Testing the efficiency of Wi-Fi data transmission in ESP-based IoT systems

1st Nikola Mitrović
Department of microelectronics
Faculty of electronic engineering
Niš, Serbia
nikola.i.mitrovic@elfak.ni.ac.rs

2ndMilan Đorđević
College of Applied Technical Sciences
Niš, Serbia
milan.nebojsa.djordjevic@gmail.com

3rd Sandra Veljković
Department of microelectronics
Faculty of electronic engineering
Niš, Serbia
sandra.veljkovic@elfak.rs

4th Danijel Danković
Department of microelectronics
Faculty of electronic engineering
Niš, Serbia
danijel.dankovic@elfak.ni.ac.rs

***Abstract*—This paper describes process of designing and testing an Internet of Things (IoT) system for continuous receiving of the input data. The goal is to design a custom IoT system and to test the reliability of the designed system, but also to offer solutions for the improvement. Entire process of designing of both hardware part of the system and the software part of the system is explained and main tool used are described. Main characteristics of the designed systems that are tested are basic RF characteristics but also transmission of data of various waveforms. Implementation and analysis of this type of testing data is important, especially because properties that are tested are part of the majority of modern IoT systems.**

Keywords—Internet of Things, Wi-Fi, ESP32, reliability, JavaScript

I. INTRODUCTION

Interest in the development of the IoT systems is continuously growing. Most of the newer devices and systems are already IoT enabled (in terms that they can connect to a remote server or to the cloud and exchange data), while older systems are being empowered with the modules and subsystems that can enable remote data monitoring. In 2011. Cisco IBSG predicted that there will be more than 50 billion devices connected to the Internet by 2020 [1]. Significant improvement in hardware Wi-Fi modules, especially with the emerging of the companies from the East is offering greater possibilities of engaging with these types of modules. Low cost and yet high performance are making them suitable for many applications and for wider research. With the appearance of the ESP8266 module, produced by Espressif in 2014, many possibilities for the application in embedded systems have arisen [2]. Further development of the hardware led to the creation of ESP32 SoC (System on Chip), which is a device that can match performance of the high-speed microcontrollers (MCU), and yet fully utilize integrated Wi-Fi module [3].

With improved availability of these devices, a lot of systems in various industrial areas are designed. This type of systems mostly consists of some modules that collects data from the environment and transmit that data to the remote server or a software platform. In medicine, IoT systems on ESP32 found application in vital health signs monitoring [4]. Kristiani et al. reported a system consisting of multiple heart rate and respiration rate modules. These devices monitor basic vital signs and in case of the irregularities user is informed through Wi-Fi connected application. Rai proposed a smart surveillance system [5]. This system acquires continuous video and transmits it using Wi-Fi capabilities of ESP32. Dhingra presented an air pollution monitoring system [6]. Many gas detecting units are connected to the Arduino board that is controlling ESP8266 devices. User can monitor activity of the gas detecting units over an Android application. Increased interest in this type of systems even led to creation of the software platforms (such as ThingSpeak or RainDrops). These types of platforms offer possibilities to the IoT system designers to pass around software part of the programming and to focus on the hardware connections and devices. These platforms even support various types of the wireless transmission (GSM modules and similar) [7,8]. In most of these types of applications, aggregated signals are mostly in limited voltage range and in limited frequency range.

Even though that there are reports on the design and operation, reports concerning reliability of this type of systems are lacking [9]. Investigation on the reliability of the software part of the IoT system is given by Meneghello [10]. It states that many low-end IoT commercial products do not yet support security mechanisms. In the matter of reliability of the hardware part and the system in general, Montoya-Munoz reported an approach based on Fog Computing [11]. This approach improves reliability of the data collection process focusing on outlier detection. Assessment of the reliability of the parts of IoT system is yet to be further analyzed.

II. DESIGN AND EXPERIMENT SETUP

The goal of the experiment setup is to design an appropriate environment for the receiving data, transmission of the data and visualization of data. Experimental setup consists of two parts, hardware part and software part.

A. Hardware part

Two main approaches of using Wi-Fi modules in embedded systems are noted [2,3]. First, that is using a

controlling, driving microcontroller that controls MCU with Wi-Fi module and second, where Wi-Fi powered MCU performs as a standalone system. First method is characteristic for older generations of Wi-Fi modules where MCUs consisting Wi-Fi modules are not able to deliver high performance (mostly used for the ESP8266-01 that is controlled by more powerful MCU using UART over AT commands). Second method became more versatile when the development of the chip consisting both Wi-Fi module and many serial interfaces came to be. This allowed users to use less devices for the system design and therefore to reduce energy demand and system cost. However, in the system where MCU with higher performance is needed, first method is the only method that can deliver appropriate results. For this research, only second method is used. Still, other MCU (STM32 chip with DAC (Digital to Analog converter) is used, but not to control standalone Wi-Fi modules, but to generate various waveforms that are used as testing signals for the Wi-Fi modules. Wi-Fi MCU accepts these signals and transmits them. Block scheme of the experiment is shown in Fig. 1.

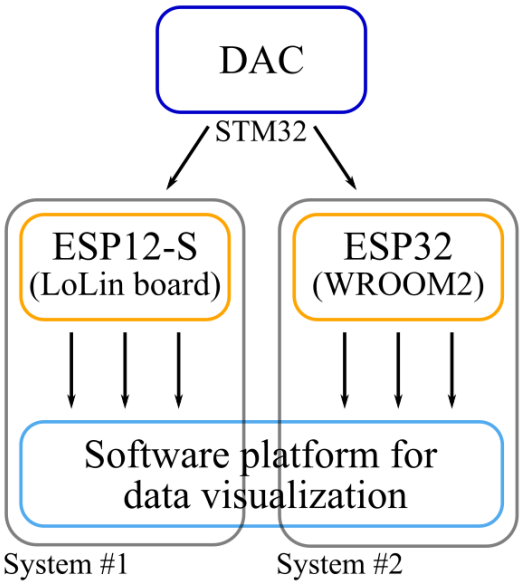

Fig. 1. Block scheme of the experiment environment.

As shown in the Fig. 1, two different systems are analyzed. In the both systems, data waveform is generated using STM32 MCU, specifically STM32H755 mounted on a Nucleo-H755ZI board [12]. Systems use identical software part of the system, while the hardware part is different in each. First system is built around ESP12-S chip [13]. This module is the improved version of the ESP8266 chip, mounted on a LoLin board. Board also contains CH340G chip (UART to USB converter) that enables flashing of the ESP12-S chip through UART. Second system is built around ESP32 WROOM2 chip [14]. This chip is also mounted on a board containing an UART to USB converter for chip flashing and other components to use with the chip peripherals. Detailed comparison of the used chips and boards is given in the Table I.

TABLE I. SPECIFICATION OF THE USED WI-FI MODULES

	ESP12S	ESP32
Voltage [V]:	2.5 - 3.6	2.5-3.6
Number of GPIO:	17	34
SRAM [kB]:	160	520
Wi-Fi Connectivity:	IEEE 802.11 b/g/n, 2.4 GHz	IEEE 802.11 b/g/n, 2.4 GHz
ADC:	10-bit	12-bit

Both of the Wi-Fi chips are programmed using Espruino [15]. Espruino is an open-souce JavaScript interpreter for the MCUs. While it is generally developed for a variety of MCUs, it is mostly used for systems including Wi-Fi interfacing. JavaScript encourages event-based programming. From the hardware standpoint, it can lead to the lower power consumption and better handling of the events. Main drawback of Espruino is that initial firmware for the chip takes more memory than most of the firmwares, so it is not suitable with the chips with the low RAM. However, both of the used modules have more than enough capacity of the memory, so this limitation is not an issue. After ADC processing, data is sent to the web server using regular http request.

B. *Software part*

Software part consists of the design of the web application for the server as well as client. Application backend is designed using JavaScript, specifically node.js [16]. In the node.js environment, module Express is used for the server setup, and built-in Events module. Module chart.js is used for the graphical interpretation of the data. For the server deployment Heroku platform is used [17]. This platform does not impose limitation to the application related to the neither response rate or the size of data being received for this project. Designed application is available at: http://ebt-app.herokuapp.com/. Interface of the designed application is given in the Fig. 2.

Fig. 2. Interface of the designed web application.

Main part of the interface is the chart that should be showing received waveform. Title of the application and conference logo are placed in the header of the interface. Every received value along with the receipt time is temporarily stored on the server. The data can be exported as a CSV file. Also, minimum and maximum received value are shown above the chart. Those values are used to confirm the amplitude of the received signal.

C. *Experiment flow*

Flow of the experiment is given in the Fig. 3. In each system, using I^2C communication, both Nucleo board and Wi-Fi module are connected to separate OLED displays with SSD1306 driver [18]. Display is used to inform user of the ongoing actions and to report possible errors. DAC generated data is forwarded to the oscilloscope (this is done so that waveform properties can be verified) and to the Wi-Fi modules (so that data can sent to the Web server).

For the experiment, data waveform as in Fig. 4 is generated from the STM32 MCU. Waveform is very simple, triangular waveform with the frequency of 1 Hz and amplitude of 3.3 V. Portion consists of five triangles. Same waveform is delivered ten times to the Wi-Fi module with the pause of 10 seconds. After that, same portion is delivered to the other Wi-Fi module. Waveform generation from the STM32 MCU is issued with the push button.

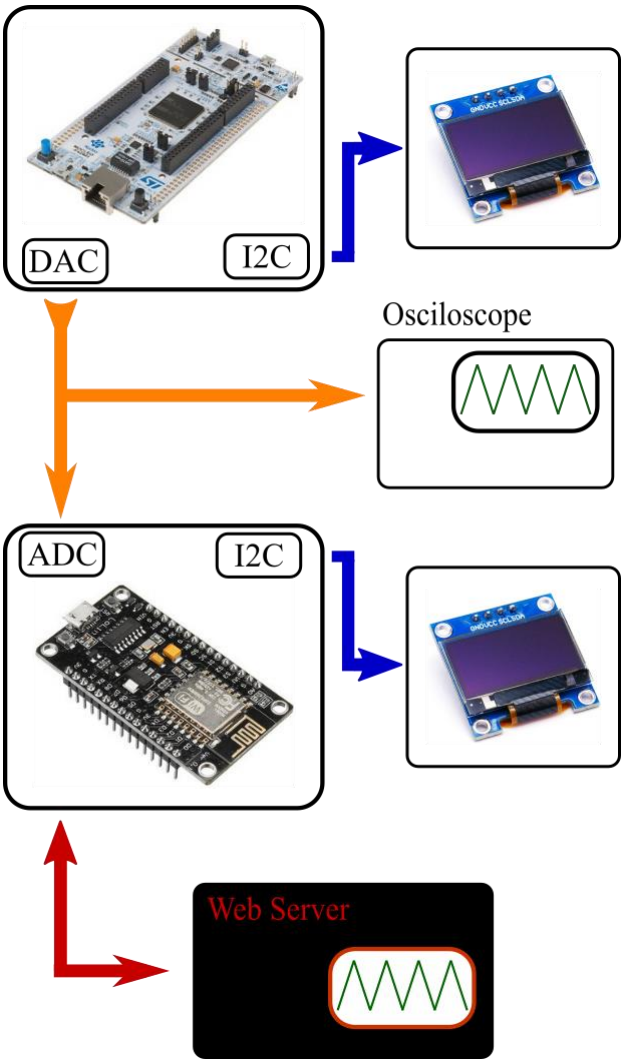

Fig. 3. Scheme of the experimental setup.

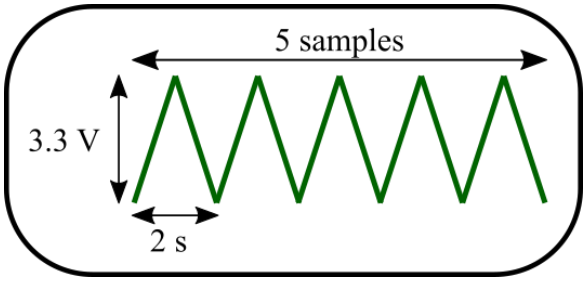

Fig. 4. Test data waveform.

Received data values together with the reception times are being saved at the server and can be exported to the CSV document from the designed server. Server application should recreate signal from the received values and show it in the chart. Algorithm of the system is presented in the Fig. 5.

On the start-up, OLED is turned on. After powering, Wi-Fi module attempts to connect to the Wi-Fi network. While attempting, user is notified with messages through display. Wi-Fi module attempts connecting until connection is established. After connecting, Wi-Fi module is in idle state, it sets STM32 MCU to idle state, and is ready to receive data.

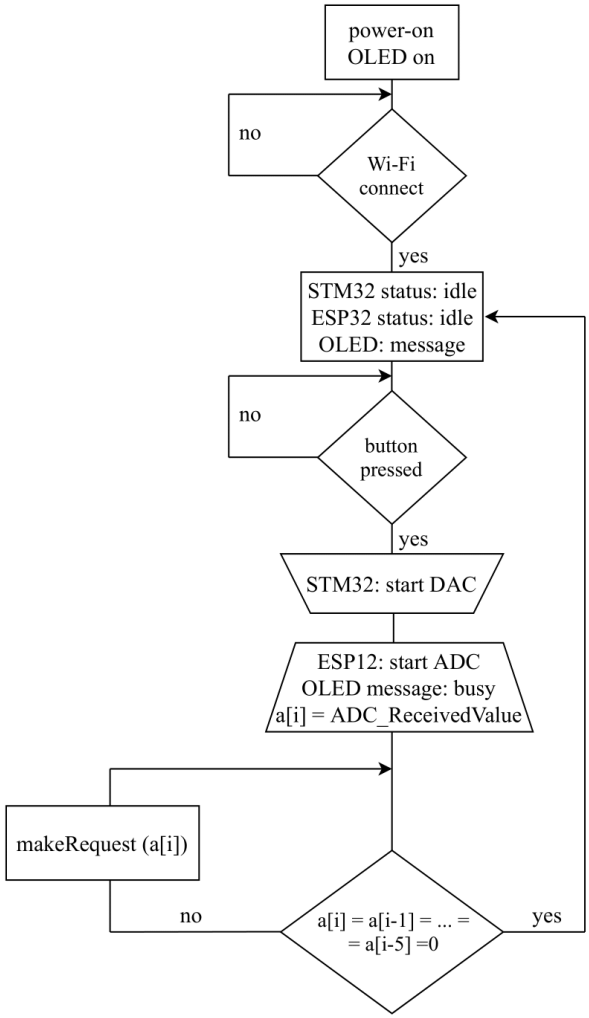

Fig. 5. Algorithm of the experiment environment.

User is notified with the message on the display, so the data generation process can be started. In 100 ms, STM32 checks if the push button is pressed. After pressing, data with five samples is generated from the DAC and forwarded to the Wi-Fi module, where ADC is performed. Data is processed, put into array and checked if it is zero. If not, a http request for sending processed data is issued, so the data is sent to the server. If five consecutive ADC data is zero (no signal, meaning that receipt is finished), Wi-Fi module returns to idle state until new data is received.

III. Experiment Results

Experiments are conducted multiple times and average values from those experiments are plotted in graphs. Form of the received data is given in Fig. 6 and Fig. 7. Fig. 6 shows data received from ESP12 Wi-Fi module while Fig. 7 presents data received from the ESP32 Wi-Fi module. In both of the graphs, waveform looks similar to the test data

waveform, but anomalies are present in both. Several conclusions were taken from these results.

Even though that DAC and ADC are performed continuously, data is not being sent continuously (as expected). As can be seen from the experiment results, new data is given to the server in intervals around 300-350 ms (average 330 ms). That fact is a serious limitation of polling communication. It is caused by http request duration.

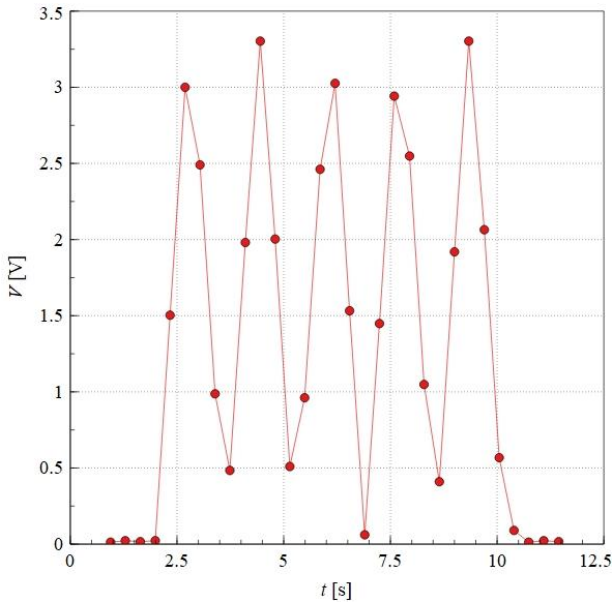

Fig. 6. Received data from the ESP12 chip.

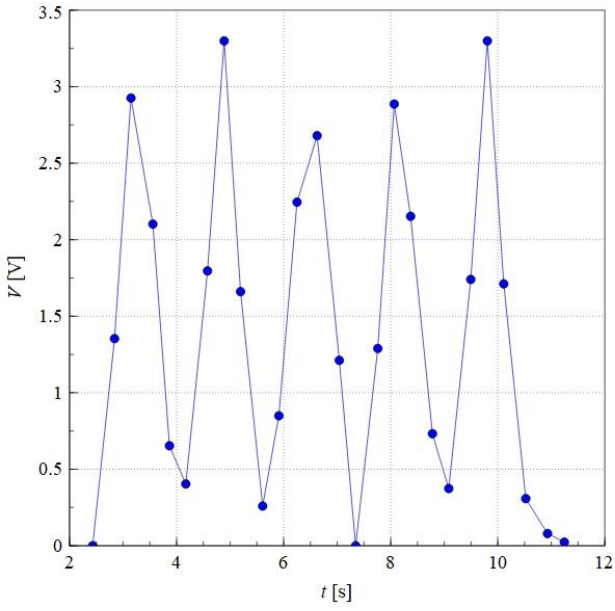

Fig. 7. Received data from the ESP32 chip.

According to the Nyquist criteria, in this case, the waveform to be recreated can have a minimum period of at least 600 ms (frequency of 1.67 Hz). Even then, because of the quantization error of the ADC and noises, received results can differ from the DAC generated data. Still, in the very low frequency range, data can be recreated reliably and completely. As expected, because of the superior capabilities ESP32 performs with greater precision than ESP12 chip. Although, duration of the http request process is similar for both chips.

Data bandwidth and the signal waveform is still limited with various properties. Data generated with DAC cannot be measured fully with the ADC. Then again, even if this limitation is overcame using powerful circuits, Wi-Fi protocol also set limitation in the bandwidth. Because of that, using simple http request method cannot deliver reliable results when the high frequency data is to be delivered in real time. For this type of application, web sockets or some other techniques are needed.

CONCLUSION

Process of designing both of the hardware and the software part of the IoT system is presented. Data is successfully being generated, then received and processed with Wi-Fi modules and transferred using Wi-Fi communication to the designed web server application. Visualization of the data is successful, although efficiency of the process highly depends on frequency of the test data. Limitations to the process are given by multiple factors with http request duration being the greatest one. To improve response time, polling techniques must be substituted with the more compact resources, such as web sockets or improved data buffering.

ACKNOWLEDGMENT

This research was performed within projects No. OI-171026 and TR-32026 supported by Ministry of Education, Science and Technological Development of Republic of Serbia and by Serbian Academy of Science and Arts (SASA) under the Grant No. F-148.

REFERENCES

[1] D. Evans, “The Internet of Things: How the Next Evolution of the Internet Is Changing Everything,” Cisco Internet Business Solutions Group (IBSG), Cisco Systems, Inc., San Jose, CA, USA, White Paper 2011.

[2] R. S. Rosli, M. H. Habaebi and M. R. Islam, “Characteristic Analysis of Received Signal Strength Indicator from ESP8266 WiFi Transceiver Module,” 7th International Conference on Computer and Communication Engineering (ICCCE), 2018, pp. 504-507.

[3] A. Maier, A. Sharp and Y. Vagapov, “Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things,” 2017 Internet Technologies and Applications (ITA), 2017, pp. 143-148.

[4] D. G. Kristiani et al., “The Measuring of Vital Signs Using Internet Of Things Technology (Heart Rate And Respiration),” 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019, pp. 417-422.

[5] P. Rai and M. Rehman, “ESP32 Based Smart Surveillance System,” 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019, pp. 1-3.

[6] S. Dhingra, R. B. Madda, A. H. Gandomi, R. Patan and M. Daneshmand, “Internet of Things Mobile–Air Pollution Monitoring System (IoT-Mobair),” IEEE Internet Things J., vol. 6, no. 3, pp. 5577-5584, June 2019.

[7] 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, July 2020.

[8] D. Danković and M. Đorđević, “A Review of Real Time Smart System Developed at University of Niš,” Facta Universitatis, vol. 33, no. 4, pp. 669-686, Dec. 2020.

[9] 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. 64641-64650, 2019.

[10] F. Meneghello, M. Calore, D. Zucchetto, M. Polese and A. Zanella, “IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices,” IEEE Internet Things J., vol. 6, no. 5, pp. 8182-8201, Oct. 2019,

[11] A. I. Montoya-Munoz and O. M. C. Rendon, “An Approach Based on Fog Computing for Providing Reliability in IoT Data Collection: A Case Study in a Colombian Coffee Smart Farm,” Appl. Sci., vol. 10, no. 24, p. 8904, Dec. 2020.

[12] Nucleo-H755ZI datasheet. Available at: https://www.st.com/en/evaluation-tools/nucleo-h755zi-q.html

[13] ESP12-S datasheet. Available at: https://www.laskarduino.cz/user/related_files/esp-12s.pdf

[14] ESP32-WROOM2 datasheet. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[15] Espruino platform. Available at: https://www.espruino.com/

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

[17] Heroku platform. Available at: https://www.heroku.com/

[18] OLED 0.96” display with SSD1306 driver. Available at: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf