Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Отчет по производственной практике

Выполнил:

студент группы 253504

Ткачёв К.С.

Руководитель от вуза:

Летохо А.С.

Руководитель от предприятия:

Минченко Д.Л.

Минск 2016

Содержание

[Введение 3](#__RefHeading___Toc750_450552699)

[Индивидуальное задание 3](#__RefHeading___Toc752_450552699)

[Структура предприятия 4](#__RefHeading___Toc754_450552699)

[Сведения о программной платформе и проекте 6](#__RefHeading___Toc756_450552699)

[Программная платформа 6](#__RefHeading___Toc758_450552699)

[WITS (Wellsite Information Transfer Specification)  10](#__RefHeading___Toc760_450552699)

[CoilCat 11](#__RefHeading___Toc762_450552699)

[OPC 12](#__RefHeading___Toc766_450552699)

[RabbitMQ 13](#__RefHeading___Toc768_450552699)

[Data Interation Module 14](#__RefHeading___Toc770_450552699)

[Реализация индивидуального задания 16](#__RefHeading___Toc772_450552699)

[Мастер настроек 17](#__RefHeading___Toc774_450552699)

[Заключение 19](#__RefHeading___Toc778_450552699)

[Список использованных материалов 20](#__RefHeading___Toc780_450552699)

[Исходные код модуля для коммуникации 21](#__RefHeading___Toc782_450552699)

1. Введение
   1. Индивидуальное задание

**Тема:** Разработка коммуникационного модуля для синхронизации с протоколами для обмена данными между модулями других нефтедобывающих компаний.

**Задачи:**

* Ознакомиться с основными понятиями предметной области.
* Изучить структуру протоколов других компаний.
* Изучить на практике возможности программной платформы и инструментария для работы с данными протоколами.
* Реализация модуля коммуникации.
* Проектирование модуля оповещения и логирования сообщений.
* Реализация этого модуля.
* Объединения модуля коммуникации и модуля оповещения.
* Тестирование.
  1. Структура предприятия

ЧУП «ИНСОФТ ИНЖИНИРИНГ» - компания в сегменте разработки программного обеспечения для нефтедобывающих, банковских и других организаций, а также динамически развивающийся аутсорсер.

Основными заказчиками ЧУП «ИНСОФТ ИНЖИНИРИНГ» являются предприятия малого и среднего бизнеса Северной Америки, Западной Европы а также России. Компания специализируется на разработке программных продуктов и аутсорсинга, предлагая широкий спектр услуг, включая IT-консалтинг, управление жизненным циклом программного обеспечения, системную интеграцию и научные исследования в сфере бизнеса.

Работники компании имеют обширный опыт в разработке решений, которые отвечают всем потребностям клиентов и даже превышают их ожидания. На протяжении многих лет компания сумела создать прочную команду IT-специалистов. Многие сотрудники ЧУП «ИНСОФТ ИНЖИНИРИНГ» имеют ученые степени в области компьютерных наук, что позволяет компании использовать научные подходы и знания в решении задач в области бизнеса.

Компания постоянно совершенствует стандарты качества предлагаемых продуктов и услуг. Все это делается с целью сократить время производства и минимизировать возможные риски, связанные с разработкой комплексных программных продуктов.

Для разработки своих продуктов компания использует следующие технологии: Java SE/EE, мобильные технологии, .NET Framework, AdobeFlex/Flash, PHP Frameworks.

Один из крупнейших проектов компании проект “Albacore” для нефтедобывающей компании.

Организация в своей деятельности руководствуется Законами Республики Беларусь, указами Президента Республики Беларусь, другими нормативными актами и уставом компании ЧУП «ИНСОФТ ИНЖИНИРИНГ».

Год основания компании – 2002.

1. Сведения о программной платформе и проекте
   1. Программная платформа

**.NET Framework**

Платформа .NET Framework — это интегрированный компонент Windows, который поддерживает создание и выполнение нового поколения приложений и веб-служб XML. При разработке платформы .NET Framework учитывались следующие цели.

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Обеспечение среды выполнения кода, минимизирующей конфликты при развертывании программного обеспечения и управлении версиями.
* Обеспечение среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем.
* Обеспечение среды выполнения кода, исключающей проблемы с производительностью сред выполнения сценариев или интерпретируемого кода.
* Обеспечение единых принципов работы разработчиков для разных типов приложений, таких как приложения Windows и веб-приложения.
* Разработка взаимодействия на основе промышленных стандартов, которое обеспечит интеграцию кода платформы .NET Framework с любым другим кодом.

Двумя основными компонентами платформы .NET Framework являются общеязыковая среда выполнения (CLR) и библиотека классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Другой основной компонент платформы .NET Framework, библиотека классов, представляет полную объектно-ориентированную коллекцию типов, которые применяются для разработки приложений, начиная от обычных, запускаемых из командной строки или с графическим интерфейсом пользователя, и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как Web Forms и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

**WPF**

*Windows Presentation Foundation* — система для построения клиентских приложений [Windows](http://ru.wikipedia.org/wiki/Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) (начиная с версии [3.0](http://ru.wikipedia.org/wiki/.NET_Framework_3.0)), использующая язык [XAML](http://ru.wikipedia.org/wiki/XAML)[.](http://ru.wikipedia.org/wiki/Windows_Presentation_Foundation" \l "cite_note-2)

WPF предустановлена в [Windows Vista](http://ru.wikipedia.org/wiki/Windows_Vista) ([.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) 3.0), [Windows 7](http://ru.wikipedia.org/wiki/Windows_7) ([.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) 3.5 SP1), [Windows 8](http://ru.wikipedia.org/wiki/Windows_8)(.NET Framework 4.0 и 4.5), [Windows 8.1](http://ru.wikipedia.org/wiki/Windows_8.1) (.NET Framework 4.5.1). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в [браузере](http://ru.wikipedia.org/wiki/Браузер) приложений[.](http://ru.wikipedia.org/wiki/Windows_Presentation_Foundation" \l "cite_note-msdn_aa970268-3)

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык [XAML](http://ru.wikipedia.org/wiki/XAML) (Extensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление[.](http://ru.wikipedia.org/wiki/Windows_Presentation_Foundation" \l "cite_note-msdn_aa970268-3)

Графической технологией, лежащей в основе WPF, является [DirectX](http://ru.wikipedia.org/wiki/DirectX), в отличие от [Windows Forms](http://ru.wikipedia.org/wiki/Windows_Forms), где используется [GDI](http://ru.wikipedia.org/wiki/GDI)/[GDI+](http://ru.wikipedia.org/wiki/GDI%2B)[[4]](http://ru.wikipedia.org/wiki/Windows_Presentation_Foundation" \l "cite_note-ProWPF_-4). Производительность [WPF](http://ru.wikipedia.org/wiki/WPF) выше, чем у[GDI+](http://ru.wikipedia.org/wiki/GDI%2B) за счёт использования аппаратного ускорения графики через [DirectX](http://ru.wikipedia.org/wiki/DirectX).

Также существует урезанная версия [CLR](http://ru.wikipedia.org/wiki/CLR), называющаяся [WPF/E](http://ru.wikipedia.org/wiki/WPF/E), она же известна как [Silverlight](http://ru.wikipedia.org/wiki/Silverlight).

[XAML](http://ru.wikipedia.org/wiki/XAML) представляет собой [XML](http://ru.wikipedia.org/wiki/XML), в котором фактически реализованы классы [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework). Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на [элементы управления](http://ru.wikipedia.org/wiki/Элемент_управления) второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента.

Для работы с WPF требуется любой [.NET](http://ru.wikipedia.org/wiki/.NET_Framework)-совместимый язык. В этот список входит множество языков: [C#](http://ru.wikipedia.org/wiki/C_Sharp), [VB](http://ru.wikipedia.org/wiki/VB), [C++](http://ru.wikipedia.org/wiki/Visual_C%2B%2B), [Ruby](http://ru.wikipedia.org/wiki/IronRuby), [Python](http://ru.wikipedia.org/wiki/IronPython), [Delphi (Prism)](http://ru.wikipedia.org/wiki/Oxygene_(язык_программирования)), [Lua](http://ru.wikipedia.org/w/index.php?title=Lua.NET&action=edit&redlink=1) и многие другие. Для полноценной работы может быть использована как [Visual Studio](http://ru.wikipedia.org/wiki/Visual_Studio), так и [Expression Blend](http://ru.wikipedia.org/wiki/Expression_Blend). Первая ориентирована на программирование, а вторая — на дизайн и позволяет делать многие вещи, не прибегая к ручному редактированию [XAML](http://ru.wikipedia.org/wiki/XAML). Примеры этому — анимация, стилизация, состояния, создание элементов управления и так далее.

* 1. WITS (Wellsite Information Transfer Specification)

телекоммуникационный протокол передачи данных бурения.  Был разработан в середине 1980-х годов для обеспечения обмена информацией на буровой, и предоставления данных станциям удалённого мониторинга бурения.

В создании принимали участие: Baker Hughes, INTEQ, Schlumberger, Anadrill, Halliburton, Sperry Sun, MD/Totco, Experience.

WITS является многоуровневым протоколом, который предлагает простоту на низких уровнях и возрастающую гибкость формирования более высоких уровней репрезентации данных. На низких уровнях используется заранее определённый текстовой поток формата ASCII, в то же время на самом высшем уровне поток данных может быть настроен  для любого вида представления информации.

* 1. **CoilCat**
  2. Автоматизированная система операций с ГНКТ CoilCAT, интегрированная с сервисами «Шлюмберже» предусмотренными для ГНКТ, является уникальной системой управления данными, полученными на буровой площадке, позволяющей принимать оперативные решения с высокой степенью уверенности. При выполнении анализа и проектирования система не ограничивается одним лишь получением необходимых данных, что повышает эффективность и надежность внутрискважинных операций. Система CoilCAT осуществляет прямое соединение компьютера с многочисленными датчиками установки ГНКТ для получения всех необходимых параметров. Система объединяет процесс проектирования, исполнения и оценки операции в реальном времени, включая модули по проектированию и расчету параметров ГНКТ CoilCADE, что позволяет анализировать возникающие осложнения, формулировать технологические решения и прогнозировать конечный результат изменения рабочих параметров в установленных для величин давления и натяжения пределах. Пользователи получают возможность сопоставлять проектные и оперативные рабочие параметры и при необходимости изменять их. Система предоставляет доступ к оперативным данным по глубине ГНКТ, рабочей скорости, массе трубы, совокупному объему и давлению. В системе предусмотрена функция мониторинга глубины трубы, массы и рабочей скорости, величины давления в затрубном пространстве, устьевого давления и давления закачки, усилия обжима, скорости нагнетания и целостности трубы, а также оборудование для регистрации данных по ГНКТ.
  3. OPC

OPC – это набор повсеместно принятых спецификаций, предоставляющих универсальный механизм обмена данными в системах контроля и управления. Аббревиатура OPC традиционно расшифровывается как OLE for Process Control.  OLE – Object Linking and Embedding (связывание и встраивание объектов).

OPC-сервер – программа, получающая данные во внутреннем формате устройства или системы и преобразующая эти данные в формат OPC. OPC-сервер является источником данных для OPC-клиентов. По своей сути OPC-сервер – это некий универсальный драйвер физического оборудования, обеспечивающий взаимодействие с любым OPC-клиентом.

OPC-клиент – программа, принимающая от OPC-серверов данные в формате OPC.

Технология OPC определяет интерфейс между OPC-клиентом и OPC-серверами.

* 1. RabbitMQ

**RabbitMQ** — платформа, реализующая систему [обмена сообщениями](https://ru.wikipedia.org/wiki/Обмен_сообщениями) между компонентами программной системы ([Message Oriented Middleware](https://ru.wikipedia.org/wiki/Message_Oriented_Middleware)) на основе стандарта [AMQP](https://ru.wikipedia.org/wiki/AMQP) (Advanced Message Queuing Protocol). RabbitMQ выпускается под [Mozilla Public License](https://ru.wikipedia.org/wiki/Mozilla_Public_License).

RabbitMQ создан на основе испытанной [Open Telecom Platform](https://ru.wikipedia.org/wiki/Open_Telecom_Platform), обеспечивающий высокую надёжность и производительность промышленного уровня и написан на языке [Erlang](https://ru.wikipedia.org/wiki/Erlang).

RabbitMQ состоит из:

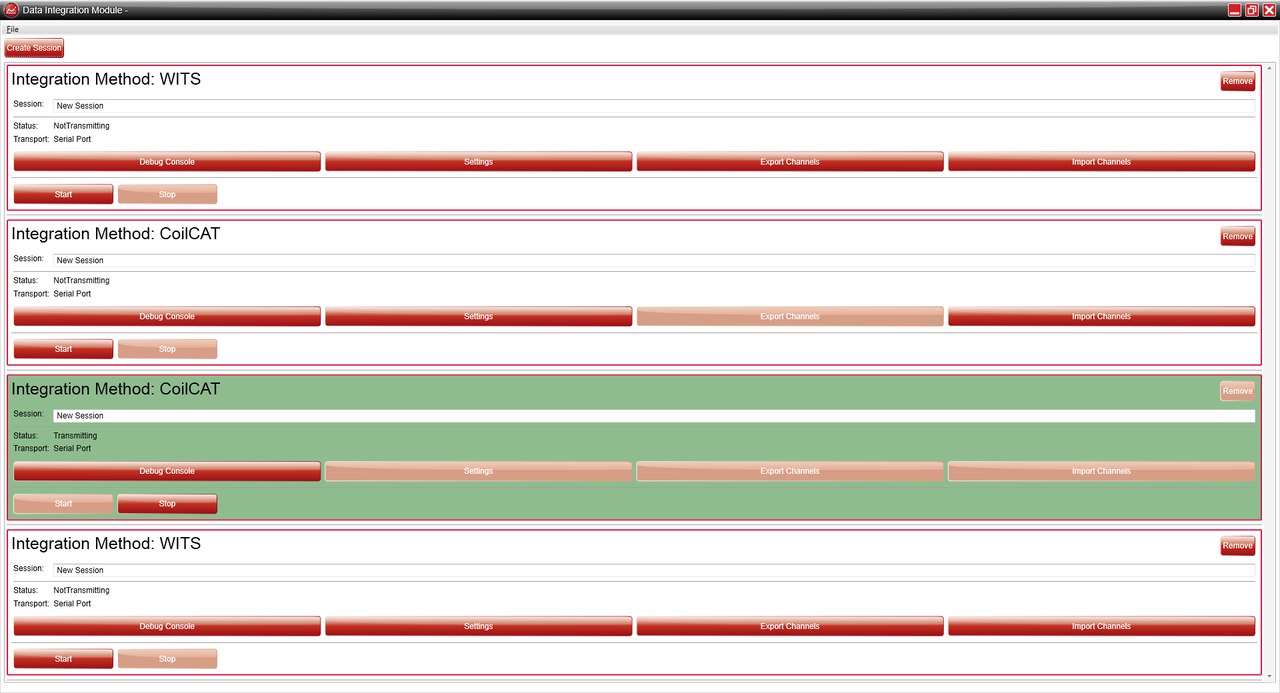
* Сервера RabbitMQ
* Поддержки протоколов [HTTP](https://ru.wikipedia.org/wiki/HTTP), [XMPP](https://ru.wikipedia.org/wiki/XMPP) и [STOMP](https://ru.wikipedia.org/w/index.php?title=STOMP&action=edit&redlink=1)
* Клиентских библиотек AMQP для [Java](https://ru.wikipedia.org/wiki/Java) и [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) (поддержка других языков программирования реализована в ПО других производителей)
* Различных плагинов (таких как плагины для мониторинга и управления через HTTP или веб-интерфейс или плагин «Shovel» для передачи сообщений между брокерами)

Поддерживается горизонтальное масштабирование для построения .

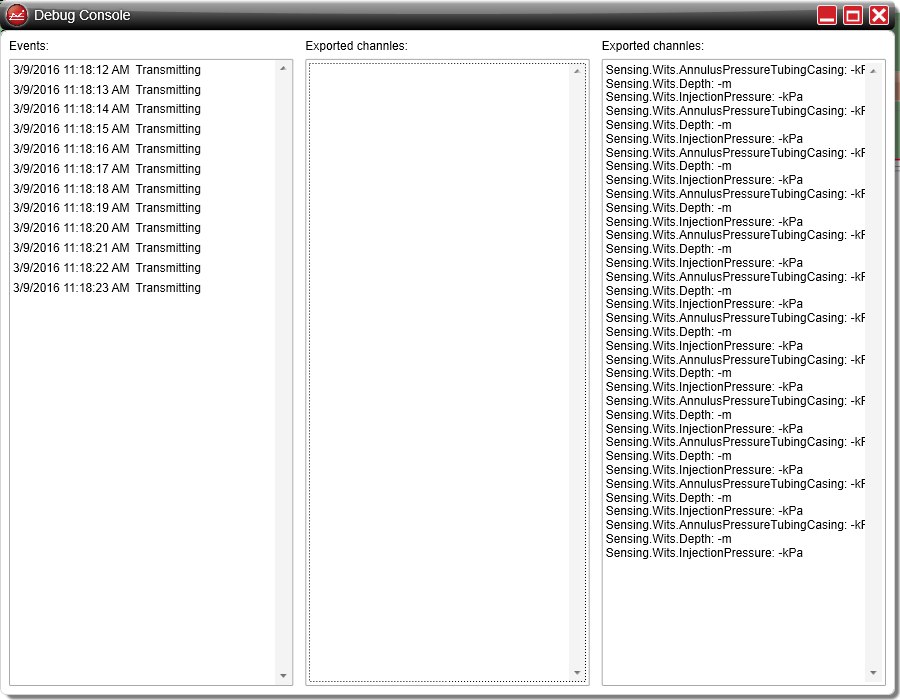
Имеется реализация клиентов для доступа к RabbitMQ для целого ряда языков программирования и платформ, широко используемых для веб-разработки: Java, .NET, [Perl](https://ru.wikipedia.org/wiki/Perl), [Python](https://ru.wikipedia.org/wiki/Python), [Ruby](https://ru.wikipedia.org/wiki/Ruby), [PHP](https://ru.wikipedia.org/wiki/PHP) и других.

* 1. Data Interation Module

Data Integration Module - программное обеспечение для нефтедобывающей компании. Проект Data Integration Module служит для коммуникации с модулями и устройствами других компаний для совместной работы при бурении скважин. Модуль позволяет импортировать и экспортировать данные, а также логирует их.



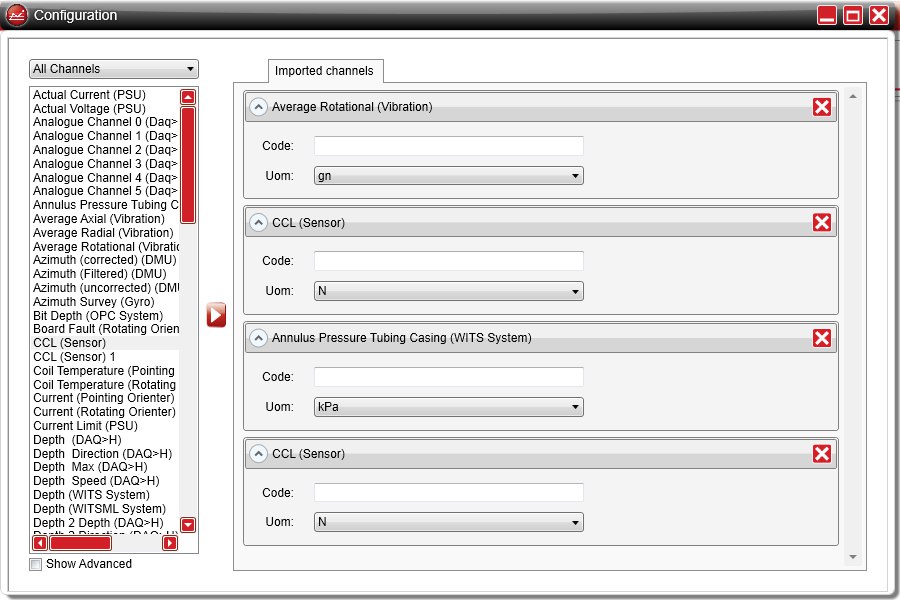
Главное окно

Окно отладки

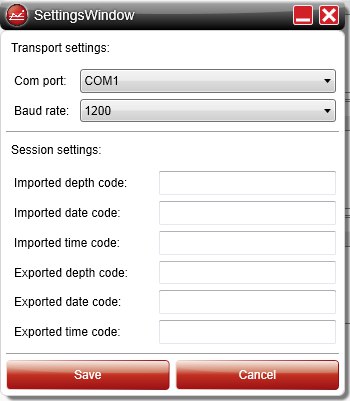
1. Реализация индивидуального задания

Индивидуально задание заключалось в необходимости реализации модуля для коммуникации с другими компании через модули, которые он используют в своём программном обеспечении. Так же необходимо было создать систему оповещения других модулей через систему сообщения RabbitMQ и модуль для сохранения и отображения текущей работы модуля.

Реализация подразумевал 3 этапа работ.

1. Реализация модуля для коммуникации с другим ПО.
2. Реализация системы для отображения статуса сессий.
3. Реализация модуля для оповещения.
   1. Мастер настроек
   2. Мастер настроек позволяет установить какие каналы и с какой системой измерения модуль будет импортировать или экспортировать данные.
   3. 

Окно настроек импорта и экспорта



Окно настроек сессии

1. Заключение

В ходе производственной практики была изучена специфика предметной области, изучены основы коммуникации с нефтедобывающими компаниями по средствами распространённых в нефтедобывающей промышленности протоколов.

Было успешно выполнено индивидуальное задание, состоящее в реализации модуля для коммуникации с другими компании через модули, которые эти компании используют в своём программном обеспечении. Данный модуль теперь является равноправным компонентом комплекса приложений Albacore.

В ходе практики были получены навыки работы по специальности. Был повышен уровень владения вышеизложенными технологиями разработки программного обеспечения, благодаря интересному и познавательному индивидуальному заданию, а также благодаря хорошем и ответственным руководителям практики.

Список использованных материалов

[1] Wits <http://wits.worldbank.org/>

[2] CoilCat, <http://www.slb.com/services/well_intervention/coiled_tubing/data_acquisition/coilcat.aspx>

[3] OPC, https://opcfoundation.org/

[4] Matthew MacDonald, [Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5](http://www.amazon.com/Pro-WPF-4-5-Presentation-Foundation/dp/1430243651/ref=sr_1_2?ie=UTF8&qid=1404084410&sr=8-2&keywords=wpf) – Apress, 2012

[5] RabbitMQ, https://habrahabr.ru/post/149694/

Исходные код модуля для коммуникации

Класс, который отвечает за парсинг данных в формате WITS:

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Globalization;

using System.Linq;

using System.Text;

using Clients.Core.Configuration.DataChannels;

using Core.CommonModel.InfoObjects;

using Core.Units;

namespace Clients.DataIntegrationModule.Core.Communication.MeasurementFormatters

{

public class WitsMeasurementsFormatter : IMeasurementsFormatter

{

#region Fields

private readonly object \_syncWitsObject = new object();

private readonly object \_syncObjectForExportedChannels = new object();

private string \_receivedWitsString = string.Empty;

private readonly string \_importedDateCode;

private readonly string \_importedTimeCode;

private readonly string \_exportedDepthCode;

private readonly string \_exportedDateCode;

private readonly string \_exportedTimeCode;

private readonly Dictionary<string, ChannelInfo> \_importedChannels;

private readonly Dictionary<string, ChannelInfo> \_exportedChannels;

private readonly IDictionary<string, List<MeasurementInfo>> \_exportWitsChannelMeasurements;

private static readonly TimeSpan \_timeBeforeDeleteMeasurement = new TimeSpan(0, 0, 10);

private static readonly TimeSpan \_timeActualDepthStamp = new TimeSpan(0, 0, 5);

#endregion

#region Events

public event EventHandler<MeasurementInfo> MeasurementInfoHandled;

#endregion

#region Constructors

public WitsMeasurementsFormatter(string importedDateCode, string importedTimeCode, string exportedDepthCode, string exportedDateCode,

string exportedTimeCode, Dictionary<string, ChannelInfo> importedChannels,

Dictionary<string, ChannelInfo> exportedChannels)

{

\_importedDateCode = importedDateCode;

\_importedTimeCode = importedTimeCode;

\_importedChannels = importedChannels;

\_exportedChannels = exportedChannels;

\_exportedDepthCode = exportedDepthCode;

\_exportedDateCode = exportedDateCode;

\_exportedTimeCode = exportedTimeCode;

\_exportWitsChannelMeasurements = \_exportedChannels.ToDictionary(x => x.Value.Routing,x => new List<MeasurementInfo>());

}

#endregion

#region Public methods

public void PushMeasurementInfoForExport(MeasurementInfo measurementInfo)

{

lock (\_syncObjectForExportedChannels)

{

#if TRACE

Trace.WriteLine(measurementInfo.RoutingKey);

#endif

if (\_exportWitsChannelMeasurements.ContainsKey(measurementInfo.RoutingKey))

{

MeasurementConversions.ConvertToCustomUnits(measurementInfo, \_exportedChannels.Single(x=>x.Value.Routing.Equals(measurementInfo.RoutingKey)).Value.UomId);

\_exportWitsChannelMeasurements[measurementInfo.RoutingKey].Add(measurementInfo);

}

}

}

public bool CheckMeasurementInfoForExport(MeasurementInfo measurementInfo)

{

return \_exportWitsChannelMeasurements.ContainsKey(measurementInfo.RoutingKey);

}

private DateTime GetWitsDateTime(string[] strings)

{

var dateString = strings.FirstOrDefault(s => s.Length > 4 && s.Substring(0, 4).Equals(\_importedDateCode));

var timeString = strings.FirstOrDefault(s => s.Length > 4 && s.Substring(0, 4).Equals(\_importedTimeCode));

if (dateString == null || timeString == null)

{

return default(DateTime);

}

try

{

var date = System.Convert.ToDateTime(dateString.Substring(4, dateString.Length - 5));

var time = System.Convert.ToDateTime(timeString.Substring(4, timeString.Length - 5));

var dateTime = new DateTime(date.Year, date.Month, date.Day, time.Hour, time.Minute, time.Second, time.Millisecond, DateTimeKind.Utc);

return dateTime;

}

catch (Exception)

{

return default(DateTime);

}

}

private MeasurementInfo GetMeasurementInfoFromWits(string str, DateTime dateTime)

{

if (str.Length < 5)

{

return null;

}

var codeString = str.Substring(0, 4);

if (!\_importedChannels.ContainsKey(codeString))

{

return null;

}

var value = GetWitsValue(str);

if (double.IsNaN(value))

{

return null;

}

var channel = \_importedChannels[codeString];

var uom =\_importedChannels.Single(x => x.Value.Routing.Equals(channel.Routing)).Value.UomId;

var measurementInfo = new MeasurementInfo(dateTime, channel.Routing, channel.UnitType, uom, value);

return measurementInfo;

}

private static double GetWitsValue(string str)

{

string valueString = str.Substring(4, str.Length - 5);

if (valueString == "-9999.99")

{

return double.NaN;

}

const NumberStyles style = NumberStyles.Number | NumberStyles.AllowCurrencySymbol;

var culture = CultureInfo.CreateSpecificCulture("en-GB");

double value;

return double.TryParse(valueString, style, culture, out value) ? value : double.NaN;

}

private void OnMeasurementInfoHandled(MeasurementInfo measurementInfo)

{

var handler = MeasurementInfoHandled;

if (handler != null)

handler(this, measurementInfo);

}

#endregion

}

}