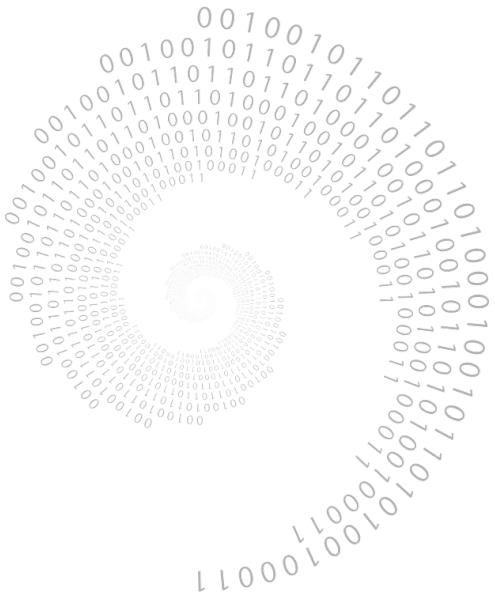
ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

Институт дополнительного образования

ВЫСШАЯ ИНЖЕНЕРНАЯ ШКОЛА

Разработка прототипа распределенного ПО для регистрации и автоматической синхронизации сообщений системы управления электроэнергетической системой судна

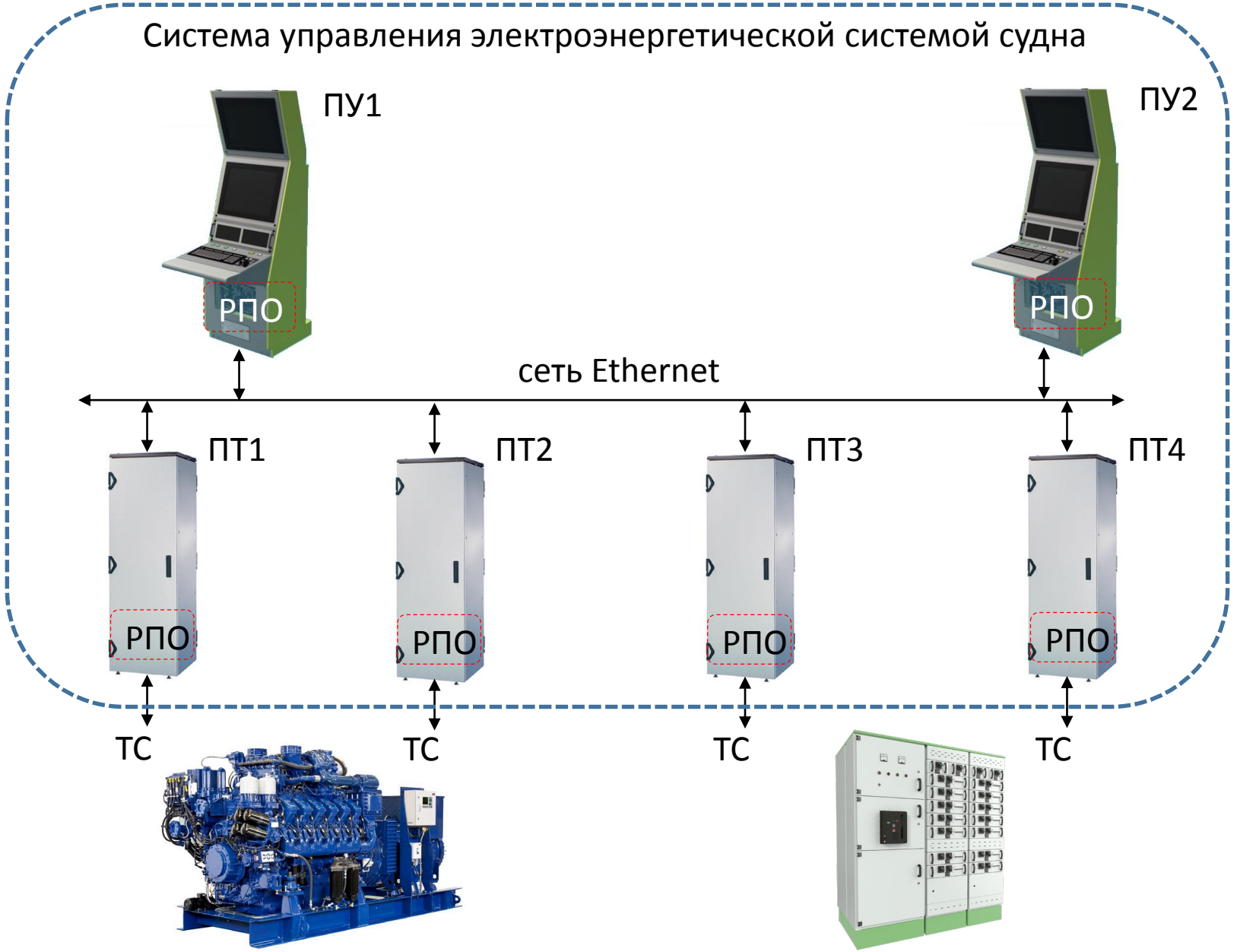


Выполнил студент: Трубин Владислав Александрович

Научный руководитель: Полубенцева Марина Игоревна

Санкт-Петербург 2021 г.

Введение в предметную область



ПУ - пульт управления;
ПТ – прибор Технологический;
ТС – технические средства (ГРЩ, ДГ, ЩП…);
РПО – разрабатываемое программное обеспечение;

Операционная система: QNX6.5, Windows

Цели и задачи

Цель: на основании полученного ТЗ и имеющейся сетевой архитектуры необходимо реализовать распределенное ПО для регистрации и синхронизации сообщений по сети. Каждый прибор должен быть в состоянии получать сигналы от технических средств и формировать сообщения даже если он изолирован, в случае обрыва сети, а при восстановлении сети передать в сеть свои данные.

Задачи:

1. Проанализировать техническое задание на ПО.

2. Разработать архитектуру ПО.

3. Разработать программные компоненты:
- для эффективного взаимодействия всех компонентов ПО с общей памятью;
- для обмена данными и синхронизацией между узлами сети Ethernet;
- для отображения данных на экране монитора и выполнения команд для управления фильтрацией данных;
- для отслеживания входных сигналов и формирования необходимых данных для общей памяти ПО;
- для сохранения или загрузки текущего состояния данных.

4. Проверить работоспособность ПО и соответствие техническому заданию.

Проектирование

Для выполнения поставленной цели и задач было принято решение структурно разделить разрабатываемое ПО на два процесса из пяти программных компонентов:

1. Программный интерфейс приложения (**API** - взаимодействие между двумя процессами) — отвечает за связь компонентов РПО между собой посредством разделяемой памяти РПО, получение команд от графического интерфейса приложения, выполнение алгоритмов фильтрации, сортировки, синхронизации сообщений.

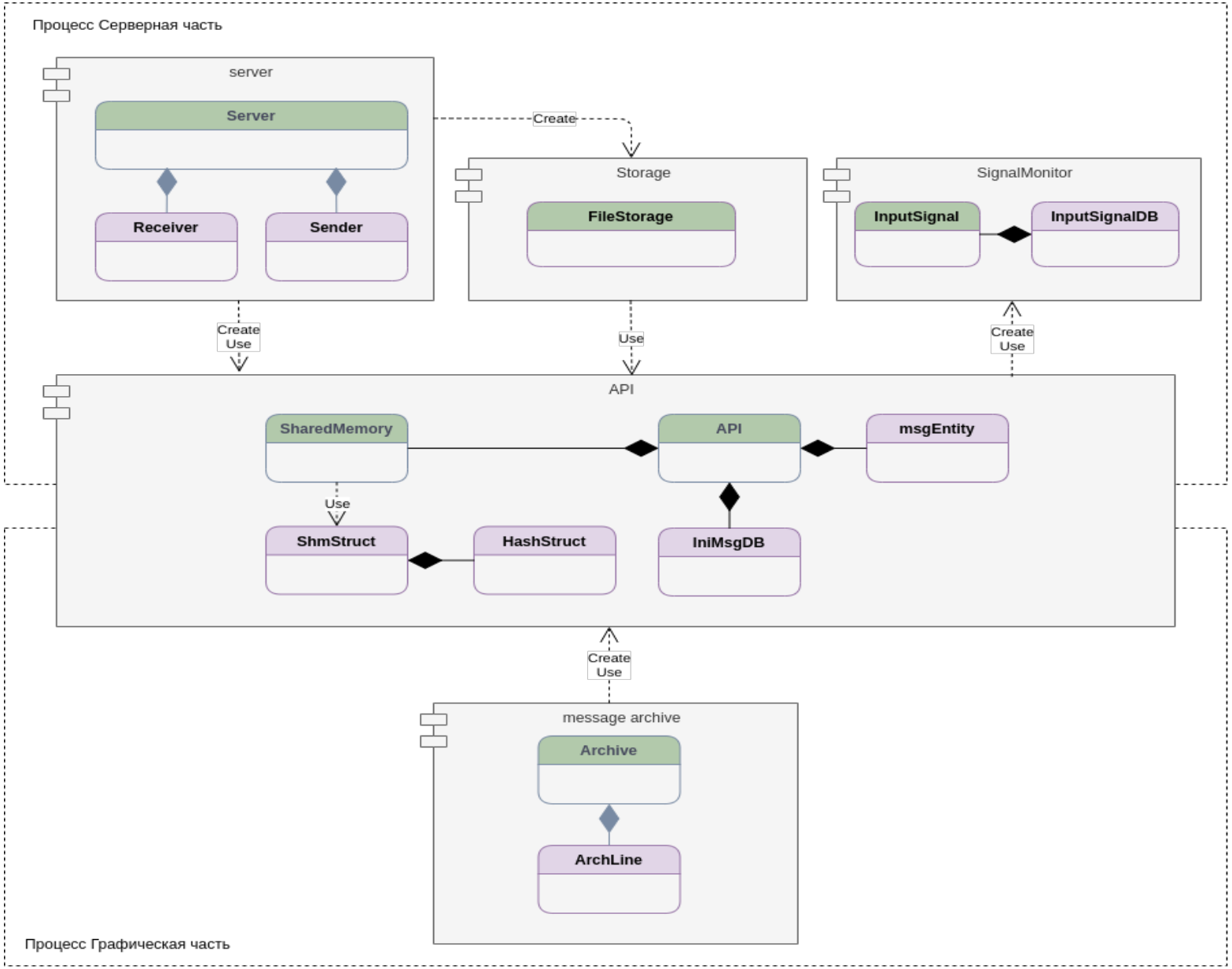
2. Клиент-серверная часть (**Server** процесс 1) — отвечает за посылку и прием информации между узлами сети Ethernet.

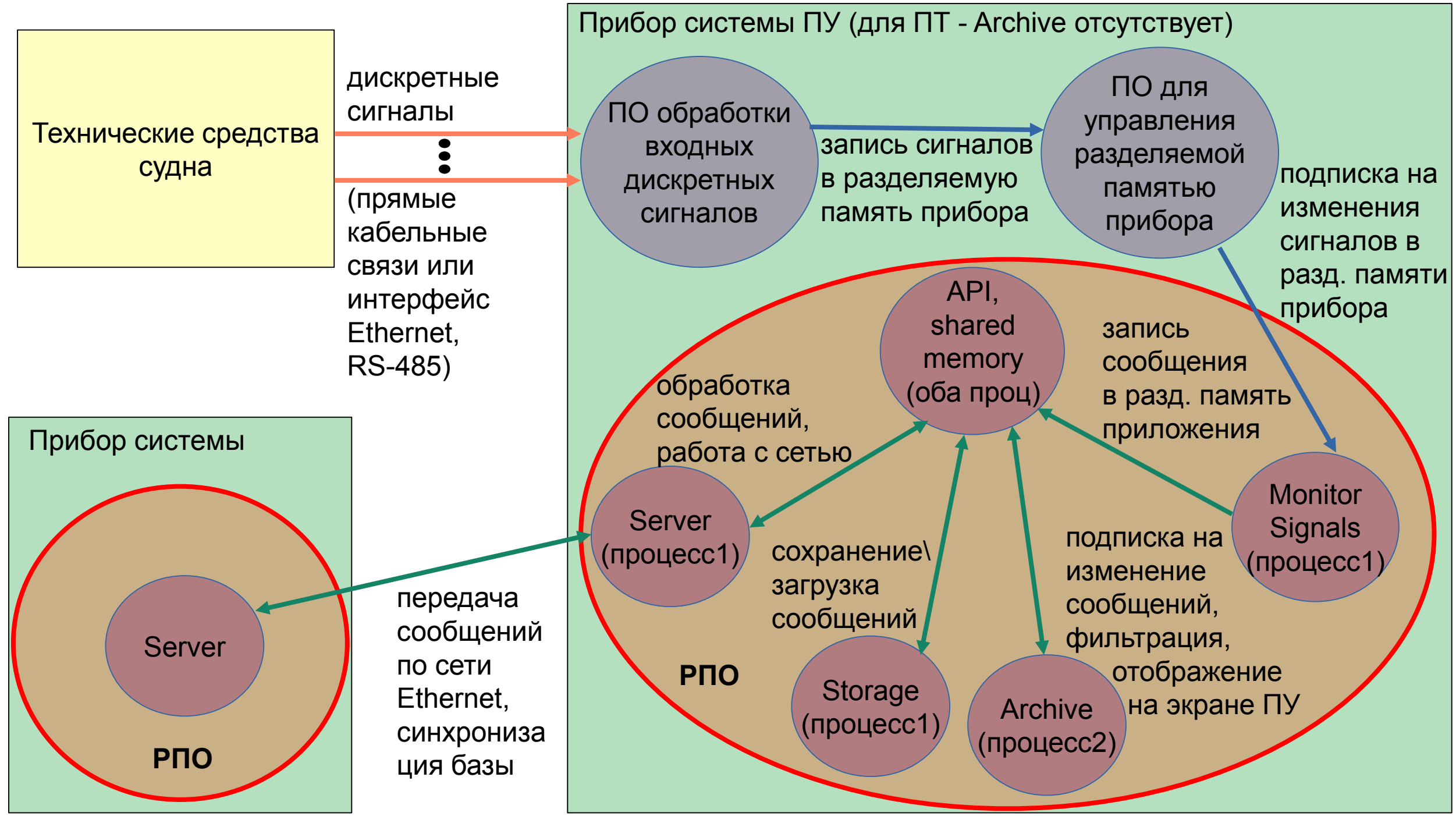
3. Компонент наблюдатель за сигналами (**Monitor Signals** процесс 1) — отвечает за прием входных сигналов от стороннего ПО узла и регистрации их в разделяемой памяти РПО.

4. Компонент хранилище (**Storage** процесс 1) — предназначен для сохранения текущего состояния базы сообщений в файл (базу данных) с заданными интервалами времени, а также загрузки при запуске прибора.

5. Графическая часть приложения (**Archive** процесс 2) — представляет собой архив информационных и аварийно-предупредительных сообщений в виде строк с необходимой информацией, а также кнопок для фильтрации по требуемым критериям.

Упрощённая диаграмма классов




Технические средства судна
дискретные сигналы
(прямые кабельные связи или интерфейс Ethernet, RS-485)
Прибор системы ПУ (для ПТ - Archive отсутствует)
ПО обработки входных дискретных сигналов
запись сигналов в разделяемую память прибора
ПО для управления разделяемой памятью прибора
подписка на изменения сигналов в разд. памяти прибора
API, shared memory (оба проц)
обработка сообщений, работа с сетью
запись сообщения в разд. память приложения
Server (процесс1)
сохранение\ загрузка сообщений
подписка на изменение сообщений, фильтрация, отображение на экране ПУ
Monitor Signals (процесс1)
Storage (процесс1)
Archive (процесс2)
РПО
Прибор системы
Server
РПО
передача сообщений по сети Ethernet, синхрониза ция базы

Реализация

1. Программный интерфейс приложения (API)

Программный интерфейс приложения представлен в виде динамической библиотеки и предназначен для управления разделяемой памятью ПО, для выполнения действий с сообщениями, предоставляет интерфейс для других компонентов ПО.

Имя класса	Назначение (краткое описание)
msgEntity	реализует все необходимые данные для сообщений и функции для работы с ними.
ShmStruct	представление сообщений в разделяемой памяти ПО.
HashStruct	представление хеш кодов сообщений в разделяемой памяти ПО.
SharedMemory	реализует класс **QSharedMemory** и отвечает за работу с разделяемой памятью используя класс **QThread** для выполнения в отдельном потоке, а также класс **QMutex** для защиты разделяемых ресурсов. Также включает класс **QQueue** для выполнения действий (квитирование, добавление, нормализация, синхронизация) над входными сообщениями по таймеру.
IniMsgDB	применятся для парсинга .ini файлов с конфигурацией сообщений. Содержит тексты для всех сообщений в системе и используется для отображения на экране монитора, каждому тексту сообщения соответствует уникальный id.
API	отвечает за создание и управление всеми классами компонента, посредством интерфейса данного класса все остальные компоненты ПО будут взаимодействовать с разделяемой памятью ПО.

Структура данных класса сообщений передаваемого по сети

Назначение данных	Количество бит
Признак звукового сигнала	1
Признак нормализации	1
Признак квитирования	1
Тип сообщения	4
Тип системы (доп. критерии)	6
Уникальный идентификатор	24
Время формирования	34
Время нормализации	29
Время квитирования	28
Общее количество бит (байт)	128 (16)

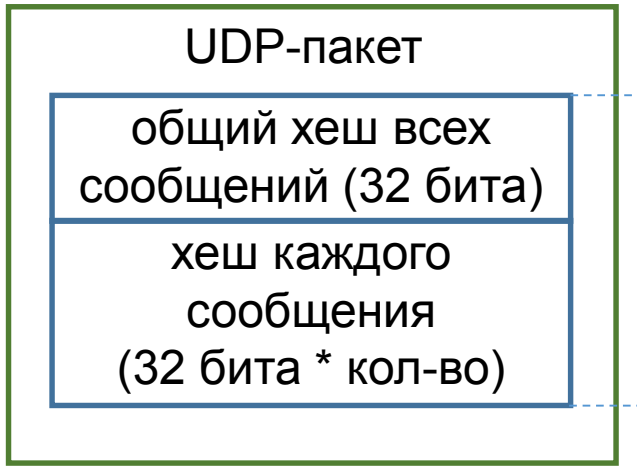
2. Клиент-серверная часть (Server)

Клиент-серверная часть ПО предназначена для обмена информацией между узлами по сети Ethernet и синхронизации баз сообщений между ними.

Имя класса	Назначение (краткое описание)
NetStruct, **DataType, Data, DataHash**	предназначены для представления данных сообщений при передаче по сети, с использованием **qCompress** для дополнительного сжатия данных при передачи.
Sender	предназначен для передачи данных по сети Ethernet посредством UDP. Отправляет общий хеш код и хеш код для каждого сообщения запрошенные у компонента **API** по таймеру. Отправляет сообщения если пришел сигнал от **Receiver** о том, что найдены отличающиеся сообщения по хеш коды от хеш кодов другого узла сети.
Reciever	предназначен для приема данных по сети Ethernet посредством UDP. Принимает общий хеш код и хеш код для каждого сообщения, запрашивает хеш коды у **API** и сравнивает с принятыми. Если найдены хеш коды, отличающиеся от хеш кодов, полученных по сети формирует контейнер с сообщениями и передает их классу **Sender** для отправки по сети. Принимает сообщения от других узлов в сети, добавляет сообщения в очередь компонента **API** для синхронизации.
Server	отвечает за создание классов **Sender**, **Receiver**, **API** серверной части РПО.

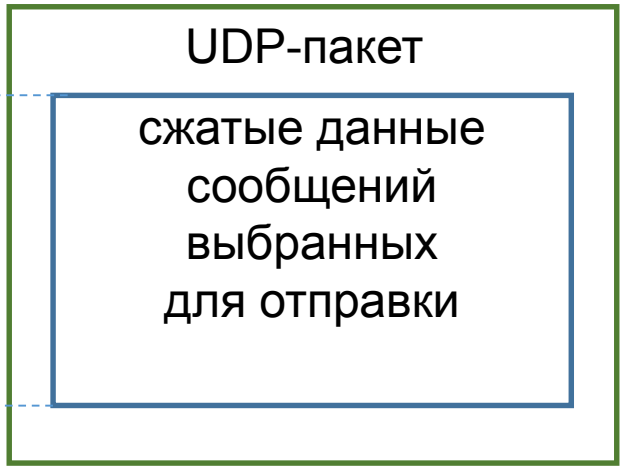
Форматы передаваемых по сети пакетов

UDP-пакет при
передаче хешированных данных

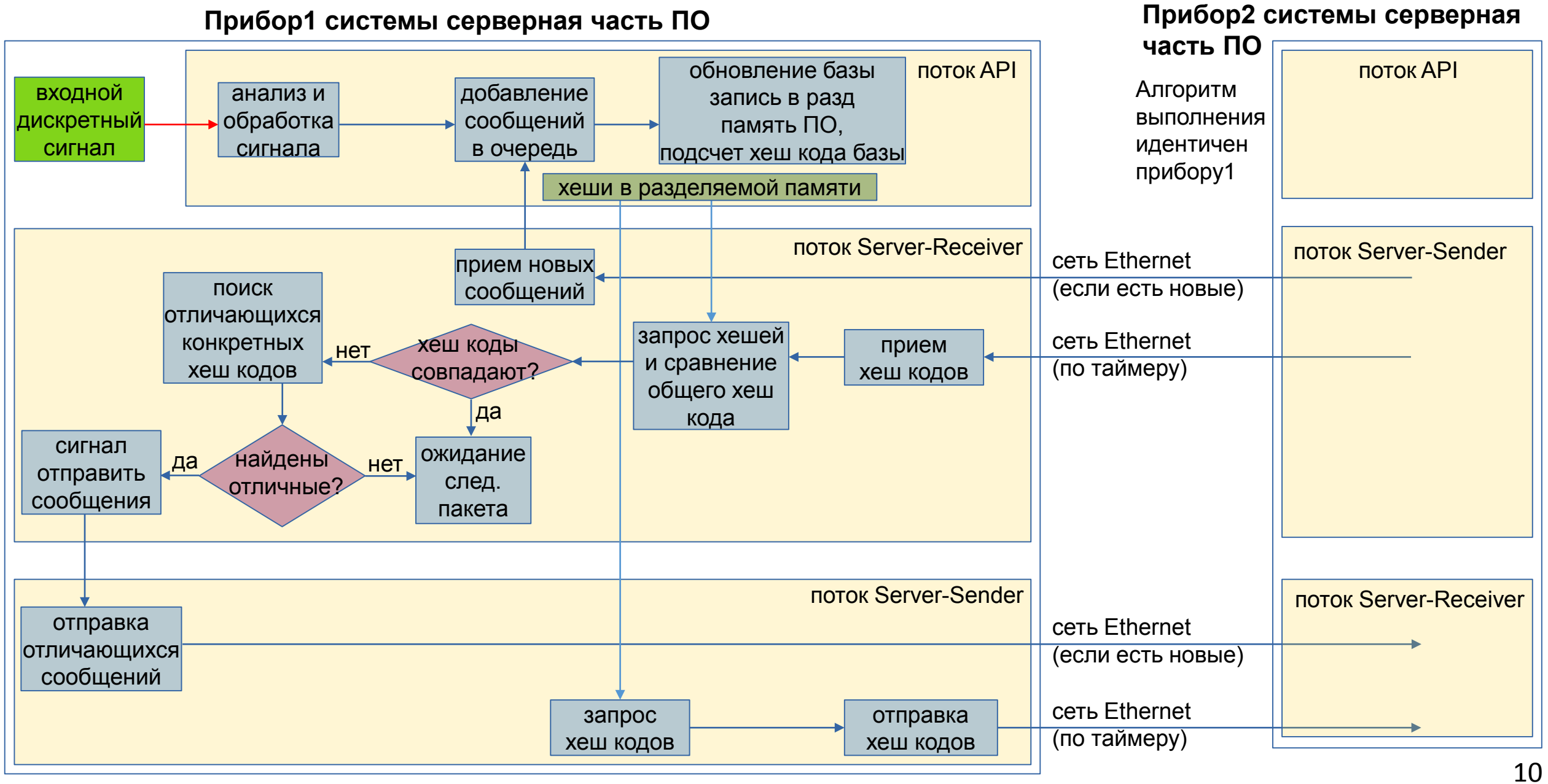


данные для обмена по сети (макс 1400 байт)

UDP-пакет при
передаче сжатых сообщений



Алгоритм синхронизации сообщений по сети



3. Компонент наблюдатель за сигналами (Monitor Signals)

Компонент наблюдатель за сигналами — предназначен для приема входных сигналов от стороннего ПО узла и регистрации их в разделяемой памяти РПО.

Имя класса	Назначение (краткое описание)
InputSignal	подписывается на сигналы от стороннего ПО узла чтобы следить за изменением дискретного сигнала и при изменении состояния помещает в очередь компонента **API** новую запись для формирования из данных нового объекта сообщения и записи в разделяемую память ПО.
ImitSignals	предназначен для тестирования ПО, является наследником класса **QMainWindow**. Имитирует стороннее ПО узла, выдает сигналы классу **InputSignal** о изменении наблюдаемого сигнала.

Графическое приложение для задания дискретных сигналов

Класс **ImitSignals** для тестирования РПО, который генерирует сигналы объектам дискретных сигналов при нажатии на кнопки.

4. Компонент хранилище (Storage)

Компонент хранилище (Storage) — предназначен для сохранения текущего состояния базы сообщений в файл с заданными интервалами времени, а также загрузки сообщений при запуске или перезагрузке прибора.

Имя класса	Назначение (краткое описание)
FileStorage	предназначен для сохранения(загрузки) текущего состояния базы сообщений в(из) файлы(ов).

5. Графическая часть приложения (Archive)

Графическая часть приложения — представляет собой архив информационных и аварийно-предупредительных сообщений в виде строк с необходимой информацией, а также кнопок для фильтрации по необходимым критериям посредством интерфейса, предоставленного компонентом API.

Имя класса	Назначение (краткое описание)
ArchLine	предназначен для представления строки в архиве на экране монитора. Создает и управляет объектами класса **QLabel**. Хранит данные необходимые для отображения на экране монитора.
Archive	наследуется от класса **QMainWindow**, предназначен для отображения строк с сообщениями (объекты класса **ArchLine**) с актуальной информацией предоставляемой компонентом **API**. Предоставляет интерфейс в виде кнопок управления оператору ПУ с помощью которых возможно фильтровать, квитировать сообщения, а также перелистывать страницы сообщений если общее количество сообщений не помещается на текущей странице.

Тип сообщения

Время формирования

Текст сообщения

Время квитирования

Время нормализации сигнала

Номер сообщения

Архив сообщений

№	ТИП	СФОРМИРОВАНО	СООБЩЕНИЕ	КВИТИРОВАНО	НОРМАЛИЗОВАНО
1	ИС	08:02:38 27.01.21	Система 2: Информационное сообщение2	08:03:02 27.01.21	08:02:38 27.01.21
2	ПС	08:02:40 27.01.21	Система 2: Предупредительное сообщение2	08:03:02 27.01.21	08:02:56 27.01.21
3	АС	08:02:42 27.01.21	Система 1: Аварийное сообщение1	08:03:03 27.01.21	08:02:52 27.01.21
4	ПС	08:02:43 27.01.21	Система 1: Предупредительное сообщение2	08:03:02 27.01.21	08:02:54 27.01.21
5	ПС	08:02:45 27.01.21	Система 1: Предупредительное сообщение1	08:03:03 27.01.21	08:02:55 27.01.21
6	АС	08:02:46 27.01.21	Система 1: Аварийное сообщение2	08:03:02 27.01.21	08:02:54 27.01.21
7	АС	08:02:47 27.01.21	Система 2: Аварийное сообщение1	08:03:01 27.01.21	08:02:56 27.01.21
8	АС	08:02:48 27.01.21	Система 2: Аварийное сообщение2	08:03:02 27.01.21	08:02:57 27.01.21
9	АС	08:03:09 27.01.21	Система 1: Аварийное сообщение1	08:03:44 27.01.21	08:03:23 27.01.21
10	ПС	08:03:10 27.01.21	Система 2: Предупредительное сообщение2	08:03:43 27.01.21	08:21:19 27.01.21
11	ПС	08:03:11 27.01.21	Система 2: Предупредительное сообщение1	08:03:44 27.01.21	08:03:23 27.01.21
12	ПС	08:03:12 27.01.21	Система 1: Предупредительное сообщение1	08:03:43 27.01.21	08:03:22 27.01.21
13	ИС	08:03:13 27.01.21	Система 1: Информационное сообщение2	08:03:43 27.01.21	08:03:13 27.01.21
14	ИС	08:03:15 27.01.21	Система 2: Информационное сообщение2	08:03:44 27.01.21	08:03:15 27.01.21
15	АС	08:03:17 27.01.21	Система 2: Аварийное сообщение1	08:03:44 27.01.21	08:21:21 27.01.21
16	АС	08:03:18 27.01.21	Система 2: Аварийное сообщение2	08:03:42 27.01.21	08:21:21 27.01.21
17	АС	08:03:19 27.01.21	Система 1: Аварийное сообщение2	08:03:44 27.01.21	08:03:23 27.01.21
18	ПС	08:03:20 27.01.21	Система 1: Предупредительное сообщение2	08:03:42 27.01.21	08:03:22 27.01.21
19	ИС	08:03:21 27.01.21	Система 2: Информационное сообщение1	08:03:43 27.01.21	08:03:21 27.01.21
20	ИС	08:03:21 27.01.21	Система 1: Информационное сообщение1	08:03:44 27.01.21	08:03:21 27.01.21
21	ПС	08:03:24 27.01.21	Система 2: Предупредительное сообщение1	08:03:43 27.01.21	08:21:19 27.01.21
22	ИС	08:21:07 27.01.21	Система 2: Информационное сообщение2		08:21:07 27.01.21
23	ИС	08:21:08 27.01.21	Система 2: Информационное сообщение1		08:21:08 27.01.21
24	ПС	08:21:10 27.01.21	Система 1: Предупредительное сообщение2		08:21:13 27.01.21
25	АС	08:21:11 27.01.21	Система 1: Аварийное сообщение1		08:21:13 27.01.21
26	АС	08:21:12 27.01.21	Система 1: Аварийное сообщение2		08:21:14 27.01.21
27	ПС	08:21:12 27.01.21	Система 1: Предупредительное сообщение1		08:21:15 27.01.21
28	ИС	08:21:17 27.01.21	Система 1: Информационное сообщение2		08:21:17 27.01.21
29	ПС	08:21:17 27.01.21	Система 1: Предупредительное сообщение1		08:21:24 27.01.21
30	ПС	08:21:18 27.01.21	Система 1: Предупредительное сообщение2		08:21:24 27.01.21
31	ПС	08:21:22 27.01.21	Система 2: Предупредительное сообщение2		08:21:27 27.01.21
32	ПС	08:21:23 27.01.21	Система 2: Предупредительное сообщение1		08:21:26 27.01.21
33	АС	08:21:24 27.01.21	Система 2: Аварийное сообщение1		08:21:28 27.01.21
34	АС	08:21:24 27.01.21	Система 1: Аварийное сообщение1		08:21:32 27.01.21
35	ПС	08:21:29 27.01.21	Система 2: Предупредительное сообщение2		08:21:46 27.01.21
36	ПС	08:21:29 27.01.21	Система 2: Предупредительное сообщение1		08:21:43 27.01.21
37	ПС	08:21:30 27.01.21	Система 1: Предупредительное сообщение2		08:21:32 27.01.21
38	ПС	08:21:31 27.01.21	Система 1: Предупредительное сообщение1		08:21:33 27.01.21
39	ИС	08:21:33 27.01.21	Система 1: Информационное сообщение2		08:21:33 27.01.21
40	ПС	08:21:34 27.01.21	Система 1: Предупредительное сообщение1		08:21:47 27.01.21
41	ПС	08:21:34 27.01.21	Система 1: Предупредительное сообщение2		08:21:37 27.01.21
42	АС	08:21:36 27.01.21	Система 1: Аварийное сообщение1		08:21:50 27.01.21

АКТИВНЫЕ СООБЩЕНИЯ | ИНФОРМАЦИОННЫЕ СООБЩЕНИЯ | АВАРИЙНЫЕ СООБЩЕНИЯ | ПРЕДУПРЕДИТЕЛЬНЫЕ СООБЩЕНИЯ | СИСТЕМА 1 | СИСТЕМА 2

КВИТИРОВАТЬ ВСЕ НА СТРАНИЦЕ | КВИТИРОВАТЬ ВСЕ

◀ Страница 1 Всего 2 ▶

Фильтрация сообщений

Квитирование сообщений

Переключение страниц

Проверка работоспособности и соответствия ТЗ

РПО было протестировано на нескольких виртуальных машинах, соединенных по сети Ethernet.

- Для имитации дополнительно было создано графическое приложение для взаимодействия с объектами дискретных сигналов;
- Произведена имитация входных дискретных сигналов, при этом проверено создание соответствующих записей в разделяемой памяти прибора приемника;
- Проверено распределение сообщений между узлами сети и корректное отображение сообщений в графическом окне на виртуальных машинах;
- Протестированы управляющие сигналы из графического приложения – квитирование и фильтрация сообщений, при этом проверена синхронизация квитированных сообщений между узлами сети и обновление отображения информации на экране монитора.
- В процессе работы ПО было проверено сохранение с заданными промежутками времени (10 с) сохранение базы сообщений в файл.
- При перезагрузке виртуальных машин была проверена загрузка сообщений из файлов и отображение на экране сохраненных сообщений.

Полученные результаты соответствуют требованиям технического задания на разрабатываемое ПО.

При разработке были исследованы и применены следующие программные средства (классы, модули):

- модуль **Qt network**, в части обмена по сети с помощью UDP, и применены классы **QUdpSocket**, **QHostInfo**;

- класс **QThread** для выполнения частей ПО в разных потоках, классы **QMutex**, **QMutexLocker** (идиома RAII) для недопущения проблем синхронизации данных между потоками;

- класс **QSharedMemory** для работы разных приложений ПО с общей разделяемой памятью, класс **QSystemSemaphore** для корректной защиты инициализации разделяемой памяти между приложениями ПО;

- класс **QSettings**, **QFile**, **QRegExp** применялись для работы с конфигурационными файлами для инициализации текстовой информации сообщений в контейнер;

- функции **qCompress**, **qUncompress** для сжатия данных при передаче по сети.

Результаты работы

- Разработана архитектура ПО
- Разработаны классы для работы с разделяемой памятью ПО и выполнения алгоритмов с данными
- Разработаны классы для обмена между узлами сети Ethernet
- Разработан класс для сохранения/загрузки данных из файлов
- Разработаны графические классы для отображения данных на экранах мониторов и задания команд управления
- Проверена работа приложения на разных операционных системах (Windows и QNX6.5)
- Проверены все аспекты функциональных возможностей разрабатываемого ПО