Методологические основы работы с протоколом Modbus TCP с примером на высокоуровневом языке программирования Python

П.Г. Фрасын, Н.В. Никитин, Д.В. Масанов, Е.А. Рыжкова

Российский государственный университет им. А.Н. Косыгина, г. Москва

Аннотация: В статье рассматривается вариант реализации систем сбора данных на базе универсального промышленного протокола Modbus TCP. В качестве сервера используется программа эмулятор протокола Modbus TCP. Для реализации клиента разработан алгоритм, на базе которого приведен пример реализации на высокоуровневом языке программирования Python с применением низкоуровневой библиотеки socket.

Ключевые слова: разработка, промышленный, протокол, высокоуровневый, язык, программирования, сервер, эмулятор, сбор, данные, python, сокет.

Одним из наиболее популярных протоколов передачи данных в промышленных системах является протокол Modbus, являющийся протоколом прикладного уровня (седьмой) модели OSI (модель взаимодействия открытых систем), которая не зависит от нижележащих уровней и может использоваться совместно с другими протоколами, такими как Ethernet TCP/IP или UDP/IP.

Modbus в качестве физической среды передачи сигналов использует последовательные интерфейсы RS-232, RS-422, RS-485, а также оптоволокно, радиоканалы (рис. 1).

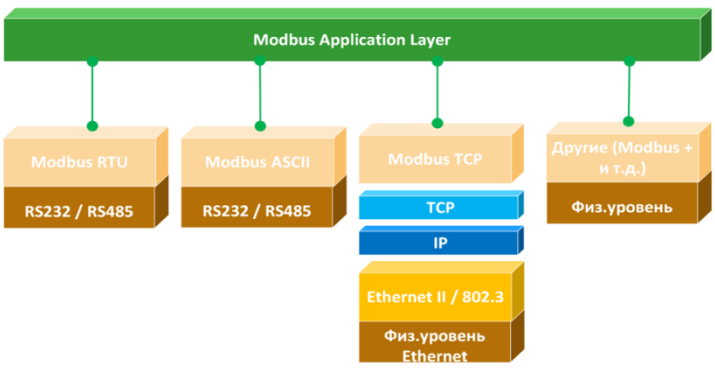

Рис. 1. Вариации исполнения протокола Modbus

Популярность Modbus связана с высокой надежностью передачи данных благодаря использованию надежного метода контроля ошибок, открытостью и относительно высокой скоростью обмена. Modbus позволяет унифицировать команды обмена за счет стандартизации номеров регистров (адресов) и их функций чтения и записи [1].

Открытость протокола Modbus позволяет разработчикам создавать собственные модификации протокола с дополнительными функциональными возможностями или расширенными командами для поддержки своих продуктов [2].

Стандартными реализациями протокола являются Modbus RTU, Modbus ASCII на базе последовательных интерфейсы для передачи данных, а также Modbus TCP, который является сетевым и работает поверх протокола TCP/IP.

В настоящее время идет активная интеграция промышленных протоколов, в том числе Modbus в системы сбора и обработки данных [10] которые впоследствии используются для систем мониторинга, диспетчеризации, а также для data science.

В классических системах автоматизации используются так называемые SCADA-системы, отвечающие за первичное получения данных с устройств, последующую обработку и вывод информации на экран оператора.

В качестве операционной системы для SCADA-систем чаще всего выступает операционная система Windows, что накладывает значимые ограничения. Также современные SCADA-системы начинают переходить на вариант оплаты лицензии по подписке, а зарубежные варианты являются недоступными на данный момент.

Возникает потребность в реализации программных комплексов, отвечающих за первичное получения данных с устройств, последующую

обработку и вывод информации на экран оператора на базе open-source решений.

Протокол Modbus TCP состоит из части сообщения Modbus RTU и специального заголовка. В заголовке – адрес устройства SlaveID; функциональный код, присущий региону группы регистров; адрес первого регистра; количество регистров (для группового пакетного опроса); контрольная хеш-сумма.

Для получения Modbus TCP из сообщения Modbus RTU адрес SlaveID в начале и контрольная сумма CRC в конце удаляются, в следствии чего создается PDU (блок данных протокола). В сообщение PDU добавляется новый 7-байтовый заголовок, называемый MBAP (заголовок приложения Modbus) [3]. Это преобразование представлено на рисунке 2.

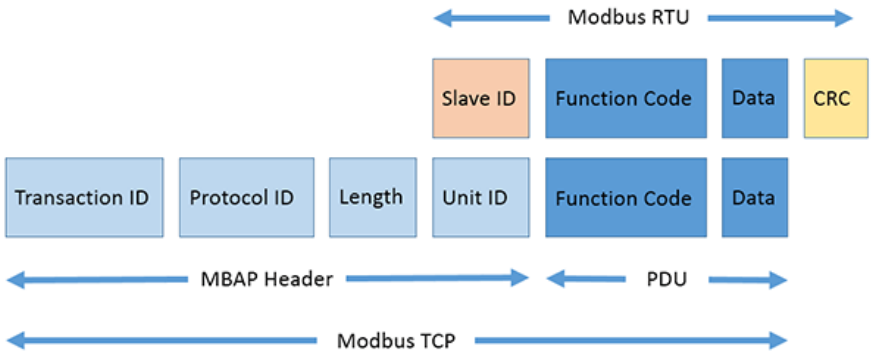

Рис. 2. Структурное сопоставление Modbus RTU и Modbus TCP

Заголовок приложения Modbus (MBAP) заключает в себе Transaction ID, Protocol ID, Length и Unit ID.

На рисунке 3 представлен перехваченный сетевой пакет трафика с использованием программы Wireshark, в которой производился запрос данных от устройства по протоколу Modbus TCP [4].

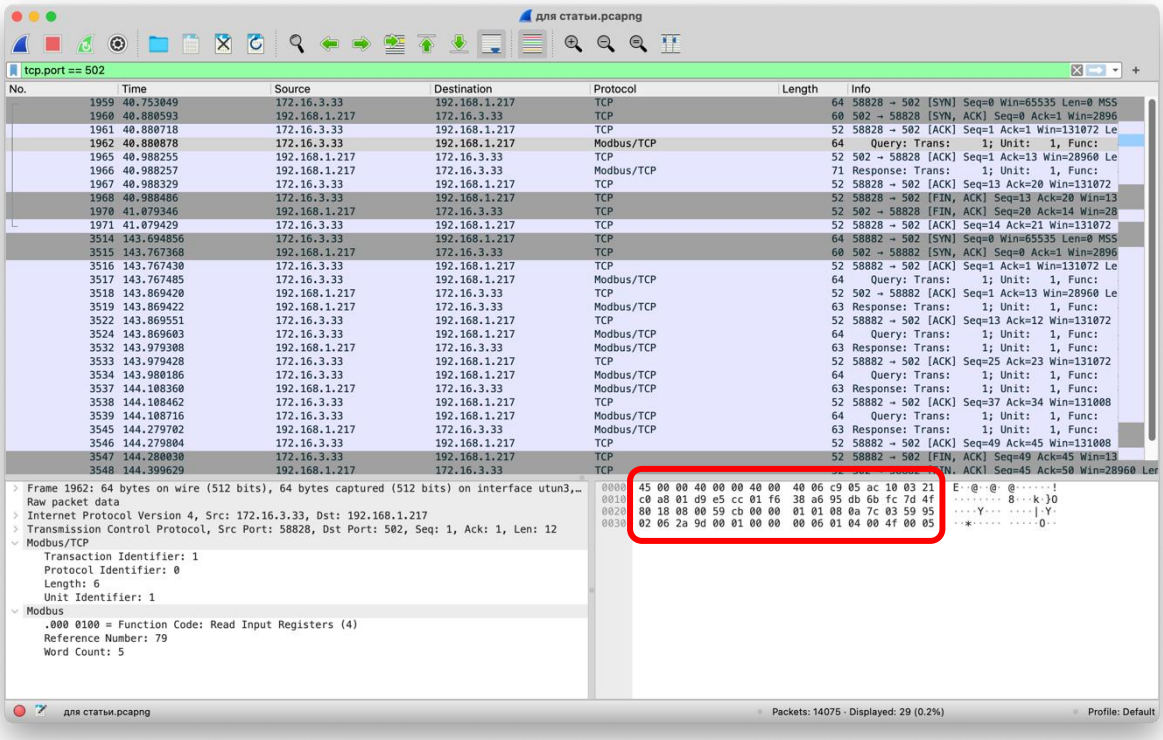

Рис. 3. Перехваченный сетевой трафик в программе Wireshark

Рассмотрим выделенный фрагмент на рисунке 3. В первой части сообщения (45 00 00 40 00 00 40 00 40 06 c9 05 ac 10 03 21 c0 a8 01 d9) содержится информация об протоколе (Internet Protocol Version 4), информации об адресе отправителя (Src: 172.16.3.33) и получателя (Dst: 192.168.1.217).

Во второй части сообщения (e5 cc 01 f6 38 a6 95 db 6b fc 7d 4f 80 18 08 00 59 cb 00 00 01 01 08 0a 7c 03 59 95 02 06 2a 9d) содержится информация о портах подключения (порт источника Src Port: 58828), порте получателя (Dst Port: 502), номере последовательности (Sequence Number: 1 (relative sequence number)), подтверждение получения данных (Acknowledgment Number: 1 (relative ack number)) и длиной сообщения (Len: 12).

В последней группе байт находится содержимое самого непосредственного Modbus TCP, состоящего из **MBAP** (00 01 00 00 00 06 01) и **PDU** (04 00 4f 00 05)

MBAP состоит из Transaction Identifier: 1 Protocol Identifier: 0 Length: 6 Unit Identifier: 1.

PDU состоит из функционального кода опроса (.000 0100 = Function Code: Read Input Registers (4)), стартового адреса регистра (Reference Number: 79), и количества регистров, следующих после стартового (Word Count: 5)

В эмуляторе Modbus протокола созданы некоторые переменные, содержащие данные и имеющие адреса 79, 80, 81, 82, 83 региона Input Registers.

На рисунке 4 представлен сетевой трафик, содержащий ответ от эмулятора Modbus устройства [5].

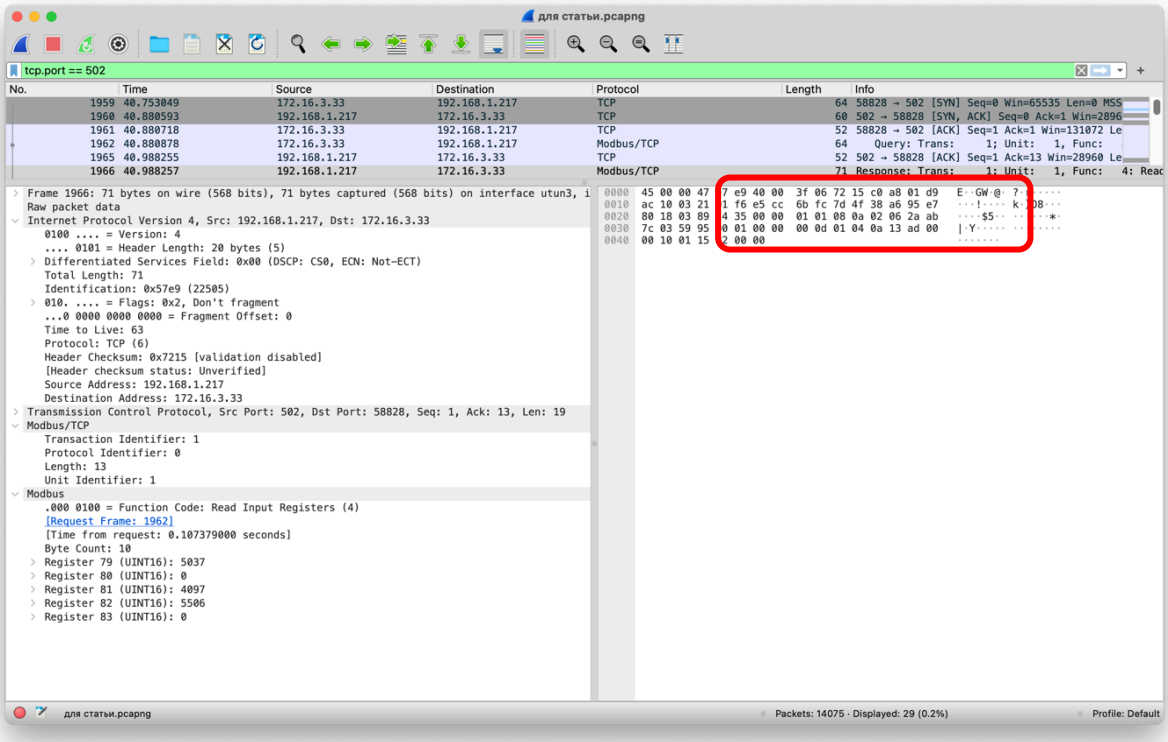

Рис. 4. Перехваченный сетевой трафик от Modbus устройства

При запросе на получение данных, байты отвечающие за заголовки транспортного протокола адрес отправителя и получателя поменяются местами.

Код MBAP остается практически прежнем, кроме длины Modbus слова. В запросе значение было равно 6, а в ответе 13. Существенные изменения произойдут в лишь блоке байтов, отвечающих за PDU (04 0a 13 ad 00 00 10 01 15 82 00 00). Функциональный код опроса не меняется (.000 0100 = Function Code: Read Input Registers (4)), количество байт равняется 10 (Byte Count: 10). Дальше следуют результаты опроса пяти регистров:

1) Регистр 79 (UINT16): 5037
2) Регистр 80 (UINT16): 0
3) Регистр 81 (UINT16): 4097
4) Регистр 82 (UINT16): 5506
5) Регистр 83 (UINT16): 0

В примере показаны результаты мультипакетного опроса Modbus устройства. При опросе одного регистра, в качестве результата будет получено только одно значение.

Для создания программного комплекса, позволяющего работать с протоколом Modbus TCP на высокоуровневых языках программирования необходимо указать принцип работы с сетевым сокетом.

Сокеты представляют собой механизм взаимодействия между устройствами через сеть, который используется для передачи данных.

TCP (Transmission Control Protocol) является важной частью сетевого взаимодействия, позволяющей двум устройствам устанавливать надежное, двустороннее соединение через сеть. TCP-сокеты обеспечивают ориентированный на поток канал связи, подходящий для протоколов, таких как HTTP, FTP, SSH и многих других, где важна целостность и надежность данных.

В языках программирования используются различные библиотеки (например, модуль socket в Python или класс Socket в Java) для создания и управления TCP-сокетами.

Для работы с протоколом Modbus TCP с использованием сетевого сокета необходимо осуществить несколько действий [6].

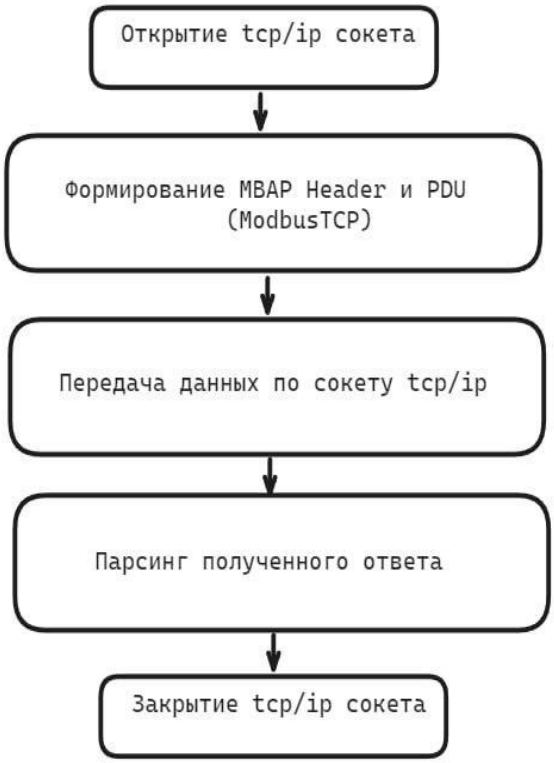

Рис. 5. Блок-схема с алгоритмом работы

Первым этапом является инициализация открытие сетевого сокета по адресу устройства, с которым будет осуществляться дальнейшее взаимодействие. Далее необходимо сформировать структуру modbus-tcp протокола, состоящую из MBAP и PDU. В зависимости от чтения или записи, MBAP соответственно будет изменятся [7].

Пакет отправляется через сокет на устройство, к которому было произведено подключение. Результат опроса устройства также читается через

подключенный сокет. После получения результата, работа с сокетом в текущей сессии завершается и сокет необходимо закрыть.

При операции чтения – данные от опрошенных регистров доступны для последующей обработки. При записи данных в указанные регистры – необходимо проверить, действительно ли была произведена запись. Проверка осуществляется через чтение.

В языке программирования Python для работы с сокетом удобно использовать низкоуровневую библиотеку socket. Для создания структуры Modbus TCP используется системная библиотека struct [8].

Следующем этапом является инициализация функции чтения регистров Modbus в программе.

Инициализируется функция MODBUS_TCP_client_read. В качестве аргументов указываются данные ip-адреса Modbus устройства; порт, через который будет осуществляться обмен данными по протоколу Modbus TCP (зависит от устройства, стандартным является 502); адрес Modbus устройства (он же unit id); функция (регион) группы регистров (3 – holding registers, 4 – input registers и т.д.); а также один или несколько регистров, которые будут опрашиваться в программе [9].

Функция записи будет иметь аналогичные аргументы, но в качестве дополнения будет содержать записываемую информацию.

Программный код описания функции чтения представлен на рисунке 6.

```
def MODBUS_TCP_client_read(IP_address: str, TCP_port: int, MODBUS_address: int, MODBUS_function: int, Register_address) -> object:
    """
    Входные данные: ip_адрес, tcp порт, адрес устройства, регион группы регистров, список опрашиваемых регистров
    :param IP_address:
    :param TCP_port:
    :param MODBUS_address:
    :param MODBUS_function:
    :param Register_address:
    :return:
    """
```

Рис. 6. Описание функции чтения

```
try:
    # создание элемента класса сокета
    Client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # осуществление подключения
    Client_socket.connect((IP_address, TCP_port))
    reg_adr_list = []
    for reg_address in Register_address:
        # ID транзакции
        Tx_Transaction_ID = 1
        # ID протокола (0 - по умолчанию для Modbus TCP)
        Tx_Protocol_ID = 0
        # Длина запроса
        Tx_Message_length = 6
        # unit_id устройства
        Tx_MODBUS_address = MODBUS_address
        # Группа регистров (3 - группа Holding Registers)
        Tx_MODBUS_function = MODBUS_function
        # Один или несколько регистров для опроса
        Tx_Register_address = reg_address
        # Кол-во регистров. Для пакетного опроса последовательных регистров указывается количество.
        Tx_Register_count = 1
        # Составляем структуру MBAP и PDU
        Tx_ADU = struct.pack( fmt: ">HHHBBHH",  *v: Tx_Transaction_ID, Tx_Protocol_ID, Tx_Message_length, Tx_MODBUS_address,
                             Tx_MODBUS_function, Tx_Register_address, Tx_Register_count)
        # Отправка данных
        Client_socket.send(Tx_ADU)
        # Получение данных после отправки запроса
        Rx_ADU = Client_socket.recv(1500)
        # Декодирование данных
        (Rx_Transaction_ID, Rx_Protocol_ID, Rx_Message_length, Rx_MODBUS_address, Rx_MODBUS_function, Rx_Byte_count,
         Rx_Register_value) = struct.unpack( _format: ">HHHBBBH", Rx_ADU)
        reg_adr_list.append(Rx_Register_value)
    # закрытие сокета и клиента
    Client_socket.close()
    Client_socket.__del__()
    # Вывод значений опрошенных регистров
    return reg_adr_list
```

Рис. 8. Программный код чтения регистров

В статье представлены алгоритм и решения чтения регистров с промышленного протокола Modbus TCP с использованием open-source технологий, позволяющий в дальнейшем масштабировать данное решение в систему сбора и обработки данных под различные задачи [10].

Удобно в дальнейшем для вывода данных также использовать open-source технологии html5 и веб-сервера [11].

Литература

1. Гришков Д.Ю., Аусилова Н.М. Язык высокого уровня программирования Python // Наука и реальность/Science & Reality. - 2022. - №1. - С. 115-117.
2. What is Modbus and How does it work? // Schneider Electric Life is On URL: se.com/us/en/faqs/FA168406/ (дата обращения: 26.09.2023).
3. Как общаются машины: протокол Modbus // Хабр: сайт. – URL: habr.com/ru/companies/advantech/articles/450234/ (дата обращения: 26.09.2023).
4. Савостин П.А., Ефремова Н.Э. Практическое применение асинхронного программирования на языке Python при помощи пакета Asyncio // Программные системы и вычислительные методы. - 2018. - №2. - С. 11-16.
5. Welcome to pyModbusTCP’s documentation // pyModbusTCP URL: pymodbustcp.readthedocs.io/en/latest/ (дата обращения: 20.10.2023).
6. struct — Interpret bytes as packed binary data // Docs Python URL: docs.python.org/3/library/struct.html (дата обращения: 26.09.2023).
7. Сафаров И.М., Богданова Н.В., Латыпов Т.И. Комплексный критерий оценки эффективности программируемых логических контроллеров // Инженерный вестник Дона, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8706
8. Сокеты в Python для начинающих // Хабр. URL: habr.com/ru/articles/149077/ (дата обращения: 26.09.2023).
9. Рыжкова Е.А., Постолаки Е.С., Комбаров Ю.С. Анализ работы программы установки периодического действия для жидкостной обработки ткани с последующей сушкой, написанной на языке LD // Инженерный вестник Дона, 2022, №6. URL: ivdon.ru/ru/magazine/archive/n6y2022/7740

10. Фрасын П.Г., Масанов Д.В., Рыжкова Е.А., Автоматизированная оценка возможности получения данных с промышленного оборудования, их идентификация и параметрический анализ // Сборник материалов Всероссийской научной конференции молодых исследователей с международным участием. - Москва: Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский государственный университет имени А.Н. Косыгина (Технологии. Дизайн. Искусство)», 2023. - С. 270-274.
11. Тимохин М.Ю, Шарани В.Ю. Искусственный интеллект и теория принятия решений: современные тенденции // Инженерный вестник Дона, 2023, №10. URL: ivdon.ru/ru/magazine/archive/n10y2023/8746

References

1. Grishkov D.Yu., Ausilova N.M. Python high-level programming language, Nauka i real`nost`. 2022. №1. pp. 115-117.
2. What is Modbus and How does it work? Schneider Electric Life is On URL: se.com/us/en/faqs/FA168406/ (date access: 26.09.2023).
3. Kak obshhayutsya mashiny`: protokol Modbus [How machines communicate: Modbus protocol]. Habr: sajt. URL: habr.com/ru/companies/advantech/articles/450234/ (date access: 26.09.2023).
4. Savostin P.A., Efremova N.E`. Programmny`e sistemy` i vy`chislitel`ny`e metody`. 2018. №2. pp. 11-16.
5. Welcome to pyModbusTCP’s documentation, pyModbusTCP. URL: pymodbustcp.readthedocs.io/en/latest/ (data obrashheniya: 20.10.2023).
6. struct. Interpret bytes as packed binary data. Docs Python. URL: docs.python.org/3/library/struct.html (date access: 26.09.2023).

7. Safarov I.M., Bogdanova N.V., Laty`pov T.I. Inzhenernyj vestnik Dona, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8706
8. Sokety` v Python dlya nachinayushhix. [Sockets in Python for Beginners]. Habr. URL: habr.com/ru/articles/149077/ (date access: 26.09.2023).
9. Ry`zhkova E.A., Postolaki E.S., Kombarov Yu.S. Inzhenernyj vestnik Dona, 2022, №6. URL: ivdon.ru/ru/magazine/archive/n6y2022/7740
10. Frasy`n P.G., Masanov D.V., Ry`zhkova E.A. Sbornik materialov Vserossijskoj nauchnoj konferencii molody`x issledovatelej s mezhdunarodny`m uchastiem. Moskva: Federal`noe gosudarstvennoe byudzhetnoe obrazovatel`noe uchrezhdenie vy`sshego obrazovaniya «Rossijskij gosudarstvenny`j universitet imeni A.N. Kosy`gina (Texnologii. Dizajn. Iskusstvo)», 2023. pp. 270-274.
11. Timoxin M.Yu, Sharani V.Yu. Inzhenernyj vestnik Dona, 2023, №10. URL: ivdon.ru/ru/magazine/archive/n10y2023/8746