Connectivity Evaluation of ESP32 in Outdoor Scenarios

Pedro Rendeiro
pedro.rendeiro@itec.ufpa.br
Federal University of Pará
Belém, Pará, Brazil

Jamile Leite
jamile.leite@itec.ufpa.br
Federal University of Pará
Belém, Pará, Brazil

Lucas Silva
lucas.damasceno.silva@itec.ufpa.br
Federal University of Pará
Belém, Pará, Brazil

Marcos Silva
marcos.lima.silva@itec.ufpa.br
Federal University of Pará
Belém, Pará, Brazil

George Sales
george.sales@celse.com.br
Centrais Elétricas de Sergipe S.A.
Barra dos Coqueiros, Sergipe, Brazil

Leonardo Ramalho
leonardolr@ufpa.br
Federal University of Pará
Belém, Pará, Brazil

ABSTRACT

The purpose of this paper is to evaluate WiFi (IEEE 802.11 b/g/n) performance of ESP32 modules in outdoor scenarios using an off-the-shelf access point (AP). This was done by measuring its received signal strength indicator (RSSI) and application data transfer rates at different locations of a relatively open area, prone to possible sources of interference, in order to simulate a real-life usage situation. The results show that this setup is suitable for medium data rate applications at distances up to 300m from the wireless access point. Based on the acquired RSSI values, the achievable data rate on the PHY layer was also estimated for each point based on the hardware documentation. Throughout the text, challenges faced during the implementation of the system are shared, as well as the solutions found, which will be useful for those who wish to replicate the experiment.

KEYWORDS

ESP32, Internet of Things (IoT), WiFi

1 INTRODUCTION

The Internet of Things (IoT) [18] market has rapidly expanded over the last years, following the increased demand for communication and control for various devices, gadgets, and applications from industry scenarios [8] to agriculture forestry [1]. The main requirement applied for modern IoT devices is to provide effective connectivity to ensure reliable remote communication and data transfer in a wireless environment.

In order to further develop the IoT ecosystem and expand the area of its applications, powerful low-cost and low-power solutions for IoT devices are required. Typically, each IoT-based unit comprises of a microcontroller (MCU) and a wireless communication module or a combination of both in a single system-on-a-chip (SoC). In particular, the ESP32 hardware modules and the facility of usage of WiFi connectivity (IEEE 802.11 b/g/n) are very popular choices for implementing IoT systems due to their high availability on the market and the great number of resources for testing. [2, 17].

Nevertheless, it is not clear to what extent this combination is suitable for outdoor applications and how far this hardware can communicate with off-the-shelf access points (APs). There are some works that perform experiments to obtain this kind of information. Many of them use received signal strength indicator (RSSI) as metric. Algorithms that use it to estimate location have proven to be advantageous because of their low cost, high coverage, and absence of requirements for hardware adaptations [24]. Furthermore, along with Signal to Noise Ratio (SNR), RSSI is one of the main factors considered in the IEEE 802.11 handover process [16].

For instance, in [4] the authors present a performance analysis of WiFi signal propagation in real scenarios of irrigated agriculture, performing measurements of RSSI as a function of distance from the AP and height above the crop. However, instead of an ESP32 module, the authors use a sensor data logger named WiField [5].

In [3], it is presented a performance analysis of Fine Time Measurement (FTM) implementation in different indoor and outdoor scenarios using an ESP32 module. It also proposed an alternative real-time implementation for distance estimation with the ESP32 using an approach based on machine learning which takes into consideration the RSSI measures from different distances.

Data transfer rate is also an important factor in IoT systems. Higher data transfer rates enable applications that would not be possible otherwise. In [7], the authors show the importance of this metric while evaluating the performance of a wireless network between a device coupled to a drone and to cameras, enabling the transmission of large amounts of photos.

In this paper, we intend to enrich the reference list by testing the network performance of ESP32 using a WiFi connection and Global Positioning System (GPS). A portable hardware prototype was built along with network infrastructure and client-server applications running over TCP/IP in order to create an IoT scenario based on WiFi and ESP32. An off-the-shelf AP was used with a TCP server to calculate RSSI and the application data transfer rates, forwarding the desired information to a database that can be accessed and visualized on a web page, where a map displays the measurements made by the MCU in relation to its location.

The remainder of this paper is organized as follows. Section 2 describes the developed prototype and its environment. Section 3 explains how the tests and measurements were executed. Section 4 shows the results and discussion. Finally, Section 5 concludes the paper and presents future works.

2 SYSTEM DESCRIPTION

2.1 Overview

An overview of the system developed to test the IEEE 802.11 b/g/n performance of ESP32 can be seen in Fig. 1. In order to evaluate connectivity in different positions of an open area, a portable hardware was developed, composed of an ESP32 evaluation board and the GPS module NEO-6M. Both use external antennas, and the former uses an external WiFi omnidirectional antenna (3dBi of gain). The hardware is supplied by a battery to power the whole system.

Designed by Espressif Systems, ESP32 is a family of powerful MCU modules. In this work, we use ESP32-WROOM-32U [14] module, released in 2017 and based on the ESP32-D0WD SoC [13], which uses a dual-core 32-bit LX6 microprocessor. This is one of the product's main advantages since high computation power is a crucial contributor to the progress of IoT solutions [6]. Besides, this module has built-in Bluetooth and WiFi connectivity and the possibility of using an external antenna to enhance downlink and uplink quality.

It must be said that ESP32-WROOM-32U is no longer recommended for new designs since ESP32-WROOM-32UE [15] with ESP32-D0WD-V3 has been released as the newer version of the series in 2020. The authors resorted to the previous model due to the unavailability of the latest one in the brazilian market. However, this does not impact the connectivity tests since this update is related to security and PSRAM memory issues [11].

Moreover, the off-the-shelf AP Action RF 1200 from Intelbras was used to connect the prototype to a local area network (LAN). This product is equipped with four omnidirectional antennas (5dBi of gain each). In this same LAN, a local TCP/IP server was installed. The server is responsible for receiving data from the prototype and sending it to a non-relational database from Google called Firebase Realtime Database [21], which is part of the Firebase development platform. The data can be retrieved by its administrators and is also made available through HTTP requests on a web page. It is also used a smartphone to trigger and monitor every measurement via Bluetooth serial communication.

Fig. 1 shows a simplified block diagram of the setup. Initially, a user starts the measurements using Bluetooth communication between a smartphone and the portable hardware. Then, the ESP32 exchanges messages with a TCP server via the network infrastructure, which is composed of a wireless link followed by Ethernet connections. With these messages, the portable hardware calculates the uplink and downlink data rates. Furthermore, the portable hardware acquires its current position from GPS, RSSI, and timestamp. Then, all this information is sent to the TCP server, which forwards the data to the database. All these steps are better discussed in the next section.

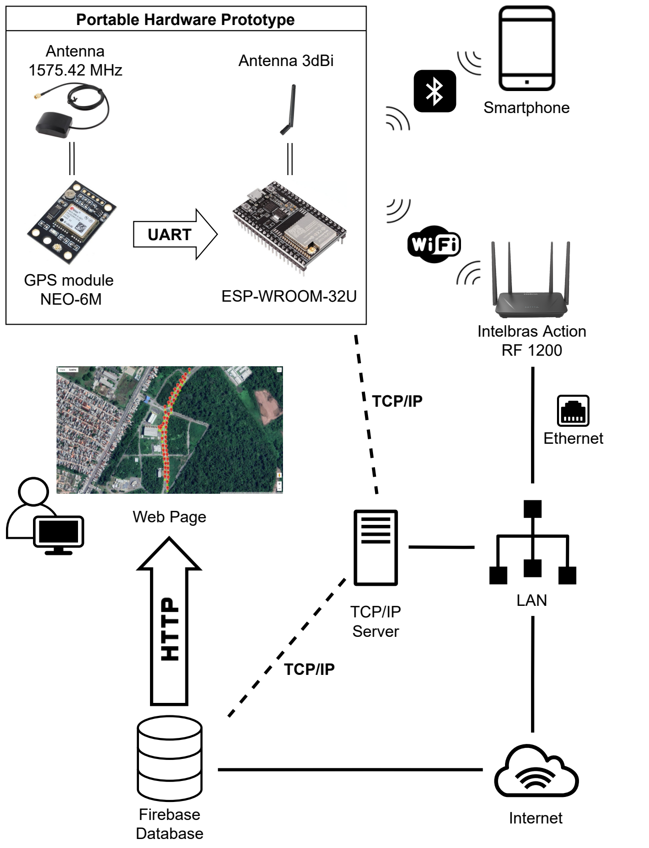


Figure 1: Simplified block diagram of the developed system with portable hardware prototype and software employed.

2.2 Finite State Machines

The firmware developed for ESP32 (client) and software for the local server were designed as finite state machines (FSMs) synchronized with each other in order to capture the RSSI and application data rate. Fig. 2 describes the expected sequence of client and server states defined to implement the expected behavior. Note that *Bluetooth* and *Position* states exist only on the client side, while state *Store* is present only on the server side. The other states were positioned symmetrically in relation to the vertical axis of the image in order to highlight the synchronization between the client and server.

In state 0, ESP waits until it receives a message via Bluetooth and, only then, tries to establish a connection with the server (state 1). Once connected, they both enter state 2, on which the MCU starts sending packets to the server during a pre-defined amount of time and stores the total data, in bytes, that was sent. Next (state 3), it starts receiving packets from the server for the same period of time and stores how many bytes were received. Once this part is finished, ESP32 captures its coordinates from the GPS module through Universal Asynchronous Receiver/Transmitter (UART) interface (state 4). At this point, in state 5, the RSSI value and timestamp in datetime format (*yyyy-mm-dd*T*hh-mm-ss*) are obtained from the network. The timestamp is provided by the Network Time Protocol (NTP), a client/server networking protocol based on the User Datagram Protocol (UDP) for clock synchronization between computer systems and devices in networks with variable latency. Based on the number of bytes successfully sent and received, ESP32 calculates the application uplink and downlink transfer rates in Mbps, respectively.

Still in state 5, all the above information is put into a JSON and sent to the server: timestamp, latitude, longitude, RSSI, uplink data rate, and downlink data rate. Then, the server forwards the data to Google Firebase Realtime Database (state 6).

2.3 Implementation Issues and Proposed Solutions

In the course of the software development, four main failure points were identified: i) interference due to the simultaneous use of Bluetooth and WiFi, ii) GPS data loss, iii) synchronization problems

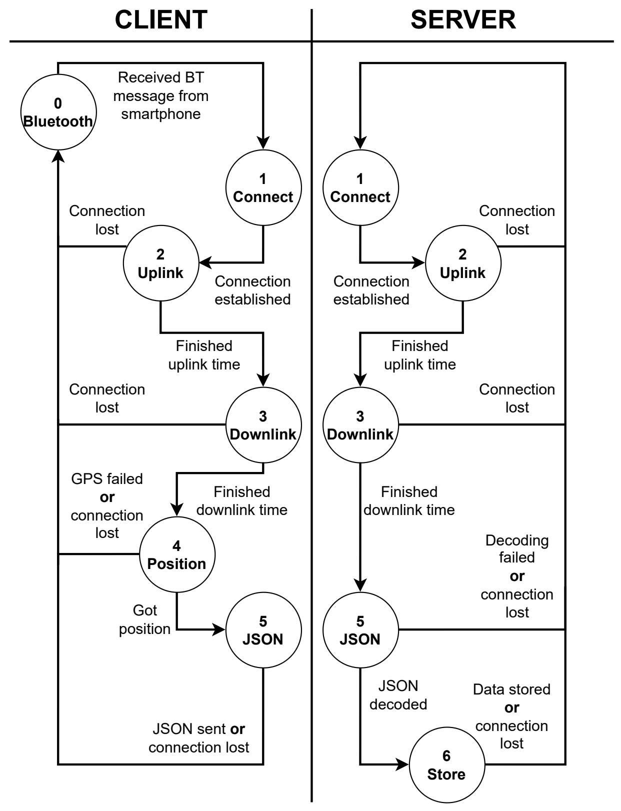


Figure 2: FSMs defined in client (ESP32) and server.

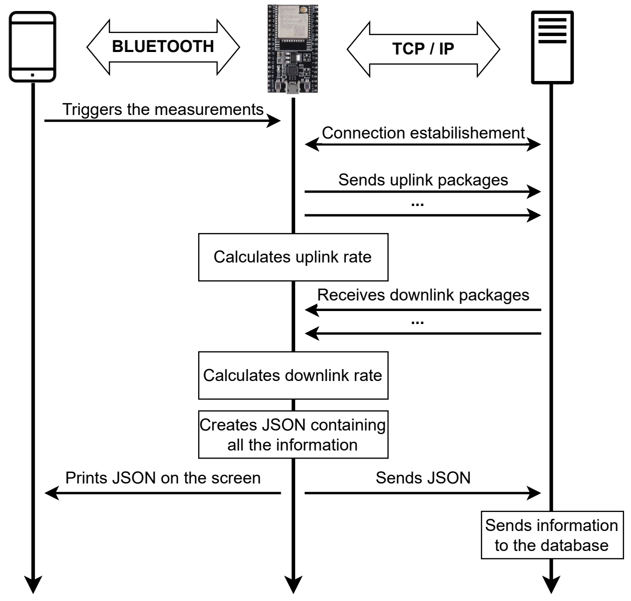

BLUETOOTH
TCP / IP
Triggers the measurements
Connection estabilishement
Sends uplink packages
...
Calculates uplink rate
Receives downlink packages
...
Calculates downlink rate
Creates JSON containing all the information
Prints JSON on the screen
Sends JSON
Sends information to the database

Figure 3: Steps of the measurements.

between the FSMs, and iv) loss of connection between ESP32 and the AP. They are briefly discussed below.

ESP32 has only one radio for both communication protocols (Bluetooth and WiFi), and they do not work properly simultaneously [9]. Thus, Bluetooth is disabled when WiFi is used (states 1, 2, 3, 4, and 5 on the client). Otherwise, the data transfer rate measurements could be compromised, as Bluetooth and WiFi would cause interference in each other.

Another problem faced during the development was the GPS data loss. More specifically, the GPS module periodically sends the data via UART, so if the MCU is busy with other tasks and the UART buffer is not read at time, the data is overwritten and gets corrupted. Thus, in order to avoid such scenario, the software is spitted into two parts, each one executed in one of the two cores available on ESP32. The first core runs the client FSM shown in Fig. 2, and the second core runs the application that reads the UART buffer and processes the GPS data.

The third challenge in the development of the proposed system was the synchronization between the FSMs of the client and severs. Initially, the uplink time counting started as soon as the TCP connection was established. However, in our tests, the server was starting the counting much earlier or later than the client. To deal with this, the counting on the server was set to start only when the server receives the first packet. In addition, another important factor is that due to network delay, some packets reach the server even after the uplink timer has expired. For this reason, it is necessary to wait a few seconds past the defined time, clear the server's RX buffer while there is still data, and only then go to the *Downlink* state. Otherwise, it is likely that the JSON will not be decoded correctly when received.

Finally, the last issue faced during the tests was the possibility of losing connection between the portable hardware and the AP. Therefore, using the ESP32's system API, we set an event handler for the MCU to clear the buffers and attempt to reconnect to the AP indefinitely in case of disconnection. When the reconnection is complete, the client returns to its initial state (*Bluetooth*). On the server side, a timeout is set to identify this and other types of connection loss. When this is reached, the TCP connection is terminated, and the server also returns to its initial state (*Connect*). As result, the system follows the cycle illustrated in Fig. 3 for each measurement.

3 TEST EXECUTION

The executed test consists of fixing the AP on a given location, walking with the prototype in an area to collect data periodically, and sending it to the database through the network. There were six pieces of data collected: latitude, longitude, timestamp, downlink data rate, uplink data rate, and RSSI of the WiFi network. The antennas between AP and portable hardware had line-of-sight to each other during the whole of the time. The AP was positioned at the height of approximately 3.7 meters, and the portable hardware was moved within its range at the height of 2 m along the path.

Both uplink and downlink measurement time were defined as 5 seconds. The payload size chosen for the packets transmitted is 1024 bytes, which is the bigger power of two that does not exceed Ethernet Maximum Segment Size (MSS), which is 1460 bytes [20]. By doing this, we respect Python documentation guidelines [22] and

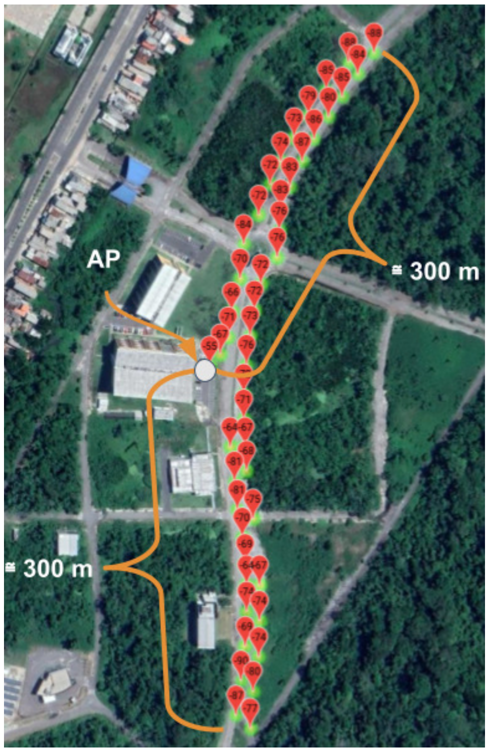
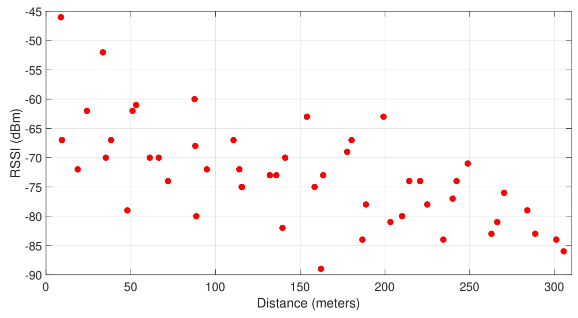
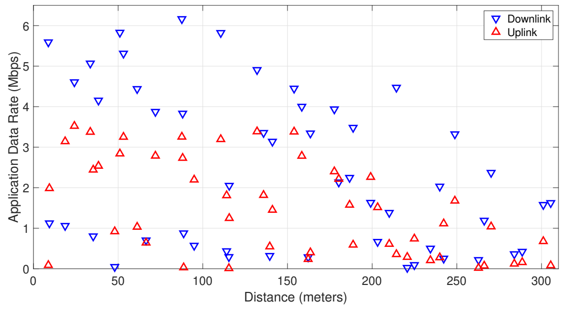


Figure 4: Web application developed to visualize the RSSI on map.



(a)



(b)

Figure 5: RSSI (a) and the application data rate (b) versus the distance between AP and the portable hardware.

avoid fragmentation in layer 3 since the TCP/IP server is connected to the AP via Ethernet. Connection timeout on the server side and GPS timeout on the client side were also defined as 5 seconds.

Furthermore, we conducted the tests on a two-way track. We started from a central spot, where the AP was located and went in one direction until the connection to the AP was lost. Then we went in the opposite direction as well up to the point where there was no more connection. At this point, we returned to the starting position. All along this path, we took measurements every 20 meters, approximately. It is important to mention that there was a light vehicle flow on the road, as well as other 2.4 GHz networks available at the site, which may have impacted some of the measurements taken. However, for our purposes, this is not a problem since the goal was to simulate a scenario close to real use.

To better visualize the acquired data, we have built a web application that consumes the data stored in Firebase Realtime Database and displays it on a map (Fig. 4).

4 RESULTS AND DISCUSSION

A total of fifty samples were collected during the tests. We have exported the data to visualize the RSSI and the application data rate versus the distance between the AP and the ESP32 module, as shown in Fig. 5. The results indicate that the ESP32 module is capable of communicating at distances as high as 300 meters in line-of-sight conditions. In the area where the tests were performed, it was not possible to place the portable hardware at distances higher than 300m with line-of-sight.

In our tests, the minimum and maximum RSSI values obtained were -89 and -46 dBm, respectively. As expected, the RSSI tends to decrease slowly as the hardware moves away from the AP, as shown in Fig. 5a. The outdoor tests performed by [3] resulted in a graphic that decays much faster, although the authors describe it as small-scale fading. Most of their samples whose distance was more than 10 m resulted in an RSSI of less than -80 dBm. In contrast, only a few of our samples resulted in powers lower than this when we were at distances shorter than 250 m.

This difference seems to be related to the transmission power and the number of antennas of the transmitter. We use the off-the-shelf AP described in section 2, which has 4 external antennas, and its maximum transmission power is 18 dBm [19]. These authors, on the other hand, used an ESP32-S2 module as AP. Such device has only one internal antenna and, depending on the modulation and coding scheme (MCS), its typical TX power ranges from 13.5 to 18 dBm [12]. In addition, in our setup, we use external antennas on the AP and in the ESP32 module, while in [3], the authors use PCB antennas on both the AP and the client. Thus, in our tests, it is more likely that the RSSIs are higher in our setup for the same distance.

As shown in Fig. 5b, in terms of application data rate, our setup achieved 6.16 Mbps on the downlink and 3.52 Mbps on the uplink, suitable for medium data rate applications. On the other hand, for a given distance, it is possible to notice some variation in the data rate.

Table 1: Minimum RSSI and PHY data rate for different WiFi configurations

WiFi configuration	Data Rate	RSSI
IEEE 802.11b 1 Mbps	1 Mbps	-98 dBm
IEEE 802.11b 11 Mbps	11 Mbps	-89 dBm
IEEE 802.n HT20 MCS 1	13 Mbps	-79 dBm
IEEE 802.n HT20 MCS 2	19.5 Mbps	-77 dBm
IEEE 802.n HT20 MCS 3	26 Mbps	-74 dBm
IEEE 802.n HT20 MCS 4	39 Mbps	-70 dBm
IEEE 802.n HT20 MCS 5	52 Mbps	-66 dBm
IEEE 802.n HT20 MCS 6	58.5 Mbps	-65 dBm
IEEE 802.n HT20 MCS 7	65 Mbps	-64 dBm

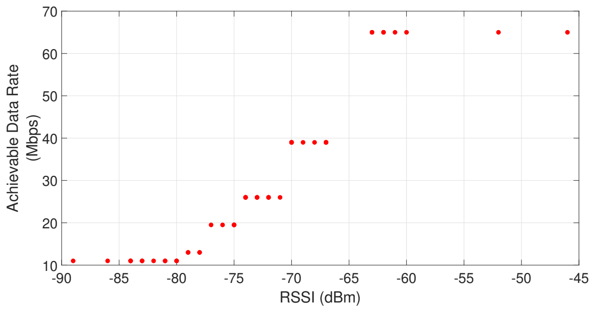
Such variability is expected since the instantaneous data transfer depends on the instantaneous quality of the wireless link, as well as other factors. The effective rate of the application also depends on the link between the AP and the WiFi network, the occupation of the WiFi network itself, the link between the TCP/IP server and the WiFi network, and other network elements between the ESP32 and the TCP/IP server.

In order to access the bounds of the ESP32 on the experiment, we evaluate the achievable wireless data rate and the application data rate versus the captured RSSI and show in Fig. 6. The former was estimated for IEEE 802.11b and IEEE 802.11n configured with 20 MHz and long guard interval for MCSs 1 to 7. These results are based on Table 1, which shows the relationship between the wireless protocol, the corresponding achievable data rate, and the minimum RSSI to use the corresponding configuration. The minimum RSSI for each configuration was captured from [23], and the achievable data rate for each one was captured from the ESP-IDF documentation [10] for IEEE 802.11n. In the case of IEEE 802.11b, the values of RSSI and data rate were captured from the datasheet of the ESP32. Then, the achievable data rate is estimated as the highest data rate for a given RSSI.

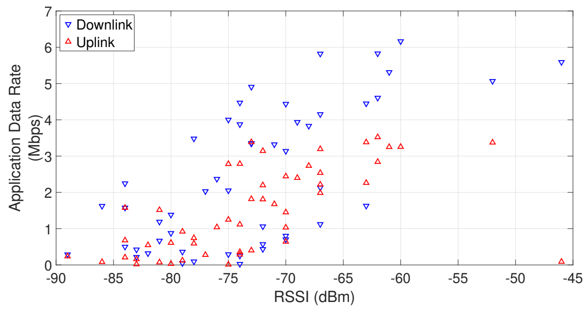
Fig. 6 shows the achievable data rate on the PHY link and the measured application data rate from ESP32 in relation to RSSI values. We can see that the plot with the rates obtained, shown in Fig. 6b, follows the same trend as Fig. 6a with the maximum rate possible, though in smaller order of magnitude. As RSSI increases, the data rate, both for uplink and downlink, tends to increase. The smaller values in Fig. 6b can be explained by the many variables that influence the application rate, such as the TCP retransmissions in case of transfer failure, all the overhead caused by the network and transport layers, and some lack of optimization on the embedded software of the portable hardware.

5 CONCLUSION

In this paper, we give insight into the implementation of a low-cost IoT system for monitoring RSSI and coordinates captured by means of a GPS module, which are transmitted to a database over a WiFi network by means of an ESP32 MCU in an outdoor environment. It was verified that ESP32 modules, in setups similar to the developed in this work, are suitable for medium data rate applications in areas up to 300m in relatively open terrain without loss of connection.



(a)



(b)

Figure 6: Achivable data rate on PHY link (a) and the application data rate (b) versus the RSSI.

One possible way to enrich this experiment is to perform multiple runs, varying some of the parameters and observing the impact on the measurements. For example, different receiver antennas and access points of different configurations could be used, and the payload size could be varied. In addition, the UDP protocol could be used instead of TCP in the communication between client and server in order to observe the data rates without retransmission attempts. Measurement of the rate of lost messages could be included in the results for better monitoring of the packets. Furthermore, more field tests could be carried out in order to increase the amount of data obtained; thereby, other statistical measurement metrics could be used for more reliable results, such as the average RSSI and rates achieved for uplink and downlink for the same distance.

6 ACKNOWLEDGMENT

This work was developed as part of the project GImpSI - Gestão dos Impactos da Salinidade em Isolamentos - with INESC P&D Brasil and Eneva S.A. - under the framework of the R&D Program of the Brazilian Electricity Regulatory Agency, code PD-11278-0001-2021.

REFERENCES

[1] Muhammad Ayaz, Mohammad Ammad-Uddin, Zubair Sharif, Ali Mansour, and El-Hadi M Aggoune. 2019. Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk. *IEEE access* 7 (2019), 129551–129583.
[2] Marek Babiuch, Petr Foltýnek, and Pavel Smutný. 2019. Using the ESP32 microcontroller for data processing. In *2019 20th International Carpathian Control Conference (ICCC)*. IEEE, 1–6.
[3] Valentín Barral, Omar Campos, Tomás Domínguez-Bolaño, Carlos J. Escudero, and José A. García-Naya. 2022. Fine Time Measurement for the Internet of

Things: A Practical Approach Using ESP32. *IEEE Internet of Things Journal* (2022), 1–1. https://doi.org/10.1109/JIOT.2022.3158701

[4] James Brinkhoff and John Hornbuckle. 2017. Characterization of WiFi signal range for agricultural WSNs. In *2017 23rd Asia-Pacific Conference on Communications (APCC)*. IEEE, 1–6.

[5] James Brinkhoff, John Hornbuckle, Wendy Quayle, Carlos Ballester Lurbe, and Tom Dowling. 2017. WiField, an IEEE 802.11-based agricultural sensor data gathering and logging platform. In *2017 Eleventh International Conference on Sensing Technology (ICST)*. IEEE, 1–6.

[6] Rym Chéour, Sabrine Khriji, Mohamed abid, and Olfa Kanoun. 2020. Microcontrollers for IoT: Optimizations, Computing Paradigms, and Future Directions. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 1–7. https://doi.org/10.1109/WF-IoT48130.2020.9221219

[7] Caroline Maul de A. Lima, Eduardo A. da Silva, and Pedro B. Velloso. 2018. Performance Evaluation of 802.11 IoT Devices for Data Collection in the Forest with Drones. In *2018 IEEE Global Communications Conference (GLOBECOM)*. 1–7. https://doi.org/10.1109/GLOCOM.2018.8647220

[8] Giulio Demilia, Antonella Gaspari, and Emanuela Natale. 2018. Measurements for smart manufacturing in an Industry 4.0 scenario a case-study on a mechatronic system. In *2018 Workshop on Metrology for Industry 4.0 and IoT*. IEEE, 1–5.

[9] Espressif. 2022. *ESP-IDF Programming Guide: Coexistence.* Retrieved 2022-07-19 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/coexist.html

[10] Espressif. 2022. *ESP-IDF Programming Guide: Wi-Fi.* Retrieved 2022-06-30 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html

[11] Espressif. 2022. *ESP32 Chip Revision v3.0: user guide.* Retrieved 2022-10-28 from https://www.espressif.com/sites/default/files/documentation/esp32_chip_revision_v3_0_user_guide_en.pdf

[12] Espressif. 2022. *ESP32-S2 Family: datasheet.* Retrieved 2022-08-01 from https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf

[13] Espressif. 2022. *ESP32 Series: datasheet.* Retrieved 2022-10-28 from https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[14] Espressif. 2022. *ESP32-WROOM-32D ESP32-WROOM-32U: datasheet.* Retrieved 2022-10-28 from https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

[15] Espressif. 2022. *ESP32-WROOM-32E ESP32-WROOM-32EU: datasheet.* Retrieved 2022-10-28 from https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

[16] Lucas Martins Figueiredo and Edelberto Franco Silva. 2020. Cognitive-LoRa: adaptation-aware of the physical layer in LoRa-based networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–6. https://doi.org/10.1109/ISCC50000.2020.9219575

[17] Petr Foltỳnek, Marek Babiuch, and Pavel Šuránek. 2019. Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board. *Measurement and Control* 52, 7-8 (2019), 970–984.

[18] Alireza Ghasempour. 2019. Internet of Things in Smart Grid: Architecture, Applications, Services, Key Technologies, and Challenges. *Inventions* 4, 1 (2019). https://doi.org/10.3390/inventions4010022

[19] Intelbras. 2020. *Manual do Usuário: ACtion RF 1200.* Retrieved 2022-08-01 from https://backend.intelbras.com/sites/default/files/2021-08/Manual-do-usuario-Action-RF-1200-02.20.pdf

[20] James F. Kurose and Keith W. Ross. 2017. *Computer Networking: a top-down approach* (7th ed.). Pearson Education, Harlow, England, Chapter Transport Layer, 263.

[21] Margaretha Ohyver, Jurike V Moniaga, Iwa Sungkawa, Bonifasius Edwin Subagyo, and Ian Argus Chandra. 2019. The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test. *Procedia Computer Science* 157 (2019), 396–405.

[22] Python Software Foundation. 2022. *Socket — low-level networking interface.* Retrieved 2022-07-19 from https://docs.python.org/3/library/socket.html#socket.socket.recv

[23] WLAN Professionals. 2022. *Laminated Card 802.11n/HT and 802.11ac/VHT | MCS, SNR and RSSI.* Retrieved 2022-06-30 from https://wlanprofessionals.com/laminated-card-802-11n-ht-and-802-11ac-vht-mcs-snr-and-rssi/

[24] Weixing Xue, Weining Qiu, Xianghong Hua, and Kegen Yu. 2017. Improved Wi-Fi RSSI Measurement for Indoor Localization. *IEEE Sensors Journal* 17, 7 (2017), 2224–2230. https://doi.org/10.1109/JSEN.2017.2660522