Научный обзор
DOI: https://doi.org/10.17816/2074-0530-634664 EDN: RGCPCD

Расширение возможностей промышленного контроллера путём интеграции дополнительных протоколов взаимодействия

С.С. Гусев[1], В.В. Макаров[2], А.Ю. Мещеряков[2]

[1] *Ростелеком, Москва, Россия;*
[2] *Институт проблем управления им. В.А. Трапезникова РАН, Москва, Россия*

АННОТАЦИЯ

Обоснование. Промышленные контроллеры в современном мире решают множество разноплановых задач. Одной из основных задач для программируемого логического контроллера является передача данных в другие контроллеры, устройства и системы. Данные могут храниться в самом контроллере, либо могут быть получены с внешних конечных устройств, других контроллеров, датчиков и так далее.

Целью выполнения работы является расширение возможностей промышленного контроллера путём интеграции дополнительных протоколов взаимодействия.

Материалы и методы. В нынешнее время протоколы передачи данных в системах автоматизации, управления и телемеханики играют важнейшую роль, предоставляя возможность обмена данными между устройствами всех уровней автоматизации. Поддержка устройством возможности отправлять и принимать информацию позволяет использовать данные решения в системах, которые требуют аккумулировать, анализировать, обрабатывать и отображать необходимые для системы данные.

Результаты. Результатом исследовательской работы является то, что на основании задачи, связанной с разработкой методики интеграции протоколов взаимодействия в промышленный контроллер взаимодействие контроллера с внешними устройствами позволяет реализовывать управление механизмами, анализ данных для последующей обработки информации, аккумулирование и обработку данных для передачи в системы верхнего уровня, такие как SCADA системы (supervisory control and data acquisition — диспетчерское управление и сбор данных).

Заключение. В результате проведённого в исследовании анализа предметной области выявлены основные представители протоколов передачи данных, наиболее часто встречающиеся в промышленных контроллерах, обозначены для них назначения и основные особенности. Предложенная в данной работе методика интегрирования протокола взаимодействия в среде программирования контроллера позволяет, следуя определенным правилам, произвести полноценное внедрение нового способа обмена данными.

Ключевые слова: промышленные котроллеры; передача данных; устройства; протоколы взаимодействия; уровни автоматизации; системы верхнего уровня; обмен данными.

Как цитировать:
Гусев С.С., Макаров В.В., Мещеряков А.Ю. Расширение возможностей промышленного контроллера путём интеграции дополнительных протоколов взаимодействия // Известия МГТУ «МАМИ». 2025. Т. 19, № 1. С. х–у.
DOI: 10.17816/2074-0530-634664 EDN: RGCPCD

Рукопись получена: 29.07.2024 Рукопись одобрена: 11.04.2025 Опубликована online: 16.06.2025

Review
DOI: https://doi.org/10.17816/2074-0530-634664 EDN: RGCPCD
ROBOTS, MECHATRONICS AND ROBOTIC SYSTEMS

Increasing the Abilities of an Industrial Controller by Means of Integration of Additional Interaction Protocols

Sergey S. Gusev[1], Vadim V. Makarov[2], Alexander Yu. Meshcheryakov[2]

[1] *Rostelecom, Moscow, Russia;*
[2] *V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russia*

ABSTRACT

BACKGROUND: Currently, industrial controllers solve numerous diversified tasks. One of the main tasks for the programmed logical controller (PLC) is data transferring to other controllers, devices and systems. The data can be stored in the controller or be obtained from external terminal devices, other controllers, sensors etc.
AIM: Increasing the abilities of an industrial controller by means of integration of additional interaction protocols.
METHODS: Currently, data transfer protocols in automation, control and telemechanics systems are critical as they give an opportunity of data exchange between devices of all levels of automation. The device's capability of sending and receiving the information allows using these solutions in the systems that require accumulation, analysis, processing and displaying the data demanded to the system.
RESULTS: Based on the problem related to the development of the integration of interaction protocols in an industrial controller, the interaction of the controller with external devices allows implementing control on mechanisms, data analysis for further information processing, accumulation and processing of the data for transferring to the upper-level systems like the SCADA (supervisory control and data acquisition) systems.
CONCLUSION: As the result of the conducted study, domain analysis was conducted, main types of data transfer protocols, widely used in industrial controllers, were found, their purposes and main features were denoted. The method of integration of the interaction protocol into the controller's programming environment obtained in this study allows for the complete implementation of the new data transfer method by following certain rules.

Keywords: industrial controllers; data transfer; devices; interaction protocols; automation levels; upper-level systems; data exchange.

TO CITE THIS ARTICLE:
Gusev SS, Makarov VV, Meshcheryakov AYu. Increasing the Abilities of an Industrial Controller by Means of Integration of Additional Interaction Protocols. *Izvestiya MGTU «MAMI»*. 2025;19(1):x–y. DOI: 10.17816/2074-0530-634664 EDN: RGCPCD

Submitted: 29.07.2024 Accepted: 11.04.2025 Published online: 16.06.2025

ВВЕДЕНИЕ

Взаимодействие контроллера с внешними устройствами позволяет реализовывать управление механизмами, анализ данных для последующей обработки информации, аккумулирование и обработку данных для передачи в системы верхнего уровня, такие как SCADA системы. Взаимодействие программируемого логического контроллера (ПЛК) с другими устройствами обеспечивается с помощью промышленных протоколов передачи данных. Промышленные протоколы передачи данных в свою очередь можно разделить на предметные области, в которых каждое решение имеет свои преимущества. Протоколы типа master–slave позволяют обеспечить беспрерывную связь между двумя устройствами в системе, как пример — протокол Modbus.

Перечень промышленных протоколов передачи данных стандартных для ПЛК представлен в табл. 1.

Таблица 1. Стандартный набор протоколов передачи данных на программируемый логический контроллер
Table 1. Standard set of data transfer protocols for a programmed logical controller

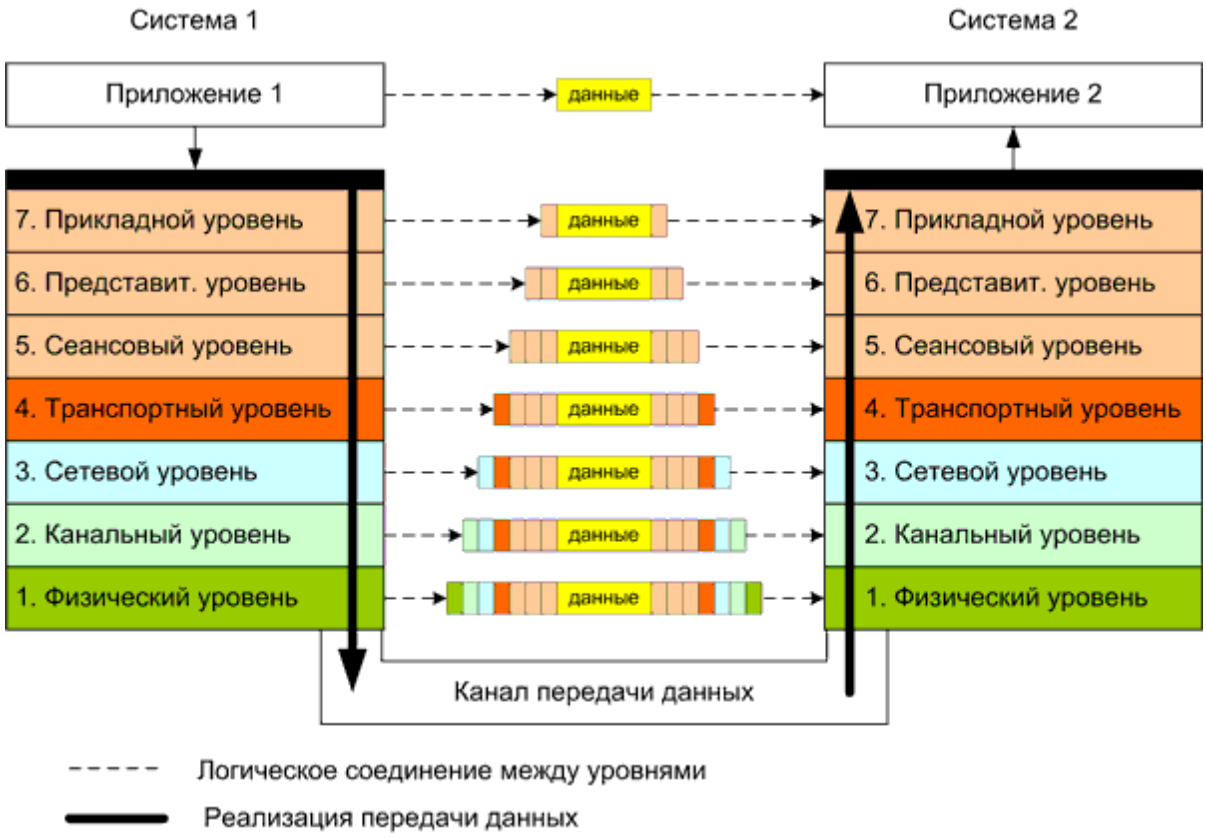
Протокол Protocol	Интерфейс Interface	Транспорт Transport	Архитектура Architecture	Предназначение Applicability
Modbus RTU	Последовательный порт	—	master–slave	Обмен данными с конечными устройствами Data exchange with terminal devices
Modbus TCP	Ethernet\IEEE 802.3	TCP\IP	master(client)–slave(server)	Обмен данными с конечными устройствами Data exchange with terminal devices
IEC 60870-5-101	Последовательный порт	—	master–slave	Передача данных в системы верхнего и среднего уровней автоматизации Data transfer to upper and middle-level automation systems
IEC 60870-5-104	Ethernet\IEEE 802.3	TCP\IP	master(client)–slave(server)	Передача данных в системы верхнего и среднего уровней автоматизации Data transfer to upper and middle-level automation systems
Telnet	Ethernet\IEEE 802.3	TCP\IP	client –server	Обеспечение получения диагностических данных Providing diagnostic data
FTP	Ethernet\IEEE 802.3	TCP\IP	client–server	Получение доступа к файловой системе контроллера Getting access to the controller's file system
NTP	Ethernet\IEEE 802.3	TCP\IP	client–server	Синхронизация времени Time synchronization

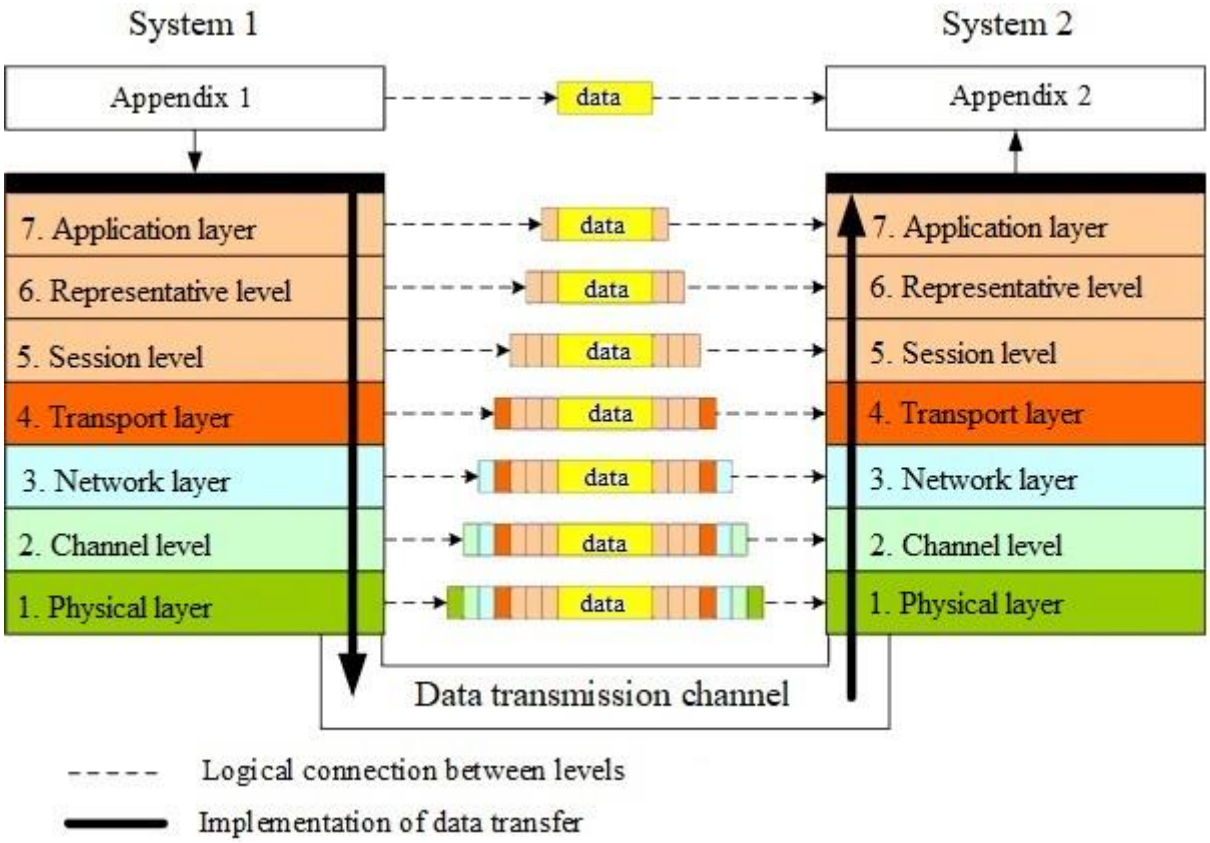
Целью выполнения работы является расширение возможностей промышленного контроллера путём интеграции дополнительных протоколов взаимодействия.

Материалы и методы. В нынешнее время протоколы передачи данных в системах автоматизации, управления и телемеханики играют важнейшую роль, предоставляя возможность обмена данными между устройствами всех уровней автоматизации. Поддержка устройством возможности отправлять и принимать информацию позволяет использовать данные решения в системах, которые требуют аккумулировать, анализировать, обрабатывать и отображать необходимые для системы данные [1].

Обеспечение поддержки контроллером протокола передачи данных

Для обеспечения поддержки контроллером того или иного протокола передачи данных следует реализовать управленческий алгоритм (драйвер) данного протокола. Протоколы порой сильно отличаются друг от друга в принципе и последовательности действий для обеспечения передачи данных. Но в каждом есть основополагающие принципы, объединяющие их. Протоколы передачи данных характеризуются различными уровнями взаимодействия систем по модели OSI (рис. 1).





Система 1	System 1
Система 2	System 2
Приложение 1	Application 1
Приложение 2	Application 2
Данные	Data
Физический уровень	Physical layer
Канальный уровень	Channel level
Сетевой уровень	Network layer
Транспортный уровень	Transport level
Сеансовый уровень	Session level
Представительский уровень	Representative level
Прикладной уровень	Application layer
Канал передачи данных	Data transmission channel
Логическое соединение между уровнями	Logical connection between levels
Реализация передачи данных	Implementation of data transfer

Рис. 1. Сетевая модель The Open Systems Interconnection model OSI.
Fig. 1. The Open System Interconnection model (OSI).

Сетевая модель OSI (The Open Systems Interconnection model) — сетевая модель стека сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии [2].

Взяв определённый протокол передачи данных, его можно описать и определить свойства по данной модели, что в свою очередь позволяет описать методику реализаций данных алгоритмов.

Выбор протокола для реализации

В первую очередь необходимо определиться в выборе протокола, подлежащего реализации. Также следует определить данный протокол по модели OSI.

Для примера обратим внимание на протокол передачи данных Modbus RTU. В стандартной реализации данный протокол сам по себе реализует прикладной уровень модели OSI. Базируется на физическом уровне, используя последовательные порты RS232 или RS485. На канальном уровне — архитектуру master–slave (табл. 2).

Таблица 2. Архитектура протокола Modbus RTU
Table 2. The Modbus RTU protocol arcitecture

Номер уровня Level number	Название уровня Name of the level	Реализация Realization
7	Прикладной Applied	Прикладной протокол MODBUS MODBUS Application Protocol
б	Уровень представления Presentation level	Нет No
5	Сеансовый Session	Нет No
4	Транспортный Transport	Нет No
3	Сетевой Network	Нет No
2	Канальный (передачи данных) Channel (data transmission)	Протокол «ведущий/ведомый» Master/Slave protocol Режимы RTU и ASCII RTU and ASCII modes
1	Физический Physical	RS-485 или RS-232 RS-485 or RS-232

Из этого следует, что для драйвера данного протокола необходимы доступ к последовательным портам устройства, его буферам RX и TX и функция передачи пакетов через последовательные порты. Канальный уровень в данном случае реализуется путём разделения отправленного пакета (TX) и ожидания прихода ответа на него (RX) [3].

В свою очередь у такого популярного протокола передачи данных как MODBUS существует отдельная реализация, именуемая MODBUS TCP\IP (табл. 3).

Таблица 3. Архитектура протокола Modbus TCP
Table 2. The Modbus TCP protocol architecture

Номер уровня Level number	Название уровня Name of the level	Реализация Realization
7	Прикладной Applied	Прикладной протокол MODBUS MODBUS Application Protocol
6	Уровень представления Presentation level	Нет No
5	Сеансовый Session	Нет No
4	Транспортный Transport	TCP TCP
3	Сетевой Network	IP IP
2	Канальный (передачи данных)	ETHERNET, IEEE 802.11

Номер уровня Level number	Название уровня Name of the level	Реализация Realization
	Channel (data transmission)	ETHERNET, IEEE 802.11
1	Физический Physical	Витая пара (RJ), оптоволокно Twisted pair (RJ), optical fiber

На всех уровнях, кроме прикладного, данный протокол отличается от своего «брата» RTU. На физическом уровне используется витая пара, оптоволокно, также возможно использование WI-FI. На канальном уровне, соответственно, при использовании витой пары, будет ETHERNET, а при использовании WI-FI — IEEE 802.11. На сетевом уровне поиск конечных устройств и их уникальных адресов будет разворачиваться с использованием протокола IP. На транспортном уровне данный протокол базируется на TCP, так как для него необходима целостность и правильный порядок отправленных и полученных данных, что не может предоставить, например, UDP [4].

Исходя из данных вводных, приходим к выводу о том, что для реализации драйвера MODBUS TCP\IP необходимо следующее.

1. Наличие на устройстве порта ETHERNET либо сетевой карты с поддержкой IEEE 802.11.
2. Наличие возможности конфигурировать IP адрес устройства, маску и шлюз.
3. Доступ для функций работы с сокетами типа TCP\IP.

Обе реализации MODBUS RTU и MODBUS TCP\IP на прикладном уровне практически не различаются, так как эти протоколы уровня приложения.

Данный пример позволят сделать вывод о том, что реализация протоколов передачи данных верхнего уровня или же уровня приложений, которые позволяют передавать информацию удобным, определённым стандартом образом практически не зависит от используемых протоколов нижних уровней, с одной стороны. Имеется в виду, что алгоритм работы с данными, определёнными протоколом, обособлен от других уровней.

С другой стороны, то, на чём базируется протокол, непосредственно влияет на его конечные свойства, быстродействие, устойчивость и т. д. Но такие параметры чаще всего уже определены для каждого протокола, т. к. разработка спецификаций уже учитывает особенности того, на чём основан тот или иной протокол.

Определение архитектуры протокола

Определившись с выбором, обращаем внимание на архитектуру и тип протокола. От этого зависит общая структура построения программы.

Далее необходимо определить, на каком физическом канале связи будет использоваться протокол. От этого зависит, на каком протоколе физического уровня будет базироваться протокол.

Рассмотрим некоторые возможные архитектуры:

- ведущий–ведомый (master–slave);
- клиент–сервер (client–server);
- издатель–подписчик (publisher–subscriber).

Архитектура ведущий–ведомый (master–slave)

В архитектуре протоколов передачи данных с ведущим и ведомым (master–slave) участвуют два устройства. Ведущее устройство отправляет запросы на получение данных, в то время как ведомое отвечает на полученные запросы. Инициатором передачи данных может быть только ведущее устройство. Таким образом, процесс передачи данных осуществляется через отправку пакета запроса и получение ответа на него [5].

Архитектура клиент–сервер (client–server)

Клиент–сервер (client–server) — архитектура организации передачи данных, к которой присутствует устройство–сервер, к которому подключаются устройства–клиенты. После подключения клиенты имеют возможность обмениваться сообщениями как с сервером, так и с клиентами, подключёнными к нему непосредственно через сервер. Инициатором подключения в данном случае будут клиенты, которые отправляют запрос на подключение. Так, при необходимости обмена данными запросы на информацию передают клиенты серверу [6].

Архитектура издатель–подписчик (publisher–subscriber)

Издатель–подписчик (publisher–subscriber) — способ организации сети, в которой участвует два типа устройств — серверы и клиенты. Отличие от клиент–серверной архитектуры заключается в том, что клиенты могут выступать в двух ролях: издатель и подписчик. Издатель имеет возможность публиковать сообщения в темы (topics), которые хранятся в сервере, а подписчик — подписываться на темы [7]. Таким образом, организация передачи данных приставляет из себя такую последовательность:

1. клиент–подписчик подписывается на тему в сервере;
2. клиент–издатель публикует сообщение в топиках;
3. сервер при получении новых сообщений в топике передаёт их всем подписавшимся.

Анализ необходимых возможностей алгоритма сервера

Проводится анализ необходимых действий, которые должен произвести алгоритм сервера для того, чтобы к нему имели возможность подключаться несколько клиентов.

Для реализации возможности подключения клиентов к серверу необходимо:

– создать сокет для подключения по TCP\IP;

– принять входящее подключение по данному сокету;

– передать данные о входящем соединении в программный блок, описывающий работу с клиентом.

Создание сокета и принятие входящего подключения производится возможностями операционной системы устройства, на котором реализуется алгоритм.

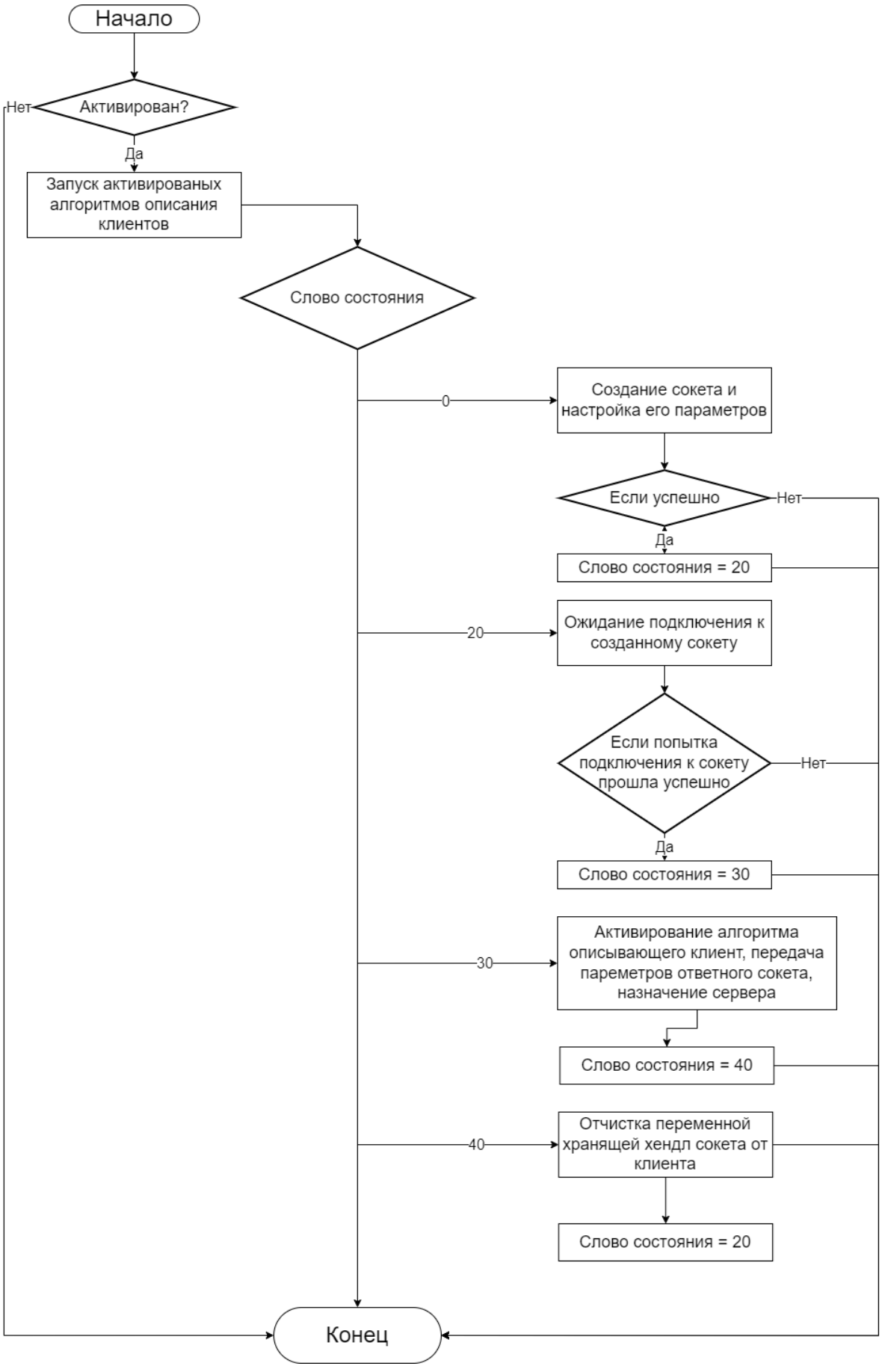
Передача информации о входящем подключении осуществляется возможностями среды программирования.

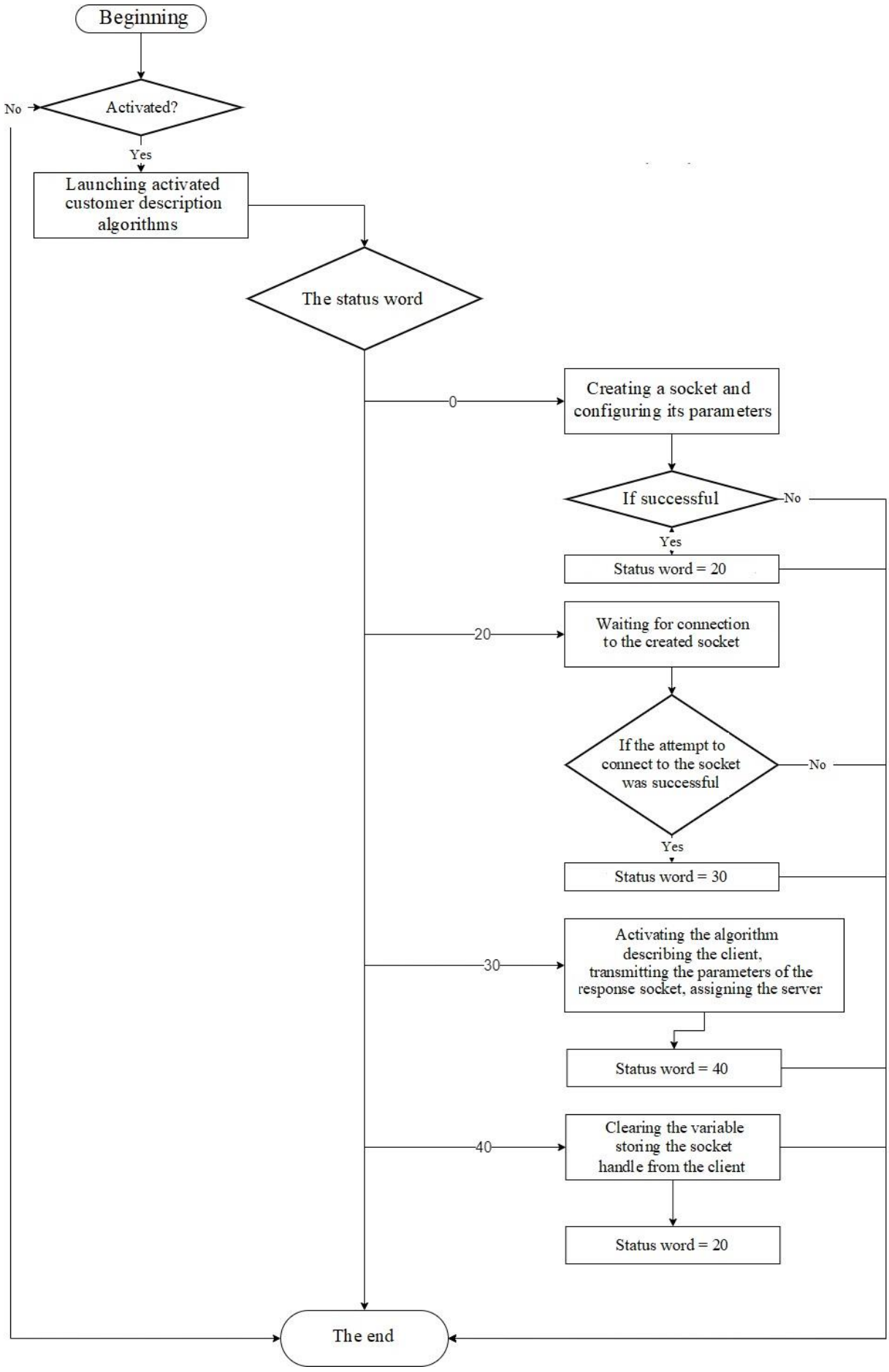
Приведена блок–схема алгоритма сервера (рис. 2), на которой отражены шаги и условные конструкции, позволяющие реализовать необходимые требования.

В программную часть сервера помимо алгоритма, обеспечивающего соединение на уровне TCP\IP, должен входить алгоритм, описывающий работу сервера прикладного уровня с клиентом.

В данном программном модуле должна быть реализована обработка получаемых данных от сервера при помощи буферизованных данных из принимаемых пакетов. А также обработка и буферизация данных, передаваемых серверу [8].

Передача и получение пакетов данных на уровне TCP\IP осуществляется при помощи возможностей, которые может предоставить операционная система.





Начало	Start

Активирован	Activated
Запуск активированных алгоритмов описания клиентов	Launching activated client description algorithms
Слово состояния	Status word
Создание сокета и настройка его параметров	Creating a socket and configuring its parameters
Если успешно	If successful
Слово состояния = 20	Status word = 20
Ожидание подключения к созданному сокету	Waiting for connection to the created socket
Если попытка подключения к сокету прошла успешно	If the attempt to connect to the socket was successful
Слово состояния = 30	Status word = 30
Активирование алгоритма, описывающего клиента, передача параметров ответного сокета, назначение сервера	Activating the client description algorithm, transmitting the parameters of the response socket, assigning the server
Слово состояния = 40	Status word = 40
Очистка переменной, хранящей хендл сокета от клиента	Clearing the variable storing the socket handle from the client
Слово состояния = 20	Status word = 20
Конец	End

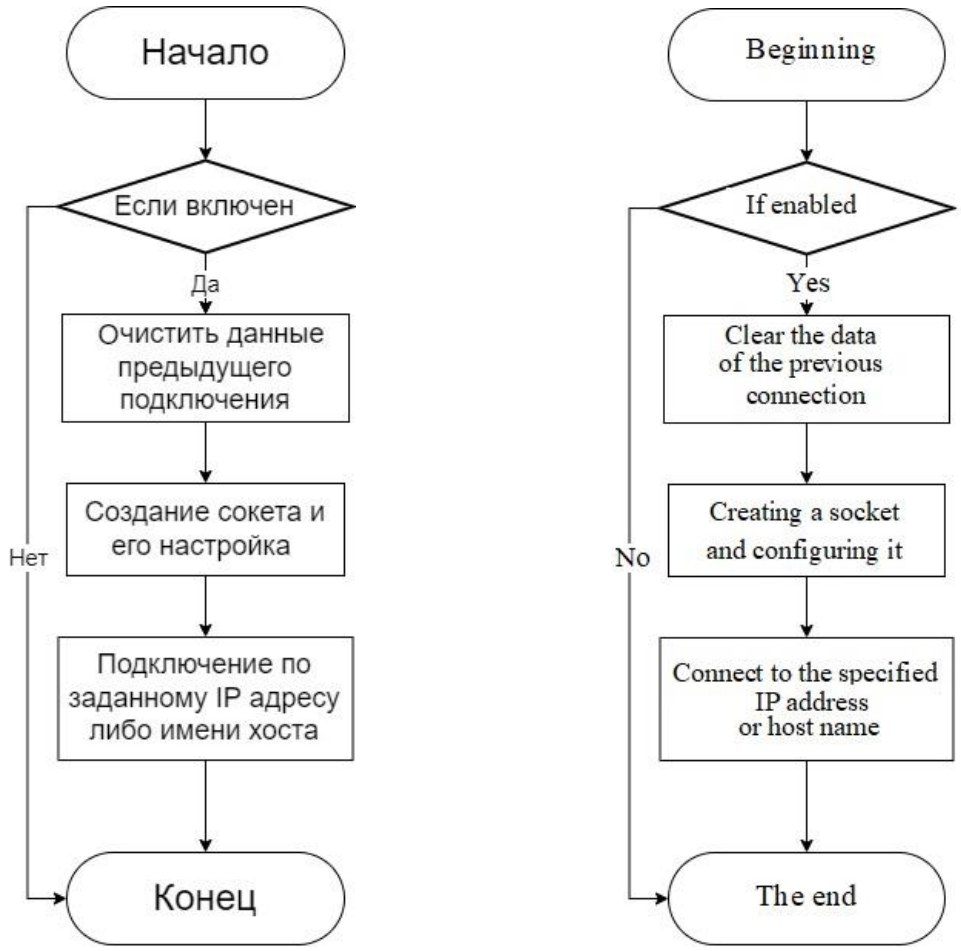
Рис. 2. Алгоритм работы сервера.
Fig. 2. The server operation algorithm.

Непосредственная реализация данной части драйвера зависит от протокола передачи данных, который необходимо внедрить на устройство.

Анализ необходимых возможностей алгоритма клиента

Проводится анализ необходимых действий, которые должен произвести алгоритм клиента для того, чтобы он мог подключиться к сокету сервера и держать данное подключение.

Алгоритм работы клиентской части соединения клиент–сервер представлен на рис. 3.



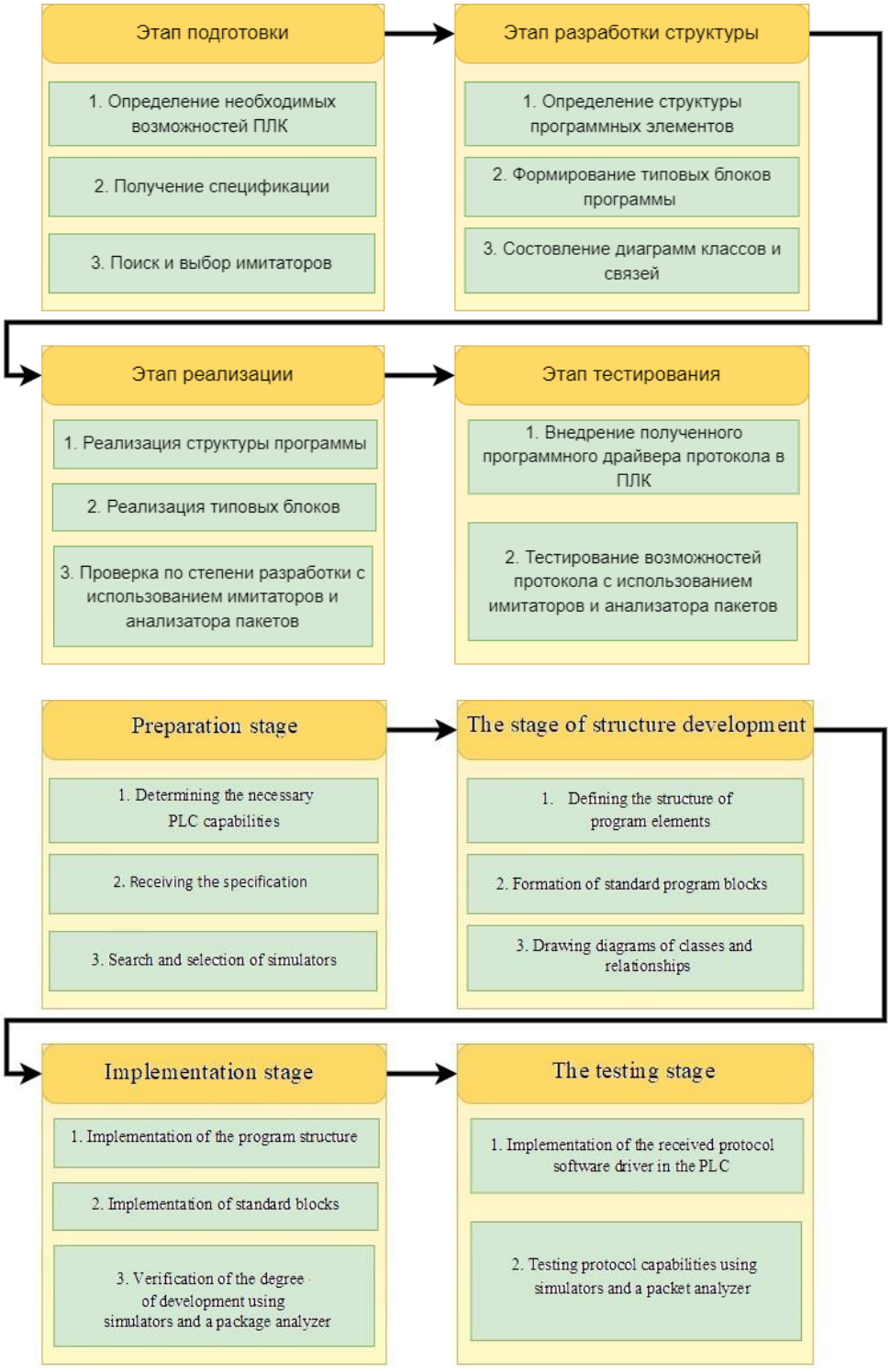
Начало	Start
Если включён	If turned on
Очистить данные предыдущего подключения	Clear the data of the previous connection
Создание сокета и его настройка	Creating a socket and configuring it
Подключение по заданному IP-адресу либо по имени хоста	Connect to the specified IP address or host name
Конец	End

Рис. 3. Алгоритм работы клиента.
Fig. 3. The client operation algorithm.

Во включённом состоянии алгоритм в первую очередь всегда единожды очищает области памяти от хранения данных предыдущего подключения. Это необходимо для того, чтобы информация о новом подключении с вероятностью в сто процентов была записана верно. Дальнейшим шагом является создание сокета, при помощи которого будет осуществляться подключение, — настройка требуется для нестандартных способов подключения. После того как сокет был успешно создан, требуется подключиться к сокету сервера по заданному IP-адресу либо имени хоста. В свою очередь при успешном подключении клиент с сервером могут обмениваться пакетами данных [9].

Методика интеграции протокола передачи данных в программируемый логический контроллер

Разработанная методика интеграции программного драйвера на ПЛК включает в себя четыре основных этапа. Каждый этап позволяет обеспечить необходимый минимум информации для того, чтобы реализовать программный драйвер. Визуальная схема данной методики представлена на рис. 4.



Этап подготовки	Preparation stage
1. Определение необходимых возможностей ПЛК	1. Determining the necessary PLC capabilities
2. Получение спецификации	2. Receiving the specification

3. Поиск и выбор имитаторов	3. Search and selection of simulators
Этап реализации	Implementation stage
1. Реализация структуры программы	1. Implementation of the program structure
2. Реализация типовых блоков	2. Implementation of standard blocks
3. Проверка по степени разработки с использованием имитаторов и анализатора пакетов	3. Verification of the degree of development using simulators and a package analyzer
Этап разработки структуры	The stage of structure development
1. Определение структуры программных элементов	1. Defining the structure of program elements
2. Формирование типовых блоков программы	2. Formation of standard program blocks
3. Составление диаграмм классов и связей	3. Drawing diagrams of classes and relationships
Этап тестирования	The testing stage
1. Внедрение полученного программного драйвера протокола в ПЛК	1. Implementation of the received protocol software driver in the PLC
2. Тестирование возможностей протокола с использованием имитаторов и анализатора пакетов	2. Testing protocol capabilities using simulators and a packet analyzer

Рис. 4. Методика интеграции протокола передачи данных в программируемый логический контроллер.
Fig. 4. The method of integration the data transfer protocol into the programmed logical controller (PLC).

Этап подготовки

В первую очередь для того чтобы решить задачу реализации протокола передачи данных для ПЛК в среде программирования Codesys 3.5, требуется определить, какой стандарт будет подлежать реализации. Определившись с протоколом передачи данных, подлежащим реализации, необходимо получить спецификацию по его работе, документ, в котором описаны механизмы работы, как и в каком виде будут передаваться данные, а также требования к работе программного драйвера.

После получения полной и исчерпывающей информации по требованиям к протоколу необходимо выбрать программируемый логический контроллер, на котором данный протокол может быть реализован.

В случае, если протокол передачи данных на физическом уровне использует последовательный порт, соответственно, данным портом должен обладать ПЛК. Абсолютно таким же образом выглядит требование к ПЛК: если протокол использует на физическом уровне сетевой канал, то контроллер должен обладать сетевой картой и портами Ethernet либо поддерживать беспроводной канал передачи данных, к примеру, WI-FI.

Этап разработки структуры

Приступая к началу разработки приложения, необходимо определить основную структуру программы. Оптимальной для данной задачи структурой приложения для ПЛК, программируемого при помощи Codesys 3.5, является проект вида библиотеки, т. к. драйвер протокола передачи данных при использовании будет являться инструментом для формирования автоматизированных систем и будет предоставлять возможности и функции для передачи информации. Непосредственный открытый код для этого программисту, использующему данный драйвер, не требуется. Достаточно ограничиться исчерпывающей документацией по работе данного программного продукта.

Использование проекта в виде библиотеки удобно для дальнейшего внесения изменений, т. к. в приложениях, использующих данную библиотеку, будет достаточно обновить её в менеджере библиотек, тем самым максимально быстро получить новые возможности программного продукта.

Структура проекта библиотеки для Codesys 3.5 представляет собой набор программных блоков. А именно функциональных блоков, функций, структур, перечислений и т. д. В свою

очередь функциональные блоки предназначены для того, чтобы хранить в себе алгоритмическую часть драйвера, т. к. позволяют иметь методы для хранения часто используемого кода. Функции, как и в любой другой среде программирования, являются набором действий, которые устройство должно выполнить за один цикл программы и вернуть результат. Структура данных представляет собой структуры из нескольких заранее определённых переменных определённого типа. Областей применения структур множество, но по большей части они требуются для того, чтобы обозначить определённого типа группы информации, к примеру, типовой пакет данных. Перечисления при программировании ПЛК чаще всего требуются для обозначения последовательностей, шагов либо кодов возврата после выполнения.

Важной возможностью драйвера протокола передачи данных является логирование, т. е. запись сообщений о том, как проходит процесс работы драйвера. Функции записи сообщений требуется добавить в каждый процесс одномоментного действия, для того чтобы задокументировать прохождение драйвером того или иного этапа программы.

Также на данном этапе разработки немаловажным процессом является определение типовых участков кода. Т. к. алгоритм программного драйвера так или иначе выполняет ограниченный набор действий, для данных действий требуется определить последовательность шагов программы.

Основными процессами для драйвера, обеспечивающего обмен информацией, являются:

– процесс инициализации;
– процесс формирования буфера данных на отправку;
– процесс передачи данных;
– процесс получения данных;
– процесс анализа полученных данных.

Во время процесса инициализации алгоритм драйвера должен определить начальные настройки, указанные во входных параметрах. Сформировать необходимые области памяти для работы и перейти к основному циклу выполнения программы.

Формирование буфера данных на отправку выполняется при необходимости дальнейшей передачи сформированной информации другому устройству. Желательными шагами во время выполнения процесса являются определение информации, которая подлежит отправке, копирование областей памяти в буфер и запись проделанных действий в лог.

Передача данных осуществляется при помощи функций, доступных в среде выполнения, в которой сформированный буфер данных передаётся по каналу связи в другое устройство. Для данного процесса необходимо определить условие при успешной отправке и обеспечить обрыв данного действия по тайм-ауту, так как возможно, что при определённых условиях отправка никогда не будет произведена успешно [10].

Во время процесса получения данных ПЛК должен циклично вызывать функцию ожидания принятия пакета данных. Так как протокол передачи данных представляет собой набор правил, в котором подразумевается, что драйвер не должен неограниченное количество времени выполнять одно действие, на получение пакета необходимо установить два условия. Условие, которое выполнится при успешном получении пакета данных, а также условие завершения процесса по тайм-ауту, для того чтобы оборвать данный процесс по истечении определённого времени.

После успешного получения пакета данных алгоритм драйвера должен произвести обработку и анализ полученных данных. Данный функционал реализуется в программном процессе анализа пакета, который должен определить пакет данных на наличие полезной нагрузки и передать полученные данные следующим шагам алгоритма.

Этап практической реализации

На данном этапе производится непосредственная реализация драйвера протокола передачи данных на основе описанных на предыдущем этапе шагов. А именно требуется создать

проект библиотеки для Codesys 3.5 и реализовать описанные ранее блоки программы, функции и структуры. По мере написания кода программы имеет смысл проверять реализованный функционал с использованием имитаторов протокола передачи данных, выбранных на этапе подготовки.

Завершающим этапом разработки программного драйвера протокола передачи данных является тестирование полученной библиотеки с использованием сторонних устройств и имитаторов. Для более точного проведения испытаний необходимо построить стенд, состоящий из минимум трёх устройств для протоколов архитектуры клиент–сервер, двух устройств — для архитектуры ведущий–ведомый и четырёх устройств — для протоколов архитектуры издатель–подписчик. Эти требования представляют собой минимальную конфигурацию, необходимую для всесторонней оценки возможностей нового драйвера протокола передачи данных. Важно отметить, что сторонние устройства и имитаторы должны не только эмулировать, но и реализовывать полный функционал протокола передачи данных, подлежащего тестированию. Это позволит в реальных условиях оценить корректность обработки сообщений, а также выявить возможные проблемы, связанные с совместимостью и производительностью.

Тестирование должно проводиться в различных сценариях, включая как стандартные, так и предельные условия работы, что поможет выявить поведение системы в ситуациях, близких к реальным эксплуатационным условиям. Например, в рамках архитектуры клиент–сервер целесообразно протестировать, как драйвер справляется с множественными одновременными запросами от клиентов, а также как он обрабатывает задержки и потери пакетов. Для архитектуры ведущий–ведомый важно проверить, как система реагирует на изменения в состоянии ведущего устройства, включая его отключение и повторное подключение. Аналогично в архитектуре издатель–подписчик необходимо убедиться, что подписчики корректно получают сообщения от издателя и могут обрабатывать их в реальном времени.

Для анализа правильности последовательности передачи пакетов данных требуется использовать анализатор пакетов — специализированную программу, которая визуально отображает пакеты данных, отправленные и полученные через устройство. Такой инструмент обеспечивает детальный мониторинг и анализ сетевого трафика, позволяя выявлять ошибки в передаче и подтверждать корректность формата сообщений. Использование анализатора пакетов является ключевым аспектом тестирования, т. к. он предоставляет возможность не только отслеживать успешные передачи, но и диагностировать проблемы, возникающие в процессе обмена данными. Например, анализатор может помочь в определении причин задержек в передаче, а также в выявлении несоответствий между ожидаемыми и фактическими данными. Таким образом, интеграция этих методов и инструментов в процесс тестирования драйвера способствует созданию надёжных и высокопроизводительных систем автоматизации, что имеет критическое значение для успешной реализации промышленных решений. Это, в свою очередь, обеспечивает уверенность в том, что разработанный драйвер будет эффективно работать в условиях реального времени, соответствуя современным требованиям к надёжности и производительности [11].

Рассмотрим интеграцию на базе методики нового протокола взаимодействия в промышленный контроллер. Для непосредственной реализации в рамках данной методики был выбран протокол MQTT версии 3.1.1.

Программируемый контроллер компании ООО «НПА Вира Реалтайм» серии «Сателлит Р» модели «ПР-10» (рис. 5) представляет собой контроллер, программируемый в среде разработки Codesys 3.5. На нём присутствует сетевая карта и реализовано на уровне операционной системы взаимодействие с использованием транспортного протокола передачи данных TCP. Технические характеристики данного устройства представлены в табл. 4.

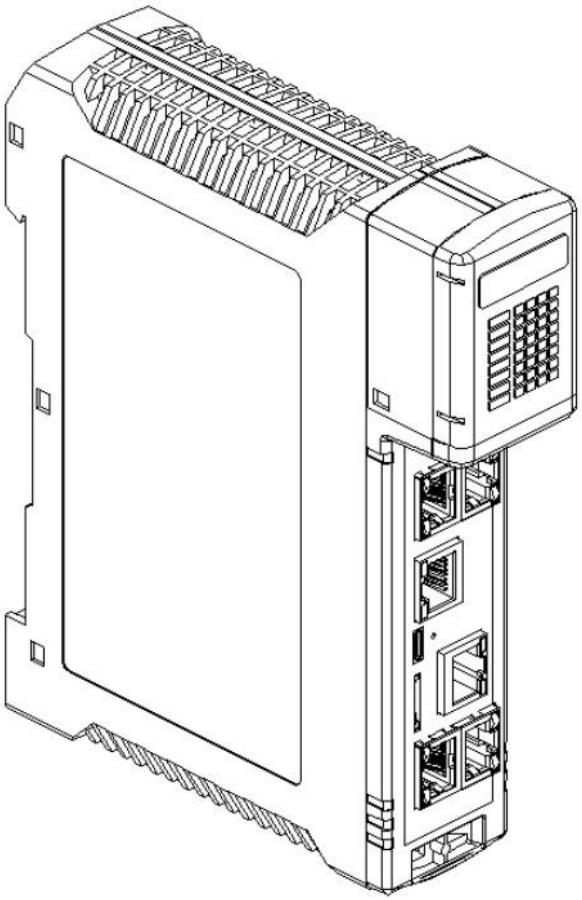


Рис. 5. Внешний вид контроллера Сателлит ПР-10.
Fig. 5. External view of the Satellit PR-10 controller.

Таблица 4. Технические характеристики Сателлит ПР-10
Table 4. Technical specification of the Satellit PR-10

Технические характеристики	Параметры
Операционная система	ОС РВ Keil RTX
Среда программирования	Программный пакет CODESYS (языки стандарта ГОСТ Р МЭК 61131-3-2016)
Тип микропроцессора	STM32F746IGT6 (ядро ARM Cortex-M7)
Тактовая частота	216 МГц
Системная память	FLASH: 1 Мбайт (встроенная в ARM) SRAM: 320 Кбайт (встроенная в ARM) FLASH Disk: microSD до 32 Гбайт SDRAM: 32 Мбайт FRAM: 16 Кбайт FLASH serial: 32 Мбайт
RTC — часы реального времени	Энергонезависимые (год, месяц, день, час, минута, секунда, миллисекунда)
Поддержка RTC	Литиевая батарея 3V (съёмная)
Последовательный порт 1	«S1»: RS232/RS485 до 115,2 кбит/с (RS485 гальваническая развязка 1500 В)
Последовательный порт 2	«S2»: RS232/RS485 до 115,2 кбит/с (RS485 гальваническая развязка 1500 В)
Порт Ethernet 1.1	«ETH1.1»: 10/100 Мбит/с
Порт Ethernet 1.2	«ETH1.2»: 10/100 Мбит/с
Порт Ethernet 2	«ETH2»: 10/100 Мбит/с

Технические характеристики	Параметры
Порт USB	«USB»: USB device port, Type micro-B
Типы протоколов	ГОСТ Р МЭК 60870-5-101 ГОСТ Р МЭК 60870-5-104 Modbus RTU, Modbus TCP SNTP, FTP, Telnet, DLC, NMEA 0183
Индикация	Светодиодная диагностика работы блока, контроллера, портов и прикладной программы
Рабочее напряжение	= 24 В (от блока шасси – 2 входа)
Потребляемая мощность	Не более 6,0 Вт (по цепи = 24 В)
Высота над уровнем моря	от - 400 м до + 4000 м
Относительная влажность	от 5% до 95% при 50 ºС (без конденсата)
Рабочая температура	от -40 °С до +70 °С
Температура хранения	от -55 °С до +85 °С
Габариты (Ш х В х Г)	40 х 180 х 145 мм
Вес	не более 0,5 кг

На этапе подготовки требуется произвести подготовительные действия перед началом интеграции протокола передачи данных в ПЛК. Следуя описанной методике, в первую очередь необходимо оценить возможность реализации драйвера протокола на выбранном ПЛК. Для стандарта MQTT от устройства требуется:

– сетевая карта с портами Ethernet либо с беспроводным каналом связи, на котором будет возможна передача данных по транспортному протоколу TCP;

– работа со строковыми типами данных.

Выбранный для ПЛК Сателлит ПР-10 удовлетворяет этим требованиям.

Для качественного и эффективного внедрения протокола передачи данных потребуются устройства с поддержкой данного стандарта или программные имитаторы протокола MQTT. В качестве имитатора клиента MQTT выбор был остановлен на приложении для ОС Windows MQTT.fx, а в качестве сервера–брокера MQTT mosquito для ОС Windows.

Проект, включающий в себя драйвер протокола передачи данных, реализуемый в Codesys 3.5, представляет собой внедряемую в проект приложения библиотеку. Структура библиотеки должна включать следующее.

1. Перечисления (ENUM) — структурные данные, представляющие собой структуру, в которой последовательно перечисляются состояния. Позволяют в удобном виде описать в строковом виде коды ошибок, этапы алгоритма и флаги.

2. Функциональный блок (FB) — для реализации алгоритма драйвера. Именно данный программный элемент будет в дальнейшем создан как экземпляр драйвера в приложения для ПЛК.

3. Глобальные переменные (константы) (GVL) — поля объявления переменного постоянного значения, необходимые для стандартизации часто используемых данных.

4. Структуры (STRUCT) — структуры данных, позволяющие объединить переменные в группы для множественного доступа.

Для удобной типизации кода потребуется сформировать типовые блоки программы. Такой блок должен быть одинаковым для частей программы, которые имеют одинаковые цели и результаты работы.

К таким блокам можно отнести последовательности действий, во время работы которых будет передаваться, приниматься и анализироваться пакет данных.

Блок, отвечающий за передачу пакета данных, имеет следующую структуру.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.

3. Указание времени для таймера тайм-аута данного этапа программы.
4. Вызов метода формирования пакета данных на отправку.
5. Передача сформированного пакета данных.

Далее следуют два условных ветвления.

Первое ветвление выполняется в случае удачной передачи пакета, последовательность действий имеет следующий вид.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Запись пакета данных в логгер.
4. Очищение буфера передачи.
5. Переход к следующей последовательности действий.

Второе ветвление выполняется в случае тайм-аута и имеет следующий вид.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Переход к шагу разрыва соединения.

Блок, отвечающий за получение пакета данных, имеет следующую структуру.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Указание времени для таймера тайм-аута данного этапа программы.
4. Ожидание получения пакета данных.

Далее следуют два условных ветвления.

Первое ветвление выполняется в случае удачного принятия пакета, последовательность действий имеет следующий вид.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Запись пакета данных в логгер.
4. Переход к следующей последовательности действий.

Второе ветвление выполняется в случае тайма-ута и имеет следующий вид.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Переход к шагу разрыва соединения.

Блок, отвечающий за анализ пакета данных, имеет следующую структуру.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Указание времени для таймера тайм-аута данного этапа программы.

Далее следует условное ветвление. В случае, если вызванный метод анализа пакета данных вернул положительный результат, то происходит следующее.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Переход к следующей последовательности действий.

В случае, если метод анализа пакета данных вернул отрицательный результат, то происходит следующее.

1. Формирование диагностического сообщения.
2. Запись диагностического сообщения в логгер.
3. Переход к шагу разрыва соединения.

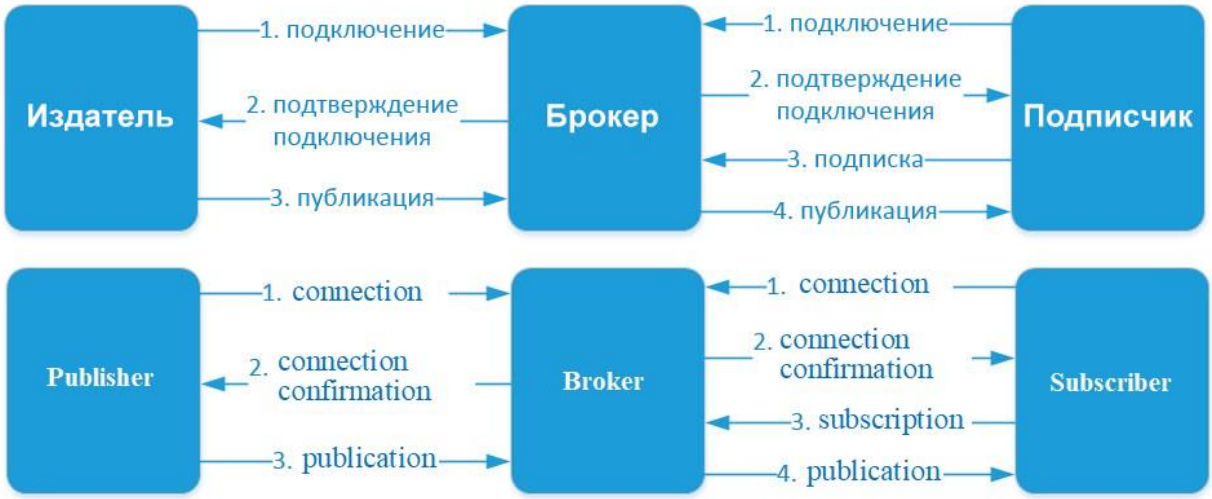
Реализация алгоритма протокола была произведена в среде программирования ПЛК Codesys 3.5 SP6 Patch 4, для промышленного контроллера Сателлит ПР-10 — в виде проекта библиотеки.

Проект представлен в виде проекта библиотеки для внедрения в проект приложения. Конструкция сетевого подключения предоставляется базовым транспортным протоколом

TCP, используемым MQTT. В стандарт протокола MQTT версии 3.1.1 входят два типа программных элементов (рис. 6):

– клиент, выполняющий роли издателя и\или подписчика;

– сервер, выполняющий роль сервера–брокера.



Издатель	Publisher
Брокер	Broker
Подписчик	Subscriber
Подключение	Connection
Подтверждение подключения	Connection confirmation
Подписка	Subscription
Публикация	Publication

Рис. 6. Схема простого взаимодействия между подписчиком, издателем и брокером.
Fig. 6. Diagram of simple interaction between a subscriber, a publisher and a broker.

Заключение

Взаимодействие контроллера с внешними устройствами позволяет реализовывать управление механизмами, анализ данных для последующей обработки информации, аккумулирование и обработку данных для передачи в системы верхнего уровня, такие как SCADA системы. Взаимодействие ПЛК с другими устройствами обеспечивается с помощью промышленных протоколов передачи данных. Промышленные протоколы передачи данных в свою очередь можно разделить на предметные области, в которых каждое решение имеет свои преимущества. Протоколы типа master–slave позволяют обеспечить беспрерывную связь между двумя устройствами в системе. Как пример, протокол Modbus [12].

В статье проведена разработка методики интеграции протокола передачи данных в контроллер. Описаны всевозможные варианты архитектур, по которым построены протоколы. Методика, включающая в себя четыре этапа разработки, описывает последовательность действий, после выполнения которых можно интегрировать новый протокол передачи данных в промышленный контроллер.

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

Вклад авторов. С.С. Гусев — поиск публикаций по теме статьи, написание текста рукописи; В.В. Макаров, А.Ю. Мещеряков — редактирование текста рукописи. Все авторы одобрили рукопись (версию для публикации), а также согласились нести ответственность за все аспекты работы, гарантируя надлежащее рассмотрение и решение вопросов, связанных с точностью и добросовестностью любой её части.

Этическая экспертиза. Неприменимо.

Источники финансирования. Отсутствуют.

Раскрытие интересов. Авторы заявляют об отсутствии отношений, деятельности и интересов за последние три года, связанных с третьими лицами (коммерческими и некоммерческими), интересы которых могут быть затронуты содержанием статьи.

Оригинальность. При создании настоящей работы были использованы фрагменты собственного текста, опубликованного ранее ([*doi: 10.17816/2074-0530-624783, doi: 10.17816/2074-0530-624784, doi: 10.17816/2074-0530-624786*], распространяется на условиях лицензии *CC-BY 4.1*).

Доступ к данным. Редакционная политика в отношении совместного использования данных к настоящей работе не применима, новые данные не собирали и не создавали.

Генеративный искусственный интеллект. При создании настоящей статьи технологии генеративного искусственного интеллекта не использовали.

Рассмотрение и рецензирование. Настоящая работа подана в журнал в инициативном порядке и рассмотрена по обычной процедуре. В рецензировании участвовали два внешних рецензента, член редакционной коллегии и научный редактор издания.

ADDITIONAL INFORMATION

Author contributions: S.S. Gusev: search for publications, writing the text of the manuscript; V.V. Makarov, A.Yu. Meshcheryakov: editing the text of the manuscript. All the authors approved the version of the manuscript to be published and agreed to be accountable for all aspects of the work, ensuring that issues related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.
Ethics approval: N/A.
Funding sources: No funding.
Disclosure of interests: The authors have no relationships, activities, or interests for the last three years related to for-profit or not-for-profit third parties whose interests may be affected by the content of the article.
Statement of originality: Parts of the authors' own text published earlier *([doi: 10.17816/2074-0530-624783, doi: 10.17816/2074-0530-624784, doi: 10.17816/2074-0530-624786]*, distributed according to the policy of the license CC-BY 4.1) were used during the work on the paper.
Data availability statement: The editorial policy regarding data sharing does not apply to this work as no new data was collected or created.
Generative AI: No generative artificial intelligence technologies were used to prepare this article.
Provenance and peer review: This paper was submitted unsolicited and reviewed following the standard procedure. The peer review involved two external reviewers, a member of the editorial board, and the in-house scientific editor.

СПИСОК ЛИТЕРАТУРЫ | REFERENCES

1. Sosonkin VL, Martinov GM. *Numerical control systems:* Textbook. Moscow: Logos; 2005. 296 p. (In Russ.) EDN: PJARBH
2. Sosonkin VL, Martinov GM. *Programming of numerical control systems*: Textbook. Moscow: Logos; 2008. 344 p. (In Russ.) EDN: PVWTGB
3. Martinov GM, Martinova LI, Pushkov RL. *Automation of technological processes in mechanical engineering. Part – I. Numerical control software*. A textbook on the training of specialists with higher professional education for personnel re-equipment of the machine-building complex of Russia. Moscow: MGTU STANKIN; 2010. 203 p. (In Russ.)
4. Martinov GM, Martinova LI, Pushkov RL. *Automation of technological processes in mechanical engineering.* Study guide. Moscow: MGTU «Stankin»; 2011. 200 p. (In Russ.)
5. Morozov VV, Grigor'ev SN, Skhirtladze AG, et al. *Milling tools: studies. Stipend***.** Vladim. gos. un-t im. A.G. i N.G. Stoletovyh. Vladimir: Izd-vo VlGU; 2014. 214 p. (In Russ.)
6. Barbashov FA. *Milling business.* A textbook for the medium. prof. tech. schools. 2nd ed. Moscow: «Vyssh. shkola»; 1975. 216 p. (In Russ.)
7. Kuvshinskij VV. *Mechanical Engineering.* Moscow: Mashinostroenie; 1977. 240 p. (In Russ.)

8. Martinov GM, Kozak NV, Nezhmetdinov RA, Lyubimov AB. The specifics of building control panels of CNC systems by the type of universal hardware and software components. *Avtomatizaciya i sovremennye tekhnologii*. 2010;(7):34–40. (In Russ.)
9. Nezhmetdinov RA, Sokolov SV, Obuhov AI, Grigor'ev AS. Expanding the functionality of CNC systems for controlling mechanical laser processing. *Avtomatizaciya v promyshlennosti*. 2011;(5):49–53. (In Russ.) EDN: OKCCRT.
10. Gusev SS, Makarov VV. Analysis of automatic design systems. *Izvestiya MGTU «MAMI»*. 2024;18(1):63–74. doi: 10.17816/2074-0530-624783 (In Russ.)
11. Gusev SS, Makarov VV. Analysis of existing tools for visualizing the trajectory of a cutting tool in CNC systems. *Izvestiya MGTU «MAMI»*. 2024;18(2):157–167. doi: 10.17816/2074-0530-624784 (In Russ.)
12. Gusev SS, Makarov VV. Development of postprocessors for CNC «Axiom Control». *Izvestiya MGTU «MAMI»*. 2024;18(3):169–179. doi: 10.17816/2074-0530-624786 (In Russ.)

ОБ АВТОРАХ / AUTHORS' INFO

* **Гусев Сергей Сергеевич,**
инженер-энергетик отдела энергетиков,
соискатель;
адрес: Россия, 123298, Москва, ул. 3-я Хорошевская, дом 17, корп. 1;
ORCID: 0000-0002-6070-9295;
eLibrary SPIN: 8934-1568;
e-mail: gs-serg@mail.ru

Соавторы:

Макаров Вадим Владимирович,
канд. техн. наук, доцент,
старший научный сотрудник лаборатории Идентификации систем управления № 41;
ORCID: 0000-0003-4874-5418;
eLibrary SPIN: 5787-3977;
e-mail: makfone@mail.ru

Мещеряков Александр Юрьевич,
канд. техн. наук, доцент,
старший научный сотрудник лаборатории Управления по неполным данным № 38;
ORCID: 0000-0001-9250-0130;
eLibrary SPIN: 6757-7244;
e-mail: aymesh@inbox.ru

*** Sergey S. Gusev,**
energy engineer of the Energy Department,
candidate for a degree;
address: 17 3rd Khoroshevskaya st, unit 1, Moscow, Russia, 123298;
ORCID: 0000-0002-6070-9295;
eLibrary SPIN: 8934-1568;
e-mail: gs-serg@mail.ru

Co-Authors:

Vadim V. Makarov,
Cand. Sci. (Engineering), Assistant Professor, senior researcher associate of the Identification of Control Systems Laboratory № 41;
ORCID: 0000-0003-4874-5418;
eLibrary SPIN: 5787-3977;
e-mail: makfone@mail.ru

Alexander Yu. Meshcheryakov,
Cand. Sci. (Engineering), Assistant Professor, senior researcher associate at the Incomplete Data Control Laboratory № 38;
ORCID: 0000-0001-9250-0130;
eLibrary SPIN: 6757-7244;
e-mail: aymesh@inbox.ru

* Автор, ответственный за переписку / Corresponding author