FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 36, No 2, June 2023, pp. 267 - 284
https://doi.org/10.2298/FUEE2302267M

Original scientific paper

IMPLEMENTATION AND TESTING OF WEBSOCKET PROTOCOL IN ESP32 BASED IOT SYSTEMS

Nikola Mitrović[1], Milan Đorđević[2], Sandra Veljković[1], Danijel Danković[1]

[1]University of Niš, Faculty of Electronic Engineering, Niš, Serbia
[2]College of Applied Technical Sciences in Niš, Serbia

Abstract. *This paper gives insight on the WebSocket communication method in Internet of Things system, where the hardware part of the system is based on ESP32 microcontroller. Method of implementation is discussed and the reliability of the real-time data transfer in Wi-Fi networks is tested and compared with the long-polling method. Special circuit is designed with the goal to stress the hardware part of the system and the client-server communication link in order to enable proper comparison of data transfer methods. For the comprehensive testing of the real-time data flow, a web server application is designed and used to visualize received data. Impact of RSSI on transfer methods is discussed as well. Eciency of the WebSocket method is discussed and then compared to the long-polling method.*

Key words: *Internet of Things, WebSockets, ESP32, JavaScript, Espruino*

1 Introduction

Field of Internet of Things (IoT) technologies experiences dramatic growth in various disciplines during the last decade. This development is based

Received October 18, 2022; revised January 18, 2023; accepted January 27, 2023
Corresponding author: Nikola Mitrović
Department of Microelectronics, University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia
E-mail: nikola.i.mitovic@elfak.ni.ac.rs
* An earlier version of this paper was presented at the 15th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS), 2021, Niš, Serbia [1].

on the remarkable advances concerning hardware circuits, that are able to be interfaced with microcontroller units (MCU) or similar units. Continuous trends, especially in the field of electronics, move in the direction of increased connectivity, that is seen through implementation of IoT technologies and technologies of wireless data access. In 2011, company Cisco IBSG made a prediction that by the end of the decade, more than 50 bilion different devices will be connected to the Internet [2]. It is estimated that this number was exceeded before the end of the decade. Moved with this broad development, many academics, as well as company developers, are exploring the IoT systems and its feasibility.

In previous years, there has been development of free Web platforms and services that can deliver some presentation and monitoring of data in IoT systems [3–6]. Tasmota, ESPhome and similar platforms enable great variety of interfaces, offering diversity to the developers and users to take advantage of different transfer protocols and data transmission methods. Having available options in mind, this paper discusses implementation of the WebSocket method, its basic tests, its comparison to the long-polling method, both implemented on the ESP32 board, in the client-server configuration.

The WebSocket protocol enables full-duplex communication between a client running code to a remote host that has opted-in to communications from that code, in a controlled environment, over single transfer control protocol connection [7]. It was designed as an improved substitute for the existing solutions, based on HTTP, since the HTTP was not initially intended to be used for bidirectional communication. Long polling on the other hand produces a HTTP request to the server, and then keeps the connection open, giving the option to server to reply back. Unlike WebSockets, long polling is supported in the most of devices and browsers, while implementation of Websockets has different variations and implementation methods.

Special attention was given to the coupling and synergy between the firmware of the hardware units and the web server application design. These types of web application are designed based on the models that usually mostly use JavaScript for coding. Because of that, firmware for the hardware was also developed using JavaScript syntax. This coupling enabled matching of data frames and sync, and with usage of appropriate tools, even great simplicity in codes on both sides, owing to the available development environments and flashing tools. Approach like this is selected to allow more reliable data transfer, but also because it can give access to further, more complex development in IoT field [8].

Despite constant emergence of novel tools and environments for software

as well as hardware part of IoT solutions, reports on reliability and efficiency of these tools, along with basic test results are lacking [9–13]. Ubidots, AskSensors and similar platforms indeed offer vast possibilities for many data transfer methods with numerous devices, but rarely give insight on some failure scenarios, especially ones hardware induced. To deal with these restraints and to deliver broader efficiency and reliability assessment, beside standard software and hardware units tests, it is essential to also evaluate the correlation between these units, meaning, to address IoT system level reliability. Reliability of IoT systems at this level is yet to be studied.

2 Literature Review

There have been several reports on analysis and implementation of the WebSocket protocol and related methods in the different types of systems [12–21]. Most of the reports cover software application solutions, without deeper analysis of the performance of portable hardware counterparts. Ma presented real-time monitoring system for intelligent buildings that was based on WebSocket method [14]. In this paper, focus is on the network architecture, where multiple measurement points communicate with the central server. Integration between the different browsers and and the WebSocket compatible server is explored. Findings in this paper suggest that WebSocket method generally causes overall lower latency time than polling method or some other solutions. Hardware implementation of the proposed techniques is beyond the scope of this paper. Soewito presented the WebSocket method as a support to Smart Home implementation [12]. WebSocket method is again compared to some other data transfer method, but the research is concentrated on the maximum bandwidth usage from the different devices in the smart home system. It is concluded that the proposed method is superior for integration with various devices, which was shown using different network protocol analyzers. Karagiannis presented a survey on application layer protocols for IoT where WebSockets were one of the analyzed protocol [15]. It is stated that, compared to some other protocols, WebSocket method demand for more hardware resources. Because of that, in earlier phases of the developent of IoT industry, it was not suitable for some resource constrained devices as some the other protocols and methods. However, with the development of IoT devices, memory constraints ceased to be a factor in many applications, thereby enabling vast usage of WebSocket method, especially in applications where real-time data transfer is needed. Linggarjati reports a monitoring system for cylinder hydraulic of a heavy equipment such as excavator [16].

This system is Wi-Fi enabled and uses WebSocket method as a primary data transfer method to send monitored data to the Web server. Linggarjati also describes the hardware side of the system, where ESP32 system is selected as a viable solution to enable Wi-Fi data transfer. Mesquita presented an overall survey on the general capabilities of the ESP8266 module, predecessor of the ESP32 module [18]. It is said that the one of the important parameters that impact all of the Wi-Fi data transfer methods is Received Signal Strength Indicator (RSSI). Since the antenna is embedded into module, position and the rotation of the module impacts the RSSI. Still it was concluded that when the RSSI is favorable, both of the data transfer methods offer appropriate possibles.

As can be seen from the literature review, there are solutions presenting and implementing WebSocket method for IoT applications. However, testing and performance of these techniques, especially on concrete devices seem not to be investigated thoroughly yet.

3 Experiment Description

Implementation of WebSockets is presented through the operation of characteristically designed IoT system. Parts of the system can be divided into the portable hardware part, where ESP32 board performs as the main unit, and the remote software part, consisting of the designed web application. In this designed IoT system, portable and remote part interact in client-server configuration, where the portable part is the client and the remote web application is the server. Features of both of the parts are more concretely demonstrated.

3.1 Used tools

Since the goal of the IoT system is to make extensive use of the capabilities of the data transfer methods (WebSockets and long polling), an appropriate web server application is developed. The goal and the need for this approach is seen through possible limitations of available off-the-shelf tools. Application is primary designed with the use of node.js [22]. Interface of the developed application is given in the Fig. 1.

Main capability of the web application is to provide mechanisms for continuous reception of the data that is sent by the clients. After the reception, data is processed with the goal to generate data points. Another part of application, based on chart.js, implements a graph and enables plotting of data

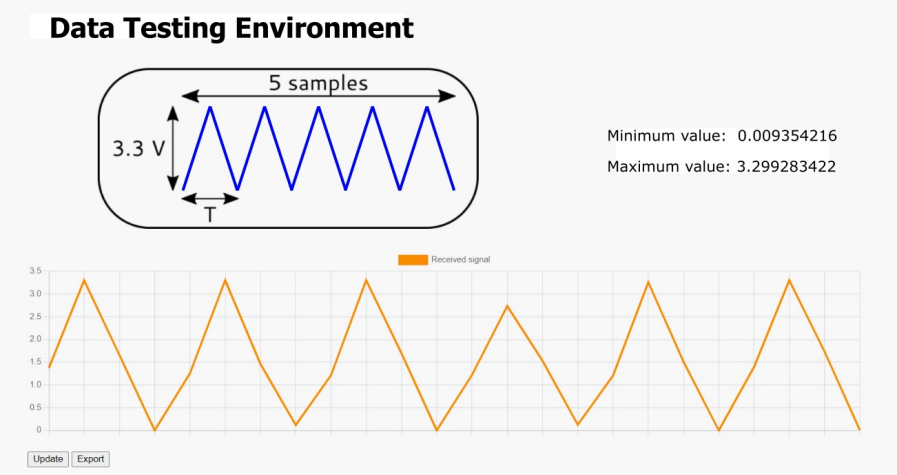

Fig. 1 Interface of the developed web server application

points as soon as they are received, thereby, in real-time. In that way, data points create waveform and give basic visualization of the received data.

In addition to the basic visualization of the data points, functions that perform calculations of maximum and minimum values of the data points are incorporated into the application, allowing calculation of the amplitude and the frequency of the visualized signal. Significant asset of the application is the data logger table. Upon the reception of the data point, its value, as well as reception time (order of magnitude of the milliseconds) and time since the last reception is stored into the data logger table. User can export this data logger table into a csv file, to perform additional investigation and analysis.

Developed web application is deployed with the usage of the Heroku platform. Project files are entirely public and the application is available through the link: `http://telsiks-app.herokuapp.com/`. Although the application will treat tabs in browsers as clients, only marked client (portable unit with ESP32) is able to deliver new data to the application.

Regarding the portable hardware unit of the designed IoT system, central device is a chip that incorporates Wi-Fi module that provides ability to connect to the Wi-Fi network, concretely ESP32-WROOM2 [23]. This chip, produced by the Esspresif company, is mounted on ESP32 board, supports 802.11 b/g/n network protocol in the regular Wi-Fi frequency range (2.4 GHz − 2.5 GHz). Even though its predecessor, wide known ESP8266, is still more present in the community, greater memory capacity as well as advanced RF characteristics are making this chip far preferable for IoT appli-

cations. Aside from Wi-Fi interface, number of available GPIO pins, various peripheral units, real-time clock and bluetooth module promote this chip for even broader field of applications. Development of the firmware for the chip as well as flashing, is done with the aid of Espruino, which is a free ware JavaScript interpreter [24]. Usage of this interpreter allows implementation of WebSockets that is almost the same as in the application (same syntax, same functions), making development of entire system rather simple and straightforward. Development of the microcontroller firmware with the usage of the high-level language as JavaScript is not typical. Still, adopting the same functions for data transfer on both remote application side as well as the hardware side, leads to the better synchronization of the transfer frames and timeouts, which can be significant when the transfer methods are discussed.

3.2 Flow of the Experiments

The aim of the experiment is to implement two data transfer methods, as well as to provide the assessment of the efficiency along with the discussion of possible errors. To perform this analysis, it is needed to establish test signal data that will be transferred using both of the methods. For the experiments, although it is not mandatory, characteristic triangular waveform of the test signal is applied, with the intent to enable that possible mismatch between the test signal waveform and received data points waveform be evident. Test signal needed for experiment is generated using another device, also a microcontroller. Main purpose of using another MCU for test signal generation, specifically Cortex-M3 core based STM32F302R8 [25], is to provide options for arbitrary programmable test signal waveform that can be adapted to the needs of the experiment. Main elements and the course of the experiment is depicted in the Fig. 2.

Chosen MCU is mounted on the xNucleo board, together with the external oscillators and the header that gives access to the various peripheral units that MCU posses. One of those units is integrated DAC (Digital to Analog Converter). With proper settings, this unit is able to deliver arbitrary waveform ranging to 100 kHz in frequency and to 3.3 V in amplitude. Pressing of the board button which is connected to the pin PC13 of the MCU activates the DAC that generates test signal on the PA4 pin of the STM32F103.

To provide better insight into the flow of the experiment, both STM32 and ESP32 chips are connected with the TFT displays. These multi-color displays are controlled over ST7735 driver which is, through SPI interface, connected to the each chip (in the Fig. 2, display 1 is connected with the STM32

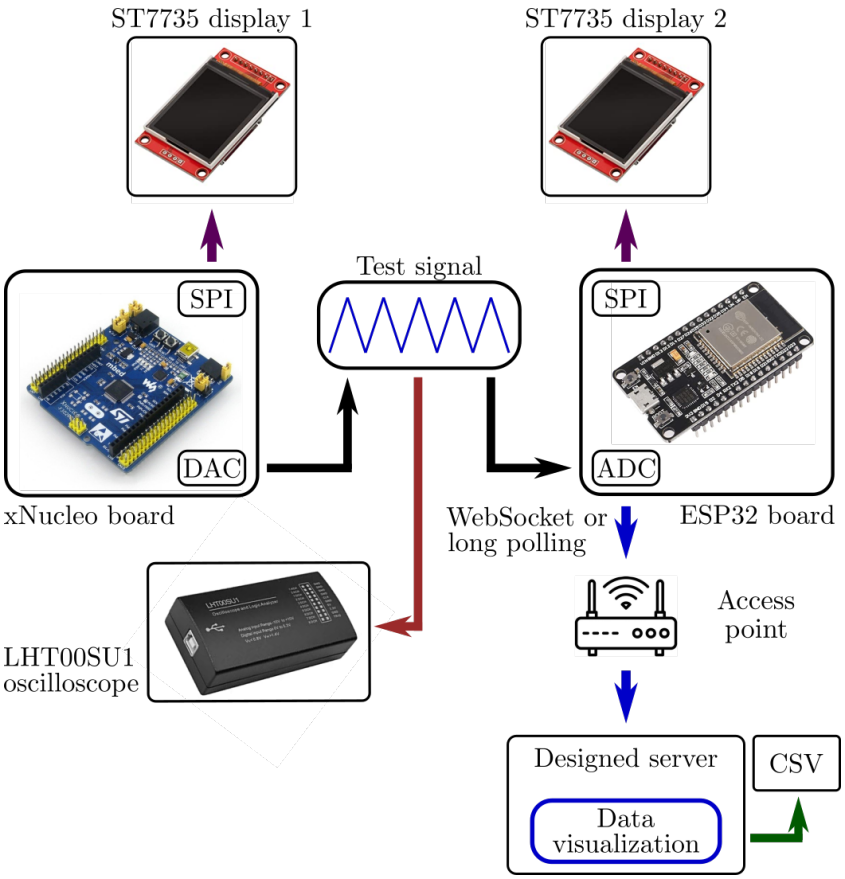

Fig. 2 Block schematic of the main elements and the course of the experiment

and the display 2 with the ESP32 chip). On the chips power-on or reset, displays are turning on, printing message "Welcome" on the screens. After five seconds, ESP32 chip attempts to connect to the pre-specified hardcoded credentials of the Wi-Fi network (pre-specified meaning that the credentials (SSID and password of the Wi-Fi network) must be inserted during the firmware development for the ESP32). If the attempt is successful, system enters idle status and appropriate message ("STATUS: IDLE") gets printed on the ST7735 display 2. On the other hard, if the attempt was not successful, and the connection was not established during the 15 seconds of attempting, system enters not initialized status, where again appropriate message gets printed on the screen. Working diagram of the setup is presented in the Fig. 3, where black elements present the working steps of ESP32, while blue ones present the working steps of STM32. If the system is in not initialized status,

it is needed to press ESP32 board button (which is connected to the pin D9). In that way, system will attempt to establish connection again.

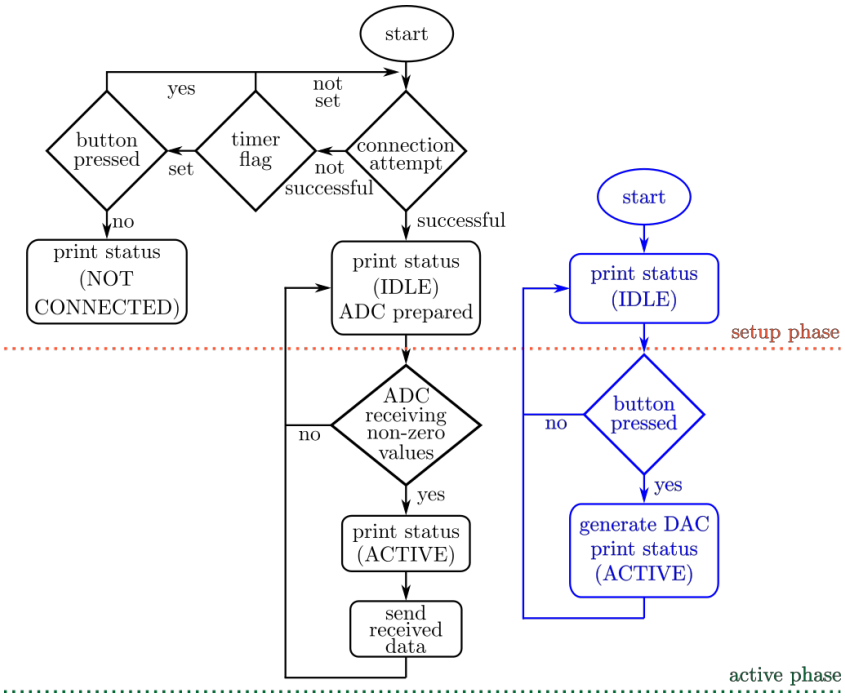

Fig. 3 Working diagram of the system workflow

When the both parts of the system confirm idle status, next phase of the experiment begins. Pressing of the xNucleo board button activates the DAC peripheral unit of the STM32 MCU. On activation DAC generates a series of five samples, where samples being the symmetrical triangles of programmable period and amplitude of 3.3 V. Upon button pressing, message gets printed on ST7735 display 1 to inform on DAC activation. Reason for not using pre-programmed time dependent procedures to activate the DAC and engage this phase of the experiment is to make described steps event driven (event of a button press) and not time driven. In wide area of IoT system usage, most of the processes are initiated with some specific targeted events (detector detects the presence of some gasses or pollutants, measuring unit measures critical value of a magnitude, different callbacks and similar), and based on these event can be further developed. Therefore, this approach was selected, whereas the selection of different approach does not impact experiment results.

After generation, test signal is forwarded to the LHT00SU1 USB oscilloscope and to the ADC pin (D35) of the ESP32 board. Chosen oscilloscope was selected for its good mobility as well as for simple USB interface to the laptop and sigrok's Pulseview [26], where its output can be observed, configured and obtained for further analysis. On the pin D35 of the ESP32 board, ADC is initiated and digital values that are product of ADC gets sent with targeted transfer method (either WebSockets or long polling). Activation of the ADC triggers process of printing a message on ST7735 display 2 that the data is being processed. Wireless data transfer to the developed application (that acts as a server) from the ESP32 (that acts a client) occurs until the ESP32 sends data, meaning until the values for all of the five samples are sent. When no data is received on the side of application for five seconds (after reception of some data), application (server) notifies ESP32 (client) that the set of samples is finished. That leads the system back to the idle status, which gets printed on the ST7735 display 2. Setup of the described experiment flow is depicted in the Fig. 4.

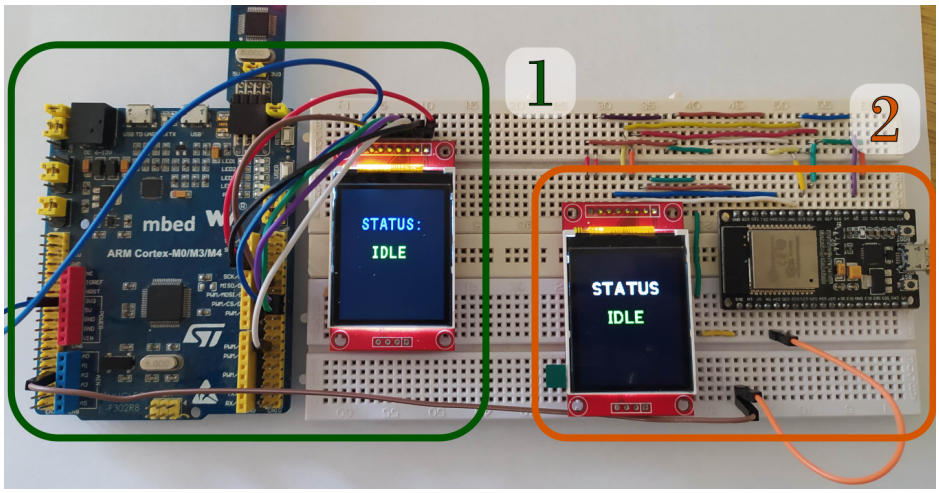

Fig. 4 Experiment setup connected and in idle state: 1) Test signal generator part designed around xNucleo board; 2) Part designed around ESP32 board

4 Experimental Results

Extensive testing in the presented manner was conducted in order to test and evaluate these wireless data transfer methods. During the first series of experiments, testing signal period was 1.8 s. Experiments were conducted and the plotted data points from the developed application are shown on

the Fig. 5. Visual evaluation of results of this series of experiments suggests that both of the data transfer methods (WebSocket and long polling) do deliver viable results, despite some anomalies and deviations, slightly more emphasized with the long polling method. Still, that is only visual evaluation. Measurable results are presented in a different way.

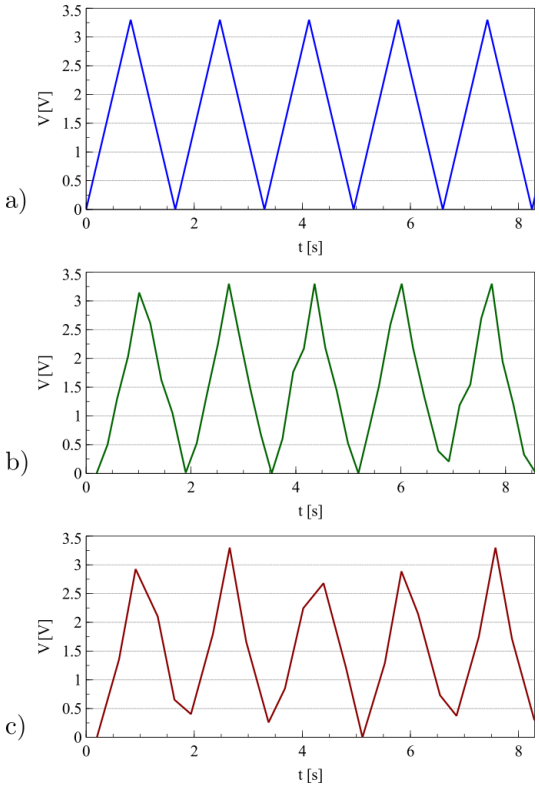

Fig. 5 Waveform of the signals a) Testing signal (monitored with LHT00SU1); b) Created with received data points using WebSocket method; c) Created with received data points using long polling method

Working principle of the long polling method imply that triggering events, regardless of quantity or the frequency of events, initiate issuing of the new HTTP request on every event, as soon as it can be issued. Still, based on the quality of the network and the technology behind the communicating parts, time span for completion of a request can vary, but has an average value of around 350 ms for ESP32 systems [8, 9]. In some applications, where the input data that should be sent, constantly generates rather fast, part of that input data never gets sent, or gets sent with evident delay. It simply gets

substituted faster than it gets send, and then, based on the implementation, it is either skipped, or, more frequently, put into a buffer where waits empty slot for sending. This principles, applied to the analysis of the results of this experiments, point to the possibilities of increased visual and technical difference between the testing signal waveform and the long polling received data points waveform, as the frequency of the testing signal increases.

Working principle of the WebSocket method presents different paradigm, that, incidentally, can avoid some of the disadvantages of the long polling method [11]. Rather than continuously opening multiple HTTP connections, this method relies on basic message framing, layered over TCP [7]. This approach increases speed of the data transfer, but the WebSocket remains open between single client and the server. Sending data from another client would request closing of the socket to the first client, and then opening the socket for the other one. This method also has its disadvantages, especially in the networks with the high number of concurrent clients.

Absolute error of the data transfer methods between the testing signal and the received data points is presented in the Fig. 6. Absolute error is measured as the difference between the received data point and the closest point from the testing signal waveform. Results presented in the Fig. 6 suggest that the absolute error is slightly more notable when the long polling is the data transfer method. Even then, critical ratings do not surpass 20 % (0.7 V), while the average relative error is around 8 %.

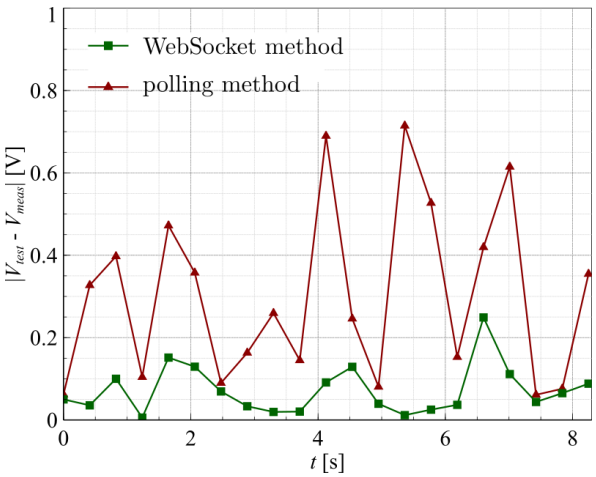

Fig. 6 Absolute errors between the received data point and the closest point from the testing signal using different methods (period 1.8 s)

On the other hand, critical results are identified in the second series of experiment, with the doubled frequency of the testing signal. Results of this series are presented in the Fig. 7. This case gives notable difference between the absolute errors when the long polling method is used and when the WebSocket method is used. For this type of application WebSocket method offers significantly improved performance, and yet demands similar resources of ESP32 chip as the other method. Relative error in this case is in the same level as in the previous series of experiments, meaning below 5 % (while absolute error averages 0.15 V), whereas the average relative error when the long polling is the data transfer method is more than 20 %.

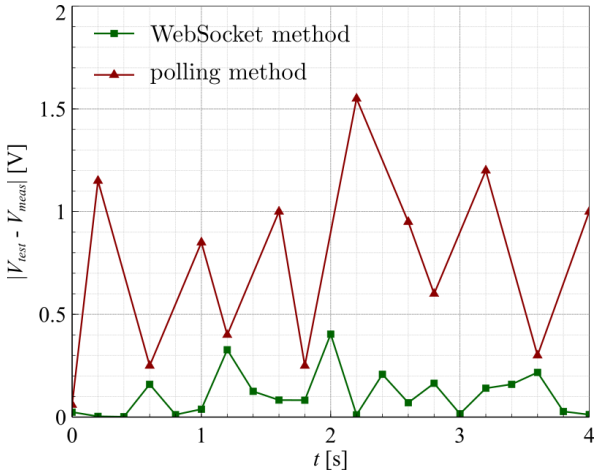

Fig. 7 Absolute errors between the received data point and the closest point from the testing signal using different methods (period 0.9 s)

Regarding the overall assessment of the WebSocket method, important output of the previous series of experiments is the number of receptions, during the reception period. As stated earlier, developed server application has the possibility, among other, to also measure time between two receptions. Despite the different testing signal frequencies and also regardless of the resolution of the ESP32 chips' ADC, transfer methods initiate data transfer with different rates for the same Wi-Fi signal strength. For the implementation of the WebSockets on ESP32 chips, that target rate is on around 150 ms, while for the long polling method is more than twice that (around 350 ms). It is also worth mentioning that for all of given analysis, there is also impact of the delay of the communication itself, so that these numbers cannot be taken absolutely. Still, since the amount of data sent is not big enough to emphasize this difference, this impact can be considered negligible.

Still, efficiencies of the both of the discussed transfer methods are dependent on the strength of the Wi-Fi signal in each of the methods. Because of that, as in similar researches, the impact of the Wi-Fi signal strength (seen through the value of received signal strength indicator (RSSI)) on methods of data transfers is discussed as well [18,27]. Values of RSSI are ranging from $-120\,\mathrm{dB\,m}$ to $0\,\mathrm{dB\,m}$. Values closer to 0 show better stregth of the signal, meaning better network and more reliable data transfer. Still, most of the IoT system often operates in the environments where RSSI of the Wi-Fi network is not that good (RSSI $= -60\,\mathrm{dB\,m}$ and lower) [3,10,11,14]. Therefore, for the testing of the proposed data transfer method on ESP32 device, similar measurements should be done, but on the different RSSI values. Firmware for the ESP32 was adjusted so that it can also print RSSI value on the TFT display to enable easier positioning, as shown in the Fig. 8.

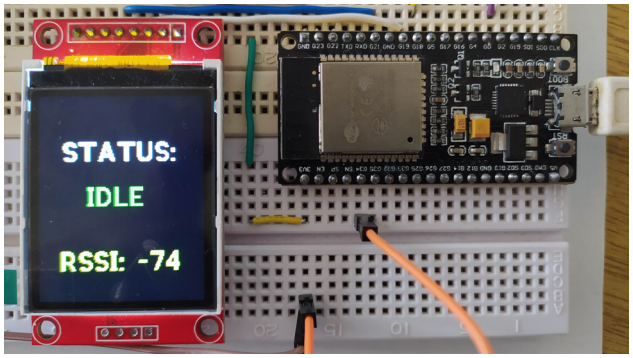

Fig. 8 RSSI value shown in the TFT display connected to the ESP32 device that performs as a client

For that reason, third round of experiments was executed. It was conducted in a big room, where there should be no obstacles for the propagation of the signals. Access point was put in the center of the room, and the value of RSSI was measured at several points in the room, marked as measuring points (MP). All of the points were in the clean line of sight to the access point, on a straight line, with different distances to the access point. Measuring points were selected based on the distance to the access point. First two point were selected to be closer to the access point, with better signal strength, while other three were selected to be farther of the access point, meaning lower Wi-Fi signal strength. Position of the MPs is illustrated in the Fig. 9.

Wi-Fi module is positioned at each of the measuring points. At each point, same testing signal waveform, with lower and higher frequency is sent

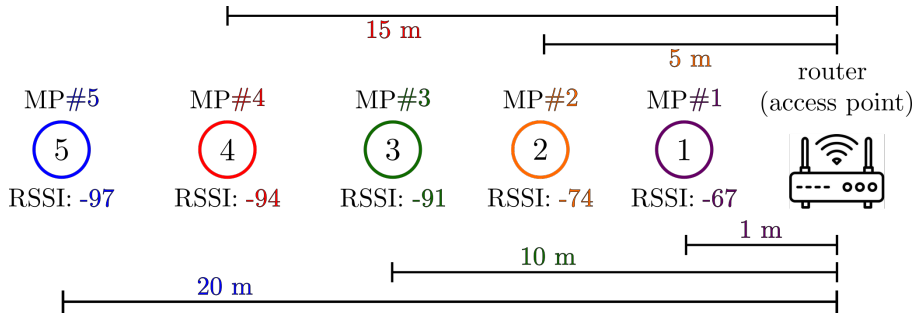

Fig. 9 Position of measuring points (MP) with distance to the access point and RSSI values (RSSI values in dbm)

to the router (access point) using long polling method and after that using WebSocket method (similar like in the first and the second round of experiments). Process of sending data using both of the methods for both of the testing signals is repeated ten times, with the goal of obtaining more comprehensive and accurate results. For clarity of the results, only average relative error in each repetition is presented. Relative errors during the transfer of the lower frequency testing signal for both of the methods in each of the 10 repetitions is shown in the Fig. 10 a) and relative errors during the transfer of the higher frequency testing signal for both of the methods in each of the 10 measurements is shown in the Fig. 10 b).

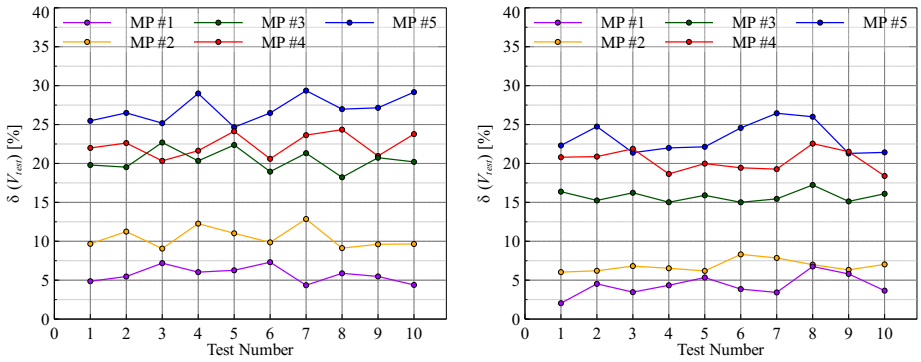

Fig. 10 Average absolute error of the received signal using long polling method: a) Triangular waveform, $f = 1\,\mathrm{Hz}$; b) Triangular waveform, $f = 2\,\mathrm{Hz}$

As can be seen from the Fig. 10 and 11, efficiency of both data transfer methods alters with different distance between ESP32 and the access point, or more concretely with different RSSI. Average values of relative errors for these experiments are given in the Table 1.

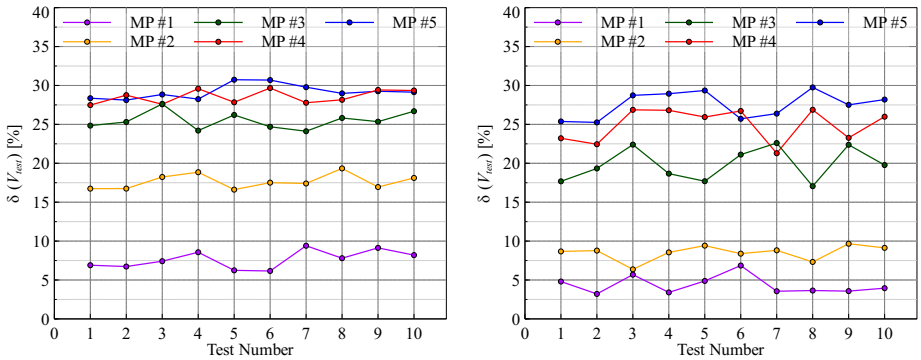

Fig. 11 Average absolute error of the received signal using WebSocket method: a) Triangular waveform, $f = 1\,\mathrm{Hz}$; b) Triangular waveform, $f = 2\,\mathrm{Hz}$

	Long polling		WebSocket	
Measuring point	$f = 1\,\mathrm{Hz}$	$f = 2\,\mathrm{Hz}$	$f = 1\,\mathrm{Hz}$	$f = 2\,\mathrm{Hz}$
MP#1	5.7217	7.6532	4.3195	4.3559
MP#2	10.4359	17.6505	17.6505	6.8274
MP#3	20.4151	25.4855	15.7679	19.8694
MP#4	22.4027	28.5626	20.3356	24.9428
MP#5	27.9943	29.2156	28.2352	28.6152

Table 1 Comparison of average relative errors

However, as can be seen from these figures, difference between the efficiency of the methods is more emphasized with higher RSSI values. At these values, WebSocket method gives evidently better results when compared with the long polling method. This is seen through the magnitude of the relative error between the testing signal values received on ESP32 device and the received signal values on the server application. At lower RSSI values, efficiency of the WebSocket method on ESP32 also tends to decrease. When decreased, there are no drastic differences between the efficiency of methods, although error with the WebSocket method is again slightly lower.

WebSocket present a method that can give two-way communication in client-server configuration. To provide comprehensive analysis of the reliability of the method, it would be needed to take into account additional parameters, as the impact of the data transfer rate, especially when greater size of data is being sent, or the impact of existance of additional traffic in the network, or activity of other clients in client-server configuration. Future steps in research include the testing of the signal receiving on the ESP32

chip, even though this is not that much represented in practical application. Long term development goal of the project is to provide techniques for the low-cost wireless oscilloscope, that is able to conduct basic measuring, deliver proper visualization, but through the browser, where it will also be able to log the results on remote database. That will enable better teamwork for remote working embedded developers, as well as greater possibilities for saving and sharing data. For application like this, some other methods as SSE (Server-sent Events) needs to be discussed, but the general approach would be based on WebSockets.

5 Conclusion

Implementation of WebSocket method in ESP32 based IoT systems and its basic testing is presented in this paper. Every part of the setup of the experiment is described, together with the experiment workflow. Multiple series of the experiments were carried out and pointed to the conclusion that the implemented WebSocket method presents a viable solution for ESP32 based IoT systems, and that it also performs superior to the more represented long polling method in some applications that demand high frequency of data sending. Findings also suggest that the WebSocket method produces delays and errors with the low RSSI, but still smaller than long polling method. Additional investigation is needed in order to asses the efficiency of the method in IoT systems with larger ESP client number under different signal strength values.

Acknowledgments

This paper was realized within the project financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia (No. 451-03-9 / 2021-14 / 200102).

Literatura

[1] N. Mitrović, M. Đorđević, S. Veljković, and D. Danković, "Implementation of WebSockets in ESP32 based IoT Systems," in *2021 15th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, Oct. 2021, pp. 261–264.

[2] D. Evans, "The Internet of Things: How the Next Evolution of the Internet is Changing Everything," *Cisco Internet Business Solutions Group (IBSG)*, Apr. 2011.

[3] H. M. Al-Kadhim and H. S. Al-Raweshidy, "Energy Efficient and Reliable Transport of Data in Cloud-Based IoT," *IEEE Access*, vol. 7, pp. 64 641–64 650, 2019.

[4] M. B. Yassein, M. Q. Shatnawi, and D. Al-zoubi, "Application layer protocols for the Internet of Things: A survey," in *2016 International Conference on Engineering & MIS (ICEMIS)*, Sep. 2016, pp. 1–4.

[5] M. Đorđević, B. Jovičić, S. Marković, V. Paunović, and D. Danković, "A smart data logger system based on sensor and Internet of Things technology as part of the smart faculty," *J. Ambient Intell. Smart Environ.*, vol. 12, no. 4, pp. 359–373, Jan. 2020.

[6] D. Danković and M. Đorđević, "A review of real time smart systems developed at University of Niš," *Facta Univ. - Ser. Electron. Energ.*, vol. 33, no. 4, pp. 669–686, 2020.

[7] I. Fette, A. Melnikov, The WebSocket Protocol, December 2011.

[8] A. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," in *2017 Internet Technologies and Applications (ITA)*, Sep. 2017, pp. 143–148.

[9] M. Babiuch, P. Foltýnek, and P. Smutný, "Using the ESP32 Microcontroller for Data Processing," in *2019 20th International Carpathian Control Conference (ICCC)*, May 2019, pp. 1–6.

[10] N. Mitrović, M. Đorđević, S. Veljković, and D. Danković, "Testing the efficiency of Wi-Fi data transmission in ESP-based IoT systems," *E-business technologies conference proceedings*, vol. 1, no. 1, pp. 172–176, Sep. 2021.

[11] V. Tyagi, N. Rawat, and M. Ram, "Reliability modelling and sensitivity analysis of IoT based flood alerting system," *Journal of Quality in Maintenance Engineering*, vol. 27, no. 2, pp. 292–307, Jan. 2020.

[12] B. Soewito, Christian, F. E. Gunawan, Diana, and I. G. P. Kusuma, "Websocket to Support Real Time Smart Home Applications," *Procedia Comput Sci*, vol. 157, pp. 560–566, Jan. 2019.

[13] E. N. Živanović, "Influence of combined gas and vacuum breakdown mechanisms on memory effect in nitrogen," *Vacuum*, vol. 107, pp. 62–67, Sep. 2014.

[14] K. Ma and R. Sun, "Introducing WebSocket-Based Real-Time Monitoring System for Remote Intelligent Buildings," *Int J Distrib Sens N*, vol. 9, no. 12, p. 867693, Dec. 2013.

[15] V. Karagiannis, P. Chatzimisios, F. Vázquez-Gallego, and J. Alonso-Zárate, "A Survey on Application Layer Protocols for the Internet of Things," *Transaction on IoT and Cloud Computing*, vol. 1, no. 1, Jan. 2015.

[16] J. Linggarjati, "Design and Prototyping of Temperature Monitoring System for Hydraulic Cylinder in Heavy Equipment using ESP32 with data logging and WiFi Connectivity," *IOP Conference Series: Earth and Environmental Science*, vol. 998, no. 1, p. 012042, Feb. 2022.

[17] L. Mastilak, M. Galinski, I. Kotuliak, and M. Ries, "Improved Smart Gateway in IoT," in *2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Nov. 2018, pp. 349–354.

[18] J. Mesquita, D. Guimarães, C. Pereira, F. Santos, and L. Almeida, "Assessing the ESP8266 WiFi module for the Internet of Things," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sep. 2018, pp. 784–791.

[19] T. M. Behera and S. K. Mohapatra, "Improving Network Lifetime by Minimizing Energy Hole Problem in WSN for the Application of IoT," *Facta Univ. - Ser. Electron. Energ.*, vol. 31, no. 2, pp. 267–277, Feb. 2018.

[20] N. Mitrović, S. Veljković, Z. Prijić, and D. Danković, "Comparison of the performance of the different GPS receivers in practical applications," in *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, May 2022, pp. 11–16.

[21] N. Mitrović, M. Đorđević, S. Veljković, and D. Danković, "NFC enabled Wi-Fi managging system for ESP32 based IoT system," *E-business technologies conference proceedings*, vol. 2, no. 1, pp. 57–60, Jun. 2022.

[22] Node.js Available at: https://nodejs.org/en/download/.

[23] ESP32-WROOM2 datasheet. Available at: https://www.espressif.com/site.

[24] Espruino tool. Available at: https://www.espruino.com/Download.

[25] XNucleo development board datasheet, 2014. Available at: https://www.waveshare.com/wiki/XNUCLEO-F302R8.

[26] PulseView tool. Available at: https://sigrok.org/wiki/LHT00SU1.

[27] R. Rosli, M. Habaebi, and M. Islam, "On the analysis of received signal strength indicator from ESP8266," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 3, pp. 933–940, Sep. 2019.