ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 5 (2), 2025

https://doi.org/10.23939/ictee2025.02.

DEVELOPMENT OF EMBEDDED SOFTWARE FOR ESP32-BASED LORA MODULES WITH ADAPTIVE CONFIGURATION AND LINK QUALITY MONITORING

Yu. Shkoropad [ORCID: 0009-0005-7892-6799], **H. Beshley** [ORCID: 0000-0001-5392-3499]

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine

Corresponding author: M. Beshley (e-mail: mykola.i.beshlei@lpnu.ua)

(Received 30 July 2025)

The article describes a new approach to developing embedded software for LoRa modules based on the ESP32 microcontroller. The main idea behind the work is to create universal firmware with a minimalist architecture and advanced configuration options that ensures reliable peer-to-peer data exchange. The developed system uses a simplified text command format (COMMAND;PARAM=VALUE) instead of JSON, which reduces computational costs and speeds up processing. This simplifies integration into application solutions and increases the efficiency of hardware resource utilization. The firmware integrates a delivery confirmation (ACK) mechanism with retransmission in case of packet loss, which increases the reliability of the communication channel. Additionally, the CONFIG_SYNC command is implemented for automatic synchronization of parameters between nodes, which ensures stability in dynamic conditions. The proposed approach also includes a PING/PING_ACK function, which, in addition to checking connection availability, provides diagnostic characteristics, including RSSI, SNR, TOA, DELAY, and data transfer rate. It is possible to transmit large messages using a packet segmentation and aggregation algorithm that overcomes the hardware limitations of the LoRa SX1276 chip. During the study, the firmware was experimentally tested with variations in key parameters: spreading factor, bandwidth, coding rate, transmission power, and preamble length. The results confirmed the patterns of influence of these parameters on delay, speed, RSSI, and signal-to-noise ratio, which made it possible to form practical recommendations for optimizing the system. The proposed solution combines ease of use, configuration flexibility, and communication quality assessment tools, providing a balance between performance and scalability. Further development involves the integration of artificial intelligence modules, in particular reinforcement learning, for automatic selection of optimal parameters in real time, which opens up prospects for the creation of intelligent self-configuring wireless systems.

Key words: *LoRa, ESP32, embedded software, command protocol, automatic synchronization, communication quality assessment.*
UDC: 621.391

Introduction

In modern wireless data transmission systems, LoRa technology has become widespread due to its combination of low power consumption and long communication range [1]. At the same time, existing software solutions for ESP32 microcontrollers with LoRa modules are often characterized by overly complex architecture, unnecessary functionality, or dependence on external services, which creates sig-

nificant obstacles for use in application development with limited resources. Typical examples are multifunctional firmware with support for mesh networks, mobile applications, or LoRaWAN infrastructure [2]. Such solutions require additional configuration, third-party services, or do not provide sufficient transparency at the data transmission level. Among the most common options are Meshtastic, LoRaWAN Stack, and Simple LoRa firmware, which demonstrate different approaches to organizing data transmission and building network infrastructure [3].

Meshtastic is firmware that implements a LoRa-based mesh network, focused on creating autonomous communication networks without a central server. The advantage of Meshtastic is its high routing reliability and support for mobile applications for convenient management. However, its complexity and redundancy of functions often complicate its use in simple devices, and the need for synchronization with a mobile application reduces autonomy.

LoRaWAN Stack is a set of firmware that provides support for the LoRaWAN protocol for connecting to global networks such as The Things Network (TTN). The main advantage is a standardized protocol that guarantees compatibility with a wide ecosystem. The disadvantages are dependence on network infrastructure, complexity of settings, and increased hardware requirements, which are not always suitable for simple direct transmission scenarios.

Simple LoRa is simplified firmware focused on basic data transmission between two nodes. It has minimal functionality, which ensures quick and easy integration, but does not support advanced features, including the lack of a mechanism for confirming received messages (ACK). The lack of built-in encryption creates a risk of data interception, which is critical for secure applications. The settings in this firmware are hard-coded, so changing them requires recompiling and reloading the code, which makes it difficult to adapt to different conditions. In addition, Simple LoRa does not support multi-channel operation, adaptive transmission power, or power-saving features, which limits its effectiveness in complex network environments and negatively affects the battery life of devices.

The paper proposes its own compact and flexible software that eliminates these limitations and provides a number of important advantages. Its architecture is based on the use of simple text commands instead of formats such as JSON, which significantly speeds up processing and integration into third-party applications. The user can configure all module parameters, and built-in synchronization mechanisms ensure consistency of configurations between different devices. Additionally, connection quality verification functions are provided, allowing for quick assessment of transmission channel reliability. Thanks to its modular structure, the software can be further integrated into artificial intelligence-oriented solutions and extended to work not only with LoRa, but also with other wireless technologies, such as Wi-Fi, LTE, BLE, or GSM.

Thus, the developed solution combines ease of use with extensive adaptability and scalability, making it suitable for both basic and more complex data transmission systems.

2. Development of unique embedded software for LoRa module

The main goal of the development is to develop universal and easy-to-use firmware for LoRa modules based on ESP32, which allows effective device control via a serial USB (UART) interface. The software should provide fast and intuitive connection without complex settings, implementing the “plug and play” concept. It is important that the same firmware code works on both the transmitter and receiver sides (peer-to-peer), thus simplifying system scaling. The firmware should provide the user with the ability to flexibly configure the main parameters that directly affect the transmission speed, connection quality, and power consumption. Key features include support for ACK message acknowledgment with the ability to retransmit in case of data loss, which increases communication reliability. It also provides real-time synchronization of settings between two modules with the ability to automatically restore the last working parameters after temporary communication loss. This solution will ensure stable system operation even in difficult conditions of radio interference and minimize the need for manual intervention. The LILYGO TTGO LoRa32 V3 module with an integrated LoRa module based on the SX1276 chip (Fig. 1) was selected for the development and testing of our own firmware.

This module combines an ESP32 microcontroller, which provides sufficient computing power and multifunctionality, with an SX1276 radio frequency module for data transmission in the 433/868/915 MHz band. The main technical characteristics of the module under study are the use of an ESP32 Dual-Core microcontroller with a clock frequency of up to 240 MHz, which provides a high level of computing power and allows the implementation of complex data processsing algorithms in real time.

Fig. 1. LILYGO TTGO LoRa32 V3

The communication subsystem is based on the Semtech SX1276 LoRa chip, which supports the configuration of the main modulation parameters (Spreading Factor, Bandwidth, Coding Rate), allowing the system to be adapted to different usage scenarios, balancing power consumption, speed, and transmission range. A USB-UART interface is used for integration and programming, which simplifies the debugging process and ensures convenient device control. The module supports power supply from both a USB port and an external battery, which increases its autonomy and expands the scope of practical application. Additional hardware includes an OLED display for displaying service information and buttons for basic local control. The cost of the module at the time of the study is about $ 20–25, making it an economically viable choice for scientific experiments, prototyping, and implementation in applied projects. Thanks to active support from the international developer community and the availability of open libraries, this module can be considered the optimal hardware solution for research and development of applications based on LoRa technology.

The embedded software was developed using the PlatformIO environment, which provides convenient dependency management, build automation, and flexible integration with the ESP32 hardware platform. During the development of the embedded software, a simple and efficient text format for exchanging commands between devices was defined. Messages can have one of two structures:

- Without parameters – COMMAND;
- With parameters – COMMAND; PARAM1=VALUE1, PARAM2=VALUE2.

his approach makes it easy to parse messages and avoids the complex processing typical of structured formats such as JSON. Initially, the message format was implemented as JSON strings, but testing revealed significant performance limitations. In particular, JSON serialization and deserialization required significant amounts of RAM, significantly increased message processing time, and generally overloaded the ESP32's computing resources. For optimization purposes, it was decided to switch to a lighter format that combines ideas from various simple and proven syntaxes: AT commands, widely used in microcontrollers and modems, as well as URL Query Parameters, where parameters are passed as key-value pairs. This format is flexible enough for most application scenarios and convenient for both manual interaction and machine processing.

To check the connection between devices, a PING-type service message mechanism has been implemented. The controller initiates a connection check by sending a command in the format PING; ID=123, where ID is a unique request identifier. This message is transmitted via the LoRa module to the other end of the connection. An experimental diagram of the data exchange architecture between two LoRa modules is shown in Fig. 2.

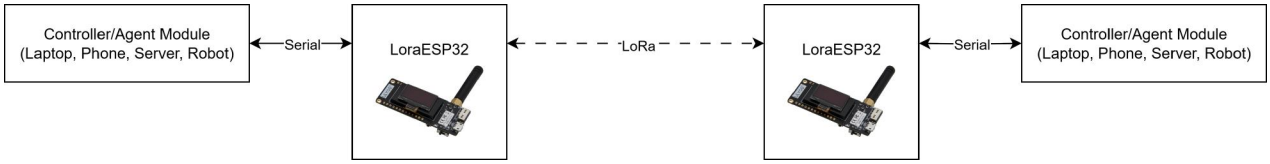

Fig. 2. Experimental diagram of the data exchange architecture between two LoRa modules

If the message is successfully received, the receiving module immediately sends a confirmation in response in the form of a PING_ACK message, storing the transmitted identifier. The message has the following format: PING_ACK; ID=123, DELAY=594, RSSI=-22.00, SNR=12.50, TOA=25, BPS=400.

The received message contains service reception parameters that characterize the quality of the communication channel and the message delivery time. The ID (Identifier) parameter is used as a request identifier, allowing the received response to be correlated with a specific sent request. The DELAY parameter reflects the delay in milliseconds from the moment the request was sent to the moment the response was received. RSSI (Received Signal Strength Indicator) characterizes the power level of the received signal. SNR (Signal to Noise Ratio) describes the signal-to-noise ratio in the communication channel. TOA (Time Over the Air) determines the packet transmission time in milliseconds. BPS (Bytes per Second) indicates the data transfer rate in bytes per second. If no response is received within 5 seconds after sending the PING request, the controller generates a service message about the lack of confirmation in the form: PING_NO_ACK; ID=123

This approach allows you to detect connection breaks, delays, or packet loss and respond accordingly, for example, by resending or switching to standby mode. All messages have a unified format and can be easily processed by automated systems or read by a human for diagnostics.

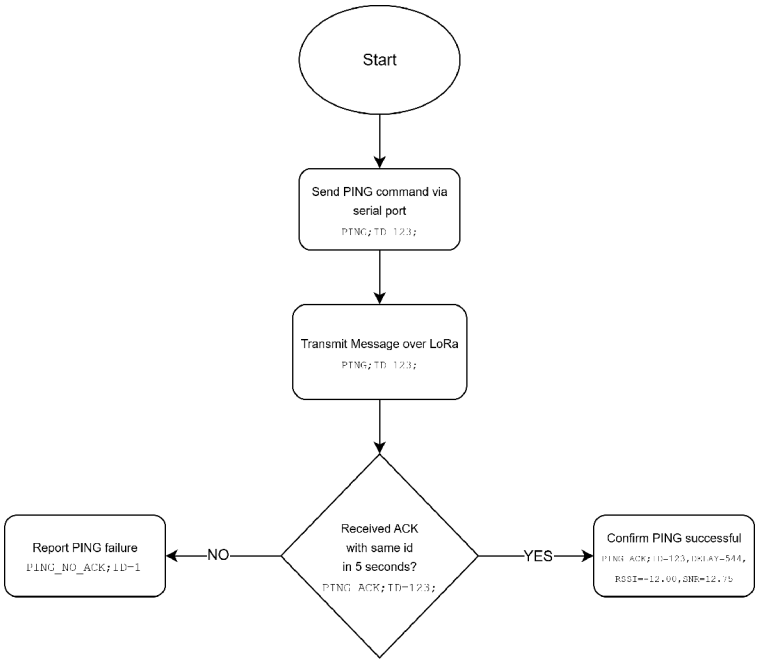

Fig. 3. Block diagram of the PING command algorithm

To ensure flexibility and optimal operation of the LoRa module in various operating conditions, the firmware provides the ability to configure a number of key parameters. To change the settings, send a command listing the parameters and their values to be replaced:

CONFIG; FQ=868, BW=125, SF=7, CR=6, TP=10, SW=171.

Table 1 shows the parameters that can be modified.

Table 1

Parameters for configuration

Parameter	Designation	Meaning
Frequency	FQ	860–870 MHz
Channel width	BW	7.8–500 kHz
Spreading factor	SF	6–12
Code rate	CR	5–8
Transmission power	TX	2–20
Synchronization word	SW	0–255
Header transmission	IH	0; 1
Packet size	HS	
Preamble length	PL	6–65535

For modules to work correctly, they must have identical configuration parameters, otherwise the module will not be able to receive messages from other devices. To simplify configuration and synchronization of parameters, the CONFIG_SYNC command has been developed, which sends a request with the parameters that need to be changed. If the module successfully receives this request, it updates its settings to the new values and sends a confirmation control packet. If a control packet with new parameters is received, the changes are considered confirmed. If no confirmation is received, the module restores the previous settings to maintain stability. Fig. 4 shows a block diagram of the algorithm for this command.

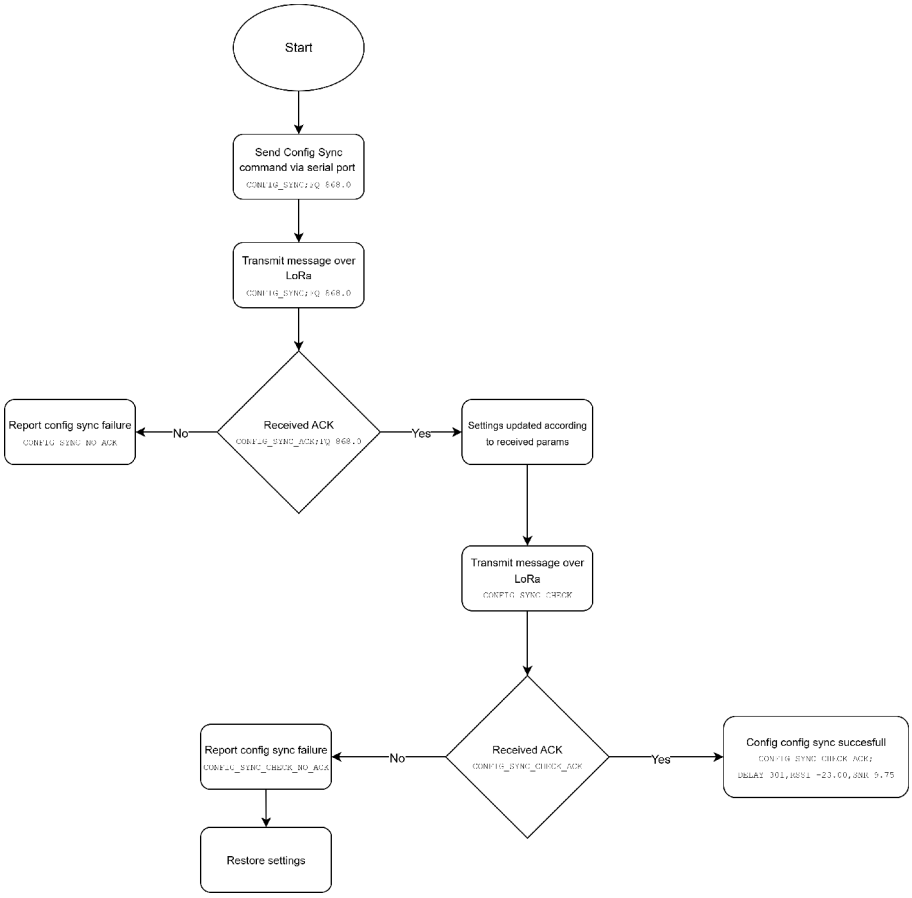

Fig. 4. Block diagram of the automatic synchronization algorithm between modules

There are certain limitations when working with Spreading Factor 6 (SF6), which provides the highest transmission speed among LoRa modes. In particular, SF6 can only be used if no packet header is sent during transmission, as this mode only supports implicit header mode.

Fig. 5. Current parameters displayed on the screen

This means that all message parameters, including its length, must be agreed upon in advance between the modules. This results in a loss of flexibility in transmitting variable amounts of data, so the packet size is set with a certain margin. If the actual message is smaller than the allowable packet size, the remaining bytes are filled with spaces to achieve the agreed length. This approach ensures compatibility with SF6 requirements and maintains minimal transmission delay, which is critical for time-sensitive applications.

For convenience, the module displays the current parameters on the display (Fig. 5).

The developed software implements a mechanism for universal transmission of arbitrary data between devices. Block diagram of the segmented guaranteed data transfer algorithm is depicted in Fig. 6.

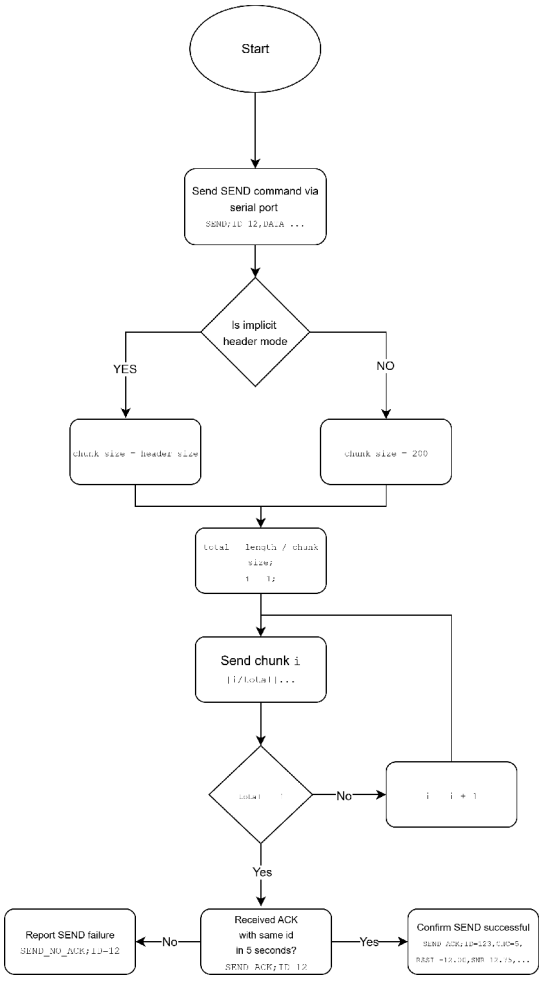

Fig. 6. Block diagram of the segmented guaranteed data transfer algorithm

Transmission is initiated by generating and sending a command to the LoRa module in the following format:

SEND; ID=12, DATA= “Test Data Hello”

After receiving this message, the LoRa module on the receiver side generates a confirmation of successful receipt, which contains the technical parameters of signal reception:

SEND_ACK; ID=12, DELAY=544, RSSI=-12.00, SNR=12.75, TOA=25, BPS=400

In turn, the receiver transmits the received message via the serial port in the following format:

DATA; ID=12, DATA= “Test Data Hello”

However, LoRa technology has certain limitations on the size of transmitted messages. In particular, the maximum size of a single packet is limited by hardware and protocol parameters and is typically up to 255 bytes (depending on the region, modulation, and configuration parameters such as Spreading Factor and Bandwidth). To enable the transmission of messages that exceed the permissible limit, a segmentation mechanism has been implemented. All large messages are automatically divided into segments of a fixed size of 200 bytes, or according to the length of the header if the no-header mode is used. Each chunk is accompanied by a service header that includes a sequence number and the total number of segments in the format:

[1/5] SEND; ID=12, DATA=....

This approach allows the receiving party to correctly track which part of the message has been received and how many are expected in total. After receiving all the chunks, the receiver combines them into the original message in the correct sequence, ensuring the integrity of the transmitted information.

After completing the transmission of all segments, the transmitting party waits for confirmation from the receiver in a standard format. The response contains service parameters, as well as a special CHC (Chunks Count) parameter, which indicates the number of segments into which the original message was divided.

3. Experimental study of communication quality in LoRa modules with developed embedded software

In order to evaluate the effectiveness of the developed embedded software, an experimental study of the main parameters of wireless communication quality between LoRa modules was conducted. During the tests, the basic settings of the module were changed, followed by the recording of key characteristics of the system’s operation, namely:

- transmission delay;
- received signal strength (RSSI);
- signal-to-noise ratio (SNR);
- time on air (TOA).

It should be emphasized that delay and transmission time in a wireless channel are not identical values: delay covers the entire cycle, namely from the moment the command is sent by the transmitter to the moment the confirmation is received by the receiver, while transmission time in the air describes only the physical duration of packet transmission.

Four series of experiments were conducted:

1. Transmission of a ping command at a distance of 15 cm (Experiment 1).
2. Transmission of 1 kilobyte of useful data at a distance of 15 cm (Experiment 2).
3. Transmission of a ping command at a distance of 100 meters (Experiment 3).
4. Transmission of 1 kilobyte of useful data at a distance of 100 meters (Experiment 4).

When transmitting 1 kilobyte of data, it is necessary to fragment it into several packets, since the SX1276 module has a hardware limitation of a 256-byte FIFO (First-In, First-Out) buffer. To ensure reliability, a fragmentation algorithm has been implemented that breaks packets larger than 200 bytes into smaller pieces. Thus, to transmit 1024 bytes, 7 packets were formed, each of which also includes a service header.

In each series of experiments, the following module parameters were changed: spreading factor, bandwidth, coding rate, transmission power, and preamble length.

Fig. 7 shows the results of an experimental study of the effect of the spreading factor (SF) on the key performance characteristics of the LoRa communication system.

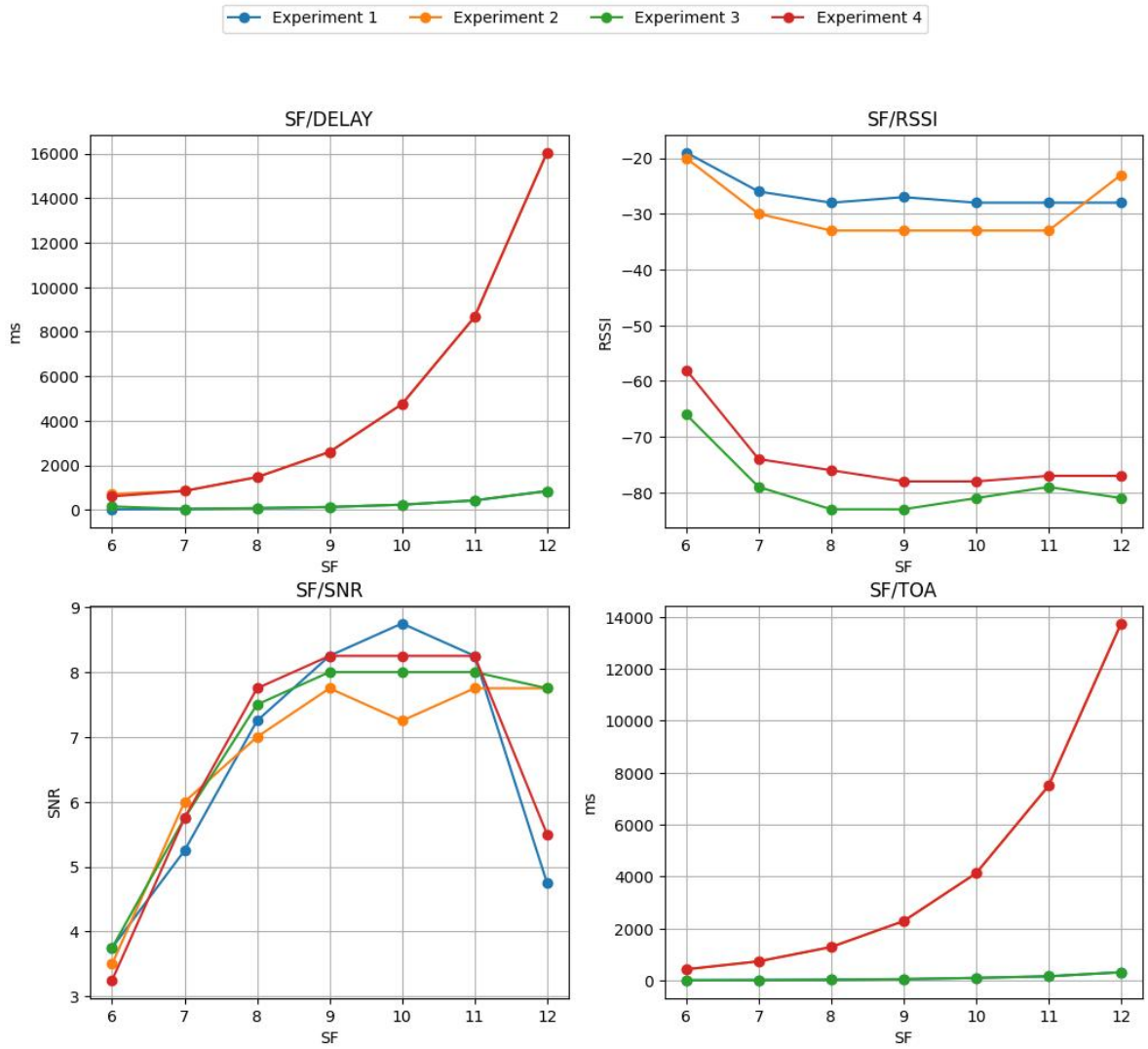

Fig. 7. Change in system characteristics at different values of the spreading factor

Analysis of the graphs shows a clear exponential dependence of data transmission latency on the SF value. This is a direct consequence of the Chirp Spread Spectrum modulation principle used in LoRa. As SF increases, each bit of information is encoded with a longer chirp, resulting in a proportional decrease in data transmission speed. Accordingly, the time required to transmit a complete data packet (Time-on-Air) increases exponentially. Thus, the trade-off between communication range (which improves with higher SF) and latency is a fundamental feature of LoRa modulation. Accordingly, the signal quality increased with the coefficient, but dropped significantly at a coefficient of 12. In the case of the first experiment, this can be explained by the receiver saturation effect, and in the case of transmitting 1 kilobyte of data, due to high sensitivity, it also registers noise.

Fig. 8 shows the results of an experimental study of the effect of bandwidth (BW) on the main operating characteristics of the LoRa communication system. The graphs show that as the bandwidth increased, the transmission delay decreased, resulting in higher data transfer rates. At the same time, with the expansion of the bandwidth, a significant deterioration in the signal-to-noise ratio (SNR) was observed,

since a wider channel covers a larger noise spectrum. The RSSI value remained virtually unchanged, since the distance between the transmitter and receiver did not change, and therefore the received signal power remained approximately constant.

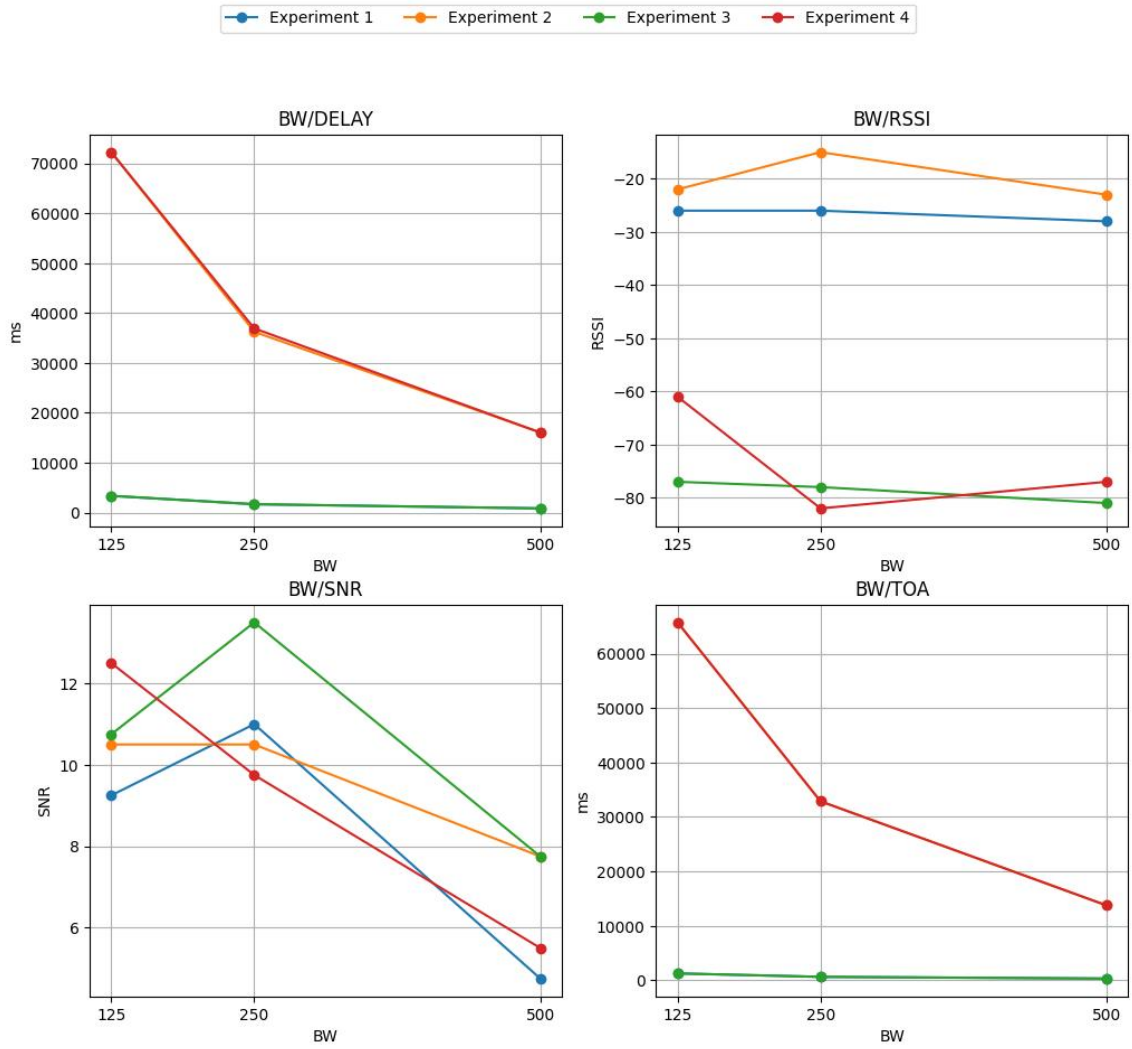

Fig. 8. Change in system characteristics at different bandwidth values

The next parameter under investigation was the coding rate (CR), as illustrated in Fig. 9. Experimental results showed that with an increase in CR, the transmission delay consistently grew. This effect arises because additional redundant symbols are appended to each transmitted frame, which allows the application of forward error correction (FEC). Such redundancy improves the system's capability to recover data in noisy environments, thus enhancing noise immunity and ensuring reliable packet delivery even under unfavorable propagation conditions.

At the same time, the introduction of redundant information reduces the proportion of useful payload data within the frame, which in turn lowers the effective throughput. Consequently, the time-on-air (ToA) for each packet increases, leading to longer end-to-end transmission times. Despite this drawback, the improved error resilience results in a noticeable gain in the signal-to-noise ratio (SNR) at the receiver, confirming that higher coding rates contribute to more robust communication links.

Overall, the findings highlight a fundamental trade-off: while higher CR values reduce spectral efficiency and increase delay, they also enhance link reliability and stability, which may be critical in long-range or interference-prone environments.

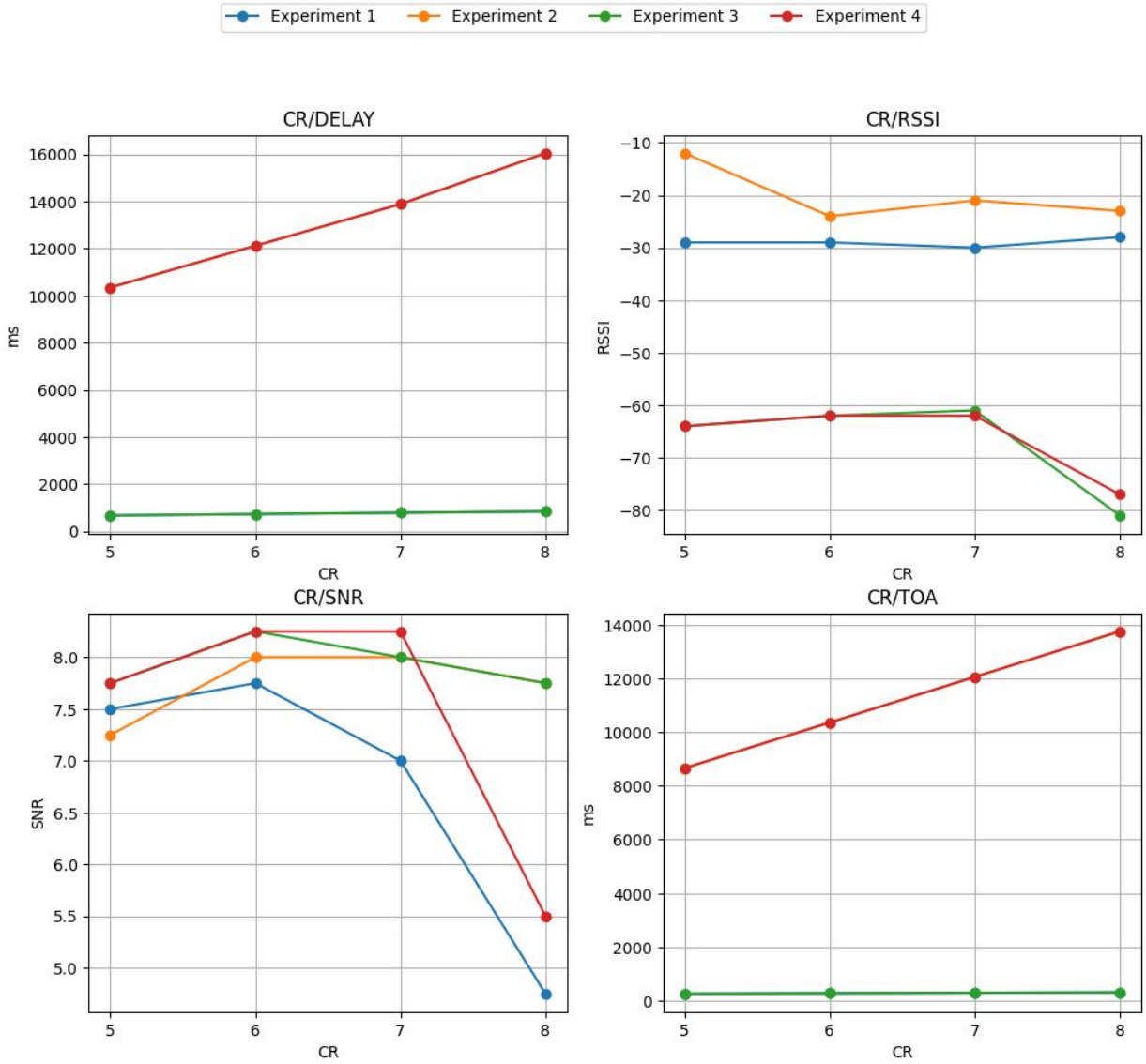

Fig. 9. Change in system characteristics at different error correction rates (Coding Rate)

Another important parameter analyzed in the experimental study was the transmission power (TP), with the corresponding results presented in Fig. 10. As shown in the graphs, variations in transmission power did not affect the transmission rate or the end-to-end delay, which remained practically constant across all experiments. This is expected, since these metrics are primarily determined by the modulation parameters (Spreading Factor, Bandwidth, Coding Rate) rather than the output power of the transceiver.

At the same time, an increase in TP had a direct impact on the received (RSSI). Higher output power resulted in a significant improvement of the received signal level, which indicates enhanced link budget and extended communication range. Moreover, the SNR also exhibited a positive trend, reflecting improved robustness of the communication channel under higher transmission power.

These results highlight that adjusting transmission power provides a straightforward mechanism for improving link reliability without altering the transmission speed. However, this comes at the expense of increased energy consumption, which may be a limiting factor for battery-powered IoT devices. Therefore, the selection of transmission power in LoRa-based systems should be optimized according to the application scenario, balancing energy efficiency with communication reliability.

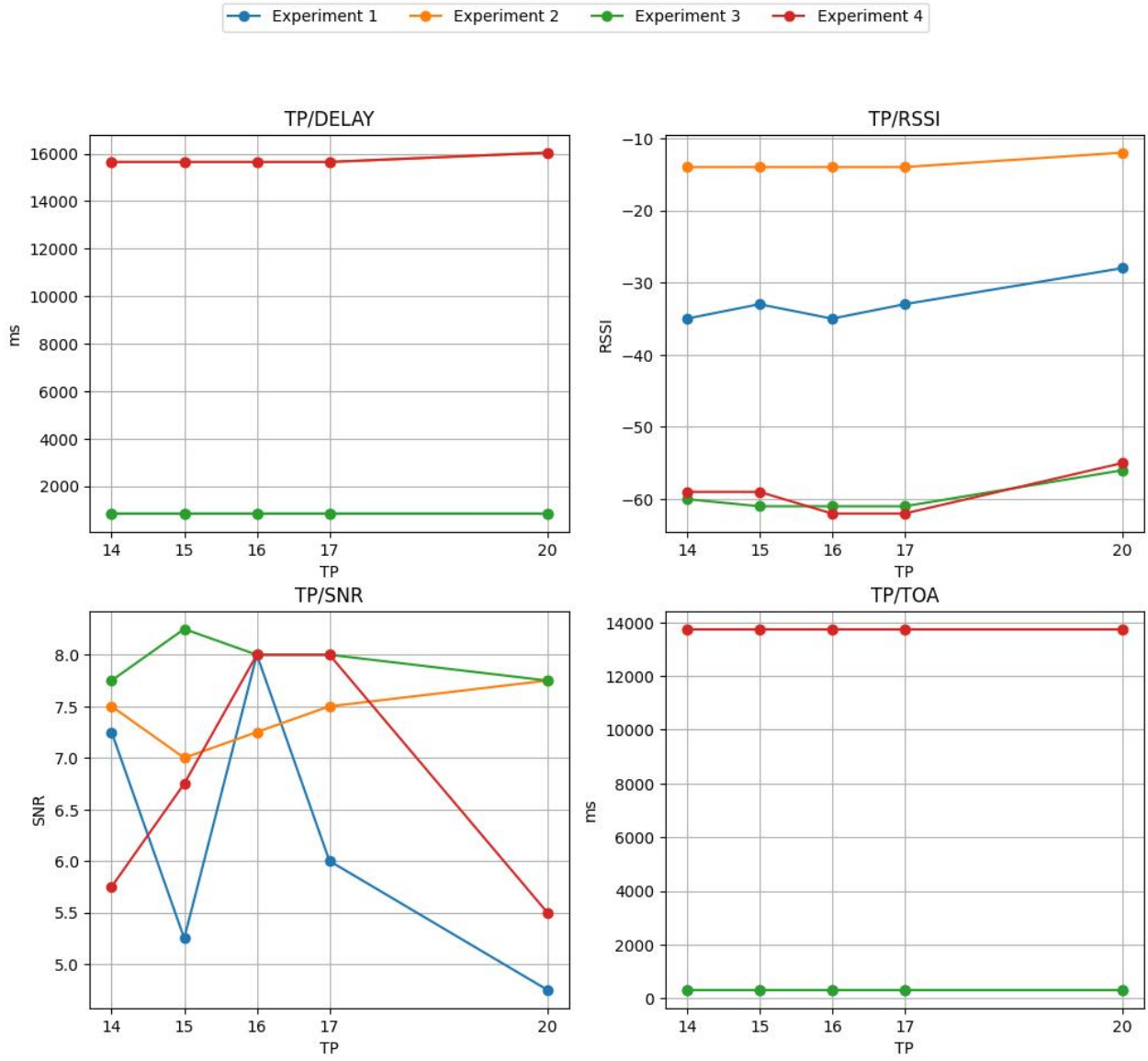

Fig. 10. Change in system characteristics at different transmission power value

The last parameter analyzed in the experimental study was the PL, with the corresponding results presented in Fig. 11. The obtained data demonstrate that an increase in PL led to a noticeable rise in ToA, which is explained by the proportional growth of the packet size due to the extension of the synchronization preamble.

At the same time, the RSSI remained nearly constant across different preamble values, confirming that PL does not affect the signal level at the physical layer. However, a significant improvement was observed in the SNR with longer preambles. This effect is associated with better synchronization at the receiver side, as a longer preamble allows more accurate detection and demodulation of the incoming signal, thereby reducing bit errors under noisy conditions.

Thus, increasing the preamble length enhances the robustness and reliability of the communication link, though at the cost of higher transmission time and, consequently, increased energy consumption. The choice of PL should therefore be adapted to the application scenario, balancing the trade-off between link stability and system efficiency.

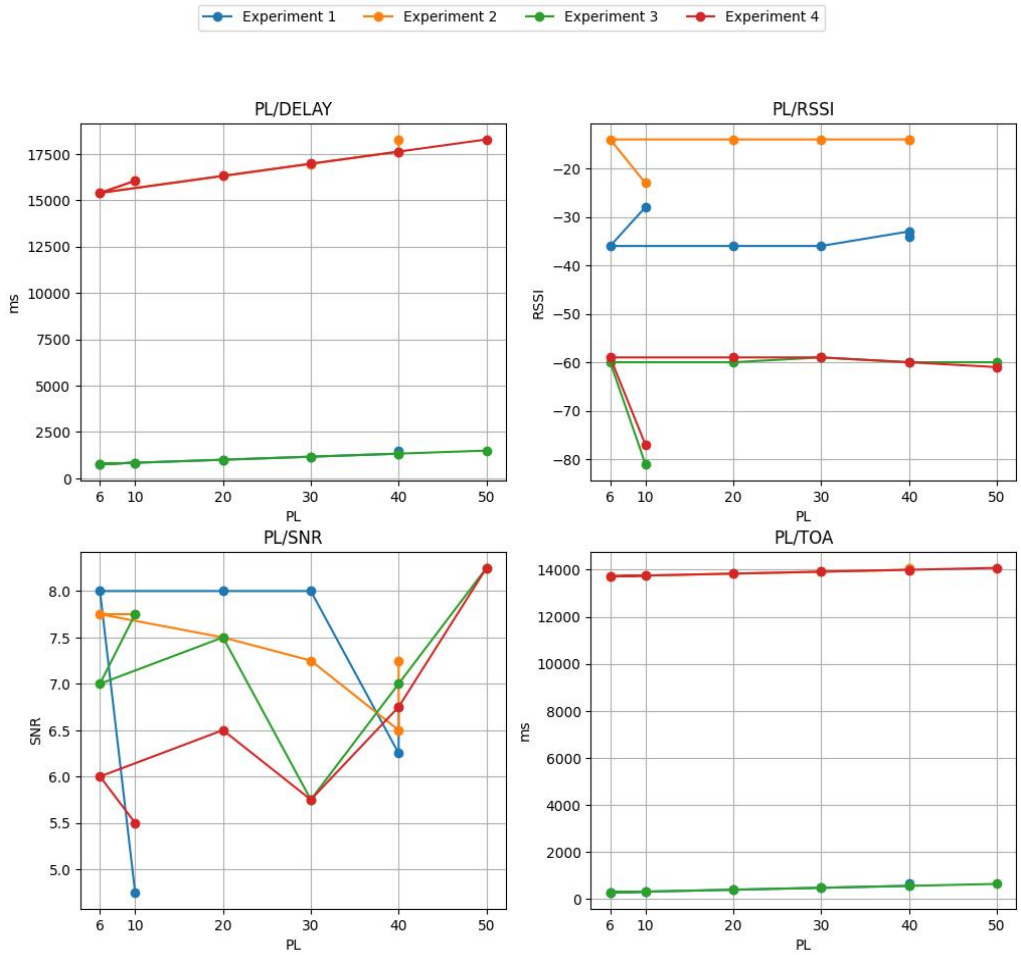

Fig. 11. Change in system characteristics at different preamble lengths

Experimental studies have confirmed that the developed software for LoRa modules not only ensures communication stability but also expands functionality compared to existing solutions. Analysis of the parameters showed patterns of their impact on transmission quality: an increase in the spreading factor increases the range but worsens the time characteristics; a wider bandwidth speeds up transmission but reduces the SNR; a higher coding rate enhances noise immunity at the cost of reduced speed; increasing the transmission power increases the signal level; a longer preamble improves SNR but increases transmission time. The scientific novelty of the work lies in the combination of a simplified command interaction protocol, automatic parameter synchronization, built-in diagnostics, and a packet fragmentation algorithm in a single firmware. This comprehensive approach not only improves data exchange efficiency, but also lays the foundation for further integration of artificial intelligence methods for adaptive real-time parameter optimization.

Conclusion

This paper presents the development of software for ESP32 microcontrollers with LoRa modules, which provides simple configuration and flexible control of wireless communication parameters. A mechanism for using simple text commands for configuration and control has been implemented, which simplifies integration into application systems and significantly reduces computational costs compared to structured formats. Additionally, communication quality assessment functions have been developed that allow analyzing channel reliability and adapting the device's operation to environmental conditions in a timely manner.The results confirm the effectiveness of the approach, which focuses on compactness, simplicity, and scalability of the system. The implemented solution can be used not only for basic data transfer scenarios, but also for more complex systems that require adaptability and scalability.

Further work will focus on developing an artificial intelligence module capable of optimizing communication parameters in real time. In particular, we plan to use reinforcement learning methods that will allow us to automatically select the optimal values for parameters such as frequency, bandwidth, and spread spectrum coefficient, according to current environmental conditions and interference levels. The computational part of the artificial intelligence algorithms will be performed on external computing modules, such as NVIDIA Jetson or Raspberry Pi, which will ensure a balance between high analysis performance and minimal load on the microcontroller firmware. This approach opens up prospects for the creation of intelligent wireless systems capable of independently adapting and optimizing communication in dynamic conditions.

References

[1] M. A. M. Almuhaya, W. A. Jabbar, N. Sulaiman, and S. Abdulmalek, "A survey on LoRaWAN technology: Recent trends, opportunities, simulation tools and future directions", Electronics, Vol. 11, No. 1, p. 164, Jan. 2022. DOI:10.3390/electronics11010164.

[2] J. M. Solé, R. P. Centelles, F. Freitag and R. Meseguer, "Implementation of a LoRa Mesh Library", in IEEE Access, Vol. 10, pp. 113158–113171, 2022. DOI: 10.1109/ACCESS.2022.3217215

[3] M. Jouhari, N. Saeed, M.-S. Alouini, and E. M. Amhoud, "A survey on scalable LoRaWAN for massive IoT: Recent advances, potentials, and challenges", IEEE Commun. Surv. Tutor., Vol. 25, No. 3, pp. 1841–1876, 2023. DOI: 10.1109/comst.2023.3274934.

РОЗРОБЛЕННЯ ВБУДОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДУЛІВ LORA НА БАЗІ ESP32 З АДАПТИВНОЮ КОНФІГУРАЦІЄЮ ТА МОНІТОРИНГОМ ЯКОСТІ ЗВ'ЯЗКУ

Юрій Шкоропад, Галина Бешлей

Національний університет "Львівська політехніка", 12, вул. С. Бандери, Львів, 79013, Україна

У статті описано новий підхід до розроблення вбудованого програмного забезпечення для LoRa-модулів на базі мікроконтролера ESP32. Основна ідея роботи полягає у створенні універсальної прошивки із мінімалістичною архітектурою та розширеними можливостями конфігурації, що забезпечує надійний обмін даними у режимі peer-to-peer. Розроблена система використовує спрощений текстовий формат команд (COMMAND;PARAM=VALUE) замість JSON, що знижує обчислювальні витрати та пришвидшує опрацювання. Завдяки цьому спрощується інтеграція у прикладні рішення та підвищується ефективність використання апаратних ресурсів. У прошивку інтегровано механізм підтвердження доставки (ACK) із повторним передаванням у разі втрати пакета, що підвищує надійність каналу зв'язку. Додатково реалізовано команду CONFIG_SYNC для автоматичної синхронізації параметрів між вузлами, що гарантує стабільність у динамічних умовах. Запропонований підхід передбачає також функцію PING/PING_ACK, яка, окрім перевірки доступності з'єднання, забезпечує отримання діагностичних характеристик, разом із RSSI, SNR, TOA, DELAY та швидкістю передавання даних. Передбачено можливість передавання великих повідомлень за допомогою алгоритму сегментації та об'єднання пакетів, що долає апаратні обмеження LoRa-чипа SX1276. У ході дослідження здійснено експериментальну перевірку роботи прошивки із варіацією ключових параметрів: spreading factor, bandwidth, coding rate, transmission power та preamble length. Отримані результати підтвердили закономірності впливу цих параметрів на затримку, швидкість, RSSI та відношення сигнал / шум, що дало змогу сформувати практичні рекомендації для оптимізації роботи системи. Запропоноване рішення поєднує простоту у використанні, гнучкість конфігурацій та інструменти оцінювання якості зв'язку, забезпечуючи баланс між продуктивністю та масштабованістю. Подальший розвиток передбачає інтеграцію модулів штучного інтелекту, зокрема reinforcement learning, для автоматичного підбирання оптимальних параметрів у реальному часі, що відкриває перспективи створення інтелектуальних самоналаштовуваних безпровідних систем.

Keywords: LoRa, ESP32, вбудоване програмне забезпечення, командний протокол, автоматична синхронізація, оцінка якості зв'язку.