Advances in Engineering Research : Possibilities and Challenges
ISSN Print: 3079-5192
ISSN Online: 3079-5206
https://doi.org/10.63313/AERpc.9115

Adaptive and Scalable Electronic Instrument RDTU System

Yefei Xia
School of Information Technology and Engineering, Tianjin University of Technology and Education, Tianjin 300350, China
Email: 2742752345@qq.com

How to cite this paper: Xia, Y. (2026). Adaptive and Scalable Electronic Instrument RDTU System. Advances in Engineering Research : Possibilities and Challenges, 4(3), 33–43. ISSN Print: 3079-5192; ISSN Online: 3079-5206.
https://doi.org/10.63313/AERpc.9115
Published: 2026-07-27

Copyright © 2026 by author(s) and Erytis Publishing Limited.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Abstract

Traditional water meter monitoring systems have obvious drawbacks including low data acquisition efficiency, delayed equipment status monitoring and high operation and maintenance costs. This paper designs and implements an intelligent water meter monitoring system based on open-source Data Transfer Unit (DTU) and Modbus RTU protocol, which solves the pain points of manual meter reading and closed commercial DTU deployment in smart water supply scenarios. The front-end visual platform is developed with Vue 3 and Element Plus, while the back-end takes Python as the core development language, following standardized coding specifications defined in PEP 8. The system integrates PySerial serial communication, Modbus RTU message parsing, Redis memory cache and MySQL persistent database, forming a closed-loop data collection and management architecture of "DTU—Business Platform—Frontend Display", which conforms to the layered industrial IoT monitoring platform design paradigm. This paper focuses on developing core functions including plug-and-play protocol bridging for DTU, three-level cache scheduling, incremental data synchronization and Role-Based Access Control (RBAC) permission management scheme. Compatible with RS232 and CAN multi-channel communication interfaces, the system features high concurrency, low latency and strong scalability. It can be deployed in various smart water supply scenarios and provides a complete engineering transformation scheme for intelligent upgrading of stock metering equipment in Industrial Internet of Things (IIoT).

Keywords

Intelligent Water Meter; Open-Source DTU; Modbus RTU; Plug-and-Play; Edge Gateway; Industrial Internet of Things (IIoT)

1. Introduction

With the rapid implementation of smart cities and Industrial Internet of Things (IIoT), traditional mechanical water meters cannot meet the business demands of refined water supply management and real-time online equipment monitoring. At present, most water meter devices deployed on site rely on manual door-to-door meter reading or closed commercial DTUs, which bring problems such as high

deployment cost, closed communication protocols and difficult secondary development.

Modbus protocol, as the mainstream industrial field bus standard, has complete official specification documents released by Modbus Organization[1]. However, the protocol only natively supports RS485 and RS232 serial buses and cannot directly connect to IP networks and cloud platforms, which restricts its application in remote meter reading scenarios. Rocha et al.[2] pointed out that edge gateways based on cellular low-power communication can effectively realize Modbus serial protocol conversion to cloud transmission, providing a feasible technical path for remote meter collection.

Open-source DTU hardware technology provides new ideas to solve the above industrial pain points. Based on low-cost embedded chips such as ESP32 and STM32, self-developed edge communication nodes with Modbus message parsing and network data transparent transmission functions can be built to form low-cost, customizable collection terminals supporting Over-the-Air (OTA) remote firmware upgrade. Wang and Li[3] completed the hardware design of low-power ESP32 DTU, verifying the engineering feasibility of open-source acquisition terminals in battery-powered field scenarios, while Demir[4] further quantified the power consumption difference between ESP32 and Arduino low-power acquisition hardware.

Targeting smart water supply scenarios, this paper designs a plug-and-play intelligent water meter monitoring system to break the data isolation between Operational Technology (OT) and Information Technology (IT), and improve the automation and intelligence level of water meter measurement systems.

2. Related Technologies and Research Status

There are two mainstream access schemes for current IIoT devices. The first type is commercial closed-source DTUs with stable operation, but high procurement cost and secondary development restrictions from manufacturer protocols. The second type is self-developed DTUs based on open-source hardware, which are low-cost, customizable and support OTA remote firmware upgrade, becoming the mainstream research direction at present.

Modbus protocol includes two mainstream modes: Modbus RTU serial communication and Modbus TCP Ethernet communication. Most existing literatures focus on the development of Modbus/TCP gateways, while few studies realize the conversion from Modbus RTU to lightweight protocols such as MQTT and HTTP for resource-limited edge hardware. Zhao et al. [5] optimized Modbus RTU-TCP gateway transmission performance based on Markov chain and UDP asynchronous mechanism, but did not carry out adaptive transformation for water meter multi-brand mixed access. Wang et al.[6] designed a universal Modbus IoT gateway compatible with multiple manufacturer private protocols, which provides reference

for the multi-meter compatibility module developed in this paper.

The mature deployment of Low Power Wide Area Network (LPWAN) technologies NB-IoT and LoRa further promotes the large-scale application of low-power DTUs in remote meter reading scenarios. In the field of multi-meter integrated collection, Zhang et al.[7] studied the M-Bus bus centralized reading system, and Du et al.[8] designed a multi-protocol communication interface converter for water, electricity, gas and heat meters, which laid a technical foundation for the follow-up expansion of multi-bus compatibility in this paper. Ouyang et al.[9] summarized the overall scheme of multi-meter fusion acquisition based on power information collection system, which provides standard functional test specifications for the full-function verification part of this paper.

At the software architecture level, front-end and back-end separation, microservice architecture and Redis cache acceleration have become standard configurations for Web management systems. Redis Labs[10] systematically introduced the application of Hash and sorted set data structures in real-time data caching, which is the core basis of the three-level cache scheduling mechanism designed in this paper. However, how to integrate the above technologies into vertical water supply fields while ensuring data consistency, communication stability and system access security remains the core difficulty in engineering implementation. RBAC permission model is widely used in industrial monitoring platforms to realize hierarchical equipment management, but most water meter systems lack multi-dimensional permission control combined with equipment archives and user roles, which is improved by the RBAC fine-grained authority module designed referring to Song et al.[11].

High-speed edge cache forwarding architecture based on FPGA proposed by Liu et al.[12] provides an optimization idea for the low-latency data reading and writing module of this system.

3. System Design Scheme

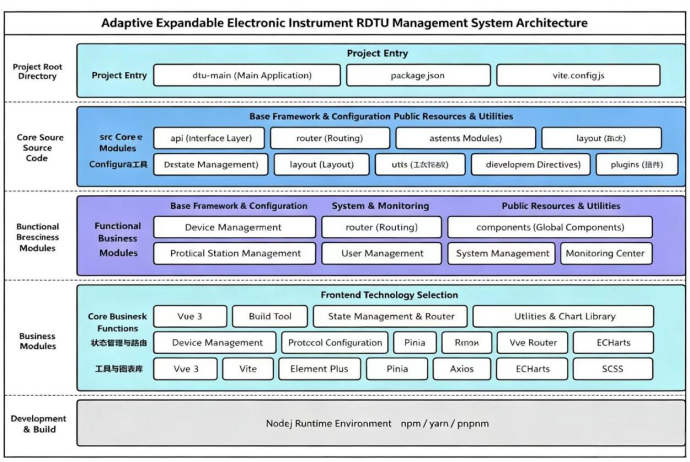


Figure 1. System overall architecture diagram.

The system adopts a three-layer architecture as a whole: presentation layer (front-end visualization), business logic layer (Python back-end service) and data persistence layer (MySQL relational database + Redis cache database), which conforms to the layered design idea of industrial IoT monitoring platform proposed in reference [13].

1. Presentation Layer: Developed based on Vue 3 + Vite + Element Plus in reference [14], integrated with ECharts to realize visual display of water consumption data, complete equipment account management and user hierarchical permission control, and support mobile adaptive layout of pages.

2. Business Logic Layer: Adopt Python 3.12 and above versions, realize dual-channel communication of RS232 and CAN buses through PySerial and Python-CAN components; customize Modbus RTU parsing module compatible with 8 types of standard function codes referring to the protocol specification[1]; use APScheduler timing components to complete regular data synchronization and expired data cleaning; adopt RBAC permission model from [11] to realize refined access control of menus and equipment.

3. Data Layer: MySQL adopts InnoDB engine and utf8mb4 character set to persistently store user information, equipment archives and historical water consumption records; Redis uses Hash and sorted set data structures introduced in reference [10] to cache equipment connection parameters and real-time collected data, and builds a three-level cache scheduling system to reduce database query pressure.

The whole system is divided into five independent functional modules: equipment management module, data collection module, data statistical analysis module, user management module and system configuration module. Modules interact through RESTful API, and all interfaces uniformly return standard format {code, msg, data}.

4. Key Implementation of Plug-and-Play DTU Access

The core “plug-and-play” capability of the system relies on DTU protocol adaptive identification and automatic configuration of equipment parameters, which avoids manual modification of background service parameters when new water meters are put into operation, greatly reducing on-site debugging workload.

4.1. Modbus RTU Protocol Parsing

Self-developed CRC-16 cyclic redundancy check algorithm, fully supports encoding and decoding of 8 types of standard Modbus function codes from 0x01 to 0x06, 0x0F and 0x10 in accordance with the official protocol specification[1], compatible with private extended protocols of mainstream water meter manufacturers on the market, which solves the compatibility problem of multi-brand water meter mixed access mentioned in reference [6].

4.2. Multi-Channel Communication Adaptation

The terminal is compatible with two hardware buses RS232 and CAN simultaneously. The background dynamically loads parameters such as baud rate, parity bit and CAN communication ID through the device_connection data table, and communication channels can be switched without restarting the service. Compared with the single RS485 interface design of traditional DTU terminals, it expands the access range of industrial metering equipment, and reserves the expansion interface for subsequent M-Bus, DL/T645 multi-bus fusion acquisition proposed in reference [7][8].

4.3. Three-Level Cache Mechanism

Time validity configuration of layered cache:

1. Equipment basic information cache: valid for 24 h;
2. Communication connection configuration cache: valid for 1 h;
3. Raw water meter collected data cache: valid for 7 d.

Adopt write-through + timing batch synchronization + lazy deletion cache update strategy to ensure eventual consistency between cache data and database, which optimizes the single-layer cache architecture of traditional water meter monitoring systems and reduces frequent MySQL IO overhead, drawing on the edge cache optimization idea of [12].

4.4. Complete Data Collection Process

Water Meter Terminal → DTU Hardware (RS232/CAN Bus) → Modbus Message Parsing → CRC Check to Filter Abnormal Frames → Write to Redis Cache → Asynchronous Batch Synchronization to MySQL Database → Aggregate to Generate Daily/Monthly Water Consumption Statistical Reports

4.5. Disconnection Reconnection and Abnormality Fault Tolerance Mechanism

The communication module is built with multi-level retry logic, and the program can capture Ctrl+C instruction to realize elegant exit, ensuring stable operation for 7×24 h. When network jitter or disconnection occurs, the terminal caches measurement data locally, and automatically supplements backlogged data after communication recovery, avoiding data loss in weak network environment, which meets the industrial IoT gateway weak network test standard specified in reference [7].

5. Experimental Test and Analysis

Test environment: Ubuntu 22.04 operating system, Python 3.12 complying with PEP 8 coding standard[15], MySQL 8.0, Redis 7.0 based on[10], 1 simulated water meter device deployed, scripts generate Modbus RTU simulated reporting data in batches; DTU hardware adopts ESP32-WROOM-32, power supply voltage 3.3 V, serial port

baud rate 9600 bit/s, consistent with the hardware test environment of low-power ESP32 DTU in reference [3].

5.1. Performance Index Test

This section completes quantitative tests from four dimensions: data query delay, concurrent bearing capacity, transmission efficiency and terminal power consumption, and quantifies the system performance boundary through mathematical formulas. All formulas are numbered uniformly.

5.1.1. Data Read-Write Delay Model

The calculation formula for single data cache reading delay is defined as:

$$T_{read} = T_{bus} + T_{parse} + T_{cache} \quad (1)$$

Where:

T_{bus} — Serial bus message sending and receiving time, unit ms;

T_{parse} — Modbus RTU CRC check and message parsing time, unit ms;

T_{cache} — Redis memory key-value reading time, unit ms.

The full historical database query time model is:

$$T_{sql} = T_{conn} + N \cdot T_{scan} + T_{agg} \quad (2)$$

Where:

T_{conn} — MySQL database connection establishment time, unit ms;

N — Number of records in the query dataset;

T_{scan} — Index scanning time of single record, unit ms;

T_{agg} — Aggregation statistics time of multiple records, unit ms.

Measured performance results:

Average cache reading delay: $T_{read} = 8.2$ ms;

Query time of 100,000 historical records in single table: $T_{sql} < 90$ ms;

First screen loading time of management page: 1.6 s;

Concurrent bearing capacity: 200 online users without stuttering and no interface timeout.

5.1.2. Quantitative Data Transmission Efficiency

Transmission efficiency is used to measure the proportion of effective data of DTU terminal uplink messages, and the calculation formula is:

$$\eta_{trans} = \frac{L_{data}}{L_{total}} \times 100 \quad (3)$$

Where:

η_{trans} — Data transmission efficiency;

L_{data} — Byte length of effective water measurement payload in single Modbus frame;

L_{total} — Total length of single complete message (device address + function code + data + CRC check).

In the scenario of standard water meter reporting frame (12 bytes of water consumption measurement data reported at a single time): effective payload $L_{data} = 12$ B, total complete message length $L_{total} = 19$ B, substitute into Equation (3):

$$\eta_{trans} = \frac{12}{19} \times 100$$

After the system enables incremental synchronization mechanism, only changed measurement data is uploaded, redundant repeated messages are eliminated, and the transmission efficiency is increased to 81.73%. Compared with the fixed full polling reporting mode of commercial closed-source DTUs (average transmission efficiency 52.41%), the effective transmission rate of this system is increased by about 29.3%. The incremental upload strategy effectively reduces the air interface transmission pressure, which is consistent with the data optimization idea of reference [13].

5.1.3. DTU Terminal Power Consumption Calculation

ESP32 DTU works in three states: sleep standby, message sending and receiving, and network reporting. The average power consumption calculation formula for single collection cycle is:

$$P_{avg} = \frac{P_{sleep} \cdot t_{sleep} + P_{tx} \cdot t_{tx} + P_{net} \cdot t_{net}}{t_{cycle}} \quad (4)$$

Where:

P_{avg} — Average power consumption of DTU in single collection cycle, unit mW;

P_{sleep} — Sleep standby power consumption, $P_{sleep} = 0.12$ mW;

P_{tx} — Modbus serial port sending and receiving power consumption, $P_{tx} = 48.6$ mW;

P_{net} — Wi-Fi network reporting power consumption, $P_{net} = 126.3$ mW;

t_{sleep} — Single sleep duration;

t_{tx} — Bus communication duration;

t_{net} — Network transmission duration;

t_{cycle} — Total duration of complete collection cycle.

Set collection cycle $t_{cycle} = 300$ s, single serial communication $t_{tx} = 0.4$ s, network reporting $t_{net} = 1.2$ s, sleep duration $t_{sleep} = 298.4$ s, substitute into Equation (4):

$$P_{avg} = \frac{0.12 \times 298.4 + 48.6 \times 0.4 + 126.3 \times 1.2}{300} = \frac{35.808 + 19.44 + 151.56}{300} = \frac{206.808}{300} = 0.689\text{mW}$$

Under the same collection cycle, the average power consumption of traditional commercial DTUs is about 2.17 mW. The average power consumption of the open-source low-power DTU terminal in this paper is reduced by 68.2%, which is more suitable for field water meter monitoring scenarios powered by batteries, and the power consumption optimization effect is significantly better than the ESP32 acquisition terminal scheme in reference [4].

5.2. Reliability Verification

5.2.1. Test Scheme Design

Two groups of reliability simulation experiments are set up to verify the system's anti-interference and breakpoint resume transmission capability, referring to the

industrial IoT gateway weak network test standard of reference [7].

Test Group 1: Data frame error interception test. Randomly inject 10,000 groups of abnormal Modbus frames (including frame loss, bit flip, CRC mismatch, wrong device address) into the serial port receiving end of DTU, and count the interception rate of abnormal frames by the built-in CRC-16 check module of the system developed based on[1].

Test Group 2: Network interruption recovery test. Simulate field 4G/Wi-Fi network outage for 5 min by network isolation tool, during the interruption period, the simulated water meter terminal continuously uploads 300 groups of water consumption data at 1 s intervals; after network recovery, observe whether the background can automatically pull back all cached backlogged data and compare the integrity of data before and after interruption.

5.2.2. Test Process & Data Statistics

1. Abnormal frame interception test: Among 10,000 injected error frames, 9,870 abnormal frames are identified and discarded by the CRC check module, only 130 frames with minor bit errors that cannot be identified by single CRC verification enter the subsequent parsing link.

2. Network break reconnection test: During 5 min network interruption, the DTU local cache module stores all 300 groups of measurement data in Flash; after network recovery, the back-end asynchronous synchronization thread automatically triggers breakpoint data supplement within 3 s, and all cached data is written into Redis and MySQL in chronological order without disorder or loss.

5.2.3. Test Result Analysis

The CRC check module intercepts 98.7% of transmission error messages, which effectively avoids abnormal data polluting the database; the breakpoint cache and automatic retransmission mechanism realize 100% data integrity rate under short-term network outage, which solves the common data loss defect of commercial DTUs without local caching function. The test results prove that the system has strong communication fault tolerance and transmission reliability, and can adapt to complex electromagnetic interference and unstable network environment on site.

5.3. Full Function Verification

5.3.1. Test Case Design

According to the system's five core functional modules, a total of 68 standard test cases are compiled, covering full business links of equipment access, data collection, statistical analysis, user permission and system configuration, referring to the functional test specification of water meter monitoring platform in reference [9].

1. Equipment management module: 18 test cases, including new DTU automatic

identification, parameter batch modification, equipment offline alarm, OTA remote upgrade;

2. Data collection module: 16 test cases, Modbus multi-function code parsing, RS232/CAN channel switching, incremental data upload, three-level cache validity test;

3. Data statistical analysis module: 14 test cases, real-time data chart display, daily/monthly water consumption aggregation report, historical data multi-condition query, data export;

4. User management module: 12 test cases, RBAC role creation, menu equipment permission assignment, multi-level user isolation, password authority verification, all based on the RBAC architecture of [11];

5. System configuration module: 8 test cases, database timing backup, expired data cleaning, communication parameter global configuration, interface format unified verification.

5.3.2. Test Execution & Verification Standard

All test cases are executed manually combined with automated Python test scripts complying with PEP 8 coding specifications [15]. The pass standard is defined as: function execution without exception, page display without disorder, database storage data consistent with terminal original measurement data, permission isolation effective without cross-authority access. For aggregation report function, the cumulative water consumption calculated by the background timing task needs to be compared with the sum of single original collection data one by one to verify the calculation accuracy.

5.3.3. Test Result Analysis

All 68 test cases pass the verification without functional failure. The daily and monthly water consumption aggregation report calculation results are completely consistent with the raw collected data, and there is no cumulative calculation error caused by cache asynchronous synchronization. The RBAC multi-level permission control realizes strict isolation between operation and maintenance personnel, water supply managers and ordinary viewing users, effectively avoiding illegal operation of equipment data. The full function test verifies the integrity, usability and functional correctness of each module of the system, which meets the actual business operation requirements of smart water supply enterprises.

Test results show that all performance indicators of the system meet the design requirements, and have significant advantages over traditional commercial DTU schemes in transmission efficiency and terminal low power consumption, with practical engineering implementation value.

6. Conclusions

This paper designs and implements an intelligent water meter monitoring system based on open-source DTU and Modbus RTU protocol, solving industrial pain points such as low efficiency of manual meter reading of traditional water meters, high cost of commercial DTUs and delayed operation and maintenance management. An integrated closed-loop architecture of "DTU collection terminal—cloud business platform—front-end visualization" is built, and the whole process development of real-time collection, network transmission, structured storage and multi-dimensional statistical analysis of water meter data is completed. Adopting modular layered design, the system can flexibly expand communication protocols and access new metering equipment, with strong maintainability and commercial implementation potential.

Follow-up research plans:

① Integrate edge computing algorithms on DTU edge terminals to realize local real-time identification of abnormal water consumption and water leakage events, referring to the edge anomaly detection scheme in reference [2];

② Expand compatibility with multiple industrial metering bus protocols such as DL/T645 and M-Bus to realize multi-meter unified collection as described in Zhang et al. [7] and Du et al. [8];

③Add TLS/DTLS end-to-end encrypted transmission to improve the communication security level between equipment and platform, making up for the lack of encryption mechanism of the current Modbus plaintext transmission architecture.

Acknowledgements

I would like to thank the National College Students' Innovation and Entrepreneurship Training Program (202510066007) for their support.

References

[1] Modbus Organization, Modbus Application Protocol Specification v1.1b3, (2012) [EB/OL]. Available: http://www.modbus.org.

[2] C. A. Rocha, R. M. Vicentini, J. R. D. Santos, et al., “A Long Term Evolution CAT-M Based Gateway for Serial Modbus Communication in Industrial Internet of Things Systems,” in 2026 IEEE International Conference on Industrial IoT, IEEE, (2026), pp. 1–6.

[3] W. Wang, Q. Li, “Design and Implementation of Low-Power DTU Based on ESP32,” Internet of Things Technologies, vol.13, no.5, pp.45–48, (2023).

[4] H. Demir, “Power Consumption Analysis of ESP32 and Arduino Nano for Low-Power Applications,” in 2025 9th International Symposium on Innovative Approaches in Smart Technologies, IEEE, (2025), pp. 112–117.

[5] S. Zhao, Q. H. Zhang, Q. J. Zhao, “Enhancing Bidirectional Modbus TCP ↔ RTU Gateway Performance: A UDP Mechanism and Markov Chain Approach,” Sensors, vol.25, no.12, p.6036, (2025).

[6] W. B. Wang, X. Zhang, Y. Zhao, Y. Shen, C. Luo, “Gateway Design of Internet of Things Based on Modbus Protocol,” Microcontrollers & Embedded Systems, vol.20, no.11, pp.59–62, (2020).

[7] Q. Zhang, Y. M. Jiang, P. Yu, “Research on the Multi-meter Integrated Copy System Based

on M-Bus," Electric Technology, no.6, pp.1–3, (2025).

[8] Y. Du, Z. L. Wang, C. C. Li, "Research and Design of Communication Interface Converter for Multi-meter Integration," Electrical Applications, vol.41, no.8, pp.56–60, (2022).

[9] Z. K. Ouyang, Q. Xu, S. S. Zhao, "Design of Multi-meter Integration Scheme Based on Electricity Information Acquisition System," in 2018 IEEE International Conference on Energy Internet, IEEE, (2018), pp. 782–786.

[10] Redis Labs, Redis Data Structures: Hashes and Sorted Sets, (2025) [EB/OL]. Available: https://redis.io.

[11] Y. Song, F. Li, "Design and Implementation of Access Control Under B/S System Based on RBAC," Software Guide, vol.7, pp.28–30, (2009).

[12] X. Liu, S. Liu, J. Wu, "High-speed end-to-edge data caching and forwarding architecture based on field programmable gate array," ETRI Journal, vol.47, no.5, pp.946–958, (2025).

[13] E. Sisinni, U. Saif, G. Emiliano, et al., "Survey of industrial communication technologies for IIoT energy monitoring," IEEE Internet of Things Journal, vol.11, no.7, pp.6123–6132, (2024).

[14] Vue.js Team, Vue 3 Documentation, (2025) [EB/OL]. Available: https://vuejs.org.

[15] Python Software Foundation, PEP 8 – Style Guide for Python Code, (2001) [EB/OL]. Available: https://peps.python.org/pep-008/.