Stability Verification of Self-Organized Wireless Networks with Block Encryption

Volodymyr Sokolov[a], Pavlo Skladannyi[a], and Hennadii Hulak[a]

[a] *Borys Grinchenko Kyiv University, 18/2 Bulvarno-Kudriavska str., Kyiv, 04053, Ukraine*

Abstract

Sensor networks allow the transmission of data for medical and research purposes. They are essential for building IoT wireless networks. The widespread use of such networks requires simplification of their construction and administration. The easiest way to develop and scale such a network is to use self-organizing networks. Since IoT networks operate in the available ISM frequency range, it constantly requires monitoring the load on the air. The paper shows the process of selecting free wireless channels using spectrum analysis. After the channel is selected, the throughput of the self-organizing system (botnets) is analyzed. The architecture and analysis of research results for botnets based on available hardware are shown. In addition, the issue of secure data transmission using fast and simple XTEA block encryption is considered.

Keywords

Spectrum analysis, bandwidth analysis, self-organizing network, wireless, botnets, IoT, block encryption, XTEA.

1. Introduction

The development of network technologies is increasing, affecting the number of digital devices connected to the network, including IoT. With this rapid development, there is a need to develop effective data exchange protocols and devices that will provide this exchange because the standard protocols used in traditional networks can not fully meet the needs of a new type of network, which became known as sensory. These networks should self-organize, as setting up a network manually on each new device is quite problematic. In addition, they should automatically repair broken connections and find new routes when the network topology changes dynamically, as most of the devices that the IoT concept combines are pretty mobile, so it is not possible to secure them in the network [1]. In addition to ensuring the smooth operation of the network, you need to provide cryptographic protection of data transmission [2]. Such encryption requires additional energy costs, which is critical for sensor networks and IoT in general. Thus, this paper considers the properties of such networks and ways to optimize them to increase the level of fault tolerance, data exchange rate, their protection, and performance control [3].

Sect. 2 analyzes the available research and highlights aspects that need further analysis. Spectrum and bandwidth analysis are provided in Sect. 3 and 4. Sect. 5 and 6 are devoted to the construction of the self-organizing network and their experimental study. Also, Sect. 7 uses block cipher for the self-organizing network. The final section outlines the directions for future research in this area.

2. Related Works

An ad hoc wireless peer-to-peer network is a decentralized wireless network [4, 5]. The network can be called “special” for a particular case (according to its name) because it does not rely on existing infrastructure, such as routers in wired networks or access points in the routed network [6].

CMIS-2022 : Fifth International Workshop on Computer Modeling and Intelligent Systems, May 12, 2022, Zaporizhzhia, Ukraine
EMAIL: v.sokolov@kubg.edu.ua (V. Sokolov); p.skladannyi@kubg.edu.ua (P. Skladannyi); h.hulak@kubg.edu.ua (H. Hulak)
ORCID: 0000-0002-9349-7946 (V. Sokolov); 0000-0002-7775-6039 (P. Skladannyi); 0000-0001-9131-9233 (H. Hulak)


Instead, each node participates in routing by sending data to other nodes, thus determining which nodes the data will be transmitted dynamically based on the network connection and the routing algorithm used in a particular case.

The decentralized nature of wireless networks makes them suitable for various applications where central nodes cannot be laid and improve networks' scalability compared to wireless managed networks [7]. Minimal configuration and rapid deployment make peer-to-peer networks suitable for emergencies such as natural disasters or military conflicts. The presence of dynamic and adaptive routing protocols allows peer-to-peer networks to be quickly configured and modified [8].

Concerning the security of data transmitted by sensor networks, it should be noted that most ad hoc networks do not exercise any control over access to the network. These networks are vulnerable to consumer attacks, where a malicious node enters packets into the network. Source [9] prevent such attacks. It is necessary to use authentication mechanisms that ensure that only authorized nodes can enter traffic into the network [10]. However, it should be understood that even with authentication, these networks are vulnerable to de-authorization packets or retard attacks, resulting in the intermediate node rejecting the package or delaying it rather than immediately sending it to the next node.

Thus, based on the dependence of the principles of the network on its purpose, it is necessary to identify the requirements for the designed network before starting work on it to avoid situations that would require further changes to the overall architecture of the program.

3. Spectrum Analysis

Spectrum analyzers do not increase the availability of information in the network but contribute to its improvement by detecting weaknesses and filling the spectrum with other networks (collision prevention/interference). Unlike standard tools available in network cards, the spectrum analyzer collects information about noise levels, for example, from magnetrons (in microwave ovens), can detect stationary interference and the operation of other equipment (Bluetooth, ZigBee, children's toys, video transceivers, medical sensors, etc.) [11]. Also, the spectrum analyzer "sees" not only the service packets, which assess the signal level in the network cards but also all other packets. Therefore, the topic of the paper is essential and highly relevant. Previous versions of industrial spectrum analyzers TI CC25xx [12] and Cypress CYRF693x were used by the authors as sensors in the study of antenna technology [13, 14] and the construction of sensor networks [15].

Fig. 1*a* and 1*b* show the results of reception at the location of the spectrum analyzer in the near zone of the transmitter for data transmission over the Bluetooth wireless channel, and in Fig. 1*c* and 1*d*—Wi-Fi band 2.4 GHz. From the figures, it is easy to see that the solid assembly showed much better results since it is better assembled using the screen and external antenna [8].

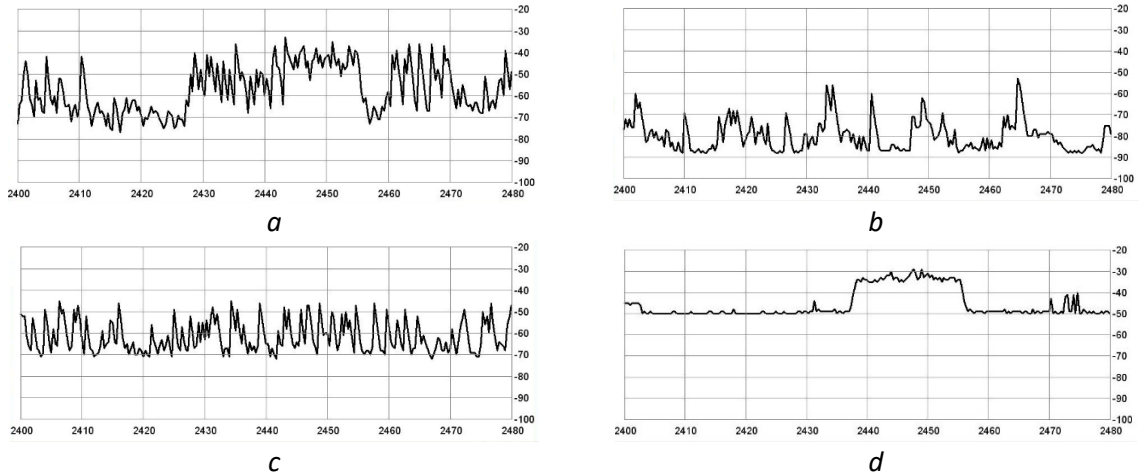


Figure 1: Examples of spectra for different devices: modular for Bluetooth (*a*); single-board for Bluetooth (*b*); modular for Wi-Fi (*c*); single-board for Wi-Fi (*d*)

After assembly, testing, and debugging, the devices are ready for technical and scientific tasks. The general view of all three devices is shown in Fig. 2.

Figure 2: General view of different devices

The paper presents the results of the design and manufacture of spectrum analyzers on finished components (transceiver chips for the IEEE 802.15.4/ZigBee standard) [16]. Designing, manufacturing printed circuit boards, assembling devices, and programming microcontrollers are described in detail. Testing and improvement of existing devices. When wiring, the board revealed the dependence of the quality of the device on the quality of its assembly, the presence of an electromagnetic screen, and the type of antenna [8].

4. Bandwidth Analysis

For the purity of the experiment, both machines are configured identically with the same operating system and the same settings. Therefore, everything written as Raspberry Pi (RPi)—applies to RPi Zero and RPi 3 (Fig. 3). The operating system (OS) used an OS without a graphical interface—Raspbian OS Lite to have a minimum load on the system. The peculiarity of this OS is the absence of "extra" programs that slow down the computer. In this version of the OS, there is a package for working with GPIO. RPi act as a wireless Wi-Fi access point, configured by example from [9]. The access points have a *vsftpd* FTP server, which is easy to set up, fast, and secure. As a result, the average of all tests will be taken into account. Every five seconds, the readings are taken: the load and temperature of the server processor, the current and voltage consumed by the server, and the download speed of the file on the FTP client.

The second stage was the automation of the process of writing history files. This will minimize human error during the recording of results and thus increase the accuracy of measurements and recording speed. A digital INA219 sensor was used to measure voltage and current. In addition to the built-in processor temperature sensor, an external DS18B20 sensor is additionally used [17]. Both sensors are connected to the GPIO interface, which allows you to receive information in text format, which is used to automate measurements.

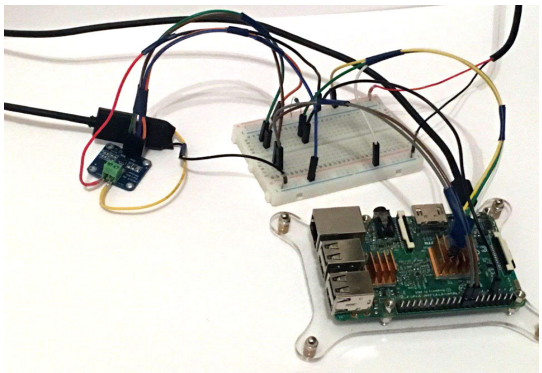


Figure 3: Test bench for Raspberry Pi 3

The *sensors.py* script is running on the Server-PC, which receives data from the DS18B20 and INA219 sensors. The ina219 library is used to work with INA219. When the DS18B20 temperature sensor is connected, a file is automatically created in which the temperature is displayed in text format. Therefore, to get the ambient temperature in *sensors.py*, it is enough to open and expand this file using the built-in OS library in Python. The general scheme of operation of the *sensors.py* script is shown in Fig. 4.

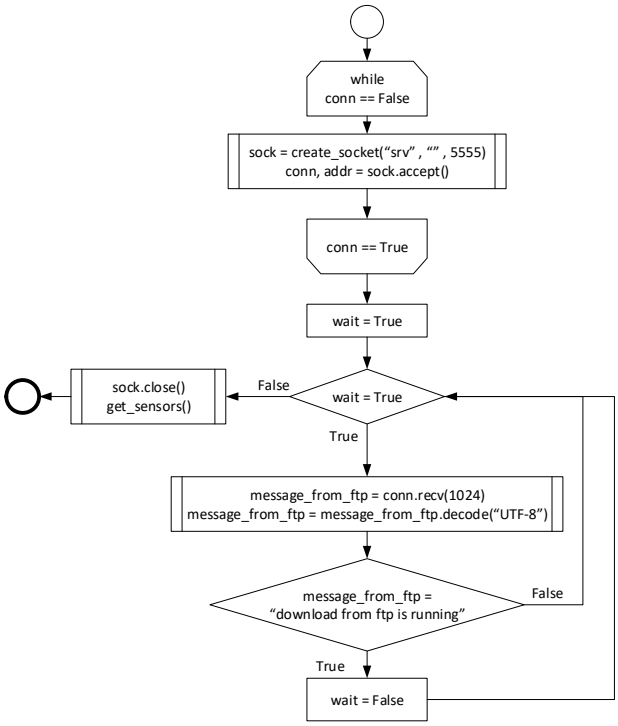


Figure 4: Sensor read algorithm

The processor on RPi 3 was less loaded than on RPi Zero, so that you can run more resources on RPi 3 than on RPi Zero. Interestingly, at a longer distance from the client from the access point, the CPU 3 CPU load was lower. RPi Zero at a longer length periodically decreased sharply but tended to the maximum value (Fig. 5).

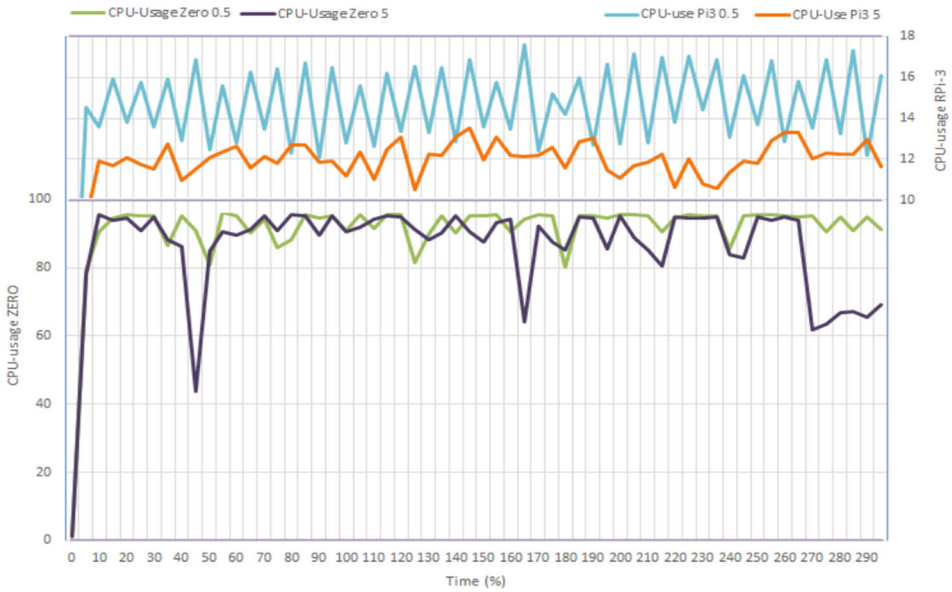


Figure 5: Comparison of RPi Zero and RPi 3 CPU usage

The injection rate decreased most markedly with a sharper decrease in current consumption. Current consumption increased sharply when necessary to reduce the CPU load to reduce the CPU temperature (Fig. 6).

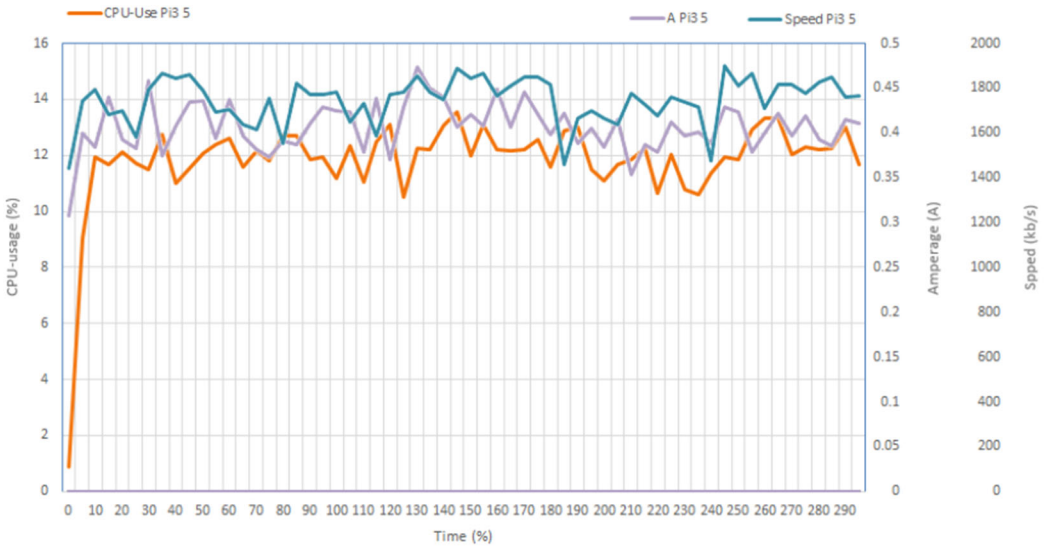


Figure 6: Comparison of CPU usage, power consumption, and file download speed of RPi 3

Using Pearson's correlation, you can calculate the relationship between the factors:

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}}, \quad (1)$$

where x is the value of one factor; y is the value of another factor to which the ratio refers.

Table 1 shows the results of calculating the Pearson correlation coefficient for different distances by expression (1). The correlation coefficient varies from +1 to –1. In the presence of a positive correlation, an increase in one indicator contributes to the rise. With a negative correlation, an increase in one indicator entails a decrease in another. The larger the modulus of the correlation coefficient, the more noticeable the change in one hand affects the difference in the other. At a factor of 0, the relationship between them is absent [18].

Table 1

The results of the calculation of the Pearson correlation coefficient for RPi Zero / RPi 3

	Speed	CPU load	CPU temp.	Temp.	Voltage	Amperage
Speed	1.00/1.00					
CPU load	0.55/0.38	1.00/1.00				
CPU temp	0.30/0.24	0.26/0.46	1.00/1.00			
Temp.	0.11/0.02	–0.02/0.05	0.87/0.19	1.00/1.00		
Voltage	–0.16/–0.04	0.06/–0.17	–0.18/–0.08	–0.18/–0.04	1.00/1.00	
Amperage	0.05/0.32	0.01/0.43	0.22/0.17	0.21/–0.07	–0.12/–0.10	1.00/1.00

This technical complex in RPi and the DS18B20 temperature sensor and the INA219 current, voltage, and power sensor can be integrated into such systems as monitoring systems in data centers, intelligent home systems, etc. This technical complex is already used for temperature monitoring servers in the data center. For visualization, monitoring, and analysis of the received data, the Grafana platform is shown in Fig. 7.

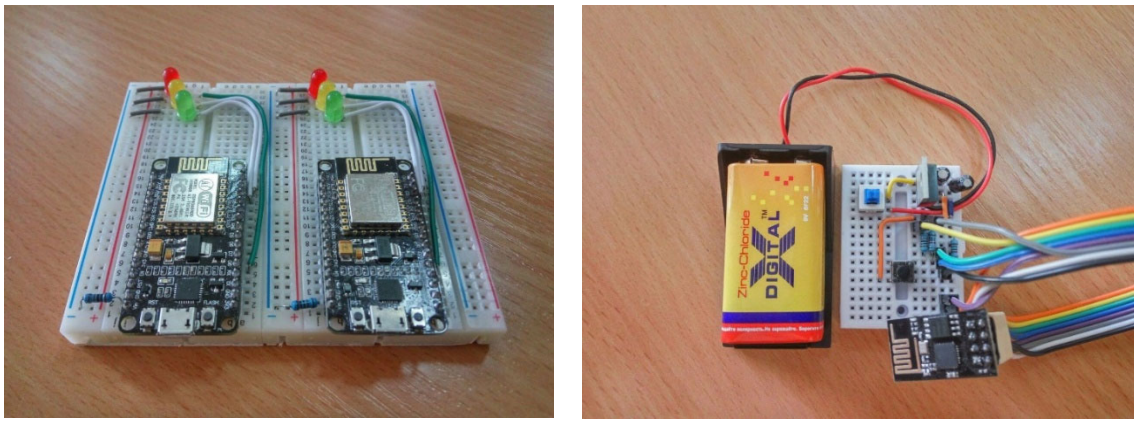
Figure 7: Grafana dashboard with temperature and voltage indicators

5. Self-Organizing Network

To test the capabilities of the ad hoc sensor network based on the IEEE 802.11 standard, we will try to build our network, which will be stolen from the modules on the ESP8266 microcontroller with a self-developed firmware [19].

Since the most widely used such networks are used in IoT solutions, we can conclude that the main function of modules in the network is to obtain a small amount of data from devices to which they are connected via serial interface, transferring this data over Wi-Fi between modules and return the result to the parent device via the serial interface. Accordingly, the system “module device” must both receive and process the received commands. Therefore, NodeMCU and ESP-12E were used to simulate these conditions for convenience.

As a result, we have approximately such a test bench necessary to begin developing and implementing the first tests, shown in Fig. 8.



a *b*

Figure 8: Test benches with NodeMCU 1.0 (*a*) and ESP-01 (*b*) modules

The only disadvantage of this solution is the voltage given by this battery: up to 9 V. The ESP8266 can operate at voltages from 3 to 3.6 V. However, the problem can be solved with a suitable voltage regulator. In this case, the stabilizer LD1117 was used. You can also use similar stabilizers, such as AMS1117 or LM1117. As a result, the test device shown in the figure was assembled, which is entirely autonomous, so when implemented on an entire board, it can be used in conditions with difficult access, thus using the device to obtain data from instruments located at remote points, while without having to go to them.

The next step was to develop firmware for the assembled system. Already at the stage of developing the program architecture, some difficulties appeared. Since full-fledged ad hoc networks

typically run on multifunctional equipment designed directly to build networks (Wi-Fi access points and routers), which has significant computing capabilities and the ability to multithreaded data processing, the problem of how to implement the same functions on a microcontroller. Significantly inferior to its technical characteristics. Several open-source solutions have been considered, including the aforementioned ESP8266WiFiMesh library. As a result, it was found that all such solutions are implemented on the alternate change of modes of operation of the module, as shown in Fig. 9.

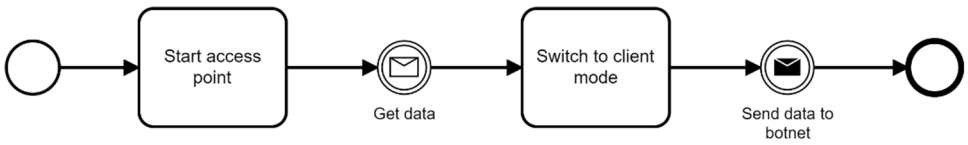


Figure 9: Mesh network operation algorithm

The exchange algorithm was implemented using the ESP8266Mesh library. Therefore the mode of operation of the module is controlled by the algorithm shown above, but with a small change, which was described above. At each stage of the cycle, the firmware listens for incoming connections. If one of the modules is connected to it processes the incoming message and sends back a response. The next step is to check the messages in the queue and, if available, switch to the client mode, send the first message from the line to related modules. Then the procedure is repeated.

Because the size of the queue can vary depending on the node load and the speed of sending messages, data is stored in dynamic memory. As a result, there were some difficulties in writing this part of the program. Due to a small error in the work with the pointers, the program suffered a fatal error, and the microcontroller rebooted in an emergency. The problem was related to a pointer that connected messages in a queue, causing the program to reference an already occupied area of memory. After correcting the error, the first working prototype of the program was obtained, thanks to which it was possible to implement data transfer between two test NodeMCU 1.0 modules [20].

6. Experimental Assessment of Self-Organizing Network

The experimental setup consists of a working network model, which includes three nodes (two NodeMCU boards from the test bench and a test device with the ESP-01 module).

The first test was the direct transfer of data between nodes and the node's received messages. For this purpose, two test messages were sent from node Node14754480, which were addressed to node Node52126. In the test message, a command was sent to turn on the LEDs on the Node52126 module. The module received the message in two ways: directly from the sender's module and through the "intermediary" Node1592748, which also received a message from Node14754480 and forwarded it. The second message was rejected as a duplicate. The system worked correctly. The following are the messages to the console from modules Node14754480 and Node52126. The message, in this case, was received directly from the sending module and processed. The red LED was lit. The message has reached the recipient. No further mailing is required. The system has switched to listening mode. The test was passed successfully.

The message transmission time was approximately measured. Depending on the conditions, the data transmission in this network can take from one to five seconds, depending on the message path and the response rate of the modules.

A test device based on an Arduino Nano board was also used for testing, to which the module must send a message for processing, after which the LEDs should light upon this device. This function works in the same way as when processing commands with LEDs on test modules with NodeMCU. The Arduino also generates a message for other modules every five seconds and transmits it to the ESP-01 module for further transmission. During testing, the problem with the incorrect reading of messages by the module was revealed. The reason for this was the special symbols provided by the transfer fee. After eliminating this problem, messages began to be sent correctly, and the board could communicate with other devices.

7. Implementation of XTEA Encryption Protocol

Based on the hardware and software features of Pololu Wixel (TI CC25xx microcontroller) devices [21, 22], the following problems were identified during the implementation of this project:

- Compilation of types.
- Buffer overflow.
- Work with pointers.
- Variables in XDATA memory.
- Looping with minimal delay.
- Compiler errors.

Casting types. Because the C programming language used in SDCC is hard-typed [23], there is a problem with typesetting. For example, the compiler forbids equating uint8 and char, although char also requires one byte. It is also impossible to equate specific arrays. Hence the need to compile data types between variables and intentionally specify a summary when passing a function parameter. Compilation of types, in most cases, does not give the desired result because the programming language does not involve the conversion of certain data types.

The user enters data into the client program; the program processes and sends this data to the sender via USB (Fig. 10). If necessary, the program sends the keys to the sender and receiver before sending the processed data. The sender encrypts the received data via XTEA, adds its unique MAC address, and sends it all over the air [24–26]. The receiver receives the sent data from the air, decrypts them, and transmits them to the client program via USB. The sender and receiver can transfer open data via USB.

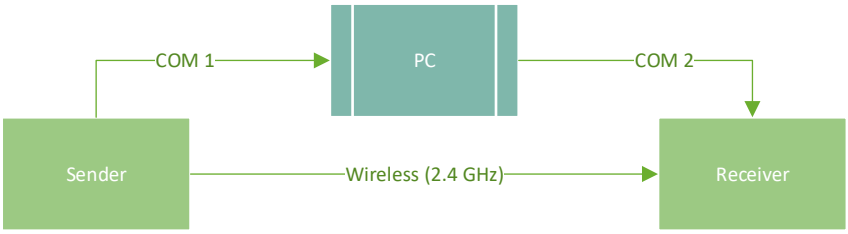


Figure 10: Schematic diagram of the project data

For the device to understand that a key will be sent now, which will need to be used instead of the standard one, we send it a group of eight identical bytes containing the ENQUIRY control symbol (number 05) from the ASCII table, which is intended for use in teletype communication and this situation can be used for its purposes (Fig. 11).

When dividing data on the computer side, the following are used: the number of iterations, the remainder of the last iteration, and the number of elements that are not enough to complete the last iteration.

First, we check whether the multiple lengths of the entered data are eight. From here, we determine how many iterations we need to perform. Then we get how many elements remain on the last (possibly incomplete) iteration. Then we calculate the number of missing elements and increase the array by this number. When you resize a .NET array, it automatically initiates all undefined elements with zeros.

You can receive data on a computer without distribution because its memory is much larger than the devices' memory [27]. In general, the integration was successful with some side issues that arose during the project implementation on the platform used, which were identified, researched, and resolved. The devices have proven to be very promising and durable, so that this project will be continued with new goals and objectives [28].

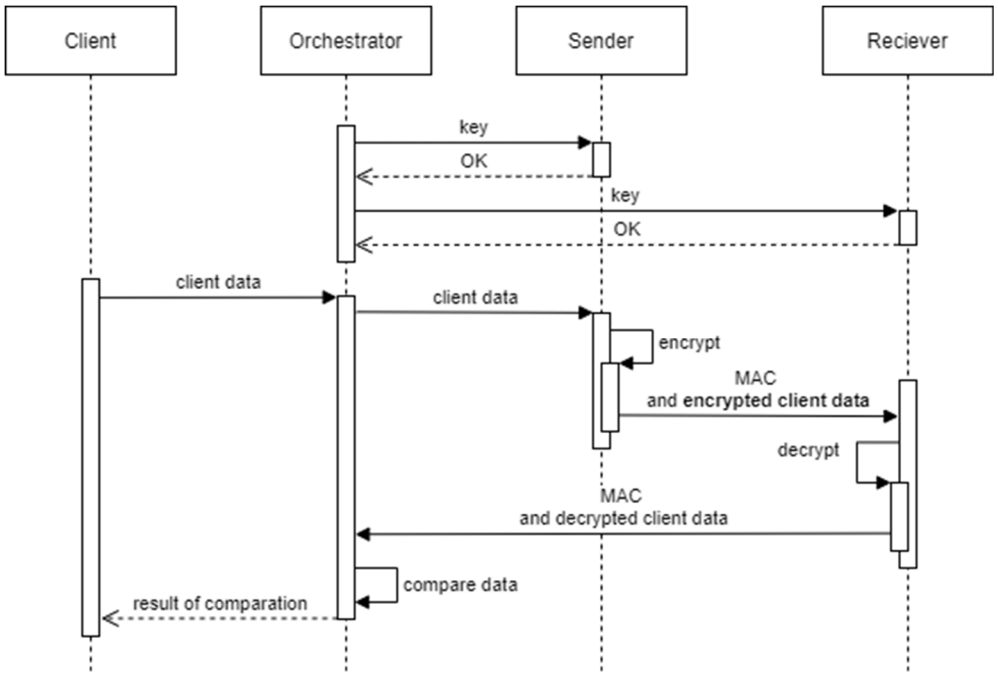


Figure 11: Key and data exchange scheme

8. Conclusions and Further Research

Based on the results, we can conclude that all the studied factors, namely the server CPU load, CPU temperature, current consumption, affect the download speed of files. When the processor is running at peak temperatures, there is a process of protection against overheating, called clock throttling. At this time, the processor's clock speed is reduced, and its performance and efficiency are reduced. This reduces the file download speed. According to the obtained data, it is seen that in RPi Zero W, the process of clock throttling occurred with a higher decrease in clock frequency and processor performance than in RPi 3.

For large amounts of wireless data transmission, RPi 3 is more suitable than RPi Zero W because the third version is more productive to support more simultaneous tasks. However, RPi Zero is more suitable for simple tasks because this version requires less power. For IoT devices, autonomy and power consumption is a significant indicator.

Based on the developed experimental layout and experimental results, we developed and created our ad hoc sensor network implementation based on the ESP8266 microcontroller. For this purpose, third-party open solutions were used in combination with our developments. Several tests were conducted, which allowed us to understand its weaknesses better, pay attention to them in the future, and think about ways to correct them. The obtained results can be effectively used for further work in this direction, which will improve the solution and further apply them in real platforms, built on both small controllers and full-fledged network devices, which is now quite relevant. Therefore, in the future, after some refinements, this system can be used in current conditions to form an intelligent home system to obtain information from specific sensors.

In this paper, there were problems in the implementation of encryption algorithms on low-power processors. Due to lack of device resources, the block XTEA encryption protocol was chosen, which, although not yet widely recognized, has shown promising results in our specific task. Several problems related to type matching, addressing, memory spaces, buffer overflow, and others were solved during the work. Resolved issues with the coordination of the receiver and transmitter. The main task of the work—building a secure communication channel by encrypting data in the channel—was solved and received firmware and application software for a full test of the devices. Also, this work has great application potential. The introduction of encryption in existing systems will have little effect on the implementation and will not affect the cost estimate of projects but will dramatically increase the security of data transmission in these networks.

Continuation of this work may verify the operability of other protocols on this and similar hardware for systems that can be embedded in data exchange projects in short-range wireless communication channels. Also of particular interest is the determination of the dependence of the information rate of data transmission on the length of the key. In the future, it is planned to check packages and implement an algorithm for retrieving lost packages; translate the client part to Python for cross-platform use; make one universal firmware for all devices (TX and RX); provide for the possibility of using many (more than two) devices on one channel; implement, test and investigate other encryption algorithms.

9. References

[1] IEEE Standard for Local and Metropolitan Area Networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). doi:10.1109/ieeestd.2011.6012487.

[2] S. Gnatyuk, Critical aviation information systems cybersecurity, in: Meeting Security Challenges Through Data Analytics and Decision Support, NATO Science for Peace and Security Series, D: Information and Communication Security. IOS Press Ebooks 47(3) (2016) 308–316.

[3] S. Gnatyuk, et al., New secure block cipher for critical applications: Design, implementation, speed and security analysis, Advances in Artificial Systems for Medicine and Education III (2020) 93–104. doi:10.1007/978-3-030-39162-1_9.

[4] F. Stajano, R. Anderson, The resurrecting duckling: Security issues for ad-hoc wireless networks, Lecture Notes in Computer Science (2000) 172–182. doi:10.1007/10720107_24.

[5] S. Zhu, et al., LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks, in: 23rd International Conference on Distributed Computing Systems Workshops, 2003. doi:10.1109/icdcsw.2003.1203642.

[6] M. M. Zanjireh, H. Larijani, A survey on centralised and distributed clustering routing algorithms for WSNs, in: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring) (2015). doi:10.1109/vtcspring.2015.7145650.

[7] V. Y. Sokolov, D. M. Kurbanmuradov, Method of counteraction in social engineering on information activity objectives, Cybersecurity: Education, Science, Technique 1 (2018) 6–16. doi:10.28925/2663-4023.2018.1.616.

[8] V. Y. Sokolov, Comparison of possible approaches for the development of low-budget spectrum analyzers for sensory networks in the range of 2.4–2.5 GHz, Cybersecurity: Education, Science, Technique 2 (2018) 31–46. doi:10.28925/2663-4023.2018.2.3146.

[9] Oestoidea, Access Point on Raspberry Pi 3 with Parameter Display (2017). URL: https://github.com/Oestoidea/Adafruit_Python_SSD1306.

[10] Python Software Foundation, pi-ina219 1.2.0. Project Description (2018). URL: https://pypi.org/project/pi-ina219/.

[11] I. Bogachuk, V. Sokolov, V. Buriachok, Monitoring subsystem for wireless systems based on miniature spectrum analyzers, in: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (2018). doi:10.1109/infocommst.2018.8632151.

[12] Texas Instruments, CC2510Fx, CC2511Fx Silicon Errata (2015). URL: http://www.ti.com/lit/er/swrz014d/swrz014d.pdf.

[13] V. Astapenya, et al., Last mile technique for wireless delivery system using an accelerating lens, in: 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (2020). doi:10.1109/picst51311.2020.9467886.

[14] V. Astapenya, V. Sokolov, D. Ageyev, Experimental Evaluation of an Accelerating Lens on Spatial Field Structure and Frequency Spectrum, in: 2020 IEEE Ukrainian Microwave Week (2020). doi:10.1109/ukrmw49653.2020.9252755.

[15] V. Sokolov, A. Carlsson, I. Kuzminykh, Scheme for dynamic channel allocation with interference reduction in wireless sensor network, in: 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (2017). doi:10.1109/ infocommst.2017.8246463.

[16] N. Sastry, D. Wagner, Security considerations for IEEE 802.15.4 networks, in: Proceedings of the 2004 ACM Workshop on Wireless Security—WiSe'04 (2004). doi:10.1145/1023646.1023654.
[17] Les Pounder, DS18B20 Temperature Sensor With Python (Raspberry Pi) (2017). URL: https://bigl.es/ds18b20-temperature-sensor-with-python-raspberry-pi/.
[18] V. Sokolov, B. Vovkotrub, Y. Zotkin, Comparative bandwidth analysis of lowpower wireless IoT-switches. Cybersecurity: Education, Science, Technique 5 (2019) 16–30. doi:10.28925/2663-4023.2019.5.1630.
[19] Aerlon, Devyte, ESP8266 WiFi Mesh (2018). URL: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFiMesh.
[20] M. Vladymyrenko, V. Sokolov, V. Astapenya, Research of stability in ad hoc self-organized wireless networks, Cybersecurity: Education, Science, Technique 3 (2019) 6–26. doi:10.28925/2663-4023.2019.3.626
[21] Pololu Corporation, Pololu Wixel User's Guide (2015). URL: https://www.pololu.com/docs/0J46/all.
[22] Pololu Corporation, Wixel SDK Documentation (2015). URL: http://pololu.github.io/wixel-sdk/.
[23] S. Dutta, SDCC Compiler User Guide, ver. 3.9.5, rev. 11239 (2019). URL: http://sdcc.sourceforge.net/doc/sdccman.pdf.
[24] Amandeep, Implications of bitsum attack on tiny encryption algorithm and XTEA, Journal of Computer Science 10(6) (2014) 1077–1083. doi:10.3844/jcssp.2014.1077.1083.
[25] Amandeep, G. Geetha, On the complexity of algorithms affecting the security of TEA and XTEA, Far East Journal of Electronics and Communications (2016) 169–176. doi:10.17654/ecsv3pi16169.
[26] O. Arsalan, A. I. Kistijantoro, Modification of key scheduling for security improvement in XTEA, in: 2015 International Conference on Information & Communication Technology and Systems (2015). doi:10.1109/icts.2015.7379904.
[27] I. Kuzminykh, et al., Investigation of the IoT device lifetime with secure data transmission, Internet of Things, Smart Spaces, and Next Generation Networks and Systems (2019) 16–27. doi:10.1007/978-3-030-30859-9_2.
[28] D. Kurbanmuradov, V. Sokolov, V. Astapenya, Implementation of XTEA encryption protocol based on IEEE 802.15.4 wireless systems, Cybersecurity: Education, Science, Technique 2(6) (2019) 32–45. doi:10.28925/2663-4023.2019.6.3245.