Open Access Journal on Engineering Applications
ISSN: 3105-7004
www.oajea.hafizlab.com

ESP32 Microcontroller: A Review of Architecture, Communication Protocols, Applications and Research Challenges

Md Hafizur Rahman[1,*], **Muhammad Shihab**[1], **Md. Arifur Rahman**[1], **and M. Naderuzzaman**[2]

[1]HafizLab, Dhaka, Bangladesh
[2]Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh

Article Info
Received: 12 Jan 2026
Revised: 28 Jan 2026
Accepted: 05 Feb 2026
Published: 14 Feb 2026
Volume No: 01
Issue No: 02
Page No: 19 - 33

Corresponding author:
Md Hafizur Rahman
hafizurfpbd@gmail.com

DOI:
10.64886/oajea.0102.003

Check for updates

Abstract

The rapid expansion of the Internet of Things (IoT) has increased the demand for low-cost, high-performance, and energy-efficient microcontrollers with integrated wireless communication capabilities. Among the available platforms, the ESP32 microcontroller has emerged as a widely adopted solution for IoT and embedded system applications due to its dual-core processing architecture, built-in Wi-Fi and Bluetooth connectivity, rich peripheral support, and flexible software ecosystem. This paper presents a comprehensive review of the ESP32 microcontroller, focusing on its hardware architecture, supported communication protocols, development frameworks, and application domains. The study systematically examines the use of ESP32 in diverse areas such as smart agriculture, healthcare and wearable systems, smart city infrastructure, industrial IoT, and environmental monitoring. In addition, a comparative analysis with other commonly used microcontrollers is provided to highlight the strengths and limitations of ESP32-based systems. Key research challenges related to power consumption, security, scalability, real-time performance, and memory constraints are critically discussed. Finally, future research directions are outlined, emphasizing opportunities in edge intelligence, hybrid communication architectures, energy-efficient designs, and secure IoT deployments. This review aims to serve as a valuable reference for researchers and practitioners in selecting, designing, and optimizing ESP32-based solutions for modern IoT applications.

Keywords:
ESP32, Wi-Fi, Applications, IoT, Microcontroller, Low Power, Devices

1. Introduction

The rapid advancement of embedded systems and wireless communication technologies has significantly accelerated the growth of the Internet of Things (IoT), enabling the development of smart, connected, and autonomous systems across diverse application domains [1–3]. IoT systems typically consist of resource-constrained devices that sense, process, and transmit data over wireless networks, making the selection of an appropriate microcontroller unit (MCU) a critical design decision. These MCUs must balance computational capability, energy efficiency, cost, and integrated communication support to meet the demands of modern IoT applications.

In recent years, low-power microcontrollers with built-in wireless connectivity have gained substantial attention from both academic researchers and industry practitioners. Traditional platforms such as Arduino Uno, although popular for prototyping, lack native wireless capabilities and advanced processing features required for scalable IoT deployments. Similarly, single-board computers like Raspberry Pi offer higher computational power but suffer from increased energy consumption and cost, making them less suitable for battery-powered and large-scale embedded systems. As a result, there has been a growing demand for versatile microcontroller platforms that combine processing efficiency with integrated wireless communication.

The ESP32 microcontroller (Figure 1), developed by Espressif Systems, has emerged as one of the most widely adopted platforms for IoT and embedded applications. It represents a significant evolution over its predecessor, the ESP8266, by introducing a dual-core 32-bit processor architecture, enhanced memory resources, and native support for both Wi-Fi and Bluetooth communication [4]. The ESP32 integrates IEEE 802.11 b/g/n Wi-Fi, Bluetooth Classic, and Bluetooth Low Energy (BLE) within a single low-cost chip, enabling seamless connectivity for a wide range of use cases. In addition, it provides extensive peripheral interfaces, including UART, SPI, I2C, CAN, ADC, DAC, PWM, and touch sensors, which make it suitable for sensor-based, real-time, and control-oriented applications.

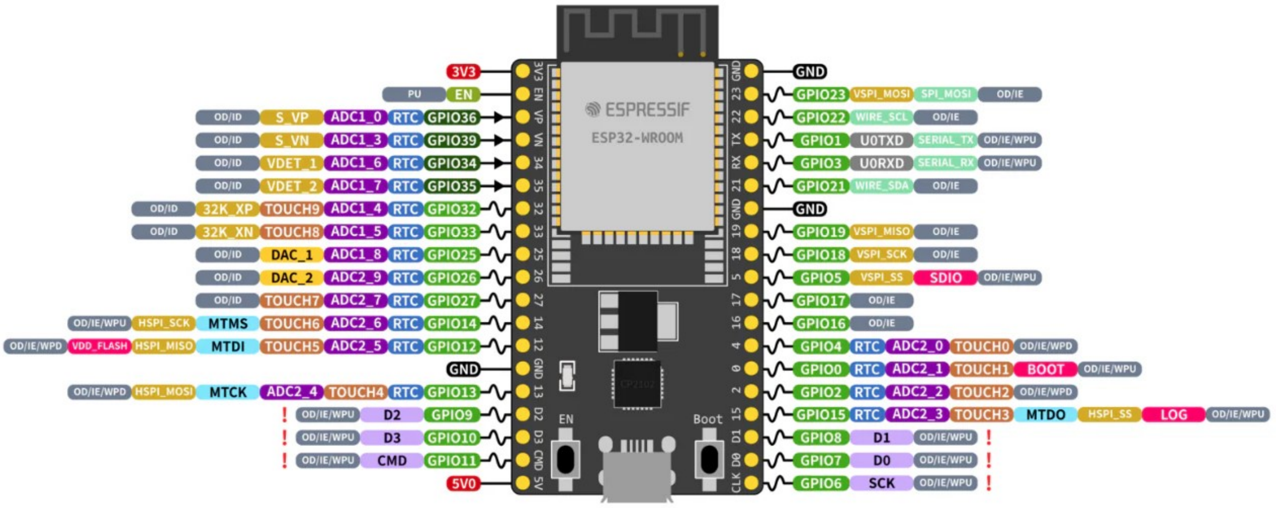
ESPRESSIF
ESP32-WROOM
EN
Boot
3V3
PU EN
OD/ID S_VP ADC1_0 RTC GPIO36
OD/ID S_VN ADC1_3 RTC GPIO39
OD/ID VDET_1 ADC1_6 RTC GPIO34
OD/ID VDET_2 ADC1_7 RTC GPIO35
OD/ID 32K_XP TOUCH9 ADC1_4 RTC GPIO32
OD/ID 32K_XN TOUCH8 ADC1_5 RTC GPIO33
OD/ID DAC_1 ADC1_8 RTC GPIO25
OD/ID DAC_2 ADC2_9 RTC GPIO26
OD/ID TOUCH7 ADC2_7 RTC GPIO27
OD/IE/WPU HSPI_SCK MTMS TOUCH6 ADC2_6 RTC GPIO14
OD/IE/WPD VDD_FLASH HSPI_MISO MTDI TOUCH5 ADC2_5 RTC GPIO12
GND
OD/IE/WPD HSPI_MOSI MTCK ADC2_4 TOUCH4 RTC GPIO13
! OD/IE/WPU D2 GPIO9
! OD/IE/WPU D3 GPIO10
! OD/IE/WPU CMD GPIO11
5V0
3V3 EN VP VN 34 35 32 33 25 26 27 14 12 GND 13 D2 D3 CMD 5V
GND 23 22 TX RX 21 GND 19 18 5 17 16 4 0 2 15 D1 D0 CLK
GND
GPIO23 VSPI_MOSI SPI_MOSI OD/IE
GPIO22 WIRE_SCL OD/IE
GPIO1 U0TXD SERIAL_TX OD/IE/WPU
GPIO3 U0RXD SERIAL_RX OD/IE/WPU
GPIO21 WIRE_SDA OD/IE
GND
GPIO19 VSPI_MISO OD/IE
GPIO18 VSPI_SCK OD/IE
GPIO5 VSPI_SS SDIO OD/IE/WPU
GPIO17 OD/IE
GPIO16 OD/IE
GPIO4 RTC ADC2_0 TOUCH0 OD/IE/WPD
GPIO0 RTC ADC2_1 TOUCH1 BOOT OD/IE/WPU
GPIO2 RTC ADC2_2 TOUCH2 OD/IE/WPD
GPIO15 RTC ADC2_3 TOUCH3 MTDO HSPI_SS LOG OD/IE/WPU
GPIO8 D1 OD/IE/WPU !
GPIO7 D0 OD/IE/WPU !
GPIO6 SCK OD/IE/WPU !

Figure 1: System Architecture Diagram of ESP32

Another notable feature of the ESP32 is its advanced power management capability, which includes multiple sleep modes, dynamic frequency scaling, and real-time clock (RTC) support. These features allow ESP32-based systems to achieve extended battery life, a key requirement for remote and long-term IoT deployments. Furthermore, the availability of hardware-based cryptographic accelerators enhances the security of communication and data storage, addressing one of the most critical concerns in IoT systems. From a software perspective, the ESP32 supports a rich development ecosystem, including the ESP-IDF framework, Arduino core, MicroPython, and FreeRTOS, enabling developers to design flexible and scalable embedded solutions.

Due to these capabilities, the ESP32 has been extensively utilized in a wide range of research and real-world applications, such as smart agriculture, healthcare monitoring, wearable devices, smart city infrastructure, industrial automation, and environmental sensing. Numerous studies have explored ESP32-based systems for data acquisition, wireless sensor networks, real-time monitoring, and edge-level processing. However, despite its widespread adoption, challenges related to power consumption under continuous wireless operation, memory constraints for computationally intensive tasks, security vulnerabilities, and real-time performance limitations remain active research concerns.

Although several studies have focused on specific ESP32-based implementations, a comprehensive and structured review that consolidates its architectural features, supported communication protocols, application domains, and associated research challenges is still necessary. Such a review is essential for guiding researchers, system designers, and practitioners in selecting and optimizing the ESP32 for different IoT scenarios, as well as identifying open research issues and future development directions.

In this context, this paper presents a comprehensive review of the ESP32 microcontroller, emphasizing its hardware architecture, communication protocols, software ecosystem, and application domains. The paper also provides a comparative discussion with other popular microcontroller platforms and critically analyzes the limitations and research challenges associated with ESP32-based systems. Finally, potential future research directions are outlined to support the development of more efficient, secure, and scalable ESP32-based IoT solutions [5, 6].

2. ESP32 Architecture

The ESP32 microcontroller is a highly integrated system-on-chip (SoC) designed to meet the computational, communication, and energy-efficiency requirements of modern IoT and embedded applications. Its architecture combines a dual-core processing unit, flexible memory organization, rich peripheral support, integrated wireless modules, and advanced power management features within a compact and low-cost platform. This section presents a detailed architectural analysis of the ESP32 (Figure 2), highlighting its core hardware components and their functional roles [1, 2].

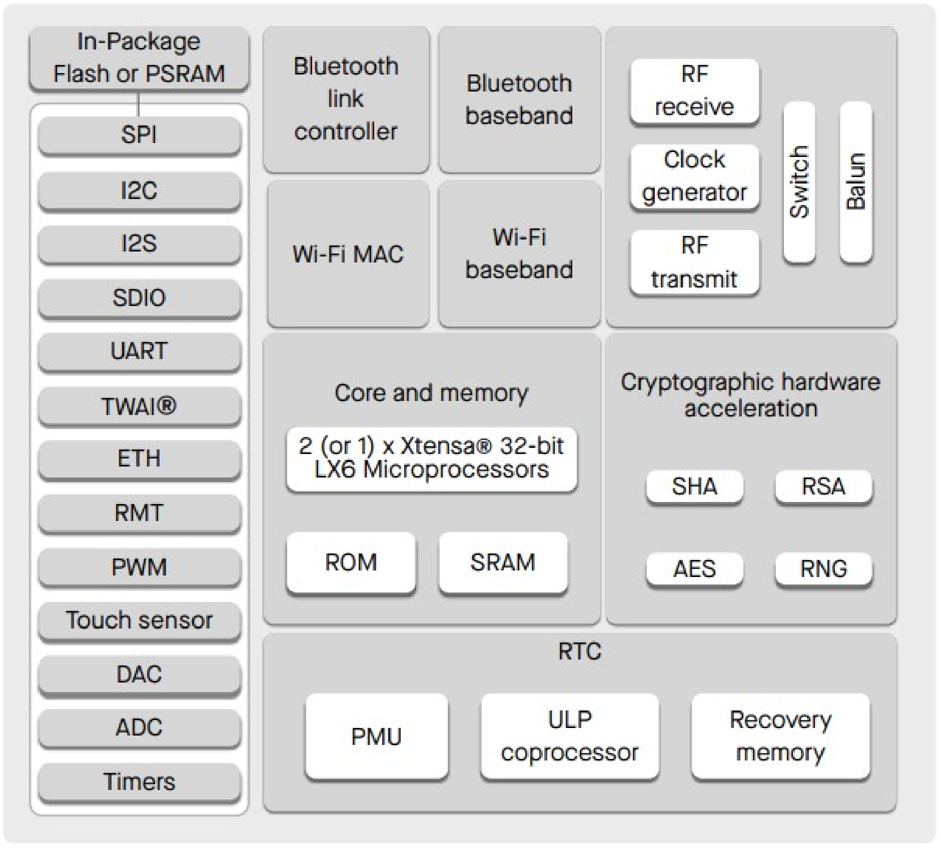

Figure 2: ESP32 Functional Block Diagram

2.1 Processing Unit

At the core of the ESP32 architecture lies a dual-core 32-bit Xtensa LX6 processor, which significantly enhances its computational capability compared to single-core microcontrollers commonly used in IoT systems. Each core can operate at a clock frequency of up to 240 MHz, enabling efficient execution of parallel tasks. The dual-core design allows one core to handle time-critical operations such as communication and real-time control, while the other core manages application-level processing, thereby improving system responsiveness and multitasking efficiency.

The ESP32 architecture supports symmetric multiprocessing and is tightly integrated with the FreeRTOS operating system, enabling task scheduling, inter-core communication, and real-time execution. This architectural feature makes the ESP32 suitable for applications requiring concurrent processing, such as sensor data acquisition combined with wireless data transmission.

2.2 Memory Architecture

The memory subsystem of the ESP32 is designed to balance performance and power efficiency. It includes multiple types of memory, each serving specific functional requirements. The ESP32 integrates static random-access memory (SRAM) for program execution and data storage, which is divided into instruction RAM and data RAM. This separation improves access speed and supports efficient execution of embedded applications.

In addition to internal SRAM, the ESP32 supports external flash memory, typically ranging from 4 MB to 16 MB, which is used for firmware storage, file systems, and over-the-air (OTA) updates. A dedicated real-time clock (RTC) memory is also included, allowing critical data to be retained during deep sleep modes. This feature is particularly useful for ultra-low-power IoT applications that rely on periodic wake-up and data transmission cycles.

2.3 Integrated Wireless Communication Modules

One of the defining features of the ESP32 architecture is its integrated wireless communication capability. The microcontroller includes built-in support for IEEE 802.11 b/g/n Wi-Fi, enabling high-speed data transmission and internet connectivity without the need for external modules. The Wi-Fi subsystem supports both station (STA) and access point (AP) modes, providing flexibility for different network configurations.

In addition to Wi-Fi, the ESP32 integrates Bluetooth Classic and Bluetooth Low Energy (BLE) modules. Bluetooth Classic is suitable for continuous data streaming applications, while BLE is optimized for low-power, short-range communication. The coexistence of Wi-Fi and Bluetooth within the same chip allows ESP32-based systems to support hybrid communication scenarios, such as gateway devices and wearable applications.

2.4 Peripheral Interfaces

The ESP32 architecture offers extensive peripheral support, making it adaptable to a wide range of embedded system designs. It provides multiple general-purpose input/output (GPIO) pins that can be dynamically configured for different functions. The microcontroller supports standard communication interfaces, including UART, SPI, I2C, and CAN, enabling seamless integration with sensors, actuators, displays, and external modules.

Analog functionality is supported through analog-to-digital converters (ADC) and digital-to-analog converters (DAC), allowing ESP32-based systems to process analog sensor signals and generate analog outputs. Additionally, the ESP32 includes pulse-width modulation (PWM) channels for motor control and power regulation, as well as capacitive touch sensors, which are commonly used in human–machine interface applications.

2.5 Power Management Unit

Energy efficiency is a critical requirement for IoT systems, and the ESP32 architecture incorporates a sophisticated power management unit (PMU) to address this need. The microcontroller supports multiple power-saving modes, including active mode, light sleep, deep sleep, and hibernation, each offering different trade-offs between power consumption and system responsiveness.

Dynamic voltage and frequency scaling (DVFS) [7] allows the ESP32 to adjust its operating frequency based on workload demands, thereby reducing unnecessary power consumption. The integration of an RTC controller enables timed wake-up events and external interrupt-based activation, making the ESP32 particularly suitable for battery-powered and energy-constrained deployments.

2.6 Security Hardware Features

Security is an essential architectural consideration in IoT systems, and the ESP32 includes several hardware-level security mechanisms. These include cryptographic accelerators for encryption and decryption operations, secure boot support, and flash encryption. By offloading cryptographic tasks to dedicated hardware, the ESP32 improves both performance and security while minimizing software overhead.

These security features enable the development of secure communication channels, authenticated firmware updates, and protected data storage, which are crucial for preventing unauthorized access and cyber threats in IoT environments.

2.7 Architectural Significance for IoT Applications

The architectural design of the ESP32 reflects a careful balance between performance, flexibility, and energy efficiency. Its dual-core processing, integrated wireless modules, extensive peripheral support, and advanced power management make it a versatile platform for a wide range of IoT applications. The inclusion of security hardware further enhances its suitability for real-world deployments where data integrity and system reliability are critical.

Overall, the ESP32 architecture provides a solid foundation for developing scalable, secure, and energy-efficient IoT systems, making it a preferred choice for both academic research and industrial applications.

3. Communication Protocols Supported by ESP32

Reliable and efficient communication is a fundamental requirement for Internet of Things (IoT) systems, as it directly affects data transmission, system scalability, and energy consumption. The ESP32 microcontroller is designed as a communication-centric platform, integrating multiple wireless and wired communication protocols within a single system-on-chip [4]. This section reviews the major communication protocols supported by the ESP32 and discusses their suitability for different application scenarios.

3.1 Wi-Fi Communication

The ESP32 integrates full support for IEEE 802.11 b/g/n Wi-Fi, operating in the 2.4 GHz frequency band. Unlike many conventional microcontrollers that rely on external networking modules, the ESP32 includes an embedded TCP/IP stack, allowing it to directly connect to local networks and the Internet. The Wi-Fi subsystem supports both station (STA) and access point (AP) modes, as well as a hybrid STA+AP configuration, enabling flexible network topologies.

In IoT applications, Wi-Fi connectivity enables high-throughput data transmission, making the ESP32 suitable for real-time monitoring, cloud-based data logging, and firmware updates over the air (OTA). However, Wi-Fi communication typically consumes more power than short-range protocols, which necessitates careful power management strategies in battery-operated systems. Despite this limitation, the widespread availability of Wi-Fi infrastructure makes ESP32-based devices easy to deploy in urban and industrial environments.

3.2 Bluetooth Classic

The ESP32 supports Bluetooth Classic, which is designed for continuous and relatively high-data-rate wireless communication. Bluetooth Classic is commonly used in applications requiring persistent connections, such as audio streaming, serial data transmission, and device-to-device communication. The protocol provides moderate data throughput and reliable connectivity over short distances, typically within a range of 10 to 100 meters depending on environmental conditions.

In embedded systems, Bluetooth Classic is often employed for configuration, diagnostics, and local data exchange between the ESP32 and smartphones, tablets, or computers. Its ability to support continuous connections makes it suitable for interactive and user-centric IoT applications.

3.3 Bluetooth Low Energy (BLE)

In addition to Bluetooth Classic, the ESP32 integrates Bluetooth Low Energy (BLE), which is optimized for ultra-low-power communication. BLE is particularly well-suited for applications that require infrequent data transmission, such as wearable devices, wireless sensors, and beacon-based systems. By operating with short data packets and low duty cycles, BLE significantly reduces energy consumption compared to Wi-Fi and Bluetooth Classic.

The ESP32's BLE stack supports common profiles and services, enabling seamless interoperability with mobile devices and other BLE-enabled platforms. This makes ESP32-based systems ideal for healthcare monitoring, fitness tracking, and proximity-based applications where battery life is a critical concern.

3.4 Coexistence of Wi-Fi and Bluetooth

A notable feature of the ESP32 is its ability to support simultaneous operation of Wi-Fi and Bluetooth through an efficient coexistence mechanism. This allows developers to design hybrid communication systems in which Wi-Fi is used for high-bandwidth data transmission, while Bluetooth or BLE is used for local configuration or low-power communication. Such coexistence is particularly valuable in gateway devices, smart home systems, and edge computing applications.

The coexistence mechanism ensures minimal interference between the two wireless protocols by dynamically managing radio resources, thereby maintaining stable and reliable communication performance.

3.5 Serial Communication Protocols

Beyond wireless connectivity, the ESP32 supports several wired communication protocols that are essential for interfacing with sensors, actuators, and external peripherals. These include Universal Asynchronous Receiver-Transmitter (UART) for serial communication, Serial Peripheral Interface (SPI) for high-speed data exchange, and Inter-Integrated Circuit (I2C) for connecting low-speed peripheral devices.

These protocols enable the ESP32 to act as a central controller in sensor networks, collecting data from multiple sources and transmitting it wirelessly to cloud servers [8] or local gateways. The flexibility of these interfaces enhances the adaptability of ESP32-based systems across diverse application domains.

3.6 Controller Area Network (CAN)

The ESP32 also supports the Controller Area Network (CAN) protocol, which is widely used in industrial automation and automotive systems. CAN provides robust and reliable communication in noisy environments, making it suitable for mission-critical applications. The availability of CAN support extends the use of ESP32 beyond consumer-level IoT into industrial and embedded control systems.

3.7 Communication Protocol Selection for IoT Applications

The wide range of communication protocols supported by the ESP32 enables system designers to select the most appropriate communication strategy based on application requirements such as data rate, power consumption, communication range, and network topology. Wi-Fi is well-suited for high-bandwidth and cloud-connected applications, while BLE is preferred for low-power and short-range communication. Wired protocols such as SPI, I2C, and CAN complement wireless connectivity by enabling efficient peripheral integration.

Overall, the communication capabilities of the ESP32 provide a flexible and scalable foundation for developing diverse IoT solutions, reinforcing its position as a versatile microcontroller platform for modern embedded systems.

4. Software Ecosystem and Development Frameworks

The widespread adoption of the ESP32 microcontroller in IoT and embedded system research is strongly supported by its rich and flexible software ecosystem. The availability of multiple development frameworks, programming environments, and real-time operating system support allows developers and researchers to design scalable, portable, and application-specific solutions. This section reviews the major software development frameworks supported by the ESP32 and highlights their characteristics, advantages, and limitations.

4.1 ESP-IDF Framework

The Espressif IoT Development Framework (ESP-IDF) is the official and most comprehensive software development framework for the ESP32. It provides low-level access to hardware resources and includes optimized drivers for peripherals, networking stacks, security features, and power management. ESP-IDF is based on the C and C++ programming languages and is tightly integrated with the FreeRTOS real-time operating system.

ESP-IDF offers fine-grained control over system resources, making it suitable for performance-critical and research-oriented applications. Features such as task scheduling, inter-process communication, memory management, and hardware abstraction layers enable the development of complex embedded systems. Additionally, ESP-IDF supports advanced functionalities including over-the-air (OTA) firmware updates, secure boot, flash encryption, and debugging tools, which are essential for reliable and secure IoT deployments.

4.2 FreeRTOS Support

The ESP32 natively supports FreeRTOS, an open-source real-time operating system widely used in embedded systems. FreeRTOS enables multitasking, deterministic task execution, and real-time scheduling, which are critical for applications requiring concurrent processing and time-sensitive operations. The dual-core architecture of the ESP32 allows tasks to be distributed across cores, improving system responsiveness and throughput.

FreeRTOS support enhances the scalability of ESP32-based systems by enabling modular software design and efficient resource utilization. Tasks such as sensor data acquisition, wireless communication, and application logic can be executed independently, reducing latency and improving reliability. This makes the ESP32 particularly suitable for real-time IoT applications, industrial monitoring, and edge computing systems.

4.3 Arduino Framework for ESP32

To simplify development and accelerate prototyping, the ESP32 supports the Arduino framework, which provides a high-level programming environment and a large collection of libraries. The Arduino core for ESP32 allows developers to write firmware using simplified APIs while still accessing many of the advanced features of the underlying hardware.

The Arduino framework is especially popular among beginners and researchers who require rapid development and experimentation. Although it offers ease of use and faster implementation, it provides less control over low-level hardware operations compared to ESP-IDF. Nevertheless, the Arduino framework is widely used in academic projects, proof-of-concept systems, and early-stage prototypes due to its simplicity and extensive community support.

4.4 MicroPython Support

The ESP32 also supports MicroPython, a lightweight implementation of the Python programming language optimized for microcontrollers. MicroPython enables developers to write embedded applications using high-level, readable syntax, significantly reducing development time. This makes it particularly attractive for educational purposes, rapid prototyping, and experimental research.

While MicroPython simplifies application development, it introduces additional memory and performance overhead compared to C/C++-based frameworks. As a result, it is more suitable for applications with moderate computational requirements rather than performance-intensive or real-time-critical systems. Despite these limitations, MicroPython remains a valuable option for researchers focusing on algorithm development and system-level experimentation.

4.5 Development Tools and Debugging Support

The ESP32 software ecosystem is complemented by a wide range of development tools, including integrated development environments (IDEs), compilers, debuggers, and monitoring utilities. Popular IDEs such as Visual Studio Code and Arduino IDE provide user-friendly interfaces for code development and debugging. The availability of serial monitoring, logging mechanisms, and hardware debugging interfaces facilitates efficient testing and performance optimization.

In addition, extensive documentation, open-source libraries, and active developer communities contribute to continuous improvement of the ESP32 software ecosystem. This strong community support accelerates problem-solving and knowledge sharing, which is particularly beneficial for academic research and long-term projects.

4.6 Comparative Overview of Development Frameworks

The availability of multiple development frameworks allows developers to select the most appropriate software environment based on application requirements. ESP-IDF is preferred for low-level control and performance optimization, FreeRTOS enables real-time and multitasking applications, the Arduino framework simplifies rapid development, and MicroPython provides a high-level programming approach.

Overall, the diverse and mature software ecosystem of the ESP32 significantly enhances its flexibility and usability, making it suitable for a wide range of IoT and embedded system applications. This software versatility, combined with robust hardware capabilities, positions the ESP32 as a powerful and adaptable platform for both research and industrial deployments.

5. Applications of ESP32-Based Systems

The architectural flexibility, integrated wireless connectivity, and rich software ecosystem of the ESP32 microcontroller have enabled its widespread adoption across diverse application domains. Researchers and practitioners have extensively utilized ESP32-based platforms to develop low-cost, scalable, and energy-efficient IoT solutions. This section reviews the major application areas in which ESP32 has been successfully deployed, highlighting representative use cases and research trends.

5.1 Smart Agriculture

Smart agriculture is one of the most prominent application domains of ESP32-based systems. In agricultural environments, ESP32 is commonly used for real-time monitoring of soil moisture (Figure 3), temperature, humidity, and environmental conditions [9]. Its integrated Wi-Fi and BLE connectivity enables seamless transmission of sensor data to cloud platforms or local gateways for analysis and decision-making [10].

Several studies have demonstrated ESP32-based smart irrigation systems (Figure 4) that optimize water usage by dynamically controlling irrigation schedules based on sensor feedback. The low cost and energy efficiency of ESP32 make it particularly suitable for deployment in large-scale agricultural fields and remote farming

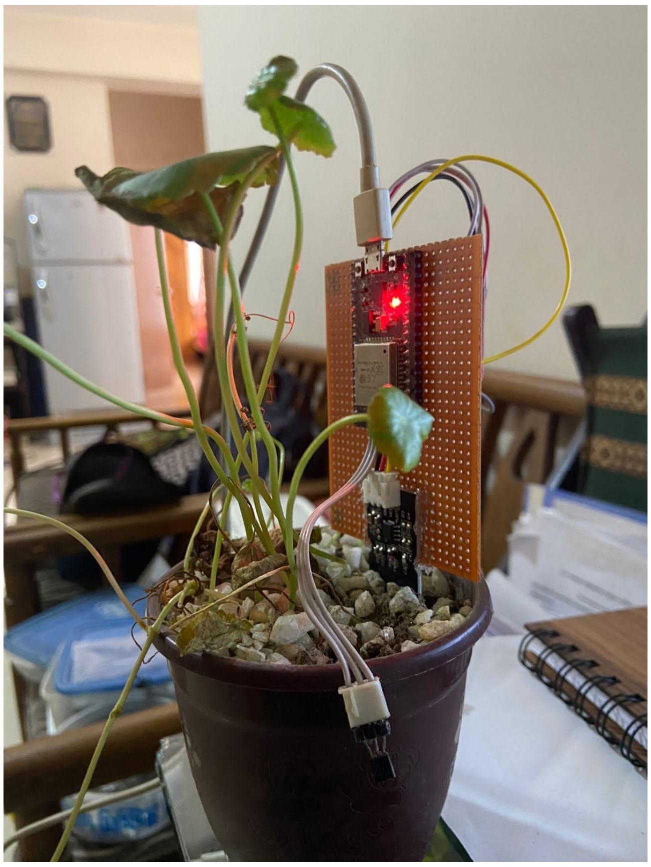

Figure 3: Soil moisture

locations. Additionally, ESP32-based weather stations and crop monitoring systems have been developed to support precision agriculture and data-driven farming practices.

5.2 Healthcare and Wearable Systems

The ESP32 has gained significant attention in healthcare and wearable applications due to its compact size, low power consumption, and BLE support. In healthcare monitoring systems, ESP32 is frequently employed to collect physiological data such as heart rate, ECG (Figure 5), body temperature, oxygen saturation, and motion signals. The collected data can be transmitted wirelessly to mobile devices or remote servers for real-time monitoring and analysis.

ESP32-based wearable devices have been proposed for elderly care, patient monitoring, and fitness tracking. BLE communication enables energy-efficient data exchange, extending battery life in wearable systems. Furthermore, the integration of security features allows secure transmission of sensitive medical data, which is essential for healthcare applications.

5.3 Smart City Applications

Smart city infrastructures rely heavily on distributed sensing and communication systems, where ESP32 plays a vital role as an edge-level processing unit. ESP32-based systems have been implemented for applications such as air quality monitoring, traffic management, smart lighting, and waste management. These systems typically collect environmental data and transmit it to centralized platforms for visualization and policy decision-making [11, 12].

The ability of the ESP32 to operate in both station and access point modes enables flexible network deployment in urban environments. Its compatibility with cloud services further supports large-scale data aggregation and analytics, making it an effective platform for smart city solutions.

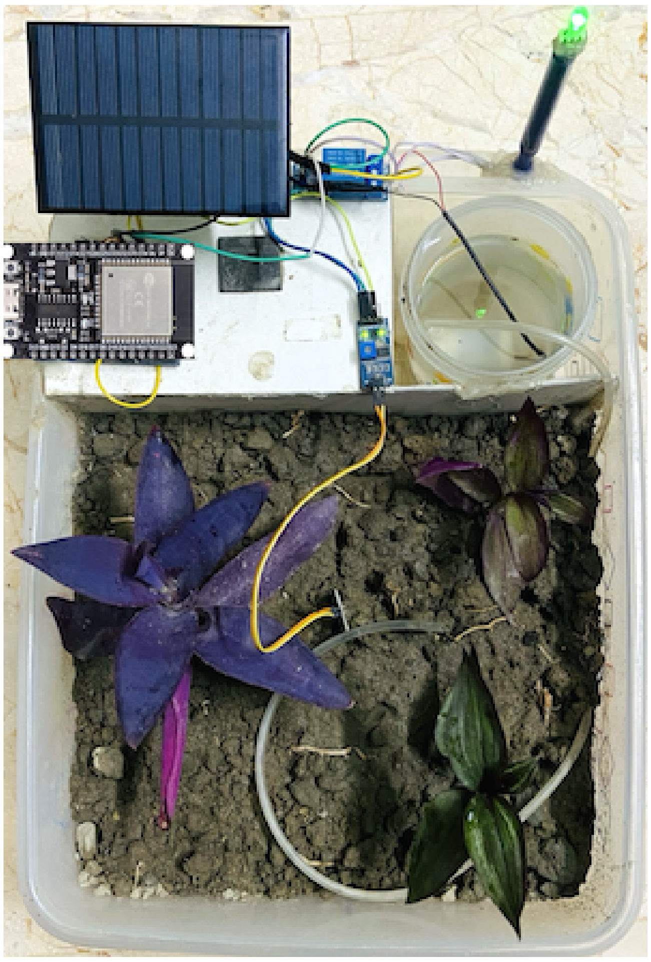

Figure 4: Energy Efficient irrigation systems

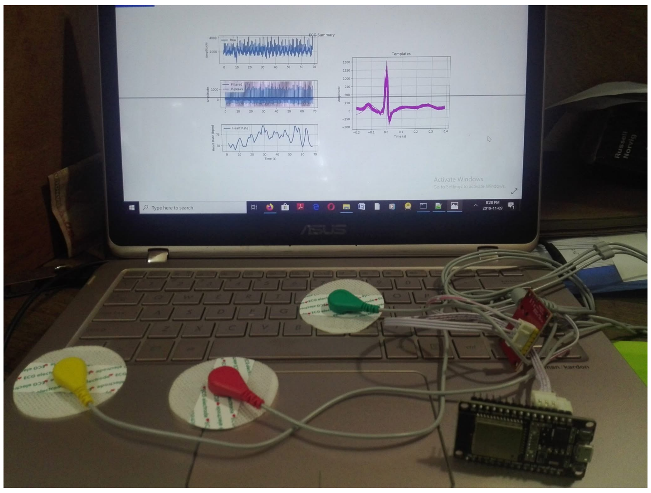

Figure 5: ECG monitoring system

5.4 Industrial Internet of Things (IIoT)

In industrial environments, ESP32 has been adopted for condition monitoring, equipment diagnostics, and predictive maintenance applications [11]. The availability of wired communication protocols such as SPI, I2C, and CAN, combined with wireless connectivity, allows ESP32-based devices to interface with industrial sensors and control systems.

ESP32-based IIoT systems have been developed to monitor parameters such as vibration, temperature (Figure 6), and power consumption in machinery. These systems enable early fault detection and reduce downtime through data-driven maintenance strategies. While ESP32 is not a replacement for high-end industrial controllers, its low cost and flexibility make it suitable for non-critical and supplementary industrial monitoring applications [13].

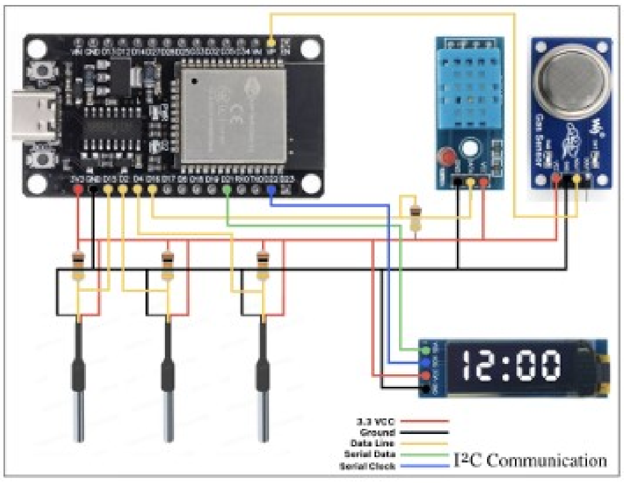

Figure 6: Temperature and Humanity

5.5 Environmental Monitoring

Environmental monitoring represents another important application area for ESP32-based systems. Researchers have utilized ESP32 to develop low-power sensor nodes for monitoring air pollution, water quality (Figure 7), noise levels, and weather conditions (Figure 8). These systems often operate in remote or outdoor environments, where energy efficiency and reliable wireless communication are critical.

ESP32-based environmental monitoring platforms support real-time data collection and long-term deployment, making them suitable for smart environmental management and sustainability initiatives. The use of sleep modes and energy-efficient communication protocols further enhances their applicability in battery-powered deployments.

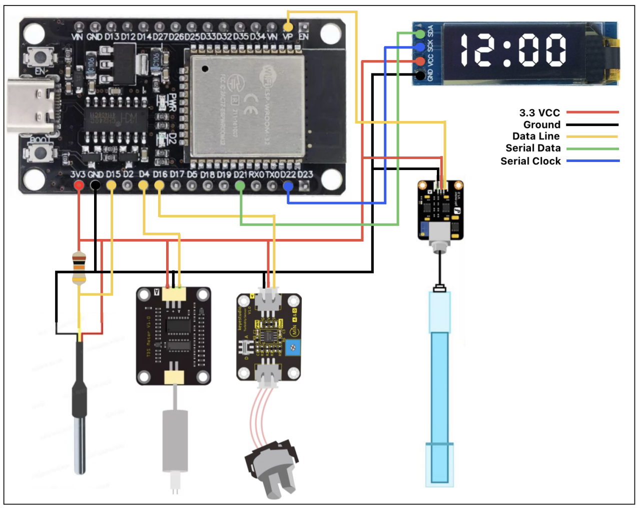

Figure 7: Water Quality Monitoring System

5.6 Smart Home and Consumer Electronics

ESP32 is widely used in smart home automation and consumer electronics due to its ease of integration, wireless connectivity, and compatibility with mobile applications. Typical applications include smart switches, lighting control systems, security devices, and voice-enabled home appliances.

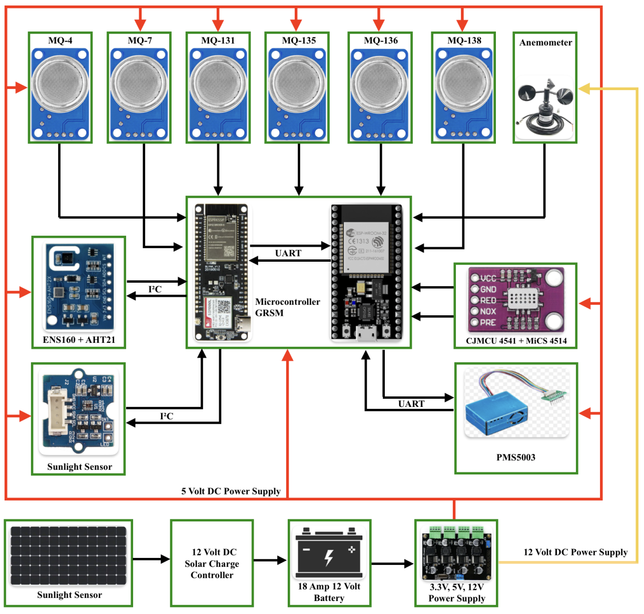

Figure 8: Air Quality

The combination of Wi-Fi and BLE enables ESP32-based devices to interact with smartphones, cloud services, and local networks, providing seamless user experiences. Its extensive peripheral support further facilitates the integration of sensors and actuators in consumer-oriented systems.

5.7 Summary of Application Trends

The reviewed application domains demonstrate that the ESP32 microcontroller serves as a versatile and scalable platform for IoT systems across agriculture, healthcare, urban infrastructure, industry, and consumer electronics. Its widespread adoption in both academic research and real-world deployments highlights its capability to address diverse application requirements.

However, application-specific challenges such as power optimization, security, and real-time performance continue to influence system design choices. These challenges are further discussed in the subsequent section focusing on research limitations and open issues.

6. Comparative Analysis with Other Microcontrollers

Selecting an appropriate microcontroller platform is a key design decision in IoT systems, as it directly influences cost, power consumption, performance, and communication capability. Although the ESP32 provides an attractive balance of integrated connectivity and processing power, it is important to compare it with other commonly used platforms to understand its suitability across different deployment scenarios. This section presents a comparative analysis of ESP32 with representative microcontrollers frequently used in IoT and embedded applications, including ESP8266, Arduino Uno (ATmega328P), Raspberry Pi Pico (RP2040), and STM32-based MCUs.

7. Research Challenges and Limitations

Despite its widespread adoption and versatility, the ESP32 microcontroller is not without limitations. While it offers an attractive balance between performance, connectivity, and cost, several technical challenges remain that influence its suitability for certain classes of IoT and embedded applications. This section critically discusses the key research challenges and limitations associated with ESP32-based systems, focusing on power consumption, security, scalability, real-time performance, and memory constraints.

7.1 Power Consumption Challenges

Power efficiency is one of the most critical concerns in IoT systems, particularly for battery-powered and remote deployments. Although the ESP32 provides multiple power-saving modes such as light sleep, deep sleep, and hibernation, continuous operation of Wi-Fi significantly increases power consumption. Applications that require persistent network connectivity or frequent data transmission may experience reduced battery life despite the availability of low-power modes.

Another challenge lies in the complexity of power optimization. Effective power management on the ESP32 often requires careful firmware design, precise task scheduling, and appropriate use of sleep states. Improper configuration of peripherals or background tasks can negate the benefits of power-saving features. Consequently, achieving ultra-low-power operation remains an active research area, especially for long-term environmental monitoring and large-scale sensor networks.

7.2 Security Challenges

Security is a fundamental requirement for IoT systems due to their exposure to public networks and potential cyber threats. While the ESP32 includes hardware-based security features such as cryptographic accelerators, secure boot, and flash encryption, security implementation largely depends on correct software configuration. In practice, misconfigured firmware, weak authentication mechanisms, and insecure communication channels can expose ESP32-based systems to attacks.

Additionally, the complexity of managing secure over-the-air (OTA) firmware updates introduces further challenges. Ensuring firmware integrity, preventing rollback attacks, and maintaining secure key storage require advanced design practices that are not always adopted in low-cost IoT deployments. As ESP32 devices are increasingly used in critical applications, strengthening end-to-end security remains a key research challenge.

7.3 Scalability and Network Management

Scalability is another important limitation when deploying large numbers of ESP32-based devices. While the microcontroller performs well in small- to medium-scale IoT systems, network congestion and management overhead can become significant in large deployments, particularly when Wi-Fi is used as the primary communication medium.

Managing device provisioning, firmware updates, synchronization, and fault tolerance across hundreds or thousands of ESP32 nodes requires additional infrastructure and system-level coordination. These challenges highlight the need for scalable network architectures, efficient communication protocols, and edge-based data aggregation strategies to reduce bandwidth usage and improve system reliability.

7.4 Real-Time Performance Limitations

Although the ESP32 supports FreeRTOS and dual-core processing, strict real-time determinism is not always guaranteed, especially when wireless communication tasks are active. Wi-Fi and Bluetooth stacks can introduce latency and interrupt overhead, which may interfere with time-critical operations.

As a result, ESP32-based systems may face limitations in applications that require hard real-time constraints, such as high-precision motor control or safety-critical industrial automation. While careful task prioritization and core allocation can mitigate some of these issues, achieving consistent real-time performance remains challenging compared to dedicated real-time microcontrollers.

7.5 Memory Constraints

Memory availability represents another limitation of the ESP32, particularly for applications involving complex algorithms, data buffering, or edge-level intelligence. The on-chip SRAM is limited, and although external flash memory can be used for storage, runtime memory constraints may restrict the implementation of memory-intensive tasks such as machine learning inference or advanced data analytics.

This limitation becomes more pronounced when using high-level development frameworks or interpreted languages, which introduce additional memory overhead. Consequently, memory optimization and efficient resource management are critical considerations for researchers developing advanced ESP32-based applications.

7.6 Summary of Research Challenges

The discussed challenges indicate that while the ESP32 is a powerful and flexible platform for IoT development, its limitations must be carefully considered during system design. Power optimization, secure firmware development, scalable deployment strategies, real-time task management, and memory-efficient programming remain active research areas. Addressing these challenges will be essential for extending the applicability of ESP32-based systems to more demanding and large-scale IoT scenarios.

8. Future Research Directions

The rapid evolution of IoT technologies and increasing application complexity continue to create new research opportunities for ESP32-based systems. While the ESP32 already provides a strong foundation for wireless embedded applications, future research can further enhance its performance, efficiency, and applicability. This section outlines key research directions that can guide future developments and investigations involving the ESP32 microcontroller.

8.1 ESP32 for Edge Intelligence and Lightweight AI

One of the most promising research directions involves integrating edge intelligence and lightweight artificial intelligence (AI) on ESP32-based platforms. With the growing demand for real-time decision-making at the edge, researchers are exploring optimized machine learning models that can operate within the computational and memory constraints of the ESP32. Techniques such as model quantization, pruning, and the use of lightweight neural networks can enable on-device inference for applications such as anomaly detection, activity recognition, and predictive monitoring [3].

Future research may focus on developing efficient AI frameworks and toolchains specifically tailored for ESP32, enabling scalable and energy-efficient edge intelligence without relying heavily on cloud resources.

8.2 Integration with Long-Range Communication Technologies

Although the ESP32 primarily supports short- to medium-range wireless communication, combining it with long-range communication technologies such as LoRa, NB-IoT, or LTE-M represents an important research direction. Such hybrid architectures can extend communication range while maintaining low power consumption, making ESP32-based systems suitable for wide-area monitoring applications, including agriculture, environmental sensing, and smart infrastructure.

Research efforts in this area may focus on communication protocol optimization, adaptive data transmission strategies, and seamless integration between short-range and long-range networks.

8.3 Energy Harvesting and Ultra-Low-Power Operation

Energy harvesting techniques, such as solar, thermal, and vibration-based power generation, offer significant potential for extending the operational lifetime of ESP32-based systems. Future research can investigate energy-aware system designs that dynamically adapt processing, sensing, and communication tasks based on available energy levels.

Optimizing firmware to exploit deep sleep modes, adaptive duty cycling, and event-driven activation can further reduce energy consumption. Such research is particularly relevant for remote and maintenance-free IoT deployments.

8.4 Secure Firmware and Trustworthy IoT Systems

As ESP32-based devices are increasingly deployed in security-sensitive environments, enhancing end-to-end security remains a critical research priority. Future studies may focus on robust secure boot mechanisms, trusted firmware updates, and advanced key management strategies. The development of lightweight cryptographic protocols optimized for resource-constrained devices can further strengthen system security without compromising performance.

8.5 Scalable System Architectures and Device Management

Large-scale ESP32 deployments require efficient system-level management solutions. Future research can explore scalable architectures that support automated device provisioning, monitoring, firmware updates, and fault tolerance. Edge-based aggregation, hierarchical network designs, and intelligent load balancing can reduce communication overhead and improve system reliability.

Standardized frameworks and middleware solutions may also facilitate interoperability between ESP32-based systems and heterogeneous IoT platforms.

8.6 Enhancing Real-Time and Deterministic Performance

Improving real-time performance remains an important research direction for expanding the use of ESP32 in time-critical applications. Future work may investigate optimized task scheduling strategies, core isolation techniques, and hybrid real-time architectures that minimize interference from wireless communication stacks.

Exploring tighter integration between FreeRTOS and application-level scheduling mechanisms could enable more deterministic execution, broadening the applicability of ESP32 in industrial and control-oriented systems.

8.7 Summary of Future Directions

The identified research directions highlight the significant potential for extending the capabilities of ESP32-based systems. Advancements in edge intelligence, communication integration, energy efficiency, security, scalability, and real-time performance can substantially enhance the role of ESP32 in next-generation IoT deployments. Continued research in these areas will contribute to more intelligent, secure, and sustainable embedded systems.

Conclusion

This paper presented a comprehensive review of the ESP32 microcontroller, focusing on its architecture, communication protocols, software ecosystem, application domains, and associated research challenges. The ESP32 has emerged as a versatile and widely adopted platform for IoT and embedded systems due to its dual-core processing capability, integrated Wi-Fi and Bluetooth connectivity, rich peripheral support, and flexible software frameworks.

Through detailed analysis, it has been shown that the ESP32 effectively bridges the gap between low-cost microcontrollers and more powerful embedded platforms, enabling scalable and connected solutions across diverse domains such as smart agriculture, healthcare monitoring, smart cities, industrial IoT, and environmental sensing. Its ability to support multiple communication protocols [14, 15] and development environments allows researchers and practitioners to tailor system designs according to application-specific requirements.

Despite its strengths, the review also highlighted several limitations that continue to influence ESP32-based system design. Challenges related to power consumption under continuous wireless operation, security configuration complexity, scalability in large deployments, real-time performance constraints, and limited on-chip memory remain active areas of research. Addressing these challenges is essential for extending the applicability of ESP32 to more demanding and mission-critical applications.

The discussion of future research directions emphasized the growing potential of ESP32 in emerging areas such as edge intelligence, hybrid communication architectures, energy harvesting, and secure IoT systems. Continued advancements in firmware optimization, lightweight AI techniques, and scalable system management are expected to further enhance the performance and reliability of ESP32-based platforms.

In conclusion, the ESP32 microcontroller represents a powerful and flexible foundation for modern IoT development. With ongoing research and innovation, it is well-positioned to remain a key enabling technology for next-generation embedded and IoT systems.

References

[1] The Internet of Things with ESP32. http://esp32.net

[2] ESP32 Series Datasheet Version 5.2. https://documentation.espressif.com

[3] Raghuwanshi, M., Borkar, P., Jhaveri, R.H., & Raut, R. (Eds.),"Parallel and High-Performance Computing in Artificial Intelligence (1st ed.)",2025, Auerbach Publications. https://doi.org/10.1201/9781003425458

[4] Md. Hafizur Rahman, M.Naderuzzaman, "A Comprehensive Review of M2M Communication Protocols", Open Access Journal on Engineering Applications (OAJEA), Volume No. 01, Issue No. 01, Page 1-13, July, 2025. https://doi.org/10.64886/oajea.0101.001

[5] Md Hafizur Rahman, M. Naderuzzaman, Md Masud Reza "IoT Devices: Classification, Features, and Trends in Modern IoT Systems", Open Access Journal on Engineering Applications (OAJEA), Volume No. 01, Issue No. 02, Page 1-8, September, 2025. https://doi.org/10.64886/oajea.0102.001

[6] Dhanalakshmi, M., et al. ”Advanced Machine Learning Innovations in Embedded Systems and Narrowband Internet of Things (NB-IoT) Devices.” In Integrating Artificial Intelligence Into the Energy Sector, edited by Abdelkader Mohamed Sghaier Derbali, 331-356. Hershey, PA: IGI Global Scientific Publishing, 2025. https://doi.org/10.4018/979-8-3693-7112-1.ch016

[7] M. Alioto, Ed., Enabling the Internet of Things. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-51482-6

[8] Kumar, M.S.V., Raju, K.S., Rajakumar, K., & Saravanakumar, S. (Eds.), ”A Study on Next-Generation Materials and Devices (1st ed.)”, CRC Press, (2025). https://doi.org/10.1201/9781003675259

[9] Rahman, M. H., Reza, M. M., Ferdous, R., Naderuzzaman, Kashem, ”Development of a IoT Based Thermologger for Real-Time Temperature Monitoring”, Internet of Things and Cloud Computing, 13(2), 28-37. https://doi.org/10.11648/j.iotcc.20251302.11

[10] Morchid, A., Qjidaa, H., Alami, R.E, ”Smart irrigation-based internet of things and cloud computing technologies for sustainable farming”. Sci Rep 16, 5293, 2026. https://doi.org/10.1038/s41598-026-35810-0

[11] Md Hafizur Rahman “A Comprehensive Review of Edge Computing: A Perspective of IoT”, Open Access Journal on Engineering Applications (OAJEA), Volume No. 01, Issue No. 01, Page 29-37, July, 2025. https://doi.org/10.64886/oajea.0101.004

[12] Dhatterwal, J.S., Berwal, P., Kaswan, K.S., & Goyal, R, ”IoT Applications in Geotechnical Engineering (1st ed.)”. CRC Press. 2025. https://doi.org/10.1201/9781003537625

[13] Md. Hafizur Rahman,Dr. Zohra Khatun, M.Naderuzzaman, “A Smart IoT-Based Environmental Monitoring System for Clinical Labs: Focus on Temperature, Humidity and Air Quality”, Open Access Journal on Engineering Applications (OAJEA), Volume No. 01, Issue No. 01, Page 14-17, July, 2025. https://doi.org/10.64886/oajea.0101.002

[14] Muhammad Zohaib, Seyed-Sajad Ahmadpour, Hadi Rasmi, Angshuman Khan, Nima Jafari Navimipour, A low-latency and area-efficient QCA-based quantum-dot design for next-generation digital sustainable systems, Sustainable Computing: Informatics and Systems, Volume 48, 2025, 101204, ISSN 2210-5379. https://doi.org/10.1016/j.suscom.2025.101204

[15] M. Alkhayyal and A. Mostafa, “Recent Developments in AI and ML for IoT: A Systematic Literature Review on LoRaWAN Energy Efficiency and Performance Optimization,” Sensors, vol. 24, no. 14, p. 4482, Jul. 2024,https://doi.org/10.3390/s24144482