Министерство образования и науки Российской федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
доцент
К.А. Домбровский
_____2017 г.

Разработка программного комплекса для конфигурирования и мониторинга контрольно-измерительных приборов, поддерживающих работу с HART-протоколом

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2017.382 ПЗ ВКР

Руководитель работы,
доцент
Ю.Г. Плаксина
_____2017 г.

Авторы работы
студенты группы КЭ-445
М.П. Вихирев
И.А. Пестов
_____2017 г.

Нормоконтроллер,
старший преподаватель
В.В. Лурье
_____2017 г.

Челябинск 2017

АННОТАЦИЯ

Вихирев М.П., Пестов И.А. Разработка программного комплекса для конфигурирования и мониторинга контрольно-измерительных приборов, поддерживающих работу с HART-протоколом. – Челябинск: ЮУрГУ, КЭ-445, 61 с., 10 илл., библиогр. список – 13 наим.

В выпускной квалификационной работе представлена разработка программного комплекса для конфигурирования и мониторинга контрольно-измерительных приборов, поддерживающих работу с HART-протоколом.

В выпускной квалификационной работе проанализированы аналогичные разработки, представленные в настоящий момент на рынке, изучена актуальная техническая документация, содержащая общие сведения о принципах работы с HART-устройствами.

Проведено обоснование выбора тех или иных инструментов, используемых при разработке программной системы. Разработана архитектура системы, учитывающая недостатки существующих аналогов. Создан прототип программной системы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. В настоящее время на промышленных предприятиях широко используются контрольно–измерительные приборы. На сегодняшний день они представляют собой микропроцессорные интеллектуальные устройства, способные к самодиагностике, поиску ошибок, тонкой настройке, записи показаний и оснащенные цифровым протоколом передачи данных. Одним из самых распространенных протоколов в мире является HART-протокол. С его помощью можно осуществлять взаимодействие с контрольно-измерительными приборами, используя специальное программное обеспечение (ПО). На данный момент на рынке имеется лишь небольшое количество подобных программ, обладающих определенными недостатками (к основным недостаткам можно отнести высокую стоимость, ограниченный набор поддерживаемых устройств, возможность работы лишь под операционными системами семейства Windows). Таким образом, создание программы, предоставляющей оператору возможность взаимодействия с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом, является актуальной задачей.

Цель работы - разработка программного продукта, предназначенного для предоставления оператору возможности взаимодействия с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом.

Задачи работы:

- провести анализ наиболее широко используемых программных продуктов, предназначенных для работы с HART-устройствами;
- определить основные подходы к разработке подобного программного обеспечения путем изучения литературы по теме выпускной квалификационной работы;
- разработать архитектуру разрабатываемой программы;
- разработать требования к разрабатываемой программе;
- создать прототип программы.

Результаты работы рекомендуется использовать при подготовке специалистов, профессиональная деятельность которых связана с работой с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом.

1 СРАВНЕНИЕ ОТЕЧЕСТВЕННЫХ И ПЕРЕДОВЫХ ЗАРУБЕЖНЫХ ТЕХНОЛОГИЙ И РЕШЕНИЙ

К наиболее популярным программам, используемым для работы с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом, на сегодняшний день относятся AMS Device Manager, PACTware, и HART Config.

1.1 AMS Device Manager

Согласно [11] программный комплекс AMS Suite: Intelligent Device Manager (далее AMSDeviceManager) производства компании Emerson Process Management – пакет программного обеспечения, предназначенный для удаленного конфигурирования микропроцессорных КИП на предприятии, непрерывной диагностики всех подключенных приборов, автоматизации операций по поверке и калибровке измерительных приборов, а также для документирования всех перечисленных операций.

Основные преимущества:

- поддержка полевых приборов с цифровыми протоколами HART, Wireless HART, Foundation Fieldbus, Profibus DP/PA;
- непрерывная диагностика позволяет прогнозировать состояние КИПиА, предотвращая возможные аварии;
- широкий перечень коммуникационных интерфейсов позволяет использовать AMSDeviceManager в составе любой системы управления верхнего уровня и при любой конфигурации аппаратных средств;
- высокий уровень безопасности достигается путем разграничения доступа к функциям системы.

Поддерживает только операционные системы семейства Windows. Стоимость покупки от 10 000$ до 30 000$.

1.2 PACTware

Согласно [12] PACTware (Process Automation Configuration Tool) представляет собой программное обеспечение для всех приборов и устройств VEGA и многих других производителей. Программные модули настройки конкретных прибо-ров называются менеджерами типа устройства (Device Type Manager) или сокращенно - DTM. DTM описывают аппаратную функциональность и обеспечивают все возможности настройки приборов.

Программа распростаняется бесплатно, но за каждый поддерживаемый файл DTM необходимо платить (порядка 1000$ за устройство).

Поддерживает только операционные системы семейства Windows.

1.3 HART Config

Согласно [13] программа используется для настройки и контроля приборов, поддерживающих HART-протокол. В силу универсальности протокола есть возможность работать с любыми HART-приборами, однако достоверная идентификация и максимальный набор функций возможен только для известных программе устройств.

Использование программы позволяет упростить и ускорить процесс конфигурирования приборов и сбора информации.

В рамках данной программы возможно:

- определение типов и параметров подключённых к компьютеру приборов (сеть приборов);
- считывание значений каналов приборов;
- визуальный просмотр данных в графическом виде;
- сохранение считанных данных в файл;
- задание количества точек для сохранения и отображения;
- сохранение настроек приборов в отдельных файлах для последующего использования;
- считывание и запись параметров устройств.

Поддерживаемые приборы:

- АИР-10Н — датчик давления;
- ЭЛЕМЕР-АИР-30 — датчик давления;
- ЭЛЕМЕР-100 — датчик давления;
- САПФИР-22ЕМ — датчик давления;
- ИП 0304/М1-Н — измерительный преобразователь для ТПУ 0304/М1-Н.

Поддерживает только операционные системы семейства Windows. Распространяется бесплатно.

Выводы по разделу один

Таким образом, можно выделить следующие основные недостатки вышеперечисленных программных продуктов:

1) Приложения поддерживают работу только под операционными системами семейства Windows.

2) Приложения поддерживают работу лишь с ограниченным набором контрольно-измерительных приборов.

3) Высокая стоимость приложений.

К достоинству данных программ можно отнести поддержку широкого набора функций для работы с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом.

2 ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор литературы по теме выпускной квалификационной работы

Основным источником информации о разработке прикладного программного обеспечения, используемого для работы с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом, послужила техническая документация, посвященная данному протоколу.

Согласно [1] контрольно-измерительные приборы подключаются к специальному устройству, называемому HART модем, который в свою очередь подключается к персональному компьютеру оператора через последовательный порт RS-232 или USB-порт. Типовая схема подключения, приведенная в [1], показана на Рисунок 1.

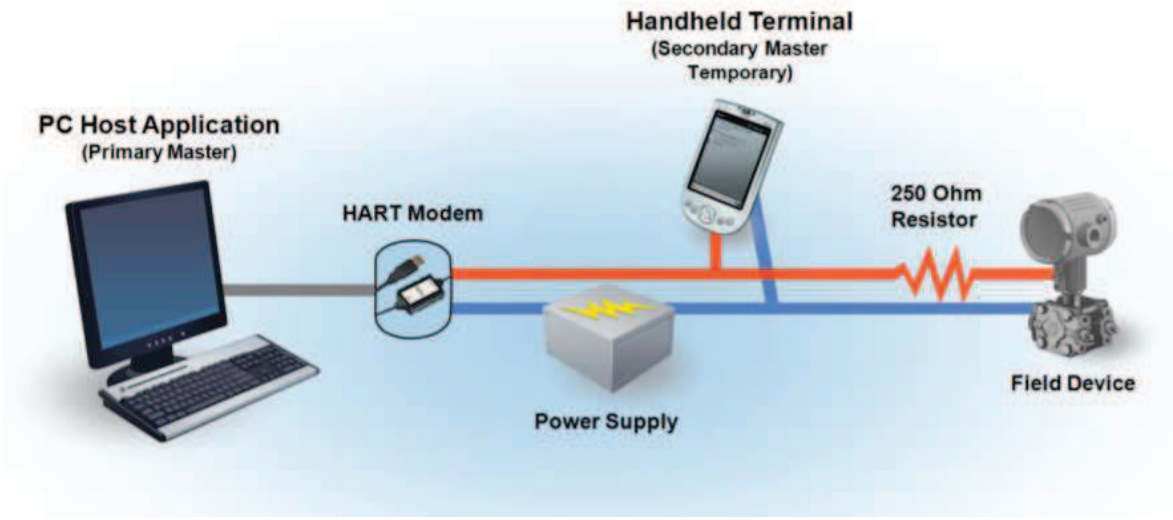

Рисунок 1 - Типовая схема подключения КИП

Также в [1] указано, что программы, применяемые для работы с контрольно-измерительными приборами, используют Device Description файлы (далее по тексту – DD-файлы) – файлы, содержащие всю необходимую программе информацию о HART-устройствах, записанную на специальном языке, называемом Device Description Language. Для каждого HART-устройства DD-файл создается его производителем.

Таким образом, разрабатываемая программа должна:

1) Использовать для обмена данными с контрольно-измерительными приборами последовательный порт RS-232 или USB-порт.

2) Поддерживать работу с DD-файлами.

2.2 Обзор литературы по обоснованию выбора технологий программирования

В качестве основного источника информации о технологиях программирования выступила литература, описывающая паттерны проектирования, применямые в объектно-ориентированном программировании.

В ходе анализа [2] и было обнаружено описание паттерна Model-View-Controller. Согласно [2], данный паттерн предполагает отделение модели объекта от его экранного представления, устанавливая при этом между ними протокол взаимодействия «подписка/оповещение» (т. е. любое изменение модели должно приводить к изменению экранного представления объекта).

Использование DD-файлов, описанных в пункте 2.1, позволяет программе строить графический пользовательский интерфейс для работы с любым контрольно-измерительным прибором на основании его описания, избавляя тем самым разработчика от необходимости создавать отдельный GUI для каждого поддерживаемого HART-устройства (необходимость предоставления пользователю различных интерфейсов связана с тем, что каждый контрольно-измерительный прибор имеет специфические для него функции).

Приведенная выше информация позволяет сделать вывод об оправданности использования паттерна Model-View-Controller в разрабатываемой программной системе: возможно построение модели контрольно-измерительного прибора по его описанию, приведенному в соответствующем DD-файле, и дальнейшее создание его экранного представления по данной модели.

Также применение DD-файлов вызывает проблемы, связанные с хранением и обновлением большого объема данных. Для их решения возможно применение в программной системе клиент-серверной архитектуры: клиентская часть будет использоваться для обеспечения взаимодействия между пользователем и контрольно-измерительными приборами, в то время как серверная – для хранения информации о HART-устройствах (добавление информации о новых устройствах или обновление информации об уже существующих предполагает обработку DD-

файлов. Данная операция также будет входить в компетенцию серверной части системы).

В качестве основного источника информации о клиент-серверной архитектуре выступила литература, посвященная распределенным вычислительным системам. Варианты организации клиент-серверной архитектуры, описываемые в [3] приведены на Рисунок 2. Так как предполагается обработка данных как на серверной, так и на клиентской части, можно говорить об использовании в разрабатываемой программной системе варианта организации «в».

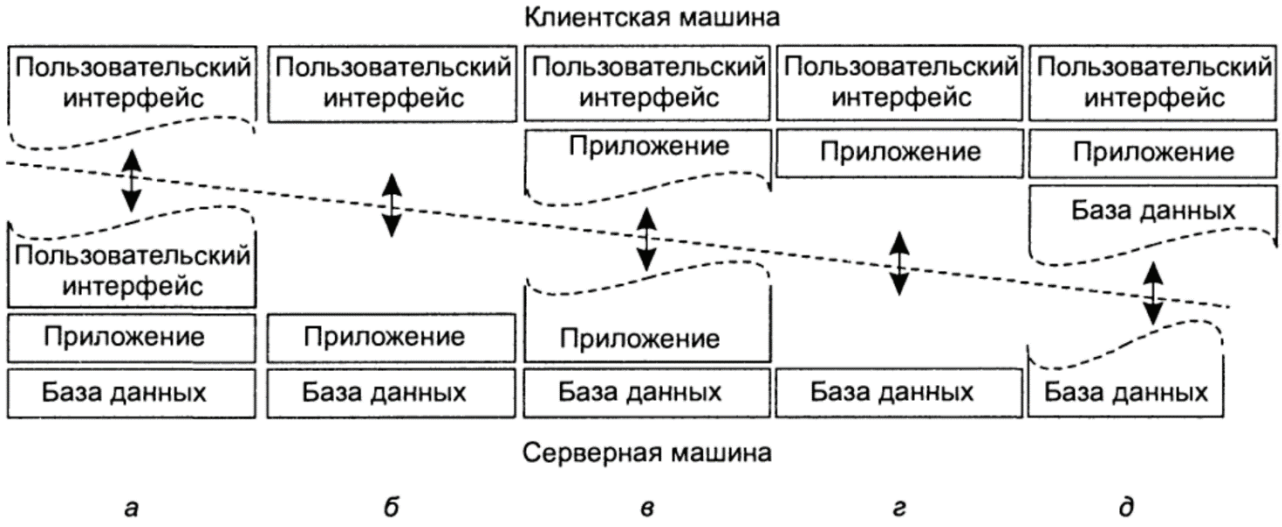

Рисунок 2 - Варианты организации клиент-серверной архитектуры

Таким образом, разрабатываемая программная система должна иметь архитектуру, представленную в приложении В. Описание компонентов системы приведено в Таблица 1.

Таблица 1 - Описание компонентов системы

Компонент	**Описание**
Компоненты системы	
Client	Клиентская часть, обеспечивающая работу пользователя с устройством при помощи данных, полученных ею от сервера
Server	Серверная часть, в задачи которой входит принятие запросов от клиентов и обработка данных для их удовлетворения

UserInterface (Client)	Пакет для построения пользовательского интерфейса, и последующего взаимодействия пользователя с ним
ClientKernel	Ядро клиента, обеспечивающее взаимодействие между его пакетами
Parser	Пакет, предназначенный для обработки десереализованного файла и приведения в формат, пригодный для формирования пользовательского интерфейса
TransactionManager	Пакет, предназначенны для обмена данными между ядром системы и COM manager
ConnectionManager	Пакет, отвечающий за общение с сервером, прием и отправку данных
COM manager	Пакет, отвечающий за работу программы непосредственно с COM портом
UserInterface (DDParser)	Компонент утилиты преобразования DD-файлов в JSON, отвечающий за предоставление GUI
DD Parser	Утилита преобразования DD-файлов в JSON
Server Core	Ядро сервера, обеспечивающее взаимодействие между его компонентами
RequestProcessor	Обработчик запросов от клиентов
DeviceModel	Модель устройства, представляющая собой описание устройства в некотором внутреннем формате, пригодном для дальнейшего использования клиентом
Database manager	Компонент, отвечающий за взаимодействие других компонентов серверной части с базами данных
DBMS	Система управления базами данных
Database	База данных, используемая для хранения моделей описания устройств, преобразованных в JSON
Внешние компоненты	
HART Modem	Устройство для сопряжение ПК с HART Device
HART Device	Полевое устройство, поддерживающее протокол HART

2.3 Обзор литературы по обоснованию выбора языков программирования

Для разработки графического интерфейса клиента наиболее подходящим является язык разметки HTML с таблицами стилей CSS, так как согласно [10] они обеспечивают необходимую гибкость для динамической отрисовки графических элементов и позволяют получить идентичный интерфейс на различных платформах. Для реализации функционала клиента используется объектно-ориентированный язык JavaScript, который согласно [9] поддерживает полную интеграцию с HTML, что позволяет эффективно объединять функциональную часть клиента и его графическую составляющую. Для расширения функционала графического интерфейса системы так же используется фреймворк AngularJS компании Google, так как его функционал направлен на улучшение работы одностраничных сайтов, написанных с применением JavaScript.

К наиболее популярным языкам программирования, наилучшим образом подходящих для разработки серверного программного обеспечения, можно отнести C#, Java и Python. Основным источником информации о данных языках программирования послужили учебники, посвященные программированию на них.

Анализ [4], [5] и [6] показал, что все три языка включают в себя достаточно мощный набор средств (в частности, все три языка поддерживают объектно-ориентированне программирование, коллекции, обработку исключений и пр.). Однако в [4] описана инфраструктура Windows Communication Foundation, поддерживаемая языком C#. Данная инфраструктура интегрирует целый набор технологий распределенной обработки в один согласованный API-интерфейс. Она позволяет разрабатывать службы, применяющие протоколы HTTP и SOAP, что позволяет внешним клиентам пользоваться их функциональностью, независимо от языка программирования или операционной системы, используемых в клиентской части системы. Таким образом, языком программирования наиболее удобным для разработки серверной части программной системы является C#.

2.4 Обзор литературы по обоснованию выбора программного обеспечения

Для обеспечения мультиплатформенности клиентская часть программного обеспечения может быть реализована в виде расширения для браузера Mozilla firefox написанном на платформе Gecko (12,8% рынка), но данная реализации менее популярна чем платформа Chromium (70,7% рынка), а также не позволяет реализовать функционал, связанный с подключением к последовательному порту компьютера.

Для решения задачи запуска клиентской части приложения на большинстве популярных платформ был сделан выбор в пользу Chrome application так как, Chrome API позволяет реализовать необходимый функционал такой как работа с последовательными портами, работа в режиме оффлайн, а также возможность обмена информации с серверной частью приложения.

Так как на серверной части приложения будет размещена база данных, необходима система управления базами данных для работы с ней, причем предпочтительно, чтобы СУБД относилась к свободному программному обеспечению. К наиболее популярным реляционным СУБД относятся PostgreSQL и MySQL. Основным источником информации о данных СУБД послужила техническая документация, предоставленная разработчиками. Согласно [7] и [8], обе СУБД предоставляют необходимый для работы программной системы функционал, однако лицензия PostgreSQL предусматривает меньшие ограничения на ее использование по сравнению с MySQL. Таким образом, для работы с базой данных будет использоваться СУБД PostgreSQL.

2.5 Обзор литературы по обоснованию выбора операционной системы

Браузеры написанные на основе платформы Chromium (Chrome, Mozilla Firefox, Opera) в настоящее время поддерживают все доступные десктопные операционные системы (Windows, MacOS, Linux), что позволяет клиентской части приложения, написанной на HTML/CSS и JavaScript с применением фреймворка AngularJS запускаться на данных операционных системах одинаково, тем самым обеспечивая его мультиплатформенность.

Для разработки серверной части программной системы был выбран язык программирования C#, использующий платформу Microsoft .NET Framework, рассчитаную на работу лишь под операционными системами семейства Windows. По этой причине серверные компоненты системы смогут поддерживать лишь операционные системы данного семейства.

Таким образом, необходимость в выборе операционной системы отсутствует, и обзор литературы по данной тематике не производился.

Выводы по разделу два

По результатам анализа литературы можно сделать следующие выводы:

1) Разрабатываемая программная система должна использовать для обмена данными с контрольно-измерительными приборами последовательный порт RS-232 или USB-порт.

2) Разрабатываемая программная система должна поддерживать работу с DD-файлами.

3) Разрабатываемая программная система должна иметь архитектуру, представленную в приложении В.

4) Для разработки серверной части программной системы должен использоваться язык программирования C#.

5) Серверная часть программной системы будет поддерживать работу под операционными системами семейства Windows.

6) Для работы с базой данных должна использоваться СУБД PostgreSQL.

7) Для разработки клиентской части программной системы должны использоваться языки HTML/CSS и JavaScript.

8) Клиентская часть программной системы должна быть разработана для браузеров, разработанных в среде Chromium.

9) Клиентская часть программной системы будет доступна для операционных систем Windows, Linux, а также MacOS.

3 РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

Перед непосредственной реализацией программной системы был составлен набор требований, которым должен будет отвечать конечный продукт для выполнения поставленных задач.

3.1 Требования к системе в целом

3.1.1 Требования к структуре и функционированию системы

Разрабатываемая программная система должна иметь архитектуру, представленную в приложении В.

3.1.2 Требования к надежности

Исключительные ситуации, возникновение которых возможно в ходе работы программной системы, перечислены в Таблица 2 с указанием возможных причин их возникновения, реакции на них программы и способов их устранения.

Таблица 2 - Перечень исключительных ситуаций

ID	Краткое название	Описание	Возможные причины	Действия программы	Как исправить
EX001	AttributeExpected	DD-объект не имеет некоторого необходимого атрибута	В DD-файле в описании данного объекта отсутствует запрашиваемый атрибут	Прервать обработку данного объекта, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX002	CannotConvertType	Не удалось преобразовать DD-тип к одному из используемых внутри приложения	В DD-файле в качестве типа переменной указано некорректное значение	Прервать обработку переменной, имеющей неизвестный тип, не добавляя ее в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению

EX003	CannotEvaluateExpression	Не удалось вычислить выражение	В DD-файле описано выражение, содержащее синтаксические ошибки	Прервать обработку объекта, использующего данное выражение, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX004	CannotFindCollection	Не удалось найти DD-коллекцию с указанным идентификационным номером	В DD-файле отсутствует описание запрашиваемой коллекции, либо ее описание содержало ошибки, и коллекция не была добавлена в модель	Прервать обработку объекта, сославшегося на данную коллекцию, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX005	CannotFindElement	Не удалось найти элемент DD-массива с указанным индексом	В DD-файле в описании данного массива отсутствует элемент с указанным индексом	Прервать обработку объекта, сославшегося на данный элемент, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX006	CannotFindEnumerationElement	Не удалось найти элемент DD-перечисления с указанным значением	В DD-файле в описании данного перечисления отсутствует элемент с указанным значением	Прервать обработку объекта, сославшегося на данный элемент, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX007	CannotFindItem	Не удалось найти DD-объект с указанным идентификационным номером	В DD-файле отсутствует описание запрашиваемого объекта, либо его описание содержало ошибки, и объект не был добавлен в модель	Прервать обработку объекта, сославшегося на данный объект, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению

EX008	CannotFindMember	Не удалось найти член DD-коллекции с указанным именем	В DD-файле в описании данной коллекции отсутствует член с указанным именем	Прервать обработку объекта, сославшегося на данный член, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX009	CannotFindRefArray	Не удалось найти DD-массив с указанным идентификационным номером	В DD-файле отсутствует описание запрашиваемого массива, либо его описание содержало ошибки, и массив не был добавлен в модель	Прервать обработку объекта, сославшегося на данный массив, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX010	CannotFindVariable	Не удалось найти переменную с указанным идентификационным номером	В DD-файле отсутствует описание запрашиваемой переменной, либо ее описание содержало ошибки, и переменная не была добавлена в модель	Прервать обработку объекта, сославшегося на данную переменную, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX011	CannotGetDictionaryString	Не удалось найти необходимую строку в стандартном словаре (в .std-файле)	В стандартном словаре отсутствует запрашиваемся строка, либо она имела некорректный формат и не была загружена	Прервать обработку объекта, сославшегося на данную строку, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Заменить используемый стандартный словарь на более новый
EX012	CannotGetLocalizedString	Не удалось найти необходимую строку на на нужном языке	В DD-файле отсутствует запрашиваемая строка на нужном языке, либо она имела некорректный формат и не была загружена	Прервать обработку объекта, сославшегося на данную строку, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению

EX013	CannotLoadItem	Не удалось загрузить DD-объект	В DD-файле описание данного объекта содержит ошибки	Прервать обработку данного объекта, не добавляя его в модель; добавить текст ошибки в поле модели Warnings; перейти к обработке следующего объекта	Не поддается исправлению
EX014	CannotLoadLocalizedString	Не удалось загрузить строку	Данная строка имеет некорректный формат	Прервать обработку данной строки, не загружаяя ее; перейти к обработке следующей строки	Не поддается исправлению
EX015	ErrorLoadingSymFile	Не удалось загрузить .sym-файл	Соответствующий .sym-файл не был обнаружен, или содержит в себе ошибки	Добавить текст ошибки в поле модели Warnings; формировать имена DD-объектов при обработке на основе их идентификационных номеров	Убедиться, что .sym-файл лежит одной папке с DD-файлом и имеет то же имя, что и он; попытаться заменить .sym-файл
EX016	DeviceDescriptionDuplicate	Не удалось добавить модель описания устройства в базу данных, так как там уже хранится описание данного устройства	В базе данных обнаружена модель описания устройства, расширенный код типа которого совпадает с аналогичным кодом, указанным в обрабатываемом DD-файле	Вывести окно с предупреждением и предоставить пользователю выбрать один из двух вариантов: заменить уже имеющуюся модель, или отказаться от добавления новой модели в базу данных	Заменить уже имеющуюся модель, или отказаться от добавления новой модели в базу данных
EX017	ErrorLoadingDD	Не удалось загрузить DD-файл	DD-файл содержит в себе ошибки, препятствующие его корректной обработке; MajorRevision DD-файла отличается от 6 и 8; не удалось получить полный путь DD-файла (например, потому что полный путь слишком	Прервать обработку DD-файла; выдать на экран текст ошибки	Переместить DD-файл в другую папку

			длинный)		
EX018	TimeOutException	Превышено время ожидания ответа устройства	Устройство отключено от компьютера, на устройство не подано питание	Вывод окна с информацией об ошибке и способами её устранения	Проверить подключение устройства к компьютеру, проверить питание устройства
EX019	ServerConnectionException	Ошибки соединения с сервером	Компьютер пользователя отключен от интернета, сервер временно недоступен	Вывод окна с информацией об ошибке, способами её устранения и предложением перехода в автономный режим	Проверить наличие соединения с интернетом
EX020	JSONModel Exctption	Ошибка модели JSON файла	Клиент получил с сервера JSON файл не той модели	Вывод окна с информацией об ошибке, и предложением перейти в автономный режим	Исправление модели описания устройства на сервере
EX021	ConformityCommandException	Ошибка соответствия команды	Ошибка в описании отправленной устройству команды	Вывод окна с информацией об ошибке	Исправление команды в файле описания устройства

3.2 Функциональные требования

3.2.1 Требования к серверному приложению

1) ПО должно обеспечивать обработку клиентских запросов, формат которых соответствует описанному в Таблица 3.

Таблица 3 - Формат запроса

Компонент сообщения	**Описание**
POST /RequestProcessingService HTTP/1.1 Content-Type: text/xml; charset=utf-8 SOAPAction: "http://hartonline.com/IRequestProcessingService/GetDeviceModel" Host: *адрес сервера* Content-Length: *размер сообщения* Expect: 100-continue Accept-Encoding: gzip, deflate Connection: Keep-Alive	HTTP-заголовок
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">	Начало SOAP-сообщения
<soap:Body>	Начало тела запроса
<GetDeviceModel xmlns=" http://hartonline.com">	Начало описания передаваемых серверу данных (в качестве тега указывается имя вызываемого метода службы)
<expandedDeviceType>*Расширенный код типа устройства*</deviceType>	Аргументы вызываемого метода службы GetDeviceModel, передаваемые клиентом
<deviceRevision>*Номер ревизии устройства*</deviceRevision>	
<ddRevision>*Номер ревизии DD-файла*</ddRevision>	
</GetDeviceModel>	Конец описания передаваемых серверу данных
</soap:Body>	Конец тела запроса
</soap:Envelope>	Конец SOAP-сообщения

2) ПО должно при получении запроса найти в базе данных модель описания устройства, используя информацию, переданную в теле запроса (расширенный код типа устройства, номер ревизии DD-файла и номер ревизии устройства), и передать ее клиенту.

3) ПО должно формировать ответное сообщение, используя формат, описанный в **Таблица** 4.

Таблица 4 - Формат ответа на запрос

Компонент сообщения	**Описание**
HTTP/1.1 200 OK Content-Length: *размер сообщения* Content-Type: text/xml; charset=utf-8 Server: Microsoft-HTTPAPI/2.0 Date: *дата/время отправки*	HTTP-заголовок
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">	Начало SOAP-сообщения
<soap:Body>	Начало тела ответа
<GetDeviceModelResponse xmlns="http://hartonline.com">	Начало описания возвращаемых клиенту данных
<GetDeviceModelResult>*Модель устройства*</ GetDeviceModelResult>	Возвращаемая клиенту модель описания устройства в формате JSON
</GetDeviceModelResponse>	Конец описания возвращаемых клиенту данных
</soap:Body>	Конец тела ответа
</soap:Envelope>	Конец SOAP-сообщения

4) ПО должно быть выполнено в виде Windows-службы.

5) ПО должно хранить настройки подключения к базе данных и адрес сервера в конфигурационном файле.

3.2.2 Требования к утилите преобразования DD-файлов

1) ПО должно строить модель описания устройства путем анализа DD-файла с описанием данного устройства.

2) ПО должно строить модель описания устройства, формат которой соответствует описанному в Таблица 5.

Таблица 5 - Формат модели описания устройства

Поле	**Описание**
Manufacturer	Код производителя устройства
DeviceType	Код типа устройства
DeviceRevision	Номер ревизии устройства
DDRevision	Номер ревизии DD-файла
Variables	Переменные, используемые устройством (формат описания переменных указан в Таблица 6)
Commands	Команды, используемые устройством (формат описания команд указан в Таблица 8)
Arrays	Массивы, используемые устройством (формат описания массивов указан в Таблица 16)
Collections	Коллекции, используемые устройством (формат описания коллекций указан в Таблица 14)
Errors	Тексты критических ошибок, обнаруженных в ходе анализа соответствующего DD-файла
Warnings	Тексты некритических ошибок, обнаруженных в ходе анализа соответствующего DD-файла

Таблица 6 - Формат описания переменной

Поле	**Описание**
Name	Имя переменной
Label	Текст, который клиентское приложение будет выводить при отображении данной переменной
Help	Вспомогательная информация
Type	Тип переменной (перечисление, используемое для обозначения типа переменной, описано в Таблица 7)
Value	Значение переменной
Length	Длина переменной в символах (задается для переменных строкового типа, для остальных равна 1)

Таблица 7 - Типы переменных

Название	**Соответствующее значение**	**Описание**
Default	0	Неизвестный тип
Unsigned8	1	8-битное беззнаковое целое
Unsigned16	2	16-битное беззнаковое целое
Unsigned24	3	24-битное беззнаковое целое
Unsigned32	4	32-битное беззнаковое целое
Unsigned40	5	40-битное беззнаковое целое
Unsigned48	6	48-битное беззнаковое целое
Unsigned56	7	56-битное беззнаковое целое
Unsigned64	8	64-битное беззнаковое целое
Signed8	9	8-битное знаковое целое
Signed16	10	16-битное знаковое целое
Signed24	11	24-битное знаковое целое
Signed32	12	32-битное знаковое целое
Signed40	13	40-битное знаковое целое
Signed48	14	48-битное знаковое целое
Signed56	15	56-битное знаковое целое
Signed64	16	64-битное знаковое целое
Float	17	32-битное с плавающей точкой
Double	18	64-битное с плавающей точкой
AsciiString	19	Строка ASCII
PackedAsciiString	20	Запакованная строка ASCII
HartDate	21	Дата
Enum	22	Перечисление
BitEnum	23	Битовое перечисление
Time	24	Время
Index	25	Индекс

Таблица 8 - Формат описания команды

Поле	**Описание**
Name	Имя команды
Number	Номер команды
Operation	Тип действий, предпринимаемых устройством при получении данной команды (перечисление, используемое для обозначения типа операции, описано в Таблица 9)
Transactions	Транзакции, используемые командой (формат описания транзакций указан в Таблица 10)
ResponseCodes	Коды ответа, которые может вернуть устройство в ответ на данную команду (формат описания кодов ответа указан в Таблица 12)

Таблица 9 - Типы операций

Название	**Соответствующее значение**	**Описание**
None	0	Неизвестный тип
Read	1	При получении такой команды устройство возвращает текущие значения набора переменных
Write	2	При получении такой команды устройство устанавливает значения набора переменных в соответствии со значениями, переданными ему клиентским приложением
Command	3	При получении такой команды устройство выполняет специфический для него набор действий

Таблица 10 - Формат описания транзакции

Поле	**Описание**
Request	Набор элементов данных, описывающих запрос, посылаемый устройству (формат описания элементов данных указан в Таблица 11)
Reply	Набор элементов данных, описывающих ответ, получаемый от устройства (формат описания элементов данных указан в Таблица 11)
ResponseCodes	Коды ответа, которые может вернуть устройство в ответ на данную транзакцию (формат описания кодов ответа указан в Таблица 12)

Таблица 11 - Формат описания элемента данных

Поле	**Описание**
Item	В зависимости от значения поля IsConstant – целочисленная беззнаковая константа или идентификационный номер некоторой переменной
IsConstant	Если данное поле принимает истинное значение, то в поле Item содержится целочисленная беззнаковая константа; в противном случае – идентификационный номер переменной
IsInfo	Если данное поле принимает истинное значение, то переменная, идентификационный номер которой содержится в поле Item, на самом деле не содержится в устройстве и используется при коммуникации для информационных целей
IsIndex	Если данное поле принимает истинное значение, то значение переменной, идентификационный номер которой содержится в поле Item, является индексом элемента в некотором массиве
HasMask	Если данное поле принимает истинное значение, то данный элемент данных имеет битовую маску
Mask	Битовая маска, которую необходимо, показывающая какую часть байта занимает значение переменной, идентификационный номер которой содержится в поле Item

Таблица 12 - Формат описания кода ответа

Поле	**Описание**
Description	Текст, отображаемый клиентским приложением при возвращении данного кода ответа устройством
Help	Текст, предоставляющий расширенное описание данного кода ответа
Value	Значение кода ответа
Type	Тип кода ответа (перечисление, используемое для обозначения типа кода ответа, описано в Таблица 13)

Таблица 13 - Типы кодов ответа

Название	**Соответствующее значение**	**Описание**
None	0	Неизвестный тип
Success	1	Команда была принята и обработана
MiscWarning	2	Команда была принята и обработана, но есть дополнительная информация, не связанная с командой, и в которой может быть заинтересован пользователь
DataEntryWarning	3	Команда была принята и обработана с некоторой модификацией переданных данных
DataEntryError	4	Команда была отклонена из-за некорректности переданных данных
ModeError	5	Команда была отклонена из-за того, что устройство находится в режиме, в котором невозможна обработка данной команды
ProcessError	6	Команда была отклонена из-за того некорректности применяемого к устройству процесса
MiscError	7	Команда была отклонена

Таблица 14 - Формат описания коллекции

Поле	**Описание**
Name	Имя коллекции
Label	Краткое описание коллекции
Help	Вспомогательная информация
Members	Члены коллекции (формат описания членов коллекции указан в Таблица 15)

Таблица 15 - Формат описания члена коллекции

Поле	**Описание**
ItemId	Идентификационный номер некоторого DD-объекта
Description	Краткое описание объекта
Help	Вспомогательная информация

Таблица 16 - Формат описания массива

Поле	**Описание**
Name	Имя массива
Label	Краткое описание массива
Help	Вспомогательная информация
Elements	Элементы массива (формат описания элементов массива указан в Таблица 17)

Таблица 17 - Формат описания элемента массива

Поле	**Описание**
ItemId	Идентификационный номер некоторого DD-объекта
Description	Краткое описание объекта
Help	Вспомогательная информация

3) ПО должно преобразовывать построенную модель описания устройства в формат JSON и помещать результат в базу данных.

4) ПО должно поддерживать работу с DD-файлами расширения .fm6 и .fm8.

5) ПО должно извлекать вспомогательную информацию об устройствах из файлов расширения .std (стандартный словарь, содержащий набор строк, переведенных на различные языки) и .sym (содержит имена переменных, команд и т. д.).

6) ПО должно иметь графический пользовательский интерфейс, соответствующий макету, изображенному на Рисунок 3.

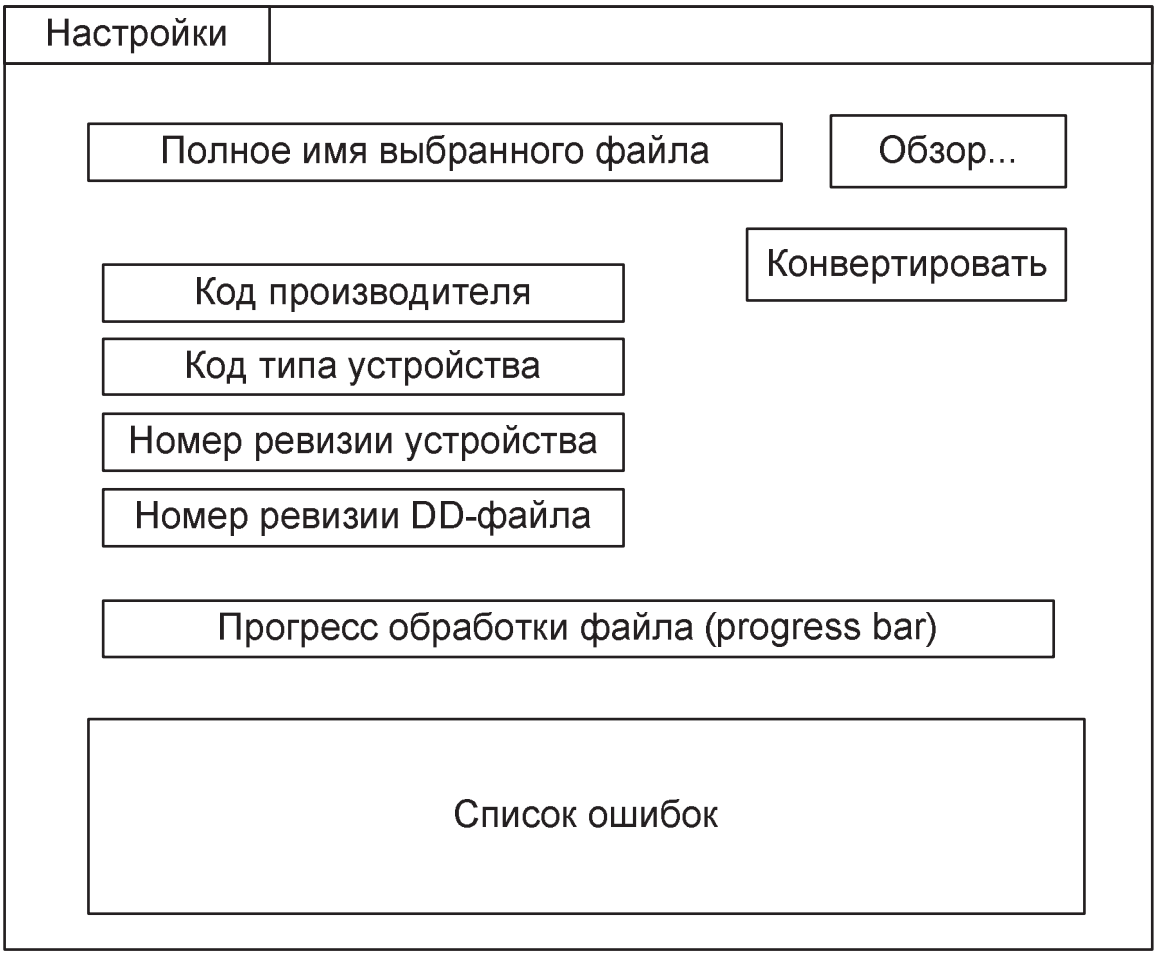

Рисунок 3 - Макет GUI утилиты по преобразованию DD-файла в JSON

3.2.3 Требования к клиентскому приложению

1) ПО должно осуществлять поиск устройств посредством COM портам последовательно с 1 по 64 адрес.

2) ПО должно подключаться к первому найденному устройству.

3) ПО должно автоматически осуществлять подбор скорости обмена с COM портом.

4) ПО должно обмениваться HART посылками с подключенным устройством.

5) ПО должно иметь графический пользовательский интерфейс, соответствующий макету, изображенному на Рисунок 4.

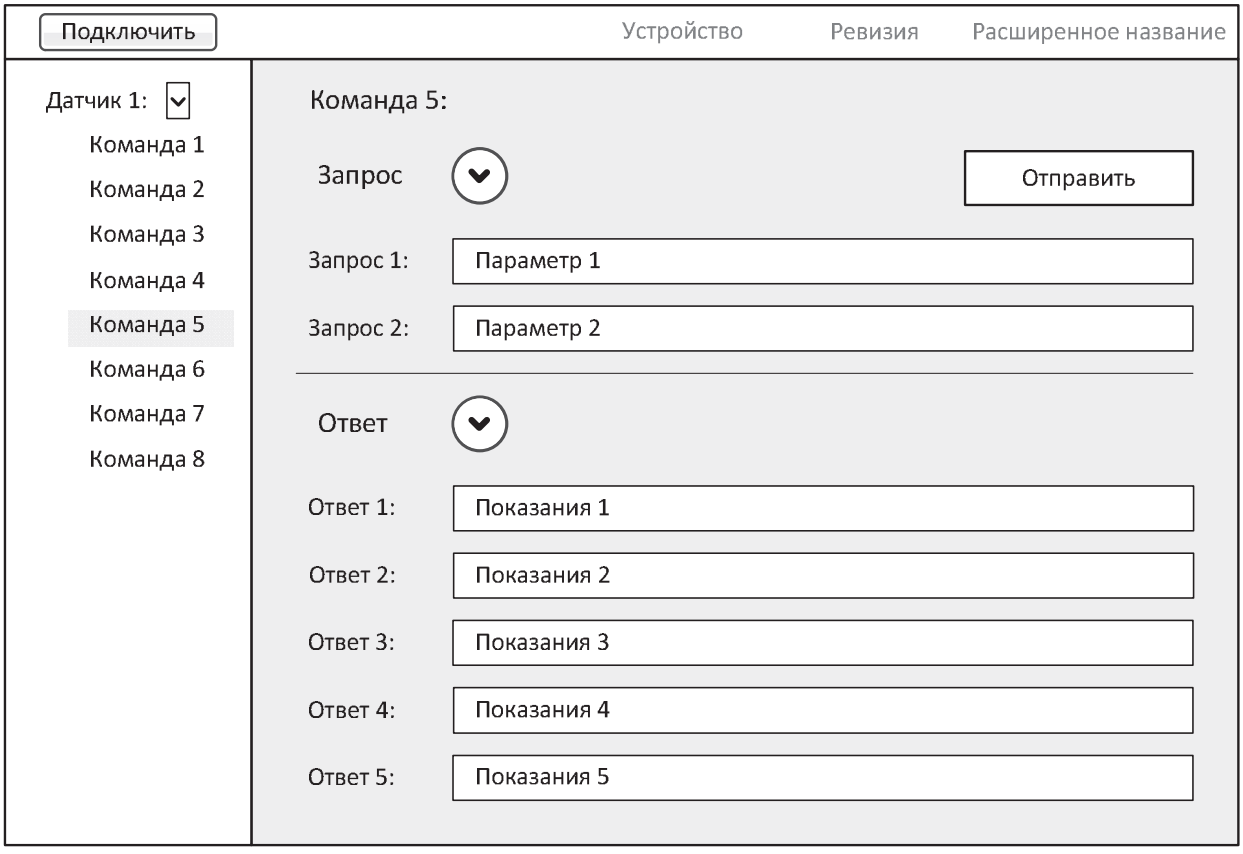

Рисунок 4 - Макет GUI клиентского приложения

6) ПО должно формировать HART посылки по нажатию пользователем на кнопку «Отправить».

7) ПО должно отправлять запрос на сервер с информацией об устройстве (расширенный код типа устройства, номер ревизии DD-файла и номер ревизии устройства).

8) ПО должно обрабатывать полученный с сервера JSON файл и на его основе строить пользовательский интерфейс.

9) ПО должно поддерживать работу в автономном режиме и иметь набор универсальных команд, которые поддерживают устройства, оснащенные HART протоколом.

10) ПО должно строить дерево с поддерживаемыми командами устройства, на основе обработанного JSON файла, полученного с сервера.

11) ПО должно реагировать на выбор пользователем команды и отображать поля ответа и запроса, выбранной команды.

4 РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ

Результат разработки программной системы представлен в виде листингов исходного кода полученного прототипа системы. Ввиду большого объема исходного кода в рамках данной пояснительной записки приведены листинги лишь наиболее важных частей кода некоторых компонентов системы. Данные листинги приведены в приложениях А1, А2, А3, А4, А5, А6.

Реализации графического пользовательского интерфейса клиентской части системы и утилиты по преобразованию DD-файлов в JSON приведены в приложениях Б1 и Б2.

Основные сценарии работы системы представлены на Рисунок 6, Рисунок 7 и Рисунок 8.

Рисунок 5 - Штатная работа системы

Рисунок 6 - Работа системы при отсутствии информации об устройстве

Рисунок 7 - Работа системы при отсутствии связи между клиентом и сервером

ЗАКЛЮЧЕНИЕ

В результате проведенной работы произведен анализ наиболее широко используемых программных продуктов, предназначенных для работы с HART-устройствами. Изучены основные подходы к разработке подобного программного обеспечения путем изучения литературы по теме выпускной квалификационной работы

Разработаны архитектура системы, учитывающая недостатки существующих аналогов, и требования к ней. Создан прототип программной системы. Таким образом, цель работы достигнута, задачи решены.

Результаты работы рекомендуется использовать при подготовке специалистов, профессиональная деятельность которых связана с работой с контрольно-измерительными приборами, поддерживающими работу с HART-протоколом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 HART Communication. Application Guide - https://www.fieldcommgroup.org/sites/default/files/technologies/hart/ApplicationGuide_r7.1.pdf

2 Приемы объектно-ориентированного программирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2001. – 368 с.

3 Танненбаум, Э. Распределенные системы. Принципы и парадигмы / Э. Танненбаум, М. ван Стеен. – СПб.: Питер, 2003. – 877 с.

4 Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – М.: Вильямс, 2013. – 1312 с.

5 Шилдт, Г. Java. Полное руководство / Г. Шилдт. – М.: Вильямс, 2012. – 1104 с.

6 Лутц, М. Изучаем Python / М. Лутц. – Санкт-Петербург.: Символ-Плюс, 2011. – 1280 с.

7 Документация к PostgreSQL 9.6.2 - http://repo.postgrespro.ru/doc/pgsql/9.6.2/ru/postgres-A4-fop.pdf

8 MySQL 5.7 Reference Manual - https://downloads.mysql.com/docs/refman-5.7-en.a4.pdf

9 Флэнаган Д. JavaScript. Подробное руководство / Д. Флэнаган. – Санкт-Петербург.: Символ-Плюс, 2016. – 1078 с.

10 Лоусон Б., Шарп Р. Изучаем HTML 5 / Б. Лоусон, Р. Шарп. – Санкт-Петербург.: Питер, 2011. – 272 с.

11 http://rosemountmarine.com/RU-RU/BRANDS/AMSSUITE/AMSDEVICEMANAGER/Pages/AMSDeviceManager.aspx

12 http://www.pactware.com/en/Products/PACTware

13 http://www.elemer.ru/po/hart_config.php

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А1

Листинг фрагмента исходного кода компонента системы UserInterface (DDParser)

```
namespace HartOnline.DDParser.GUI
{
    using System;
    using System.ComponentModel;
    using System.Globalization;
    using System.IO;
    using System.Windows.Forms;
    using System.Windows.Threading;

    using HartOnline.DatabaseManagement;
    using HartOnline.DDParser.FmParsing;
    using HartOnline.DDParser.GUI.Properties;
    using HartOnline.DDParser.GUI.Resources;
    using HartOnline.DDParser.Model;
    using Newtonsoft.Json;

    /// <summary>
    /// Main window of program
    /// </summary>
    public partial class MainWindow : Form
    {
        /// <summary>
        /// Background worker, that is used for parsing and working with database
        /// </summary>
        private BackgroundWorker parsingWorker;

        /// <summary>
        /// Dispatcher, that is linked with UI thread
        /// </summary>
        private Dispatcher dispatcher;

        /// <summary>
        /// Initializes a new instance of the <see cref="MainWindow"/> class.
        /// </summary>
        public MainWindow()
        {
            this.InitializeComponent();

            this.Text = Localization.strProgramName;
            this.browseButton.Text = Localization.strBrowseButton;
            this.convertButton.Text = Localization.strConvertButton;
            this.manufacturerCodeLabel.Text = Localization.strManufacturerCodeLabel;
            this.deviceTypeLabel.Text = Localization.strDeviceTypeLabel;
            this.deviceRevisionLabel.Text = Localization.strDeviceRevisionLabel;
            this.ddRevisionLabel.Text = Localization.strDDRevisionLabel;
            this.errorListLabel.Text = Localization.strErrorListLabel;
            this.progressBar.MarqueeAnimationSpeed = 10;
            this.openFileDialog.InitialDirectory =
                Path.GetPathRoot(Environment.CurrentDirectory);
            this.menuStrip.Items[0].Text = Localization.strSettingsMenuItem;

            this.parsingWorker = new BackgroundWorker();
            this.dispatcher = Dispatcher.CurrentDispatcher;
```

```
this.parsingWorker.DoWork +=
    new DoWorkEventHandler((s, eventArgs) =>
    {
        IParser metadataProvider = new FmParser();
        DeviceModel model =
            metadataProvider.ExtractMetadata(ddFilePathTextbox.Text);

        if (model.Errors.Count == 0)
        {
            using (StringWriter writer = new StringWriter())
            {
                JsonSerializer serializer = new JsonSerializer();
                serializer.Serialize(writer, model);

                DatabaseManager databaseManager =
                    new DatabaseManager(
                        Settings.Default.Server,
                        Settings.Default.Login,
                        Settings.Default.Password,
                        Settings.Default.Database);

                long expandedDeviceType =
                    ((long)model.ManufacturerCode << (sizeof(ushort) * 8)) +
                         model.DeviceTypeCode;

                try
                {
                    if (string.IsNullOrEmpty(
                        databaseManager.GetModel(expandedDeviceType,
                                model.DeviceRevision, model.DDRevision)))
                    {
                        databaseManager.InsertModel(
                            expandedDeviceType,
                            model.DeviceRevision,
                            model.DDRevision,
                            writer.ToString());
                    }
                    else
                    {
                        DialogResult dialogResult =
                                dispatcher.Invoke<DialogResult>(() =>
                        {
                            return MessageBox.Show(
                                this,
                                Localization.strModelAlreadyExists,
                                Localization.strProgramName,
                                MessageBoxButtons.YesNo);
                        });

                        if (dialogResult == DialogResult.Yes)
                        {
                            databaseManager.UpdateModel(
                                expandedDeviceType,
                                model.DeviceRevision,
                                model.DDRevision,
                                writer.ToString());
                        }
                    }
                }
                catch (Exception e)
                {
                    // Cannot insert data into database. Exception: {0}
                    model.Errors.Add(
                        string.Format(
```

```
                                        CultureInfo.CurrentCulture,
                                        Localization.strCannotInsertData,
                                        e.Message));
                                }
                            }
                        }

                        eventArgs.Result = model;
                    });

                this.parsingWorker.RunWorkerCompleted +=
                    new RunWorkerCompletedEventHandler((s, eventArgs) =>
                    {
                        manufacturerCodeTextBox.Text =
                            string.Format("0x{0,-6:X6}", (eventArgs.Result as
                                  DeviceModel).ManufacturerCode);
                        deviceTypeTextBox.Text =
                            string.Format("0x{0,-4:X4}", (eventArgs.Result as
                                  DeviceModel).DeviceTypeCode);
                        deviceRevisionTextBox.Text =
                            string.Format("0x{0,-2:X2}", (eventArgs.Result as
                                  DeviceModel).DeviceRevision);
                        ddRevisionTextBox.Text =
                            string.Format("0x{0,-2:X2}", (eventArgs.Result as
                                  DeviceModel).DDRevision);

                        foreach (string warning in (eventArgs.Result as DeviceModel).Warnings)
                        {
                            errorsDataGrid.Rows.Add(string.Format("Warning: {0}", warning));
                        }

                        foreach (string error in (eventArgs.Result as DeviceModel).Errors)
                        {
                            errorsDataGrid.Rows.Add(string.Format("Error: {0}", error));
                        }

                        progressBar.Visible = false;
                    });
            }

            private void browseButton_Click(object sender, EventArgs e)
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    ddFilePathTextbox.Text = openFileDialog.FileName;
                    openFileDialog.InitialDirectory =
                        Path.GetDirectoryName(openFileDialog.FileName);
                }
            }

            private void convertButton_Click(object sender, EventArgs e)
            {
                errorsDataGrid.Rows.Clear();
                progressBar.Visible = true;
                this.parsingWorker.RunWorkerAsync();
            }

            private void settingsToolStripMenuItem_Click(object sender, EventArgs e)
            {
                SettingsWindow settingsWindow = new SettingsWindow();
                settingsWindow.ShowDialog(this);
            }
        }
    }
```

ПРИЛОЖЕНИЕ А2

Листинг фрагмента исходного кода компонента системы RequestProcessor

```
namespace HartOnline.Core.RequestProcessing
{
    using HartOnline.DatabaseManagement;

    public class RequestProcessingService : IRequestProcessingService
    {
        public string GetDeviceModel(long expandedDeviceType, byte deviceRevision,
                byte ddRevision)
        {
            DatabaseManager databaseManager =
                new DatabaseManager(
                    Settings.Default.Server,
                    Settings.Default.Login,
                    Settings.Default.Password,
                    Settings.Default.Database);

            string result = databaseManager.GetModel(expandedDeviceType, deviceRevision,
                                    ddRevision);

            if (string.IsNullOrEmpty(result))
            {
                result = databaseManager.GetModel(0, 0, 0);
            }

            return result;
        }
    }
}
```

ПРИЛОЖЕНИЕ А3

Листинг фрагмента исходного кода компонента системы DatabaseManager

```
namespace HartOnline.DatabaseManagement
{
    using System.Globalization;
    using Npgsql;

    /// <summary>
    /// Provides access to PostgreSQL database
    /// </summary>
    public class DatabaseManager
    {
        /// <summary>
        /// Parameters, that are used to connect to the database
        /// </summary>
        private string connectionString;

        /// <summary>
        /// Initializes a new instance of the <see cref="DatabaseManager"/> class.
        /// </summary>
        /// <param name="server">Name of database server</param>
        /// <param name="login">Login of user, that will be used to connect to the
            database</param>
        /// <param name="password">Password of user, that will be used to connect to the
            database</param>
        /// <param name="database">Name of database</param>
        public DatabaseManager(string server, string login, string password,
             string database)
        {
            NpgsqlConnectionStringBuilder stringBuilder =
                new NpgsqlConnectionStringBuilder();

            stringBuilder.Host = server;
            stringBuilder.Username = login;
            stringBuilder.Password = password;
            stringBuilder.Database = database;

            this.connectionString = stringBuilder.ToString();
        }

        /// <summary>
        /// Inserts device description model into database
        /// </summary>
        /// <param name="expandedDeviceType">Expanded device type code</param>
        /// <param name="deviceRevision">Device revision</param>
        /// <param name="ddRevision">DD-file revision</param>
        /// <param name="model">Device description model</param>
        public void InsertModel(
            long expandedDeviceType,
            byte deviceRevision,
            byte ddRevision,
            string model)
        {
            using (NpgsqlConnection connection =
                new NpgsqlConnection(this.connectionString))
            {
                connection.Open();

                using (var cmd = new NpgsqlCommand())
```

```
                {
                    cmd.Connection = connection;
                    cmd.CommandText = string.Format(
                            CultureInfo.CurrentCulture,
                            "INSERT INTO models VALUES ({0}, {1}, {2}, $${3}$$)",
                            expandedDeviceType,
                            deviceRevision,
                            ddRevision,
                            model);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        /// <summary>
        /// Updates device description model into database
        /// </summary>
        /// <param name="expandedDeviceType">Expanded device type code</param>
        /// <param name="deviceRevision">Device revision</param>
        /// <param name="ddRevision">DD-file revision</param>
        /// <param name="model">Device description model</param>
        public void UpdateModel(
            long expandedDeviceType,
            byte deviceRevision,
            byte ddRevision,
            string model)
        {
            using (NpgsqlConnection connection = new
                NpgsqlConnection(this.connectionString))
            {
                connection.Open();

                using (var cmd = new NpgsqlCommand())
                {
                    cmd.Connection = connection;
                    cmd.CommandText = string.Format(
                            CultureInfo.CurrentCulture,
                            "UPDATE models SET model=$${3}$$ WHERE
\"expandedDeviceType\"={0} AND \"deviceRevision\"={1} AND \"ddRevision\"={2}",
                            expandedDeviceType,
                            deviceRevision,
                            ddRevision,
                            model);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        /// <summary>
        /// Gets device description model from database
        /// </summary>
        /// <param name="expandedDeviceType">Expanded device type code</param>
        /// <param name="deviceRevision">Device revision</param>
        /// <param name="ddRevision">DD-file revision</param>
        /// <returns>Device description model</returns>
        public string GetModel(long expandedDeviceType, byte deviceRevision,
             byte ddRevision)
        {
            string result;

            using (NpgsqlConnection connection =
                new NpgsqlConnection(this.connectionString))
            {
                connection.Open();
```

```
                using (var cmd = new NpgsqlCommand())
                {
                    cmd.Connection = connection;
                    cmd.CommandText =
                        string.Format(
                            CultureInfo.CurrentCulture,
                            "SELECT model FROM models WHERE \"expandedDeviceType\"={0} AND
\"deviceRevision\"={1} AND \"ddRevision\"={2}",
                            expandedDeviceType,
                            deviceRevision,
                            ddRevision);

                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            result = reader.GetString(0);
                        }
                        else
                        {
                            result = string.Empty;
                        }
                    }
                }
            }

            return result;
        }
    }
}
```

ПРИЛОЖЕНИЕ А4

Листинг фрагмента исходного кода компонента системы DDParser

```
namespace HartOnline.DDParser.FmParsing
{
    using System;
    using System.CodeDom.Compiler;
    using System.Collections.Generic;
    using System.Globalization;
    using System.IO;
    using System.Text.RegularExpressions;

    using HartOnline.DDParser.FmParsing.DDExtracting;
    using HartOnline.DDParser.FmParsing.MetadataExtracting;
    using HartOnline.DDParser.FmParsing.Resources;
    using HartOnline.DDParser.Model;

    /// <summary>
    /// Retrieves the DeviceModel metadata from the .fm6 and .fm8 files.
    /// </summary>
    public class FmParser : IParser
    {
        /// <summary>
        /// The Extract method retrieves the metadata
        /// </summary>
        /// <param name="dataSourcePath">A data source path</param>
        /// <returns>Hart specification metadata</returns>
        /// <exception cref="ArgumentException">The 'dataSourcePath' parameter is
              empty</exception>
        public DeviceModel ExtractMetadata(string dataSourcePath)
        {
            DeviceModel result = new DeviceModel();
            result.Errors = new List<string>();
            result.Warnings = new List<string>();

            string fullPath = null;
            DeviceDescription deviceDescription = new DeviceDescription();
            SymFileTable symFileTable = new SymFileTable();
            MainExtractor dataExtractor = new MainExtractor();

            // Get full path
            try
            {
                fullPath = Path.GetFullPath(dataSourcePath);
            }
            catch (PathTooLongException e)
            {
                // Error getting full path (or file name) from dataSourcePath: {0}. Exception:
                // {1}
                result.Errors.Add(string.Format(
                        CultureInfo.CurrentCulture,
                        Localization.strErrorGettingFullPath,
                        dataSourcePath,
                        e.Message));
            }
            catch (ArgumentException e)
            {
                // Error getting full path (or file name) from dataSourcePath: {0}. Exception:
                // {1}
                result.Errors.Add(string.Format(
                        CultureInfo.CurrentCulture,
```

```
                Localization.strErrorGettingFullPath,
                dataSourcePath,
                e.Message));
    }

    // Load device description
    if (result.Errors.Count == 0)
    {
        try
        {
            deviceDescription.Load(fullPath);
        }
        catch (IOException e)
        {
            // Error loading of device description from {0}. Exception: {1}
            result.Errors.Add(string.Format(
                    CultureInfo.CurrentCulture,
                    Localization.strErrorLoadingDD,
                    fullPath,
                    e.Message));
        }

        foreach (CompilerError error in deviceDescription.Errors)
        {
            if (error.IsWarning)
            {
                result.Warnings.Add(error.ErrorText);
            }
            else
            {
                result.Errors.Add(error.ErrorText);
            }
        }
    }

    // Load .sym file
    if (result.Errors.Count == 0)
    {
        string symFilePath = string.Format(
            CultureInfo.InvariantCulture,
            "{0}.sym",
            Regex.Match(fullPath, @"(.*)\.(fm6|fm8)").Groups[1].Value);

        try
        {
            symFileTable.Load(symFilePath);
        }
        catch (InvalidDataException e)
        {
            // Error loading .sym file from {0}. Exception: {1}
            result.Errors.Add(string.Format(
                    CultureInfo.CurrentCulture,
                    Localization.strErrorLoadingSymFile,
                    symFilePath,
                    e.Message));
        }
        catch (IOException e)
        {
            // Error loading .sym file from {0}. Exception: {1}
            result.Errors.Add(string.Format(
                    CultureInfo.CurrentCulture,
                    Localization.strErrorLoadingSymFile,
                    symFilePath,
                    e.Message));
```

```
                }
            }

            // Extract metadata
            if (result.Errors.Count == 0)
            {
                try
                {
                    dataExtractor.ExtractData(
                        deviceDescription,
                        symFileTable);
                }
                catch (Exception e)
                {
                    // Error extracting metadata. Exception: {0}
                    result.Errors.Add(string.Format(
                            CultureInfo.CurrentCulture,
                            Localization.strErrorExtractingMetadata,
                            e.Message));
                }
            }

            // Get device model
            if (result.Errors.Count == 0)
            {
                result.DDRevision = deviceDescription.HeaderInfo.DeviceId.DDRevision;
                result.DeviceRevision = deviceDescription.HeaderInfo.DeviceId.DeviceRevision;
                result.ManufacturerCode =
                    deviceDescription.HeaderInfo.DeviceId.Manufacturer.AsUInt32;
                result.DeviceTypeCode = deviceDescription.HeaderInfo.DeviceId.DeviceType;

                result.Variables = dataExtractor.ExtractedVariables;
                result.Commands = dataExtractor.ExtractedCommands;
                result.Arrays = dataExtractor.ExtractedReferenceArrays;
                result.Collections = dataExtractor.ExtractedCollections;

                foreach (CompilerError error in dataExtractor.Errors)
                {
                    if (error.IsWarning)
                    {
                        result.Warnings.Add(error.ErrorText);
                    }
                    else
                    {
                        result.Errors.Add(error.ErrorText);
                    }
                }
            }

            return result;
        }
    }
}
```

ПРИЛОЖЕНИЕ А5

Листинг фрагмента исходного кода компонента системы ConnectionManager

```
document.getElementById("connectionButton").addEventListener("click", soapRequest);

var string;

var frame = window.frames.sandBox;

function soapRequest() {
    var  str =   '<?xml version="1.0" encoding="UTF-8"?>' +
        '<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">' +
        '<s:Body>' +
        '<GetDeviceModel xmlns="http://tempuri.org/">' +
        '<expandedDeviceType>2490435</expandedDeviceType>' +
        '<deviceRevision>1</deviceRevision>' +
        '<ddRevision>4</ddRevision>' +
        '</GetDeviceModel>' +
        '</s:Body>' +
        '</s:Envelope>';


    var xhr = new XMLHttpRequest();

    xhr.open("POST", "http://localhost:80/RequestProcessingService", true);
    if(!xhr){
        console.log("XHR ne robit");
    }

    xhr.onload = function () {
       var result = xhr.responseText;
        var lastIndex = result.indexOf("</GetDeviceModelResult>");
        string = result.substring(146, lastIndex);

        // console.log(string);

        frame.postMessage(string, "*");
    }
    xhr.setRequestHeader("SOAPAction",
"http://tempuri.org/IRequestProcessingService/GetDeviceModel");
    xhr.setRequestHeader("Content-Type", "text/xml");
    xhr.send(str);
}
```

ПРИЛОЖЕНИЕ А6

Листинг фрагмента исходного кода компонента системы UserInterface (Client)

```
var sensor = JSON.parse(test);
console.log(sensor);

var myApp = angular.module('hOnline', [], function($provide) {
    $provide.decorator('$sniffer', function($delegate) {
        $delegate.history = false;
        return $delegate;
    });
});

var mes;


var messageHeandler = function listener(event) {
 mes = JSON.parse(event.data);
    sensor = mes;
    console.log(mes);

}

window.addEventListener("message", messageHeandler);



myApp.controller('sensorController', function($scope) {

    $scope.showBlock = false;
    $scope.sensors = sensor;

    $scope.hideRequest = false;
    $scope.requestStatus = "Hide";

    $scope.hideResponce = false;
    $scope.responceStatus = "Hide";

    $scope.treeStatus = "Show";

    $scope.soobsh = mes;


    $scope.chooseCommand  = function (obj) {
        console.log("i'm work!");
        $scope.currentCommand = obj.Transactions[0];
        console.log($scope.currentCommand);
    };

    $scope.showTree = function () {

        $scope.sensors = sensor;

        $scope.showBlock = !$scope.showBlock

        if($scope.showBlock == true)
        {
            $scope.treeStatus = "Hide";
        }
        else
        {
            $scope.treeStatus = "Show";
```

```
        }

        console.log("Функция вызвалась");
    };

    $scope.showRequest = function () {
        $scope.hideRequest = !$scope.hideRequest;

        if($scope.hideRequest == false)
        {
            $scope.requestStatus = "Hide";
        }
        else
        {
            $scope.requestStatus = "Show";
        }
    }

    $scope.displayVariableName = function (string) {
        return $scope.sensors.Variables[string].Name;
    }

    $scope.displayVariableValue = function (string) {
        return $scope.sensors.Variables[string].Value;
    }

    $scope.showResponce = function () {
        $scope.hideResponce = !$scope.hideResponce;

        if($scope.hideResponce == false)
        {
            $scope.responceStatus = "Hide";
        }
        else
        {
            $scope.responceStatus = "Show";
        }
    }
});


myApp.directive("showCommand", function () {
    return {
        templateUrl: "htmlElements/mainContent.html",
        link: function (scope, element, attrs) {
            console.log("I'm here!");
            console.log(scope);
        }
    }
});
```

ПРИЛОЖЕНИЕ Б1

Графический пользовательский интерфейс утилиты по преобразованию DD-файлов в JSON

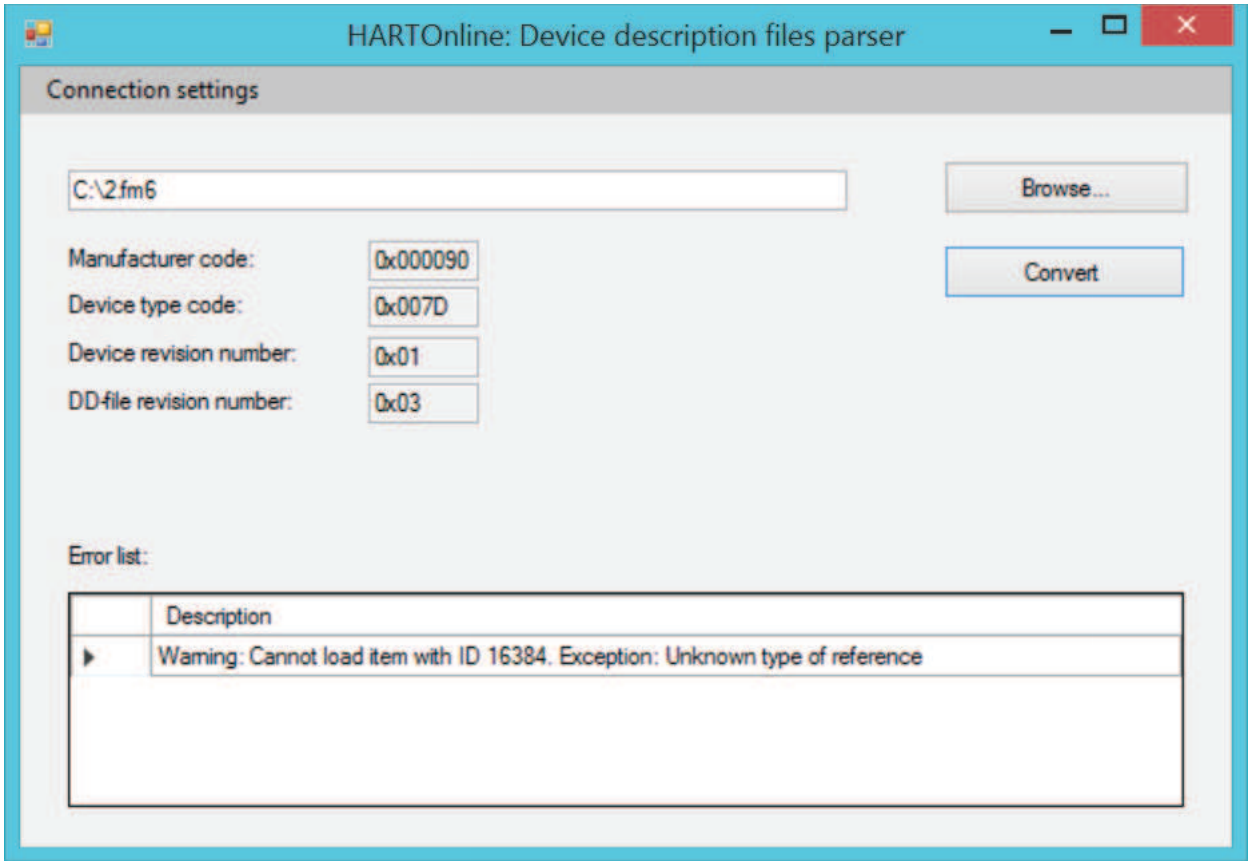

Рисунок 8 - Графический пользовательский интерфейс утилиты по преобразованию DD-файлов в JSON

ПРИЛОЖЕНИЕ Б2

Графический пользовательский интерфейс клиентской части системы

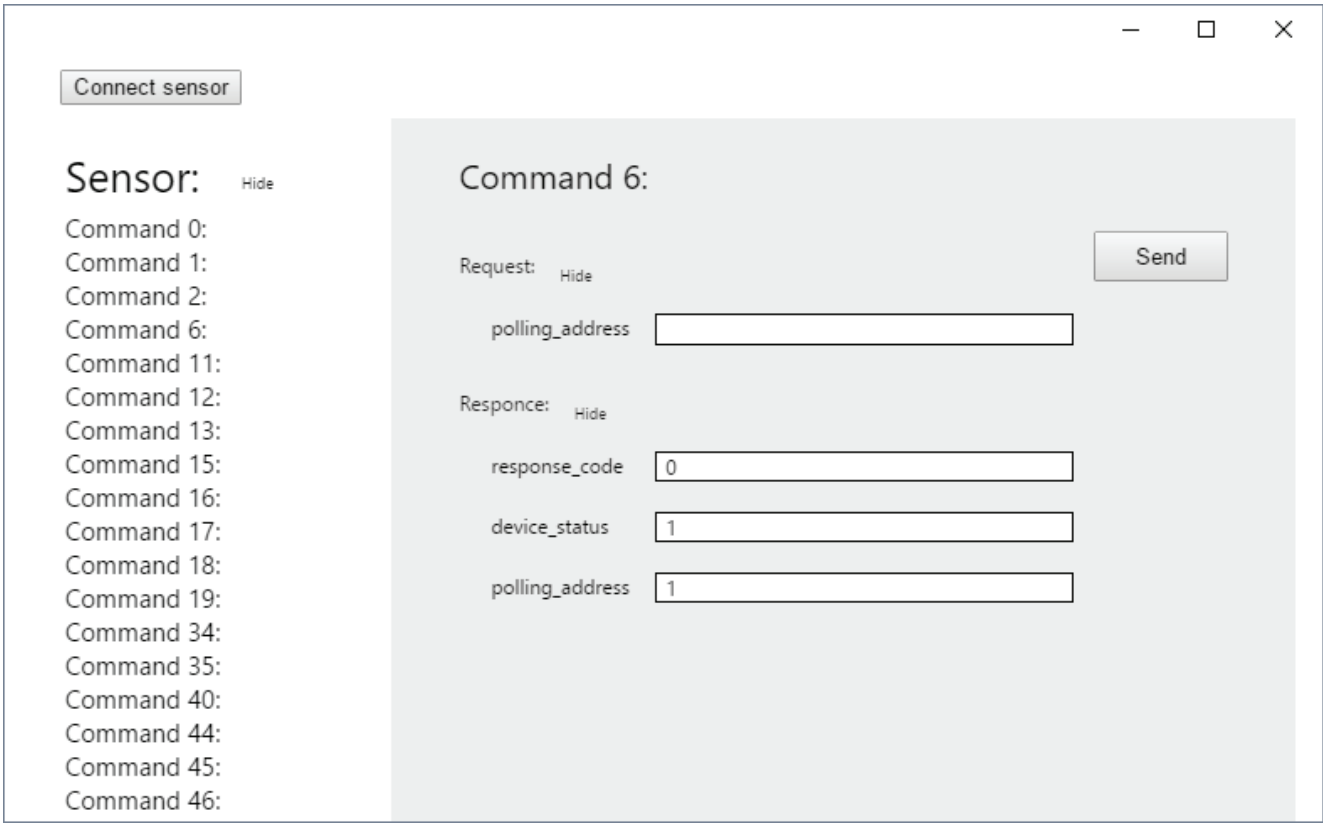

Рисунок 9 - Графический пользовательский интерфейс клиентской части системы

ПРИЛОЖЕНИЕ В

Архитектура разрабатываемой системы

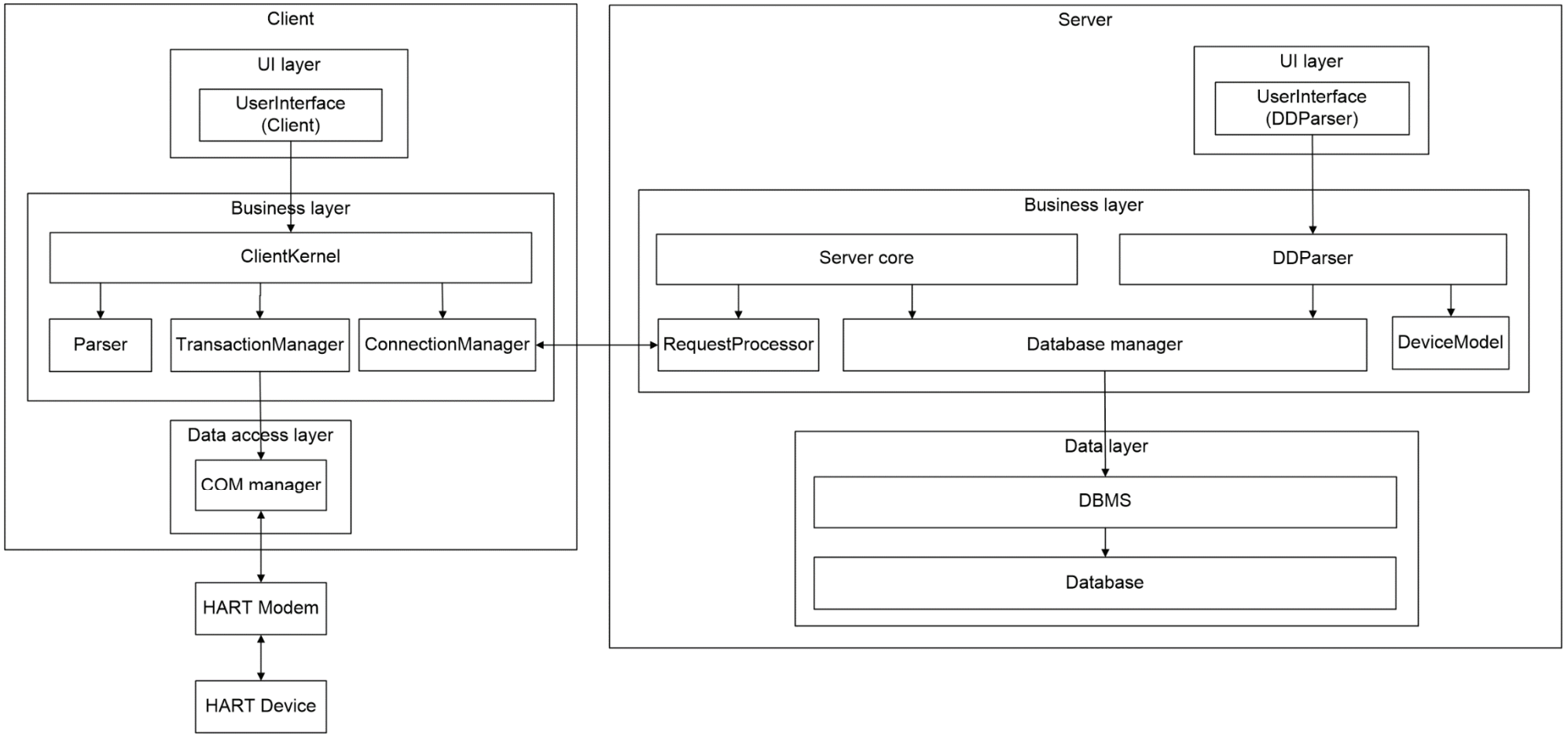

Рисунок 10 - Архитектура разрабатываемой системы