

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА**  **Российский технологический университет»**  **РТУ МИРЭА** |

**Институт искусственного интеллекта (ИИИ) Кафедра промышленной информатики (ПИ)**

**ПРАКТИЧЕСКАЯ РАБОТА 3**

**по дисциплине**

**«Разработка автоматизированных систем реального времени»**

**Тема: Разработка автоматизированной системы реального времени технологического процесса стерилизации консервов**

|  |  |
| --- | --- |
| Отчет представлен к рассмотрению:  Студент группы КВБО-03-21 | Батурин М.П. |
| (подпись) |  |
| Преподаватель |  |
|  | Зорина Н.В. |
| (подпись) |  |

Содержание

[3 Разработка АС 3](#_Toc194349224)

[3.1 Выбор средств ведения разработки 3](#_Toc194349225)

[3.2 Разработка БД 5](#_Toc194349226)

[3.3 Разработка системы автоматизации 7](#_Toc194349227)

[3.4 Обработка данных в режиме реального времени 7](#_Toc194349228)

[3.5 Разработка интерфейса 8](#_Toc194349229)

[Заключение 13](#_Toc194349230)

[Приложение А 14](#_Toc194349231)

[Функция обработки сообщений с Mqqt брокера 14](#_Toc194349232)

[Приложение Б 15](#_Toc194349233)

[Код класса, отвечающего за работу по web socket 15](#_Toc194349234)

# Разработка АС

## Выбор средств ведения разработки

Для создания системы мониторинга был выбран современный стек технологий, обеспечивающий надежность, масштабируемость и высокую производительность приложения. Основным языком разработки является Python, который обладает богатой экосистемой библиотек для работы с данными и построения веб-приложений.

В качестве фреймворка для создания веб-интерфейса и API выбран FastAPI — современный, высокопроизводительный веб-фреймворк с поддержкой асинхронных операций. FastAPI обеспечивает автоматическую генерацию документации API, валидацию данных и типизацию, что значительно снижает количество ошибок на этапе разработки. Выбор FastAPI позволил реализовать как API для взаимодействия с системой, так и веб-интерфейс для мониторинга производства и управления параметрами.

Для хранения данных используется PostgreSQL ― надежная, проверенная временем реляционная СУБД с открытым исходным кодом. PostgreSQL обеспечивает транзакционную целостность данных, поддержку JSON для хранения сложных структур данных датчиков и высокую производительность при обработке больших объемов информации. Взаимодействие с базой данных осуществляется через асинхронный драйвер asyncpg и ORM SQLAlchemy, что позволяет эффективно работать с данными даже при высоких нагрузках.

Архитектура системы основана на принципах событийно-ориентированного программирования и использует брокеры сообщений для обеспечения надежного обмена данными между компонентами. Для получения данных с датчиков применяется протокол MQTT — легковесный протокол обмена сообщениями, который идеально подходит для устройств Интернета вещей и промышленных датчиков. В качестве MQTT брокера используется Mosquitto. Обработка потоков данных осуществляется с помощью Apache Kafka, обеспечивающей надежную доставку сообщений и масштабируемость при растущих объемах данных.

Для обеспечения безопасности в системе реализована аутентификация на основе JWT (JSON Web Tokens). Это позволяет обеспечить безопасный доступ к функциям системы в соответствии с ролями пользователей (администратор, оператор, технический специалист, менеджер).

Интерфейс пользователя построен с использованием современных веб-технологий HTML5, CSS3 и JavaScript. Для улучшения внешнего вида и создания адаптивного интерфейса применяется фреймворк Bootstrap, обеспечивающий единообразие элементов управления и правильное отображение на различных устройствах. Обновление данных в реальном времени реализовано через WebSockets, что позволяет моментально отображать изменения состояния датчиков и оповещения без необходимости обновления страницы.

Программная архитектура системы построена по модульному принципу и состоит из следующих основных компонентов:

* Модуль сбора данных датчиков — отвечает за получение информации с физических датчиков или симулятора через протокол MQTT.
* Модуль обработки и анализа данных — выполняет первичную обработку данных, проверяет соответствие показаний допустимым значениям и генерирует оповещения при отклонениях от нормы.
* Модуль хранения данных — обеспечивает сохранение информации в базе данных PostgreSQL и предоставляет интерфейс для выполнения запросов.
* Модуль оповещений — формирует уведомления о критических ситуациях и отклонениях в работе оборудования.
* Модуль аутентификации и авторизации — управляет процессами идентификации пользователей и контролирует доступ к функциям системы.
* Веб-модуль — предоставляет интерфейс пользователя для мониторинга и управления производственным процессом.
* API модуль — обеспечивает программный интерфейс для интеграции с другими системами.

## Разработка БД

Центральным элементом модели является таблица датчиков (sensors), которая содержит информацию о всех установленных на производстве измерительных устройствах. Датчики расположены в определенных местах производственной линии, что отражено в связи с таблицей местоположений (location). Для каждого датчика определены допустимые диапазоны значений и интервалы измерений, хранящиеся в таблице настроек оборудования (equipment\_settings).

Таблица показаний датчиков (sensor\_readings) представляет собой основное хранилище данных, поступающих от датчиков в реальном времени. Показания хранятся в формате строки JSON, что обеспечивает гибкость при работе с различными типами датчиков и форматами данных. При отклонении показаний от нормы система генерирует события, которые сохраняются в таблице событий (events).

Доступ к системе контролируется через таблицы сотрудников (employees) и ролей (roles), обеспечивая разграничение прав в соответствии с должностными обязанностями пользователей.

Sql скрипт создания таблиц приведен на Рисунке 3 .1.

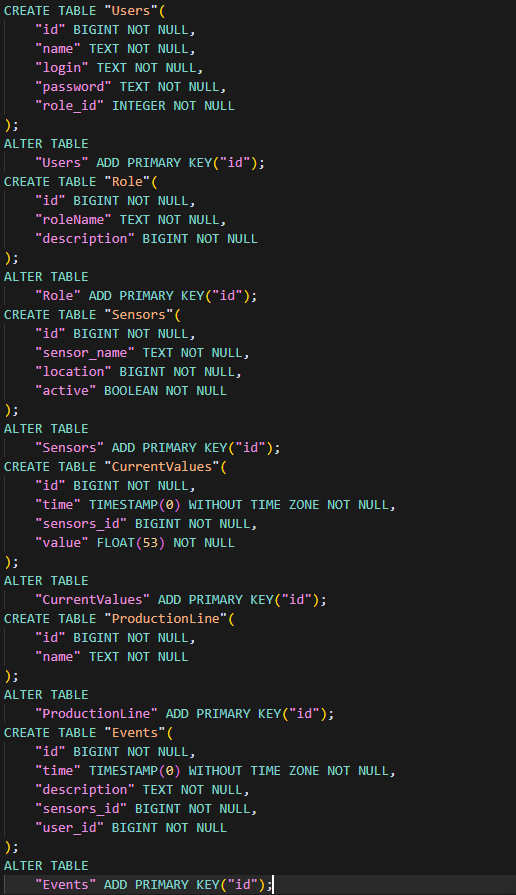


Рисунок . — Sql скрипт создания таблиц

## Разработка системы автоматизации

Разработка системы включила несколько ключевых компонентов, реализующих основную функциональность - от сбора данных с датчиков до их обработки и отображения в веб-интерфейсе. В данном разделе приведено описание процесса разработки основных модулей системы и их взаимодействия. Сбор данных с датчиков осуществляется через протокол MQTT, обеспечивающий надежную передачу сообщений даже в условиях нестабильного сетевого соединения. Для реализации клиента MQTT использовалась библиотека paho-mqtt с асинхронной обработкой сообщений.

Функция обработки сообщений с Mqqt брокера приведена в Приложении А.

Разработанный MQTT клиент обеспечивает:

* подключение к MQTT брокеру;
* подписку на необходимые топики;
* обработку входящих сообщений;
* отказоустойчивость при разрыве соединения.

Ключевым компонентом бизнес-логики является модуль обработки и анализа данных, который анализирует поступающие значения и генерирует оповещения при выходе показаний за допустимые пределы.

API системы реализовано с использованием FastAPI и предоставляет доступ к данным о текущем состоянии производства, истории показаний датчиков и оповещениях. API поддерживает JWT-аутентификацию и разграничение доступа по ролям.

## Обработка данных в режиме реального времени

Обработка данных в режиме реального времени является ключевым аспектом разработанной системы. В этом разделе рассматривается архитектура и реализация компонентов, обеспечивающих сбор, обработку, анализ и визуализацию данных в реальном времени.

Система обработки данных в режиме реального времени построена на принципах событийно-ориентированной архитектуры (Event-Driven Architecture), где каждое изменение показаний датчика рассматривается как событие, требующее обработки.

Такая архитектура обеспечивает:

* низкую задержку между получением данных и их отображением;
* масштабируемость системы при увеличении количества датчиков;
* устойчивость к сбоям отдельных компонентов;
* возможность параллельной обработки больших объемов данных.

MQTT (Message Queuing Telemetry Transport) был выбран как основной протокол для сбора данных с датчиков благодаря его легковесности, надежности и поддержке модели публикации/подписки. Система использует иерархическую структуру топиков для организации потоков данных от различных участков производства.

Для обеспечения обновления данных в веб-интерфейсе без перезагрузки страницы используется технология WebSocket. Код класса, отвечающего за работу по web socket, приведен в Приложении Б.

Реализация WebSocket сервера поддерживает следующие функции:

* подключение клиентов к различным каналам данных;
* широковещательная рассылка обновлений;
* целевая отправка оповещений конкретным клиентам.

## Разработка интерфейса

Разработка интерфейсов для системы была ориентирована на предоставление удобного, информативного и эффективного инструмента для работы операторов и руководителей производства. Основной целью являлось создание интуитивно понятного веб-интерфейса, обеспечивающего оперативный мониторинг и управление процессами производства в режиме реального времени.

Для разработки фронтенд-части системы были использованы следующие технологии:

* HTML5/CSS3/JavaScript — основа веб-интерфейса.
* Bootstrap 5 — фреймворк для создания адаптивного и современного дизайна.
* Chart.js — библиотека для визуализации данных.
* WebSocket API — для обеспечения обновления данных в реальном времени.
* Jinja2 — шаблонизатор для генерации HTML на стороне сервера.
* FontAwesome — набор иконок для улучшения визуального восприятия интерфейса.
* Fetch API — для асинхронного взаимодействия с REST API сервера.

**Структура пользовательского интерфейса.**

Интерфейс системы организован в виде единой веб-панели управления с различными функциональными разделами (Рисунок 3.2).

Основная структура включает:

* Навигационное меню — обеспечивает быстрый доступ ко всем разделам системы
* Информационная панель (дашборд) — отображает общее состояние производства.
* Детальные страницы датчиков — для просмотра подробной информации по каждому датчику.
* Раздел оповещений и событий — для отслеживания нештатных ситуаций.
* Аналитические отчеты и графики — для анализа производственных показателей.
* Панель настроек — для конфигурирования параметров системы.
* Страница авторизации — для аутентификации пользователей с разными уровнями доступа.

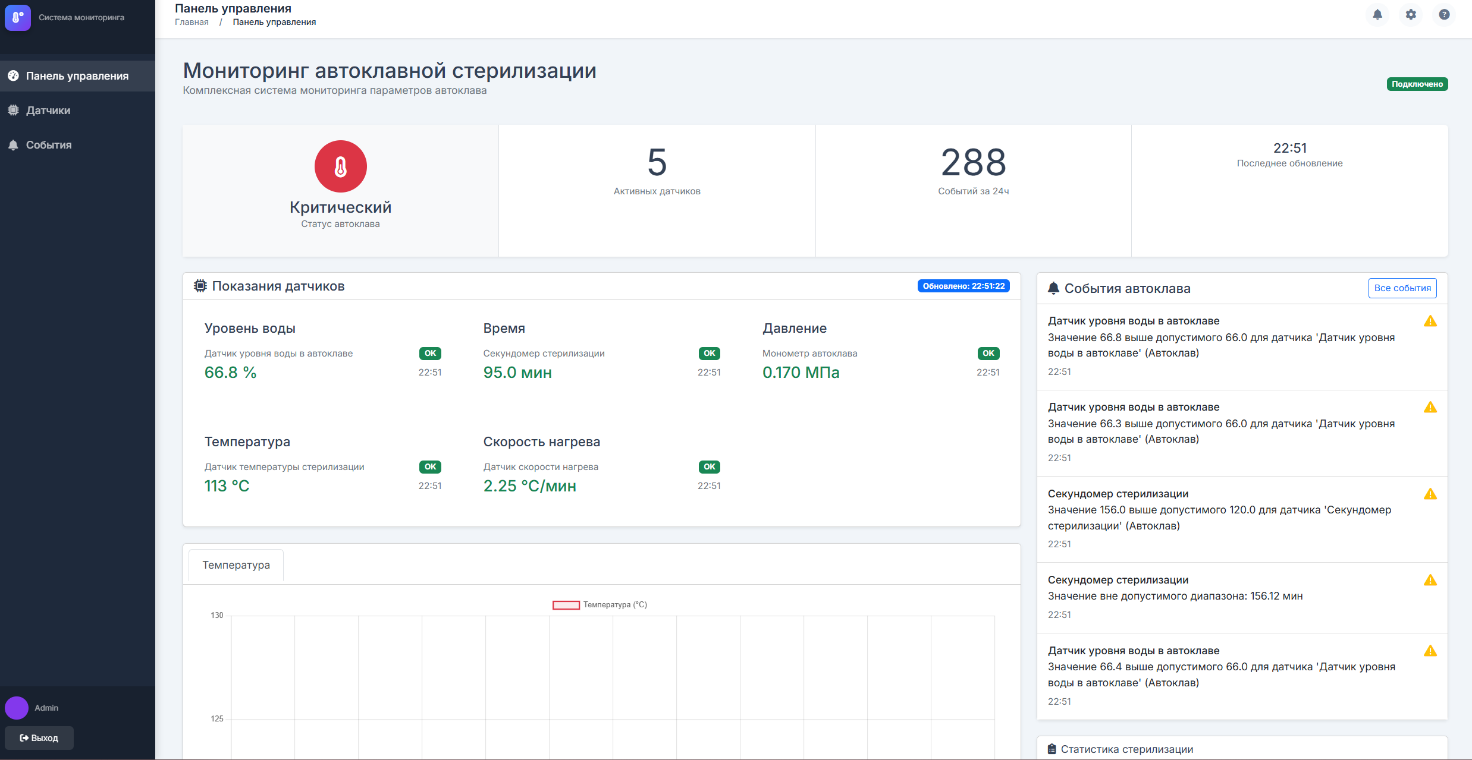


Рисунок . — Интерфейс автоматизированной системы

Форма аутентификации пользователя приведена на Рисунке 3.3.

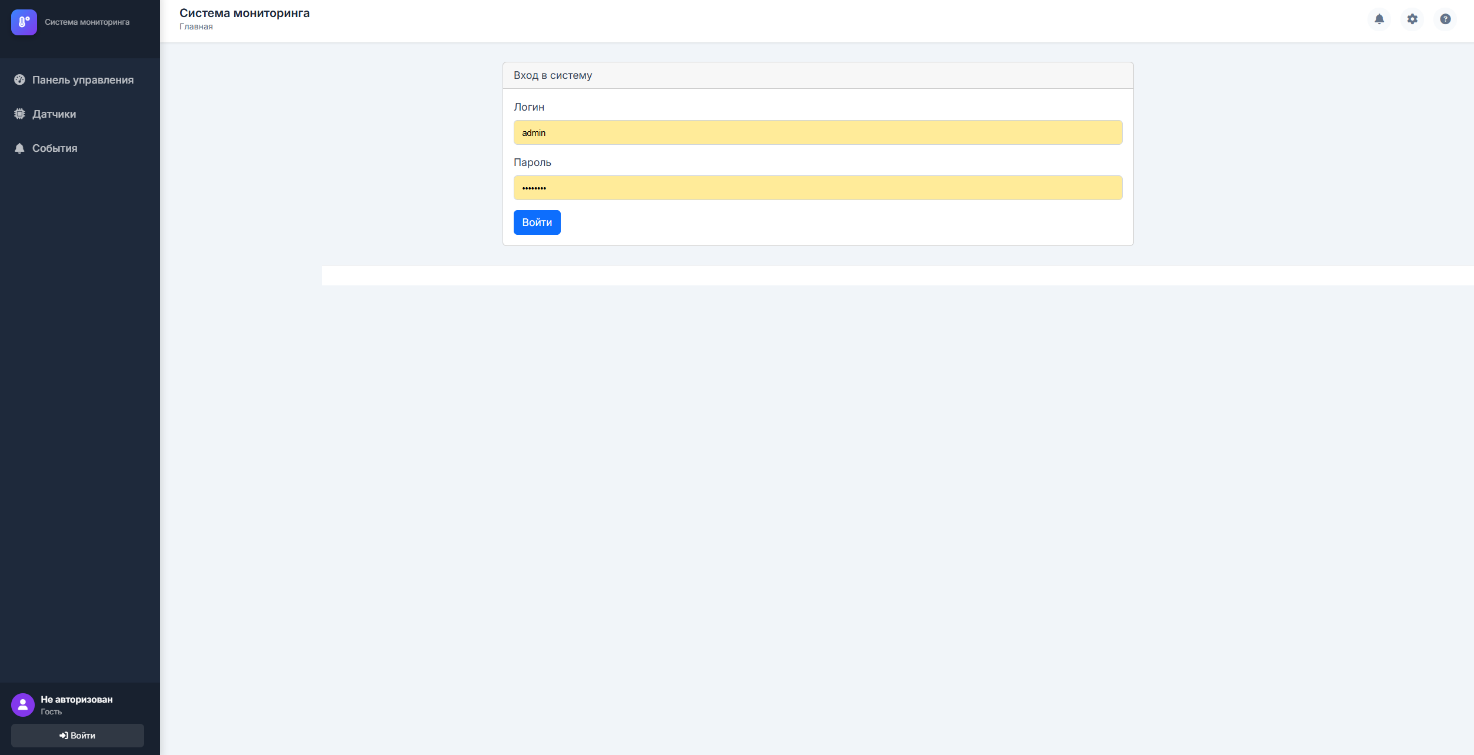


Рисунок . — Форма аутентификации пользователя

Форма мониторинга датчиков приведена на Рисунке 3.4.

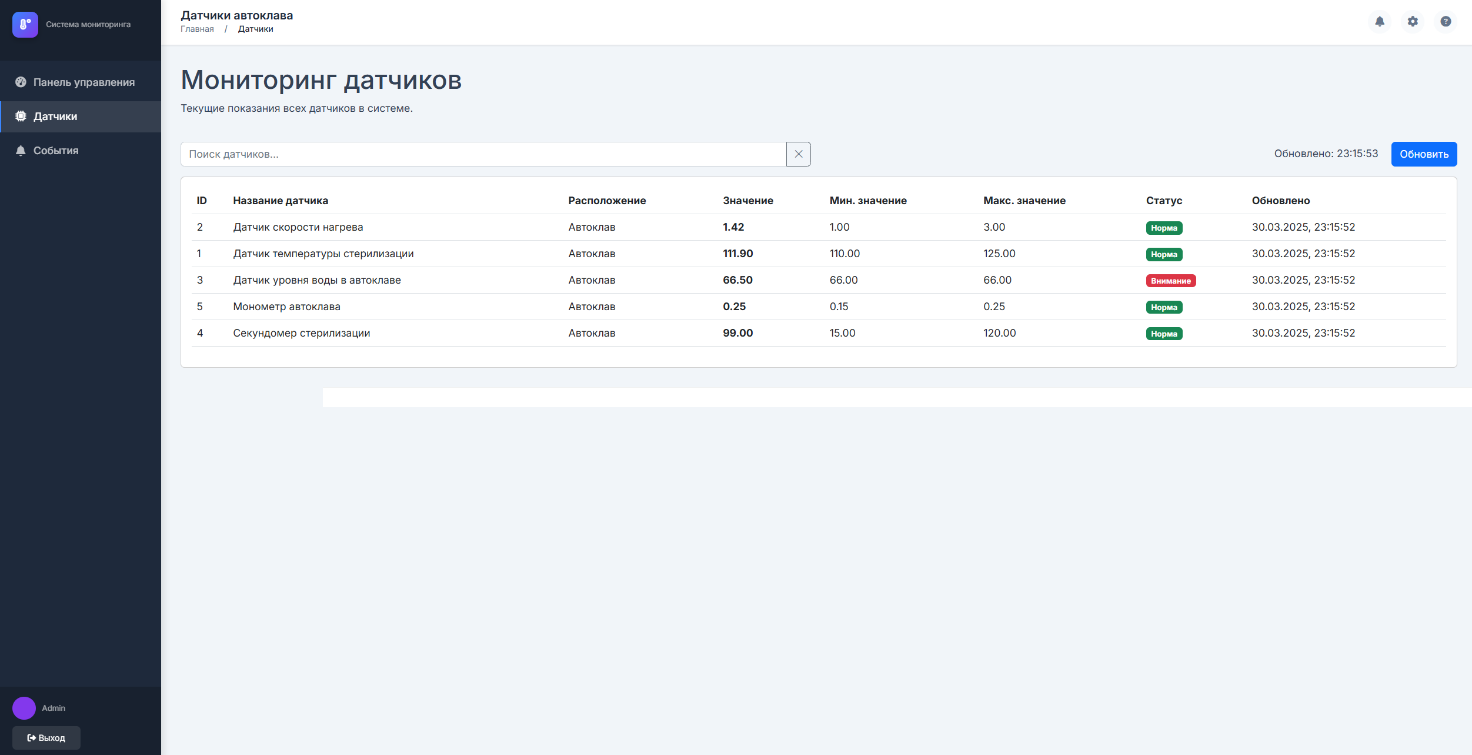


Рисунок . — Мониторинг датчиков

Заключение

В рамках данной работы создана комплексная система автоматизации мониторинга за процессом стерилизации свиных консервов, обеспечивающая мониторинг, анализ и управление производственными процессами в режиме реального времени. Система реализована на основе событийно-ориентированной архитектуры с использованием современных технологий: Python, FastAPI, PostgreSQL, MQTT, Apache Kafka и WebSocket.

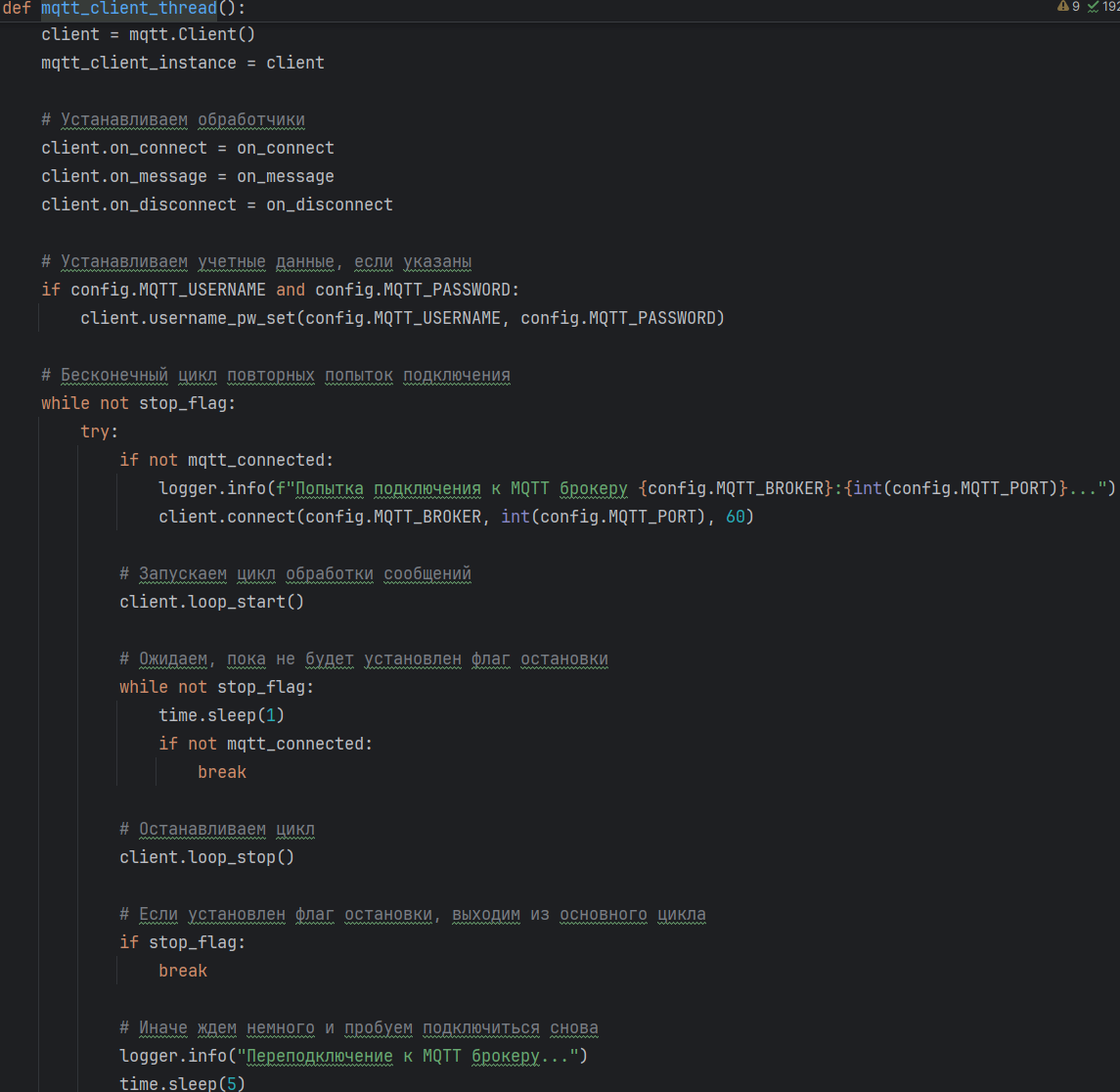
Созданная система решает основные производственные задачи: непрерывный мониторинг всех этапов производства, раннее выявление отклонений, повышение эффективности производства и снижение влияния человеческого фактора.

Практическая значимость разработки заключается в возможности снижения брака, уменьшения энергозатрат, оптимизации режимов работы оборудования, увеличения производительности и сокращения простоев.

Результат работы представляет собой готовое к промышленному внедрению решение, объединяющее современные технологии программной инженерии с глубоким пониманием технологического процесса стерилизации консервов.

Приложение А

Функция обработки сообщений с Mqqt брокера



Приложение Б

Код класса, отвечающего за работу по web socket

