UDC 004.05 **doi: 10.26906/SUNZ.2025.4.51-55**

Oleh Drozd, Oleksii Sytnyk, Maksym Nesterenko

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

MODELS AND METHODS FOR BUILDING A DECENTRALIZED IOT SYSTEM – ESP32-BASED SENSORS USING THE MQTT PROTOCOL

Abstract. Relevance. The relevance is due to the growing interest in ESP32-based IoT systems, the limitations of centralized systems, and the potential of the MQTT protocol. **Article subject.** Methods for implementing decentralized WSNs and their components. **Purpose.** The article's main purpose is to propose a decentralized architecture for Wireless Sensor Networks (WSNs). It aims to shift from the traditional centralized model, where all sensor nodes depend on a single hub, to a more resilient system where each node possesses direct internet access. The goal of this proposed design is to significantly boost the reliability and overall productivity of the network by eliminating the single point of failure inherent in a centralized setup. **Article results.** The article culminates in a specific and cost-effective recommendation of components to build a decentralized sensor node. These include: the ESP32 module, communication modules, a development environment, a data transfer protocol, and brokers to ensure its operation. **Conclusions.** Based on an analysis of available hardware and software, the article concludes that building a reliable and cost-effective decentralized Wireless Sensor Network (WSN) is highly feasible.

Keywords: Decentralization, WSN, ESP32, MQTT, Wi-Fi, 2G, LTE, Ethernet.

Introduction

WSN technology is used in various applications, from smart homes to the military industry. What they all have in common is the need to collect data in complex and challenging conditions. They have been developing since the 1950s, and during this time, many technologies have been developed, which allow this industry to grow. New hardware components, data collection methods, and routing protocols are the main development forces for this field of knowledge.

However, the vast majority of them use a centralized organization scheme, which provides for the presence of a hub that transmits data to the Internet. This article aims to introduce a decentralized construction scheme into this field to improve the reliability and productivity of the system.

Main part

Review of existing solutions. In centralized WSNs, all nodes form a mesh topology network through which data is transmitted to a sink, which has access to the internet (Fig. 1) [1].

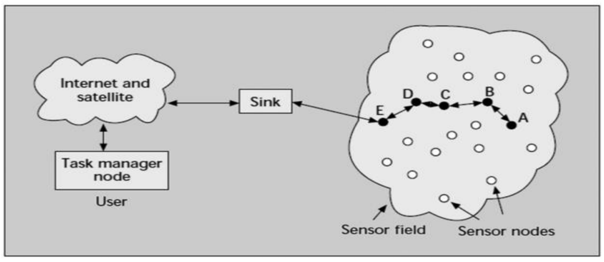

Fig. 1. Centralized WSN

The transition to a decentralized network architecture necessitates that each sensor node be equipped with direct internet access. This configuration allows each node to operate as an independent hub, which enhances network stability and resilience by eliminating dependence on a single point of failure – the central sink.

The fundamental hardware design of the nodes remains consistent with their centralized counterparts (Fig. 2). The primary modification is the replacement of the IEEE 802.15.4 communication module with alternatives such as Wi-Fi, Ethernet, 2G, 3G or LTE. The selection among these modules is contingent upon the specific application requirements and the availability of suitable network coverage.

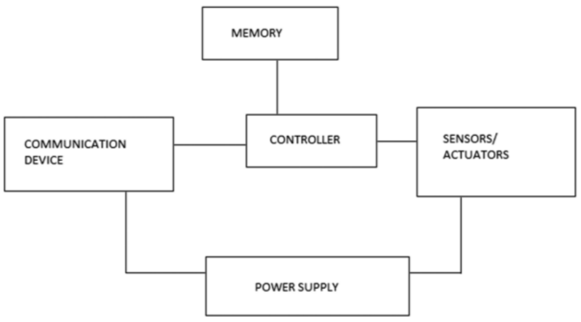

Fig. 2. Node structure

For each data transfer method, there are ready-made solutions in the form of developer kits that already have the necessary communication modules built in, such as the ESP32 4G LTE Gateway (Gen.1) (Fig. 3) [2].

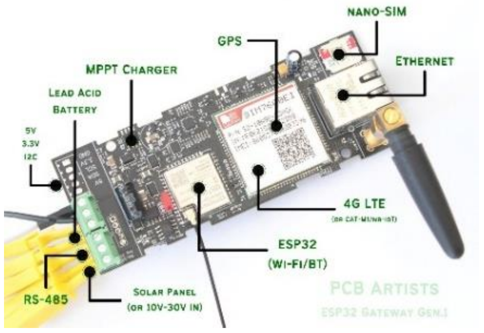

Fig. 3. ESP32 4G LTE Gateway (Gen. 1)

They contain all the modules that can provide decentralized operation of the node and completely eliminate the assembly process, but they are all prohibitively expensive compared to separately purchased boards and modules. Their use is reasonable during debugging, but when implemented in each node, it becomes too expensive.

Ethernet shield. Fortunately, there are separate Ethernet shields that can be connected to a microcontroller board if it has an SPI interface (Fig. 4).

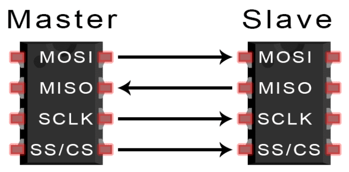

Fig. 4. SPI interface

The most popular ones at the moment are the W5500 and ENC28J60 (Fig. 5). The main and most important difference is the W5500's hardware TCP/IP stack. All the complex work of processing network protocols (TCP, UDP, IP) is performed directly by the W5500 chip. This significantly reduces the load on the host microcontroller (e.g., ESP32), allowing it to focus on application tasks. The W5500 also offers 8 independent hardware sockets for simultaneous connection management [3].

In contrast, the ENC28J60 is a standalone Ethernet controller that includes MAC (media access control) and PHY (physical layer) but does not have a built-in TCP/IP stack. This means that upper-layer protocols (IP, TCP, UDP) must be implemented in software on the host microcontroller in order to operate on the network.

This places a significantly greater load on the host processor and requires more memory resources, which may limit the use of the ENC28J60 in resource-intensive applications or when using less powerful microcontrollers [4].

In addition, the W5500 supports speeds of 100 Mbps (10BaseT/100BaseTX), which is ten times higher than the maximum speed of the ENC28J60 (10BASE-T, 10 Mbps). The W5500 also has four times more built-in buffer memory (32 KB vs. 8 KB), which contributes to more efficient network packet processing. Despite all this, the W5500 turns out to be cheaper than ENC28J60.

Fig. 5. WIZnet W5500 (on the left) and ENC28J60 (on the right)

SIM module. There are two SIM modules that would be most suitable for a decentralized sensor. These are the SIM800L and SIM7600G modules (Fig. 6).

The comparison between the SIM7600G and the SIM800L modules reveals significant architectural differences driven by their supported network technologies. The SIM7600G is a multi-band solution that supports LTE-FDD/LTE-TDD/HSPA+ and GSM/GPRS/EDGE networks, classifying it as a modern high-throughput device. Consequently, its maximum data speed via LTE CAT1 reaches up to 10 Mbps downlink, and speeds can climb to 42.0 Mbps downlink when operating in HSPA+ mode. Conversely, the SIM800L is a Quad-band GSM/GPRS module, and its maximum data transfer rate using GPRS is restricted to 85.6 kbps. Regarding interfaces and power, both modules operate on a comparable voltage range (e.g., SIM7600G uses 3.4V ~ 4.2V and SIM800L uses 3.4V ~ 4.4V); however, the SIM7600G offers a higher-speed data interface, including USB 2.0 with speeds up to 480 Mbps, while the SIM800H/SIM800L primarily utilizes a full modem serial port (UART) and requires the power supply to be able to provide current up to

2.0A during its transmit burst and for long-term stable operation, a DC-to-DC converter will be required [5][6].

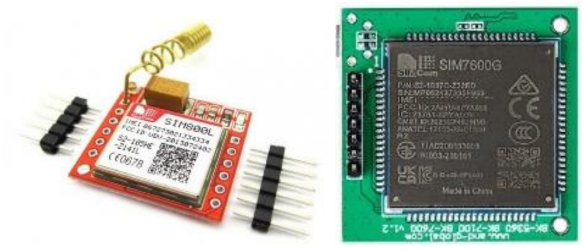

Fig. 6. SIM800L (on the left) and SIM7600G (on the right)

Both of these modules are suitable because they can work almost anywhere. The SIM800L supports GSM/GPRS networks, which have universal coverage (Fig. 7) [7]. The SIM7600G also supports more modern network versions, allowing it to work much more efficiently than the SIM800L module. However, these modules differ significantly in price. While the SIM800L is extremely cheap, the SIM7600G is extremely expensive.

ESP32-based board. Various versions of ESP are often used as controllers due to their simplicity and low cost. This accessibility has given rise to a close-knit community that develops and distributes open source software, shares knowledge, and creates extensive databases of projects and guides. As a result, there are many ready-made solutions to many problems, which greatly simplifies working with this platform. They can also be quite energy efficient when configured correctly.

The ESP-WROOM-32-based board (Fig. 8) is ideal for node tasks.

The ESP-WROOM-32 module is one of the most popular and widely used components for ESP32-based boards. Its popularity is due to the fact that it is a complete and ready-to-use solution. A single compact board combines a dual-core ESP32 microcontroller with a clock speed of up to 240 MHz, built-in Wi-Fi and Bluetooth modules, flash memory, a quartz resonator, and even an antenna.

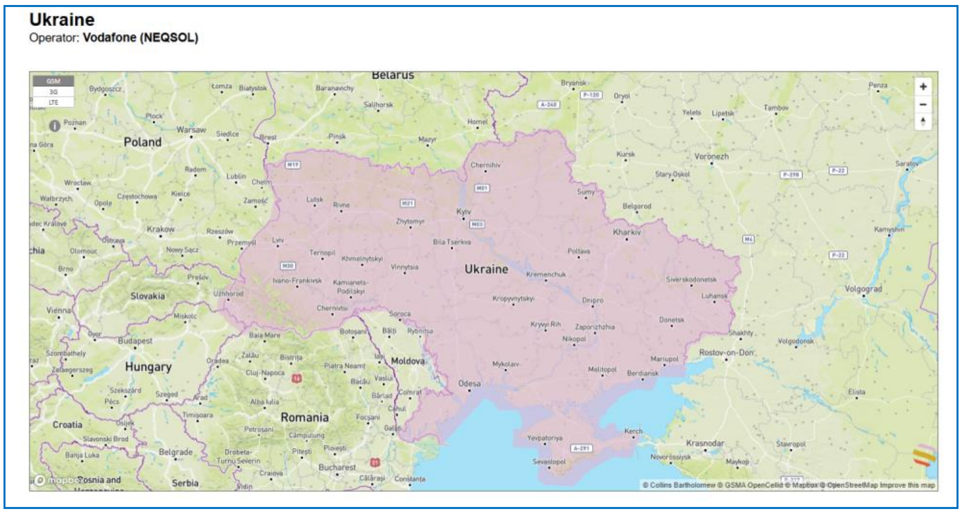
Ukraine
Operator: Vodafone (NEQSOL)
GSM
3G
LTE
Belarus
Poland
Ukraine
Slovakia
Hungary
Moldova
Romania
Croatia
Serbia
© Collins Bartholomew © GSMA OpenCellid © Mapbox © OpenStreetMap Improve this map

Fig. 7. GSM Network coverage on the example of Ukraine

Fig. 8. ESP-WROOM-32-based board

This eliminates the need for developers to design complex high-frequency circuitry, which greatly simplifies and speeds up the device creation process [8].

A key advantage of the ESP32 is its exceptional hardware versatility, making it a perfect central controller for diverse communication needs. Beyond its native Wi-Fi capability, it is fully equipped to handle both wired and cellular connections. It supports the SPI (Serial Peripheral Interface) protocol (Fig.9), which is essential for high-speed data exchange with Ethernet shields like the W5500. Furthermore, the ESP32 has multiple UART (Universal Asynchronous Receiver-Transmitter) ports (Fig.8), which provide the standard serial interface needed to communicate with and control SIM modules like the SIM800L. This built-in support for multiple communication protocols ensures seamless integration with all the necessary connectivity modules for a decentralized sensor node.

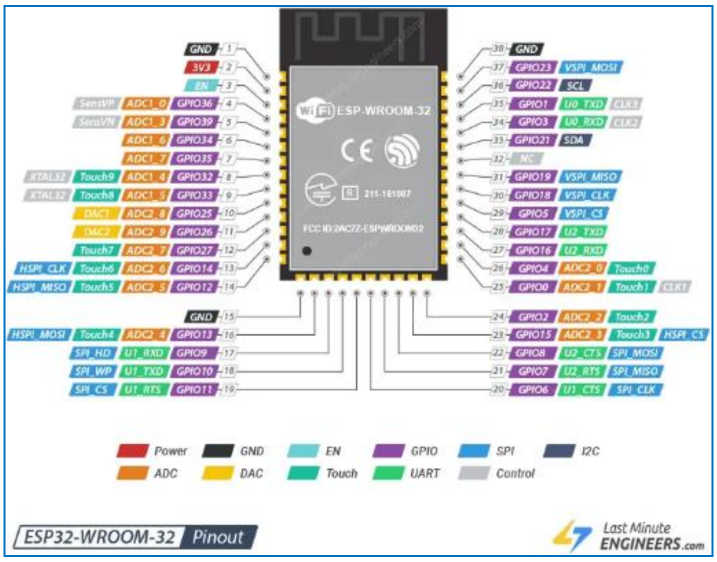

Fig. 9. ESP-WROOM-32 pinout

Data transfer protocol. Protocol that commonly used for this purpose is MQTT. MQTT's lightweight, publish-subscribe model (Fig. 10) is particularly well-suited for resource-constrained devices, allowing them to efficiently transmit data without the overhead of a centralized gateway. This software layer is critical for enabling the decentralized architecture, ensuring seamless and reliable data flow from each individual node to its final destination [9].

Although MQTT itself is not heavyweight, it does have an extension version – MQTT-S, which is designed to run on low-performance, battery-powered sensor/actuator devices and operate in WSNs with limited bandwidth, such as ZigBee-based networks [10].

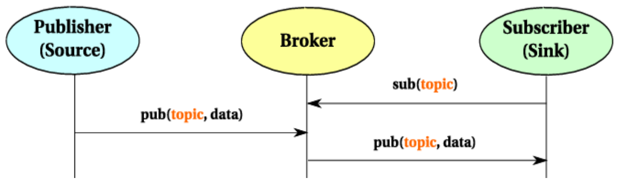

Fig. 10. MQTT operating principle

Also, MQTT has such feature as Quality of Service. MQTT Quality of Service (QoS) defines the message delivery reliability guarantee level. This framework balances reliability and network efficiency. The three levels are:

– QoS 0 ("At most once"), a best-effort, "fire and forget" delivery;

– QoS 1 ("At least once"), which requires a PUBACK acknowledgment to ensure delivery;

– QoS 2 ("Exactly once"), which guarantees single delivery using a four-step handshake for critical communications [11].

MQTT brokers can be broadly categorized as either public or private, each offering distinct advantages depending on the project's requirements for simplicity, control, and scale. Public brokers, or more accurately, managed IoT platforms, are designed to drastically simplify the development and configuration process, though often at the cost of flexibility and raw performance. Blynk is a prime example of this category [12]. It operates as a low-code service that allows you to implement a decentralized network of sensors with minimalistic sketches. Its primary strength lies in its abstraction of complexity; developers can simply connect their board to the Blynk library, and the platform's cloud server handles device authentication, data routing, and provides a polished, pre-built mobile application with a drag-and-drop interface for creating dashboards [13]. This makes it an exceptional choice for beginners, rapid prototyping, and projects where a user-friendly interface is needed quickly without any backend or app development.

For applications requiring greater security, performance, and data ownership, private brokers are the necessary solution. These can be deployed in two main ways: on a self-hosted local server or using a dedicated cloud service. A local server, such as a Raspberry Pi running an open-source broker like Mosquitto, grants you complete and absolute control over your data and security [14]. All information remains within your private network, which is critical for sensitive applications and ensures the system continues to operate flawlessly without an internet connection. This approach eliminates recurring costs and provides the lowest possible latency for local communication. However, it places the responsibility of setup, security hardening (like configuring TLS encryption), and maintenance squarely on the developer.

On the other end of the spectrum are enterprise-grade cloud services like AWS IoT Core or Microsoft Azure IoT Hub [15]. These platforms are engineered for massive scalability, high reliability, and deep integration into a broader ecosystem of cloud services. They provide simple and secure device connectivity, robust management for fleets of devices, and a choice of protocols with end-to-end encryption. Their true power lies in their ability to seamlessly pipe IoT data into other services for storage, real-time analytics, and machine learning, making them the ideal foundation for large-scale commercial and industrial projects where data processing and robust infrastructure are paramount. While they involve a pay-as-you-go cost model and depend on internet connectivity, they offload the immense challenge of building and maintaining a globally scalable and secure infrastructure.

Review conclusion

Based on the above-mentioned ready-made solutions, I consider the following components to be the most optimal.

– WIZnet W5500, as an Ethernet shield – its main advantage is that it can handle socket processing without overloading the microcontroller, which significantly affects the speed of operation. It is also small and ideal for node tasks. The price is even lower than that of competitors.

– SIM800L, as a mobile communication module – it will work even in areas with poor coverage, which is its main advantage. This module has already become popular and there is a lot of documentation available for it. It is also very inexpensive. One of its disadvantages is that it requires a very specific power supply for stable operation. But if this is provided, the module will deliver good results.

– Board based on the ESP-WROOM-32 module, as a microcontroller – this module eliminates the need to design the entire board yourself. Such boards already have everything you need to work, including a Wi-Fi module, so there is no need to purchase it separately, as is the case with Arduino. It also supports all the technologies needed to achieve node decentralization, has sufficient computing power, and supports energy-saving modes.

– Arduino IDE, as a development environment – is the ideal environment for developing on ESP32. It combines ease of use, a large number of libraries, a built-in compiler, a port monitor, and a programmer.

– MQTT, as a data transfer protocol – this protocol is easy to use, uses small headers, has a convenient publisher/subscriber model, and offers different QoS levels.

– Raspberry Pi, AWS, or Blynk as an MQTT broker. Raspberry Pi is an excellent choice if you already have one available, as it allows you to avoid unnecessary expenses. AWS is an excellent choice if you don't have a Raspberry Pi. It provides simple and secure device connectivity to the cloud, reliable management and scaling, as well as a choice of protocols and end-to-end encryption. Blynk is the best choice for those who are not familiar with programming and want a convenient, ready-to-use system.

General conclusions

Based on an analysis of available hardware and software, the article concludes that building a reliable and cost-effective decentralized Wireless Sensor Network (WSN) is highly feasible. The optimal configuration involves using an ESP-WROOM-32 based board as the core microcontroller due to its integrated features and strong community support. For connectivity, the WIZnet W5500 is recommended for wired Ethernet applications because of its superior performance with a hardware TCP/IP stack, while the inexpensive SIM800L module is ideal for cellular communication thanks to its universal coverage. This hardware stack is best supported by the MQTT protocol for its lightweight and efficient data transfer, developed within the Arduino IDE. The choice of an MQTT broker is flexible, ranging from a local Raspberry Pi for full control, a scalable cloud solution like AWS for large projects, or a simple platform like Blynk for ease of use, collectively offering a versatile framework for creating robust, decentralized sensor nodes.

References

1. A Complete Guide to Wireless Sensor Networks / A. Dumka et al. Boca Raton, FL 33487, USA : CRC Press, 2019. 357 p. DOI: https://doi.org/10.1201/9780429286841-2
2. PCBArtists. ESP32 4G LTE Gateway (Gen.1) URL: https://pcbartists.com/product/esp32-4g-lte-gateway-gen1/
3. Wiznet. W5500 datasheet. URL: https://docs.wiznet.io/img/products/w5500/W5500_ds_v110e.pdf
4. Microchip. ENC28J60 datasheet. URL: https://ww1.microchip.com/downloads/en/devicedoc/39662a.pdf
5. SIMCOM. SIM800L datasheet. URL: https://www.laskakit.cz/user/related_files/sim800l_v2_1.pdf
6. SIMCOM. SIM7600G datasheet URL: https://www.tme.com/Document/29390ba2a617ac7afa0f581de3b295ee/SIM7600G.pdf
7. GSMA. Network coverage maps. URL: https://www.gsma.com/coverage/
8. Espressif. ESP-WROOM-32 datasheet URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
9. AWS. What is MQTT? URL: https://aws.amazon.com/what-is/mqtt/
10. Urs Hunkeler; Hong Linh Truong; Andy Stanford-Clark (2008). MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. DOI: https://doi.org/10.1109/COMSWA.2008.4554519
11. https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ – MQTT QoS
12. Blynk. Low-code IoT cloud platform with user experience at its core. URL: https://blynk.io/
13. Blynk. Github. URL: https://github.com/blynkkk/blynk-library
14. Randomnerdtutorials. How To Install Mosquitto MQTT Broker on Raspberry Pi. URL: https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi
15. AWS. IoT Core. URL: https://aws.amazon.com/iot-core/

Received (Надійшла) 01.08.2025
Accepted for publication (Прийнята до друку) 05.11.2025

Відомості про авторів / About the Authors

Дрозд Олег Юрійович – аспірант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Oleh Drozd – PhD student, Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: oleh.drozd@nure.ua; ORCID Author ID: http://orcid.org/0009-0007-4285-4505.

Ситник Олексій Вячеславович – аспірант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Oleksii Sytnyk – PhD student, Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: oleksii.sytnyk@nure.ua; ORCID Author ID: http://orcid.org/0009-0000-8257-0426.

Нестеренко Максим Андрійович – студент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Maksym Nesterenko – student, Department of Electronic Computers, Kharkiv National University of Radio Electronics Kharkiv, Ukraine;
e-mail: maksym.nesterenko1@nure.ua; ORCID Author ID: https://orcid.org/0009-0006-5385-1652.

Моделі та методи побудови децентралізованої системи Інтернету речей – датчики на основі ESP32 з використанням протоколу MQTT

О. Ю. Дрозд, О. В. Ситник, М. А. Нестеренко

Анотація. Актуальність. Актуальність зумовлена зростаючим інтересом до систем IoT на базі ESP32, обмеженнями централізованих систем та потенціалом протоколу MQTT. **Об’єкт дослідження:** Методи реалізації децентралізованих бездротових сенсорних мереж (WSN) та їх компонентів. **Мета статті:** Основна мета статті – запропонувати децентралізовану архітектуру для бездротових сенсорних мереж (WSN). Вона спрямована на перехід від традиційної централізованої моделі, де всі сенсорні вузли залежать від єдиного хаба, до більш стійкої системи, де кожен вузол має прямий доступ до Інтернету. Кінцевою метою запропонованої конструкції є значне підвищення надійності та загальної продуктивності мережі шляхом усунення єдиної точки відмови, властивої централізованій конфігурації. **Результати дослідження.** Стаття завершується конкретною та економічно ефективною рекомендацією щодо компонентів для побудови децентралізованого сенсорного вузла. До них належать: модуль ESP32, модулі зв'язку, середовище розробки, протокол передачі даних та брокери для забезпечення його функціонування. **Висновки.** На основі аналізу доступного обладнання та програмного забезпечення в статті робиться висновок, що побудова надійної та економічно ефективної децентралізованої бездротової сенсорної мережі (WSN) є цілком реальною.

Ключові слова: децентралізація, WSN, ESP32, MQTT, Wi-Fi, 2G, LTE, Ethernet.