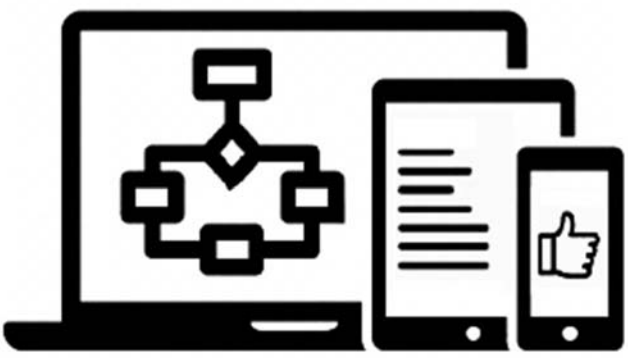
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ПРОМЫШЛЕННЫЙ ИНТЕРНЕТ ВЕЩЕЙ

Методические рекомендации к лабораторным работам для руководящих работников и специалистов, обучающихся по тематике «Промышленный интернет вещей в рамках реализации концепции „Индустрия 4.0”» (для специалистов организаций промышленного профиля)



Могилев 2026

УДК 004.4:65.011.56
ББК 32.973.018:30.12
П81

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «4» февраля 2026 г., протокол № 7

Составители: канд. техн. наук, доц. Н. Н. Горбатенко;
канд. техн. наук, доц. В. В. Кутузов

Рецензент канд. техн. наук, доц. М. А. Рабыко

Методические рекомендации к выполнению лабораторных работ по дисциплине «Промышленный интернет вещей».

Учебное издание

ПРОМЫШЛЕННЫЙ ИНТЕРНЕТ ВЕЩЕЙ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 26 экз. Заказ № .

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Лабораторная работа № 1. Архитектура промышленного интернета вещей

Цель работы: изучить концепцию промышленного интернета вещей (IIoT); освоить многоуровневую архитектуру IIoT-систем; проанализировать компоненты каждого уровня и разработать структурную схему IIoT-решения для конкретного промышленного объекта.

Теоретические сведения

Промышленный интернет вещей (*IIoT – Industrial Internet of Things*) – это многоуровневая система, объединяющая датчики, контроллеры, исполнительные устройства, средства передачи данных и программные платформы для мониторинга, анализа и управления промышленными процессами в режиме реального времени.

Отличия IIoT от потребительского IoT:

– высокие требования к надёжности, отказоустойчивости и безопасности;

– работа в экстремальных условиях (температура, вибрация, влажность);

– требования к детерминированности и минимальным задержкам передачи данных;

– интеграция со SCADA-системами;

– длительный жизненный цикл оборудования (10–20 лет).

Существуют несколько общепринятых моделей архитектуры (таблица 1.1).

Таблица 1.1 – Архитектурные модели IIoT

Модель	Уровень	Особенность
3-уровневая	Устройства → Сеть → Приложения	Упрощённая модель для базового понимания
4-уровневая	Восприятие → Сеть → Обработка → Приложения	Наиболее распространённая в учебной литературе
RAMI 4.0	3D-модель: иерархия, жизненный цикл, слои взаимодействия	Стандарт немецкой архитектуры промышленного интернета вещей в рамках концепции «Индустрия 4.0»
IIRA	Бизнес → Использование → Функциональность → Реализация → Доверие	Эталонная архитектура промышленного интернета

В рамках лабораторной работы будем использовать 4-уровневую модель, как наиболее наглядную для целей обучения.

Анализ уровней архитектуры IIoT.

Уровень 1. Устройства (Perception Layer / Edge Layer).

Компоненты: датчики (температуры, давления, вибрации, уровня), исполнительные механизмы (актуаторы), программируемые логические контроллеры (ПЛК), шлюзы IIoT, edge-устройства.

Функции: сбор первичных данных, предварительная фильтрация и агрегация, локальное управление.

Технологии: аналоговые/цифровые датчики, протоколы Modbus, OPC UA, MQTT-SN.

Уровень 2. Сеть (Network / Connectivity Layer).

Компоненты: проводные (Ethernet, Industrial Ethernet, PROFIBUS) и беспроводные (Wi-Fi 6, LoRaWAN, NB-IoT, 5G) сети передачи данных, маршрутизаторы, коммутаторы.

Функции: надёжная передача данных между уровнями, обеспечение требуемой пропускной способности и минимальных задержек.

Особенности: сегментация сети, приоритизация трафика (QoS), резервирование каналов связи.

Уровень 3. Платформа обработки данных (Platform / Processing Layer).

Компоненты: облачные платформы (AWS IoT, Azure IoT, MindSphere), on-premise-серверы, системы хранения данных (тайм-серия базы данных), средства аналитики и машинного обучения.

Функции: хранение исторических данных, обработка больших массивов информации, визуализация, прогнозная аналитика, цифровые двойники.

Технологии: Big Data (Apache Kafka, Spark), time-series DB (InfluxDB, TimescaleDB), ML-фреймворки.

Уровень 4. Приложения (Application Layer).

Компоненты: веб- и мобильные интерфейсы, системы управления производством (MES), системы управления предприятием (ERP), специализированные приложения (предиктивное обслуживание, оптимизация энергопотребления).

Функции: предоставление бизнес-логики, принятие решений на основе анализа данных, интеграция с корпоративными системами.

Уровень 5. Безопасность (сквозной уровень).

Принцип: безопасность реализуется на всех уровнях архитектуры, а не как отдельный слой.

Меры защиты:

– уровень устройств: аутентификация устройств, secure boot, шифрование на уровне микроконтроллера;

– уровень сети: шифрование трафика (TLS/DTLS), сегментация, межсетевые экраны;

– уровень платформы: управление доступом (RBAC), аудит, защита от DDoS;

– уровень приложений: двухфакторная аутентификация, защита персональных данных.

Примеры применения IIoT приведены в таблице 1.2.

Задание для самостоятельного выполнения

Выберите один из вариантов промышленных объектов (по указанию преподавателя) из таблицы 1.3 и разработайте структурную схему IIoT.

Таблица 1.2 – Архитектурные модели IIoT

Отрасль	Применение	Эффект
Производство	Предиктивное обслуживание оборудования на основе анализа вибрации и температуры	Снижение простоев на 20 %...40 %, увеличение срока службы оборудования
Энергетика	Умные сети (Smart Grid) с датчиками качества электроэнергии и автоматическим управлением нагрузкой	Снижение потерь энергии на 15 %, повышение надёжности сети
Нефтегаз	Мониторинг состояния трубопроводов с помощью распределённых акустических датчиков	Раннее обнаружение утечек, предотвращение аварий
Транспорт	Умная логистика: отслеживание местоположения, температуры и влажности грузов в реальном времени	Снижение потерь грузов, оптимизация маршрутов
Сельское хозяйство	Прецезионное земледелие: датчики влажности почвы, автоматизированный полив	Экономия воды до 30 %, повышение урожайности

Таблица 1.3 – Варианты заданий архитектурных моделей IIoT

Вариант	Объект	Задача мониторинга/управления
1	Цех металлообработки	Контроль износа режущего инструмента и предиктивное обслуживание станков с ЧПУ
2	Насосная станция водоснабжения	Мониторинг давления, расхода воды и энергопотребления насосов
3	Складские помещения	Контроль температуры, влажности и уровня заполнения ёмкостей
4	Ветровая электростанция	Сбор данных о скорости ветра, нагрузке на лопасти и генерируемой мощности
5	Конвейерная линия упаковки	Отслеживание скорости конвейера, количества упакованных изделий, выявление брака

Разработанная структурная схема должна содержать следующее.

1 Уровень устройств:

– перечень конкретных датчиков и актуаторов с указанием измеряемых параметров;

– типы интерфейсов подключения (аналоговый 4…20 мА, цифровой Modbus RTU и др.);

– места установки на технологическом объекте.

2 Уровень сети:

– типы используемых сетевых технологий (проводные/беспроводные);

– топология сети (звезда, кольцо);

– узлы агрегации данных (шлюзы, edge-серверы).

3 Уровень платформы:

– выбор облачной/on-premise платформы;

– схема потоков данных (какие данные куда передаются);

– функции обработки (хранение, аналитика, визуализация).

4 Уровень приложений:

– типы конечных приложений (панель мониторинга, система оповещений, отчёты);

– пользовательские роли и права доступа.

5 Безопасность:

– меры защиты на каждом уровне (минимум по две меры на уровень);

– схема управления ключами и сертификатами.

Порядок выполнения работы

1 Подготовительный этап:

– изучите теоретические материалы лабораторной работы;

– получите вариант задания у преподавателя.

2 Анализ объекта:

– определите ключевые параметры для мониторинга/управления;

– выявите требования к надёжности, задержкам и объёму данных;

– составьте перечень необходимых датчиков и актуаторов.

3 Проектирование архитектуры:

– нарисуйте блок-схему с выделением четырех уровней архитектуры;

– укажите конкретные технологии и протоколы для каждого уровня;

– добавьте элементы обеспечения безопасности.

4 Оформление отчёта.

Контрольные вопросы

1 В чём принципиальное отличие архитектуры IIoT от потребительского IoT?

2 Почему безопасность в IIoT рассматривается как сквозной уровень, а не отдельный слой?

3 Какие требования предъявляются к сетевому уровню IIoT в условиях производственного предприятия?

4 Что такое «цифровой двойник» и на каком уровне архитектуры он реализуется?

5 Какие протоколы применяются на уровне устройств в промышленных условиях и почему?

6 В чём преимущества гибридной архитектуры (edge + cloud) перед чисто облачной?

7 Как обеспечивается совместимость новых IIoT-устройств со старыми системами управления (SCADA, АСУ ТП)?

8 Приведите пример киберфизической атаки на промышленную систему и меры защиты от неё.

2 Лабораторная работа № 2. Принципы функционирования IIoT-систем

Цель работы: изучить архитектуру промышленных систем интернета вещей (IIoT); проанализировать этапы жизненного цикла данных (сбор → передача → обработка → анализ); освоить принципы распределения вычислительных функций между уровнями «устройство – периферия – облако» на примерах реальных промышленных сценариев.

Теоретические сведения

Архитектура IIoT по модели IIRA.

Согласно эталонной архитектуре промышленного интернета вещей (IIRA), система состоит из трёх основных уровней (таблица 2.1).

Таблица 2.1 – Основные уровни IIoT

Уровень	Функция	Типичная задача
Edge Layer (периферия)	Сбор данных, предварительная обработка, управление в реальном времени	Фильтрация «мусорных» данных, детекция аномалий, локальное управление исполнительными механизмами
Platform Layer (платформа)	Агрегация, хранение, аналитика среднего уровня	Корреляция данных с нескольких устройств, машинное обучение, цифровые двойники
Enterprise Layer (предприятие)	Бизнес-аналитика, интеграция с ERP/MES	Прогнозирование спроса, оптимизация цепочек поставок, отчётность

Этапы жизненного цикла данных в IIoT представлены на рисунке 2.1.

```
[Устройства] → СБОР → [Сеть] → ПЕРЕДАЧА → [Edge] → ПЕРВИЧНАЯ ОБРАБОТКА
      ↓
[Облако/Платформа] ← АГРЕГАЦИЯ ← [Edge/Fog] ← ФИЛЬТРАЦИЯ
      ↓
[Аналитические сервисы] → АНАЛИЗ → [BI-системы] → ПРИНЯТИЕ РЕШЕНИЙ
```

Рисунок 2.1 – Этапы жизненного цикла данных в IIoT

Этап 1. Сбор данных осуществляется через:

1) датчики (температура, вибрация, давление, изображение);
2) промышленные контроллеры через интерфейсы OPC UA;
3) интеллектуальные счётчики и измерительные комплексы.

Этап 2. Передача данных осуществляется через:

1) протоколы: MQTT (Message Queuing Telemetry Transport) – легкий IoT-протокол «издатель-подписчик» для быстрой передачи небольших сообщений; OPC UA (Open Platform Communications Unified Architecture) – сложный

промышленный стандарт для безопасного, структурированного взаимодействия систем с контекстом данных. Оба обеспечивают связь устройств, но MQTT лучше для облаков и IoT, а OPC UA – для АСУ ТП;

2) сетевую инфраструктуру: промышленный Ethernet, 5G, LoRaWAN для удалённых объектов.

Этап 3. Обработка данных. Распределение по уровням вычислений представлено в таблице 2.2.

Таблица 2.2 – Уровни вычислений данных

Уровень	Задержка	Объём данных	Пример задачи
Устройство	< 1 мс	100 % «сырых» данных	Фильтрация шума, триггеры аварий
Edge/Fog	1...100 мс	10 %...30 % после фильтрации	Детекция аномалий, предиктивная аналитика в реальном времени
Облако	> 100 мс	< 5 % агрегированных данных	Обучение ML-моделей, сравнительный анализ между заводами

Выбор уровня обработки определяется критерием *задержки*: для управления роботом требуется обработка на уровне устройства/edge, для бизнес-аналитики – в облаке.

Этап 4. Анализ и визуализация.

1 Предиктивное обслуживание (анализ вибрации для прогноза отказа подшипников).

2 Цифровые двойники (цифровая копия физического актива для симуляции).

3 BI-дашборды: интерактивные панели управления, визуализирующие ключевые показатели в реальном времени, для операторов и менеджеров.

Примеры промышленного применения IIoT приведены в таблице 2.3.

Таблица 2.3 – Промышленное применение IIoT

Отрасль	Сценарий	Распределение функций
Машиностроение	Контроль качества на конвейере	Камера + edge-устройство (анализ изображения в реальном времени) → облако (статистика дефектов)
Энергетика	Мониторинг трансформаторных подстанций	Датчики → шлюз (предварительная обработка) → облако (прогноз нагрузки)
Нефтегазовая	Удалённый мониторинг скважин	Датчики давления/температуры → спутниковая связь → облако (аналитика добычи) stlpartners.com
Пищевая промышленность	Слежение за цепочкой хранения	RFID-метки → шлюз → блокчейн в облаке (подтверждение условий хранения)

Порядок выполнения работы

Этап 1. Изучение архитектуры.

1 Ознакомьтесь со схемой трёхуровневой архитектуры IIRA.

2 Заполните таблицу 2.4.

Таблица 2.4 – Соответствие функций уровням архитектуры

Функция системы	Уровень устройства	Уровень Edge/Fog	Уровень облака/ платформы	Обосно-вание
Фильтрация шумовых данных				
Детекция аномалий вибрации				
Обучение модели предиктивного обслу-живания				
Формирование ежедневного отчёта по энергопотреблению				
Аварийная остановка оборудования при превышении порога				

Этап 2. Анализ протоколов передачи.

1 Изучите характеристики протоколов MQTT и OPC UA.

2 Выполните задание: «*Обоснуйте выбор протокола для сценария*: *датчики вибрации на насосной станции с нестабильным 4G-соединением*».

Этап 3. Моделирование распределения задач.

Дана производственная система: 50 датчиков температуры (опрос 10 Гц), 10 камер контроля качества (30 кадров/с). Требуется детектировать перегрев оборудования за < 50 мс и ежедневная аналитика энергопотребления.

Задание для самостоятельного выполнения

Разработайте схему распределения задач между уровнями.

1 Какие данные обрабатываются на уровне устройства?

2 Какие – на уровне edge-шлюза?

3 Какие – в облаке?

4 Обоснуйте выбор с указанием требований к задержке и объёму данных.

Этап 4. Практическое задание.

Проанализируйте реальный кейс применения IIoT (выдаётся преподавателем): определите этапы жизненного цикла данных, выявите критические требования к задержке, предложите архитектурное решение с распределением функций.

Контрольные вопросы

1 В чём принципиальное отличие архитектуры IIoT от потребительского IoT?

2 Почему для управления промышленным роботом недостаточно облачной обработки?

3 Какие данные целесообразно фильтровать на уровне edge, а какие – передавать в облако?

4 Сравните протоколы MQTT и OPC UA по критериям: надёжность, безопасность, поддержка типизированных данных.

5 Приведите пример сценария, где требуется гибридная архитектура «edge + облако».

6 Как цифровой двойник использует данные с разных уровней архитектуры IIoT?

3 Лабораторная работа № 3. Архитектура и принципы работы программируемого логического контроллера

Цель работы: изучить назначение, архитектуру и принципы работы программируемых логических контроллеров на примере ОВЕН ПЛК210.

Теоретические сведения

Программируемый логический контроллер (ПЛК) – это специализированное микропроцессорное устройство, предназначенное для автоматизации технологических процессов в промышленности. ПЛК обеспечивает сбор данных от датчиков, их обработку по заданному алгоритму и управление исполнительными механизмами в реальном времени.

ОВЕН ПЛК210 представляет собой современный моноблочный контроллер российского производства, предназначенный для автоматизации средних и распределённых систем. Контроллер характеризуется высокой производительностью, расширенными сетевыми возможностями и поддержкой различных архитектурных решений, включая резервирование.

ОВЕН ПЛК210 выполнен в моноблочном исполнении и включает следующие ключевые компоненты (таблица 3.1).

Таблица 3.1 – Ключевые компоненты ОВЕН ПЛК210

Компонент	Характеристика
Центральный процессор	RISC-процессор с частотой 800 МГц logo-prom.by
Память	Оперативная память (RAM): до 2 ГБ. Флеш-память (ROM/NAND): до 8 ГБ. Retain-память (MRAM): 64…256 Кбайт (сохраняется при отключении питания)
Встроенные интерфейсы	2 × Ethernet (10/100 Мбит/с). RS-485 (для подключения периферии и связи по промышленным протоколам). USB
Встроенные каналы ввода/вывода	Дискретные входы/выходы и аналоговые входы/выходы, размещённые непосредственно на корпусе контроллера
Операционная система	Linux с RT-патчем (обеспечивает работу в реальном времени)
Габариты и масса	105 × 124 × 83 мм, масса ≤ 1,2 кг, степень защиты IP20

Аппаратная архитектура ПЛК210 включает.

1 Процессорный модуль – ядро системы, выполняющее пользовательскую программу.

2 Память – для хранения программы, данных и сохраняемых переменных.

3 Блок питания – преобразует входное напряжение (10…48 В, номинально 24 В) в напряжения, необходимые для работы внутренних узлов.

4 Цифровые и аналоговые каналы ввода/вывода – для подключения датчиков и исполнительных устройств.

5 Коммуникационные интерфейсы – для связи с АСУ ТП, HMI, другими ПЛК и сетевым оборудованием.

6 Часы реального времени (RTC) – для таймстемпинга событий и планирования задач.

Цикл работы любого ПЛК, включая ОВЕН ПЛК210, представляет собой повторяющуюся последовательность операций, выполняемых за фиксированный или переменный временной интервал (рисунок 3.1).

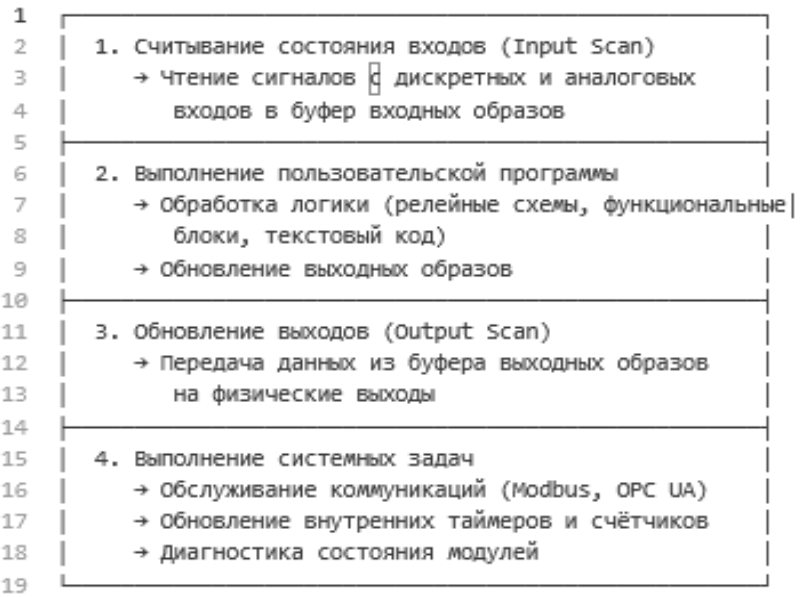


Рисунок 3.1 – Этапы цикла сканирования

Конфигурирование входных и выходных модулей

Для ОВЕН ПЛК210 доступны две среды программирования:

1) CODESYS – международная среда разработки (для модификаций ПЛК210, ПЛК210-KR);

2) Полигон: российская среда разработки (для модификаций ПЛК210-PL).

Конфигурирование аппаратной части выполняется через утилиту PLC Configuration (в CODESYS) или через иерархическую структуру проекта в среде Полигон.

Процедура конфигурирования.

Шаг 1. Создание проекта и выбор целевой платформы.

1 В CODESYS: Project → Create Project → выбрать ОВЕН ПЛК210 из списка

устройств.

2 Загрузить актуальный пакет таргет-файлов ОВЕН (версия 3.5.17.31 или более новая).

Шаг 2. Настройка встроенных модулей ввода/вывода.

1 Открыть вкладку Resources → PLC Configuration.

2 В дереве конфигурации выбрать модуль контроллера.

3 Для дискретных входов/выходов указать:

– тип сигнала (сухой контакт, потенциальный);

– фильтрацию помех (время задержки срабатывания).

4 Для аналоговых входов задать:

– тип датчика (ток 4...20 мА, напряжение 0...10 В, термосопротивление);

– диапазон измерения;

– фильтрацию (усреднение).

Шаг 3. Подключение модулей расширения (при необходимости).

ПЛК210 поддерживает подключение модулей ввода/вывода серии МВ через интерфейс RS-485.

В конфигураторе добавить модуль (например, МВА8 – восемь аналоговых входов) и указать его адрес в сети.

Важно: конфигурация модулей расширения требует соответствующей настройки в пользовательской программе.

Шаг 4. Адресация переменных.

Каждому физическому входу/выходу ставится в соответствие переменная в программе (например, %IX0.0 – дискретный вход 0.0).

Для удобства рекомендуется использовать символьные имена (например, DI_Pump_Start вместо %IX0.0).

Пример конфигурации аналогового входа представлен на рисунке 3.2.

```
Модуль: ПЛК210 (встроенный аналоговый вход AI0)
Параметры:
  • Тип сигнала: ток 4–20 мА
  • Диапазон измерения: 0–100 °C
  • Фильтрация: усреднение за 100 мс
  • Переменная в программе: AI_Temperature : REAL
```

Рисунок 3.2 – Конфигурации аналогового входа

Задания для самостоятельного выполнения

1 Изучение аппаратной структуры:

– изучите паспортные данные ОВЕН ПЛК210;

– определите количество и тип встроенных дискретных и аналоговых каналов;

– составьте схему подключения датчика температуры (4…20 мА) к аналоговому входу.

2 Анализ цикла работы:

– загрузите тестовую программу в ПЛК;

– с помощью отладчика измерьте фактическое время цикла;

– проанализируйте влияние увеличения объёма программы на время цикла.

3 Конфигурирование модулей:

– настройте в среде CODESYS два дискретных входа и один аналоговый вход;

– создайте программу, индицирующую состояние входов на встроенных светодиодах;

– протестируйте работу с имитацией сигналов.

Контрольные вопросы

1 Какие основные компоненты входят в аппаратную структуру ОВЕН ПЛК210?

2 В чём отличие Retain-памяти от оперативной памяти?

3 Перечислите этапы цикла сканирования ПЛК и их назначение.

4 Как операционная система влияет на работу ПЛК в реальном времени?

5 Для чего необходима фильтрация аналоговых сигналов при конфигурировании?

6 Какие режимы работы процессора ПЛК существуют и когда они применяются?

7 В чём преимущество моноблочной архитектуры ПЛК210 перед модульной?

4 Лабораторная работа № 4. Разработка и отладка простой программы управления ОВЕН ПЛК210

Цель работы: освоить базовые навыки разработки, тестирования и отладки программ в среде программирования CODESYS для программируемых логических контроллеров на примере ОВЕН ПЛК210.

Теоретические сведения

CODESYS (Controller Development System) – кроссплатформенная среда разработки программ для ПЛК, полностью соответствующая стандарту МЭК 61131-3. Поддерживает пять языков программирования:

1) LD (Ladder Diagram) – релейно-контактные схемы;

2) FBD (Function Block Diagram) – функциональные блок-схемы;

3) SFC (Sequential Function Chart) – последовательные функциональные схемы;

4) ST (Structured Text) – структурированный текст (высокоуровневый текстовый язык);

5) IL (Instruction List) – список инструкций.

ST – текстовый язык программирования, синтаксис которого близок к

языкам Pascal и C.

Основные конструкции:

– присваивание (:=);

– условные операторы (IF ... THEN ... ELSIF ... ELSE ... END_IF);

– циклы (FOR, WHILE, REPEAT);

– операторы выбора (CASE ... OF ... END_CASE);

– логические операции (AND, OR, XOR, NOT);

– арифметические операции (+, –, *, /, MOD).

Пример фрагмента кода:

```
IF StartButton AND NOT EmergencyStop THEN
   Motor := TRUE;
ELSIF StopButton OR EmergencyStop THEN
   Motor := FALSE;
END_IF;
```

Порядок выполнения работы

Этап 1. Подготовка среды и создание проекта.

1 Запустите среду CODESYS;

2 Выберите File → New Project (или нажмите Ctrl + N).

3 В диалоговом окне:

- укажите имя проекта (например, ПЛК210_Светофор);
- выберите тип проекта: Standard project;
- нажмите OK.

4 В мастере выбора устройства:

- в поле поиска введите ОВЕН или OWEN;
- выберите контроллер ПЛК210 или соответствующую модификацию (например, ПЛК210-04-CS);
- если устройство отсутствует в списке, установите таргет-файл производителя через меню Tools → Device Repository.

5 Выберите язык программирования ST (Structured Text) для основной программы.

Этап 2. Конфигурация аппаратных ресурсов.

1 В дереве проекта откройте узел Device → Device Description.

2 На вкладке Resources проверьте наличие задачи (например, MainTask) с периодом цикла 20…50 мс.

3 На вкладке I/O Mapping настройте входы/выходы:

- для симуляции можно использовать виртуальные входы/выходы;
- для реального контроллера укажите соответствие физическим клеммам (например, %IX0.0 – вход 0.0, %QX0.0 – выход 0.0).

Этап 3. Разработка программы на языке ST для задачи управления трёхсекционным светофором.

1 Объявите переменные в разделе VAR POU (Program Organization Unit):

```
PROGRAM PLC_PRG
```

```
VAR
   // Входные переменные (симуляция кнопки запуска)
   StartButton : BOOL;     // Кнопка запуска
   StopButton  : BOOL;     // Кнопка останова

   // Выходные переменные (сигналы светофора)
   RedLight    : BOOL;     // Красный сигнал
   YellowLight : BOOL;     // Жёлтый сигнал
   GreenLight  : BOOL;     // Зелёный сигнал

   // Внутренние переменные
   Timer1      : TON;      // Таймер для зелёного сигнала (20 с)
   Timer2      : TON;      // Таймер для жёлтого сигнала (3 с)
   Timer3      : TON;      // Таймер для красного сигнала (25 с)
   State       : INT;      // Состояние автомата (0-2)
   AutoMode    : BOOL;     // Флаг автоматического режима
END_VAR
```

2 Реализуйте логику управления в теле программы:

```
// Управление запуском/остановом
IF StartButton AND NOT AutoMode THEN
   AutoMode := TRUE;
   State := 0;  // Начинаем с красного сигнала
ELSIF StopButton THEN
   AutoMode := FALSE;
   RedLight := FALSE;
   YellowLight := FALSE;
   GreenLight := FALSE;
END_IF;

// Автомат управления в режиме работы
IF AutoMode THEN
   CASE State OF
      0: // Красный сигнал (25 с)
         RedLight := TRUE;
         YellowLight := FALSE;
         GreenLight := FALSE;
         Timer3(IN := TRUE, PT := T#25S);
         IF Timer3.Q THEN
            Timer3(IN := FALSE);
            State := 1;
         END_IF;

      1: // Зелёный сигнал (20 с)
         RedLight := FALSE;
         GreenLight := TRUE;
         YellowLight := FALSE;
         Timer1(IN := TRUE, PT := T#20S);
         IF Timer1.Q THEN
            Timer1(IN := FALSE);
            State := 2;
         END_IF;

      2: // Жёлтый сигнал (3 с)
         RedLight := FALSE;
         GreenLight := FALSE;
```

```
            YellowLight := TRUE;
            Timer2(IN := TRUE, PT := T#3S);
            IF Timer2.Q THEN
               Timer2(IN := FALSE);
               State := 0;
            END_IF;
      END_CASE;
   END_IF;
```

Этап 4. Отладка и симуляция программы.

1 Компиляция проекта:

– нажмите Build → Build (F7) для проверки синтаксиса;

– устраните все ошибки и предупреждения.

2 Запуск симулятора:

– выберите Tools → Simulation Mode (или нажмите на иконку Simulation на панели инструментов);

– нажмите Login (F5) для загрузки программы в симулятор;

– нажмите Start (F6) для запуска выполнения программы.

3 Мониторинг переменных:

– откройте окно Watch List (View → Watch List);

– добавьте переменные для наблюдения: StartButton, StopButton, RedLight, YellowLight, GreenLight, State;

– в режиме выполнения значения переменных обновляются в реальном времени.

4 Изменение входных сигналов (симуляция):

– в окне Watch List щёлкните правой кнопкой мыши по переменной StartButton;

– выберите Force Value → TRUE для имитации нажатия кнопки;

– аналогично можно форсировать StopButton.

5 Пошаговая отладка:

– установите точку останова (breakpoint): щёлкните слева от номера строки в редакторе кода;

– запустите программу в режиме отладки (Debug → Start/Continue, F5);

– используйте команды:

а) Step Into (F11) – войти в подпрограмму;

б) Step Over (F10) – выполнить текущую строку;

в) Step Out (Shift+F11) – выйти из подпрограммы;

– анализ временных диаграмм:

а) откройте Visualization → Online Chart;

б) добавьте сигналы RedLight, YellowLight, GreenLight;

в) наблюдайте за изменением состояний во времени для проверки корректности временных интервалов.

Задание для самостоятельного выполнения

Разработать программу управления насосной станцией:

– запуск насоса по сигналу датчика уровня «Низкий»;

– остановка по сигналу «Высокий уровень»;
– аварийная остановка при срабатывании датчика перегрева;
– индикация режимов работы (работа/останов/авария) на выходах.

Контрольные вопросы

1 Какие языки программирования стандарта МЭК 61131-3 поддерживаются в CODESYS?
2 В чём преимущества языка ST перед графическими языками (LD, FBD)?
3 Как объявить таймер в программе на языке ST? Приведите пример.
4 Какие режимы отладки доступны в CODESYS? Чем отличается режим симуляции от онлайн-отладки на реальном контроллере?
5 Какие типы данных используются в языке ST для работы с дискретными сигналами?
6 Как организовать циклическое выполнение программы в CODESYS?
7 Какие особенности имеет архитектура ОВЕН ПЛК210 по сравнению с другими контроллерами?
8 Как выполнить принудительное изменение значения переменной в режиме симуляции?

5 Лабораторная работа № 5. Разработка ПЛК-программы с использованием конечного автомата

Цель работы: освоить методику проектирования и реализации ПЛК-программ с использованием модели конечного автомата; приобрести навыки формализации технологических процессов в виде диаграмм состояний и их программной реализации на языке Structured Text (ST) в соответствии со стандартом МЭК 61131-3.

Теоретические сведения

Конечный автомат (КА) – это математическая модель поведения системы с конечным числом состояний, переходами между ними по определённым условиям и действиями, выполняемыми при входе в состояние, выходе из него или во время пребывания. В программировании ПЛК методология конечных автоматов позволяет структурировать сложную логику управления, повысить надёжность, читаемость и упростить сопровождение программ.

Конечный автомат описывает поведение системы как совокупность:
– конечного множества состояний – дискретных режимов работы (например, «Ожидание», «Запуск», «Работа», «Авария»);
– начального состояния – состояние, в котором автомат находится после включения или сброса;
– множества входных событий (условий перехода) – сигналов или логических условий, вызывающих переход между состояниями;

– функции переходов – правила, определяющие, в какое состояние перейдёт автомат при наступлении определённого события из текущего состояния;

– действий (выходных функций) – операций, выполняемых при входе в состояние, выходе из него или в процессе нахождения в нём.

Преимущества использования конечных автоматов в ПЛК-программировании:

– структурирование сложной логики: процесс разбивается на понятные, изолированные этапы;

– повышение читаемости: диаграмма состояний наглядно отражает поведение системы;

– упрощение отладки: легко отследить текущее состояние и причину перехода;

– удобство модификации: добавление нового состояния или перехода не нарушает существующую логику;

– поддержка стандартов: методология соответствует подходам, описанным в стандарте МЭК 61131-3.

Этапы разработки ПЛК-программы на основе конечного автомата.

Шаг 1. Анализ задачи и выделение состояний.

Разбейте технологический процесс на логические этапы.

Пример для системы розлива жидкости:

ОЖИДАНИЕ → НАПОЛНЕНИЕ → НАГРЕВ → СЛИВ → ОЧИСТКА.

Шаг 2. Определение условий переходов.

Для каждой пары состояний укажите, при каком условии происходит переход.

Пример:

ОЖИДАНИЕ → НАПОЛНЕНИЕ: кнопка «Пуск» нажата и крышка закрыта.

НАПОЛНЕНИЕ → НАГРЕВ: датчик уровня = полный.

Шаг 3. Формализация на диаграмме состояний.

Постройте диаграмму (рисунок 5.1), где:

– кружки – состояния;

– стрелки – переходы с указанием условий;

– над/под стрелками – действия при переходе (опционально).

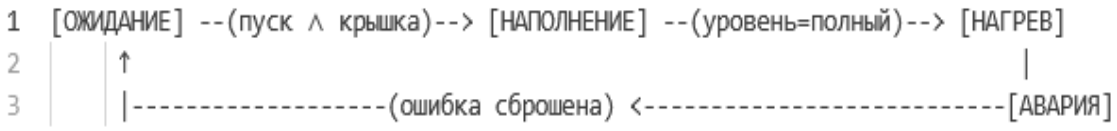


Рисунок 5.1 – Диаграмма конечного автомата для системы розлива жидкости

Шаг 4. Реализация в коде (язык Structured Text, ST).

Используйте конструкцию CASE...OF для обработки состояний.

Рекомендуется:

– хранить текущее состояние в переменной перечислимого типа (ENUM);

– выполнять проверку условий переходов после выполнения действий

состояния;

– использовать таймеры и флаги для временных условий.

Пример ПЛК-программы на языке ST (IEC 61131-3):

```
// 1. Объявление типов и переменных
TYPE
   T_State : (IDLE, FILLING, HEATING, DRAINING, ERROR);
END_TYPE

VAR
   currentState : T_State := IDLE;  // Текущее состояние автомата
   nextState    : T_State;          // Следующее состояние

   // Входные сигналы
   btnStart     : BOOL;             // Кнопка "Пуск"
   sensorFull   : BOOL;             // Датчик уровня (бак полон)
   tempReached  : BOOL;             // Достигнута заданная температура
   btnReset     : BOOL;             // Кнопка "Сброс" после аварии
   emergency    : BOOL;             // Аварийный сигнал (датчик перегрева и т.п.)

   // Выходные сигналы
   valveIn      : BOOL := FALSE;    // Клапан наполнения
   heater       : BOOL := FALSE;    // Нагреватель
   valveOut     : BOOL := FALSE;    // Клапан слива
   alarmLamp    : BOOL := FALSE;    // Сигнальная лампа аварии

   // Таймер нагрева (10 секунд)
   heatTimer    : TON;
END_VAR

// 2. Основной цикл управления (в программе или функциональном блоке)
// Шаг 2.1: Определение следующего состояния на основе текущего и условий
CASE currentState OF

   IDLE:
      // Действия при входе (выполняются один раз при переходе в состояние)
      // В данном случае — сброс выходов
      valveIn := FALSE;
      heater := FALSE;
      valveOut := FALSE;

      // Условия перехода
      IF btnStart AND NOT emergency THEN
         nextState := FILLING;
      ELSE
         nextState := IDLE;
      END_IF;

   FILLING:
      // Действие в состоянии: открыть клапан наполнения
      valveIn := TRUE;

      // Условие перехода
      IF sensorFull THEN
         valveIn := FALSE;  // Действие при выходе
```

```
            nextState := HEATING;
        ELSIF emergency THEN
            valveIn := FALSE;
            nextState := ERROR;
        ELSE
            nextState := FILLING;
        END_IF;

    HEATING:
        // Действие в состоянии: включить нагрев и таймер
        heater := TRUE;
        heatTimer(IN := TRUE, PT := T#10s);

        // Условия перехода
        IF heatTimer.Q OR tempReached THEN
            heater := FALSE;
            heatTimer(IN := FALSE);  // Сброс таймера
            nextState := DRAINING;
        ELSIF emergency THEN
            heater := FALSE;
            heatTimer(IN := FALSE);
            nextState := ERROR;
        ELSE
            nextState := HEATING;
        END_IF;

    DRAINING:
        valveOut := TRUE;

        IF NOT sensorFull THEN  // Бак опустошён
            valveOut := FALSE;
            nextState := IDLE;
        ELSIF emergency THEN
            valveOut := FALSE;
            nextState := ERROR;
        ELSE
            nextState := DRAINING;
        END_IF;

    ERROR:
        alarmLamp := TRUE;
        valveIn := FALSE;
        heater := FALSE;
        valveOut := FALSE;

        // Ожидание ручного сброса
        IF btnReset AND NOT emergency THEN
            alarmLamp := FALSE;
            nextState := IDLE;
        ELSE
            nextState := ERROR;
        END_IF;

    ELSE
        nextState := IDLE;  // Защита от неопределённого состояния
END_CASE;
```

```
// Шаг 2.2: Обновление текущего состояния
currentState := nextState;
```

Задание для самостоятельного выполнения

Разработать ПЛК-программу управления дверью лифта, которая открывается при остановке кабины на этаже и закрывается через заданное время или при нажатии кнопки «Закрыть».

Входные сигналы:

– doorAtFloor – датчик положения (кабина на этаже);

– btnOpen – кнопка «Открыть дверь» (в кабине);

– btnClose – кнопка «Закрыть дверь»;

– obstacleDetected – датчик препятствия в проёме двери;

– doorFullyOpen – датчик «Дверь открыта полностью»;

– doorFullyClosed – датчик «Дверь закрыта полностью».

Выходные сигналы:

– motorOpen – двигатель открытия двери;

– motorClose – двигатель закрытия двери;

– buzzer – звуковой сигнал при препятствии.

Требования к функционалу.

1 Состояния автомата: ЗАКРЫТА, ОТКРЫТИЕ, ОТКРЫТА, ЗАКРЫТИЕ, БЛОКИРОВКА (при препятствии).

2 При остановке на этаже (doorAtFloor = TRUE) дверь должна открыться.

3 В состоянии ОТКРЫТА запускается таймер на 5 с. По истечении времени начинается закрытие.

4 При обнаружении препятствия (obstacleDetected = TRUE) во время закрытия – немедленный переход в ОТКРЫТИЕ, включение buzzer на 1 с.

5 Кнопка btnClose ускоряет закрытие (переход в ЗАКРЫТИЕ, даже если таймер не истёк).

6 Кнопка btnOpen повторно открывает дверь из состояния ОТКРЫТА (сброс таймера).

6 Лабораторная работа № 6. Реализация конечного автомата на языках ST и LD для ОВЕН ПЛК210

Цель работы: освоить методику проектирования, программной реализации и отладки конечных автоматов на языках стандарта МЭК 61131-3 (Structured Text и Ladder Diagram) для решения практических задач промышленной автоматизации на базе контроллера ОВЕН ПЛК210.

Теоретические сведения

Конечный автомат – это математическая модель устройства с конечным числом состояний, переходящего из одного состояния в другое под воздействием входных сигналов.

Существует два основных типа конечных автоматов:

1) автомат Мура: выходные сигналы зависят только от текущего состояния;

2) автомат Мили: выходные сигналы зависят от текущего состояния и входных сигналов одновременно.

Для промышленных ПЛК предпочтительнее использовать автомат Мура, т. к. он обеспечивает большую предсказуемость и устойчивость к кратковременным помехам на входах.

Формально автомат описывается пятью параметрами: $A = (S, X, Y, \delta, \lambda)$, где:

– S – множество состояний $\{s_0, s_1, ..., s_n\}$;

– X – множество входных сигналов;

– Y – множество выходных сигналов;

– δ: $S \times X \rightarrow S$ – функция переходов;

– λ: $S \rightarrow Y$ (Мура) или λ: $S \times X \rightarrow Y$ (Мили) – функция выходов.

Структура программы, реализующей конечный автомат на языке ST:

```
// 1. Объявление типов и переменных
TYPE
   T_State : (IDLE, START, WORKING, STOPPING, ERROR);
END_TYPE

VAR
   currentState : T_State := IDLE;  // Текущее состояние
   nextState    : T_State;          // Следующее состояние
   // Входные сигналы
   startBtn     : BOOL;
   stopBtn      : BOOL;
   emergency    : BOOL;
   sensorOk     : BOOL;
   // Выходные сигналы
   motorOn      : BOOL;
   valveOpen    : BOOL;
   alarmLight   : BOOL;
END_VAR
```

Пример – Автомат управления насосной станцией (автомат Мура):

```
PROGRAM PumpControl_ST
VAR
   // ... объявление переменных (см. п. 2.1)
END_VAR

// ============ ЭТАП 1: ОПРЕДЕЛЕНИЕ СЛЕДУЮЩЕГО СОСТОЯНИЯ ============
CASE currentState OF
   IDLE:
      IF startBtn AND NOT emergency THEN
         nextState := START;
      ELSE
         nextState := IDLE;
      END_IF;

   START:
      IF sensorOk THEN
         nextState := WORKING;
      ELSIF stopBtn OR emergency THEN
         nextState := STOPPING;
      ELSE
```

```
        nextState := START;
      END_IF;

    WORKING:
      IF stopBtn OR NOT sensorOk OR emergency THEN
        nextState := STOPPING;
      ELSE
        nextState := WORKING;
      END_IF;

    STOPPING:
      // Автоматический переход после завершения останова
      IF timeToStop = 0 THEN  // условие завершения таймера
        nextState := IDLE;
      ELSE
        nextState := STOPPING;
      END_IF;

    ERROR:
      IF NOT emergency THEN
        nextState := IDLE;
      ELSE
        nextState := ERROR;
      END_IF;
END_CASE

// ============ ЭТАП 2: ФОРМИРОВАНИЕ ВЫХОДНЫХ СИГНАЛОВ (функция λ) =======
CASE currentState OF
    IDLE:
        motorOn := FALSE;
        valveOpen := FALSE;
        alarmLight := FALSE;

    START:
        motorOn := TRUE;
        valveOpen := FALSE;
        alarmLight := FALSE;

    WORKING:
        motorOn := TRUE;
        valveOpen := TRUE;
        alarmLight := FALSE;

    STOPPING:
        motorOn := FALSE;
        valveOpen := FALSE;
        alarmLight := FALSE;

    ERROR:
        motorOn := FALSE;
        valveOpen := FALSE;
        alarmLight := TRUE;
END_CASE

// ============ ЭТАП 3: ПЕРЕХОД В НОВОЕ СОСТОЯНИЕ ============
IF systemCycle THEN  // Выполнение один раз за цикл ПЛК
    currentState := nextState;
END_IF;
```

Основные принципы реализации.

1 Четкое разделение на три этапа: расчёт следующего состояния → формирование выходов → переход.

2 Все переходы выполняются синхронно по фронту цикла ПЛК.

3 Для таймерных операций использовать встроенные функциональные блоки TON, TOF.

С целью оптимизация кода необходимо:

1) использовать перечисления (ENUM) вместо целочисленных констант для состояний;

2) вынести логику переходов в отдельную функцию GetNextState();

3) применять защиту от неопределённых состояний:

```
IF NOT (currentState IN [IDLE, START, WORKING, STOPPING, ERROR]) THEN
   currentState := IDLE;  // Защита от "зависания" в несуществующем состоянии
END_IF;
```

Особенности графической реализации конечного автомата на языке LD. В языке LD автомат реализуется через комбинацию:

– реле памяти (вспомогательные переменные) для хранения состояний;

– логических цепей для формирования условий перехода;

– самоблокировок для удержания состояния.

Пример – Упрощённый автомат «Пуск-Стоп» трёх состояний (рисунок 6.1).

```
1  Сеть 1: Состояние IDLE (ожидание)
2  |   |----[ ]----[ ]----( )----|
3  |   |  !M1    !M2    IDLE   |
4  |   |                       |
5  |   |----[ ]------------------|
6  |   |   IDLE                |
7
8  Сеть 2: Переход IDLE → WORKING по кнопке "Пуск"
9  |   |----[ ]----[ ]----[ ]----( )----|
10 |   |  ПУСК   !СТОП   IDLE   M1    |  // M1 = состояние WORKING
11 |   |                              |
12
13 Сеть 3: Самоблокировка состояния WORKING
14 |   |----[ ]------------------( )----|
15 |   |   M1                   M1     |
16 |   |                               |
17 |   |----[ ]----[ ]------------------|
18 |   |  !ПУСК   СТОП                 |
19
20 Сеть 4: Выходные сигналы для состояния WORKING
21 |   |----[ ]----( )----|
22 |   |   M1    Мотор   |
23 |   |                 |
24 |   |----[ ]----( )----|
25 |   |   M1   Клапан   |
```

Рисунок 6.1 – Реализация конечного автомата на языке LD

Пояснения к примеру (см. рисунок 6.1).

1 M1, M2 – вспомогательные реле (память состояний).

2 Каждое состояние представлено отдельным реле.

3 Взаимная блокировка состояний обеспечивается через инверсные контакты (!M1, !M2).

4 Выходные сигналы формируются непосредственно от реле состояния (автомат Мура).

Рекомендации по проектированию конечного автомата на LD.

1 Ограничение сложности: для автоматов с > 5 состояний LD становится трудночитаемым – рекомендуется использовать ST или SFC.

2 Стандартизация обозначений:

– Mxx – память состояний (например, M10 = состояние «РАБОТА»);

– Txx – таймеры;

– Cxx – счётчики.

3 Комментирование: каждая сеть должна содержать текстовый комментарий с описанием логики.

Отладка и тестирование программ на ОВЕН ПЛК210

Настройка среды программирования.

1 Создайте новый проект:

– Device Selection → OWEN → PLC210;

– выберите соответствующую модификацию (например, ПЛК210-14-CS).

2 Настройте интерфейс связи:

– для отладки через Ethernet: Project → Communication Settings → Ethernet/IP;

– для программирования через USB: установите драйверы ОВЕН, выберите COM-порт.

Методы отладки.

1 Онлайн-мониторинг:

– в режиме Online отслеживайте значения переменных состояния в таблице наблюдения (Watch List).

– используйте цветовую индикацию в редакторе LD/ST:

а) зелёный – активный контакт/оператор;

б) серый – неактивный;

в) красный – ошибка выполнения.

2 Пошаговая отладка (для ST):

```
// Добавьте отладочные точки
IF debugMode THEN
   // Точка останова №1
   // Проверка: корректность входных сигналов
   debug_state := currentState;
   debug_inputs := WORD#16#0000;
   IF startBtn THEN debug_inputs := debug_inputs OR 16#0001; END_IF;
   IF stopBtn  THEN debug_inputs := debug_inputs OR 16#0002; END_IF;
END_IF;
```

3 Логирование переходов:

```
// Запись истории состояний в циклический буфер
VAR
   stateLog: ARRAY[0..99] OF T_State;
   logIndex: INT := 0;
END_VAR
```

```
IF currentState <> nextState THEN
   stateLog[logIndex] := nextState;
   logIndex := (logIndex + 1) MOD 100;
   // Фиксация времени перехода
   transitionTime := GetSystemTime();
END_IF;
```

Тестирование алгоритма.

Этап 1. Модульное тестирование в симуляторе CODESYS.

1 Активируйте встроенный симулятор: Tools → Simulation Mode.

2 Создайте тестовые сценарии:

- нормальный цикл: ПУСК → РАБОТА → СТОП;
- аварийная остановка в каждом состоянии;
- кратковременные помехи на входах (дребезг кнопок).

3 Проверьте корректность выходных сигналов для каждого состояния.

Этап 2. Тестирование на реальном ПЛК.

1 Загрузите программу в ПЛК210 через меню Online → Login.

2 Подключите тестовые нагрузки к выходам (светодиоды/реле).

3 Выполните проверку по контрольному листу (таблица 6.1).

Таблица 6.1 – Контрольный лист

Сценарий	Ожидаемое поведение	Результат
Нажатие «Пуск» из состояния IDLE	Переход в START → WORKING, включение мотора	
Аварийная кнопка в состоянии WORKING	Мгновенный переход в ERROR, включение аварийной сигнализации	
Повторное нажатие «Пуск» в состоянии WORKING	Состояние не меняется (защита от дребезга)	

Этап 3. Стресс-тестирование.

1 Имитируйте частые переключения входов (частота > 10 Гц).

2 Проверьте реакцию на одновременное появление конфликтующих сигналов (ПУСК + СТОП).

3 Убедитесь в отсутствии «гонок» состояний при переходах.

Задание для самостоятельного выполнения

Автомат управления конвейерной линией с аварийной защитой.

Постановка задачи. Реализовать систему управления трёхсекционным конвейером с датчиками присутствия объекта и аварийной остановкой. Автомат должен предотвращать столкновение объектов и обеспечивать безопасную остановку при аварии.

Требования к функционалу.

1 Входные сигналы:

- btnStart, btnStop – кнопки управления;
- sensor1, sensor2, sensor3 – датчики присутствия объекта на секциях конвейера;

– emergencyStop – аварийная кнопка (размыкающий контакт).

2 Выходные сигналы:

– motor1, motor2, motor3 – управление двигателями секций;

– alarm – аварийная сигнализация.

3 Состояния автомата:

– IDLE – ожидание запуска;

– RUNNING – нормальная работа;

– PAUSED – пауза (объект в зоне датчика 2);

– JAM – затор (объекты одновременно в зонах датчиков 2 и 3);

– EMERGENCY – аварийная остановка.

4 Логика переходов:

– из IDLE → RUNNING: при нажатии btnStart и отсутствии объектов на конвейере;

– из RUNNING → PAUSED: при срабатывании sensor2 при активном sensor1 (риск столкновения);

– из RUNNING → JAM: при одновременном срабатывании sensor2 и sensor3;

– любой переход → EMERGENCY: при активации emergencyStop;

– из EMERGENCY → IDLE: только после снятия аварии и ручного сброса.

Требования к реализации.

1 Язык программирования: ST (Structured Text).

2 Использовать перечислимый тип для состояний.

3 Реализовать функцию GetNextState() для расчёта следующего состояния.

4 Для таймера паузы использовать TON с временем 5 с (автоматический переход в RUNNING после освобождения зоны).

5 Все выходы должны формироваться только на основе текущего состояния (автомат Мура).

Требования к тестированию.

1 Сценарий нормальной работы: запуск → перемещение объекта по конвейеру → остановка.

2 Сценарий затора: имитация одновременного присутствия объектов у датчиков 2 и 3 → проверка перехода в состояние JAM и остановки всех двигателей.

3 Сценарий аварии: активация emergencyStop в любом состоянии → мгновенная остановка всех двигателей и включение сигнализации.

4 Проверка невозможности запуска при наличии объекта на конвейере (защита от повторного запуска).

Контрольные вопросы

1 Почему в состоянии JAM нельзя автоматически возобновлять работу после устранения затора?

2 Как модифицировать автомат для поддержки реверса двигателей при заторе?

3 Какие преимущества даёт разделение логики переходов и формирования выходов в отдельные блоки кода?

7 Лабораторная работа № 7. Подключение и настройка датчиков в IIoT-системе

7.1 Типы датчиков и их подключение к ОВЕН ПЛК210

7.1.1 Датчики температуры.

1 Термосопротивления (Pt100, Pt1000) – подключение к специализированным модулям.

2 Термопары (K, J, T) – через преобразователи.

3 Аналоговые выходы: 4…20 мА, 0…10 В.

Подключение.

1 Для сигналов 4…20 мА используйте аналоговые входы AI с двухпроводной схемой (питание датчика от внешнего источника 24 В).

2 Для сигналов 0…10 В – трёхпроводная схема: «+», «–», «сигнал».

3 Аналоговые входы ПЛК210 имеют 16-битный АЦП, что обеспечивает высокую точность измерений

7.1.2 Датчики давления.

Типовые сигналы: 4…20 мА (наиболее распространённый), 0…10 В, 0…5 В.

Подключение представлено на рисунке 7.1.

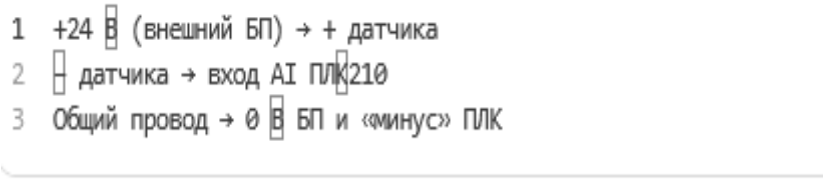


Рисунок 7.1 – Подключение датчиков давления

Важно: для активных датчиков с выходом 4…20 мА питание должно подаваться от внешнего блока питания, а не от ПЛК.

7.1.3 Датчики уровня.

Типы:

– гидростатические (погружные) – 4…20 мА;

– радарные – 4…20 мА или цифровой выход;

– емкостные – 4…20 мА.

Подключение: аналогично датчикам давления.

Для измерения уровня жидкости в резервуаре высотой 0…5 м с выходом 4…20 мА:

1) 4 мА = 0 м (дно резервуара);

2) 20 мА = 5 м (верхний уровень).

7.1.4 Датчики движения.

Типы:

– PNP (источник тока) – при срабатывании подаёт +24 В на вход ПЛК;

– NPN (потребитель тока) – при срабатывании замыкает вход на 0 В;

– сухой контакт (реле) – замыкание/размыкание цепи.

Подключение к дискретным входам DI.

1 Для датчиков PNP: выход датчика → вход DI, общий «–» ПЛК → «–» датчика.

2 Для сухих контактов: один вывод контакта → +24 В, второй → вход DI.

7.2 Считывание и обработка сигналов в среде CODESYS V2.3

Настройка аналоговых входов.

1 Откройте проект в CODESYS V2.3.

2 В дереве объектов перейдите: Device → PLC Configuration → Analog Inputs.

Для каждого входа задайте:

– тип сигнала (4…20 мА, 0…10 В и др.);

– диапазон измерения в инженерных единицах (например, 0 °C…100 °C);

– фильтрацию сигнала (при необходимости).

Пример программной обработки сигналов датчиков (язык ST):

```
// Объявление переменных
VAR
    aiTempRaw: INT;       // Сырое значение с АЦП
    fTemp: REAL;          // Температура в °C
    aiPressRaw: INT;      // Сырое значение давления
    fPressure: REAL;      // Давление в бар
    diMotion: BOOL;       // Сигнал с датчика движения
    bAlarmHigh: BOOL;     // Авария по верхнему пределу
    bAlarmLow: BOOL;      // Авария по нижнему пределу
END_VAR

// Считывание аналоговых значений
aiTempRaw := %IW100;      // Адрес аналогового входа 1
aiPressRaw := %IW102;     // Адрес аналогового входа 2

// Масштабирование 4–20 мА → 0–100 °C
// Формула: Y = (X - Xmin) * (Ymax - Ymin) / (Xmax - Xmin) + Ymin
fTemp := (REAL(aiTempRaw) - 6554.0) * (100.0 - 0.0) / (32767.0 - 6554.0) + 0.0;

// Масштабирование 4–20 мА → 0–10 бар
fPressure := (REAL(aiPressRaw) - 6554.0) * (10.0 - 0.0) / (32767.0 - 6554.0) + 0.0;

// Считывание дискретного входа (движение)
diMotion := %IX100.0;

// Логика аварийной сигнализации
bAlarmHigh := fTemp > 90.0;   // Верхний предел 90 °C
bAlarmLow := fTemp < 5.0;     // Нижний предел 5 °C

// Управление выходом (реле/сигнализация)
%QX100.0 := bAlarmHigh OR bAlarmLow;
```

7.3 Интеграция с IIoT-платформой

Для передачи данных в облако:

– настройте Modbus TCP-сервер в ПЛК210 (порт 502);

– создайте переменные для экспорта (температура, давление, статус движения);

– подключите шлюз IIoT (например, на базе Raspberry Pi) или используйте встроенные возможности ПЛК для отправки данных по протоколу MQTT/HTTP через один из четырёх Ethernet-портов.

Задания для самостоятельного выполнения

Задание 1. Мониторинг температурного режима склада

Цель: настроить систему контроля температуры в помещении с аварийной сигнализацией.

Исходные данные.

1 Датчик температуры с выходом 4...20 мА, диапазон измерения –20 °С…+50 °С.

2 ОВЕН ПЛК210 с аналоговым входом AI1.

3 Световая сигнализация (светодиодная лампа 24 В) на дискретном выходе Q0.0.

Задачи.

1 Составить схему электрического подключения датчика к ПЛК.

2 Написать программу на ST для:

– масштабирования сигнала 4…20 мА → –20 °С…+50 °С;

– формирования аварийного сигнала при температуре < 0 °С или > 30 °С;

– индикации аварии через дискретный выход.

3 Реализовать программную фильтрацию «дрожания» показаний (скользящее среднее за пять циклов).

Задание 2. Построение мини-IIoT-узла для передачи данных в облако.

Цель: интегрировать ПЛК210 в облачную платформу для визуализации данных.

Исходные данные.

1 ОВЕН ПЛК210 с подключёнными датчиками:

– температура (AI1, 4…20 мА, 0 °С…100 °С);

– давление (AI2, 4…20 мА, 0…10 бар);

– движение (I0.0, PNP).

2 Локальная сеть с доступом в интернет.

3 Бесплатная облачная платформа (например, ThingsBoard, Ubidots или аналог).

Задачи.

1 Настроить в ПЛК210:

– Modbus TCP-сервер с отображением переменных температуры, давления и статуса движения;

– период опроса данных – 5 с.

2 Настроить шлюз (ПК или одноплатный компьютер):

– установить ПО для опроса Modbus-регистров ПЛК;

– настроить отправку данных в облако по протоколу MQTT/HTTP.

3 Создать на облачной платформе:

– панель мониторинга с графиками температуры и давления;

– индикатор статуса движения («Движение обнаружено» / «Нет движения»);

– правило оповещения по email/SMS при превышении температуры 80 °C.

8 Лабораторная работа № 8. Управление исполнительными устройствами с использованием ОВЕН ПЛК210

8.1 Управление электродвигателями

8.1.1 Прямое подключение Пуск-Останов.

Для управления асинхронными двигателями малой мощности (до 0,75 кВт) возможно прямое подключение через релейные выходы ПЛК210:

1) выход %QX0.0 – пуск «Вперёд»;
2) выход %QX0.1 – пуск «Назад»;
3) выход %QX0.2 – стоп/торможение.

Важно: для двигателей мощностью свыше 0,75 кВт необходимо использовать промежуточные устройства:

– магнитные пускатели (через релейные выходы ПЛК);

– частотные преобразователи (через аналоговые выходы 0…10 В или интерфейс RS-485 по протоколу Modbus RTU).

8.1.2 Защита двигателей.

В программе необходимо реализовывать:

– блокировку одновременного включения «Вперёд»/«Назад» (механическая и программная);

– контроль перегрузки через тепловое реле (вход %IX0.0);

– таймер пуска для предотвращения частых включений.

8.2 Управление электромагнитными реле

Подключение:

– релейные выходы ПЛК210 рассчитаны на коммутацию нагрузки до 2 А при 250 В;

– для мощных нагрузок (св. 2 А) использовать промежуточные реле;

– обязательна установка гасящих диодов (для катушек постоянного тока) или варисторов (для катушек переменного тока) параллельно катушке реле для подавления ЭДС самоиндукции.

Пример программы (язык LD – Ladder Diagram):

```
| %IX0.0    %M0       %QX0.0 |
|----| |------|/|------------( )----|  // Пуск реле при нажатии кнопки
|                                 |
| %IX0.1    %QX0.0              |
|----| |-------------------------|  // Самоподхват
|
| %IX0.2    %QX0.0              |
|----| |------|/|----------------|  // Отключение реле
```

8.3 Управление электрогидравлическими клапанами

Типы клапанов и их подключение к ПЛК представлены в таблице 8.1.

Таблица 8.1 – Подключение клапанов к ПЛК

Тип клапана	Сигнал управления	Подключение к ПЛК210
Двухпозиционный (2/2)	Дискретный 24 В	Выход %QX через промежуточное реле
Трёхпозиционный (3/2)	Два дискретных сигнала	Выходы %QX0.0, %QX0.1
Пропорциональный	Аналоговый 0…10 В, 4…20 мА	Аналоговый выход %QW0

8.3.1 Особенности программной реализации.

Для пропорциональных клапанов использовать функциональный блок RAMP для плавного изменения сигнала (предотвращение гидроударов).

Реализовать обратную связь через датчики давления/положения поршня.

Ввести тайм-ауты контроля исполнения команды (например, если за 5 с поршень не достиг конечного положения – аварийная остановка).

8.4 Взаимодействие исполнительных устройств с датчиками

Подключение типовых датчиков к ПЛК представлено в таблице 8.2.

Пример логики безопасности

```
// Проверка готовности системы перед запуском
IF %IX0.0 AND NOT %IX0.1 AND (%IW0 < 80) THEN  // Кнопка «Пуск» + нет аварии + температура < 80°C
    %QX0.0 := TRUE;  // Включение двигателя
ELSIF %IX0.2 OR (%IW0 >= 90) THEN  // Кнопка «Стоп» ИЛИ перегрев
    %QX0.0 := FALSE;
END_IF;
```

Таблица 8.1 – Подключение датчиков к ПЛК

Датчик	Тип сигнала	Применение	Вход ПЛК210
Концевой выключатель	Дискретный	Контроль конечных положений	%IX
Датчик давления	Аналоговый 4…20 мА	Контроль гидросистемы	%IW
Датчик температуры	Аналоговый 0…10 В	Защита двигателя	%IW
Энкодер	Импульсный	Позиционирование	Выделенные входы высокой скорости

8.5 Реализация автоматического и ручного режимов управления

8.5.1 Архитектура программы.

Рекомендуется структурировать программу по режимам работы:

```
PROGRAM Main
VAR
    Mode_Auto: BOOL := FALSE;  // Переключатель режима (вход %IX1.0)
    Start_Auto: BOOL;          // Кнопка «Старт авто» (%IX1.1)
    Step: INT := 0;            // Шаг автомата
END_VAR

IF NOT Mode_Auto THEN
    // ============ РУЧНОЙ РЕЖИМ ============
    %QX0.0 := %IX2.0;  // Прямой контроль выхода кнопкой
    %QX0.1 := %IX2.1;
ELSIF Start_Auto AND (Step = 0) THEN
    // ============ АВТОМАТИЧЕСКИЙ РЕЖИМ ============
    CASE Step OF
        0: // Инициализация
            %QX0.0 := TRUE;
            IF %IX3.0 THEN Step := 1; END_IF;  // Датчик «Достигнуто положение 1»
        1:
            %QX0.1 := TRUE;
            IF %IX3.1 THEN Step := 2; END_IF;  // Датчик «Достигнуто положение 2»
        2:
            %QX0.0 := FALSE;
            %QX0.1 := FALSE;
            Step := 0;
    END_CASE;
END_IF;
```

8.5.2 Требования к переключению режимов.

1 Переключение «Ручной ↔ Автомат» должно происходить только в безопасном состоянии (все исполнительные устройства отключены).

2 При переходе в ручной режим автоматический цикл должен прерываться без повреждения оборудования.

3 Визуальная индикация текущего режима (светодиоды на панели или на экране оператора).

Задание для самостоятельного выполнения

Управление конвейерной линией.

Цель: реализовать управление трёхсекционным конвейером с защитой от перегрузки.

Исходные данные:

– три асинхронных двигателя (каждый через магнитный пускатель);

– датчики контроля движения ленты (3 шт., дискретные);

– датчик перегрузки (аналоговый 4…20 мА);

– кнопки: «Пуск», «Стоп», «Сброс аварии»;

– переключатель режимов: «Ручной»/«Автомат».

Требования к программе.

1 В ручном режиме – независимое управление каждой секцией кнопками «Вперёд»/«Назад».

2 В автоматическом режиме – запуск секций по схеме «третья → вторая → первая» с задержкой 2 с между включениями.

3 При отсутствии сигнала с датчика движения в течение 5 с – аварийная остановка соответствующей секции.

4 При превышении тока (сигнал >18 мА) – останов всех секций с индикацией «Перегрузка».

5 После устранения причины аварии – требуется нажатие кнопки «Сброс».

Отчет. Схема подключения, листинг программы на LD/FBD, временная диаграмма работы.

9 Лабораторная работа № 9. Организация обмена данными по протоколу Modbus

9.1 Теоретические сведения

Протокол Modbus – один из самых распространённых промышленных протоколов обмена данными, разработанный в 1979 г. компанией Modicon. Он реализуется в двух основных вариантах:

1) Modbus RTU – передача по последовательному интерфейсу (RS-485/RS-232) с бинарной упаковкой данных и CRC-контролем;

2) Modbus TCP – передача по сети Ethernet с инкапсуляцией PDU в TCP-пакет (порт 502).

ОВЕН ПЛК210 поддерживает оба варианта и может выступать как в роли мастера (инициатора запросов), так и слейва (устройства-ответчика).

9.1.1 Настройка связи по протоколу Modbus.

Modbus RTU (на базе интерфейса RS-485).

Физическое подключение:

– используется интерфейс COM1 (RS-485) ОВЕН ПЛК210;

– топология «шина»: все устройства подключаются параллельно по двухпроводной линии A(+) / B(–);

– обязательна терминация на концах линии (120 Ом);
– максимальная длина линии – до 1200 м (при скорости 9600 бод).
Параметры связи должны соответствовать данным, приведенным в таблице 9.1.
Настройка в среде CODSYS:
1 Откройте проект для ПЛК210.
2 В дереве проекта: Device → Communication Settings → Serial Interface.
3 Укажите параметры порта (скорость, чётность и т. д.).
4 Добавьте библиотеку ModbusRTU_Master или ModbusRTU_Slave.
5 Настройте таблицу обмена: адреса регистров, типы данных, период опроса.

Таблица 9.1 – Параметры последовательного интерфейса COM1

Параметр	Рекомендуемое значение
Скорость, бит/с	9600 / 19200 / 38400
Бит данных	8
Стоп-биты	1

Окончание таблицы 9.1

Параметр	Рекомендуемое значение
Чётность	Нет / Чётная / Нечётная
Адрес устройства	1–247 (уникальный для каждого слейва)

Modbus TCP.
Физическое подключение:
– используется встроенный Ethernet-порт ПЛК210;
– устройства объединяются в локальную сеть через коммутатор.
Настройка IP-адресации:
– назначьте ПЛК статический IP-адрес (например, 192.168.1.10);
– слейвы должны находиться в той же подсети.
Настройка в ОВЕН ПЛК.
1 Device → Communication Settings → Ethernet.
2 Укажите IP-адрес, маску подсети, шлюз.
3 Добавьте библиотеку ModbusTCP_Master или ModbusTCP_Slave.
4 Для мастера укажите IP-адреса и порты (502) слейв-устройств.
5 Настройте таблицу обмена аналогично RTU.

9.1.2 Обмен данными между ОВЕН ПЛК210 и устройствами.
Архитектура обмена.
ПЛК210 чаще выступает мастером, опрашивая слейвы (датчики, счётчики, преобразователи частоты). Примеры устройств-слейвов:
– датчики температуры ОВЕН ТРМ (через модуль связи);
– счётчики импульсов ОВЕН СИ;
– преобразователи частоты VACON, INNOVERT;
– модули ввода-вывода ОВЕН МВА.
Функциональные коды Modbus (основные) приведены в таблице 9.2.

Таблица 9.2 – Функциональные коды Modbus

Код	Назначение	Область памяти
01 (0x01)	Чтение дискретных выходов	Coil Status
02 (0x02)	Чтение дискретных входов	Input Status
03 (0x03)	Чтение регистров хранения	Holding Registers
04 (0x04)	Чтение входных регистров	Input Registers
05 (0x05)	Запись одного выхода	Coil Status
06 (0x06)	Запись одного регистра	Holding Registers
15 (0x0F)	Запись нескольких выходов	Coil Status
16 (0x10)	Запись нескольких регистров	Holding Registers

Пример программной реализации (язык ST):

```
// Инициализация мастера Modbus RTU
mb_master(
    ENABLE := TRUE,
    MODE := 1,          // 1 = RTU
    BAUD := 9600,
    PARITY := 0,        // 0 = нет чётности
    STOP := 1,
    DEVICE := 1,        // COM1
    RESPONSE_TIMEOUT := 100,
    RETRIES := 3
);

// Чтение 2 регистров из устройства с адресом 5, начиная с адреса 40001
mb_read_holding_registers(
    ENABLE := trigger,
    SLAVE := 5,
    START_ADDRESS := 0,   // 40001 = адрес 0 в нумерации 0-based
    QUANTITY := 2,
    DATA_PTR := ADR(data_array)
);

// Запись значения 1234 в регистр 40001 устройства 5
mb_write_single_register(
    ENABLE := trigger_write,
    SLAVE := 5,
    REGISTER_ADDRESS := 0,
    VALUE := 1234
);
```

Задание для самостоятельного выполнения

Настройка Modbus RTU и опрос датчика температуры.

Цель: освоить настройку последовательного интерфейса и чтение данных со слейв-устройства.

Исходные данные:

– ОВЕН ПЛК210 (мастер);

– модуль связи ОВЕН МКС с подключённым датчиком температуры (слейв, адрес 3);

– скорость 9600 бод, 8N1.

Задачи.

1 Настроить COM1 ПЛК210 на работу в режиме Modbus RTU.

2 Реализовать циклический опрос регистра 40001 (температура, тип INT) с периодом 500 мс.

3 Вывести значение температуры на панель оператора (например, ОВЕН СП34).

4 Зафиксировать 100 циклов обмена и рассчитать коэффициент успешности.

Дополнительно. Изменить скорость на 19200 бод и сравнить устойчивость связи.

Отчет. Схема подключения, скриншот настроек порта, листинг программы, таблица результатов теста.

Контрольные вопросы

1 В чём принципиальное различие между адресацией в документации (40001) и программной реализации (адрес 0)?

2 Почему в линии RS-485 запрещена «звёздная» топология?

3 Какой функциональный код используется для записи нескольких дискретных выходов? Приведите пример запроса в шестнадцатеричном виде.

4 Чем отличается структура кадра Modbus RTU от Modbus TCP?

5 Какие меры повышают помехоустойчивость линии RS-485 в условиях промышленного предприятия?

10 Лабораторная работа № 10. Использование промышленных протоколов обмена данными OPC UA и MQTT в системах IIoT

Современные системы промышленного интернета вещей (IIoT) требуют надёжных, масштабируемых и безопасных механизмов обмена данными между устройствами, шлюзами, облачными платформами и корпоративными системами. Два ключевых протокола, ставших де-факто стандартами в промышленной автоматизации, – OPC UA (Open Platform Communications Unified Architecture) и MQTT (Message Queuing Telemetry Transport) – решают разные задачи в архитектуре IIoT и часто применяются совместно. Методические указания предназначены для освоения принципов работы, архитектурных особенностей и практического применения этих протоколов.

10.1 OPC UA: назначение и архитектура

Назначение. OPC UA – это платформонезависимый, сервис-ориентированный протокол для безопасной передачи данных в промышленной автоматизации.

Основные задачи.

1 Интеграция устройств разных вендоров (ПЛК, датчики).

2 Передача не только «сырых» данных, но и семантической модели (метаданные, единицы измерения, исторические данные).

3 Обеспечение сквозной безопасности (аутентификация, шифрование, аудит).

Архитектура.

1 Сервер – предоставляет данные (например, ПЛК с OPC UA-стеком или шлюз).

2 Клиент – запрашивает и подписывается на данные (SCADA, MES, облачное приложение).

3 Адресное пространство – иерархическая объектная модель (узлы, атрибуты, методы).

4 Транспортные уровни: TCP, HTTPS, WebSockets.

5 Безопасность: X.509-сертификаты, алгоритмы шифрования AES, поддержка политик безопасности (None, Sign, SignAndEncrypt).

Принцип работы. Клиент устанавливает сессию с сервером → проходит аутентификацию → читает/записывает значения → подписывается на изменения (мониторинг событий в реальном времени).

10.2 MQTT: назначение и архитектура

Назначение. MQTT – лёгкий протокол публикации/подписки для передачи данных в сетях с ограниченной пропускной способностью и нестабильным соединением. Применяется для:

– сбора данных с распределённых датчиков и устройств;

– интеграции с облачными платформами (AWS IoT, Azure IoT Hub);

– построения событийно-ориентированных архитектур.

Архитектура:

– брокер (Broker) – центральный узел маршрутизации сообщений (Mosquitto, HiveMQ, EMQX);

– публикант (Publisher) – устройство, отправляющее данные в топик;

– подписчик (Subscriber) – клиент, получающий сообщения из топика;

– топик (Topic) – иерархический путь для маршрутизации (например, plant/line1/temperature);

– QoS (Quality of Service): 0 (at most once), 1 (at least once), 2 (exactly once).

Принцип работы. Устройство подключается к брокеру → публикует сообщение в топик → брокер доставляет сообщение всем подписчикам топика. Поддерживает «последнее известное значение» (Retained Messages) и «завещание» (Last Will and Testament).

10.3 Настройка OPC UA-сервера и клиента

Инструменты.

1 Сервер: open62541 (open-source), Prosys OPC UA Simulation Server.

2 Клиент: UA Expert (Unified Automation), Python + библиотека opcua-asyncio.

Шаги настройки (пример с open62541):

```c
// Пример минимального сервера на C (open62541)
#include <open62541/server.h>
#include <open62541/server_config_default.h>

int main(void) {
    UA_Server *server = UA_Server_new();
    UA_ServerConfig_setDefault(UA_Server_getConfig(server));

    // Добавление переменной в адресное пространство
    UA_VariableAttributes attr = UA_VariableAttributes_default;
    UA_Int32 myVar = 42;
    UA_Variant_setScalar(&attr.value, &myVar, &UA_TYPES[UA_TYPES_INT32]);
    attr.displayName = UA_LOCALIZEDTEXT("en-US", "Temperature");
    attr.accessLevel = UA_ACCESSLEVELMASK_READ | UA_ACCESSLEVELMASK_WRITE;

    UA_NodeId nodeId = UA_NODEID_STRING(1, "temperature");
    UA_Server_addVariableNode(server, nodeId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES),
                              UA_QUALIFIEDNAME(1, "Temperature"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
                              attr, NULL, NULL);

    UA_Server_runUntilInterrupt(server);
    UA_Server_delete(server);
    return 0;
}
```

Подключение клиента.
1 Запустить сервер.
2 В UA Expert: Connect → opc.tcp://localhost:4840.
3 Перейти в Data Access View → найти узел temperature.
4 Правой кнопкой → Add to Subscription для мониторинга в реальном времени.

10.4 Передача данных через MQTT-брокер

Инструменты.
1 Брокер: Mosquitto (локальный), HiveMQ Cloud (облачный).
2 Клиенты: mosquitto_pub / mosquitto_sub, Python + paho-mqtt.
Установка Mosquitto (Ubuntu):

```
sudo apt install mosquitto mosquitto-clients
sudo systemctl start mosquitto
```

Публикация и подписка через CLI:

```
# Подписка на топик
mosquitto_sub -h localhost -t "plant/+/temperature" -v

# Публикация данных
mosquitto_pub -h localhost -t "plant/line1/temperature" -m "24.5" -q 1 -r
# -r — сохранить последнее значение как retained message
```

Задание для самостоятельного выполнения

Построение канала передачи данных «датчик → облако» через MQTT.

Цель: реализовать сквозной канал передачи данных с применением брокера MQTT.

Задачи.

1 Развернуть локальный брокер Mosquitto с аутентификацией (логин/пароль).

2 Написать симулятор датчика на Python, который:

– публикует температуру в топик sensors/{id}/temperature каждые 3 с;

– использует QoS=1 и retained message;

– отправляет «завещание» при отключении: sensors/{id}/status → offline.

3 Написать подписчика, который:

– сохраняет все полученные значения в CSV-файл с меткой времени;

– визуализирует последние 100 значений с помощью matplotlib.

4 Настроить мост (bridge) между локальным Mosquitto и облачным брокером (например, HiveMQ Cloud Trial).

11 Лабораторная работа № 11. Интеграция ОВЕН ПЛК210 со SCADA/облачной платформой

11.1 Передача технологических данных во внешнюю систему

Этап 1. Настройка ПЛК210 в среде CODESYS V3.5.

1 Создайте проект в CODESYS V3.5 и выберите устройство ОВЕН ПЛК210 (предварительно установите таргет-пакет с сайта owen.ru).

2 Определите структуру данных для экспорта в глобальной переменной или функциональном блоке (рисунок 11.1):

```
TYPE T_Telemetry :
STRUCT
    temperature : REAL;      // Температура, °C
    pressure : REAL;         // Давление, бар
    motor_state : BOOL;      // Состояние двигателя
    cycle_counter : UDINT;   // Счётчик циклов
END_STRUCT
END_TYPE
```

Рисунок 11.1 – Структура данных

3 Свяжите переменные с физическими входами/выходами через конфигурацию устройства (вкладка «Device» → «IO Mapping»).

Этап 2. Настройка протоколов передачи.

Приведена в таблице 11.1.

Этап 3. Тестирование связи.

1 Для проверки Modbus TCP используйте утилиту Modbus Poll (чтение регистров 4xxxx).

2 Для OPC UA – клиент UA Expert (просмотр дерева узлов).
3 Для MQTT – MQTT Explorer или встроенный мониторинг в ThingsBoard.

Таблица 11.1 – Параметры протоколов передачи

Протокол	Настройка в ПЛК210	Рекомендация
Modbus TCP	Встроенный сервер активируется в веб-конфигураторе ПЛК210 (адрес: http://<IP_ПЛК> → вкладка «Сеть» → «Modbus TCP»). Укажите порт (стандартно 502), диапазон регистров для отображения переменных	Универсальный выбор для интеграции с большинством SCADA (Trace Mode, Master SCADA, Ignition)
OPC UA	Сервер активируется в веб-конфигураторе (вкладка «OPC UA»). Создайте пространство имён, определите узлы для переменных. Поддерживается шифрование (безопасность «Sign & Encrypt»)	Предпочтителен для современных систем с требованиями к безопасности
MQTT	Настройка в веб-конфигураторе (вкладка «MQTT»): брокер: thingsboard.cloud или локальный Mosquitto порт: 1883 (незашифрованный) / 8883 (с TLS) клиентский ID: PLC210_<номер> топик публикации: v1/devices/me/telemetry payload: JSON-формат	Используется для прямой передачи в облако. Поддерживает QoS 0/1/2

11.2 Визуализация данных (*графики, дашборды*)

Вариант А. Локальная SCADA (Trace Mode, Master SCADA, Ignition)
1 Создайте проект в выбранной SCADA.
2 Добавьте драйвер Modbus TCP Master:
- IP-адрес ПЛК210;
- порт 502;
- тайм-аут ответа 1000 мс.

3 Настройте теги с привязкой к регистрам ПЛК (например, Holding Register 40001 → температура).
4 Разработайте визуализацию:
- цифровые индикаторы для аналоговых параметров;
- светодиоды для дискретных состояний;
- график тренда (архивация с шагом 1 с).

Вариант Б. Облачная платформа ThingsBoard.
1 Зарегистрируйте аккаунт на сайте thingsboard.cloud (бесплатный демо-режим).
2 Создайте устройство → получите Device Token.
3 В веб-конфигураторе ПЛК210 настройте MQTT-клиент:
- Host: thingsboard.cloud;
- Port: 1883;
- Username: <Device Token>;
- Password: оставить пустым;

– Topic: v1/devices/me/telemetry;
– Payload: {"temperature":25.5,"pressure":6.2}.

4 Создайте дашборд в ThingsBoard:
– виджет «Линейный график» для температуры (история за 24 ч);
– виджет «Гаусметр» для давления с цветовыми зонами;
– виджет «Светодиод» для состояния двигателя.

11.3 Удаленный мониторинг состояния оборудования

ПЛК210 позволяет передавать не только технологические, но и диагностические параметры, показанные в таблице 11.2.

Таблица 11.2 – Диагностические параметры

Параметр	Источник в ПЛК210	Способ передачи
Цикловое время	Системная переменная %SW6 (в CODESYS)	Передать как телеметрию в облако
Состояние связи по RS-485	Системные флаги ошибок (например, %MX100.0)	Дискретный тег в SCADA
Температура ЦПУ	Через веб-интерфейс ПЛК210 (вкладка «Система»)	Опрос по HTTP API или включение в телеметрию
Статус «онлайн/оффлайн»	Отсутствие MQTT-публикаций > 60 с	Правило в ThingsBoard: «Inactivity Alarm»

Настройка оповещений в ThingsBoard.
1 Создайте правило (Rule Chain) → узел «Create Alarm».
2 Условие: temperature > 80 ИЛИ cycle_time > 15 мс.
3 Действие: отправка email через интеграцию с почтовым сервисом.

Задание для самостоятельного выполнения
Интеграция ПЛК210 с локальной SCADA через Modbus TCP.
Цель: освоить базовую передачу данных от ПЛК210 в SCADA-систему.
Исходные данные:
– ОВЕН ПЛК210 (любая модификация с Ethernet);
– SCADA Trace Mode 7 или бесплатная альтернатива – Ignition (демо-версия);
– среда программирования CODESYS V3.5.
***Задачи*.**
1 В CODESYS создайте проект с четырьмя переменными: temp (REAL), press (REAL), pump_on (BOOL), counter (UDINT).
2 Настройте веб-конфигуратор ПЛК210:
– назначьте статический IP-адрес в той же подсети, что и ПК со SCADA;
– активируйте сервер Modbus TCP, сопоставьте переменные с регистрами:
a) temp → Holding Register 40001;

б) press → Holding Register 40002;
в) pump_on → Coil 00001;
г) counter → Holding Register 40003.

3 В SCADA создайте драйвер Modbus TCP Master, добавьте теги с указанными адресами.

4 Разработайте визуализацию:
– цифровым табло для температуры и давления;
– анимированным насосом (вращение при pump_on = TRUE);
– графиком тренда для температуры (архивация 1 точка/с).

5 Протестируйте работу: изменяйте значения переменных в CODESYS (в режиме онлайн) и наблюдайте отражение на мнемосхеме.

Контрольные вопросы

1 Какие протоколы передачи данных поддерживает ПЛК210 «из коробки» без дополнительных шлюзов?

2 В чём преимущество использования встроенного OPC UA-сервера ПЛК210 перед традиционным Modbus TCP?

3 Как настроить передачу данных из ПЛК210 в облако при отсутствии статического белого IP-адреса?

4 Какие диагностические параметры ПЛК210 можно использовать для прогнозирования отказов?

5 Чем отличается архитектура «ПЛК210 → OwenCloud» от «ПЛК210 → ThingsBoard»?

Список литературы

1 Industrial Internet Consortium. Industrial Internet Reference Architecture (IIRA). Версия 1.10.2022. – URL: https://www.iiconsortium.org/iira.

2 Информационные технологии. Интернет вещей. Термины и определения: ГОСТ Р 71777–2024.

3 Каталог продукции компании «ОВЕН». – URL: https://owen.ru/product/.

4 Руководство по эксплуатации ПЛК210. – URL: https://owen.ru/product/plk210.

5 CODESYS Development System V3. Руководство пользователя. 3S-Smart Software Solutions GmbH. – URL: https://owen.ru/product/codesys_v3.

6 Видеокурс «ПЛК210. Быстрый старт». – URL: https:// owen.ru/media/video.

7 Примеры проектов для CODESYS: раздел «Поддержка» → «Примеры применения». – URL: https:// owen.ru.

8 ThingsBoard Cloud (бесплатный демо). – URL: https://thingsboard.cloud.

9 Ignition (30-дневная демоверсия). – URL: https://inductiveautomation.com.

10 Modbus Poll (утилита для тестирования). – URL: https://modbustools.com.

11 OwenCloud. Облачный сервис. Руководство пользователя. – URL: https://www.elecomural.ru/media/content/owen_files/content_docs/rp_owencloud.pdf.