязью или нагаром, что может ухудшить его работу. Регулярная чистка и проверка датчика помогут поддерживать его в рабочем состоянии.

Неисправности датчика: Поломки или износ элементов датчика могут привести к неправильной работе двигателя, включая проблемы с запуском, потерю мощности и увеличенные выбросы. В таких случаях требуется замена датчика.

Проблемы с проводкой: Повреждение проводов или разъемов может вызвать перебои в передаче сигналов от датчика к ECU. Проверка и ремонт проводки помогут устранить такие проблемы.

Важность датчика положения распределительного вала

Датчик положения распределительного вала является критически важным элементом системы управления двигателем. Он обеспечивает точное управление фазами работы цилиндров, что необходимо для оптимального функционирования двигателя. Своевременное обслуживание и замена неисправных датчиков помогают поддерживать высокую производительность и надежность работы автомобиля.

**Датчик положения коленчатого вала** (также известный как датчик коленвала, CKP — Crankshaft Position Sensor) является важным компонентом системы управления двигателем внутреннего сгорания. Он отслеживает положение и скорость вращения коленчатого вала, что необходимо для синхронизации работы системы впрыска топлива и зажигания. Основные аспекты работы и особенности датчика положения коленчатого вала включают следующие элементы:

Основные функции

Определение положения коленчатого вала: Датчик фиксирует угловое положение коленчатого вала и передает эту информацию в блок управления двигателем (ECU).

Измерение скорости вращения: Датчик также измеряет скорость вращения коленчатого вала, что помогает блоку управления двигателем регулировать работу двигателя в различных режимах.

Синхронизация работы двигателя: Данные от датчика используются для точной синхронизации впрыска топлива и зажигания, что обеспечивает эффективное сгорание топливной смеси и оптимальную работу двигателя.

Конструкция и компоненты

Корпус датчика: Обычно изготавливается из пластика или металла и защищает внутренние компоненты датчика от внешних воздействий.

Магнитный или индуктивный элемент: В большинстве датчиков используется магнитный или индуктивный элемент для обнаружения движения и положения коленчатого вала. Магнитные датчики генерируют сигнал на основе изменений магнитного поля, а индуктивные — на основе индукции электрического тока.

Электронный блок: Внутри датчика находится электронная схема, которая преобразует сигнал от магнитного или индуктивного элемента в электрические импульсы, передаваемые в блок управления двигателем.

Соединительные провода и разъемы: Обеспечивают электрическое соединение датчика с блоком управления двигателем.

Принцип работы

Датчик положения коленчатого вала устанавливается рядом с зубчатым колесом (шкивом) на коленчатом вале. В зависимости от конструкции датчика, он может работать следующим образом:

Индуктивные датчики: В индуктивных датчиках используется катушка и магнит. Когда зубья шестерни проходят мимо датчика, это вызывает изменения в магнитном поле, которые катушка преобразует в электрические сигналы. Эти сигналы передаются в блок управления двигателем.

Магниторезистивные датчики: В магниторезистивных датчиках используется магниторезистивный элемент, который изменяет свое сопротивление в ответ на изменения магнитного поля, создаваемого зубьями шестерни. Эти изменения преобразуются в электрические сигналы, передаваемые в блок управления двигателем.

Холловые датчики: В Холловых датчиках используется эффект Холла для измерения изменений в магнитном поле, создаваемом зубьями шестерни. Эти изменения преобразуются в электрические сигналы, передаваемые в блок управления двигателем.

Обслуживание и проблемы

Износ и загрязнение: Со временем датчик может загрязняться маслом, грязью или нагаром, что может ухудшить его работу. Регулярная чистка и проверка датчика помогут поддерживать его в рабочем состоянии.

Неисправности датчика: Поломки или износ элементов датчика могут привести к неправильной работе двигателя, включая проблемы с запуском, потерю мощности и увеличенные выбросы. В таких случаях требуется замена датчика.

Проблемы с проводкой: Повреждение проводов или разъемов может вызвать перебои в передаче сигналов от датчика к блоку управления двигателем. Проверка и ремонт проводки помогут устранить такие проблемы.

Важность датчика положения коленчатого вала

Датчик положения коленчатого вала является критически важным элементом системы управления двигателем. Он обеспечивает точное управление синхронизацией работы системы впрыска топлива и зажигания, что необходимо для оптимальной работы двигателя. Своевременное обслуживание и замена неисправных датчиков помогают поддерживать высокую производительность и надежность работы автомобиля.

**Кислородный датчик** (О2 датчик) в автомобиле играет ключевую роль в регулировке смеси топлива и воздуха, подаваемой в двигатель. Этот датчик измеряет содержание кислорода в отработавших газах и передает информацию блоку управления двигателем (ECU). Основные аспекты работы и особенности кислородного датчика включают следующие элементы:

Основные функции

Измерение содержания кислорода: Датчик измеряет уровень кислорода в отработавших газах, выходящих из выхлопной системы.

Регулировка смеси топлива и воздуха: Полученные данные используются блоке управления двигателем для регулировки смеси топлива и воздуха, обеспечивая оптимальное сгорание топливной смеси.

Контроль эффективности катализатора: Кислородный датчик также помогает контролировать эффективность работы катализатора, предотвращая загрязнение и повреждение.

Конструкция и компоненты

Сонда датчика: Основной частью кислородного датчика является сонда, которая размещается в выхлопной системе автомобиля.

Электрические провода: Датчик подключается к блоку управления двигателем с помощью электрических проводов и разъемов.

Тепловой элемент: В современных автомобилях кислородные датчики также включают тепловой элемент для быстрого достижения рабочей температуры.

Принцип работы

Кислородный датчик работает следующим образом:

Замкнутая цепь: Датчик создает "замкнутую цепь" с управляющим блоком двигателя (ECU). Он генерирует сигналы, которые блок управления двигателем использует для анализа содержания кислорода в отработавших газах.

Оптимальное сгорание: блок управления двигателем использует данные от датчика для регулировки смеси топлива и воздуха, чтобы обеспечить оптимальное сгорание топлива в цилиндрах двигателя.

Обратная связь: Кроме того, кислородный датчик предоставляет обратную связь о работе катализатора, что помогает блоку управления двигателем оптимизировать его эффективность.

**Иммобилайзер в автомобиле** — это электронная система, предназначенная для защиты от угона путем блокировки работы двигателя. Основная функция иммобилайзера состоит в том, чтобы предотвратить запуск двигателя без правильного ключа или другого авторизованного способа доступа. Вот основные аспекты работы и особенности иммобилайзера:

Основные функции

Блокировка запуска двигателя: Когда автомобиль находится в защищенном режиме, иммобилайзер блокирует работу двигателя, что предотвращает его запуск.

Авторизация доступа: Для снятия защиты иммобилайзера необходимо использовать правильный ключ, имеющий специальный чип или другой авторизованный способ доступа.

Защита от взлома и подделки ключей: Иммобилайзеры часто используют шифрование и специальные коды, чтобы предотвратить подделку ключей или их сканирование.

Конструкция и компоненты

Транспондеры: Иммобилайзеры часто используют транспондеры, которые встроены в ключи или карты доступа. Транспондеры генерируют уникальные коды, которые распознаются системой иммобилайзера.

Контроллеры: Электронные контроллеры управляют иммобилайзером, проверяя аутентичность ключей и принимая решение о разрешении запуска двигателя.

Сенсоры и датчики: Некоторые иммобилайзеры также могут использовать датчики для обнаружения несанкционированного доступа или попыток взлома.

Принцип работы

Идентификация ключа: При вставке ключа в зажигание или при приближении к автомобилю, иммобилайзер считывает уникальный код с транспондера.

Сверка с базой данных: Полученный код сравнивается с заранее сохраненными данными в базе данных иммобилайзера. Если код соответствует, иммобилайзер разрешает запуск двигателя.

Запуск двигателя: При успешной проверке ключа иммобилайзер отключает блокировку, позволяя запустить двигатель.

Обслуживание и проблемы

Проблемы с ключом: Потеря или повреждение ключа может вызвать невозможность запуска двигателя из-за работы иммобилайзера.

Неисправности иммобилайзера: Возможны сбои в работе иммобилайзера, вызванные электрическими проблемами или неисправностью компонентов.

Ремонт и замена: При неисправности иммобилайзера часто требуется профессиональный ремонт или замена компонентов системы.

Важность иммобилайзера

Иммобилайзер является важным элементом системы безопасности автомобиля, который помогает предотвратить угоны. Современные иммобилайзеры обычно сочетаются с системами сигнализации и другими мерами защиты, увеличивая надежность защиты автомобиля от кражи.

**Антиблокировочная система** (ABS) в автомобиле является системой активной безопасности, которая предназначена для предотвращения блокировки колес при резком торможении. Это позволяет водителю сохранять контроль над управлением автомобилем и снижает риск потери управляемости во время экстренного торможения. Вот основные аспекты работы и особенности системы ABS:

Основные функции

Предотвращение блокировки колес: ABS контролирует скорость вращения колес и при необходимости уменьшает тормозное давление на конкретном колесе, чтобы предотвратить его блокировку.

Сохранение управляемости: Предотвращение блокировки колес позволяет водителю сохранить управляемость автомобиля даже при резком торможении или на скользком покрытии.

Сокращение тормозного пути: Благодаря возможности водителя продолжать управление автомобилем даже во время торможения, система ABS помогает сократить тормозной путь и улучшить безопасность.

Конструкция и компоненты

ABS блок: В центре системы ABS находится гидравлический блок, который контролирует давление тормозной жидкости в тормозных механизмах каждого колеса.

Датчики скорости колес: Каждое колесо оборудовано датчиком скорости, который передает информацию о скорости вращения колеса на гидравлический блок ABS.

Электронный блок управления (ECU): ECU обрабатывает данные от датчиков скорости колес и принимает решение о необходимости корректировки тормозного давления.

Актуаторы: Актуаторы, контролируемые гидравлическим блоком, регулируют давление тормозной жидкости для предотвращения блокировки колес.

Принцип работы

Обнаружение начала блокировки: Когда водитель нажимает на педаль тормоза, датчики скорости колес мониторят скорость вращения каждого колеса.

Регулировка тормозного давления: Если система обнаружит, что скорость вращения колеса слишком быстро снижается (что может указывать на блокировку), ECU сигнализирует гидравлическому блоку ABS уменьшить давление тормозной жидкости на этом колесе.

Модулирование тормозного давления: Гидравлический блок ABS быстро модулирует тормозное давление на каждом колесе, что позволяет колесам вращаться и продолжать генерировать сцепление с дорогой.

Восстановление управления: После корректировки давления на колесе, система ABS позволяет водителю продолжить управление автомобилем, обеспечивая максимально эффективное торможение.

Обслуживание и проблемы

Сбои в системе: Неисправности в датчиках скорости колес, гидравлическом блоке или электронной системе могут вызвать сбои в работе системы ABS.

Обслуживание: Регулярная проверка и обслуживание системы ABS, включая замену тормозной жидкости, помогают сохранить ее надежную работу.

Ремонт и замена: При необходимости производится ремонт или замена компонентов системы ABS для восстановления ее функциональности.

Важность системы ABS

Система антиблокировочного тормоза (ABS) является одной из важнейших систем безопасности в современных автомобилях, которая существенно повышает безопасность водителей и пассажиров. Она помогает предотвратить блокировку колес и сохранить управляемость автомобиля в критических ситуациях, что существенно снижает риск аварий и повышает общую безопасность на дороге.

# Особенности автомобильной диагностики

Автомобильная диагностика представляет собой комплекс мероприятий, направленных на определение технического состояния автомобиля, выявление неисправностей и оценку работоспособности его систем и узлов. Этот процесс включает использование различных инструментов и технологий, которые позволяют быстро и точно определить причины неисправностей и разработать стратегии их устранения. Основные особенности автомобильной диагностики включают следующие аспекты:

Комплексность систем автомобиля: Современные автомобили оснащены множеством сложных электронных и механических систем, таких как двигатель, трансмиссия, тормозная система, система управления двигателем, система безопасности и другие. Диагностика каждой из этих систем требует специфических знаний и оборудования.

Использование диагностического оборудования: Для проведения точной диагностики используются специализированные сканеры и тестеры, которые подключаются к бортовым системам автомобиля через диагностический разъем (OBD-II). Эти устройства считывают данные с различных датчиков и контроллеров, позволяя выявлять ошибки и неисправности.

Программное обеспечение для диагностики: Современные диагностические инструменты оснащены программным обеспечением, которое позволяет интерпретировать данные, полученные от автомобиля. Это ПО помогает механикам и техническим специалистам расшифровывать коды ошибок, проводить тестирование систем и анализировать данные в режиме реального времени.

Коды ошибок и их расшифровка: В процессе диагностики важную роль играют коды ошибок, которые хранятся в памяти электронных блоков управления автомобиля. Эти коды указывают на конкретные проблемы в работе систем и узлов, позволяя специалистам быстро определить причину неисправности и приступить к ее устранению.

Технические данные и спецификации: Для эффективной диагностики важно иметь доступ к актуальным техническим данным и спецификациям производителя. Это включает схемы электрических соединений, параметры работы систем, процедуры тестирования и рекомендации по устранению неисправностей.

Методики диагностики: Существует множество методик и подходов к проведению диагностики, включая визуальный осмотр, механические тесты, использование диагностических приборов и программного обеспечения, а также анализ данных и сигналов, поступающих от различных систем автомобиля.

Квалификация специалистов: Успешная диагностика требует высококвалифицированных специалистов, обладающих глубокими знаниями в области автомобильной электроники, механики и программирования. Специалисты должны уметь правильно интерпретировать данные, полученные в процессе диагностики, и применять их для устранения неисправностей.

Автомобильная диагностика является неотъемлемой частью современного автосервиса, обеспечивая надежную и безопасную эксплуатацию транспортных средств. Развитие технологий и появление новых диагностических инструментов позволяет значительно повысить точность и эффективность диагностических процедур, что в свою очередь способствует повышению качества обслуживания автомобилей.

# Особенности функционального программирования

Что использует функциональное программирование:

Функции как первоклассные объекты: Функции в функциональном программировании могут быть переданы как аргументы другим функциям, возвращены как значения из других функций и сохранены в переменных.

Рекурсия: Функциональное программирование использует рекурсию для итерации вместо циклов, что является одним из способов работы с данными.

Функции высшего порядка: Это функции, которые могут принимать другие функции в качестве аргументов или возвращать их в качестве результата.

Чистые функции: Функции, которые не зависят от состояния программы или внешних данных и всегда возвращают одинаковый результат для одних и тех же входных данных. Они не имеют побочных эффектов.

Функциональные структуры данных: В функциональном программировании используются специфические структуры данных, такие как списки, карты, кортежи и деревья.

Лямбда-исчисление: Функциональное программирование часто основано на математическом лямбда-исчислении, которое формализует функции как преобразования данных.

Функции как элементы композиции: Композиция функций (создание новых функций путем объединения существующих) является важной техникой в функциональном программировании.

Эти особенности делают функциональное программирование мощным инструментом для работы с данными и обработки информации в чистом и предсказуемом стиле.

# Выбор языка

Перед началом написания курсовой работы, я имел возможность выбрать один из трёх языков программирования: Scala, Haskell, Prolog. В качестве языка программирования не был выбран Prolog, так как он является логическим языком программирования, и программа в нем строится от предикатов, что не подходит под мою тему. Язык Haskell же менее удобен при написании графического интерфейса, нежели Scala.

Scala же, в свою очередь имеет несколько плюсов по сравнению с этими двумя языками:

Выразительность: Scala предлагает богатый набор функциональных конструкций, таких как функции высшего порядка, неизменяемые структуры данных, рекурсия и т.д. Это позволяет студентам глубже понять особенности функционального программирования.

Широкая поддержка: Scala имеет активное сообщество и обширную документацию, что упрощает изучение и разработку проектов на этом языке.

Использование в индустрии: Scala широко используется в индустрии, особенно в крупных корпоративных приложениях и Big Data решениях, что делает знание этого языка полезным для будущей карьеры.

Интеграция с Java: Scala работает на JVM и без проблем интегрируется с существующими Java-библиотеками и инструментами, что упрощает использование его в реальных проектах.

Именно интеграция c Java стала ключевым плюсом при выборе языка.

# Описание переменных, функций

**Переменные**

ThrottleValve: Тип – булевая, отвечает за то, есть ли свзять с дроссельной заслонкой  
ThrottleValvePos: Тип – целочисленная, отвечает за положение дроссельной заслонки, используется при адаптации положения  
ABS: Тип – строковая, отвечает за связь с блоком АБС и его настройки   
CrankshaftPS: Тип – булевая, отвечает за связь с датчиком положения коленчатого вала  
CamshaftPS: Тип- булевая, отвечает за связь с датчиком положения распределительных валов  
Immo: Тип – булевая, отвечает за связь с блоком иммобилайзера автомобиля  
OxygenSensor: Тип – Булевая, отвечает за связь с лямбда-зондом (кислородным датчиком)  
Attempts: Тип – целочисленная, отвечает за количество попыток, которые имеет пользователь на устранение неисправности  
ErrorInfo: Тип – строковая, отвечает за справочную информацию по ошибке автомобиля  
ErrorCode: Тип – строковая, отвечает за код ошибки  
Message: Тип – строковая, отвечает за справочное сообщение пользователю

**Функции**

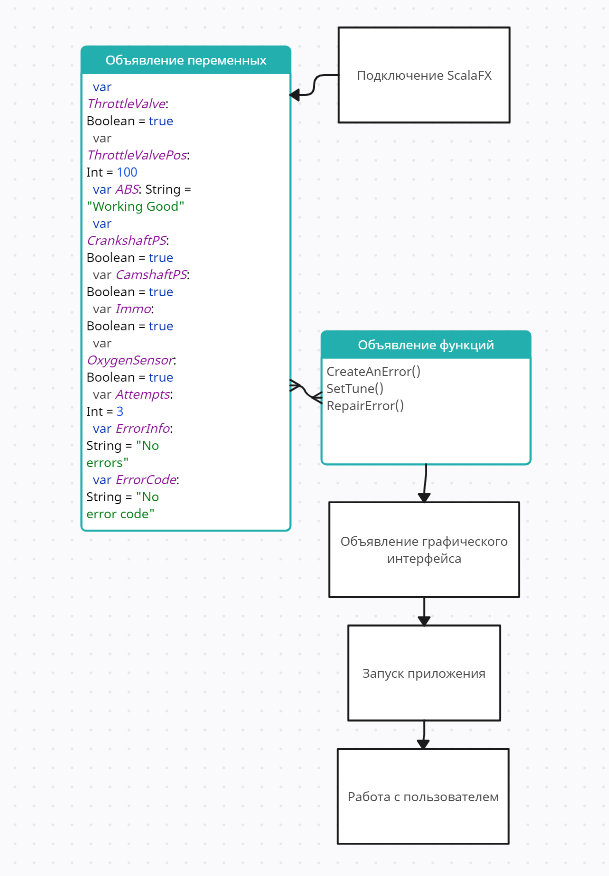
CreateAnError: Функция генерации ошибки в автомобиле, влияет на переменные датчиков, изменяя их

SetTune(setting: String): Функция настройки параметров датчиков, влияет на переменные, отвечающие за позицию или статус автомобиля, принимает аргументом параметр, который необходимо настроить

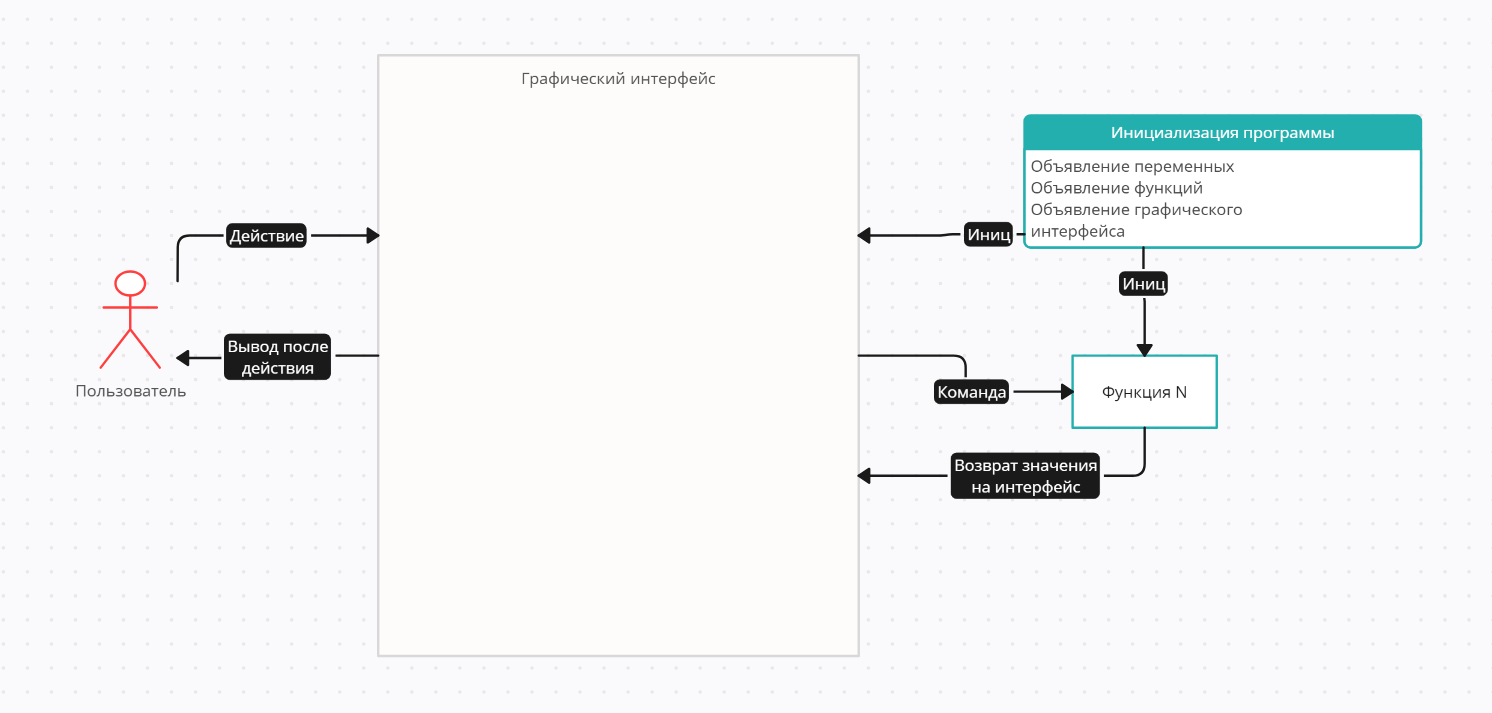
RepairError(err1: String): Функция исправления неполадок датчиков автомобиля. Принимает на вход параметр который необходимо исправить

App.main() Функция открытия графического интерфейса программы.

# Диаграмма работы программы



# Диаграмма работы программы с пользователем



# Используемая библиотека для графического интерфейса

ScalaFX - это библиотека для создания пользовательских интерфейсов (GUI) на Scala с использованием JavaFX. ScalaFX позволяет создавать высококачественные GUI-приложения на Scala, используя мощные функциональные и декларативные возможности этого языка.

Что умеет данная библиотека:

Интеграция с JavaFX: ScalaFX предоставляет Scala-стильные обертки и DSL (Domain Specific Language) для JavaFX API. JavaFX сам по себе представляет собой современную и мощную библиотеку для создания GUI приложений.

Функциональный подход: ScalaFX позволяет использовать функциональные конструкции Scala для создания и управления компонентами GUI. Это включает в себя использование лямбда-выражений для обработки событий, функции высшего порядка для композиции компонентов и т.д.

Декларативный стиль: ScalaFX поддерживает декларативное описание интерфейса пользователя с использованием Scala DSL. Это упрощает создание и поддержку сложных GUI.

Интеграция с Java и JVM: ScalaFX использует JavaFX, который полностью интегрирован с Java и работает на JVM. Это означает, что вы можете легко использовать существующие JavaFX библиотеки и инструменты.

# Используемая литература

<https://www.scalafx.org/docs/home/>

<https://habr.com/ru/companies/macloud/articles/552984/>

<https://koleso.ru/articles/datchiki-v-avtomobile-kakie-byivayut-i-za-chto-otvechayut/>

<https://docs.scala-lang.org>

# Приложение

import scala.util.Random  
import scala.io.StdIn.readLine  
import scalafx.Includes.\*  
import scalafx.application.JFXApp3  
import scalafx.scene.Scene  
import scalafx.scene.control.{Alert, Button, Label, TextInputDialog}  
import scalafx.event.ActionEvent  
import scalafx.scene.image.\*  
import scalafx.scene.control.Alert.AlertType  
import scala.io.Source  
  
import java.io.File  
  
object main {  
 *// Глобальные переменные* val *fileContents*: String = Source.*fromFile*("ErrorHelper.txt").getLines.mkString  
 var *ThrottleValve*: Boolean = true  
 var *ThrottleValvePos*: Int = 100  
 var *ABS*: String = "Working Good"  
 var *CrankshaftPS*: Boolean = true  
 var *CamshaftPS*: Boolean = true  
 var *Immo*: Boolean = true  
 var *OxygenSensor*: Boolean = true  
 var *Attempts*: Int = 3  
 var *ErrorInfo*: String = "No errors"  
 var *ErrorCode*: String = "No error code"  
 var *Message*: String = "No message"  
  
 def CreateAnError(): Unit = {  
 val random = new Random()  
 val action = random.nextInt(5)  
  
 action match {  
 case 0 => *// Неисправность ДПКВ  
 CrankshaftPS* = false  
 *ErrorInfo* = "Can't start the engine"  
 *ErrorCode* = "P0001"  
  
 case 1 => *// Неисправность Кислородника  
 OxygenSensor* = false  
 *ErrorInfo* = "Too much fuel injection"  
 *ErrorCode* = "P0002"  
  
 case 2 => *// Неисправность ДроссЗ* val rnd = new Random()  
 val act = rnd.nextInt(3)  
 act match {  
 case 0 => *ErrorInfo* = "Can't start the engine"; *ThrottleValve* = false; *ErrorCode* = "P0003"  
 case 1 => *ErrorInfo* = "Too much fuel injection"; *ThrottleValve* = false;  
 case 2 => *ErrorInfo* = "Unstable idling"; *ThrottleValvePos* = 5;  
 }  
  
 case 3 => *// Неисправность ДПРВ  
 CamshaftPS* = false  
 *ErrorInfo* = "Can't start the engine"  
 *ErrorCode* = "P0004"  
  
 case 4 => *// Неисправность Иммо  
 Immo* = false  
 *ErrorInfo* = "Can't start the engine"  
  
 case 5 => *// Ошибка по АБС* val rnd = new Random()  
 val act = rnd.nextInt(2)  
 act match {  
 case 0 => *ABS* = "Error on ABS sensor"  
 *ErrorInfo* = "ABS sensor out of range"  
 *ErrorCode* = "P0005(1)"  
 case 1 => *ABS* = "Not Learned"  
 *ErrorInfo* = "ABS was not learned"  
 *ErrorCode* = "P0005(2)"  
 }  
  
 case \_ => *println*("ajajaja") *// В случае, если случайное число не попадает в диапазон 0-4* }  
 }  
  
 def SetTune(setting:String): Unit = {  
 setting match{  
 case "throttle" => *ThrottleValvePos* = 100  
 *ErrorInfo* = "No errors"  
 *println*("Throttle position adjusted!")  
 case "abs" => *ABS* = "Working Good"  
 *ErrorInfo* = "No errors"  
 *println*("It works")  
 }  
 }  
  
 def repairError(err1:String): Unit = {  
 *//while (Attempts != 0) {* if (*Attempts* == 0) {  
 *println*("Out of attempts")  
 *Message* = "Out of attempts"  
 return  
 }  
 val err = err1  
 err match {  
 case "crankshaft" => if (*CrankshaftPS* == true) {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("The error was successfully repaired")  
 *ErrorInfo* = "No errors"  
 }  
  
 case "camshaft" => if (*CamshaftPS* == true) {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("The error was successfully repaired")  
 *ErrorInfo* = "No errors"  
 }  
  
 case "throttle" => if (*ThrottleValve* == true) {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("The error was successfully repaired")  
 *ErrorInfo* = "No errors"  
 }  
  
 case "immo" => if (*Immo* == true) {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("The error was successfully repaired")  
 *ErrorInfo* = "No errors"  
 }  
  
 case "oxygen" => if (*OxygenSensor* == true) {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("The error was successfully repaired")  
 *ErrorInfo* = "No errors"  
 }  
  
 case "abs" =>if (*ABS* == "Working Good") {  
 *println*("Wrong error, - 1 attempt");  
 *Message* = "Wrong error, - 1 attempt"  
 *Attempts* = *Attempts* - 1  
 }  
 else {  
 *println*("ABS was replaced, need to learn")  
 *ABS* = "Not Learned"  
 *ErrorInfo* = "ABS was not learned"  
 *ErrorCode* = "P0005(2)"  
 }  
  
 case \_ => *println*("Input error, try again")  
 *Message* = "Input error, try again"  
 }  
 }  
 *//}  
  
// Графический интерфейс* object app extends JFXApp3 {  
 override def start(): Unit = {  
 *stage* = new JFXApp3.PrimaryStage {  
 title = "Car repairer"  
 scene = new Scene(800, 400) {  
  
 val *img* = new Image("file:crash.jpg")  
 val *view* = new ImageView(*img*)  
 *view*.layoutX = 200  
  
 val *imgRep* = new Image("file:repaired.jpg")  
 val *viewRep* = new ImageView(*imgRep*)  
  
 val *button* = new Button("Wat Do?")  
 *button*.layoutX = 20  
 *button*.layoutY = 20  
 *button*.onAction = \_ => {  
 val dialog = new TextInputDialog(defaultValue = "Input error") {  
 initOwner(*stage*)  
 title = "Text Input Dialog"  
 headerText = "Error inputer"  
 contentText = "Please enter your error:"  
 }  
  
 val result = dialog.showAndWait()  
  
 result match {  
 case Some(err) => *repairError*(err)  
 if (*Attempts* == 0){content = *view*}  
 if (*ErrorInfo* != "No errors"){new Alert(AlertType.Information) { headerText = *Message*}.showAndWait()}  
 else{new Alert(AlertType.Information) { headerText = "Success!"}.showAndWait(); content = List(*viewRep*)}  
 case None => *println*("Dialog was canceled.")  
 }  
  
 }  
  
 val *button1* = new Button("Start Engine")  
 *button1*.layoutX = 100  
 *button1*.layoutY = 60  
 *button1*.onAction = \_ => {  
 new Alert(AlertType.Information) { headerText = *ErrorInfo*}.showAndWait()  
 }  
  
 val *ErrorCodeButton* = new Button("Scan code")  
 *ErrorCodeButton*.layoutX = 100  
 *ErrorCodeButton*.layoutY = 20  
 *ErrorCodeButton*.onAction = \_ =>{new Alert(AlertType.Information) { headerText = *ErrorCode*}.showAndWait()}  
  
 val *SetTuneButton* = new Button("Wat do?")  
 *SetTuneButton*.layoutX = 20  
 *SetTuneButton*.layoutY = 20  
 *SetTuneButton*.onAction = \_ => {  
 val dialog = new TextInputDialog(defaultValue = "Input tune setting") {  
 initOwner(*stage*)  
 title = "Text Input Dialog"  
 headerText = "setting inputer"  
 contentText = "Please enter your setting:"  
 }  
  
 val result = dialog.showAndWait()  
  
 result match {  
 case Some(err) => *SetTune*(err)  
 if (*Attempts* == 0){content = *view*}  
 if (*ErrorInfo* != "No errors"){new Alert(AlertType.Information) { headerText = *Message*}.showAndWait()}  
 else{new Alert(AlertType.Information) { headerText = "Success!"}.showAndWait(); content = List(*viewRep*)}  
 case None => *println*("Dialog was canceled.")  
 }  
 }  
  
 val *ReturnButton* = new Button("Return Button")  
 *ReturnButton*.layoutX = 20  
 *ReturnButton*.layoutY = 200  
 *ReturnButton*.onAction = \_ => {  
 content = List(*button*, *button1*, *ErrorCodeButton*, *TuneButton*, *Helper*, *view*)  
 }  
  
 val *TuneButton* = new Button("Tune-up")  
 *TuneButton*.layoutX = 20  
 *TuneButton*.layoutY = 60  
 *TuneButton*.onAction = \_ =>{  
 content = List(*ErrorCodeButton*, *SetTuneButton*, *ReturnButton*, *Helper*, *view*)  
 }  
  
 val *Helper* = new Button("Help Info")  
 *Helper*.layoutX = 100  
 *Helper*.layoutY = 350  
 *Helper*.onAction = \_ => {  
 new Alert(AlertType.Information) { contentText = *fileContents*}.showAndWait()  
 }  
  
 content = List(*button*, *button1*, *ErrorCodeButton*, *TuneButton*, *Helper*, *view*)  
 }  
 }  
 }  
}  
  
  
 def main(args: Array[String]): Unit = {  
 *CreateAnError*()  
 app.main(args)  
 }  
}