**Министерство науки и высшего образования РФ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Институт магистратуры

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Специальность 09.04.01 Информатика и вычислительная техника

Специализация Проектирование интеллектуальных систем

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

на тему:

Разработка и реализация алгоритма вычисления уровня топлива в баках сложной конфигурации

|  |  |
| --- | --- |
| **Студент** | Притчин Иван Сергеевич |
| **Зав. кафедрой** | канд. техн. наук, проф. Поляков В.М. |
| **Руководитель** | к.ф.-м.н., Зуев С.В. |
| **Консультант** |  |

**К защите допустить**

**Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/**Поляков В.М.**/**

**«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** 2021 г.

**Белгород 2021г.**

**Министерство науки и высшего образования РФ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Институт магистратуры

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Направление подготовки 09.04.01 Информатика и вычислительная техника

Направленность (профиль, специализация) образовательной программы: Проектирование интеллектуальных систем.

|  |  |
| --- | --- |
|  | Утверждаю:  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г. |

**ЗАДАНИЕ**

на выпускную квалификационную работу студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Притчина Ивана Сергеевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество)

1. Вид выпускной квалификационной работы (ВКР) \_\_\_\_\_магистерская диссертация\_\_\_\_\_\_\_

(бакалаврская работа, дипломный проект, дипломная работа, магистерская диссертация)

|  |
| --- |
| 2. Тема ВКР: Разработка и реализация алгоритма вычисления уровня топлива в баках |
| сложной конфигурации |
|  |

|  |
| --- |
| утверждено приказом по университету от «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021\_г. № |

|  |
| --- |
| 3. Срок сдачи студентом законченной ВКР |
|  |

|  |
| --- |
| 4. Исходные данные: данные компании «Экспертком», методы обработки данных |
|  |
|  |
|  |
|  |
|  |

5. Содержание ВКР: (перечень подлежащих разработке разделов) 1. Анализ предметной области. 2. Описание подхода для вычисления количества топлива в баках сложной конфигурации по нескольким датчикам уровня топлива. 3. Реализация алгоритма разбиения датчиков уровня топлива и построения зависимостей. 4. Написание пояснительной записки к ВКР.

|  |
| --- |
| 6. Перечень графического материала: |
|  |

56 рисунков;

|  |
| --- |
| 12 таблиц; |
| 10 слайдов презентации: |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Консультанты по работе с указанием относящихся к ним разделов

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Задание выдал  (подпись, дата) | Задание принял  (подпись, дата) |
|  |  |  |  | |

Дата выдачи задания « » 2021 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Задание принял к исполнению

(подпись студента)

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов работы | Срок выполнения  этапов работы | Примечание |
| 1 | Изучение предметной области, сопутствующих проблем при определении уровня топлива в баках сложной конфигурации |  | Выполнено |
| 2 | Постановка задачи и формирование требований к разрабатываемым программным средствам. |  | Выполнено |
| 3 | Проектирование консольного приложения. |  | Выполнено |
| 4 | Разработка программной системы; её тестирование и отладка; написание пояснительной записки к ВКР |  | Выполнено |
| 5 | Подготовка презентационного материала и защитной речи |  | Выполнено |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Притчин И.С.

(подпись) (Ф.И.О.)

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.ф.-м.н., Зуев С.В.

(подпись) (Ф.И.О.)

Содержание

[Введение 7](#_Toc73644153)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc73644154)

[Спутниковый мониторинг транспорта (СМТ) 9](#_Toc73644155)

[Контроллеры и трекеры 10](#_Toc73644156)

[Датчики уровня топлива 13](#_Toc73644157)

[Программное обеспечение 15](#_Toc73644158)

[Проблема вычисления уровня топлива 19](#_Toc73644159)

[Цель и задачи выпускной квалификационной работы 23](#_Toc73644160)

[2. ОПИСАНИЕ ПОДХОДА ДЛЯ ВЫЧИСЛЕНИЯ КОЛИЧЕСТВА ТОПЛИВА В БАКАХ СЛОЖНОЙ КОНФИГУРАЦИИ ПО НЕСКОЛЬКИМ ДАТЧИКАМ УРОВНЯ ТОПЛИВА 24](#_Toc73644161)

[Баки – не сообщающиеся сосуды 24](#_Toc73644162)

[Баки – сообщающиеся сосуды 30](#_Toc73644163)

[Требования к программному обеспечению 35](#_Toc73644164)

[Выбор инструментов для разработки 38](#_Toc73644165)

[Выводы 39](#_Toc73644166)

[3. РЕАЛИЗАЦИЯ АЛГОРИТМА ДЛЯ РАЗБИЕНИЯ ДАТЧИКОВ УРОВНЯ ТОПЛИВА И ПОСТРОЕНИЯ ЗАВИСИМОСТЕЙ 40](#_Toc73644167)

[Описание алгоритма 40](#_Toc73644168)

[Тестирование программного обеспечения 53](#_Toc73644169)

[Установка программного обеспечения 58](#_Toc73644170)

[Вывод 70](#_Toc73644171)

[Заключение 71](#_Toc73644172)

[Список литературы 72](#_Toc73644173)

## Введение

Заметный рост количества транспортных единиц, аппаратных и программных возможностей позволили лучше контролировать ресурсы организаций, использующих автотранспорт. Комплексы наблюдения за техникой состоят из трекера, подключаемого к нему оборудования, аппаратного обеспечения, программного обеспечения, которые участвуют в процессе обработки данных.

Появление отечественных и зарубежных систем спутникового мониторинга позволяет в довольно быстрые сроки осуществлять контроль в данном вопросе. Клиенты, использующие приложения, получают удобный интерфейс для решения своих задач, которого в большинстве случаев хватает, чтобы покрыть свои нужды.

Организации, осуществляющие контроль над количеством топлива в баках, столкнулись с необходимостью более точного определения данного показателя, который может иметь погрешность по ряду причин: наклон техники на местности (1), колебания при работе в тяжелых условиях. Логичным решением было добавление дополнительных датчиков уровня топлива в бак, что позволяло по получаемым данным вычислять среднее. Это привело к тому, что внешние факторы стали оказывать меньшее влияние на качество получаемой информации.

Однако проблема не была решена для баков сложной конфигурации. Сервер получает данные с некоторого множества датчиков и подставляет их в формулу, задаваемую пользователем системы спутникового мониторинга. Несмотря на ожидаемую, с нулевого приближения, простоту вычислений, наблюдаются определенные сложности в создании формулы для баков сложной конфигурации, по которым будет производиться расчёт.

В рамках магистерской диссертации рассматривается вопрос разработки и реализации алгоритма вычисления количества топлива в баках сложной конфигурации с использованием нескольких датчиков уровня топлива, а также механизм адаптации данного подхода для систем спутникового мониторинга.

В первом разделе даны основные определения, которые касаются процессов установки, настройки и конфигурирования оборудования для контроля уровня топлива в транспортном средстве; описаны вычислительные проблемы при определении уровня топлива. В конце раздела поставлена цель выпускной квалификационной работы и сформулированы задачи.

Во втором разделе описан предлагаемый алгоритм (без детализации) для баков – не сообщающихся сосудов и баков – сообщающихся сосудов, а также сформулированы требования к программному обеспечению.

В третьем разделе представлена реализация алгоритма, приведены результаты тестирования, а также сведения по установке ПО, его эксплуатации, вкупе с системой спутникового мониторинга Wialon.

В заключении приведены выводы о проделанной работе.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Спутниковый мониторинг транспорта (СМТ)

К появлению систем спутникового мониторинга привела необходимость отслеживания единиц в реальном времени для получения координат. Со временем, потребности клиентов возрастали: возникла потребность получения и других данных. Трекер стал представлять собой более сложное устройство, с возможностью подключения множества датчиков, которые позволили получать данные более широкого спектра.

Перед современными системами спутникового мониторинга поставлен ряд задач, среди которых:

* Задача мониторинга (получение информации о транспортном средстве: координат, скорости движения, расходе топлива, подсчёте моточасов).
* Контроль над графиком движения (учет перемещения подвижных единиц, например, при доставке грузов).
* Сбор статистики и анализ (оценка результатов работы техники за промежуток времени в вопросе затраты ресурсов).
* Обеспечение безопасности.
* Идентификация персонала, работающего с транспортным средством.

Комплексы наблюдения за техникой состоят из следующих составляющих:

* Транспортное средство, оборудованное GPS (Global Positioning System) или ГЛОНАСС (Глоба́льная навигацио́нная спу́тниковая систе́ма) контроллером или трекером, который получает данные со спутников и передаёт их на сервер при помощи GSM (Global System for Mobile Communications) или CDMA (Code Division Multiple Access). Можно встретить варианты, работающие за счёт спутниковой или УКВ связи в случае плохого GSM-покрытия.
* Множество датчиков, связываемых с трекером, для передачи дополнительных данных.
* Сервер с программным обеспечением для приёма, хранения, обработки и анализа получаемой информации.
* Компьютер диспетчера, производящий наблюдение за транспортом.

Рассмотрим составляющие более подробно.

## Контроллеры и трекеры

Большинство видов оборудования данного класса имеют схожие функциональные возможности:

* Вычисление собственного местоположения и скорости движения на основании сигналов спутников GPS или ГЛОНАСС.
* Подключение датчиков через аналоговые или цифровые входы.
* Считывание данных с бортового оборудования, имеющего последовательный порт (интерфейс стандарта RS-232) или более специализированный интерфейс CAN (Controller Area Network). Контроль напряжения автомобильного и встроенного в трекер аккумуляторов.
* Хранение данных во внутренней энергонезависимой памяти устройства на случай отсутствия возможности их передачи на сервер.
* Передача данных на сервер согласно расписанию для дальнейшей обработки.

Рассмотрим принципы функционирования трекера на примере ГЛОНАСС-трекера SMART S-2433. Устройство осуществляет контроль:

* состояния подключенных датчиков
* напряжения основного источника питания и встроенного аккумулятора
* уровня сигнала GSM-модема
* работоспособности и показаний навигационного датчика (ГЛОНАСС/GPS) и т.д.

Устройство трекера представлено на рисунках 1.1-1.2 и представляет собой (2):

1. Передняя крышка корпуса
2. Крепежное отверстие
3. Системный светодиодный индикатор
4. Светодиодный индикатор GSM
5. Светодиодный индикатор ГЛОНАСС/GPS
6. 14-контактный разъем
7. Разъем MiniUSB
8. Выталкиватель держателя SIM-карты
9. Держатель SIM-карты 1 (внешний)
10. Задняя крышка корпуса;
11. Крепежный винт задней крышки корпуса – 4 шт.

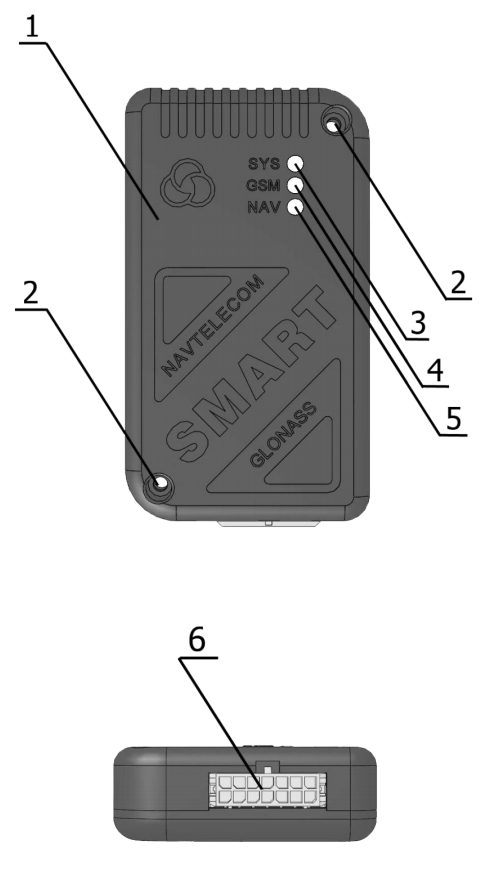


Рисунок 1.1 — ГЛОНАСС-трекер СМАРТ S-2433

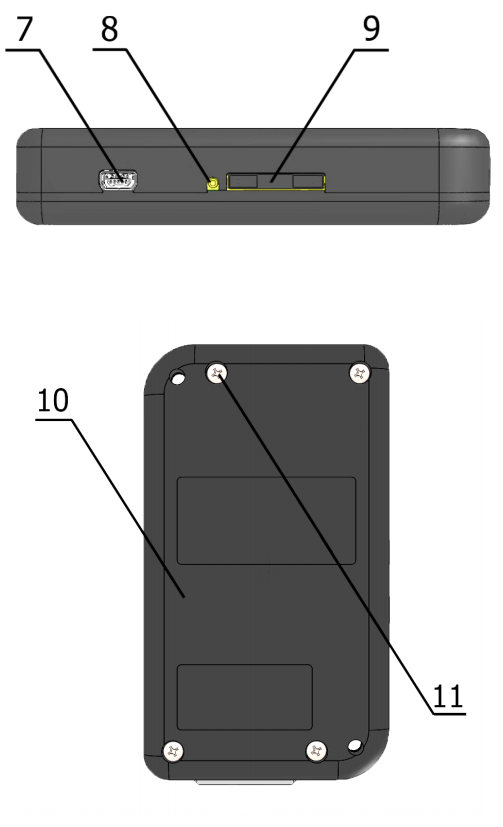


Рисунок 1.2 — ГЛОНАСС-трекер СМАРТ S-2433

При подключении к датчику через компьютер посредством USB или за счёт основного питания происходит включение устройства. При отключении питания устройство работает от встроенной аккумуляторной батареи. При разряде встроенной АКБ до 3B происходит выключение устройства.

В процессе работы трекер формирует и отправляет сообщения, которые создаются при возникновении события, некоторые из них:

* Изменение курса
* Срабатывание таймера в движении или при стоянке
* Срабатывание датчика входной линии
* Изменение значения аналогового или цифрового датчика

Сообщения записываются в память и отправляются на сервер последовательно вместе с порядковым номером и кодом, определяющим причину формирования. Исключением в порядке отправки будут сообщения, сформированные по «тревожным» событиям (например, срабатывание датчиков удара).

В зависимости от настроек прибора, происходит сохранение информации в энергонезависимую память. При невозможности отправить сообщения, данные будут выгружены по устранению проблемы.

При передаче информации трекер ждёт от сервера ответ о подтверждении получения. Если трекер не получил информацию об успешной операции он пробует отправить данный пакет снова.

Приведенный алгоритм, который заложен в протоколе передачи данных, позволяет обеспечить надежность отправки сообщений на сервер. Стоит так же отметить, что производители оборудования выпускают конфигураторы, которые дают возможность, относительно просто, настроить работу трекера под свои задачи.

## Датчики уровня топлива

Ранее отмечалось, что операторы систем спутникового мониторинга хотят получать информацию не только о местоположении объекта, но и иные данные, которые могут быть получены при помощи внешних датчиков. Наиболее часто можно встретить датчики уровня топлива (ДУТ). ДУТы могут монтироваться как на стационарные (резервуары с горючим, автозаправочные станции), так и на подвижные объекты. Интеграция данного оборудования вкупе с программным обеспечением позволяет в режиме реального времени получать информацию о:

* фактическом объеме топлива в баке на текущий момент;
* расходе топлива за определенный промежуток времени;
* усредненном расходе топлива (на 100 км);
* наличии фактов сливов и заправок и их объемы.

По конструктивным особенностям выделяют следующие виды:

* потенциометрические (поплавковые);
* ёмкостные;
* ультразвуковые.

Одним из часто используемых датчиков является Omnicomm LLS5 (3) (рисунок 1.3).



Рисунок 1.3 — Датчик Omnicomm LLS5

Датчик монтируется максимально приближенно к геометрическому центру бака (рисунок 1.4) и подключается к трекеру:

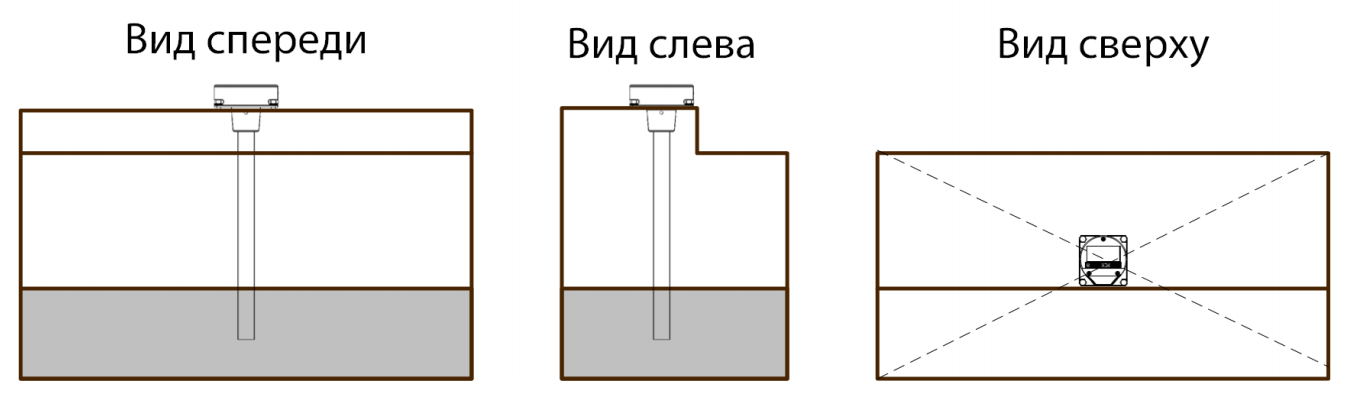


Рисунок 1.4 — Монтаж одного ДУТ

Установка нескольких датчиков уровня топлива позволяет значительно уменьшить зависимость уровня топлива от угла наклона транспортного средства. Например, в случае монтажа двух ДУТ они могут быть размещены следующим образом (рисунок 1.5):

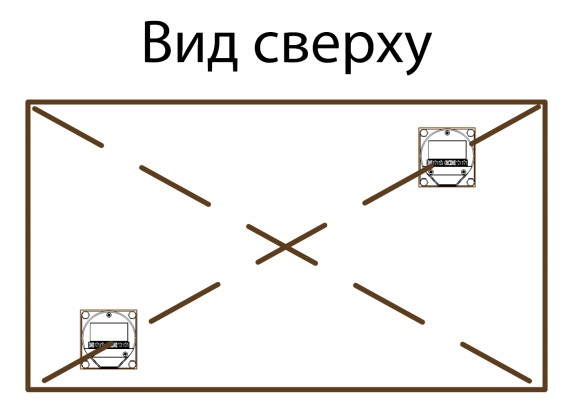


Рисунок 1.5 — Установка двух ДУТов в бак

Приём с использованием нескольких датчиков также часто используется при работе с баками сложной конфигурации. Подробнее об этом пойдёт речь во второй части работы.

## Программное обеспечение

Трекеры отсылают данные на сервер, где они обрабатываются программным обеспечением. Информация предоставляется пользователю в удобном виде для решения практических задач (рисунок 1.6):

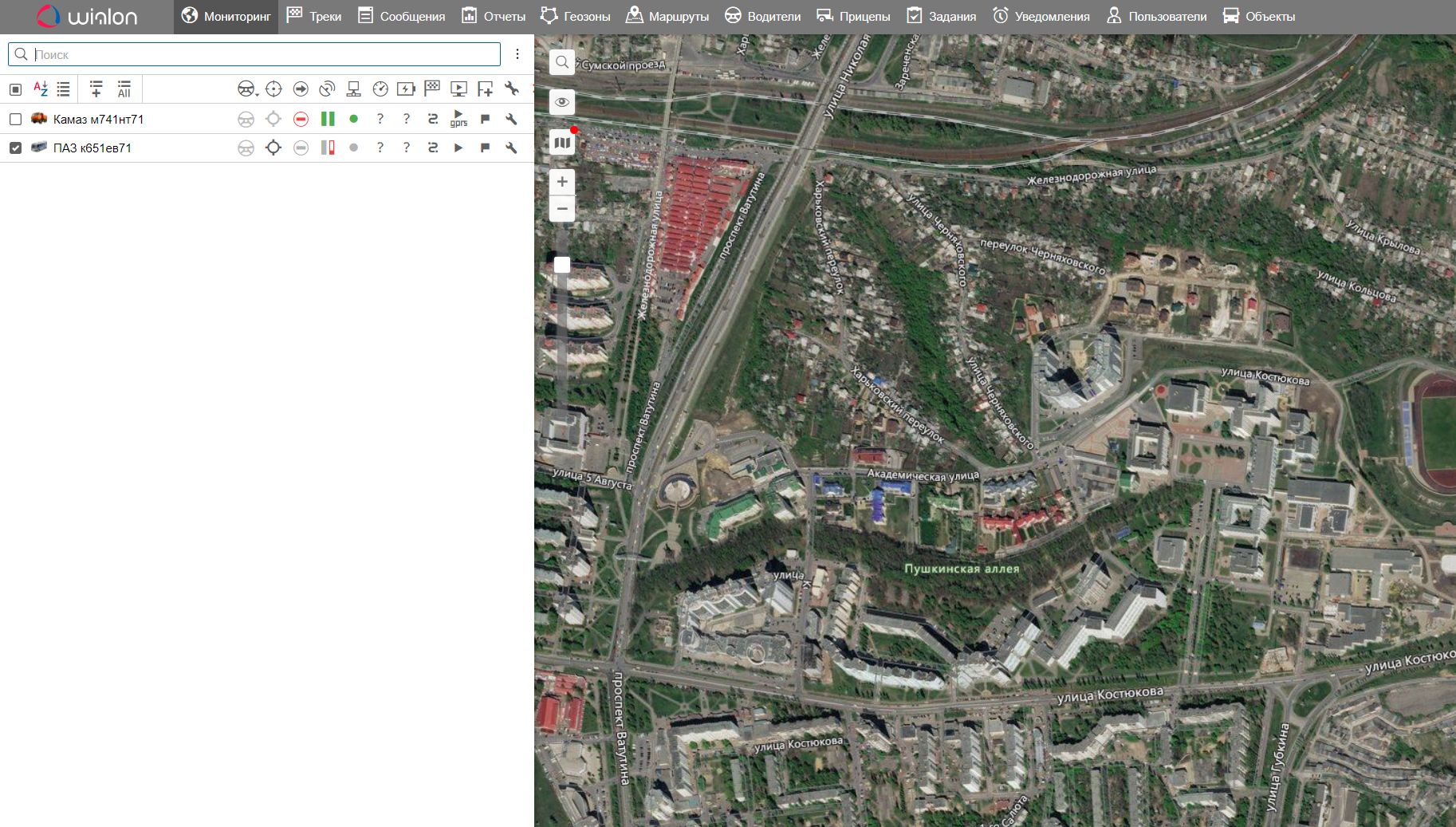


Рисунок 1.6 — Система спутникового мониторинга Wialon

Одним из центральных определений является объект. С физической точки зрения это некоторая точка, которая отслеживается трекером. В контексте выпускной квалификационной работы нас будет интересовать автотранспорт с ДУТ. Данные единицы характеризуются массой набором датчиков и их настроек. Говоря более строго, датчик (с точки зрения систем спутникового мониторинга) представляет собой некоторую функцию или суперпозицию функций. В качестве аргумента выступают данные, получаемые с трекера или значения с других датчиков, создаваемых пользователем. В качестве значения, выдаваемое датчиком, выступает некоторое число, реже - логическое значение.

Рассмотрим механизм работы на примере топливного бака с двумя ДУТ. Единица, которую мы будем разбирать далее, в необходимые моменты времени отсылает сообщения. Например, некоторые из параметров (таблица 1.1):

Таблица 1.1 — Фрагмент сообщения

|  |  |
| --- | --- |
| Наименование параметра | Значение |
| Время | 2021-04-01 09:40:37 |
| Скорость | 0 |
| Координаты | 50.7258678, 37.5213918 |
| Количество спутников | 4 |
| Высота | 187.3 |
| pwr\_ext | 12.4 |
| clls1 | 935 |
| clls2 | 820 |

В данном случае, трекер отсылает информацию о напряжении сети автомобиля и информацию с двух датчиков уровня топлива. Информация, выдаваемая датчиком уровня топлива, представлена не в виде какого-то количества топлива в литрах, а имеет некоторый характер, нуждающийся в дополнительной обработке.

После монтажа ДУТов в бак происходит так называемый процесс тарировки, который заключается в постепенном (с некоторым шагом) заполнении бака топливом и снятием показаний с датчика при заданном количестве литров. Данные заносятся в тарировочную таблицу, которая содержит некоторое множество точек, задающих функцию:

где – количество топлива, – показания с датчика уровня топлива. Значения между точками получаются посредством линейной интерполяции. В случае, если точка лежит за пределами – применяется алгоритм линейной экстраполяции.

Очевидно, что в случае с несколькими независимыми баками суммарный объем топлива будет вычисляться по формуле:

В системе спутникового мониторинга для объекта настраиваются нужные датчики. В нашем случае это:

* два ДУТа на , ;
* два датчика для преобразования показаний с ДУТов в литры по тарировочным таблицам
* датчик для сохранения информации о суммарном количестве топлива
* датчик «Топливо в баке», который заносится с типом «Датчик уровня топлива» и используется в построении отчетов[[1]](#footnote-1).

Заведенные датчики представлены на рисунке 1.7.

Механизм работы системы с точки зрения передачи и обработки данных представлен на рисунке 1.8.

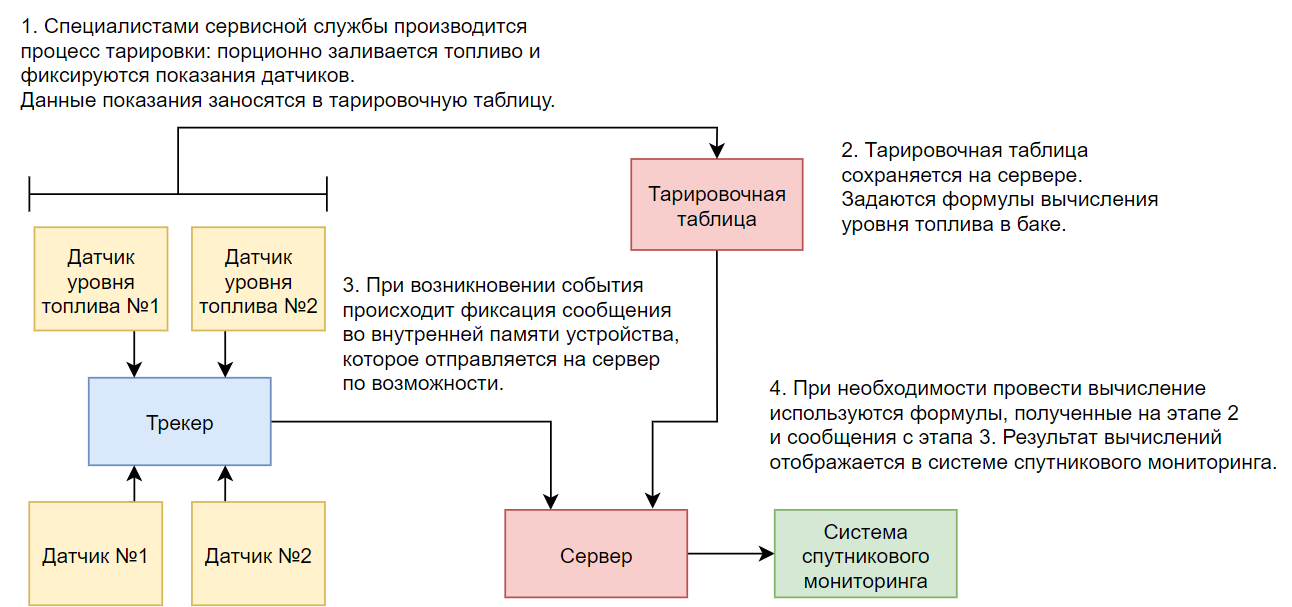


Рисунок 1.7 — Механизм работы системы

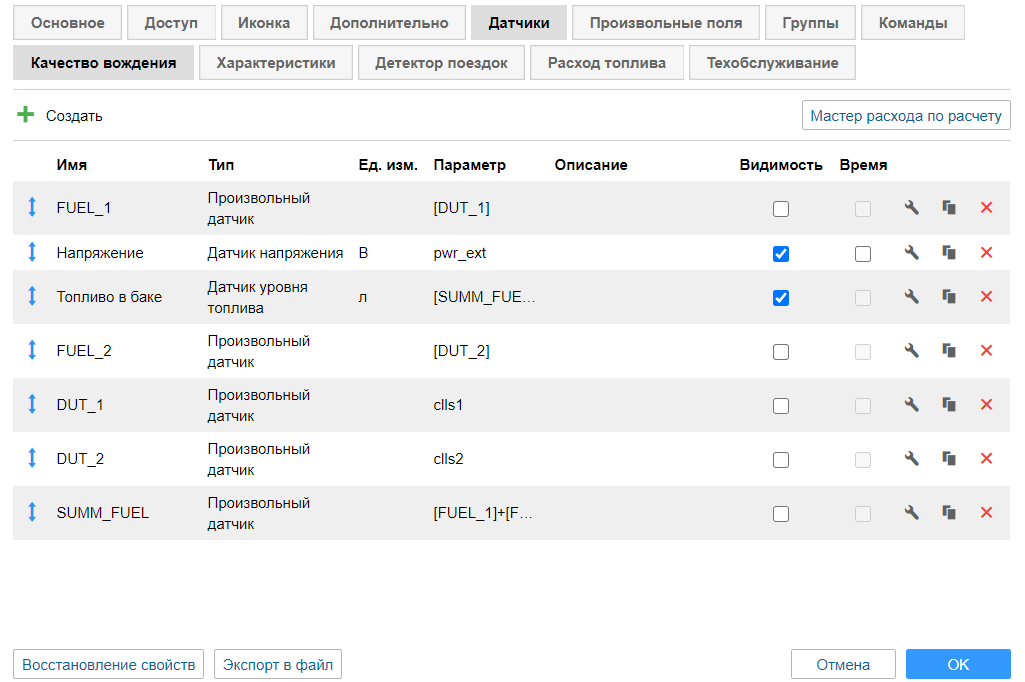


Рисунок 1.8 — Заведенные датчики в системе спутникового мониторинга Wialon

## Проблема вычисления уровня топлива

Большинство проанализированных работ направлены на создание тех или иных датчиков уровня топлива (4), (5), (6), (7). Некоторые работы направлены на специализированную технику, такие как летательные аппараты (самолёты, ракеты (8)) или специфический вид топлива (9). Данная же работа фокусируется не на аппаратных моментах, а на системах, которые обрабатывают показания с датчиков уровня топлива.

Рассмотрим проблему, которая возникает в баках сложной конфигурации. Ранее упоминалось, что для повышения точности измерений уровня топлива используется несколько ДУТов в одном баке. Они позволяют уменьшить влияние колебаний топлива в ходе работы техники. Также можно снять более точные показания в случае наклона единицы. В случае с прямоугольным баком мы могли бы наблюдать следующую картину (рисунок 1.9).

Реальное количество топлива в баке может быть определено как первым ДУТом, так и вторым. В данном примере не существует таких зон, в которой какой-то из ДУТов не может зафиксировать тот или иной объем топлива в баке.

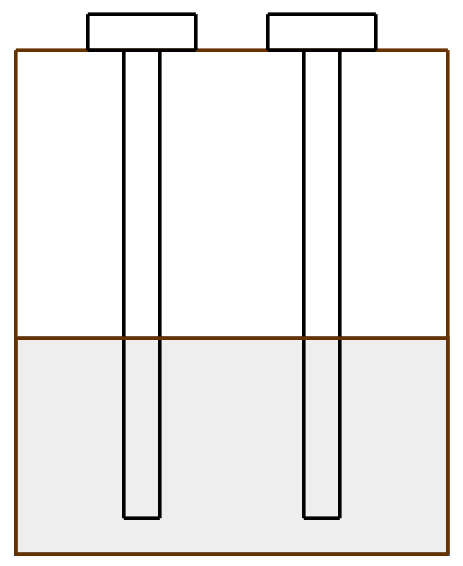


Рисунок 1.9 — Два ДУТа в одном баке

Пример тарировочных таблиц:

Таблица 1.2 — Тарировочная таблица к рисунку 1.9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 0 | 30 |  | 0 | 30 |
| 10 | 491 |  | 10 | 409 |
| 20 | 614 |  | 20 | 982 |
| 30 | 1309 |  | 30 | 1227 |
| 40 | 1759 |  | 40 | 1554 |
| 50 | 2086 |  | 50 | 1841 |
| 60 | 2372 |  | 60 | 2331 |
| 70 | 2863 |  | 70 | 2904 |
| 80 | 3149 |  | 80 | 3231 |
| 90 | 3599 |  | 90 | 3681 |
| 100 | 4090 |  | 100 | 4093 |

Предположим, что в баке находится 50 литров топлива. Тогда первый ДУТ должен показывать около 2086, а второй ДУТ – 1841. Очевидно, что системы мониторинга производят обратный процесс, переводят показания ДУТов. Но в данном случае проблем не возникает ни при каких значениях DUT: если бы мы получили значения , мы высчитали бы объем топлива посредством усреднения показаний на ДУТах:

Таким образом, даже если техника была наклонена, то при наличии двух ДУТов можно получить более точные значения.

Рассмотрим другой случай: предположим, что бак имеет чуть более сложную форму, и мы монтируем в него два датчика уровня топлива. Первый из них покрывает весь диапазон значений литров в баке. Второй же – перекрывает часть (рисунок 1.10)

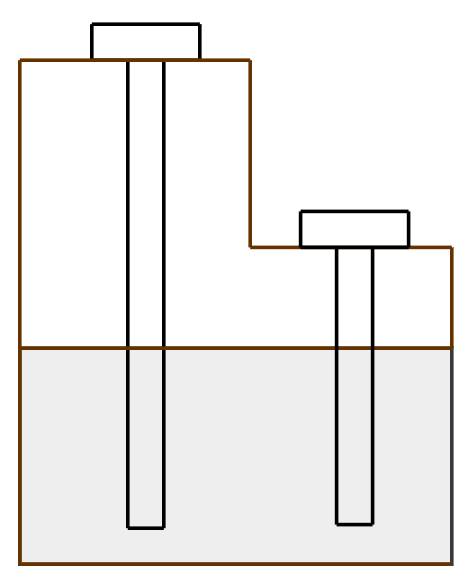


Рисунок 1.10 — Два ДУТа в одном баке (ситуация 2)

Тарировка представлена в таблице 1.3.

Таблица 1.3 — Тарировочная таблица к рисунку 1.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 0 | 30 |  | 0 | 30 |
| 10 | 245 |  | 10 | 532 |
| 20 | 695 |  | 20 | 1023 |
| 30 | 1145 |  | 30 | 1432 |
| 40 | 1841 |  | 40 | 1841 |
| 50 | 1841 |  | 50 | 1963 |
| 60 | 2413 |  | 60 | 2454 |
| 70 | 2945 |  | 70 | 2740 |
| 80 | 3395 |  | 80 | 2740 |
| 90 | 3763 |  | 90 | 2740 |
| 100 | 4090 |  | 100 | 2740 |

Для вычисления топлива в баке будут заведены два датчика, которые преобразуют значения, получаемые с ДУТов, в объем топлива в баке. Предположим, что в баке находится 30 литров. Выдаваемые значения с ДУТов ожидаются в районе 1145 и 1432. И формула для вычисления могла бы выглядеть следующим образом:

Вернёмся к тарировочной таблице 1.3, которая задаёт функцию преобразования. Но из определения функции следует, что при одном и том же значении аргумента не могут выдаваться разные значения функции. Здравый смысл подсказывает, что значение 2740 говорит о том, что 70 литров в баке по показаниям второго ДУТа действительно имеется. И было бы ожидаемым считать, что

Но представим, что баке оказалось большее количество топлива: он был полон. Формула вычисления суммы топлива в баке задана на сервере и не может изменяться в зависимости от каких-то условий. Была выбрана следующая:

Но если бак полон, тогда ДУТы показывают значения 4090 и 2740. Получаемый объем топлива:

не соответствует действительности. Очевидно, что ошибка будет наблюдаться в верхней части бака, и можно было бы не заправлять бак объемами топлива больше 70 литров, что всё равно не является решением. Однако хотелось бы иметь метод, который выдавал бы верные показания при произвольном количестве топлива в баке.

## Цель и задачи выпускной квалификационной работы

Целью данной выпускной квалификационной работы является повышение точности определения количества топлива в баках сложной конфигурации за счет использования большего количества датчиков уровня топлива. В связи с этим были выделены следующие задачи:

* исследовать существующие подходы к решению задач определения уровня топлива в баках сложной конфигурации.
* определить требования к программному обеспечению.
* разработать и реализовать в программном обеспечении алгоритм адаптации данных, получаемых с датчиков уровня топлива для систем спутникового мониторинга.
* провести эксплуатационное приёмочное тестирование разработанного продукта.
* описать алгоритм действия для специалистов технической поддержки по внедрению метода вычислений в системы спутникового мониторинга.

**Вывод**

В разделе были даны основные определения, которые касаются процессов установки, настройки и конфигурирования оборудования для контроля уровня топлива. Описаны вычислительные проблемы при определении уровня топлива. В результате поставлена цель выпускной квалификационной работы и сформулированы задачи.

# ОПИСАНИЕ ПОДХОДА ДЛЯ ВЫЧИСЛЕНИЯ КОЛИЧЕСТВА ТОПЛИВА В БАКАХ СЛОЖНОЙ КОНФИГУРАЦИИ ПО НЕСКОЛЬКИМ ДАТЧИКАМ УРОВНЯ ТОПЛИВА

## Баки – не сообщающиеся сосуды

В первой части работы была показана проблема, возникающая при вычислении количества топлива в баках сложной конфигурации. Под последними же будем понимать произвольный бак, для контроля которого требуется использование двух или более ДУТов, и какой-то из датчиков, в силу расположения, шлет одно и то же показание при разном количестве топлива.

Со стороны сервера выдвигается следующее требование: необходимо составить такую формулу, которая бы не зависела от условий и корректно производила вычисления. Рассмотрим следующую конфигурацию топливного бака (рисунок 2.1):



Рисунок 2.1 — Конфигурация топливного бака

с тремя датчиками уровня топлива , , . Поделим бак на зоны таким образом, чтобы в каждой из зон не было смены количества датчиков уровня топлива (рисунок 2.2):

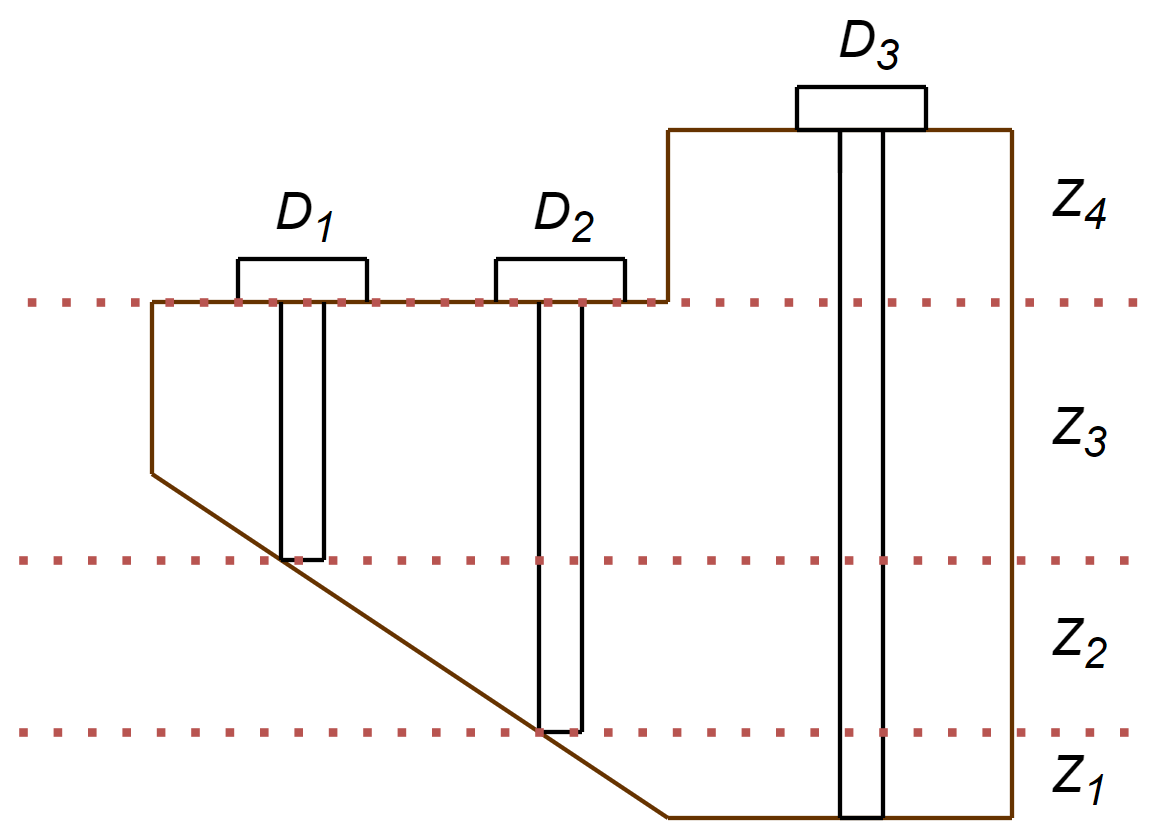


Рисунок 2.2 — Конфигурация топливного бака с выделенными зонами

В таком случае, в и вычисления производятся только под датчику , в – по и , в – по , , . При условии того, что техника стоит ровно, вычисления объема топлива могли бы осуществляться как среднее арифметические показателей датчиков для данной зоны:

Однако реализовать такую разветвляющуюся структуру на стороне сервера невозможно. Как минимум, необходимо знать: какой уровень топлива наблюдается в контексте зон. Но чтобы ответить на этот вопрос, необходимо знать количество топлива в баке. Но ведь именно уровень топлива мы и хотим найти. Следовательно, должен существовать некоторый другой способ вычисления, в котором отсутствовали бы такие разветвления.

Предлагаемое решение сводится к разделению одного физического ДУТа на несколько виртуальных / логических (рисунок 2.3):

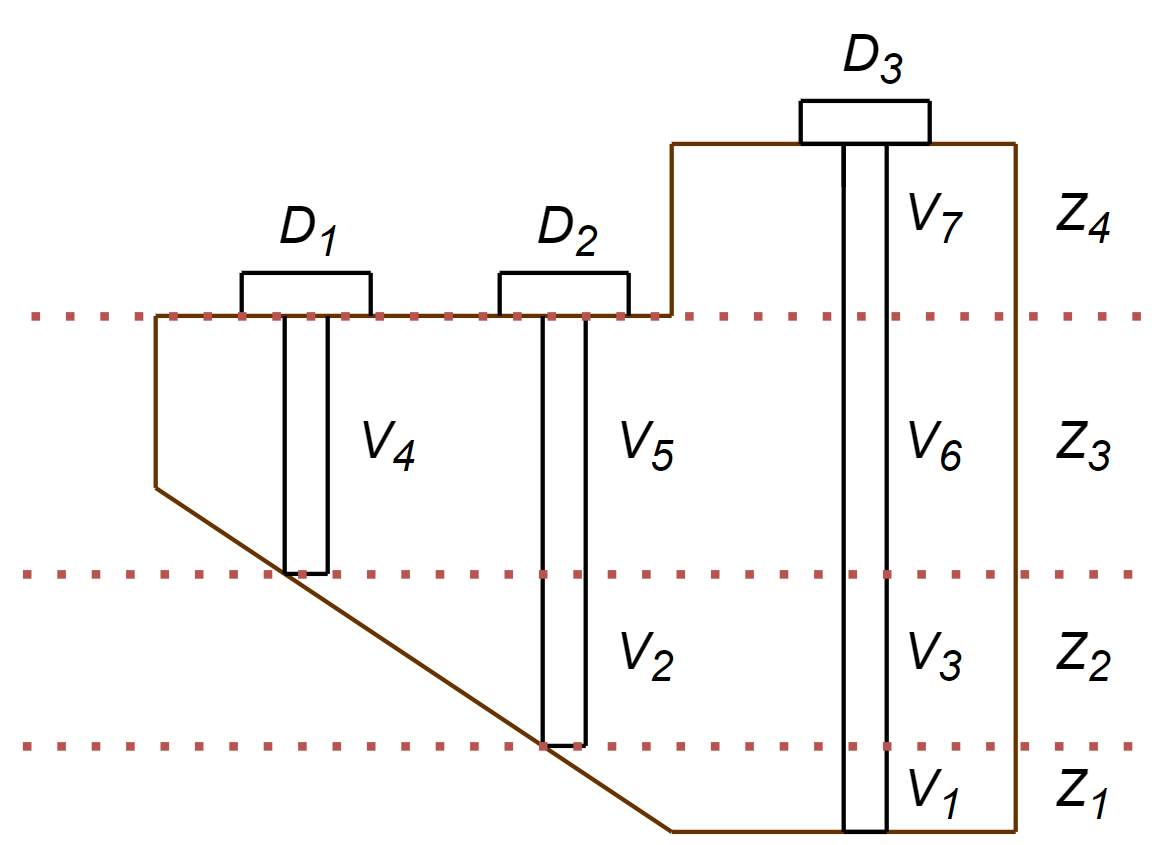


Рисунок 2.3 — Конфигурация топливного бака с выделенными виртуальными датчиками уровня топлива

И требуется задать некоторую функцию где – количество виртуальных датчиков.

Значения с виртуальных датчиков должны вычисляться всегда и входить в итоговую формулу. Будем считать, что виртуальные датчики обладают следующими свойствами:

* Нижняя граница датчика соответствует уровню топлива 0.
* Значения берутся с соответствующего .

Рассмотрим на примере следующей тарировочной таблицы (синим показаны зоны, по которым происходит изменения показаний на датчике уровне топлива при заполнении бака (таблица 2.3)):

Таблица 2.1 — Тарировочная таблица для рисунка 2.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 30 | 30 | 30 |
| 10 | 30 | 30 | 100 |
|  | 20 | 30 | 30 | 213 |
| 30 | 30 | 112 | 345 |
| 40 | 30 | 210 | 450 |
| 50 | 30 | 360 | 567 |
|  | 60 | 30 | 439 | 710 |
| 70 | 120 | 560 | 842 |
| 80 | 241 | 715 | 960 |
| 90 | 365 | 840 | 1080 |
| 100 | 480 | 965 | 1200 |
| 110 | 600 | 1074 | 1315 |
|  | 120 | 715 | 1205 | 1435 |
| 130 | 715 | 1205 | 1553 |
| 140 | 715 | 1205 | 1670 |
| 150 | 715 | 1205 | 1800 |

Она соответствует рисунку 2.3. ДУТ разделяется на 4 виртуальных , , , . Опишем процесс получения их тарировочных таблиц:

1. Разделить тарировочную таблицу ДУТа на множество таблиц меньшего размера в зависимости от зон. Точки, лежащие на границах зоны, должны оказаться в виртуальном ДУТе и первой и второй зоны. Например, значение 223 на должно попасть как в , так и в .
2. Произведём вычитание показаний литров для каждого уровня таким образом, чтобы нижняя граница соответствующего ДУТа соответствовала значению 0.

Таблица 2.2 с пошаговыми вычислениями для разбиения датчика представлена на следующей странице:

Таблица 2.2 — Получение тарировочных таблиц для виртуальных датчиков на (рисунок 2.3)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 30 |  | 0 | 30 |  | 0 | 30 |  | 0 | 30 |  | 0 | 30 |
| 10 | 100 |  | 10 | 100 |  | 10 | 100 |  | 10 | 100 |  | 10 | 100 |
| 20 | 213 |  | 20 | 213 |  | 20 | 213 |  | 20 | 213 |  | 20 | 213 |
| 30 | 345 |  | 30 | 345 |  |  |  |  |  |  |  |  |  |  |
| 40 | 450 |  | 40 | 450 |  |  |  |  |  |  |  |  |  |  |
| 50 | 567 |  | 50 | 567 |  | 20 | 213 |  | 20-20 | 213 |  | 0 | 213 |
| 60 | 710 |  | 60 | 710 |  | 30 | 345 |  | 30-20 | 345 |  | 10 | 345 |
| 70 | 842 |  | 70 | 842 |  | 40 | 450 |  | 40-20 | 450 |  | 20 | 450 |
| 80 | 960 |  | 80 | 960 |  | 50 | 567 |  | 50-20 | 567 |  | 30 | 567 |
| 90 | 1080 |  | 90 | 1080 |  | 60 | 710 |  | 60-20 | 710 |  | 40 | 710 |
| 100 | 1200 |  | 100 | 1200 |  |  |  |  |  |  |  |  |  |  |
| 110 | 1315 |  | 110 | 1315 |  |  |  |  |  |  |  |  |  |  |
| 120 | 1435 |  | 120 | 1435 |  | 60 | 710 |  | 60-60 | 710 |  | 0 | 710 |
| 130 | 1553 |  | 130 | 1553 |  | 70 | 842 |  | 70-60 | 842 |  | 10 | 842 |
| 140 | 1670 |  | 140 | 1670 |  | 80 | 960 |  | 80-60 | 960 |  | 20 | 960 |
| 150 | 1800 |  | 150 | 1800 |  | 90 | 1080 |  | 90-60 | 1080 |  | 30 | 1080 |
|  |  |  |  |  |  | 100 | 1200 |  | 100-60 | 1200 |  | 40 | 1200 |
|  |  |  |  |  |  | 110 | 1315 |  | 110-60 | 1315 |  | 50 | 1315 |
|  |  |  |  |  |  | 120 | 1435 |  | 120-60 | 1435 |  | 60 | 1435 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | 120 | 1435 |  | 120-120 | 1435 |  | 0 | 1435 |
|  |  |  |  |  |  | 130 | 1553 |  | 130-120 | 1553 |  | 10 | 1553 |
|  |  |  |  |  |  | 140 | 1670 |  | 140-120 | 1670 |  | 20 | 1670 |
|  |  |  |  |  |  | 150 | 1800 |  | 150-120 | 1800 |  | 30 | 1800 |

Таким образом, физический датчик был разбит на множество виртуальных, которые также задают функции. Рассмотрим виртуальный датчик чуть более подробно (таблица 2.3):

Таблица 2.3 — значения виртуального датчика

|  |  |
| --- | --- |
|  |  |
| 0 | 710 |
| 10 | 842 |
| 20 | 960 |
| 30 | 1080 |
| 40 | 1200 |
| 50 | 1315 |
| 60 | 1435 |

Если с придёт значение меньше чем 710 – в системах спутникового мониторинга вычисление будет осуществляться за счёт экстраполяции, и мы получим отрицательное количество топлива по данному ДУТу. С этой целью для каждого датчика добавляются дополнительные значения, чтобы подавить экстраполяцию (таблица 2.4):

Таблица 2.4 — значения виртуального датчика с добавленными значениями для подавления экстраполяции

|  |  |
| --- | --- |
|  |  |
| 0 | 709 |
| 0 | 710 |
| 10 | 842 |
| 20 | 960 |
| 30 | 1080 |
| 40 | 1200 |
| 50 | 1315 |
| 60 | 1435 |
| 60 | 1436 |

И тогда формула для разбираемого случая может быть выражена следующим образом:

Ещё раз подчеркиваю, для каждого из виртуальных датчиков возможны следующие ситуации:

* Значение на физическом ДУТе меньше, чем покрываемый интервал виртуальным ДУТом – виртуальный ДУТ будет показывать 0 литров
* Значение на физическом ДУТе попадает в покрываемый интервал – виртуальный ДУТ будет показывать видимое им количество литров
* Значение на физическом ДУТе больше покрываемого интервала – виртуальный ДУТ будет показывать максимальное видимое значение на виртуальном ДУТе.

Какое бы значение не шло по физическому ДУТу, виртуальный ДУТ всегда может выдать значение уровня топлива.

## Баки – сообщающиеся сосуды

Подход, описанный ранее, может быть применён и для баков – сообщающихся сосудов. Рассмотрим пример (рисунок 2.4):

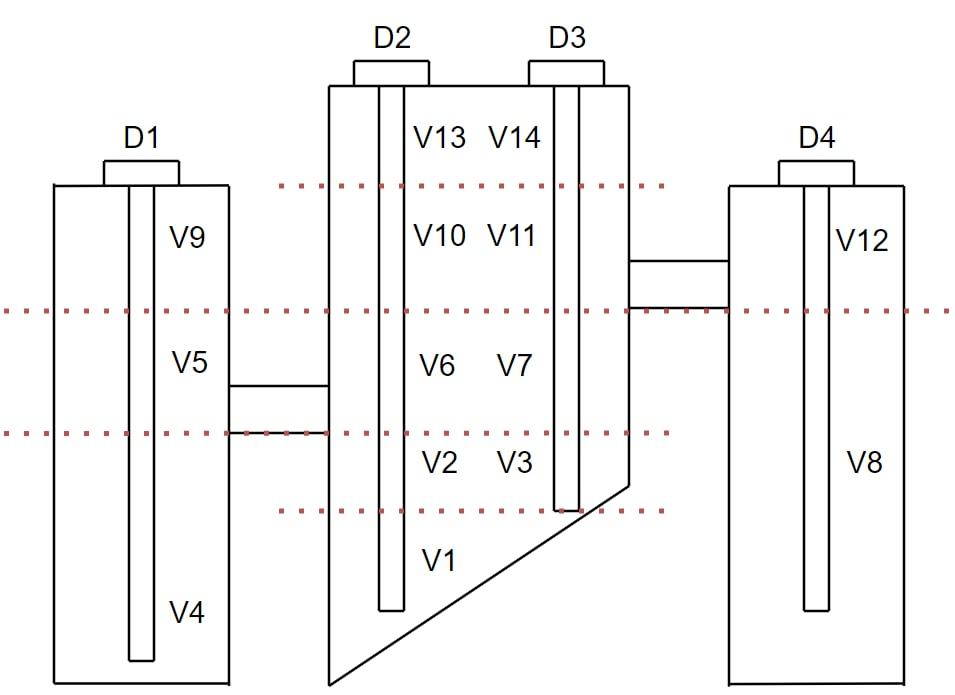


Рисунок 2.4 — Топливный бак в форме сообщающегося сосудов

который носит учебный характер, но позволяет лучше разобраться с последовательностью действий при работе с такой конфигурацией. Сгенерируем пример тарировочной таблицы. В таблице 2.5 синим цветом выделены изменения показаний с ДУТов при заполнении бака топливом. Правая же часть таблицы показывает диапазон значений в литрах, за которую отвечает тот или иной виртуальный ДУТ:

Таблица 2.5 — Тарировочная таблица для рисунка 2.4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | | ( | | ( | |
| 0 | 30 | 30 | 30 | 30 |  |  | | 0 | | 20 | |
| 10 | 30 | 100 | 30 | 30 |  |  | | 20 | | 50 | |
| 20 | 30 | 160 | 30 | 30 |  |  | |
| 30 | 30 | 220 | 100 | 30 |  |  | | 50 | | 100 | |
| 40 | 30 | 280 | 160 | 30 |  |  | | 100 | | 130 | |
| 50 | 30 | 350 | 220 | 30 |  |  | |
| 60 | 100 | 350 | 220 | 30 |  |  | |
| 70 | 160 | 350 | 220 | 30 |  |  | | 130 | | 200 | |
| 90 | 220 | 350 | 220 | 30 |  |  | | 200 | | 240 | |
| 100 | 280 | 350 | 220 | 30 |  |  | |
| 110 | 340 | 440 | 280 | 30 |  |  | |
| 120 | 400 | 500 | 347 | 30 |  |  | |
| 130 | 460 | 560 | 400 | 30 |  |  | | 240 | | 260 | |
| 140 | 460 | 560 | 400 | 100 |  |  | |
| 150 | 460 | 560 | 400 | 160 |  | |  | |  | |  | |
| 160 | 460 | 560 | 400 | 220 |  | |  | |  | |  | |
| 170 | 460 | 560 | 400 | 280 |  | |  | |  | |  | |
| 180 | 460 | 560 | 400 | 340 |  | |  | |  | |  | |
| 190 | 460 | 560 | 400 | 400 |  | |  | |  | |  | |
| 200 | 460 | 560 | 400 | 460 |  | |  | |  | |  | |
| 210 | 520 | 620 | 460 | 520 |  | |  | |  | |  | |
| 220 | 580 | 680 | 520 | 580 |  | |  | |  | |  | |
| 230 | 600 | 740 | 580 | 600 |  | |  | |  | |  | |
| 240 | 664 | 800 | 620 | 664 |  | |  | |  | |  | |
| 250 | 664 | 850 | 680 | 664 |  | |  | |  | |  | |
| 260 | 664 | 900 | 730 | 664 |  | |  | |  | |  | |

Прерывания синих столбцов говорят нам о том, что началась зона сообщения, и топливо начало переливаться в другой сосуд. При достижении уровня сообщения показания с ДУТов продолжают расти далее.

Тарировочную таблицу в представлении слева несколько трудно будет впоследствии разбивать на датчики. Рекомендуется выполнить дополнительное преобразование, заключающееся в дублировании некоторых строк исходной таблицы. После выполнения дублирования можно легко вычленить логические датчики (пример разбиения представлен в таблице 2.6).

После проведения преобразования выполним разделение таблицы на виртуальные датчики уровня топлива. Каждый из виртуальных датчиков уровня топлива будем характеризовать следующим набором свойств:

* V\_DUT\_NUMBER – номер виртуального ДУТа
* PARENT\_DUT\_NAME – имя родительского ДУТа
* PARENT\_LITERS – показания литров родительского ДУТА на промежутке виртуального датчика уровня топлива
* V\_DUT\_LITERS - преобразованные показания литров по формуле:
* DUT\_VALUES – показания датчика уровня топлива на промежутке
* ZONE\_NUMBER – номер зоны, в которой расположен виртуальный датчик уровня топлива.

При обработке таблицы программой вы должны получить программный объект со следующим содержимым:

V\_DUT\_NUMBER 1

PARENT\_DUT\_NAME D2

PARENT\_LITERS [0, 10, 20]

V\_DUT\_LITERS [0, 10, 20]

DUT\_VALUES [30, 100, 160]

ZONE\_NUMBER 1

V\_DUT\_NUMBER 2

PARENT\_DUT\_NAME D2

PARENT\_LITERS [20, 30, 40, 50]

V\_DUT\_LITERS [0, 10, 20, 30]

DUT\_VALUES [160, 220, 280, 350]

ZONE\_NUMBER 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 2.6 — Преобразование таблицы | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 30 | 30 | 30 | 30 |  | 0 | 30 | 30 | 30 | 30 |  | 0 |  |  |  |  |  |
| 10 | 30 | 100 | 30 | 30 |  | 10 | 30 | 100 | 30 | 30 |  | 10 |  |  |  |
| 20 | 30 | 160 | 30 | 30 |  | 20 | 30 | 160 | 30 | 30 |  | 20 |  |  |  |
| 30 | 30 | 220 | 100 | 30 |  | 20 | 30 | 160 | 30 | 30 |  | 20 |  |  |  |  |  |
| 40 | 30 | 280 | 160 | 30 |  | 30 | 30 | 220 | 100 | 30 |  | 30 |  |  |
| 50 | 30 | 350 | 220 | 30 |  | 40 | 30 | 280 | 160 | 30 |  | 40 |  |  |
| 60 | 100 | 350 | 220 | 30 |  | 50 | 30 | 350 | 220 | 30 |  | 50 |  |  |
| 70 | 160 | 350 | 220 | 30 |  | 50 | 30 | 350 | 220 | 30 |  | 50 |  |  |  |  |  |
| 90 | 220 | 350 | 220 | 30 |  | 60 | 100 | 350 | 220 | 30 |  | 60 |  |  |  |
| 100 | 280 | 350 | 220 | 30 |  | 70 | 160 | 350 | 220 | 30 |  | 70 |  |  |  |
| 110 | 340 | 440 | 280 | 30 |  | 90 | 220 | 350 | 220 | 30 |  | 90 |  |  |  |
| 120 | 400 | 500 | 347 | 30 |  | 100 | 280 | 350 | 220 | 30 |  | 100 |  |  |  |
| 130 | 460 | 560 | 400 | 30 |  | 100 | 280 | 350 | 220 | 30 |  | 100 |  |  |  |  |  |
| 140 | 460 | 560 | 400 | 100 |  | 110 | 340 | 440 | 280 | 30 |  | 110 |  |
| 150 | 460 | 560 | 400 | 160 |  | 120 | 400 | 500 | 347 | 30 |  | 120 |  |
| 160 | 460 | 560 | 400 | 220 |  | 130 | 460 | 560 | 400 | 30 |  | 130 |  |
| 170 | 460 | 560 | 400 | 280 |  | 130 | 460 | 560 | 400 | 30 |  | 130 |  |  |  |  |  |
| 180 | 460 | 560 | 400 | 340 |  | 140 | 460 | 560 | 400 | 100 |  | 140 |  |  |  |
| 190 | 460 | 560 | 400 | 400 |  | 150 | 460 | 560 | 400 | 160 |  | 150 |  |  |  |
| 200 | 460 | 560 | 400 | 460 |  | 160 | 460 | 560 | 400 | 220 |  | 160 |  |  |  |
| 210 | 520 | 620 | 460 | 520 |  | 170 | 460 | 560 | 400 | 280 |  | 170 |  |  |  |
| 220 | 580 | 680 | 520 | 580 |  | 180 | 460 | 560 | 400 | 340 |  | 180 |  |  |  |
| 230 | 600 | 740 | 580 | 600 |  | 190 | 460 | 560 | 400 | 400 |  | 190 |  |  |  |
| 240 | 664 | 800 | 620 | 664 |  | 200 | 460 | 560 | 400 | 460 |  | 200 |  |  |  |
| 250 | 664 | 850 | 680 | 664 |  | 200 | 460 | 560 | 400 | 460 |  | 200 |  |  |  |  |  |
| 260 | 664 | 900 | 730 | 664 |  | 210 | 520 | 620 | 460 | 520 |  | 210 |
|  |  |  |  |  |  | 220 | 580 | 680 | 520 | 580 |  | 220 |
|  |  |  |  |  |  | 230 | 600 | 740 | 580 | 600 |  | 230 |
|  |  |  |  |  |  | 240 | 664 | 800 | 620 | 664 |  | 240 |
|  |  |  |  |  |  | 240 | 664 | 800 | 620 | 664 |  | 240 |  |  |  |  |  |
|  |  |  |  |  |  | 250 | 664 | 850 | 680 | 664 |  | 250 |  |  |
|  |  |  |  |  |  | 260 | 664 | 900 | 730 | 664 |  | 260 |  |  |

V\_DUT\_NUMBER 3

PARENT\_DUT\_NAME D3

PARENT\_LITERS [20, 30, 40, 50]

V\_DUT\_LITERS [0, 10, 20, 30]

DUT\_VALUES [30, 100, 160, 220]

ZONE\_NUMBER 2

V\_DUT\_NUMBER 4

PARENT\_DUT\_NAME D1

PARENT\_LITERS [50, 60, 70, 90, 100]

V\_DUT\_LITERS [0, 10, 20, 40, 50]

DUT\_VALUES [30, 100, 160, 220, 280]

ZONE\_NUMBER 3

V\_DUT\_NUMBER 5

PARENT\_DUT\_NAME D1

PARENT\_LITERS [100, 110, 120, 130]

V\_DUT\_LITERS [0, 10, 20, 30]

DUT\_VALUES [280, 340, 400, 460]

ZONE\_NUMBER 4

V\_DUT\_NUMBER 6

PARENT\_DUT\_NAME D2

PARENT\_LITERS [100, 110, 120, 130]

V\_DUT\_LITERS [0, 10, 20, 30]

DUT\_VALUES [350, 440, 500, 560]

ZONE\_NUMBER 4

V\_DUT\_NUMBER 7

PARENT\_DUT\_NAME D3

PARENT\_LITERS [100, 110, 120, 130]

V\_DUT\_LITERS [0, 10, 20, 30]

DUT\_VALUES [220, 280, 347, 400]

ZONE\_NUMBER 4

и так далее до:

V\_DUT\_NUMBER 12

PARENT\_DUT\_NAME D4

PARENT\_LITERS [200, 210, 220, 230, 240]

V\_DUT\_LITERS [0, 10, 20, 30, 40]

DUT\_VALUES [460, 520, 580, 600, 664]

ZONE\_NUMBER 6

V\_DUT\_NUMBER 13

PARENT\_DUT\_NAME D2

PARENT\_LITERS [240, 250, 260]

V\_DUT\_LITERS [0, 10, 20]

DUT\_VALUES [800, 850, 900]

ZONE\_NUMBER 7

V\_DUT\_NUMBER 14

PARENT\_DUT\_NAME D3

PARENT\_LITERS [240, 250, 260]

V\_DUT\_LITERS [0, 10, 20]

DUT\_VALUES [620, 680, 730]

ZONE\_NUMBER 7

Полученные значения могут быть использованы в дальнейшем для вычисления формулы и последующего внедрения в системы спутникового мониторинга.

Аналогично ситуации без сообщающихся сосудов нужно добавить дополнительные значения для избегания некорректной экстраполяции в системах спутникового мониторинга (см. таблицы 2.3-2.4).

Описанный выше процесс является довольно трудоемким, и проведение его человеком может привести к вычислительным ошибкам. С другой стороны, хотелось бы избавить пользователя в принципе от проведения таких расчётов в принципе и возложить данный процесс на программное обеспечение. Именно решению данной проблемы будет отведена оставшаяся часть работы.

## Требования к программному обеспечению

Данная выпускная квалификационная работа не ставит целью сбор исчерпывающего набора требований (с этим вопросом можно ознакомиться в (10), (11)). Ниже будут перечислены только высокоуровневые требования, касаемые аспектов поведения программы:

* Исходными данными в задаче является набор тарировочных таблиц. Пример, созданный в MS Excel (рисунок 2.5):

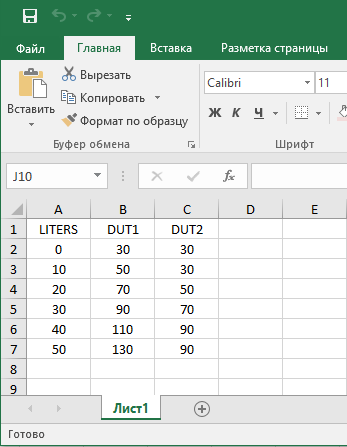


Рисунок 2.5 — Формат исходных данных

* + Первый столбец обязан иметь название LITERS. Именовать последующие столбцы разрешается произвольно (указанные значения будут использоваться в качестве имени родительского датчика уровня топлива).
* В случае наличия ошибок в исходных данных, программа информирует пользователя. Под ошибками считать:
  + Наличие пропусков в таблице в произвольном столбце тарировочной таблицы (рисунок 2.6):

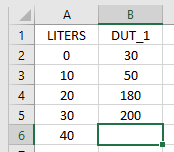


Рисунок 2.6 — Таблица с пропущенным значениям

* + Колонка LITERS не является возрастающей (рисунок 2.7):

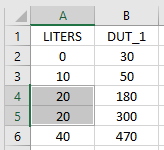


Рисунок 2.7 — Тарировочная таблица с невозрастающими показаниями количества уровня топлива

* + Наличие убывающих значений в показании ДУТов (все оставшиеся колонки, кроме LITERS) при увеличении количества топлива (рисунок 2.8):

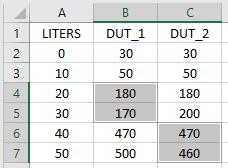


Рисунок 2.8 — Убывающие значения LITERS

* + Наличие двух последовательных рядов таблицы, в которых не наблюдается изменение показаний для ДУТов (рисунок 2.9):

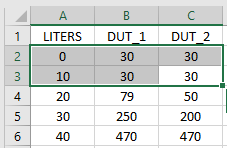


Рисунок 2.9 — Строки с не отличающимися значениями по всем датчикам уровня топлива

* Поддерживаемые форматы файлов исходных данных: XLSX, XLS.
* При наличии нескольких баков требуется каждый отдельный бак завести на отдельном листе (рисунок 2.10). Имя листа будет использовано в качестве имени бака.

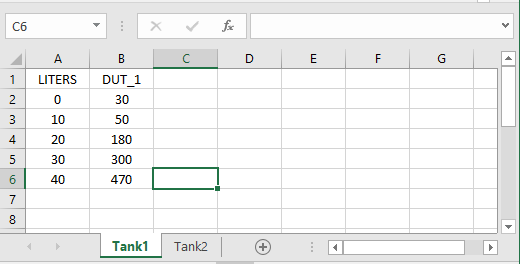
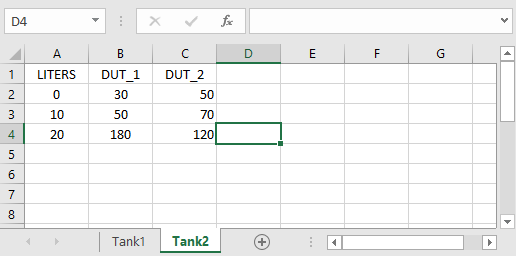


Рисунок 2.10 — Пример исходных данных при нескольких датчиках уровня топлива

* Нумерация создаваемых датчиков должна производиться на основании порядка заполнения топлива баком
* Программа получает на вход файл из некоторой директории. После его обработки в той же директории должна создаться папка, которая содержит в себе следующие файлы:
  + Разбиение всех физических датчиков на логические (виртуальный датчики) и их тарировочные таблицы.
  + Текстовый файл с формулой для вычисления уровня топлива бака
  + Графики:
    - Расположение ДУТов относительно друг друга

## Выбор инструментов для разработки

Для разработки приложения были выбраны следующие инструменты:

* Python 3.9 – высокоуровневый язык общего назначения, который ориентирован на быстрое написание прототипа, читаемости кода и его качества. Среда разрботки: PyCharm Community. Предпочтение было отдано языку, так как он является бесплатным, и его сторонние библиотеки позволяют легко оперировать данными.
* Git – система контроля версий.

Библиотеки:

* Pandas. Библиотека, предоставляющая классы Series и DataFrame, которая позволяет удобно манипулировать многомерными массивами. Опирается на работы Numpy.
* Numpy. Библиотека предоставляет набор инструментов для работы с многомерными массивами. Такие встроенные типы данных как кортежи ведут к неоправданной затрате памяти. Данный же инструмент позволяет использовать более оптимизированные структуры данных для решения задач.
* Matplotlib. Библиотека разработана для визуализации данных. Инструмент не отличается дружелюбный интерфейсом к новичкам, но для опытных пользователей предоставляет широкий спектр низкоуровневых возможностей для работы с графикой.
* Openpyxl, xlrd. Библиотеки для работы с форматами xlsx.

## Выводы

Во втором разделе был описан предлагаемый алгоритм (без детализации) для баков – не сообщающихся сосудов и баков – сообщающихся сосудов, а также сформулированы требования к программному обеспечению.

# РЕАЛИЗАЦИЯ АЛГОРИТМА ДЛЯ РАЗБИЕНИЯ ДАТЧИКОВ УРОВНЯ ТОПЛИВА И ПОСТРОЕНИЯ ЗАВИСИМОСТЕЙ

## Описание алгоритма

Процесс разработки организуем «сверху вниз» (нисходящее программирование), которое начинается с определения целей решения проблемы, после чего будем детализировать каждый этап до тех пор, пока не будет разработана детальная программа.

Наибольшую ценность для пользователя представляют:

* Тарировочные таблицы в формате .csv для виртуальных датчиков уровня топлива.
* Текстовый файл с формулой вычисления уровня топлива по виртуальным датчикам уровня топлива.
* Графические зависимости.

Блок-схема в укрупненных блоках представлена на рисунке 3.1.

Опишем удобное представление для объекта fuel\_system\_config:

fuel\_system\_config = {

“tank\_name\_1”: {

“calibration\_table” : <pd.DataFrame>

“virtual\_duts\_table”: <pd.DataFrame>

}

“tank\_name\_2”: {

“calibration\_table” : <pd.DataFrame>

“virtual\_duts\_table”: <pd.DataFrame>

}

}

где tank\_name\_i – наименование бака, calibration\_table – тарировочная таблица для бака, virtual\_duts\_table – таблица со столбцами: V\_DUT\_NUMBER, PARENT\_DUT\_NAME, PARENT\_LITERS, V\_DUT\_LITERS, DUT\_VALUES, ZONE\_NUMBER.

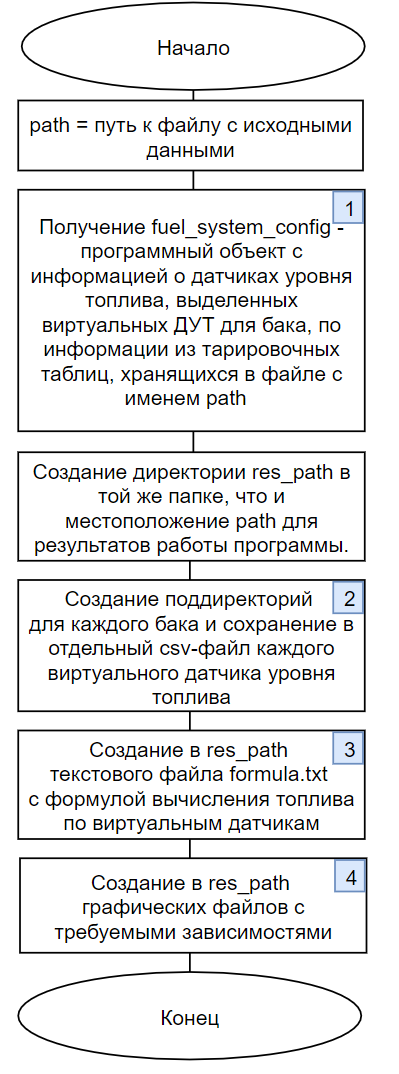


Рисунок 3.1 — Блок-схема в укрупненных блоках

Получение программного объекта fuel\_system\_config состоит из следующих этапов (рисунок 3.2; листинг 3.1):



Рисунок 3.2 — Получение объекта fuel\_system\_config

import pandas as pd  
import openpyxl  
  
from DataChecker import DataChecker  
from TableTransformer import TableTransformer  
from TableSplitter import TableSplitter  
  
class FuelSystemConfigCreator:  
  
 def \_\_init\_\_(self, path):  
 self.path = path  
 self.fuel\_system\_config = {}  
 self.open\_file()  
 self.get\_virtual\_dut\_table()  
  
 def open\_file(self):  
 try:  
 wb = openpyxl.load\_workbook(self.path)  
 except Exception:  
 print(**f"open error file** {self.path}**."**)  
 n\_sheets = len(wb.sheetnames)  
 for sheet in range(n\_sheets):  
 self.fuel\_system\_config[sheet] = {}  
 calibration\_table = pd.read\_excel(self.path, sheet\_name=sheet, engine=**'openpyxl'**)  
 DataChecker(calibration\_table).check\_table()  
 self.fuel\_system\_config[sheet][**"calibration\_table"**] = calibration\_table  
  
 def get\_virtual\_dut\_table(self):  
 for tank\_name in self.fuel\_system\_config:  
 calibration\_table = self.fuel\_system\_config[tank\_name][**"calibration\_table"**]  
 \_ = TableTransformer(calibration\_table).transform()  
 self.fuel\_system\_config[tank\_name][**"virtual\_duts\_table"**] = TableSplitter(\_).split()  
  
 def create(self):  
 self.open\_file()  
 self.get\_virtual\_dut\_table()  
 return self.fuel\_system\_config

Листинг 3.1 – реализация алгоритма, описанного на рисунке 3.2

Описание алгоритмов проверки исходных данных на предмет ошибок производиться не будет, так как функции довольно просты.

После чтения исходных данных последовательно перебираются тарировочные таблицы и выполняется создание промежуточной таблицы (описанной во 2 части работы), которая упрощает проведение дальнейших вычислений (блок-схема представлена на рисунке 3.4).

Преобразованная таблица разбивается на виртуальные датчики уровня топлива (блок схема представлена на рисунке 3.5). Детализация блока 1.2.1 посредством блок-схем описываться не будет, но реализация может быть позаимствована из с листинга 3.2.

При вычислении промежуточной таблицы и в процессе разбиения на виртуальные ДУТы используется функция, описанная на рисунке 3.6:

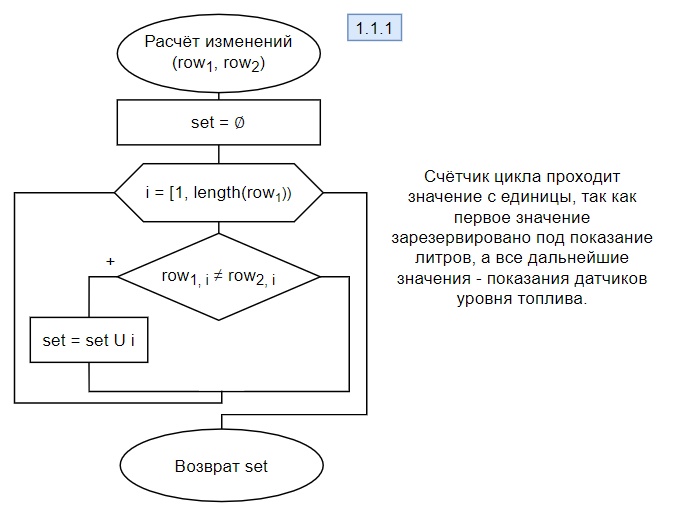


Рисунок 3.3 — Получение объекта fuel\_system\_config

которая возвращает множество номеров датчиков уровня топлива, зафиксировавших изменения между двумя строками тарировочной таблицы.

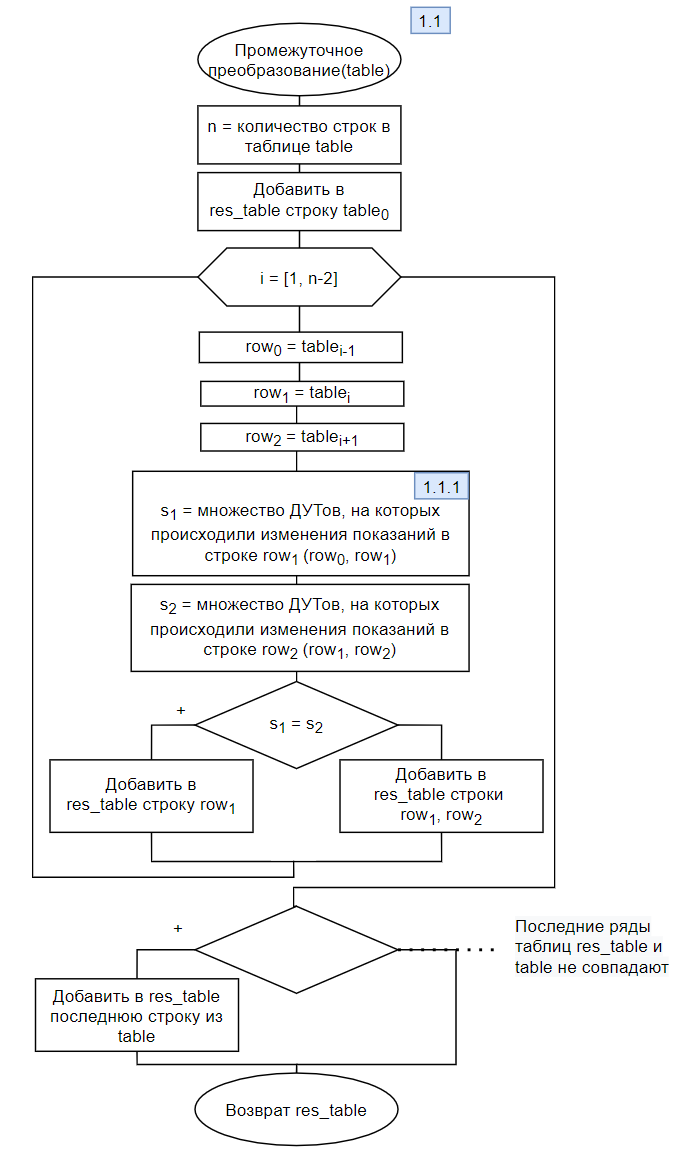


Рисунок 3.4 — Преобразование таблицы

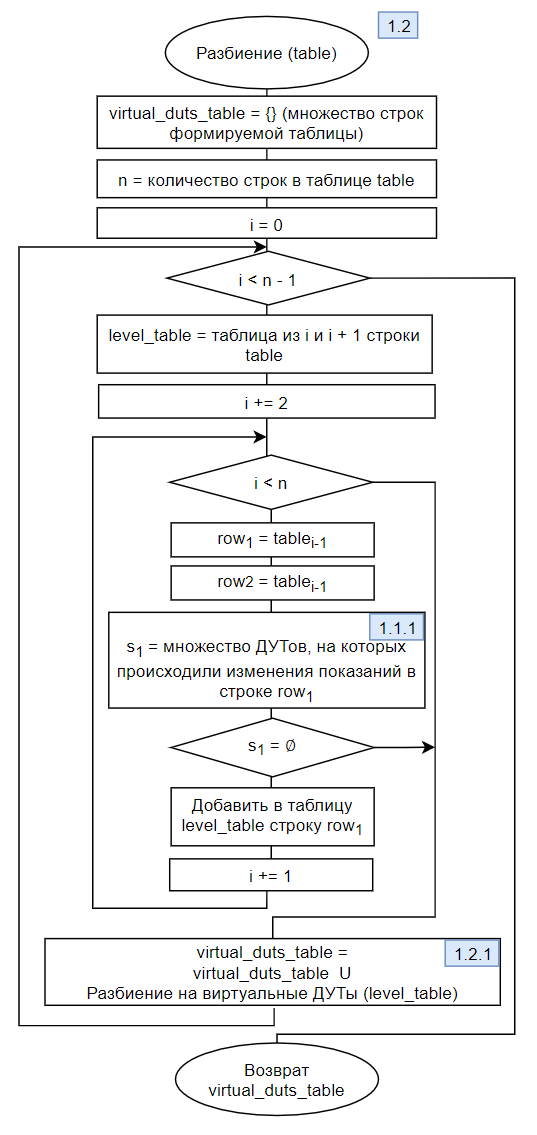


Рисунок 3.5 — Разбиение на виртуальные датчики уровня топлива

import pandas as pd  
  
  
class TableTransformer:  
  
 def \_\_init\_\_(self, pd\_df: pd.DataFrame):  
 self.table = pd\_df  
  
 def transform(self):  
 res\_table = [self.table.iloc[0]]  
 for i in range(1, self.table.shape[0] - 1):  
 row0 = self.table.iloc[i - 1]  
 row1 = self.table.iloc[i]  
 row2 = self.table.iloc[i + 1]  
 s1 = [i for i in range(1, len(row1)) if row0[i] != row1[i]]  
 s2 = [i for i in range(1, len(row2)) if row1[i] != row2[i]]  
 if s1 == s2:  
 res\_table.append(self.table.iloc[i])  
 else:  
 res\_table.append(self.table.iloc[i])  
 res\_table.append(self.table.iloc[i])  
 last\_row\_res = res\_table[-1]  
 last\_row\_table = self.table.iloc[-1, :]  
 if any(last\_row\_res != last\_row\_table):  
 res\_table.append(last\_row\_table)  
  
 res\_table = pd.DataFrame(res\_table)  
 res\_table.index = range(res\_table.shape[0])  
  
 return res\_table

Листинг 3.2 – реализация алгоритма, описанного на рисунке 3.4

import pandas as pd  
  
  
class TableSplitter:  
  
 def \_\_init\_\_(self, table: pd.DataFrame):  
 self.table = table  
 self.virtual\_duts\_table = []  
 self.zone\_number = 0  
 self.\_\_split()  
  
 def \_\_split(self):  
 n = len(self.table)  
 i = 0  
 while i < n - 1:  
 tmp = self.table.iloc[i:i+2][:]  
 i += 2  
 while i < n:  
 row1 = self.table.iloc[i - 1][:]  
 row2 = self.table.iloc[i][:]  
 change = [i for i in range(1, len(row1)) if row1[i]!=row2[i]]  
 if not change:  
 break  
 tmp = tmp.append(row2)  
 i += 1  
 self.\_\_update(tmp)  
 return self

def \_\_update(self, tmp):  
 self.zone\_number += 1  
 for dut\_name in tmp.columns[1:]:  
 if tmp[dut\_name].iloc[0] != tmp[dut\_name].iloc[-1]:  
 d = pd.Series({  
 **"V\_DUT\_NUMBER"**: len(self.virtual\_duts\_table) + 1,  
 **"PARENT\_DUT\_NAME"**: dut\_name,  
 **"PARENT\_LITERS"**: tmp.LITERS.values,  
 **"V\_DUT\_LITERS"**: [x - tmp.LITERS.values[0]

for x in tmp.LITERS.values],  
 **"DUT\_VALUES"**: tmp[dut\_name].values,  
 **"ZONE\_NUMBER"**: self.zone\_number  
 })  
 self.virtual\_duts\_table.append(d)  
  
 def split(self):  
 return self.virtual\_duts\_table

Листинг 3.3 – реализация алгоритма, описанного на рисунке 3.5

Создание графических файлов необходимо для контроля качества тарировки. Наличие специфических скачков позволяет обнаружить ошибки в процессе тарирования. Для контроля выводимой формулы хотелось бы также иметь представление о взаимном расположении датчиков уровня топлива в баке: существует необходимость понимать - за какой диапазон показаний отвечает тот или иной датчик.

Количество графиков должно создаваться в зависимости от конфигурации системы и определяется как:

Для каждого из баков должны быть построены следующие зависимости:

* Количество топлива в баке от показаний на датчике уровня топлива
* Покрываемый диапазон топлива тем или иным датчиком уровня топлива.

Решение предлагается реализовать посредством библиотеки , которая предоставляет низкоуровневый интерфейс для работы с графикой.

Код для генерации изображения довольно сложен и приведение его в тексте работы было бы избыточно: внимание хочется уделить получаемым результатам. Для тарировочной таблицы 2.6, которая является отображением следующей конфигурации (рисунок 3.6):

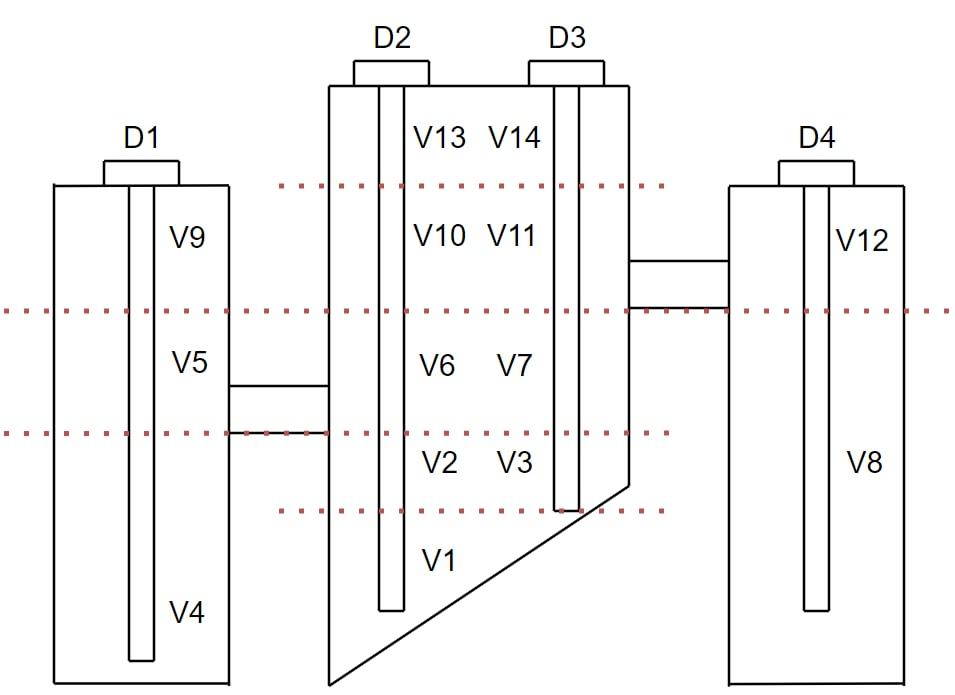


Рисунок 3.6 — Конфигурация бака

выполняется построение следующих графиков (рисунок 3.7):

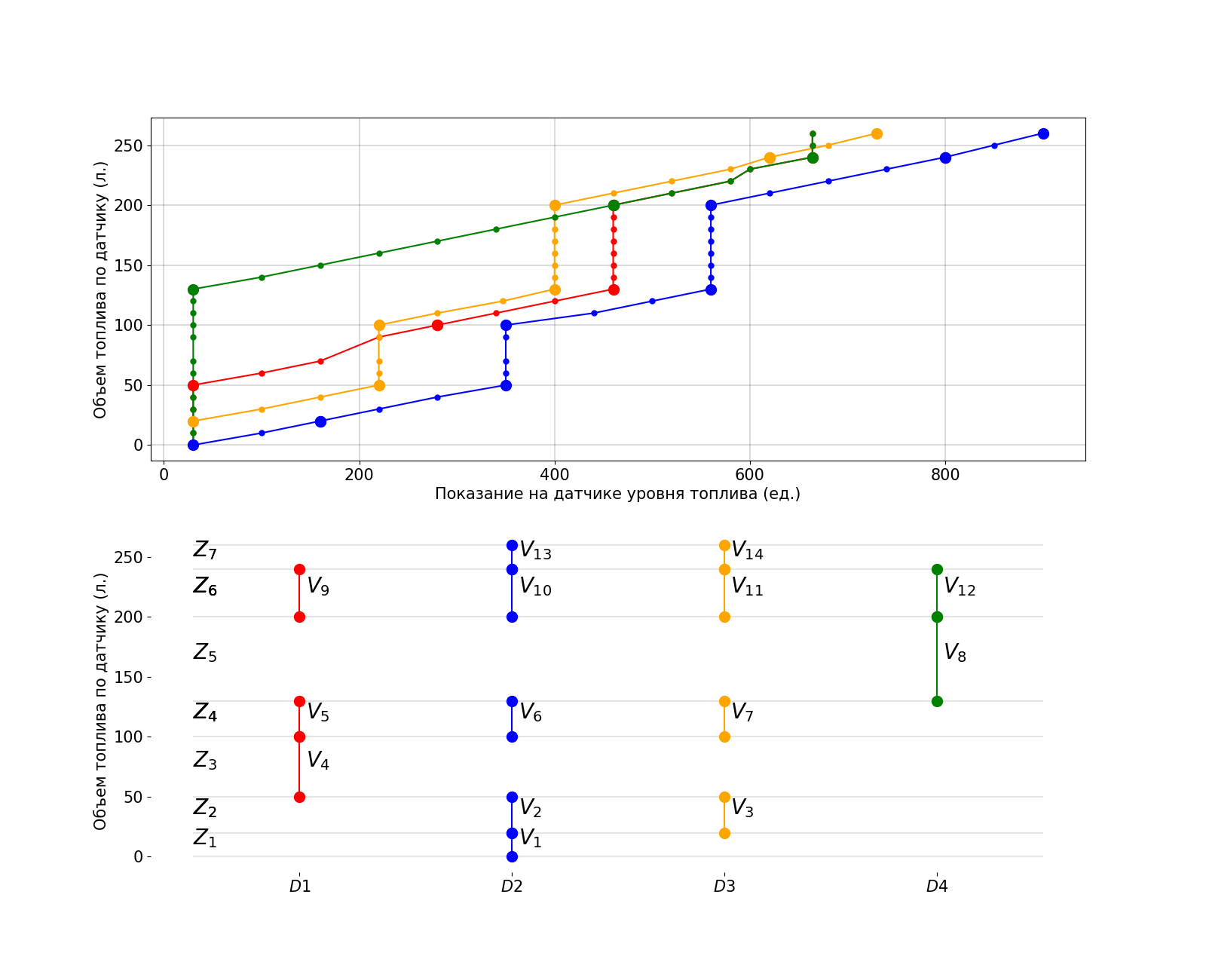


Рисунок 3.7 — Графики для конфигурации к рисунку 3.6

Сфокусируемся на выводимых данных. Реализованная программа выделила 7 зон. Область покрытия каждого из ДУТов отображена на нижнем графике. Видно, что на имеется разрыв между виртуальными датчиками и что является свидетельством того, что на данном промежутке началось переливание топлива к некоторому другому датчику. И им является датчик . Как только показания на достигнут максимума – начнут происходить изменения по всем датчикам уровня топлива.

Код для построения графиков может быть найден в приложении к данной работе.

Вычисление формулы при наличии разделения на виртуальные ДУТы не представляет сложности. Необходимо пройтись по всем зонам , и тогда количество топлива может быть вычислено по следующей формуле:

где – среднее арифметическое показаний по виртуальным датчикам уровня топлива на уровне .

Файл, который является точкой входа в программу описан в листинге 3.4. В нём объявлена функция process, которая может быть модифицирована в зависимости от нужд. В данном случае создаётся директория с файлами-результатами работы программы там же, где и исходный файл (подробнее описано в требованиях к заданию).

Структура проекта представлена на рисунке 3.8. Исходный код программы поделен на 7 файлов:

* Класс DataChecker предназначен для проверки исходных данных на наличие ошибок;
* Класс FormulaCreator генерирует формулы для вычисления топлива в различном представлении для дальнейшего использования;
* Класс FuelSystemConfigCreator выполняет создание программного объекта fuel\_system\_config, содержание которого описано в разделе 2.
* Файл main – точка входа в программу.
* Класс Plotter выполняет построение зависимостей в виде графиков.
* Класс TableSplitter производит разбиение тарировочной таблицы на виртуальные датчики уровня топлива.
* Класс TableTransformer выполняет подготовку исходных данных для удобного разбиения данных функцией TableSplitter.

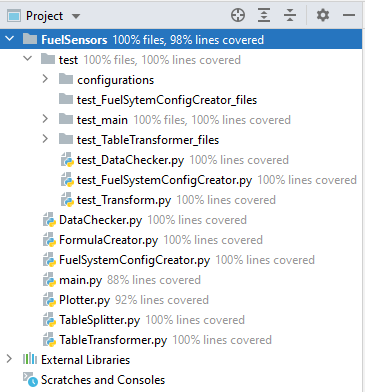


Рисунок 3.8 — Структура разработанного проекта

Основные аспекты, которые касаются рассмотрения алгоритмов были описаны. Остановимся подробнее на процессе установки приложения и интеграции результатов вычислений в систему спутникового мониторинга .

import os  
from tkinter import filedialog  
  
import pandas as pd  
  
from FuelSystemConfigCreator import FuelSystemConfigCreator  
from Plotter import Plotter  
from FormulaCreator import FormulaCreator  
  
  
def process(path=None, save\_path=None):  
 if path is None:  
 path = filedialog.askopenfilename(filetypes=[(**"Excel files"**, **".xlsx .xls"**)])  
 save\_path = **"."**.join(path.split(**"."**)[:-1]) + **"/"** os.makedirs(save\_path, exist\_ok=True)  
 fsc = FuelSystemConfigCreator(path).create()  
 fc = FormulaCreator(fsc)  
  
 for tank\_name in fsc:  
 os.makedirs(save\_path + **r"/"** + tank\_name, exist\_ok=True)  
 for virtual\_dut in fsc[tank\_name][**"virtual\_duts\_table"**]:  
 liters = list(virtual\_dut.V\_DUT\_LITERS)  
 liters = [liters[0]] + liters + [liters[-1]]  
 dut\_values = list(virtual\_dut.DUT\_VALUES)  
 dut\_values = [dut\_values[0] - 1] + dut\_values + [dut\_values[-1] + 1]  
 df = pd.DataFrame({  
 **"values"**: dut\_values,  
 **"liters"**: liters,  
  
 })  
 with open(save\_path + **r"/"** + tank\_name + **r"/wialon formula.txt"**, **"w"**) as f:  
 f.write(fc.wialon\_formula[tank\_name])  
 df.to\_csv(save\_path + **r"/"** + tank\_name + **f"//V**{virtual\_dut.V\_DUT\_NUMBER}**.csv"**,  
 sep=**";"**, header=False, index=False)  
  
 Plotter().get\_fx(fsc, fc.latex\_formula\_full, save\_path=save\_path)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 process()

## Тестирование программного обеспечения

В случае некорректной настройки оборудования на стороне сервера возможно наблюдение ложных скачков уровня топлива. Возникновение таких ситуаций приведёт к тому, что специалисту сервисной службы придётся выезжать непосредственно к единице и проводить ряд мероприятий. Очень часто бывает, что техника находится в заметном удалении и ложный выезд обращается компании в убыток.

Таким образом, к корректности данного программного обеспечения предъявляются повышенные требования. В качестве тестовых данных послужили реальные тарировочные таблицы, накопленные в ходе исследования, которые проверяют следующий набор ситуаций (в скобках укажем марку транспортного средства). Тестовую ситуацию опишем в подписи к рисунку.

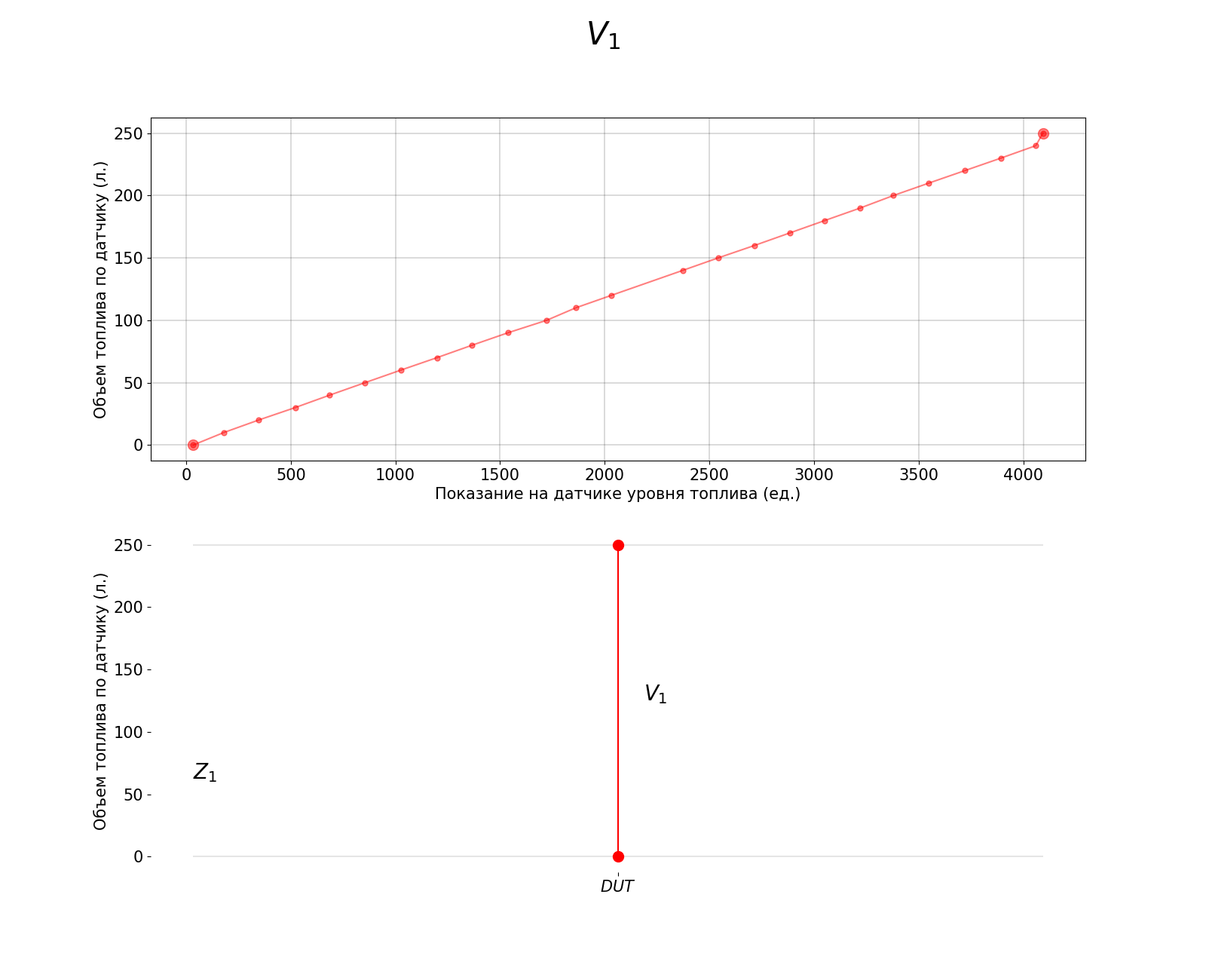


Рисунок 3.9 — Один датчик уровня топлива, один бак (Камаз)

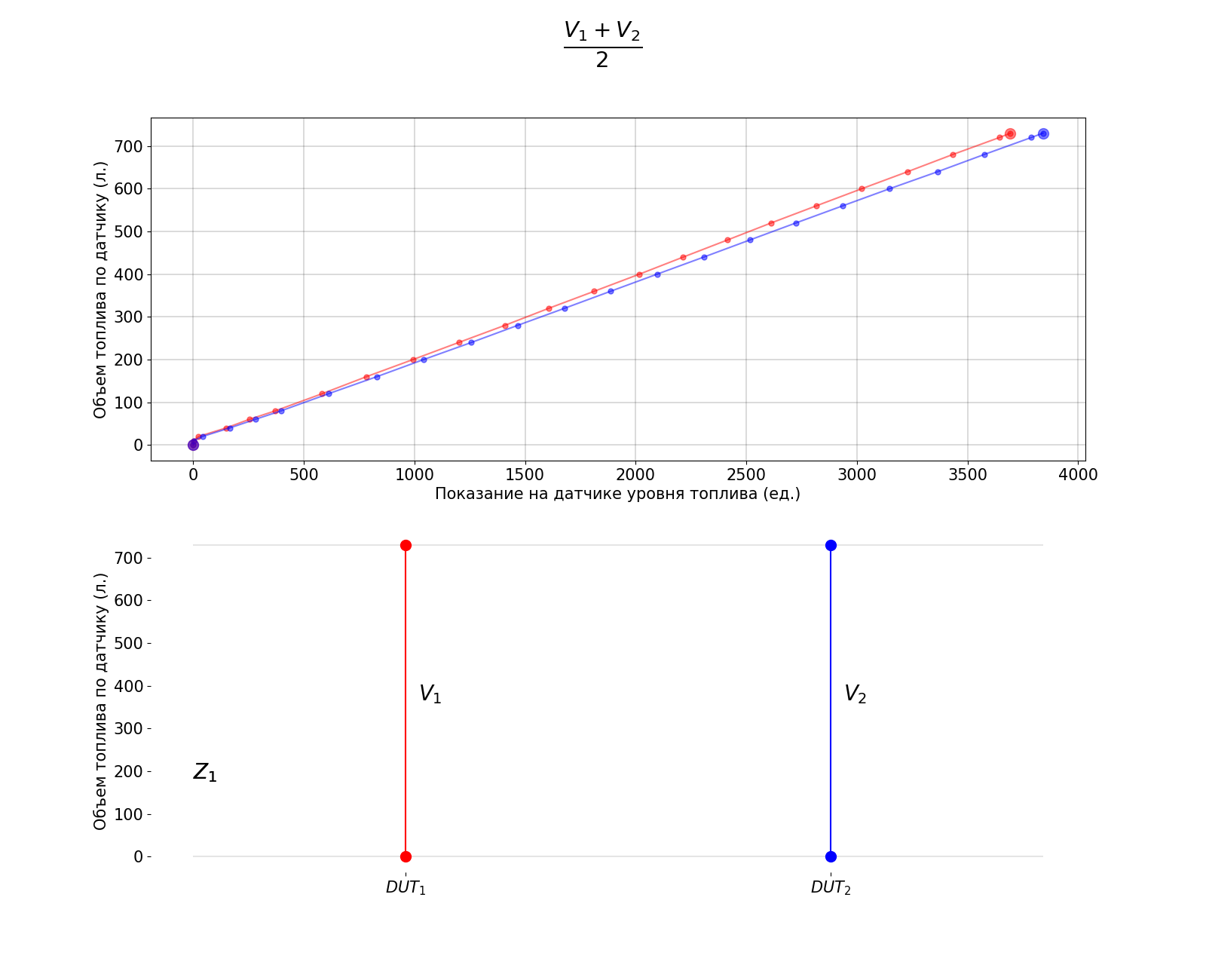


Рисунок 3.10 — Два датчика уровня топлива, установленных параллельно друг другу (MAN)

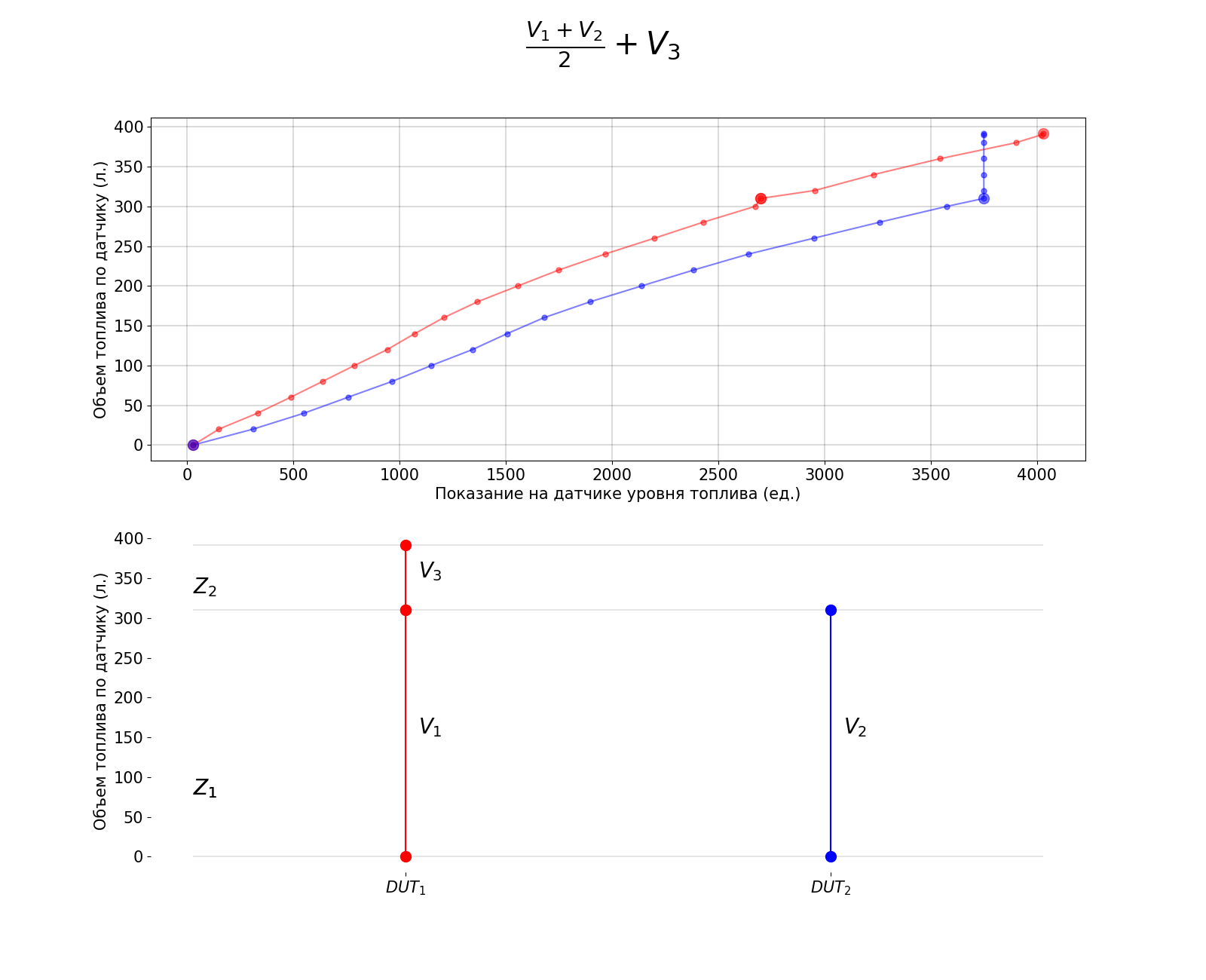


Рисунок 3.11 — Два датчика уровня топлива, один из которых не перекрывает верхнюю часть бака (Case)

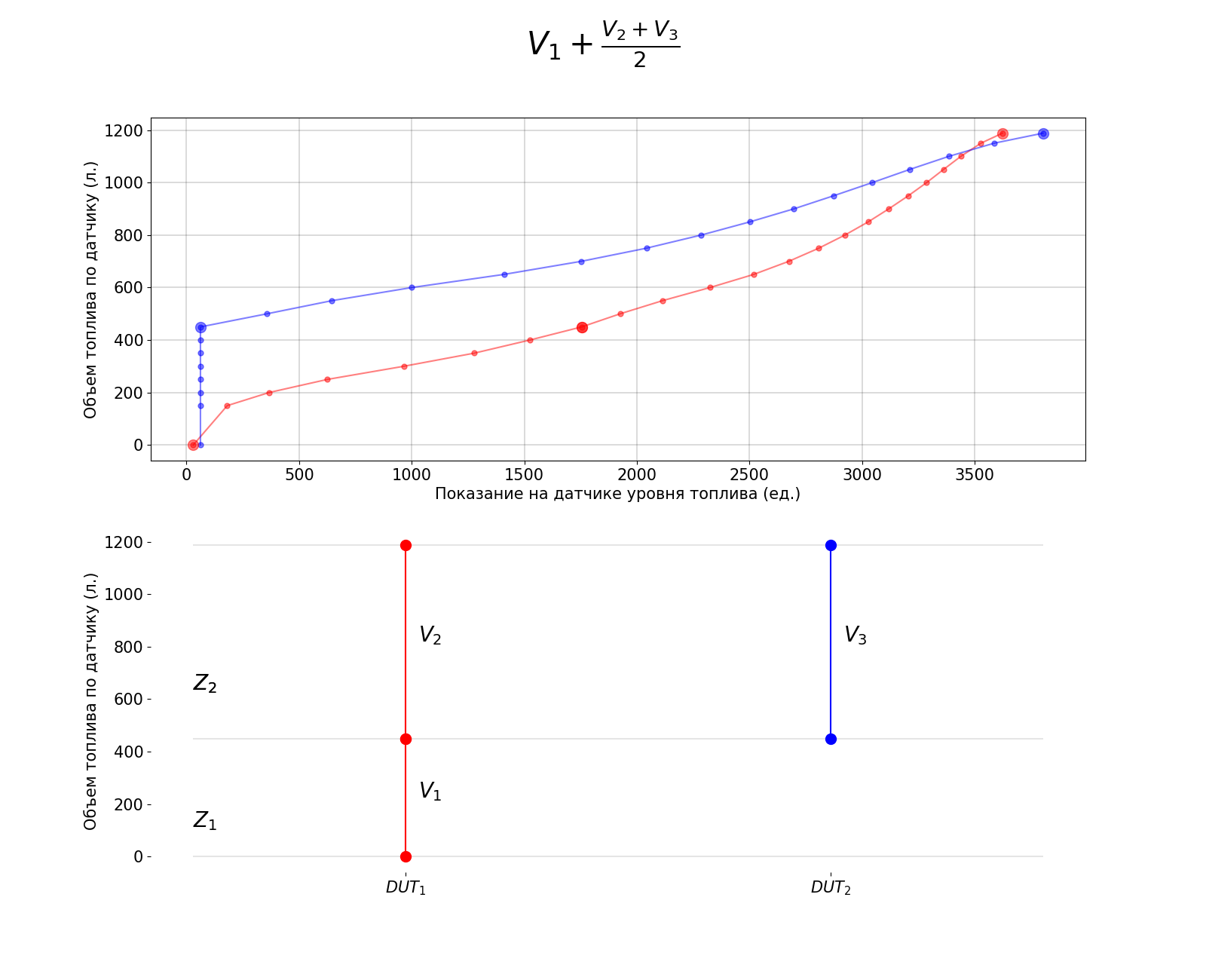


Рисунок 3.12 — Два датчика уровня топлива, один из которых не перекрывает нижнюю часть бака (Challenger)

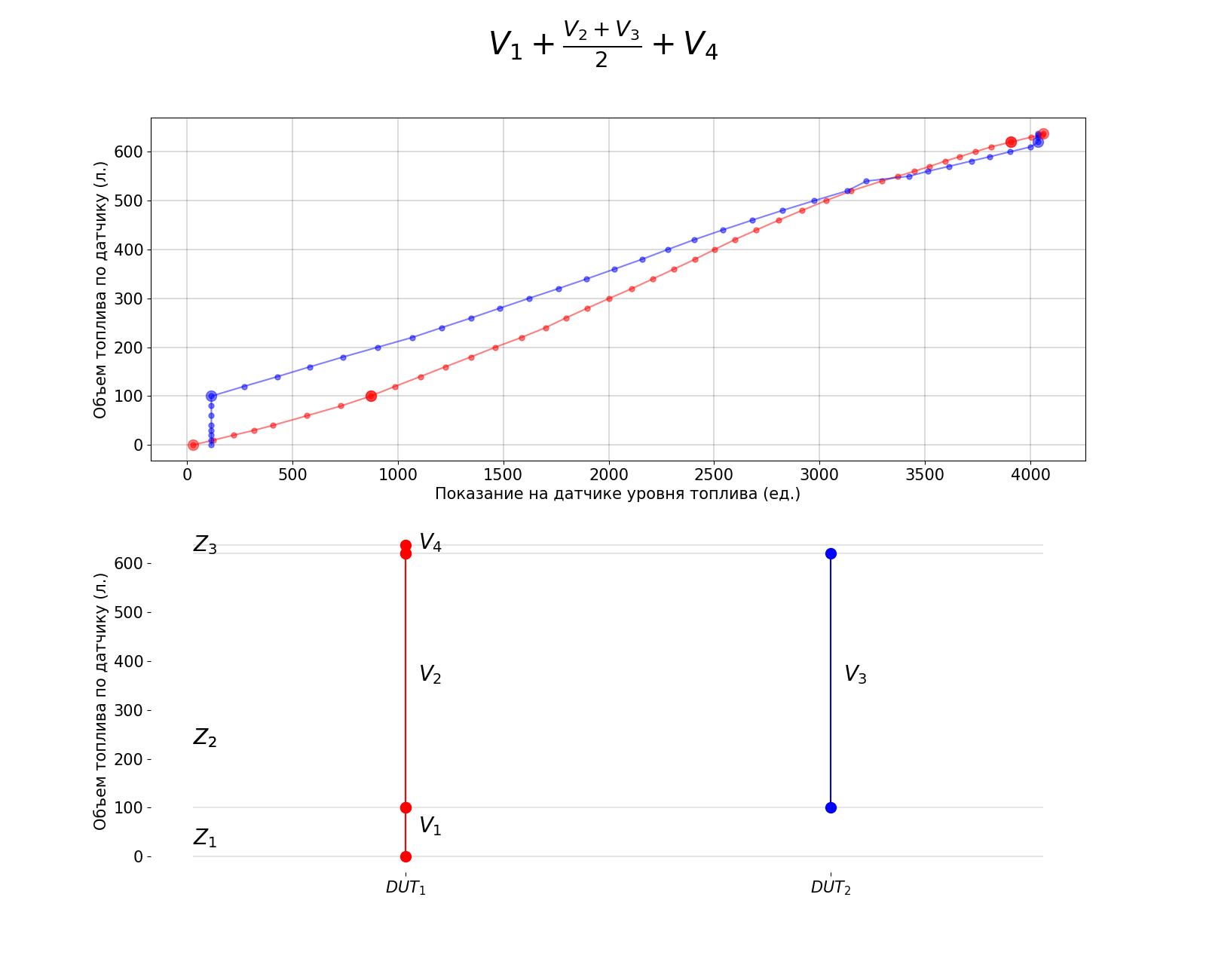


Рисунок 3.13 — Два датчика, один из которых покрывает всю высоту бака, второй – центральную часть бака

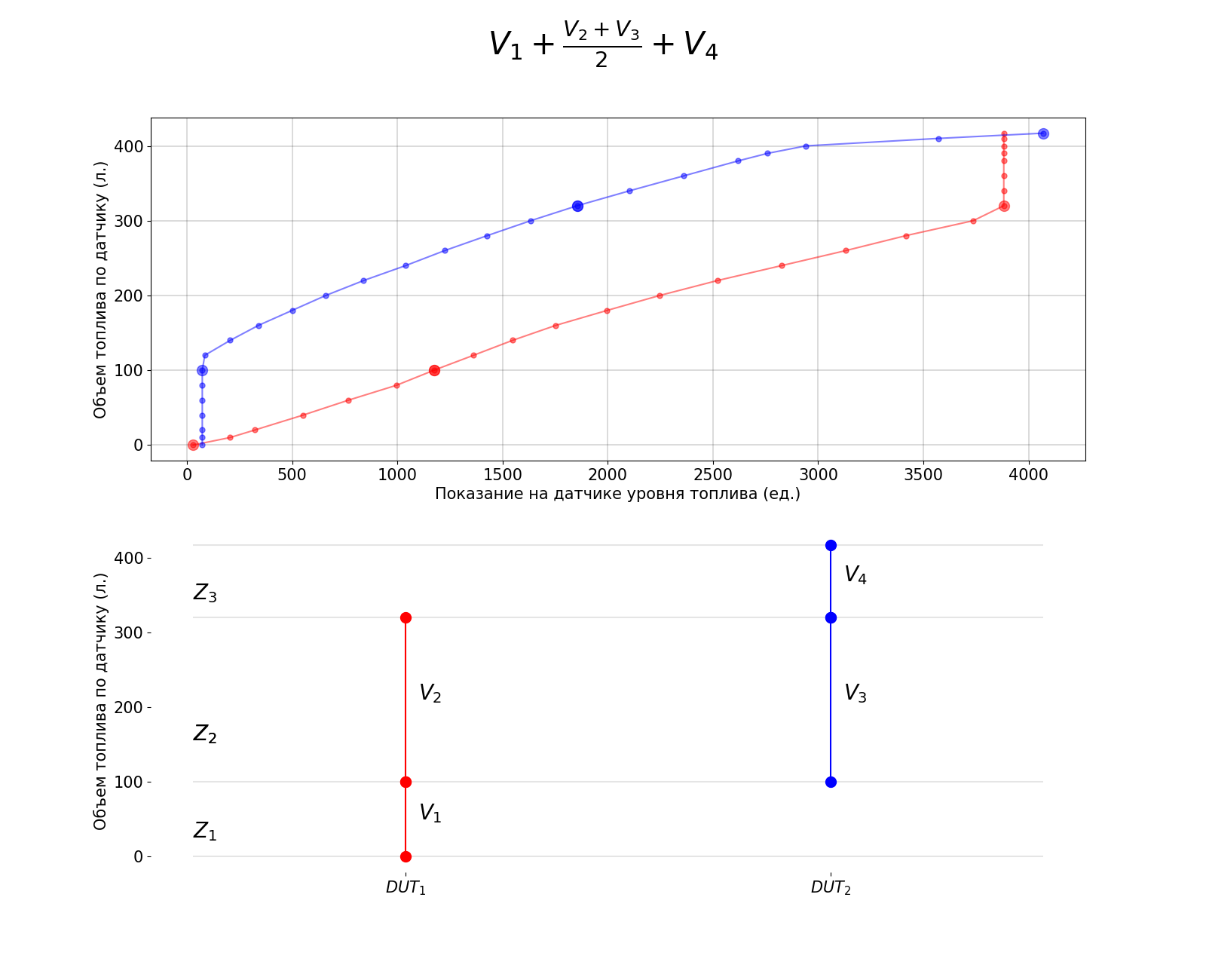


Рисунок 3.14 — Два датчика, один из которых покрывает нижнюю часть бака, но не покрывает верхнюю; второй датчик – перекрывает верхнюю часть, но не перекрывает нижнюю (New Holland)

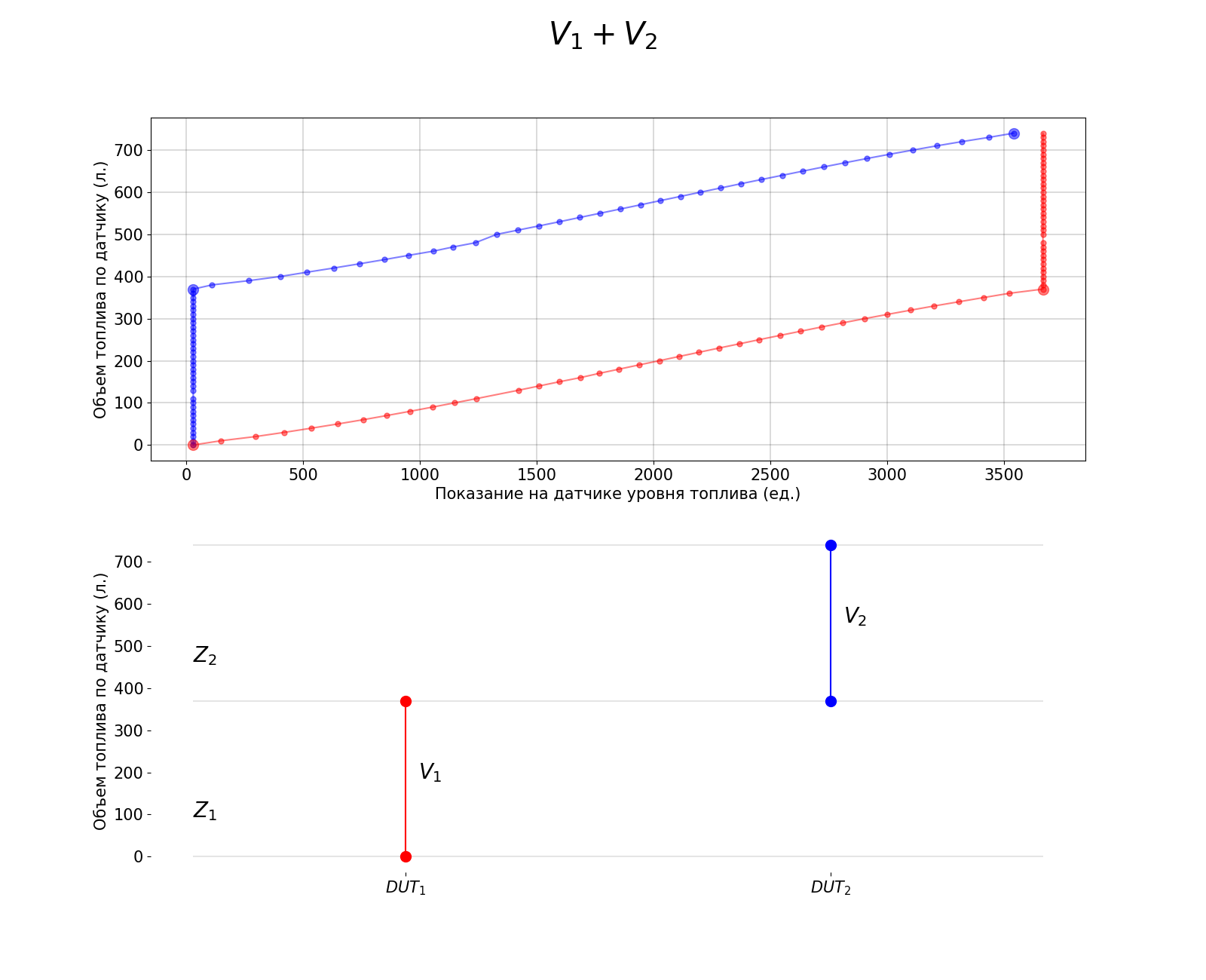


Рисунок 3.15 — Два датчика, показания второго начинаются по окончанию первого датчика[[2]](#footnote-2)

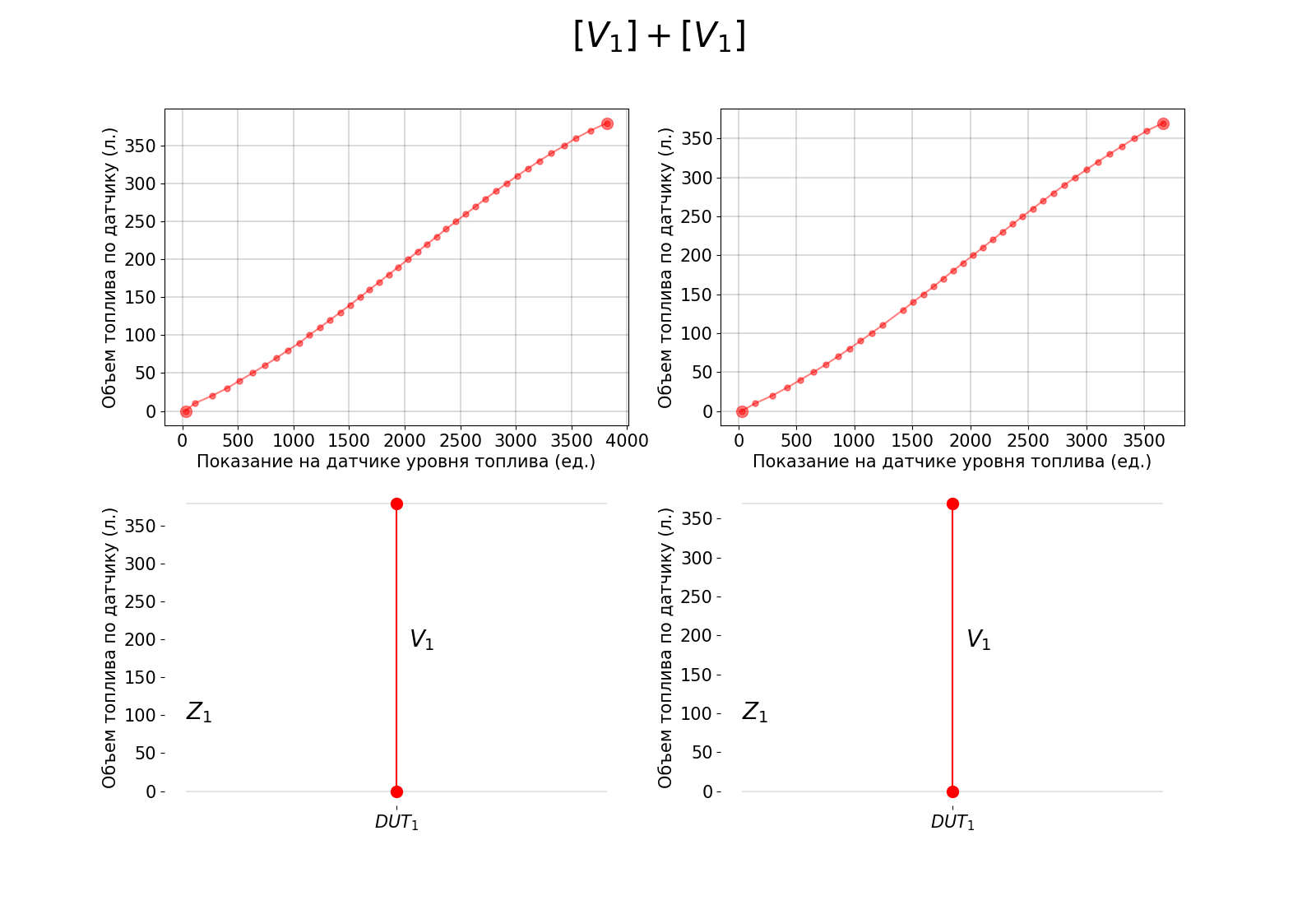


Рисунок 3.16 — Два датчика расположенных в разных баках (Freightliner)

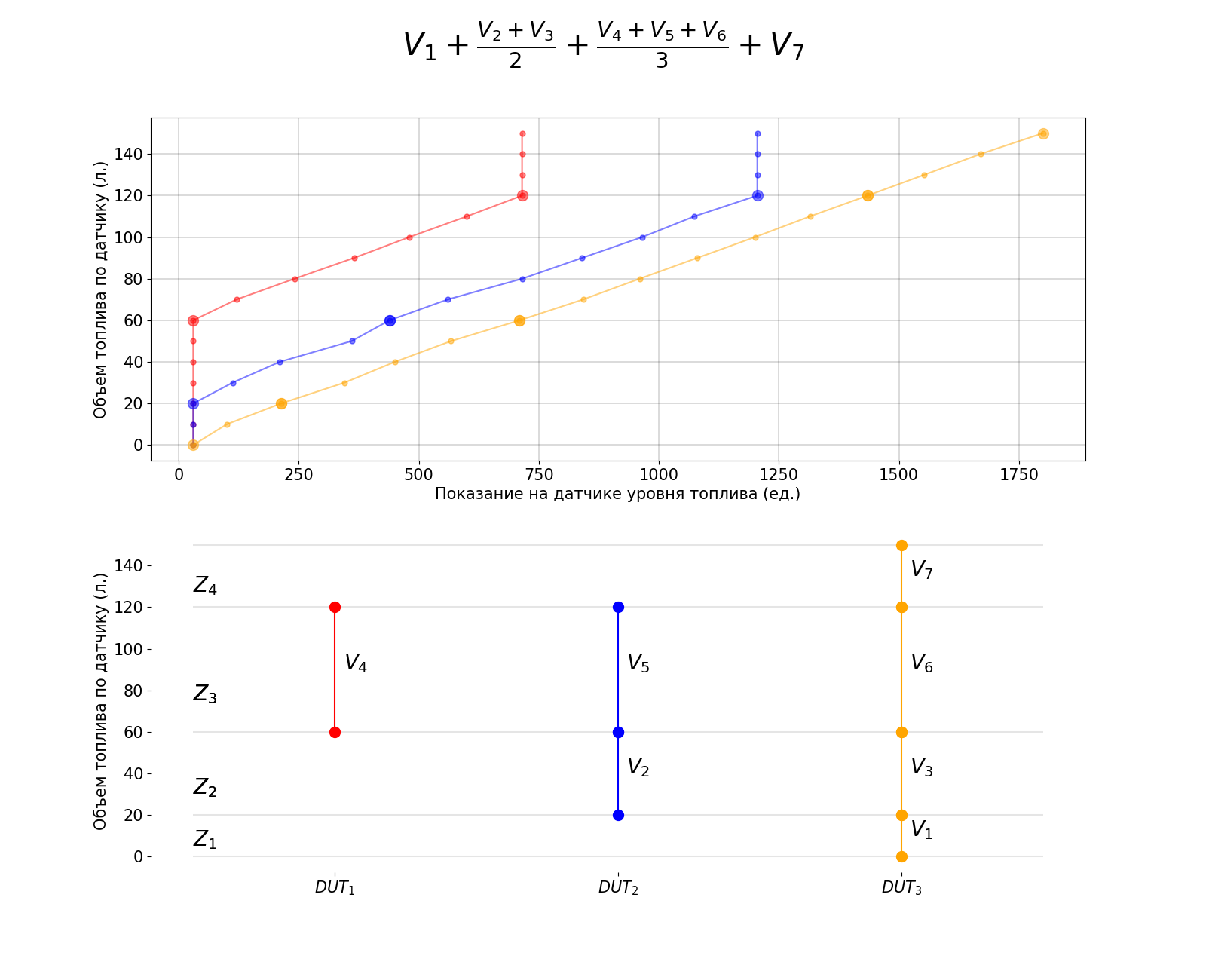


Рисунок 3.17 — Три датчика уровня топлива в одном баке (данные из таблицы 2.1)

## Установка программного обеспечения

Установка приложения для операционной системы windows состоит из следующих этапов:

* Установка интерпретатора языка программирования Python
* Установка необходимых пакетов
* Создание bat-файла для удобного запуска приложения

### Установка интерпретатора Python

Для установки интерпретатора Python необходимо перейти на официальный сайт Python: <https://www.python.org/downloads/> и скачать последнюю версию (рисунок 3.17):

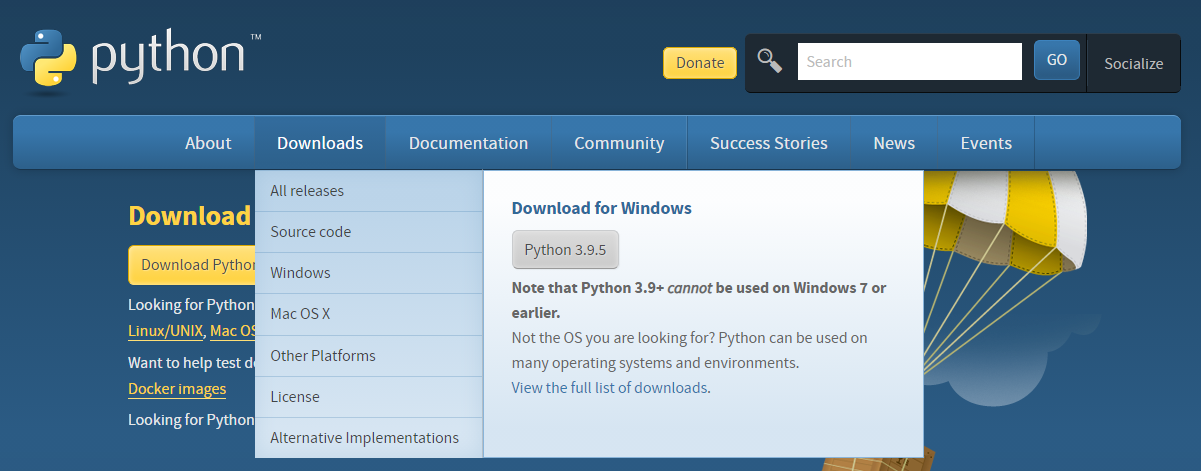


Рисунок 3.17 — Установка Python

### Установка пакетов

Для установки пакетов необходимо выполнить следующие действия:

1. Переход в папку Scripts
   1. Ввести в меню «Пуск» запрос Python;
   2. Нажать правой кнопкой мыши по ярлыку и выбрать «Свойства» (рисунок 3.18).

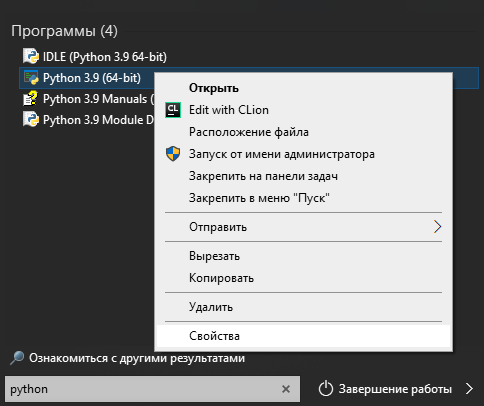


Рисунок 3.18 — Получение свойств

* 1. В появившемся диалоговом окне путь может быть найден (рисунок 3.19):

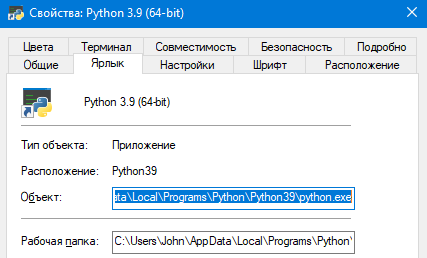


Рисунок 3.19 — Получение пути к файлу python.exe

* 1. Скопируйте весь путь за исключением “python.exe”. В примере выше будет получен путь

“C:\Users\John\AppData\Local\Programs\Python\Python39\”;

* 1. Откройте командную строку от имени администратора:
     1. Нажмите комбинацию клавиш Win+R;
     2. Введите в окне cmd и нажмите OK (рисунок 3.20);

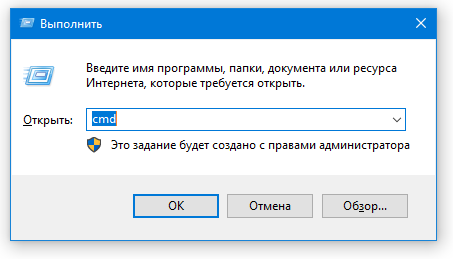


Рисунок 3.20 — Окно «Выполнить».

* 1. В консоли введите команду cd и вставьте скопированный на этапе d путь и выполните переход в папку Scripts (рисунок 3.21).

Для установки пакета необходимо ввести команду «pip install <имя пакета>»:



Рисунок 3.20 — Установка пакета pandas

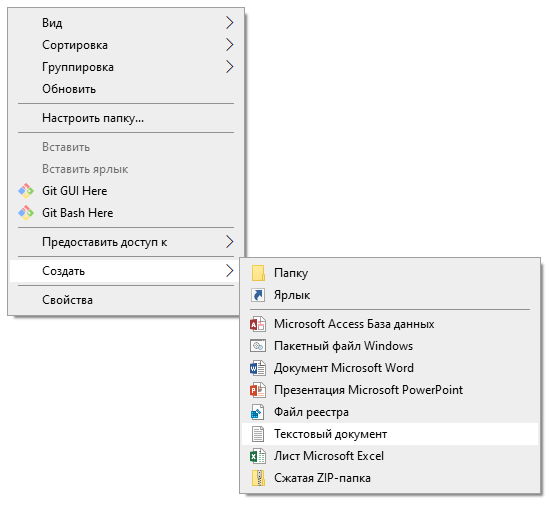
Для работы приложения необходимо установить следующие пакеты:

* pandas (12)
* matplotlib (13)
* xlrd (14)
* openpyxl (15)

### Создание bat-файла

Чтобы возможно было выполнять запуск приложения с ярлыка, необходимо:

* Создать в удобном месте файл с расширением bat (рисунок 3.21):



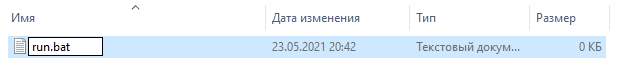


Рисунок 3.21 — Создание bat-файла

* Откройте файл для редактирования при помощи блокнота (рисунок 3.22):

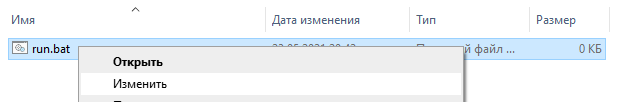


Рисунок 3.22 — Изменение bat-файла

* В качестве текста файла укажите два пути: первый путь – к интерпретатору Python python.exe, второй путь – путь к файлу main с архивом программы (рисунок 3.23):

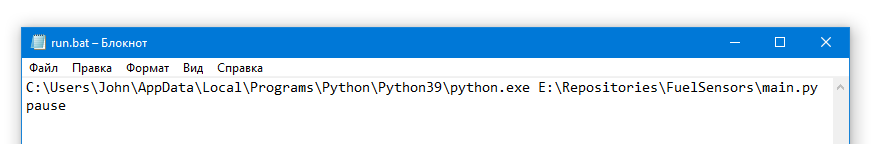


Рисунок 3.23 — Изменение bat-файла

После выполнения вышеперечисленных шагов, можно осуществлять запуск программы посредством двойного щелчка левой кнопки мыши по ярлыку.

24 мая 2021 года в лаборатории технической защиты информации было проведено инсталляционное тестирование на одной из предоставленных машин. Среди учетных записей пользователей не было найдено ни одной, которая бы не содержала русских букв. Пришлось осуществить запуск приложения из консоли указав необходимые пути.

Информация о машине, работе с консолью (установка пакетов и запуск приложения), результаты работы программы предоставлены на рисунке 3.24)

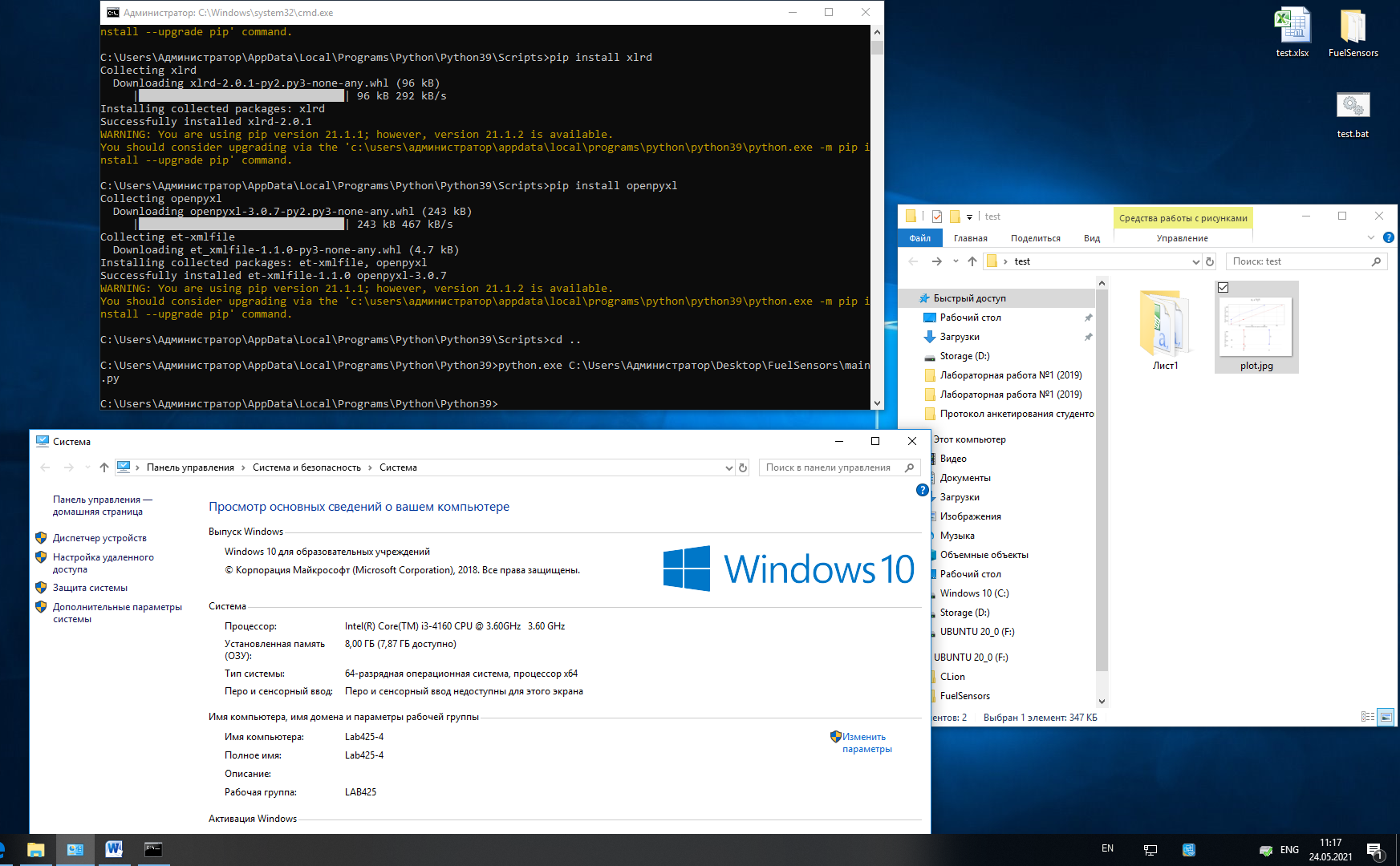


Рисунок 3.23 — Изменение bat-файла

### Процедура работы с системой спутникового мониторинга

После того, как тарировочная таблица была разбита на виртуальные датчики необходимо сделать следующее (на примере системы ):

* Выполнить поиск объекта в системе спутникового мониторинга и зайти в свойства объекта. И перейти вкладку «Датчики»

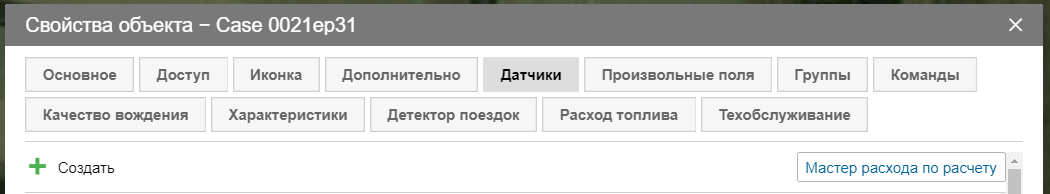


Рисунок 3.24 — Вкладка «Датчики» для объекта Case 0021ер31.

* В процессе установки ДУТов, тарировки или непосредственно из сообщений можно узнать, какой параметр отвечает за тот или иной датчик. В нашем случае имеется два ДУТа: один подключен на clls1, другой – на clls2.
* Выполняется создание датчиков в системе. Для этого необходимо нажать на кнопку «Создать» и заполнить поля следующим образом:

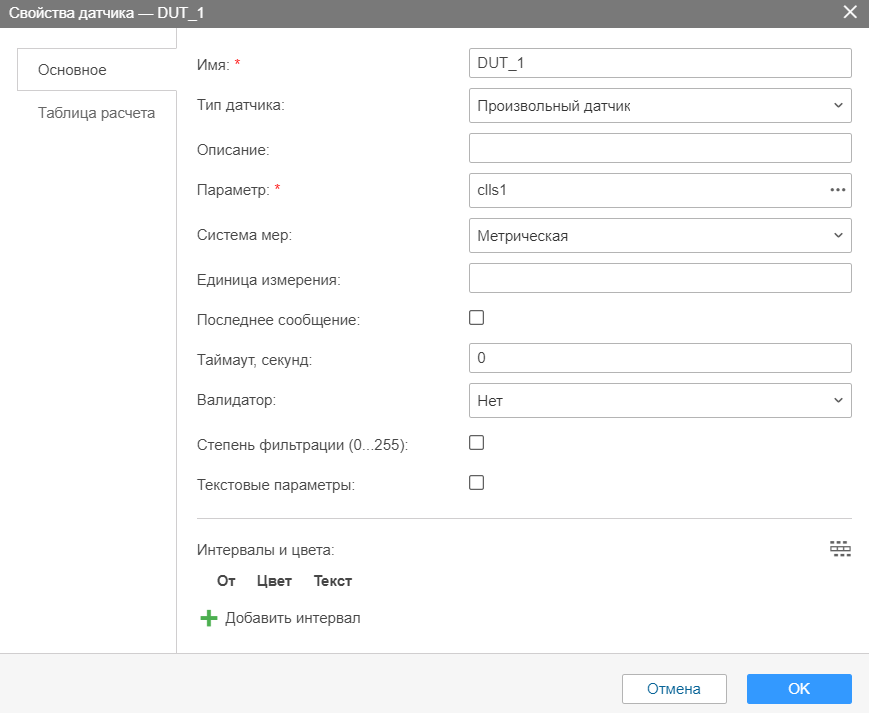


Рисунок 3.25 — Создание датчика

В процессе создания датчика ему нужно задать имя. Рекомендуется использовать , и т. д. по количеству ДУТов. В примере имеем два датчика. Поэтому после выполнения указанных шагов должны получить следующее (рисунок 3.26):

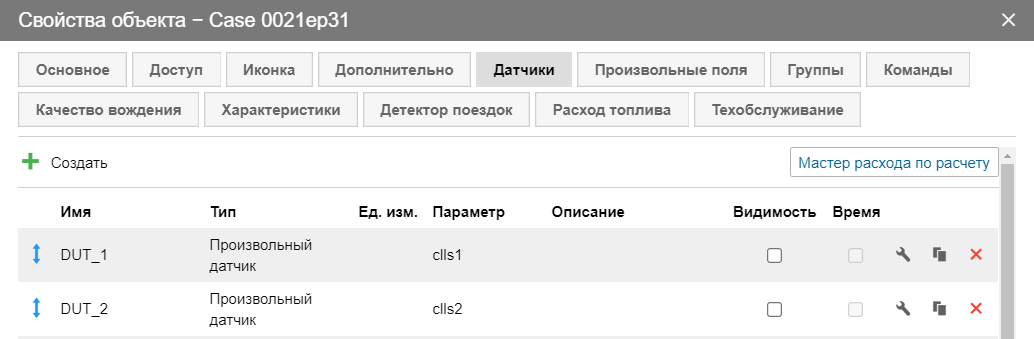


Рисунок 3.26 — Создание датчиков ,

* Занести тарировку в .xlsx-файл (рисунок 3.27):

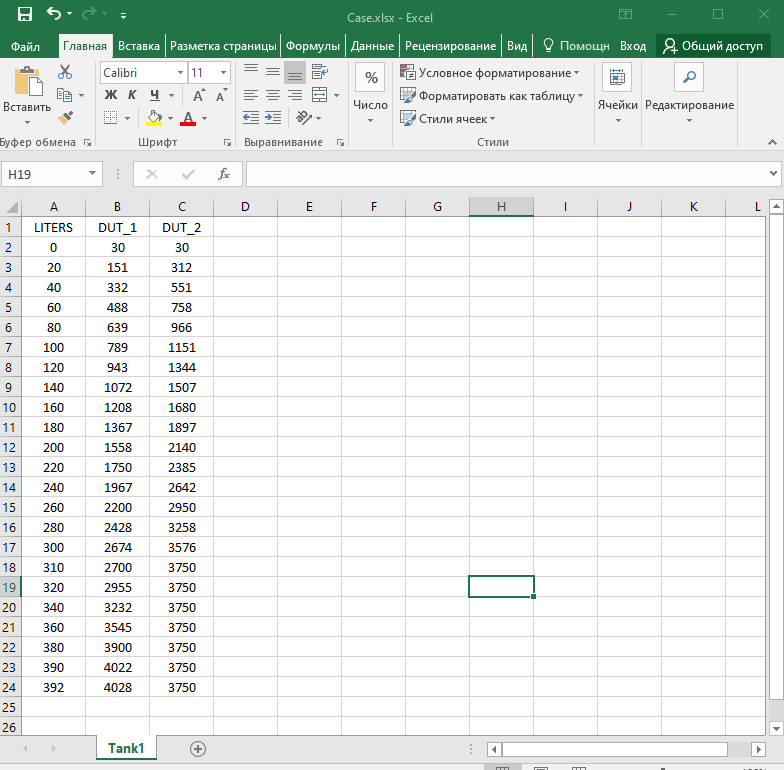


Рисунок 3.27 — Тарировочная таблица для Case 0021ер31

* Выполнить запуск разработанного приложения, который выполнит запрос файла. Необходимо указать файл с тарировочной таблицей.
* В той же директории, где и файл с тарировкой должна появиться одноимённая папка со следующим содержимым (рисунок 3.28):

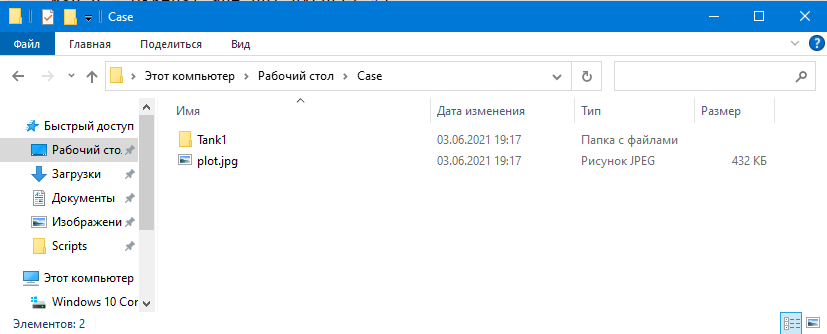


Рисунок 3.28 — Результат работы программы

При желании можно ознакомиться с получившимися функциональными зависимостями (рисунок 3.29):

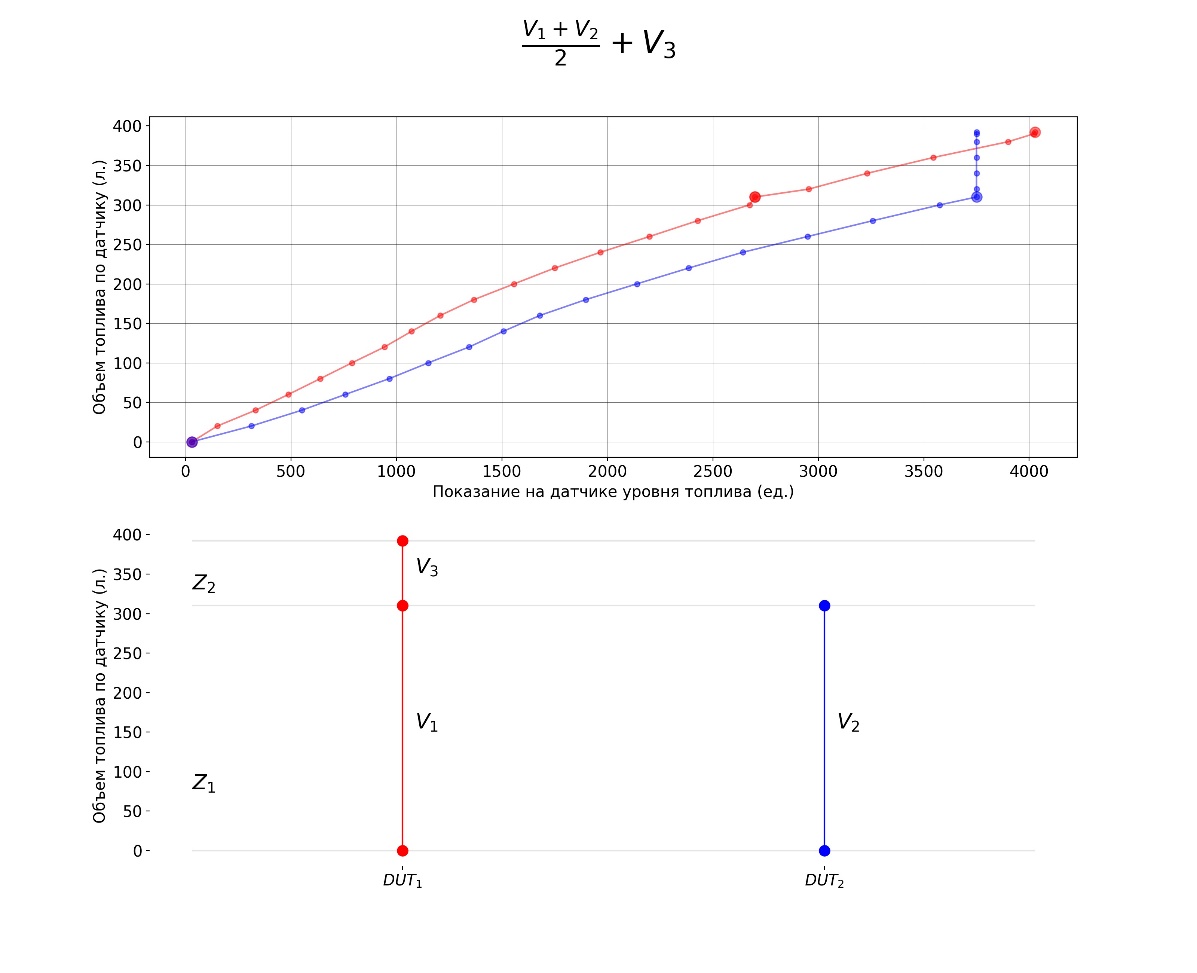


Рисунок 3.29 — Графики, построенные программой

* На следующем этапе необходимо создать виртуальные датчики. Исходя их графика, можно сказать, что имеется 3 виртуальных датчика. Опишем создание одного из них:

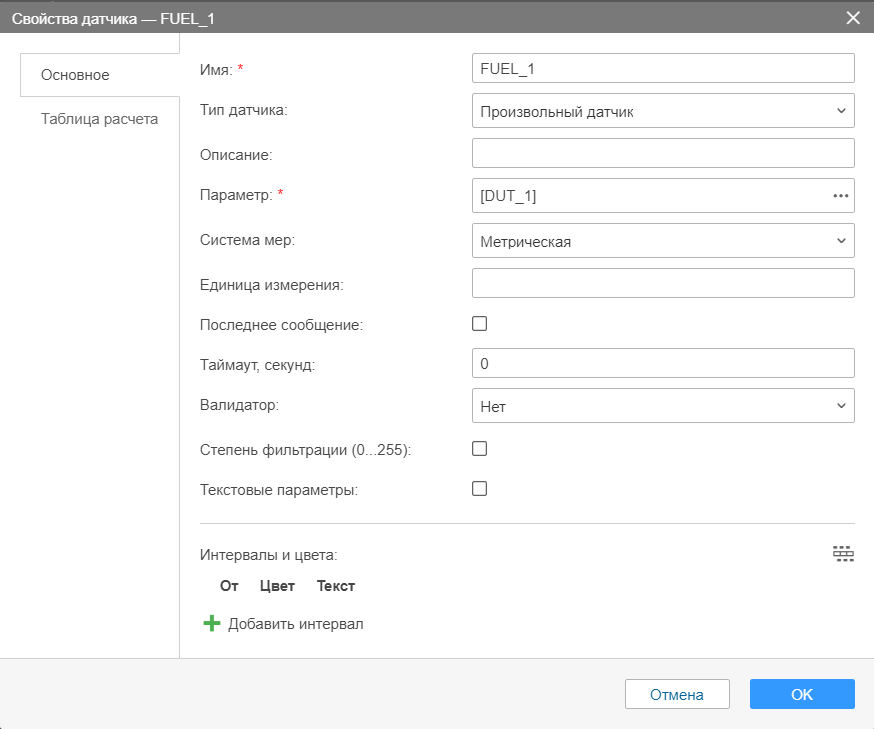


Рисунок 3.30 — Создание датчика

Особое внимание следует уделить следующим моментам:

* + Имя датчика: – имя датчика. Рекомендуется давать имена , , … .
  + В качестве параметра нужно указать имя физического ДУТа.
* На вкладке «Таблица расчета» необходимо выставить галочку в «Пары XY» а потом импортировать тарировочную таблицу с виртуальным датчиком (рисунок 3.31).

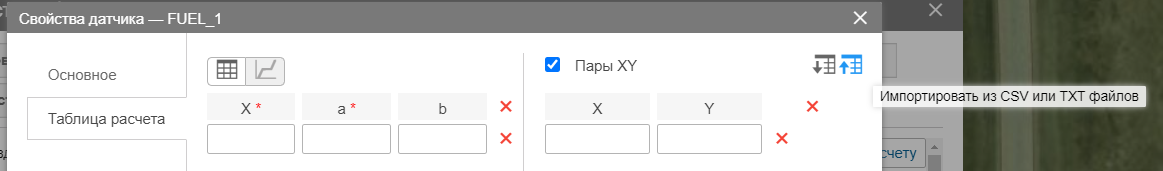


Рисунок 3.31 — Импортирование таблицы

* После импортирования необходимо нажать кнопку «Генерировать», после чего можно увидеть (рисунок 3.32):

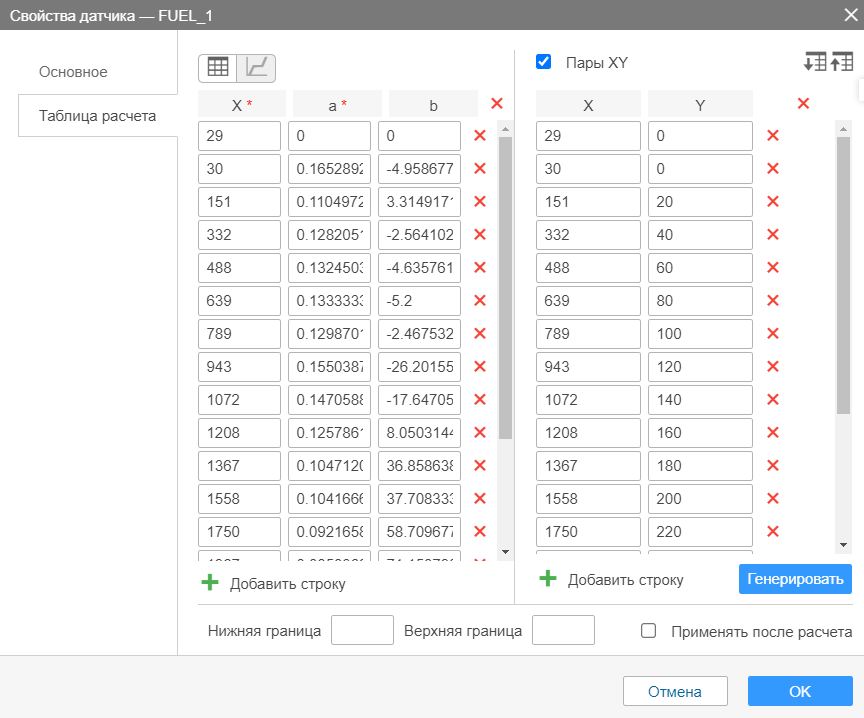


Рисунок 3.32 — Импортирование таблицы

* Если нажать на кнопку «Показать график» (рисунок 3.32), можно увидеть зависимость для виртуального ДУТа как и на рисунке 3.29, но до жирной красной точки. После неё идёт следующий виртуальный датчик уровня топлива.
* После занесения всех виртуальных датчиков, будем наблюдать следующую картину (рисунок 3.33):

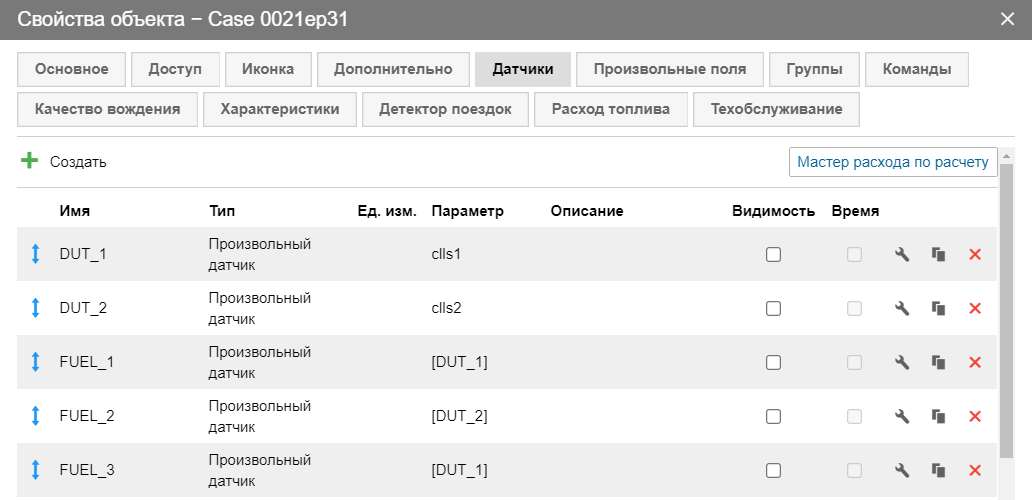
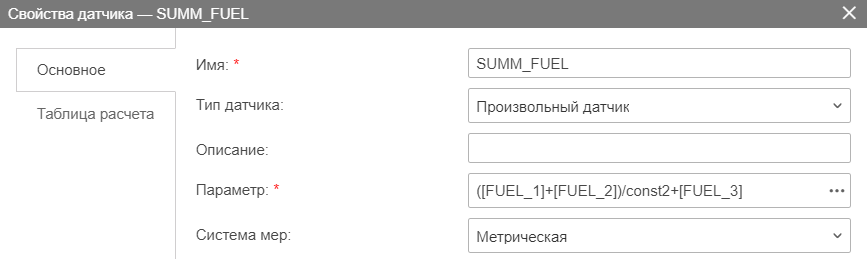


Рисунок 3.33 — Заведенные виртуальные датчики уровня топлива

* На последнем этапе создаются ещё два датчика. Программа генерирует текстовый файл, из которого можно скопировать формулу для данной системы спутникового мониторинга и использовать в поле параметр (рисунок 3.34).



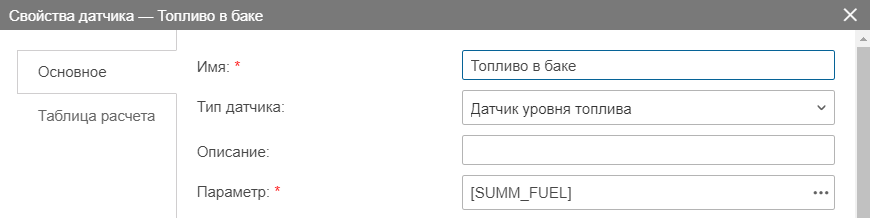


Рисунок 3.34 — Заведенные виртуальные датчики уровня топлива

Для датчика уровня топлива необходимо выставить также дополнительные настройки:

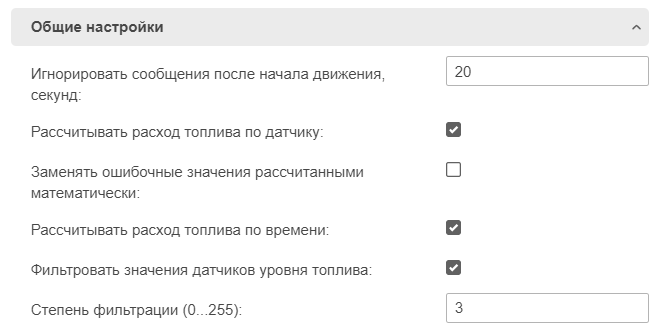


Рисунок 3.34 — Дополнительные настройки датчиков уровня топлива

Итоговая конфигурация представлена на рисунке 3.35:

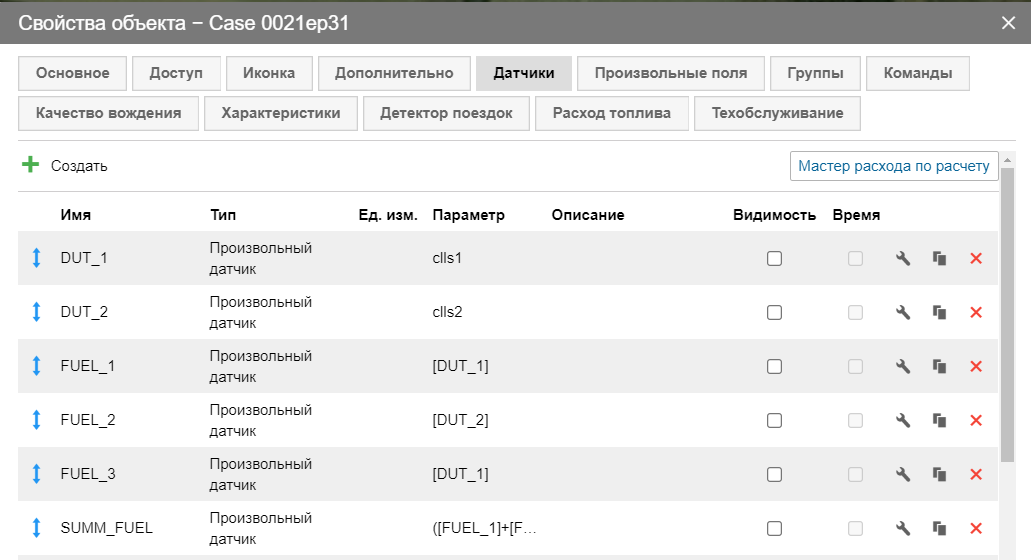


Рисунок 3.35 — Итоговая конфигурация

* После проведения настроек объекта рекомендуется выполнить построение отчёта по топливу с целью дополнительной проверки корректности настроек (фрагмент примера отчёта представлен на рисунке 3.36).

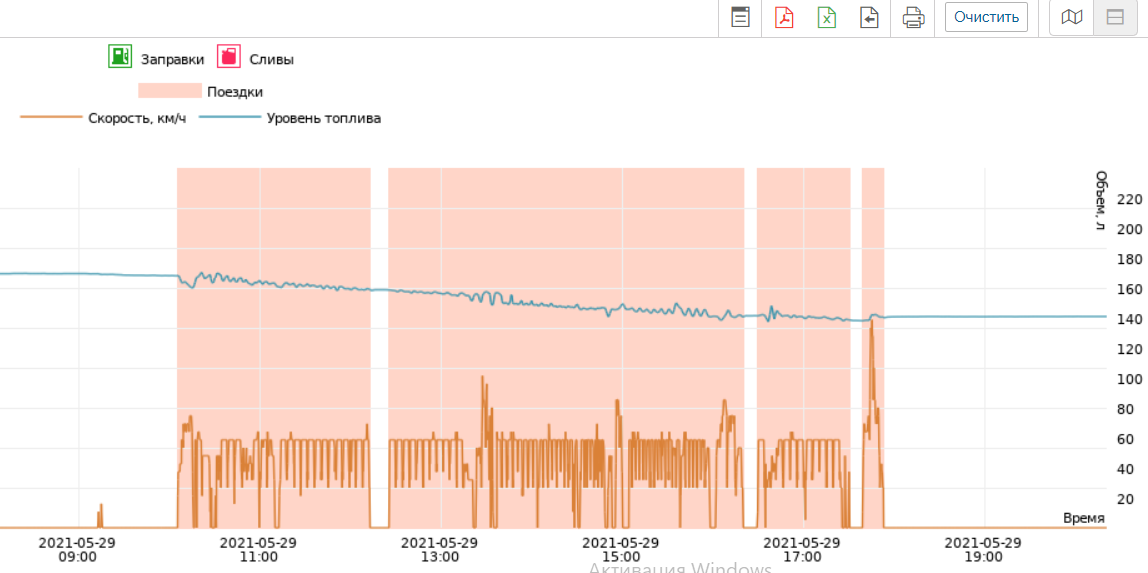


Рисунок 3.36 — Фрагмент отчёта

При правильно проведенной настройке объекта не должно наблюдаться резких скачков показаний уровня топлива, что мы и наблюдаем в данном случае.

## Вывод

В третьем разделе представлена реализации алгоритма, приведены результаты тестирования, а также сведения по установке ПО, и эксплуатации результатов, вкупе с системой спутникового мониторинга Wialon.

# Заключение

В рамках выпускной квалификационной работы по теме «Разработка и реализация алгоритма вычисления уровня топлива в баках сложной конфигурации» было выполнено исследование предметной области, описаны требования к алгоритму вычисления уровня топлива со стороны сервера.

В ходе работы был предложен алгоритм вычисления уровня топлива посредством разбиения физических датчиков уровня топлива на виртуальные (логические). Выработаны функциональные и нефункциональные требования к программному обеспечению, производящему разбиения, а также для построения графических зависимостей.

Представлена реализация алгоритма на языке программирования Python с использованием библиотек pandas, matplotlib, xlrd, openpyxl. Было проведено модульное, инсталляционное, эксплуатационное приёмочное тестирование разработанного продукта, которое показало корректность на различных исходных данных. Разработанное программное обеспечение внедрено в ООО «ЭкспертКом». Планируется дальнейшее использование разработанных алгоритмов и подходов для разработки станции для автоматизированного тарирования с передачей настроек на сервер.

# Список литературы

1. **Николаев, Р. А.** Погрешность измерения уровня топлива, вызванная наклонами автомобиля. [ред.] Родионов В. В. *Информационные системы и измерительно-вычислительные комплексы. Сборник докладов студентов и аспирантов кафедры «Измерительно-вычислительные комплексы» на научно-технических конференциях.* 2010 г., стр. 44-48.

2. **Навтелеком.** Руководство по экплуатации СМАРТ S-2433. Установка и подключение устройства. Москва : б.н., 2020 г.

3. **Omnicomm.** Датчики уровня топлива Omnicomm LLS 5 и LLS-Ex 5. *Руководство пользователя.* 2021 г.

4. **Анакшин А. А., Чулючкин В. В., Акчурин Н. Г.** *Устройство для измерения уровня топлива. RU 2445585 C1* 2012 г. патент на изобретение.

5. *Труды международного симпозиума "Надежность и качество".* **Мурашкина Т.И., Архипов А.В., Пивкин А.Г., Серебряков Д.И.** Пенза : б.н., 2012. Волоконно-оптическая система измерения уровня топлива. стр. 15.

6. **Л.В., Кузнецова.** *Герконовый датчик для измерения уровня топлива в баке. RU 152949 U1* Россия, 2015 г.

7. **А.Г., Медведев.** Разработка и исследование поверхностных емкостных датчиков для измерения уровня топлива. 2008 г.

8. *Физико-математическая модель емкостного метода измерения уровня топлива в баках ракет-носителей с учётом полей рассеивания.* **Захаров Р.С., Скворцов Б.В., Таипова Д.Р.** б.м. : Самарский национальный исследовательский университет имени академика С.П. Королёва.

9. *7-я международная научно-техническая конференция "К. Э. Циолковский - 160 лет со дня рождения". Космонавтика. Радиоэлектроника. Геоинформатика.* **Муравьёв С.А., Новиков С.В.** Повышение точности измерений уровня заправки жидкостных компонентов ракетного топлива.

10. **Куликов, Святослав.** *Тестирование программного обеспечения.* Минск : Четыре четверти, 2017. ISBN 978-985-581-125-2..

11. **Wiegers, Karl E.** *Software Requirements.* Москва : Русская редакция, 2004. ISBN 5-7502-0240-2.

12. **Team, Wes McKinney and the Pandas Development.** Pandas: powerful Python data analysis. *pandas: powerful Python data analysis.* April 12, 2021.

13. **John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the matplotlib development team.** Matplotlib. 2021 г.

14. **Machin, Stephen John.** Xlrd. *xlrd.readthedocs.io.* [Online] [Cited: May 23, 2021.] https://xlrd.readthedocs.io/en/latest/.

15. **Eric Gazoni, Charlie Clark.** A Python library to read/write Excel 2010 xlsx/xlsm files. *openpyxl.readthedocs.io.* [В Интернете] 09 March 2021 г. [Цитировано: 23 May 2021 г.] https://openpyxl.readthedocs.io/en/stable/.

1. Как правило, у клиентов не имеется необходимости отслеживать баки раздельно: их интересует суммарный объем топлива [↑](#footnote-ref-1)
2. Данная конфигурация бака встречается крайне редко, на момент написания работы не представилось возможным найти реальные исторические данные по такому типу объектов. [↑](#footnote-ref-2)