МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Программа для мониторинга состояния вращающегося объекта

Студент 03.06.2020 Ежов Игорь Сергеевич
(подпись, дата)

Руководитель 03.06.2020 Дубинин В.Н.
(подпись, дата)

Нормоконтролёр Кучин А.В.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой Митрохин М.А.
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 17.06.2020 № 9)

Секретарь ГЭК Гурин Е.И.
(подпись)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«30» апреля 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Ежов Игорь Сергеевич гр. 16ВВ1 факультета вычислительной техники.

2. Тема работы: «Программа для мониторинга состояния вращающегося объекта».

Тема утверждена приказом ПГУ №0335а/о от "30" апреля 2020 г.

3. Руководитель бакалаврской работы д.т.н., профессор каф. ВТ ПГУ Дубинин В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать программу, реализующую опрос датчика частоты вращения ОХ 1 в соответствии с используемым протоколом информационного взаимодействия (ПИВ). Функционал программы должен обеспечивать как настройку ПО в части выбора подключаемого COM-порта, так приём и передачу данных с датчика ОХ 1.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, выбрать архитектуру приложения, определить язык программирования и среду разработки, разработать протокол информационного взаимодействия.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.2020 – 04.05.2020	
2.	Выбор архитектуры приложения	10	04.05.2020 – 07.05.2020	
3.	Выбор языка программирования и среды разработки	10	07.05.2020 – 14.05.2020	
4.	Разработка протокола информационного взаимодействия	10	14.05.2020 – 18.05.2020	
5.	Разработка приложения	20	18.05.2020 – 25.05.2020	
6.	Отладка и тестирование	20	25.05.2020 – 28.05.2020	
7.	Оформление выпускной работы	20	28.05.2020 – 03.06.2020	

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы _____ 03.06.2020 Дубинин В.Н.
(подпись, дата)

Задание к исполнению принял студент _____ 30.04.2020 Ежов И.С.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Выпускная квалификационная работа содержит 58 листов, 13 рисунков, 3 таблицы, 14 источников, 2 приложения.

COM-ПОРТ, QT, RS-232, ВРАЩАЮЩИЙСЯ ОБЪЕКТ, ВАЛ, ПЕРЕДАЧА ДАННЫХ, ЧАСТОТА.

Цель работы – разработать программу, предназначенную для опроса датчика частоты вращения ОХ 1 в соответствии с используемым протоколом информационного взаимодействия (ПИВ).

Объектом разработки является приложение, состоящее из программы с удобным для пользователя графическим интерфейсом, которая способна принимать входящие данные, выполнять их обработку и выводить для пользователя.

В результате проведённой работы реализован проект, полностью удовлетворяющий поставленной задаче, а именно, создано приложение, осуществляющее приём и обработку данных, принимаемых по COM-порту.

Данное приложение рекомендуется использовать для контроля параметров вращающегося объекта и анализа режимов его работы, что подразумевает сбор данных с вращающегося объекта с последующей их обработкой.

Внедрение разработанного модуля позволяет повысить эффективность и безопасность технологического процесса, а также общий коэффициент автоматизации системы, в рамках которой протекает указанный процесс.

Перечень принятых сокращений

ПИВ – Протокол информационного взаимодействия;

ПО – программное обеспечение;

COM-порт – Communications port (Последовательный порт);

RS-232 – Recommended Standard 232 (Стандарт физического уровня для асинхронного интерфейса);

LPT – Line Print Terminal (Параллельный порт, порт принтера);

UART – Universal Asynchronous Receiver-Transmitter (Универсальный асинхронный приёмопередатчик);

MOC – Meta Object Compiler (Мета-объектный компилятор);

GUI – Graphical User Interface (Графический пользовательский интерфейс).

Содержание

РЕФЕРАТ 2

Перечень принятых сокращений 3

Содержание 4

Введение 6

1. Анализ предметной области 8

1.1. Интерфейс передачи данных 8

1.1.1. COM-порт 8

1.1.2. Принцип работы последовательного порта и его отличие от параллельного. 8

1.1.3. Последовательный интерфейс RS-232 9

1.2. Выбор платформы для разработки и языка программирования 10

1.2.1. Описание Qt 10

1.2.2. Преимущество Qt над Visual Studio 11

1.3 Определения частоты вращения вала 13

2. Реализация 17

2.1. Разработка структуры программного обеспечения 17

2.2. Разработка программы 17

2.3. Протокол информационного взаимодействия 22

2.4. UML-диаграммы 23

2.4.1. UML-диаграмма классов 23

2.4.2. UML-диаграмма последовательности 24

2.4.3. UML-диаграмма вариантов использования 24

2.4.4. UML-диаграмма деятельности 25

3. Инструкция пользователя 27

4. Безопасность жизнедеятельности 33

Заключение 34

Список используемых источников 35

Приложение А. Листинги программы 37

Файл «mainwindow.h» 37
Файл «mainwindow.cpp» 39
Файл «main.cpp» 48
Приложение В. Презентация 49

Введение

В настоящее время человеком создано огромное количество различных устройств, каждое из которых выполняет свои поставленные цели и задачи. Они используются повсеместно – как в повседневной жизни, так и на каком-либо производстве. Многие устройства не имеют смысла своего существования без использования компьютера.

Так последовательный порт является одним из традиционных портов ввода-вывода компьютера, использовавшимся еще в первых ПК. Стандарт на него был разработан в 1969 году. COM-порты в операционной системе типа Windows – это именованные каналы для передачи данных, называемые обычно COM1, COM2 и т. д. по порядку обнаружения драйверов соответствующих устройств.

На многих производственных предприятиях используются данные, получаемые после измерения частоты скорости вращения вала (ротора), которые можно получить путём считывания скорости появления под ними зубцов.

Задача считывания информации и предоставления её для пользователя ложится непосредственно на программное обеспечение. Для её решения используются специальные программы, принимающие и обрабатывающие данные, которые получаются в результате опроса датчиков.

Целю данной работы является разработка программы, необходимой для приёма и обработки данных, получаемых по последовательному порту в результате опроса датчика. Входными данными должны быть пакеты, принимаемые от датчика в виде цифровых сигналов по интерфейсу COM. Выходными данными является численное отображение принятых данных с датчиков.

Функционал программы должен обеспечивать настройку программного обеспечения в части выбора подключаемого COM-порта, и приём и обработку данных с датчика ОХ 1.

Работа с файлами сейчас используется повсеместно. Поэтому для сохранения принятых значений частоты следует использовать обычный текстовый файл, из которого можно будет брать данные и проводить над ними соответствующие операции.

1. Анализ предметной области

В современной научно-технической литературе существует большое количество работ, посвящённых передаче данных по последовательным и параллельным портам. Для каждой проблемы всегда имеется несколько путей её решения, из которых нужно выбрать самый оптимальный. Данный раздел содержит обзор интерфейсов передачи данных: их сравнение, главные положительные качества и недостатки. Кроме того, обосновывается выбор того или иного технологического решения выполнения поставленной задачи, который является наиболее подходящим для решения проблемы.

1.1. Интерфейс передачи данных

1.1.1. COM-порт

В полученном задании основным методом подключения датчиков является доступ через COM-порт.

COM-port (Communication port, последовательный порт, Serial Port) – это стандартный двунаправленный последовательный порт компьютера, используется для передачи данных согласно протоколу RS-232.

Основные сферы применения COM-портов:

1. Подключение терминалов;
2. Внешних модемов;
3. Принтеров и плоттеров;
4. Мыши;
5. Прямое соединение двух компьютеров.

1.1.2. Принцип работы последовательного порта и его отличие от параллельного.

В отличие от параллельного (LPT) порта, последовательный порт передает данные побитно по одной-единственной линии, а не по нескольким одновременно. Последовательности битов группируются в серии данных,

начинающиеся стартовым битом и кончающиеся стоповым битом, а также битами контроля четности, использующимися для контроля ошибок.

Последовательный порт имеет две линии, по которым передаются собственно данные – это линии для передачи данных от терминала (ПК) к коммуникационному устройству и обратно. Кроме того, существует еще несколько управляющих линий. Обслуживает Serial port специальная микросхема UART, которая способна поддерживать относительно высокую скорость передачи данных, достигающую 115 000 бод (байт/с).

Порт использует электрические сигналы сравнительного высокого напряжения – до +15 В и -15 В. Уровень логического нуля последовательного порта составляет +12 В, а логической единицы – -12 В. Такой большой перепад напряжений позволяет гарантировать высокую степень помехоустойчивости передаваемых данных.

1.1.3. Последовательный интерфейс RS-232

Работа Serial port на ПК базируется на стандарте передачи данных для последовательных устройств RS-232. Этот стандарт описывает процесс обмена данными между телекоммуникационным устройством, например, модемом и компьютерным терминалом. Стандарт RS-232 определяет электрические характеристики сигналов, их назначение, длительность, а также размеры коннекторов и схему выводов для них. При этом RS-232 описывает лишь физический уровень процесса передачи данных и не касается используемых при этом транспортных протоколов, которые могут меняться в зависимости от используемого коммуникационного оборудования и программного обеспечения.

1.2. Выбор платформы для разработки и языка программирования

1.2.1. Описание Qt

Qt — это кроссплатформенный фреймворк для разработки ПО на языке программирования C++ (и не только).

Qt полностью объектно-ориентированная, кроссплатформенная. Дает возможность разрабатывать платформо-независимое ПО, написанный код можно компилировать для Linux, Windows, Mac OS X и других операционных систем. Включает в себя множество классов для работы с сетью, базами данных, классы-контейнеры, а также для создания графического интерфейса и множество других.

Qt использует MOC (Meta Object Compiler) для предварительной компиляции программ. Исходный текст программы обрабатывается MOC, который ищет в классах программы макрос Q_OBJECT и переводит исходный код в мета-объектный код, после чего мета-объектный код компилируется компилятором C++. MOC расширяет функциональность фреймворка, благодаря ему добавляются такие понятия, как слоты и сигналы.

В Qt имеется огромный набор виджетов (Widget), таких как: кнопки, прогресс бары, переключатели, checkbox, и другие – они обеспечивают стандартную функциональность GUI (графический интерфейс пользователя). Позволяет использовать весь функционал пользовательского интерфейса – меню, контекстные меню, drag&drop.

Основной средой разработки для Qt является Qt Creator. Данная среда включает в себя Qt Designer, с помощью которого можно создавать графический интерфейс. Визуальное создание интерфейса позволяет легко и просто создавать интерфейс, перетаскивая различные виджеты (выпадающие списки, кнопки, переключатели) на форму.

Qt поставляется вместе с Qt Assistant – интерактивным справочником, содержащим в себе информацию по работе с Qt. В состав Qt также входит Qt Linguist, которая позволяет локализировать приложение для разных языков.

QtWidgets — это объект пользовательского интерфейса: он получает события мыши, клавиатуры и другие события от оконной системы и рисует свое изображение на экране. Каждый виджет имеет прямоугольную форму, и все они отсортированы в порядке наложения (Z-order). Виджет ограничен своим родителем и другими виджетами, расположенными перед ним.

QtOpenGL — это специальный модуль для работы с OpenGL находящийся в библиотеке QT. Модуль QtOpenGL реализован как платформонезависимая Qt/C++ надстройка (стандартизация внешних обращений) над платформозависимыми GLX (OpenGL в Linux), WGL (OpenGL в Windows) или AGL (OpenGL в MacOS) API.

QtUiTools – это модуль, который предоставляет классы для обработки форм, созданных с помощью Qt Designer. Эти формы обрабатываются во время выполнения для получения динамически сгенерированных пользовательских интерфейсов. Для того чтобы сгенерировать форму во время выполнения необходим файл ресурсов, содержащий в себе файл .ui.

QML - декларативный язык программирования, основанный на JavaScript, предназначенный для дизайна приложений, делающих основной упор на пользовательский интерфейс. Является частью Qt Quick, среды разработки пользовательского интерфейса, распространяемой вместе с Qt. В основном используется для создания приложений, ориентированных на мобильные устройства с сенсорным управлением.

1.2.2. Преимущество Qt над Visual Studio

Основные преимущества Qt над Visual Studio представлены ниже (Таблица 1).

Таблица 1 – Преимущества Qt над Visual Studio

Параметр	QT Creator	Visual Studio
Поддержка версий	Проект, разработанный в более поздней версии программы QT Creator можно без усилий запустить на ранней версии.	Что бы запустить проект, на более старой версии, который собран на новой версии, нужно создать новый проект и переместить туда все данные.
Метод распространения	Есть возможность использовать QT Creator бесплатно по лицензии LGPL. Данная лицензия предназначена для разработки не коммерческого ПО и обучению пользоваться данным фреймворком.	В Visual Studio необходимо сразу купить полную версию чтобы была возможность создавать разные оконные приложения.
Метод построения интерфейсов	Позволяет создавать графические интерфейсы пользователя при помощи ряда инструментов. Существует панель инструментов «Панель виджетов», в которой доступны для использования элементы интерфейса — виджеты, такие как, например, «выпадающий список» ComboBox, «поле ввода» LineEdit, «кнопка» PushButton и многие другие. Qt Designer — кроссплатформенная свободная среда для разработки графических интерфейсов (GUI) программ,	WinAPI – это 32-разрядный API для современных версий Windows. Базовые функции реализованы в динамически подключаемых библиотеках kernel32.dll и advapi32.dll Win32 – это архитектура операционной системы и приложений, поддерживаемых в ней для запуска и работы.

Продолжение таблицы 1

Параметр	QT Creator	Visual Studio
	использующих библиотеку Qt. QtGui — компоненты графического интерфейса, основанные на визуальном представлении. QtWidgets — это объект пользовательского интерфейса: он получает события мыши, клавиатуры и другие события от оконной системы и рисует свое изображение на экране. Модуль QtUiTools предоставляет классы для обработки форм, созданных с помощью Qt Designer. Эти формы обрабатываются во время выполнения для получения динамически сгенерированных пользовательских интерфейсов. QML – это декларативный язык программирования, основанный на JavaScript, предназначенный для дизайна приложений, делающих основной упор на пользовательский интерфейс.	Пакет MFC - библиотека, призванная облегчить разработку GUI-приложений для Microsoft Windows путём использования богатого набора библиотечных классов. wxWidgets - кроссплатформенная библи отека инструментов с открытым исходным кодом для разработки кроссплатформе нных на уровне исходного кода приложений. Основным применением wxWidgets является построение GUI.

1.3. Определения частоты вращения вала

Определим частоту вращения вала на примере электродвигателя

Требуемая частота вращения вала электродвигателя определяется по формуле

$n_{эд} = n_2 \cdot i$,

где i − передаточное отношение привода.

В дальнейших расчетах вместо передаточного отношения $i = n_{эд} / n_2$ применяют общее передаточное число привода uобщ

Общее передаточное число привода

$$u_{общ} = \frac{n_{дв}}{n_{в}} = u_1 \cdot u_2, \ldots,$$

где nдв – асинхронная частота вращения двигателя, мин-1;

nВ – частота вращения приводного вала рабочего органа, мин-1;

u1, u2 – передаточные числа элементов привода.

Частота вращения приводного вала составляет, например,

для ленточного транспортера:

$$n_{в} = \frac{60v}{\pi D},$$

для цепной передачи (звездочки):

$$n_{в} = \frac{60v \cdot 10^3}{z \cdot p},$$

где D – диаметр барабана или звездочки, м;

z – число зубьев звездочки;

p – шаг тяговой цепи, мм.

Применение u вместо i связано только с принятой формой расчетных зависимостей для контактных напряжений, значения которых не зависят от того, какое из зубчатых колес является ведущим.

Руководствуясь рекомендациями по выбору значений передаточных чисел в соответствии с заданным типом передачи в редукторе определяют возможный диапазон частот вращения вала электродвигателя

nэд = n2 · (umin…umax).

По рассчитанной мощности P и диапазону nэд выбирают электродвигатель таким образом, чтобы его номинальная мощность Pном≥P, а номинальная частота nном вращения вала была самой близкой (из возможных вариантов) к большему значению диапазона nэд. В этом случае размеры и стоимость электродвигателя будут наименьшими. При этом следует иметь в виду, что большая частота вращения вала электродвигателя при одинаковой мощности вызывает увеличение передаточного числа редуктора, а, следовательно, увеличение его длины и высоты. Меньшая частота вращения

вызывает увеличение размеров электродвигателя и увеличение ширины зубчатых колес, а, следовательно, уменьшение размеров редуктора.

Если скоростной диапазон достаточно большой, т.е. по скоростной характеристике можно выбрать несколько двигателей, окончательное решение принимается с учетом следующих соображений. Быстроходные двигатели легче и дешевле тихоходных, поэтому предпочтительнее. Однако выбор быстроходного двигателя приводит к увеличению общего передаточного отношения редуктора и, как правило, к увеличению его габаритов, массы и стоимости. Если позволяет скоростной диапазон, рекомендуется выбирать два двигателя с различной скоростной характеристикой и последующий расчет вести параллельно. В конце расчета производится анализ вариантов по кинематическим, технико-экономическим и другим признакам и выбирается окончательный вариант.

Одновременно необходимо учитывать рекомендуемые значения передаточных чисел различных типов передач. Значения передаточных чисел редуктора не должны выходить за пределы допускаемых отклонений, предусмотренных ГОСТ 12289-76.

По выбранному электродвигателю определяют расчетное передаточное число зубчатой передачи редуктора

$$u = \frac{n_{ном}}{n_2}.$$

При окружных скоростях более 6 м/с целесообразно применять колеса косозубые и шевронные.

Номинальные значения передаточных чисел в зубчатых редукторах общего назначения, выполненных в виде самостоятельных агрегатов по:

1-й ряд: 1,0; 1,25; 1,6; 2,0; 2,5; 3,15; 4,0; 5,0; 6,3; 8; 10; 12,5;

2-й ряд: 1,12; 1,4; 1,8; 2,24; 2,8; 3,55; 4,5; 5,6; 7,1; 9,0;11,2.

Примечание. 1-й ряд следует предпочесть второму ряду.

Угловая скорость вала электродвигателя

$$\omega_{дв} = \frac{\pi n_{дв}}{30},$$

Далее можно определить угловые скорости других валов привода

$$\omega_2 = \frac{\omega_{дв}}{u_1},$$

$$\omega_3 = \frac{\omega_2}{u_2}.$$

Крутящие моменты на валах определяются с учетом потерь на трение

$$T_1 = \frac{P_{тр} \cdot 10^3}{\omega_{дв}}.$$

Крутящий момент ведомого вала

$$T_2 = T_1 \cdot u_1 \cdot \eta_1.$$

2. Реализация

В данной главе описана реализация заданной программы. Особое внимание уделяется выделению о описанию различных сущностей, и их взаимодействию между собой. При этом освещены основные особенности реализации, связанные с программированием на языке C++ платформы Qt. Целью данного раздела является объяснение принципа работы и внутреннее устройство проектируемого ПО.

2.1. Разработка структуры программного обеспечения

Логическая организация данной программы представляет собой 2 взаимосвязанных блока (Рисунок 1).

Рисунок 1 – Структурная схема.

2.2. Разработка программы

Программа состоит из одного компонента: «OX1.exe» – исполняемый файл, точка входа в программу, обеспечивающая все функции.

Компонент «OX1.exe» состоит из файла исходных текстов «main.cpp» и одного класса, который включает в себя заголовочный файл и файл исходных

текстов – Класс MainWindow. Файл исходных текстов «main.cpp» содержит одну функцию - int main(int argc, char *argv[]) – точка входа в программу.

Класс MainWindow содержит в себе файлы mainwindow.h и mainwindow.cpp, включающие в себя описание свойств и методов класса и их реализацию (Таблица , **Error! Reference source not found.**), а также файл интерфейса mainwindow.ui.

Таблица 2 – Свойства класса MainWindow

№ п/п	Свойства класса MainWindow	Описание свойства класса MainWindow
1	Ui::MainWindow *ui	Указатель на пользовательский интерфейс, форму окна
2	QSerialPort *my_port = NULL	Указатель на экземпляр стандартного класса для работы с последовательным портом
3	QString p_name	Строковая переменная для хранения имени COM-порта
4	int rate	Целочисленная переменная для хранения значения скорости обмена данными
5	int filter	Целочисленная переменная для хранения ширины окна фильтра
6	int m	Целочисленная переменная для хранения количества вставок
7	double lastFilterData	Дробная переменная для хранения последнего значения, полученного в результате применения фильтра
8	bool isStopped	Булева переменная для хранения состояния опроса (запущен или остановлен)
9	bool isGraphic	Булева переменная для хранения состояния окна отображения данных (открыто или закрыто)
№ п/п	Свойства класса MainWindow	Описание свойства класса MainWindow
10	QString oneData	Строковая переменная для хранения информационной части одного полученного пакета данных

Продолжение таблицы 2

11	QVector oneDataPoint	Переменная контейнерного типа для хранения пакетов данных в формате строк
12	int oneIntData	Целочисленная переменная для хранения информационной части одного полученного пакета данных
13	OneSample os	Экземпляр класса OneSample для работы с одним полученным пакетом данных
14	QTimer *myStatusTime = NULL	Указатель на экземпляр стандартного класса для работы с таймером, реализующим циклическое обновление строки состояния
15	QTimer *myReadTime = NULL	Указатель на экземпляр стандартного класса для работы с таймером, реализующим временные рамки ожидания данных от датчика
16	QTimer *myReaderTimer = NULL	Указатель на экземпляр стандартного класса для работы с таймером, реализующим временные рамки чтения данных из буфера порта
17	QTimer *myShowTime = NULL	Указатель на экземпляр стандартного класса для работы с таймером, реализующим временные рамки обновления стоки состояния
18	QTime myTime	Экземпляр стандартного класса для хранения метки времени
19	QString statusMessage	Строковая переменная для хранения сообщения о текущем состоянии программы
20	QIntValidator *mValid = NULL	Указатель на экземпляр стандартного класса для проверки корректности вводимого целого числа
21	GraphDialog *ohGD = NULL	Указатель на экземпляр класса GraphDialog для реализации отображения графиков
22	QVector <double> graphData	Экземпляр стандартного класса для работы с двунаправленным списком, содержащим в себе дробные переменные для хранения данных для текущего отображения

Таблица 3 – Методы класса MainWindow

№ п/п	Методы, слоты и сигналы класса MainWindow	Описание методов, слотов и сигналов класса MainWindow
Методы		

Продолжение таблицы 3

1	explicit MainWindow(QWidget *parent = 0)	Конструктор класса
2	~MainWindow()	Деструктор класса
3	QString DeleteNulls(QString data)	Функция удаления нулей из строковой переменной. Возвращает отредактированную строку
4	QString AddNulls(QString data)	Функция добавления нулей к строковой константе с длиной меньше требуемой. Возвращает отредактированную строку
5	void ClearAll()	Функция очищения всех индикаторов окна
Сигналы		
1	void IHaveData(const QString&)	Сигнал появления требуемого количества данных в буфере порта и необходимости их обработки
2	void IHaveGraphData(const QVector<double>&, const unsigned long&)	Сигнал появления требуемого количества данных для отображения и необходимости их обработки.
3	void InCorrectData()	Сигнал появления некорректного пакета данных
Слоты		
1	void GetPort()	Слот получения списка доступных COM-портов
2	void SetPort(QAction *c)	Слот установки выбранного COM-порта в качестве используемого
3	void SetRate(QAction *r)	Слот установки выбранной скорости передачи данных в качестве используемой
4	void OpenPort()	Слот открытия COM-порта
5	bool isOpened()	Слот проверки результата открытия COM-порта
6	void ClosePort()	Слот закрытия COM-порта
7	void SetFilter(QAction *f)	Слот установки выбранного фильтра в качестве используемого

Продолжение таблицы 3

8	double GetFilterData(double data)	Слот получения фильтрованного значения из текущего. Возвращает фильтрованное значение
9	int ReadData()	Слот чтения данных из COM-порта
10	void GetData(QString data)	Слот обработки пакета прочитанных данных
11	void NoData()	Слот обработки ситуации отсутствия данных в буфере COM-порта
12	void StopData()	Слот обработки остановки процесса опроса датчика
13	bool IsDataStop()	Слот проверки результата остановки процесса опроса датчика
14	void INeedStop()	Слот обработки необходимости остановки процесса опроса датчика
15	void UpdateStatusBar()	Слот обновления информации в строке состояния
16	void NonGraphic()	Слот обработки отсутствия необходимости отображения данных
17	void IfDataIncorrect()	Слот обработки некорректного пакета данных
18	void on_btnStart_clicked()	Слот обработки нажатия кнопки «Старт»
19	void on_btnStop_clicked()	Слот обработки нажатия кнопки «Стоп»
20	void on_actionOpenGraphics_triggered( )	Слот обработки выбора пункта меню «Отображение» → «Графики»
21	void on_btnChangeM_clicked()	Слот обработки нажатия кнопки «Изменить»
22	void on_lineEditM_returnPressed()	Слот обработки изменения значения в поле «Количество вставок»
23	void ShowStringData()	Слот отображения данных на индикаторах окна
24	void on_btnFilter100_clicked ()	Слот обработки нажатия кнопки «Фильтр»

Продолжение таблицы 3

25	void on_lineEdFilterSize_returnPressed ()	Слот обработки изменения значения в поле «Фильтр»
26	void ClearFile()	Слот очистки файла
27	void ShowStats()	Слот вывода среднего значения частоты

2.3. Протокол информационного взаимодействия

Входными данными должны быть пакеты, принимаемые от датчика в виде цифровых сигналов по интерфейсу COM.

Выходными данными является численное отображение принятых данных с датчиков.

Ответ датчика представляется в виде команды, длиной 8 бит (Рисунок 2).

#	N	0	0	0	0	0	!

Рисунок 2 – Код команды.

Описание значащих разрядов:

– Старт-бит, признак начала команды

! – Стоп-бит, признак конца команды

N – Код команды

Датчик отправляет данные в виде пакета, содержащего в себе 3 вида команд:

N = 1 – код частоты

N = 2 – код ЦАП

N = 3 – код температуры

Частота отправления пакетов данных зависит от скорости вращения вала (ротора).

2.4. UML-диаграммы

2.4.1. UML-диаграмма классов

Программа содержит один основной класс MainWindow, представленный на UML-диаграмме классов (Рисунок 3).

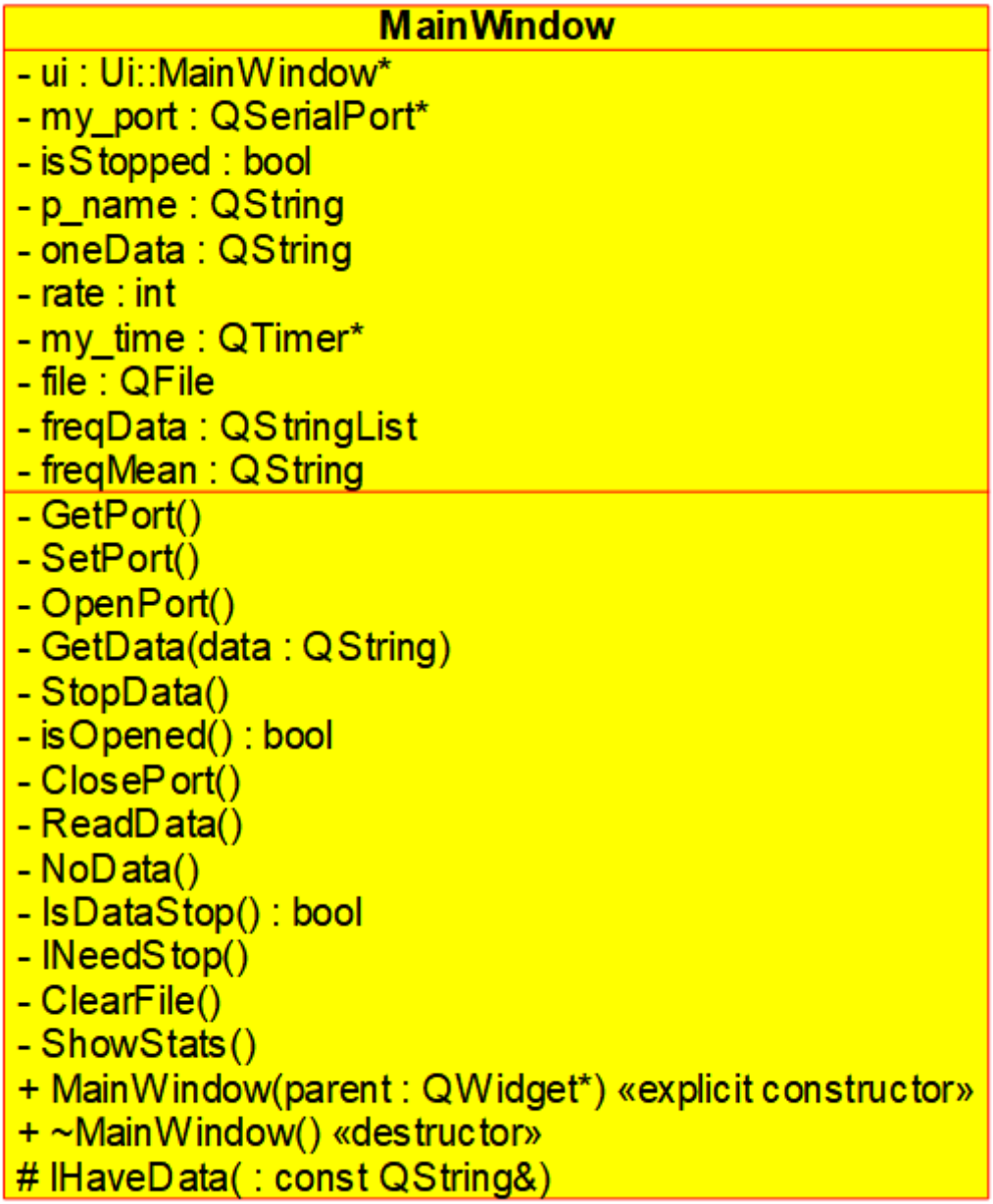


Рисунок 3 – UML-диаграмма кылассов.

2.4.2. UML-диаграмма последовательности

На основе описания работы приложения можно оставить UML-диаграмму последовательности (Рисунок 4). Она визуализирует действия, описанные выше.

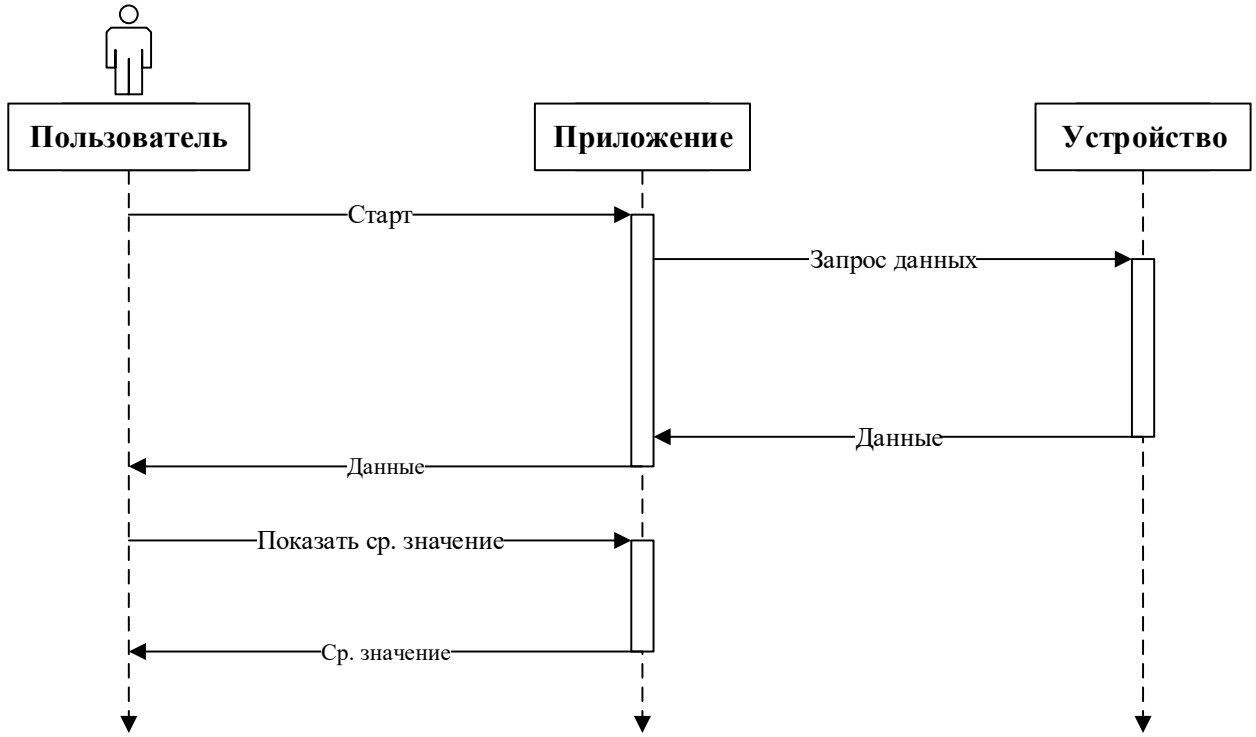


Рисунок 4 – UML-диаграмма последовательности.

2.4.3. UML-диаграмма вариантов использования

UML-диаграмма вариантов использования так же представлена (Рисунок 5). Она отражает возможности пользователя.

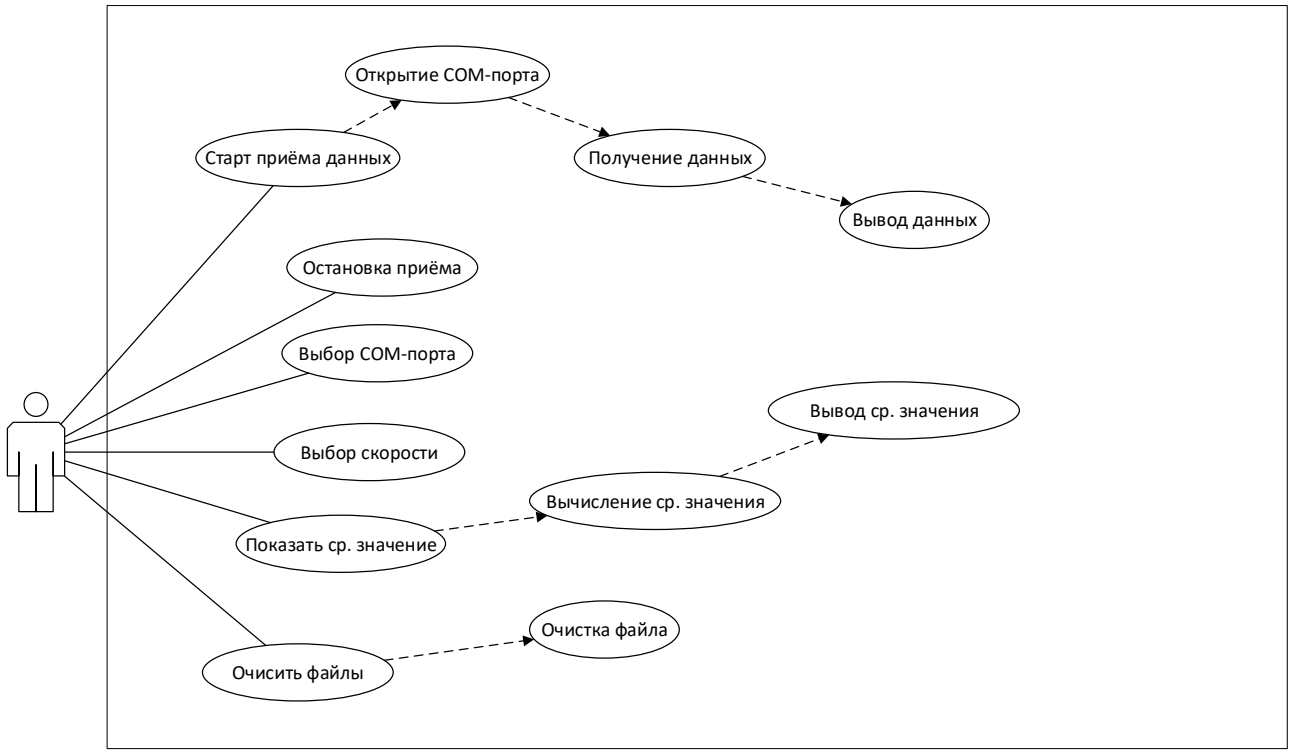


Рисунок 5 – UML-диаграмма вариантов использования.

2.4.4. UML-диаграмма деятельности

Диаграмма деятельности программы тоже имеет место быть (Рисунок 6). Она отражает жизненный цикл программы.

Из диаграммы деятельности можно выделить следующие состояния программы:

1. Состояние ожидания старта.
2. Состояние ожидания остановки.
3. Состояние ожидания данных.

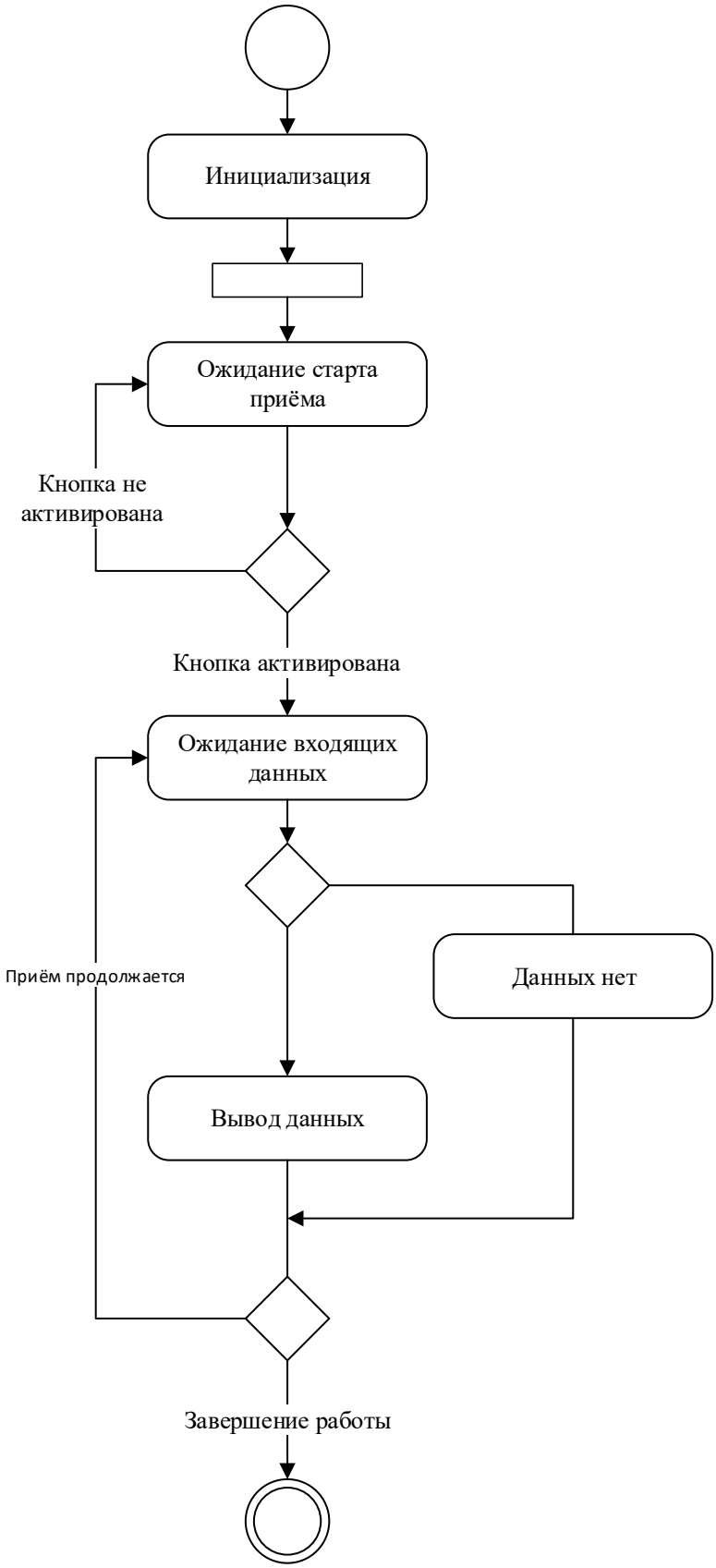


Рисунок 6 – UML-диаграмма деятельности.

3. Инструкция пользователя

После загрузки программы «OX1.exe» на экране должно появиться главное окно приложения (Рисунок 1).

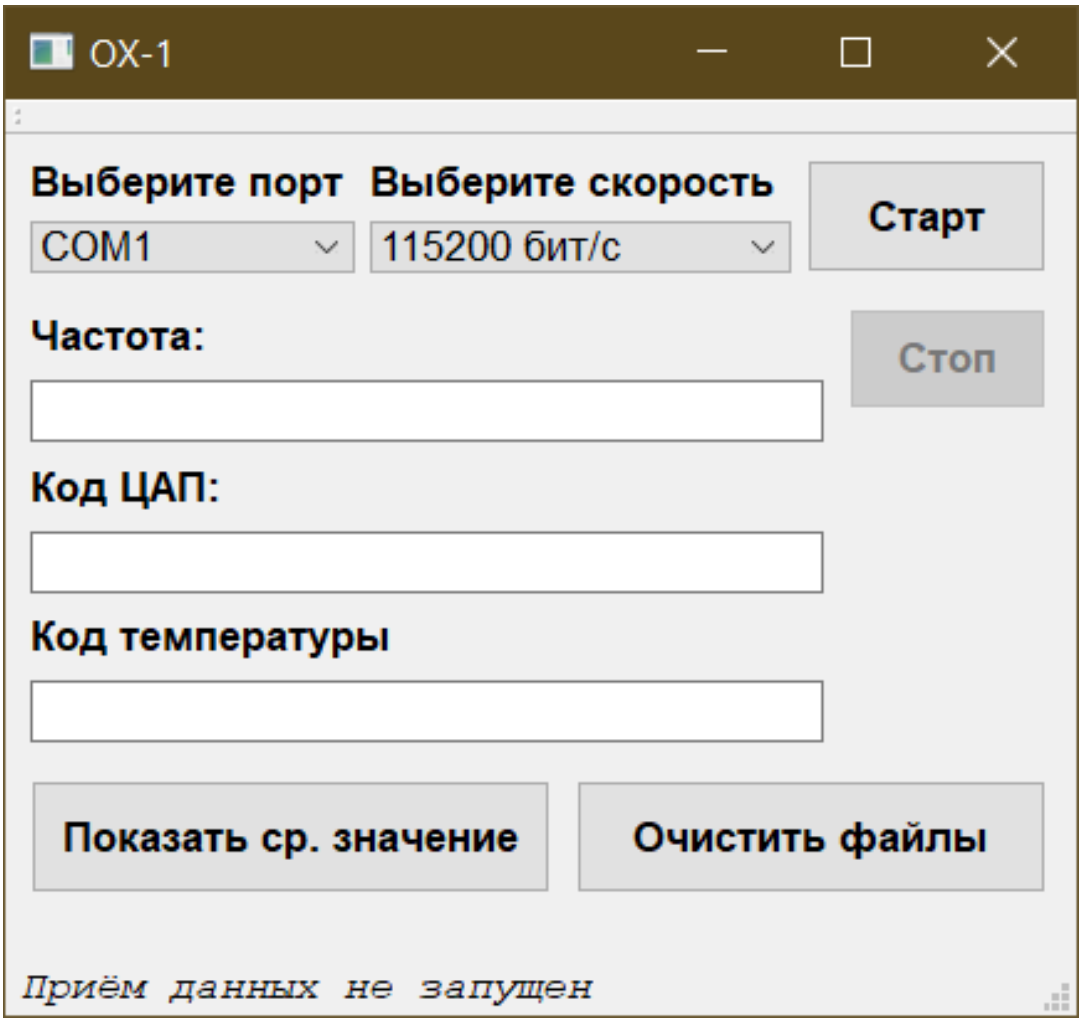


Рисунок 7 – Главное окно программы.

Внизу окна расположена строка состояния, отображающая информацию о текущем состоянии программы. При первом запуске в ней сообщается о том, что прием данных не запущен. Кнопки «Старт» и «Стоп» предназначены для запуска и остановки опроса датчика соответственно. Кнопка «Стоп» недоступна, если программа не начала принимать данные. Программа содержит меню с двумя функциями. Первая – «Выберите порт» – предоставляет список доступных COM-портов (Рисунок 8).

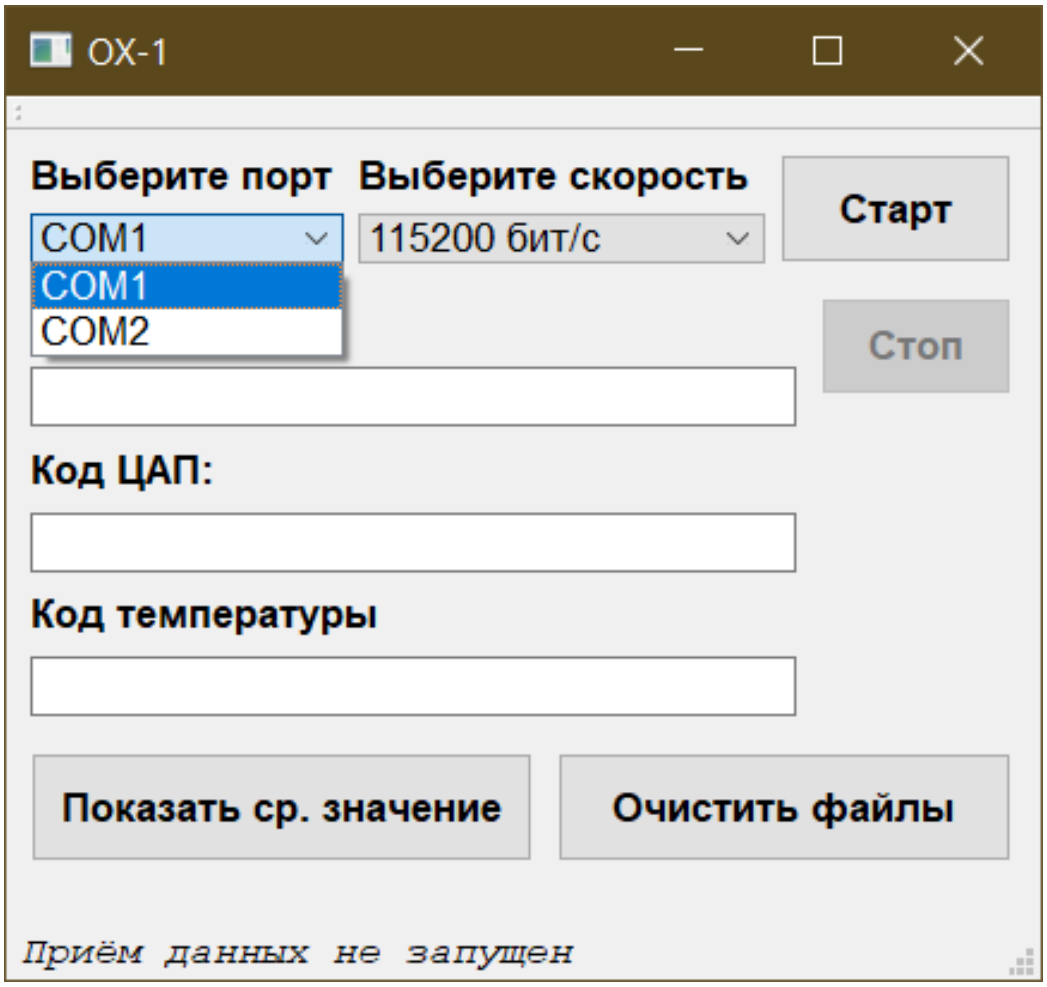


Рисунок 8 – Выбор COM-порта.

В меню «Скорость» отображается список стандартных скоростей обмена для COM-портов (Рисунок 9). По умолчанию установлено значение 115200 бит/с, при его изменении скорость перейдет к новому значению. Для опроса датчика OX 1 рекомендуется устанавливать максимальную доступную скорость, т.е. 115200 бит/с.

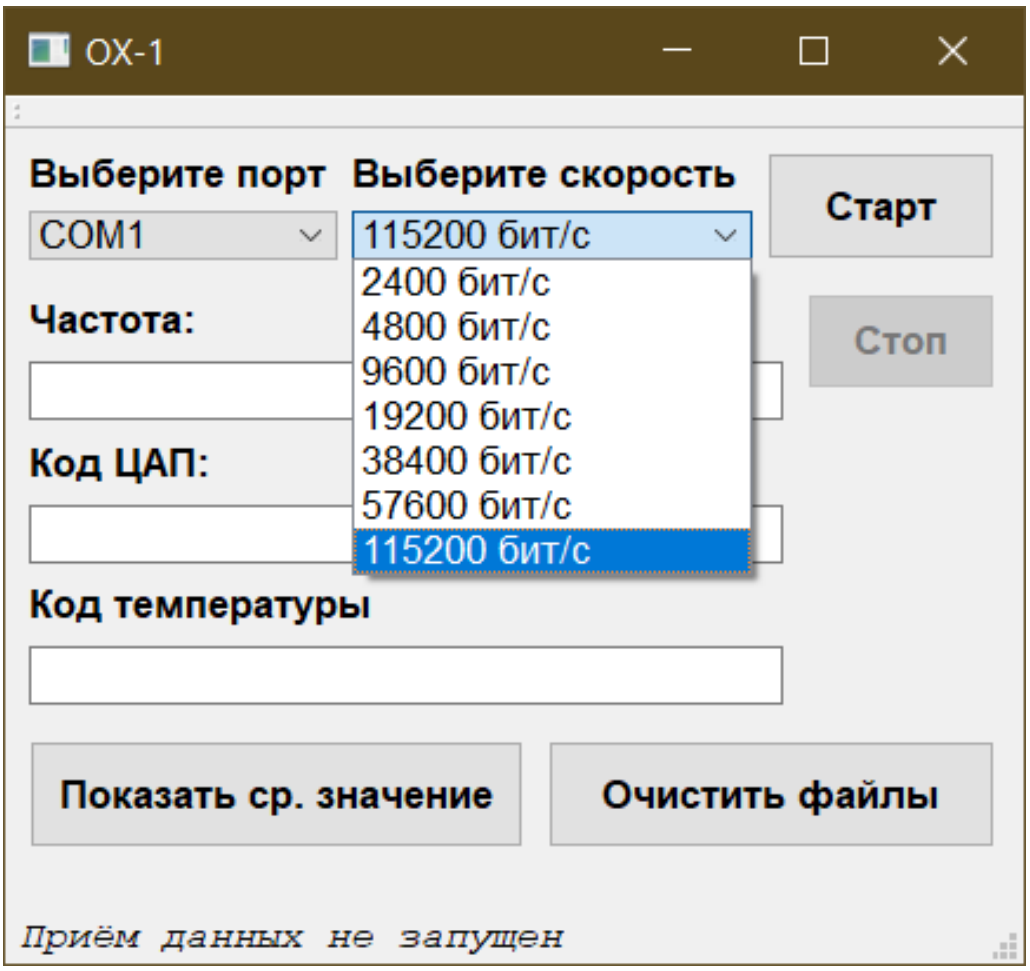


Рисунок 9 – Выбор скорости.

После нажатия кнопки «Старт» в полях вывода отображаются принятые данные в численном формате. Строка состояния изменяется, пользователь видит, что идет прием данных (Рисунок 10).

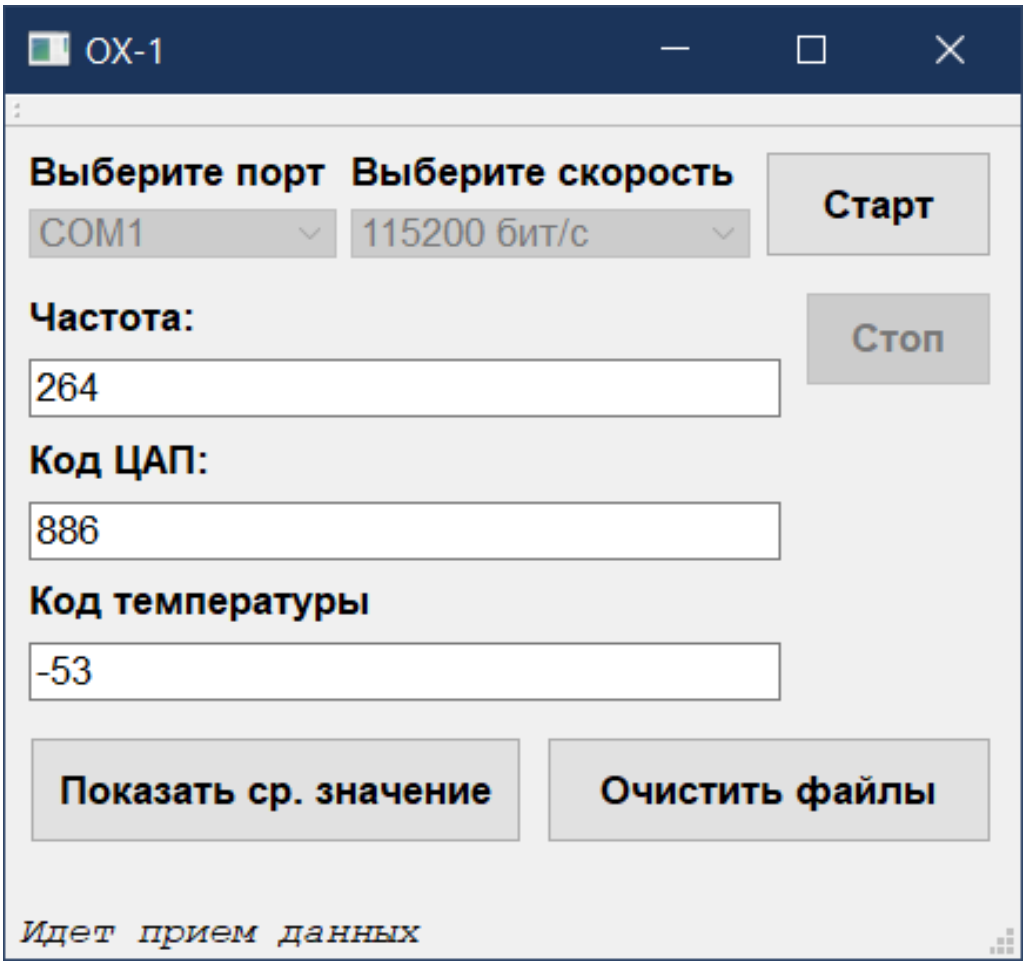


Рисунок 10 – Приём данных.

В случае, если данных не будет обнаружено, это будет отображено в строке состояния (Рисунок 11).

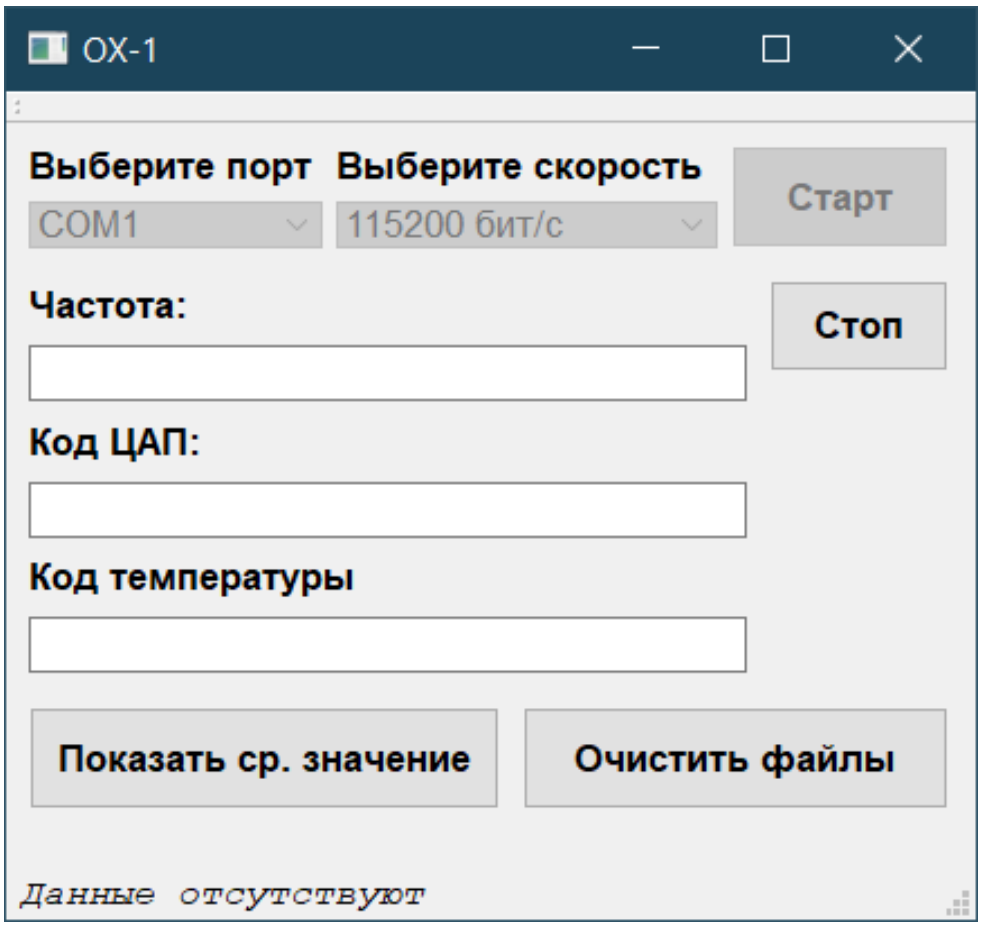


Рисунок 11 – Отсутствие принятых данных.

После нажатия кнопки «Стоп» процесс приема данных останавливается, поля вывода очищаются. В сроке состояния выводится сообщения, что прием данных завершен (Рисунок 12).

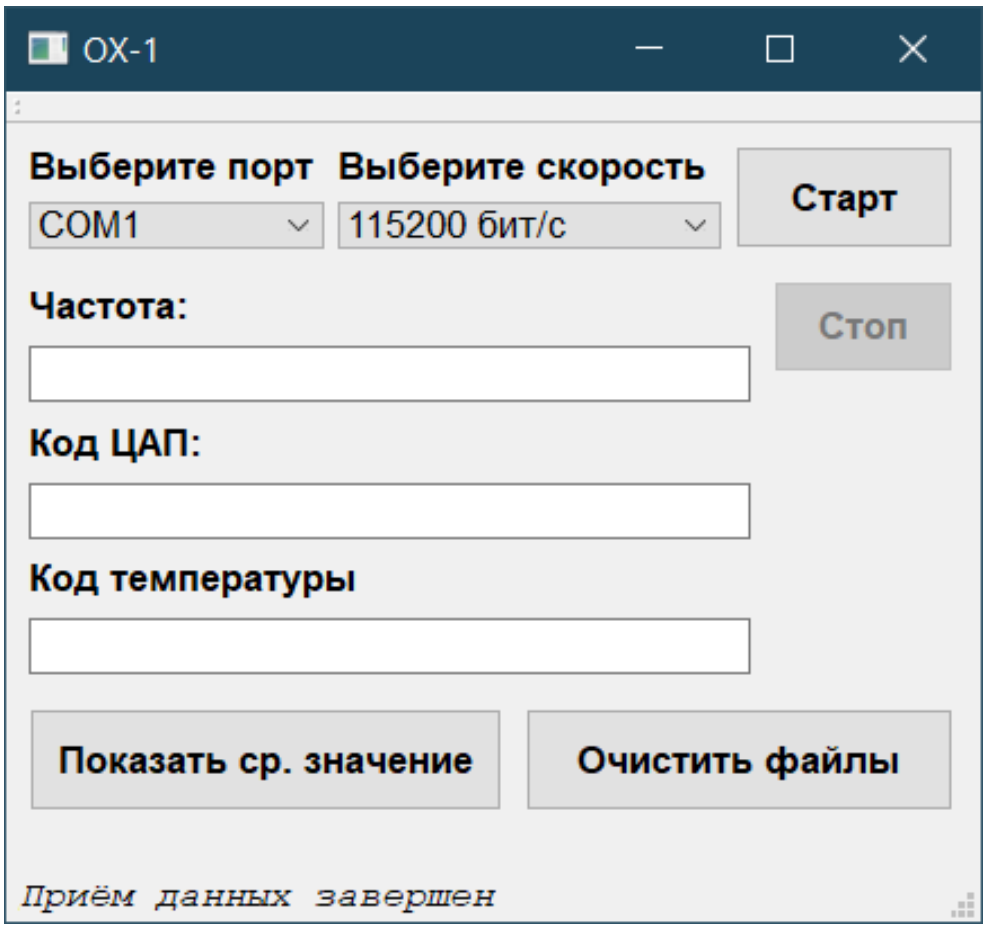


Рисунок 12 – Результат завершения работы программы.

В случае нажатия кнопки «Показать ср. значение» для пользователя будет отражено среднее значение частоты, которое получилось в результате анализа сохранённых в файле freq.txt параметров (Рисунок 13).

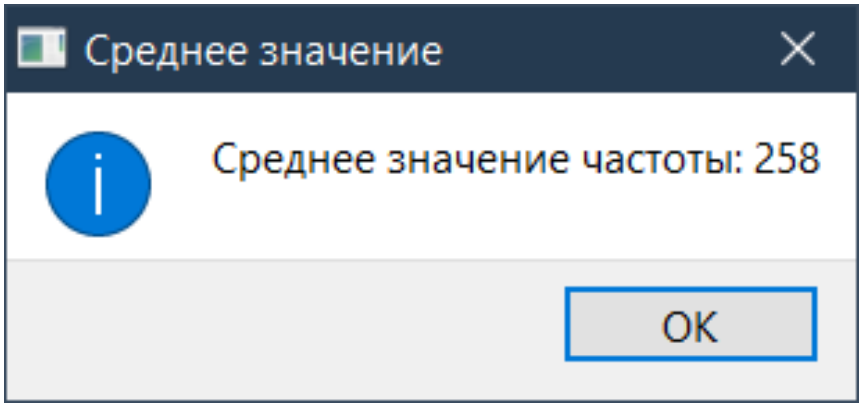


Рисунок 13 – Отображение среднего значения частоты.

Для очистки данного файла предусмотрено наличие кнопки «Очистить файлы».

4. Безопасность жизнедеятельности

Большое количество личного времени современный человек проводит за компьютером. Для правильного взаимодействия пользователя и программного обеспечения необходимо соблюдать рекомендации по обеспечению общих санитарно-технических норм к помещению и рабочим местам.

Помещение, в котором планируется производиться взаимодействие с разработанным продуктом, а точнее его размеры – площадь и объём, должны соответствовать количеству полностью оснащённых рабочих мест сотрудников. Помимо этого, необходимо обязательно учитывать соответствующие нормы освещения, температуры, чистоты воздуха, обеспечивать изоляцию от производственных шумов и т.п. На одного работающего сотрудника, основываясь на санитарные нормы труда СН 245-71, должно быть выделено помещение объёмом не менее 15 $м^3$, площадью не менее 4,5 $м^2$.

В ГОСТ 12.2.032 78 прописаны основные требования к организации рабочего место и его оборудованию. Высота рабочей поверхности стола для сотрудника должна регулироваться в пределах 680-800 мм; при отсутствии таковой возможности высота рабочей поверхности стола должна составлять 725 мм. Параметры рабочего стола должны быть следующими: ширина соответствует 800, 1200 или 1400 мм, глубина – 800 или 1000 мм при нерегулируемой высоте, которая равна 725 мм. Кроме того, пространство для ног высотой в рабочей области должно быть не менее 600 мм, а шириной – не менее 500 мм, глубиной не уровне колен – не менее 450 мм, и на уровне вытянутых ног – не менее 650 мм.

Рабочий стул (кресло) должен быть подъёмно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также по расстоянию спинки до переднего края сиденья.

Заключение

В ходе выполнения данной выпускной квалификационной работы были изучены способы приёма данных с помощью последовательного порта, способы обработки информации, работы с текстовыми файлами, также были получены навыки в области программирования на языке C++ на платформе Qt, навыки в разработке пользовательского интерфейса.

Результатом решения поставленной задачи можно считать разработанное и протестированное приложение для приёма и обработки данных посредством использования COM-порта. Разработанное программное обеспечение позволяет пользователю принимать информацию с датчика ОХ 1, обрабатывать их, сохранять в файл и предоставлять некоторую статистику.

В процессе тестирования приложения были выявлены различные проблемы. Так, например, программа вела себя некорректно в том случае, если во время начала приёма данных были доступны для изменения пункты выбора COM-порта и скорости. Решением проблемы являлось отключение данного выбора в момент приёма входящей информации.

В дальнейшем программа может быть улучшена путём добавления вывода дополнительной статистики. Можно будет добавить дополнительные поля вывода предыдущего значения, для наглядности, для каждого из принимаемых параметров.

Список используемых источников

1. Российские космические системы. [Электронный ресурс] // Режим доступа: http://russianspacesystems.ru.

2. Научно-исследовательский институт физических измерений. [Электронный ресурс] // Режим доступа: http://niifi.ru.

3. Biosgid – https://biosgid.ru/osnovy-ustrojstva-pk/com-port-ili-posledovatelnyj-port.html.

4. Nicknixer – https://nicknixer.ru/programmirovanie/kratkij-obzor-krossplatformennogo-frejmvorka-qt/.

5. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 5-е изд. — СПб.: Питер, 2016. — 992 с.: ил. — (Серия «Учебник для вузов»).

6. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.: ил.

7. Бланшет Ж., Саммерфилд М. Qt 4: программирование GUI на C++. Пер. с англ. 2-е изд., доп. — М.: КУДИЦ-ПРЕСС, 2008. — 736 с.

8. Боровский А.Н. Qt4.7+. Практическое программирование на C++. – СПб.: БХВ-Петербург, 2012. — 496 с.: ил. — (Профессиональное программирование).

9. Интерфейс RS-232 в каналах последовательной передачи данных: Метод. указания / Самар. гос. аэрокосм. ун-т; Сост. В. Г. Иоффе, К. Е. Климентьев. Самара, 2003. – 26 с.

10. Qt, справочная документация – URL: https://doc.qt.io/.

11. Qt, справочная документация на русском языке – URL: http://doc.crossplatform.ru.

12. Форум программистов и сисадминов Киберфорум – URL: https://www.cyberforum.ru/.

13. Система вопросов и ответов о программировании Stack Overflow – URL: https://stackoverflow.com/.

14. Система вопросов и ответов о программировании Stack Overflow на русском языке – URL: https://ru.stackoverflow.com/.

Приложение А. Листинги программы

Файл «mainwindow.h»

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <QMessageBox>
#include <QTimer>
#include <QLocale>
#include <qfile.h>
#include <QFileDialog>

namespace Ui
{
     class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

     private:
        // Указатель на пользовательский интерфейс, форму окна
          Ui::MainWindow *ui;
        // Указатель на экземпляр стандартного класса для работы с
последовательным портом
          QSerialPort *my_port;
        // Переменная для хранения состояния опроса (запущен или
остановлен)
          bool isStopped;
        // Переменная для хранения имени COM-порта
          QString p_name;
        // Переменная для хранения информационной части одного
полученного пакета данных
          QString oneData;
        // Целочисленная переменная для хранения значения скорости
обмена данными
          int rate;
        // Экземпляр стандартного класса для хранения метки
времени
```

```
          QTimer *my_time;

        // Файл для сохранения значений частоты
        QFile file;

        // Переменная для сохранения значений частоты
        QStringList freqData;

        // Переменная для среднего значения частоты
        QString freqMean;

    private slots:
        // Слот получения списка доступных COM-портов
          void GetPort();
        //  Слот  установки  выбранного  COM-порта  в  качестве
используемого
          void SetPort();
        // Слот открытия COM-порта
          void OpenPort();
        // Слот обработки пакета прочитанных данных
          void GetData(QString data);
        // Слот обработки остановки процесса опроса датчика
          void StopData();
        // Слот проверки результата открытия COM-порта
          bool isOpened();
        // Слот закрытия COM-порта
          void ClosePort();
        // Слот чтения данных
          void ReadData();
        // Слот обработки ситуации отсутствия данных в буфере COM-
порта
          void NoData();
        //  Слот  проверки  результата  остановки  процесса  опроса
датчика
          bool IsDataStop();
        // Слот обработки необходимости остановки процесса опроса
датчика
          void INeedStop();

        // Слот очистки файла
        void ClearFile();
        // Слот вывода среднего значения частоты
        void ShowStats();
```

```
    public:
        // Конструктор класса
          explicit MainWindow(QWidget *parent = 0);
        // Деструктор класса
          ~MainWindow();

    signals:
        // Сигнал появления требуемого количества данных
        // в буфере порта и необходимости их обработки
          void IHaveData(const QString&);

};

#endif // MAINWINDOW_H
```

Файл «mainwindow.cpp»

```
#include "mainwindow.h"
#include "ui_mainwindow.h"


// Констурктор
MainWindow::MainWindow(QWidget   *parent)   :   QMainWindow(parent),
ui(new Ui::MainWindow)
{
    // Инициализация GUI
    ui->setupUi(this);
    // Инициализация списка доступных параметров скорости передачи
данных по COM-порту
    ui->rate_list->addItem(QStringLiteral("2400           бит/с"),
QSerialPort::Baud2400);
    ui->rate_list->addItem(QStringLiteral("4800           бит/с"),
QSerialPort::Baud4800);
    ui->rate_list->addItem(QStringLiteral("9600           бит/с"),
QSerialPort::Baud9600);
    ui->rate_list->addItem(QStringLiteral("19200          бит/с"),
QSerialPort::Baud19200);
    ui->rate_list->addItem(QStringLiteral("38400          бит/с"),
QSerialPort::Baud38400);
    ui->rate_list->addItem(QStringLiteral("57600          бит/с"),
QSerialPort::Baud57600);
    ui->rate_list->addItem(QStringLiteral("115200         бит/с"),
QSerialPort::Baud115200);
    // Установка скорости по умолчанию
```

```
    ui->rate_list->setCurrentIndex(6);
    // Установка строки состояния
    ui->statusBar->setFont(QFont("Courier", 9, 1, true));
    ui->statusBar->showMessage("Приём данных не запущен", 0);

    // Получение списка доступных COM-портов
    GetPort();

    // Обновлене списка доступных COM-портов происходит каждые 5
секунд
    // Инициализация таймера
    my_time = new QTimer(this);
    // Установка сигнала на обновление списка COM-портов
    connect(my_time, SIGNAL(timeout()), this, SLOT(GetPort()));
    // Запуск таймера
    my_time->start(5000);

    // Установка имени файла для сохранения значений частоты
    file.setFileName("freq.txt");

    // Установка сигналов
    connect(ui->port_list, SIGNAL(currentIndexChanged(int)), this,
SLOT(SetPort()));
    connect(ui->rate_list, SIGNAL(currentIndexChanged(int)), this,
SLOT(SetPort()));
    connect(ui->start_button,     SIGNAL(clicked(bool)),     this,
SLOT(OpenPort()));
    connect(ui->stop_button,      SIGNAL(clicked(bool)),     this,
SLOT(StopData()));
    connect(ui->show_button,      SIGNAL(clicked(bool)),     this,
SLOT(ShowStats()));
    connect(ui->clear_button,     SIGNAL(clicked(bool)),     this,
SLOT(ClearFile()));

}


// Деструктор
MainWindow::~MainWindow()
{
    // Остановка приёма данных
    StopData();
    // Закрытие порта
    ClosePort();
```

```
    // Удаляем таймер
    delete my_time;
    // Удаляем порт
    delete my_port;
    // Удаляем интерфейс
    delete ui;
}


// Слот получения списка доступных COM-портов
void MainWindow::GetPort()
{
    // Удаляем список портов
    ui->port_list->clear();
    // Получаем информацию о портах
    const auto infos = QSerialPortInfo::availablePorts();
    // Обновляем список портов
    for(const QSerialPortInfo &my_info : infos)
    {
        // Добавляем COM-порт в список
        ui->port_list->addItem(my_info.portName());
    }

}


// Слот установки выбранного COM-порта в качестве используемого
void MainWindow::SetPort()
{
    // Устанавливаем имя порта
    p_name = ui->port_list->currentText();
    // Устанавливаем скорость
    rate    =    static_cast<QSerialPort::DataBits>(ui->rate_list-
>itemData(ui->rate_list->currentIndex()).toInt());
}


// Слот открытия COM-порта
void MainWindow::OpenPort()
{
    // Получение порта
    my_port = new QSerialPort(p_name);
    // Открытие порта
    if(my_port->open(QIODevice::ReadWrite))
```

```
    {
        // Остановка обновления списка COM-портов
        my_time->stop();
        // Отключаем приём сигналов на обновление списка портов
        disconnect(my_time,         SIGNAL(timeout()),         this,
SLOT(GetPort()));

        // Устанавливаем приём сигналов
        connect(my_time, SIGNAL(timeout()), this, SLOT(NoData()));
        // Запускаем таймер
        my_time->start();

        // Отключаем выбор порта и скорости
        ui->port_list->setEnabled(false);
        ui->rate_list->setEnabled(false);

        // Установка параметров порта
        // Установка скорости
        my_port->setBaudRate(rate);
        my_port->setParity(QSerialPort::NoParity);
        my_port->setDataBits(QSerialPort::Data8);
        my_port->setStopBits(QSerialPort::OneStop);
        my_port->setTextModeEnabled(true);
        // Изменение строки состояния
        ui->statusBar->showMessage("Начат прием данных", 0);

        // Установка сигнала, если получены данные
        connect(this,                   SIGNAL(IHaveData(QString)),
SLOT(GetData(QString)));
        // Установка сигнала на получение данных
        connect(my_port,          SIGNAL(readyRead()),         this,
SLOT(ReadData()));

        // Отключаем остановку
        isStopped = false;
        // Читаем данные
        ReadData();

        // Включаем выбор порта и скорости
        ui->stop_button->setEnabled(true);
        ui->start_button->setEnabled(false);

    }
    // В случае ошибки открытия порта
```

```
    else
    {
        QMessageBox::critical(this,      tr("Ошибка      открытия"),
my_port->errorString());
    }

}


// Слот обработки пакета прочитанных данных
void MainWindow::GetData(QString data)
{
    QLocale t;
    int line = 0, num = 0;
    // Обновляем строку состояния
    ui->statusBar->showMessage("Идет прием данных", 0);
    // Если пришли нужные данные
    if(data.left(1) == "#")
    {
        // Определение протокола взаимодействия
        line = data.mid(1, 1).toInt();
        num = data.mid(2, data.indexOf("!") - 2).toInt();
        switch(line)
        {
            // Частота
            case 1:
                ui->freq_line->setText(t.toString(num));
                // Открываем файл на добавление текста
                file.open(QIODevice::Append | QIODevice::Text);
                // Записываем полученное значение в файл
                file.write((t.toString(num)).toUtf8() + "\n");
                // Закрываем файл
                file.close();
                break;

            // ЦАП
            case 2:
                ui->dac_line->setText(t.toString(num));
                break;

            // Температура
            case 3:
                ui->temp_line->setText(t.toString(num));
                break;
```

```
            default: break;
        }
    }
}


// Слот обработки остановки процесса опроса датчика
void MainWindow::StopData()
{
    // Указываем на необходимости остановки процесса опроса
датчика
    INeedStop();
    // Если приём данных завершился
    if(IsDataStop())
    {
        // Изменяем GUI
        ui->stop_button->setEnabled(false);
        ui->start_button->setEnabled(true);
        ui->dac_line->clear();
        ui->freq_line->clear();
        ui->temp_line->clear();

        // Устанавливаем строку состояния
        ui->statusBar->showMessage("Приём данных завершен", 0);
        // Отключаем слот
        disconnect(this,     SIGNAL(IHaveData(QString)),     this,
SLOT(GetData(QString)));
        // Закрываем порт
        ClosePort();

        // Запрещаем выбор порта и скорости
        ui->port_list->setEnabled(true);
        ui->rate_list->setEnabled(true);

        // Останавливаем таймер
        my_time->stop();
        // Отключаем слот
        disconnect(my_time,         SIGNAL(timeout()),         this,
SLOT(NoData()));
        // Устанавливаем слот получения списка доступных COM-
портов
        connect(my_time,          SIGNAL(timeout()),          this,
SLOT(GetPort()));
```

```
        // Устанавливаем таймер
        my_time->start(5000);
    }
}


// Слот проверки результата открытия COM-порта
bool MainWindow::isOpened()
{
    return my_port->isOpen();
}


// Слот закрытия COM-порта
void MainWindow::ClosePort()
{
    // Если порт не закрыт, то закрываем
    if(my_port->isOpen())
        my_port->close();
}


// Слот чтения данных
void MainWindow::ReadData()
{
    // Размер текста
    static int textSize = 8;
    // Буфер
    static char buffer[8];
    // Если что-то пришло
    if(my_port->bytesAvailable() == 0)
        NoData();
    // Пока есть данные
    while(my_port->bytesAvailable() >= textSize && !isStopped)
    {
        // Читаем данные
        my_port->read(buffer, textSize);
        // Сигнал появления требуемого количества данных
        // в буфере порта и необходимости их обработки
        emit IHaveData(QString(buffer));
    }

}
```

```
// Слот обработки необходимости остановки процесса опроса датчика
void MainWindow::INeedStop()
{
    isStopped = true;
}


// Слот проверки результата остановки процесса опроса датчика
bool MainWindow::IsDataStop()
{
    return isStopped;
}


// Слот обработки ситуации отсутствия данных в буфере COM-порта
void MainWindow::NoData()
{
    // Если данные не пришли
    if(my_port->bytesAvailable() == 0)
    {
        ui->statusBar->showMessage("Данные отсутствуют", 0);
        ui->dac_line->clear();
        ui->freq_line->clear();
        ui->temp_line->clear();
    }
}


// Слот очистки файла со значениями частоты
void MainWindow::ClearFile()
{
    // Открываем файл и очищаем
    if (file.open(QIODevice::WriteOnly | QIODevice::Truncate))
    {
        // Закрываем файл
        file.close();
    }
}


// Слот вывода среднего значения частоты
void MainWindow::ShowStats()
{
```

```
    // Открываем файл на чтение
    if(file.open(QIODevice::ReadOnly |QIODevice::Text))
    {
        // Пока не перебрали все строки
        while(!file.atEnd())
        {
            // Читаем строку
            QString str = file.readLine();
            // Добавляем значение в список
            freqData.append(str);
        }
    }

    // Переменная для хранения суммы значений частот
    int freqSum = 0;
    // Перебираем значения
    foreach(QString str, freqData)
    {
        // Добавляем очередное значение
        freqSum += str.toInt();
    }

    // Вычисляем среднее значение частоты
    float freqMeanFloat = freqSum/freqData.size();
    // Переводим полученное значение в строковое
    freqMean = QString::number(freqMeanFloat);
    // Выводим значение на экран
    QMessageBox::information(this,
                             "Среднее значение",
                             "Среднее значение частоты: " +
freqMean);

}
```

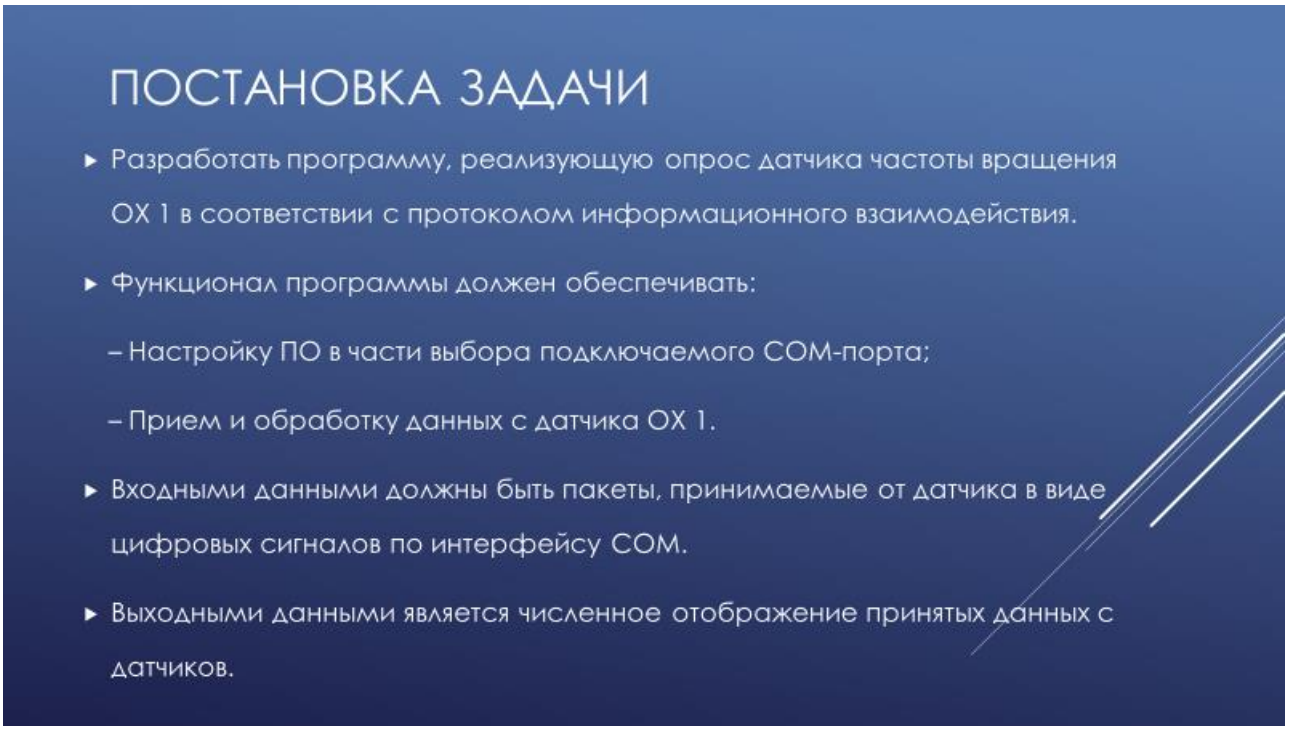
Файл «main.cpp»

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
     QApplication a(argc, argv);
     MainWindow w;
     w.show();

     return a.exec();
}
```

Приложение В. Презентация



СТРУКТУРНАЯ СХЕМА

ПРОТОКОЛ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

Ответ датчика представляется в виде команды, длиной 8 бит

#	N	0	0	0	0	0	!

Описание значащих разрядов:

– Старт-бит, признак начала команды

! – Стоп-бит, признак конца команды

N – Код команды

Датчик отправляет данные в виде пакета, содержащего в себе 3 вида команд:

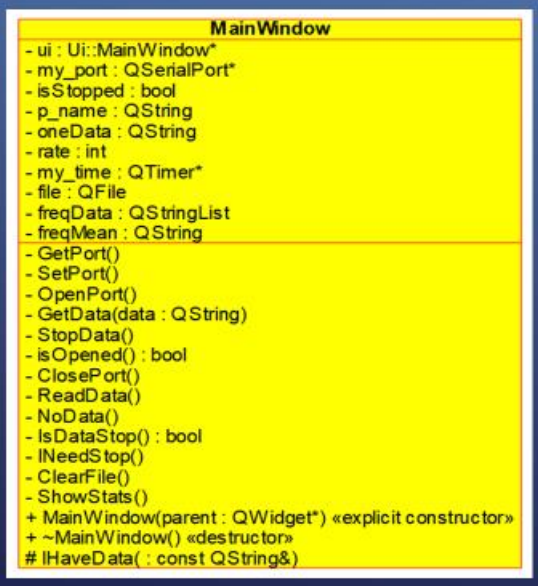
N = 1 – код частоты

N = 2 – код ЦАП

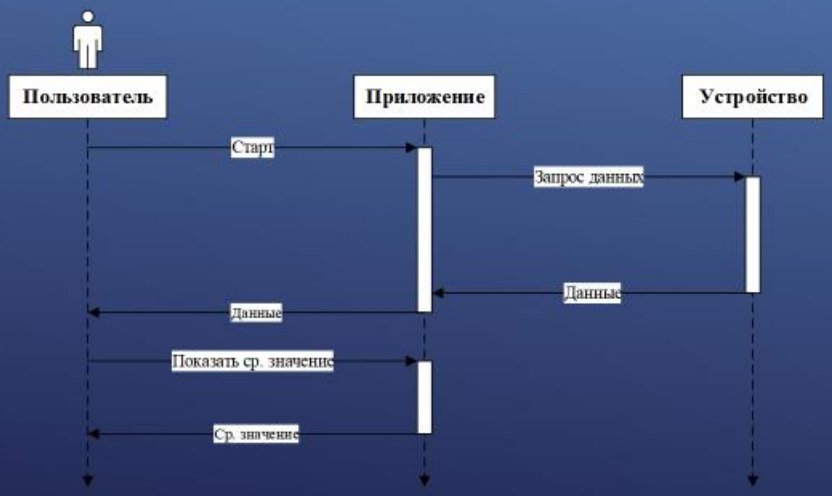
N = 3 – код температуры

Частота отправления пакетов данных зависит от скорости вращения вала (ротора).

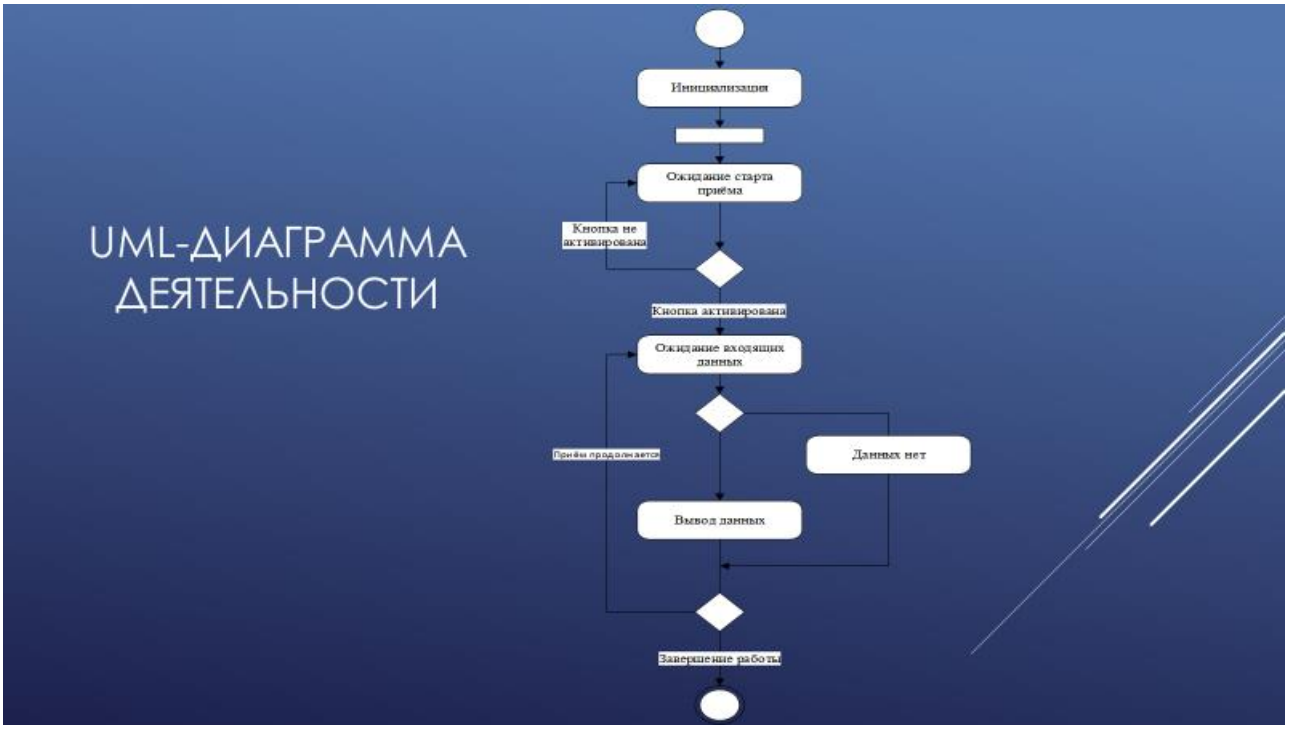
UML-ДИАГРАММА КЛАССОВ



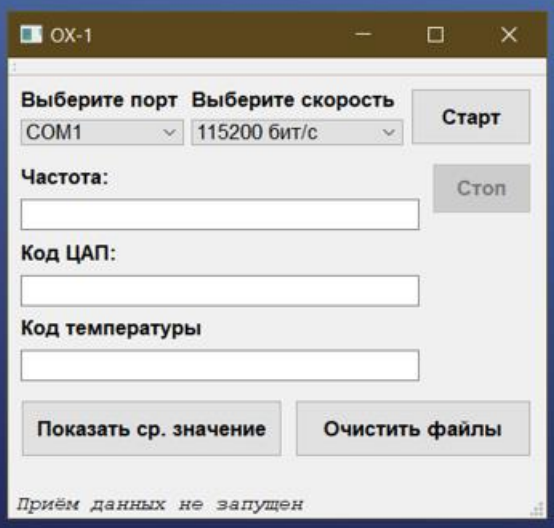
UML-ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ



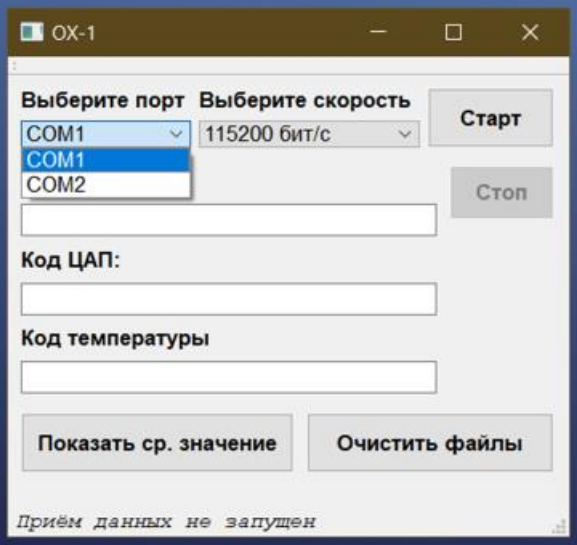
UML-ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ
Открытие COM-порта
Старт приёма данных
Получение данных
Вывод данных
Остановка приёма
Выбор COM-порта
Вывод ср. значения
Выбор скорости
Вычисление ср. значения
Показать ср. значение
Очистка файла
Очисить файлы

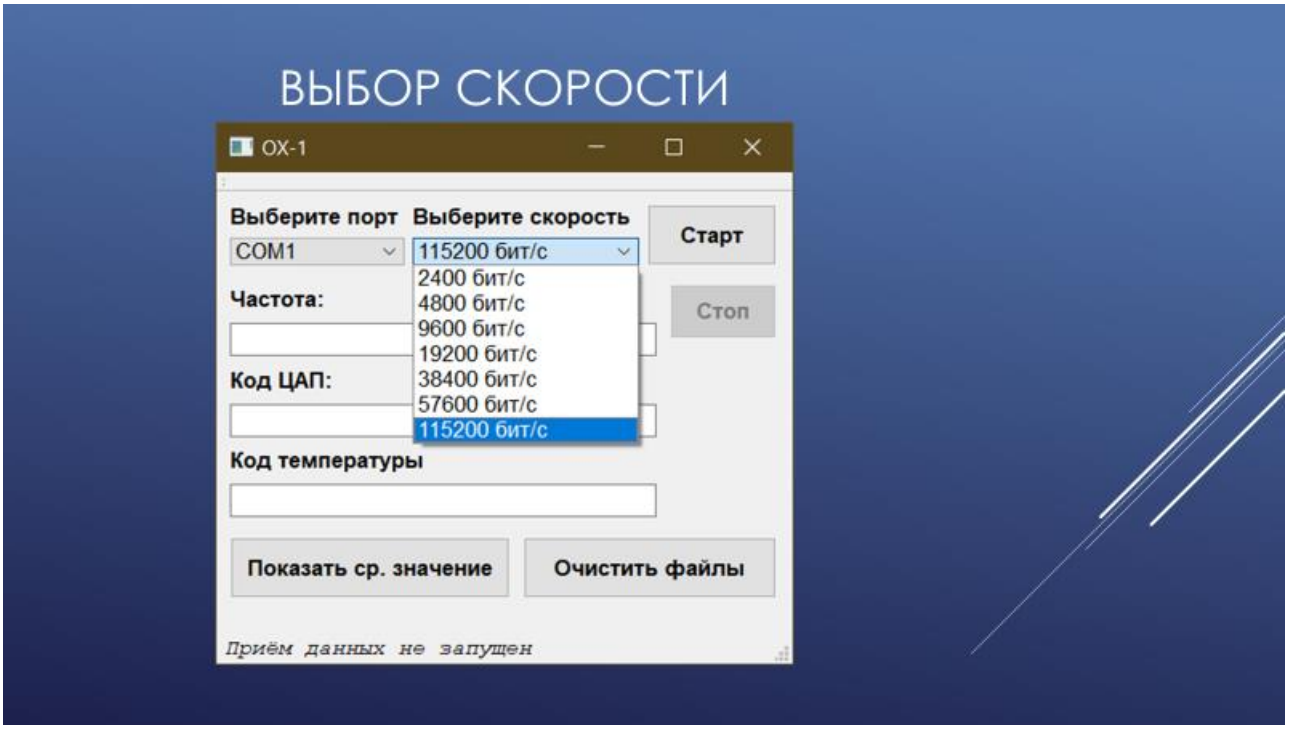

UML-ДИАГРАММА ДЕЯТЕЛЬНОСТИ
Инициализация
Ожидание старта приёма
Кнопка не активирована
Кнопка активирована
Ожидание входящих данных
Приём продолжается
Данных нет
Вывод данных
Завершение работы

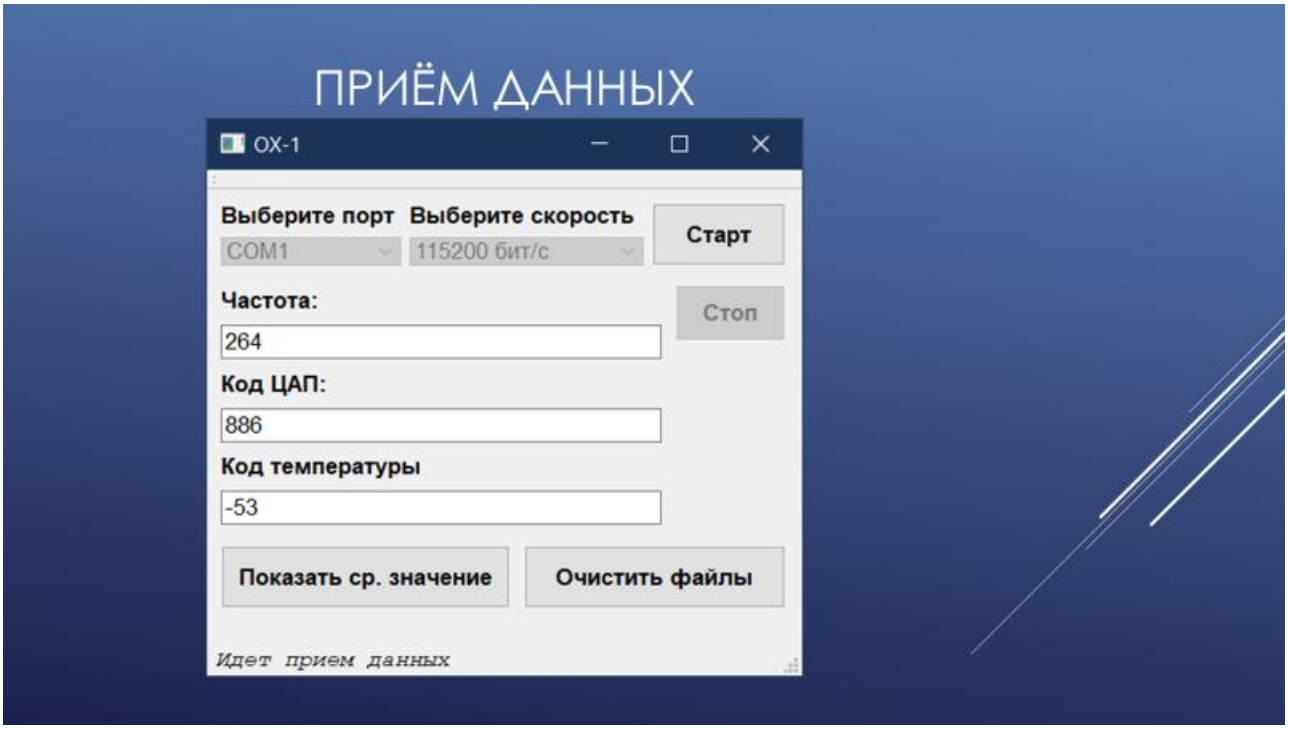
ГЛАВНОЕ ОКНО ПРОГРАММЫ



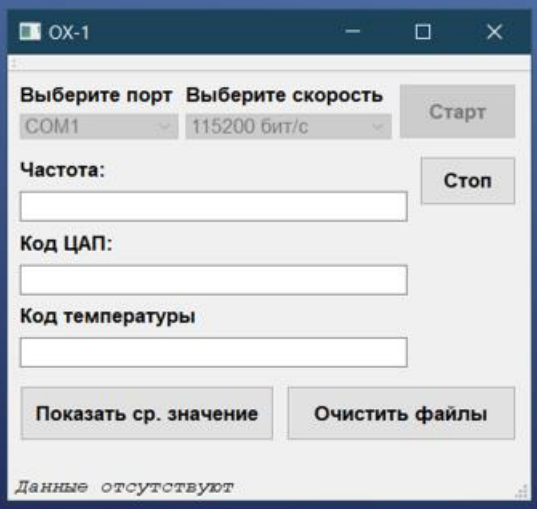
ВЫБОР COM-ПОРТА



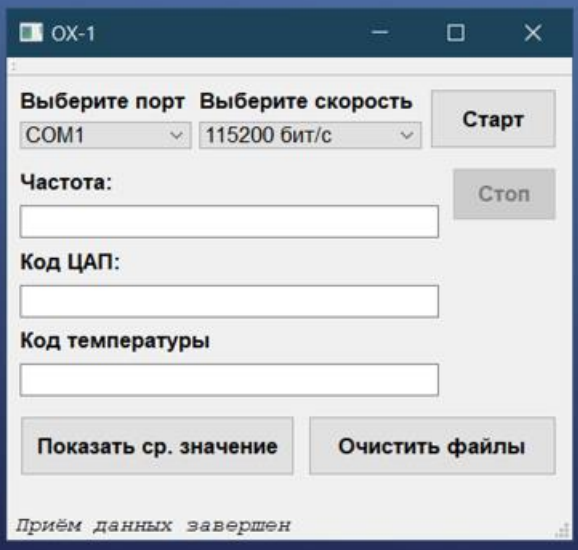

ВЫБОР СКОРОСТИ
ОХ-1
Выберите порт
Выберите скорость
COM1
115200 бит/с
Старт
Частота:
Стоп
Код ЦАП:
2400 бит/с
4800 бит/с
9600 бит/с
19200 бит/с
38400 бит/с
57600 бит/с
115200 бит/с
Код температуры
Показать ср. значение
Очистить файлы
Приём данных не запущен


ПРИЁМ ДАННЫХ
ОХ-1
Выберите порт
Выберите скорость
COM1
115200 бит/с
Старт
Частота:
Стоп
264
Код ЦАП:
886
Код температуры
-53
Показать ср. значение
Очистить файлы
Идет прием данных

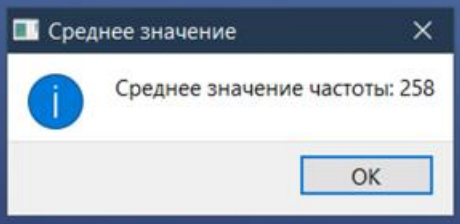
ОТСУТСТВИЕ ПРИНЯТЫХ ДАННЫХ



РЕЗУЛЬТАТ ЗАВЕРШЕНИЯ РАБОТЫ

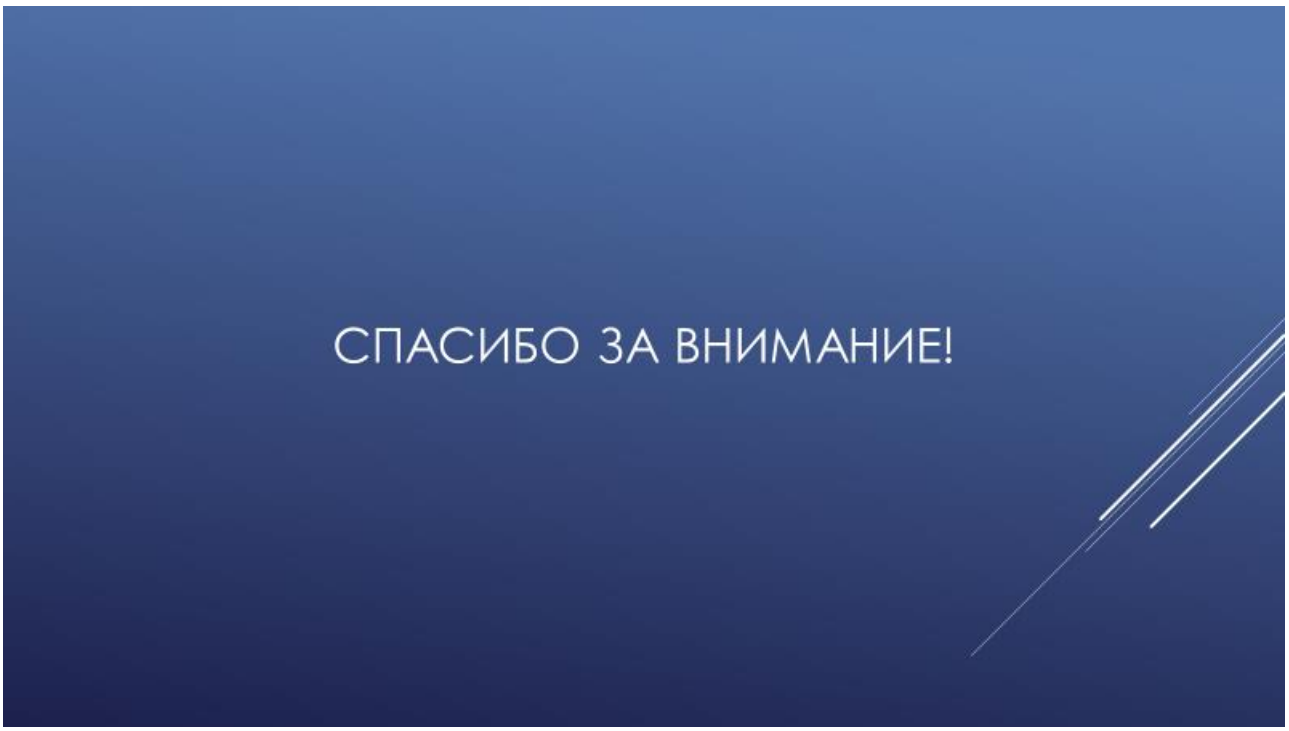


ОТОБРАЖЕНИЕ СРЕДНЕГО ЗНАЧЕНИЯ ЧАСТОТЫ



ЗАКЛЮЧЕНИЕ

- В ходе выполнения данной выпускной квалификационной работы были изучены способы приёма данных с помощью последовательного порта, способы обработки информации, работы с текстовыми файлами, также были получены навыки в области программирования на языке C++ на платформе Qt, навыки в разработке пользовательского интерфейса.
- Результатом решения поставленной задачи можно считать разработанное и протестированное приложение для приёма и обработки данных посредством использования COM-порта. Разработанное программное обеспечение позволяет пользователю принимать информацию с датчика ОХ 1, обрабатывать их, сохранять в файл и предоставлять некоторую статистику.


СПАСИБО ЗА ВНИМАНИЕ!