International Journal of Engineering in Computer Science

E-ISSN: 2663-3590
P-ISSN: 2663-3582
Impact Factor (RJIF): 5.52
www.computersciencejournals.com/ijecs
IJECS 2025; 7(2): 313-317
Received: 15-10-2025
Accepted: 20-11-2025

Andrey Berezhnoy
Bachelor's Degree, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

Corresponding Author:
Andrey Berezhnoy
Bachelor's Degree, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

Integration of STM32- and ESP32-based microcontroller devices into monitoring and control systems

Andrey Berezhnoy

DOI: https://doi.org/10.33545/26633582.2025.v7.i2d.231

Abstract
The article examines the integration of STM32- and ESP32-based microcontroller devices into monitoring and control systems. It analyzes architectural solutions for embedding microcontrollers into cyber-physical systems that enable interaction with sensor peripherals, gateways, and cloud platforms. The importance of selecting appropriate communication protocols affecting the reliability, scalability, and energy efficiency of the system is emphasized. Particular attention is given to data protection, device authentication, and the use of hardware security mechanisms. It is disclosed that the interfacing between STM32 and ESP32 allows for the creation of scalable distributed architectures with integrating local autonomy, networked connectivity, and economic efficiency for industry and Internet of Things applications.

Keywords: STM32, ESP32, microcontroller, monitoring systems, cyber-physical systems, communication protocols, data security

1. Introduction
Modern monitoring and control systems are rapidly evolving in response to increasing demands for intelligence and automation across various sectors - from industrial production and energy to agriculture and smart home technologies. At the heart of this transformation lie microcontroller-based devices, which enable autonomous data processing, interaction with sensor modules, and communication with networked or cloud infrastructures. Among these, two prominent microcontroller families - STM32 by STMicroelectronics and ESP32 by Espressif Systems - have emerged as foundational components in the development of distributed cyber-physical systems due to their high energy efficiency, flexible configurability, and support for a wide range of communication protocols.
STM32 and ESP32 are widely utilized in industrial IoT solutions, telemetry data collection, resource management systems, and robotics platforms. These microcontrollers support both local control logic execution and data transmission to higher-level systems such as SCADA platforms, RESTful services, or cloud infrastructures using protocols like MQTT, HTTP, and Modbus. The aim of this article is to present an analytical and review-based examination of the architectures, communication protocols, and integration strategies for embedding STM32- and ESP32-based microcontroller solutions into monitoring and control systems of various types.

2. Main part. Architecture of microcontroller systems
In control and monitoring systems, a microcontroller system is a set of hardware and software modules based on a general-purpose microcontroller to receive, send, and, if required, store data from sensor modules or the external world (fig. 1).

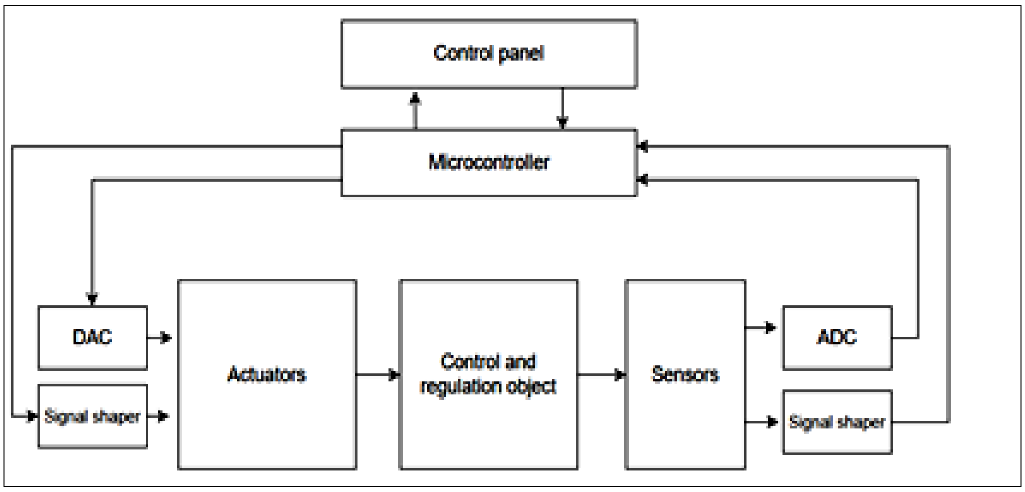

Fig 1: Architecture of a microcontroller system

In the broad context of cyber-physical systems, microcontrollers such as STM32 and ESP32 are the primary interface layer between digital infrastructure and the physical world. They take in analog or discrete signals from sensors, convert them into structured data, and provide feedback loops through actuators or network services. This data-driven control loop ensures real-time synchronization between physical processes and computational models, forming the foundation of intelligent distributed systems.

Microcontroller is a single chip that integrates a processor core, memory, and peripheral interfaces [1]. It is of low power consumption, has low size, and offers real-time support, making it essential in distributed cyber-physical systems. These systems accept, preprocess, and transmit sensor data and drive actuators in local or distributed environments. The targeted implementation depends on the architecture of the desired platform, with STM32 and ESP32 being the most used because of their distinctive design logic, degrees of integration, and use cases.

STMicroelectronics' STM32 family is based on ARM Cortex-M cores (M0 to M7) with a wide spectrum of performance from ultra-low power to high processing capacity. As it is modular and scalable in architecture, it is suited for industrial automation, medical, and automotive markets with the requirements of reliability and deterministic execution [2]. The variety of STM32 series enables adaptation to tasks of varying complexity - from simple sensor nodes to high-performance computing units (table 1).

Table 1: Comparative overview of STM32 microcontroller families [3, 4]

STM32 family	ARM cortex core	Clock frequency (MHz)	Power consumption	Flash/SRAM memory	Typical applications
STM32F0	M0	up to 48	Low	up to 128 KB / 16 KB	Low-cost devices, basic automation.
STM32F1	M3	up to 72	Moderate	up to 1 MB / 96 KB	General-purpose embedded systems, instrumentation.
STM32F3	M4	up to 72	Moderate	up to 512 KB / 80 KB	Motor control, analog signal processing.
STM32F4	M4	up to 180	Medium	up to 2 MB / 256 KB	Industrial controllers, HMI systems.
STM32H7	M7	up to 480	Above average	up to 2 MB / 1 MB	High-performance systems, real-time data handling.
STM32L0	M0+	up to 32	Ultra low	up to 192 KB / 20 KB	Battery-powered sensors, wearable electronics.
STM32L4	M4	up to 80	Ultra low	up to 1 MB / 128 KB	Energy-efficient IoT and portable devices.

In STM32-based applications, the microcontroller interacts with external communication modules - Ethernet, GSM, Wi-Fi, or LoRa - according to the needed channel. They lack on-board wireless interfaces in most of them but are highly configurable. They typically are low-level edge devices for multi-level control architectures for embedding specific algorithms, synchronized data acquisition, and communication with SCADA systems or higher-level gateways. Real-time support and high noise immunity make STM32 controllers the best fit for time-critical and noisy environments.

Espressif Systems ESP32 is a dual-core Tensilica Xtensa LX6 processor (240 MHz) system-on-chip (SoC) with built-in Wi-Fi and Bluetooth [5]. ESP32 is specifically designed for wireless data communication, fast IoT development, and effortless integration with cloud or client-server applications. Unlike classical microcontrollers, ESP32 combines computation, networking, and peripherals into a single chip, reducing external components and making circuit design effortless. It has built-in support for TCP/IP, HTTP, MQTT, and FreeRTOS and therefore can be set up as a local server with web interface, network client, or stand-alone data controller. The main technical specifications of ESP32 are given in table 2.

Table 2: Key technical specifications of the ESP32 platform [6, 7]

Parameter	Specification
Microcontroller core	Dual-core Tensilica Xtensa LX6 (240 MHz).
Wireless interfaces	Wi-Fi 802.11 b/g/n, Bluetooth 4.2 (classic + BLE).
RAM	Up to 520 KB internal SRAM.
Flash memory	Up to 16 MB (external SPI flash).
GPIO	Up to 34 general-purpose I/O pins.
Analog inputs	18 channels (12-bit ADC), 2 × 8-bit DAC.
Communication interfaces	UART, SPI, I²C, CAN, I2S, PWM.
Network protocols (stack)	TCP/IP, HTTP/HTTPS, MQTT, mDNS, WebSocket.
Real-time support	FreeRTOS with hardware timers and interrupts.
Power consumption (deep sleep)	~10 µA (typical).
Operating voltage	2.3 V - 3.6 V.
Security features	AES, SHA-2, RSA, Secure Boot, Flash Encryption.
Development tools	ESP-IDF, Arduino IDE, PlatformIO, MicroPython.
Typical applications	Smart home, wearable devices, sensor nodes, IoT hubs.

Unlike STM32, ESP32 offers less hardware configuration freedom (less model and peripheral selection) but excels in prototyping speed, access ease, and wireless integration ease. There are undoubtedly trade-offs with each platform, however, as far as performance, power usage, and cost. STM32 provides high-precision timing and industrious industrial reliability at the expense of higher component and development cost, while ESP32 delivers an integrated connection with fast deployment and minimal real-time determinism as well as lower memory capacity. The two therefore have to be selected according to the trade-off of computational efficiency, energy budget, and system complexity needed.

One key aspect of microcontroller systems is reliable and flexible interaction with peripheral components - both passive (sensors, actuators) and active (other controllers, interface modules). STM32 and ESP32 both have large sets of hardware interfaces to attach analog and digital devices with high-data-rate and low-latency data communication. STM32 includes several ADC and PWM channels, SPI, I²C, and UART interfaces that render it compatible with sensor and industrial modules, and DMA and interrupts reduce CPU load for taking measurements at high rates. ESP32, though smaller in size, provides support for as many as 18 ADC channels, two DACs, I2S, and touch-sensitive GPIO for simple inputs. Wireless peripherals and mesh networking are supported through built-in Wi-Fi and Bluetooth, and OTA updates provide easier remote management. Both platforms support multi-level peripheral interaction, including sensor polling, threshold event response, and buffered data transfer, enabling adaptive and energy-efficient monitoring architectures.

3. Integration into multi-tier IT architectures (edge-cloud, client-server)

Modern monitoring and control systems are increasingly designed as multi-layered distributed IT architectures, where each level performs specific functions - from data acquisition to analysis, visualization, and decision-making. Within such structures, STM32 and ESP32 microcontroller-based devices serve as essential edge nodes, providing primary data processing, object control, and communication with higher-level components. In an edge-cloud architecture, the microcontroller operates at the system's edge, performing sensor data acquisition, filtering, normalization, and depending on configuration, aggregation and compression (fig. 2).

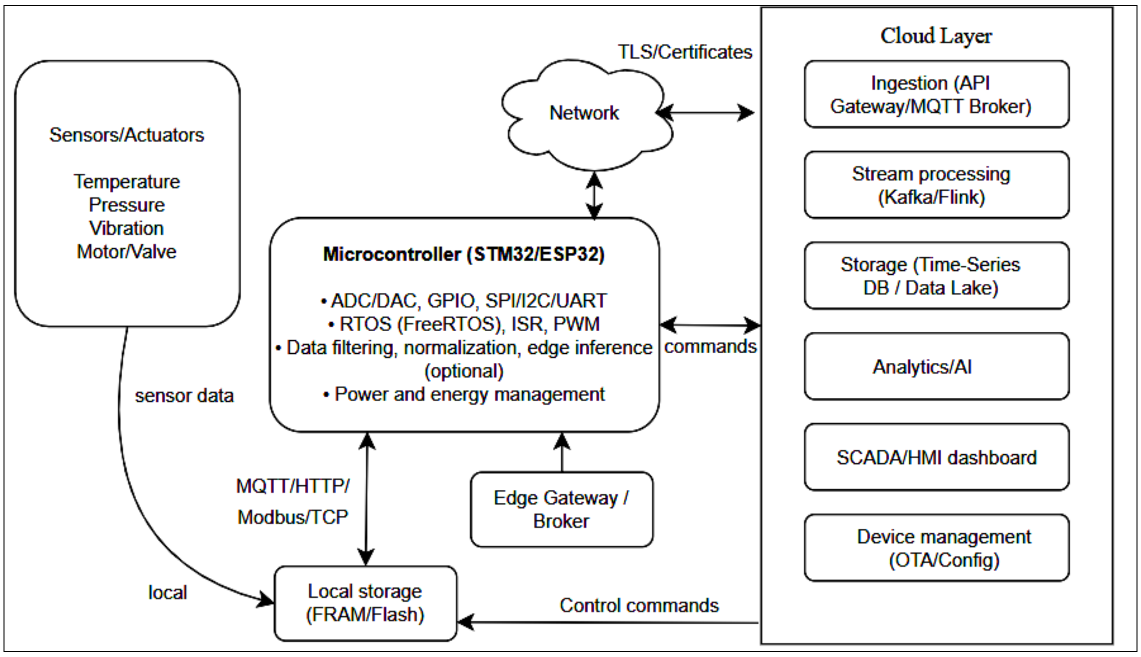
Sensors/Actuators
Temperature
Pressure
Vibration
Motor/Valve
TLS/Certificates
Network
Cloud Layer
Ingestion (API Gateway/MQTT Broker)
Stream processing (Kafka/Flink)
Storage (Time-Series DB / Data Lake)
Analytics/AI
SCADA/HMI dashboard
Device management (OTA/Config)
Microcontroller (STM32/ESP32)
• ADC/DAC, GPIO, SPI/I2C/UART
• RTOS (FreeRTOS), ISR, PWM
• Data filtering, normalization, edge inference (optional)
• Power and energy management
commands
sensor data
MQTT/HTTP/ Modbus/TCP
Edge Gateway / Broker
local
Local storage (FRAM/Flash)
Control commands

Fig 2: Placement of the microcontroller in an edge-cloud architecture

Processed data are transmitted to the cloud infrastructure through a gateway or directly (in the case of ESP32) for storage, visualization, and analytics. This is the kind of model typical for scalable IoT solutions where the bulk of the logic is executed in the cloud (e.g., AWS IoT Core, Azure IoT Hub, etc.) [8]. STM32, lacking on-chip communication modules, typically connects to cloud services via intermediary nodes such as Raspberry Pi gateways, ESP32 bridges, or industrial PLCs with network stacks via UART, CAN, and SPI interfaces.

The ESP32, however, can be easily integrated into cloud infrastructure. Having onboard Wi-Fi and Bluetooth modules, and support for MQTT, HTTP/HTTPS, CoAP, and TLS/SSL protocols, it is able to transmit data securely to the cloud platforms without the necessity of incorporating additional gateways. This reduces hardware costs and optimizes infrastructure, particularly in environments with limited space or at a distance. Provision for OTA firmware updates, device authentication, and encryption makes the ESP32 a suitable choice for security-oriented applications where data integrity and security are of prime importance. An alternative approach is the client-server architecture, typical of local automated systems where interaction occurs within a single network without accessing cloud components (fig. 3).

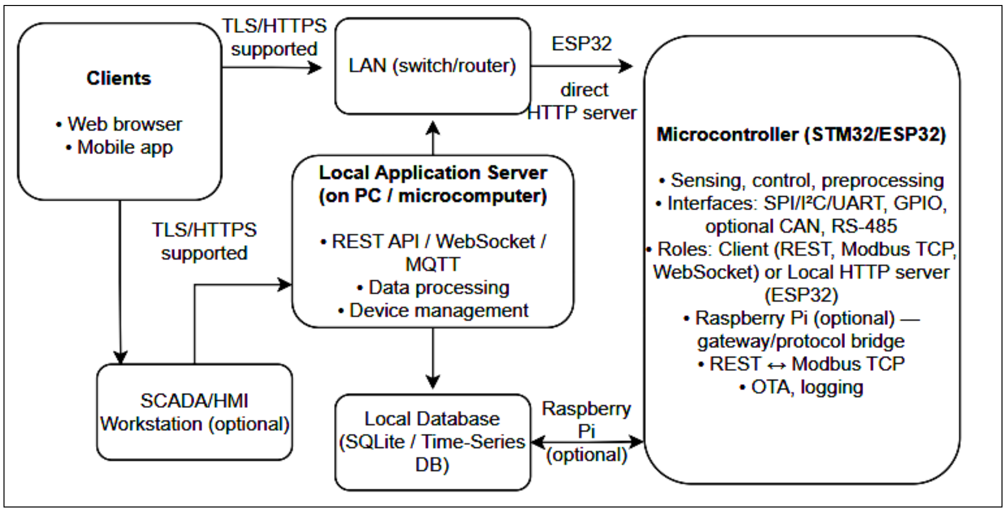
TLS/HTTPS supported
Clients
• Web browser
• Mobile app
LAN (switch/router)
ESP32
direct HTTP server
Microcontroller (STM32/ESP32)
• Sensing, control, preprocessing
• Interfaces: SPI/I²C/UART, GPIO, optional CAN, RS-485
• Roles: Client (REST, Modbus TCP, WebSocket) or Local HTTP server (ESP32)
• Raspberry Pi (optional) — gateway/protocol bridge
• REST ↔ Modbus TCP
• OTA, logging
TLS/HTTPS supported
Local Application Server (on PC / microcomputer)
• REST API / WebSocket / MQTT
• Data processing
• Device management
SCADA/HMI Workstation (optional)
Local Database (SQLite / Time-Series DB)
Raspberry Pi (optional)

Fig 3: Client-server architecture: microcontroller placement in a local network

In such configurations, microcontrollers can act either as clients exchanging data via REST API or as local servers. The ESP32 can be configured as an HTTP server handling requests from a browser or mobile app, enabling a built-in user interface for smart home systems or telemetry points [9]. The STM32 can also join a local network using an external Ethernet module (e.g., W5500), supporting Modbus TCP, HTTP, or WebSocket communication through an intermediate microcomputer. Thus, STM32 and ESP32 microcontrollers can be effectively integrated across various layers of digital architecture - from low-level process control to data transmission to global cloud platforms. The choice of architecture depends on latency, security, autonomy, and infrastructure capabilities, while the ESP32's wireless flexibility and the STM32's precise timing and peripheral expandability make them complementary in distributed intelligent systems.

From an applied perspective, the proper selection of microcontroller architecture also has measurable economic implications. Optimized integration reduces capital and operational expenditures (CAPEX and OPEX) by minimizing hardware redundancy, simplifying maintenance, and enabling scalable IoT infrastructures that can evolve with changing system requirements.

4. Communication Protocols and Data Exchange

The reliability and efficiency of data exchange between the microcontroller, sensor peripherals, and external computing systems directly depend on the interfaces and communication protocols used. STM32- and ESP32-based systems support a wide range of both wired and wireless interfaces, allowing the architecture to be adapted to specific requirements - from high-speed real-time communication to energy-efficient data transmission in wireless distributed environments. The choice of protocol is determined by several factors, including data volume and frequency, reliability, energy efficiency, security requirements, and the complexity of firmware and hardware implementation (table 3).

Table 3: Comparison of communication protocols for STM32 and ESP32 microcontrollers [10, 11]

Protocol	Supported by	Typical use case	Max data rate	Remarks
UART	STM32, ESP32	Sensor interfacing, debugging.	up to 1 Mbps	Simple, widely supported.
SPI	STM32, ESP32	High-speed data exchange.	up to 80 Mbps (ESP32)	Used for displays, flash memory.
I²C	STM32, ESP32	Low-speed multi-device communication.	up to 1 Mbps	Requires pull-up resistors.
CAN	STM32, ESP32* STM32 (via external modules/stack)	Industrial communication.	1 Mbps (standard)	Native in STM32, optional in ESP32.
Ethernet	STM32	Wired network communication.	10/100 Mbps	Requires external PHY module.
Wi-Fi	ESP32	Cloud communication, web servers.	up to 150 Mbps	Built-in support in ESP32.
MQTT	ESP32	Lightweight IoT messaging.	Depends on transport (TCP/IP)	Requires TCP/IP stack, cloud ready.

The protocols presented in table 3 are among the most commonly used when designing microcontroller-based systems built on STM32 and ESP32. It should be noted that other interfaces are also widely applied in embedded solutions - such as USB (Device/Host/OTG), Modbus RTU/TCP, WebSocket, CoAP, LoRaWAN, Zigbee, and others.

Thus, the choice of a communication protocol represents not only an engineering but also an architectural decision that directly affects the scalability, energy efficiency, and reliability of the entire monitoring or control system. Low-level protocols such as UART, SPI, and I²C remain relevant for local interaction with peripheral modules, while higher-level protocols such as MQTT, HTTP, and Modbus TCP provide connectivity to external systems, gateways, and cloud platforms. The right combination of several protocols within a single project ensures architectural flexibility, fault tolerance, and adaptability to specific application scenarios - from industrial networks to distributed IoT environments [12].

With effective communication, system integration of STM32- and ESP32-based systems also requires special attention towards secure communication and data protection. Both platforms also provide hardware-based security features to ensure data integrity, device authentication, and trusted updates. STM32 microcontrollers offer cryptographic accelerators, true random number generators, and secure boot with firmware encryption and authentication. ESP32 includes AES, SHA-2, RSA engines, eFuse key storage, and Flash Encryption, in addition to secure boot and signed OTA updates. These are supported by TLS or DTLS for MQTT, HTTP, or CoAP communications and provide security against cloning, data snooping, and third-party remote control. The application of mTLS, certificate-based authentication, and partitioned privileges allows microcontroller-based designs to maintain confidentiality and reliability even when used in distributed environments at large scale.

5. Conclusion

Integration of STM32- and ESP32-based microcontroller units into monitoring and control systems is a technology-driven, architecturally feasible approach to building sophisticated cyber-physical systems. STM32 provides extensive functionality for building effective, scalable, and power-efficient solutions with accurate performance and deterministic processing, and as such is the optimal solution for industrial automation and real-time applications. ESP32, by contrast, offers a compact platform with built-in wireless interfaces and an entire network stack and enables rapid development of IoT devices and seamless integration into cloud and client-server networks. Inclusion of such platforms in multi-tier IT architecture allows for development of well-balanced systems incorporating local autonomy, network connectivity, and efficient utilization of computational and energy resources.

References

1. Idris MR, Irdayanti MN, Ramlee M. A review of microcontroller design in future industrial revolution. AIP Conference Proceedings. AIP Publishing LLC, 2023; 2750(1): 050031.
2. Cai H, Wu Z, Chen M. Design of *STM32*-based Quadrotor UAV Control System. KSII Trans. Internet Inf. Syst. 2023; 17(2): 353-368.
3. Siddiqui A, Iyer SS, Ranade A. *STM32* and *LoRa*-based wireless sensor network for precision agriculture. Technological Innovations & Applications in Industry 4.0. CRC Press, 2023; 82-88.
4. Vdovichenko O, Perepelitsyn A. Analysis of nomenclature of microcontrollers: Decision-making during building of home automation elements. Conference on Integrated Computer Technologies in Mechanical Engineering Synergetic Engineering. Cham: Springer Nature Switzerland, 2024; 520-532.
5. Litayem N, Al-Sa'di A. Exploring the programming model, security vulnerabilities, and usability of *ESP8266* and *ESP32* platforms for IoT development. 2023 IEEE 3rd International Conference on Computer Systems (ICCS). IEEE, 2023; 150-157.
6. Espinosa-Gavira MJ. Characterization and performance evaluation of *ESP32* for real-time synchronized sensor networks. Procedia Computer Science. 2024; 237: 261-268.
7. Yang LJ. Two-wheeled self-balancing robot controlled by dual-core *STM32* and *ESP32*. 2025 IEEE 12th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). IEEE, 2025; 12: 1111-1115.
8. Wang M, Xu CA, Lin Y, Lu Z, Sun J, Gui G. A distributed sensor system based on cloud-edge-end network for industrial internet of things. Future Internet. 2023; 15(5): 171-174.
9. Khalil MR, Mohammed LA, Yousif ON. Customer application protocol for data transfer between embedded processor and microcontroller systems. TELKOMNIKA (Telecommunication Computing Electronics and Control). 2021; 19(3): 801-808.
10. Gupta E, Gupta V. Comparative analysis of *ESP32* and *ESP8266* for AI-powered applications. 2025 International Conference on Next Generation Communication & Information Processing (INCIP). IEEE, 2025; 992-996.
11. Maroşan A. Real-time data acquisition with *ESP32* for IoT applications using open-source MQTT brokers. Proceedings in Manufacturing Systems. 2024; 19(2): 61-68.
12. Drogunova Y. The impact of software quality assurance practices on the competitiveness of the technology sector. International Journal of Advanced Research in Science, Communication and Technology. 2025; 5(1): 735-740.