Design and Development of IoT-Based Distributed Vibration Recording Instrument

Rendy Ramadhana Wahyudi[1] and Yudi Rosandi[1*]

[1] Department of Geophysics, Universitas Padjadjaran, Bandung, Indonesia
rosandi@geophys.unpad.ac.id

Abstract. The IoT technology has a vast range of applications in nowadays human activities. In this work we report the development of wireless vibration sensors in order to detect the ground vibrations, based on the SM-4 geophone sensors. This study demonstrates the feasibility of deploying a cost-effective geophone sensor system, through testing the wireless functionality. The IoT Ground Vibration Recorder integrates a 10 Hz vertical geophone sensor with an ESP32 board, using the ESP NOW protocol for wireless communication and data transmission. Field tests encompass diverse conditions: open land, buffer zones, and dense vegetation areas. The findings demonstrate the suitability of the ESP NOW protocol for sending seismic data, limited to 250 bytes per transmission. The maximum effective distance reached 95.3 meters, with varying results. Stable connections were observed at shorter distances, whereas longer distances resulted in poorer connections. The seismic sensor systems via ESP32 integration, has successfully transmitted seismic data packages and established effective sensor communication. Furthermore, the study proposes utilizing LoRa modules to extend connection distances and battery life optimization to enhance overall performance. This work suggests that the ESP32 is a promising option for the aforementioned application, offering good seismic data transmission and effective inter-sensor communication.

Keywords: seismic, geophone, iot, esp32

Introduction

Ground waves or seismic waves, as one of the physical phenomena then can be detected using vibration sensors, such as geophone [1]. The data acquired from the sensor can be processed to identify the physical properties of the measured medium. Recordings of ground vibration provide crucial information with relevance to both scientific studies, exemplified by passive seismic research [2], and practical applications in fields like civil and environmental engineering. As noted that the typical applications include site characterization, soil improvement, nondestructive testing of pavements, offshore and near-shore site characterization [3].

Research on a wireless geophone system with integrated data processing capabilities [4]. The geophones were deployed in areas of interest susceptible to landslide risks and were categorized into three frequency bands, i.e. the high, middle, and low frequencies.

F. Khoerunnisa et al. (eds.), *Proceedings of the 9th Mathematics, Science, and Computer Science Education International Seminar (MSCEIS 2023)*, Advances in Social Science, Education and Humanities Research 860,
https://doi.org/10.2991/978-2-38476-283-5_7

Geophone sensors transmitted the collected data to a centralized Data Center for further analysis. Data analysis process encompassed three primary functions: signal denoising, detailed analytical assessment, and issuing alerts to pertinent authorities based on the results of the analysis. Research conducted by [5] focused around the development of an extensive wireless system tailored for the acquisition of seismic data on a large scale. The proposed wireless system presents distinct advantages, notably in terms of efficient data transmission, efficiency was achieved by ensuring consistent energy consumption across each sensor. The work by [6] was concentrated on the creation of a highly efficient Wireless Sensor Network (WSN), achieved through the implementation of a high-gain directional antenna system, enabling the RF front-end to strategically direct the radiated beam towards the nearest gateway. As a result, energy expenditure was minimized, leading to an augmentation in achieved transmission ranges. According to [7], powered by a 2000 mAh battery the ESP32 can operate for up to a month, using the deep sleep functionality between data reception and transmission periods.

ESP32 microcontrollers (μC) exhibit promising potential when employed in conjunction with the ESP NOW protocol [8]. This protocol offers a versatile means to configure wireless communication between ESP32 devices. The capability proves particularly advantageous in the context of constructing Wireless Sensor Network (WSN) designed for the monitoring of seismic events. Several attempts to produce a WSN network utilized Arduino and Raspberry Pi as microcontrollers platforms. However, our research introduces the ESP32 as a viable alternative. The choice was taken due to the support of deep sleep mode, the embedded internal storage, and the robust community ecosystem. Another strong factor of choice was the cost-effectiveness and the availability within the research region. Researchers have consistently explored energy-efficient algorithms to solve many problems and applications, such as seismic surveys, and disaster mitigation. The necessity of such an algorithm is crucial in a vast coverage and extensive geographical regions. Up to now, the application of ESP32 as a microcontroller in seismic WSN systems is relatively unexplored.

Table 1. A comparison of several key aspects that underpin the development of wireless systems.

Aspect	Proposed IoT-Based Tools	Existing Wired Tools
Communication	Utilize ESP NOW protocol for wireless communication, offering good seismic data transmission and effective inter-sensor communication.	May rely on traditional wired connections or less efficient wireless protocols
Power Efficiency	Incorporates deep sleep mode and energy efficient algorithm to extend battery life, crucial for remote sensing apllications.	Typically, due to a lesser focus on power efficiency, the device must be used close to a power source.
Cost-Effectiveness	Cost-effective due to the use of ESP32 microcontroller and the potential for using LoRa rnodules for extended connection distances.	Existing systems may be more expensive due to the use of more

Aspect	Proposed IoT-Based Tools	Existing Wired Tools
Range	Achieved a maximum effective distance of 220 meters theoretically in field tests, with the possibility of extending this range using LoRa modules	complex or proprietary technology Wired systems are limited by cable length, and wireless systems may not achieve the same range without additional equipment.

Table 1 provides a comparative perspective on the advantages and disadvantages of the novel tool versus the predecessor tool across several key aspects. The ideas on building a wireless seismic survey apparatus help to simplify the field measurement process that relies on cable connected instruments. This type of instrument could be impractical in some cases. The use of many channels makes it difficult for seismic instruments that use multi-conductor cables, especially with the expansion of CMP and Land Streamer surveys [9]. To overcome this problem, we propose using the ESP32 microcontroller wireless communication of measurement data. In this study we also do experiments on power optimization by means of deep sleep techniques, as well as checking the range and quality of data transmission.

2 Methodology

2.1 Instrument Workflow

The wireless seismic sensor (WSN) system consists of two parts, which are the node sensor and the main unit. The node sensor comprises the essential element, i.e. the vertical geophone sensor of type SM-4, alongside the microcontroller ESP32, a charging module, and a battery. The main unit is the interface device between the system and an acquisition computer. This unit is equipped with a program to manage and give instruction to sensor nodes. User interaction with the unit can be done through a text terminal interface.

As illustrated in Figure 1, the sensor nodes establish communication with the main unit via ESP NOW protocol to transmit raw data and the device status. To perform the available operations the sensor unit receives commands sent from the main unit connected to a controlling computer through a USB cable. The used communication protocol was the standard UART serial interface [10]. Any information sent by the sensor node is promptly displayed on the terminal user interface (TUI). Using this interface, users can issue commands to a specific sensor node or to all nodes via the broadcast mechanism. Incoming data is stored in JSON format within the internal memory. The data is erased upon subsequent transmission cycles, JSON data format was chosen in order ease further analysis on a computer system.

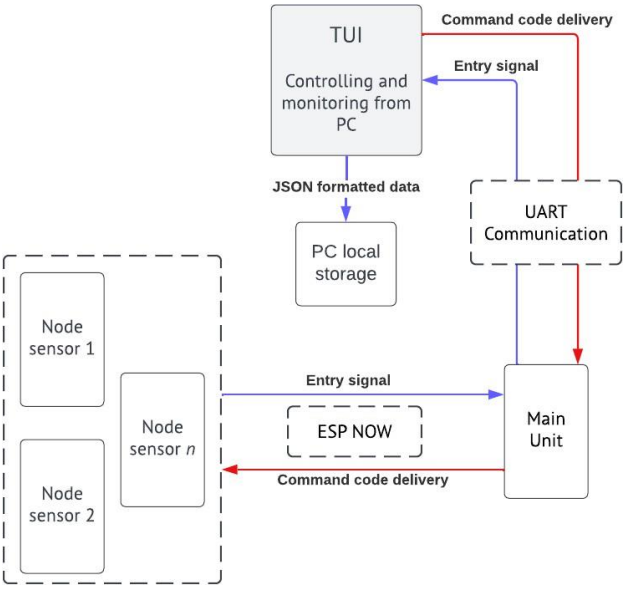

Fig. 1. The flowchart illustrates the configuration of the WSN seismic sensor. Control and monitoring are facilitated through the main unit.

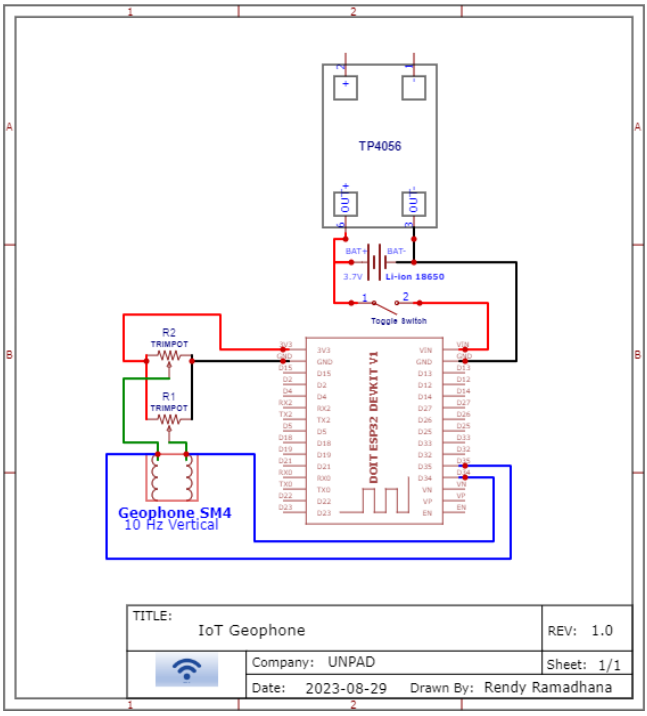

Fig. 2. Electronic schematic composed of ESP32, SM4, TP4056, trimpot, Li-ion battery 18650.

2.2 Hardware

Schematic diagram of the sensor unit is shown in Figure 2, the system requires six pinouts of the ESP32. Geophone SM4 is connected to two adjustable trimpots on GPIO34 and GPIO35. This configuration was used to obtain the differential measurement of the attached SM4 sensor. To power the ESP32 board a battery was attached to the 3.3V, development board is equipped by a voltage regulator, however the range must not be too high to avoid overheating and board damage. In our case the voltage regulator can accept 3.7 V from the battery unit. A TP4056 charger module was attached to charge the node sensor using a standard USB cable. Li-ion 18650 3.7V battery is connected to a toggle switch which is used to connect and short the V_{in} and ground terminals. ESP32 is equipped with a 2.4 GHz Wi-Fi module that can be used for wireless connection for the WSN applications. This peripheral is very useful in the design of wireless seismic sensing. The module is accompanied by an integrated antenna with a gain of 2 dBi, having impedance of approximately 50 ohms. This complies with the requirements of most RF components.

The SM4 geophone element is the main component in this project. This sensor has a natural frequency of 10 Hz and has been used in many seismic related projects in the past, such as an explosive and vibroseis survey method in Antarctica, in 2010 to 2011 [11]. The SM4 geophone has a damping coefficient of 0.271, which is convenient to be used in a near offset seismic survey.

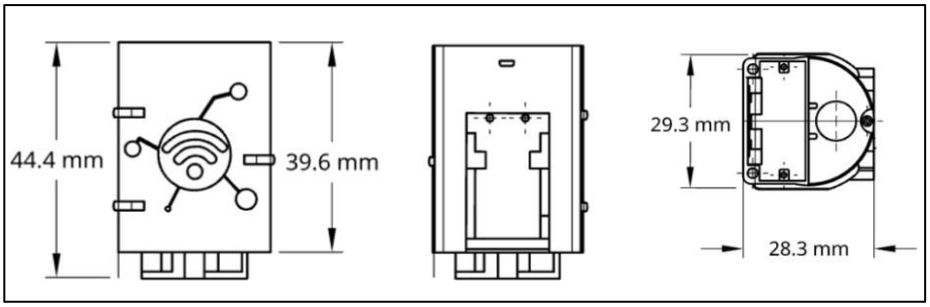

Fig. 3. The sketch of the instrument housing, presented from various angles and incorporates dimensions, focusing on its external appearance.

In this work we designed a housing system for the node sensors, as shown in Figure 3. First layer serves as the housing for the SM-4 element, placed close to the ground level for optimal readings. To fix the system in place, an additional spike needle can be attached. The second layer is designated to place the ESP32 development board, ensuring short wiring access for both to the sensor and to the charger module, which are placed on the top. The battery is positioned beside it. Topmost layer accommodates the charging module and toggle switch. The housing of the system was 3D printed using Poly Lactic Acid (PLA) material. The 3D design was performed using cloud software, Onshape, which was notified by [12] for its user-friendly yet powerful features. CURA was chosen as the slicing software as recommended by [13]. The PLA material

was used as the test material. According to [14] the material has a decent stability and replicability, which is sufficient for prototyping purposes.

2.3 Software

The software was written in C++ and Python. For the microcontroller programming, Arduino IDE interface was employed. Terminal User Interface (TUI) was constructed using the Python curses library. The sensor node program integrates algorithms of data acquisition and transmission. For temporary data storage in the microcontroller, the LittleFS library was used. This library offers an advanced and efficient routine for storing data on the ESP32 flash memory [15]. In this project the file system is used to store the acquired seismic data from the SM4 sensor before transmission to the main unit.

The main unit program has a function to issue commands which will be executed by the designated node sensor. The unit also acts as the receiver of messages containing information and seismic data sent by the sensor units. The TUI comprises two primary windows, i.e. the main menu and the serial monitor window. Serial monitor window is partitioned into two sections through a multi-threaded approach, hence the communication routines do not block each other. This approach gives several benefits, i.e. the enhanced execution time, the improvement of responsiveness, and a more efficient resource utilization [16]. The first part of the window functions as the command column for operator inputs, while the second part as a monitoring window to display messages received from any sensor node in real-time.

As mentioned previously, we apply the ESP NOW protocol on our project. This protocol offers a robust and efficient wireless communication protocol, particularly in scenarios where seamless device pairing is crucial. In contrast to a conventional WiFi protocol, ESP-NOW facilitates data transfer without the need for handshaking processes. It provides high noise tolerance and a relatively high data transfer rate of 300 Mbit/s [17]. The protocol can transmit up to 250 bytes of data in a single transfer. With the ability to accommodate up to 20 device connections, ESP NOW is the correct choice for our project. The communication design is adaptable and capable of functioning unilaterally or bilaterally [18].

The deep sleep mode can be implemented in the ESP32 by deactivating both Wi-Fi and ESP NOW connections. This operation is most efficient when it runs on the ULP co-processor on the ESP32 board [19]. This functionality can substantially reduce the power consumption. An energy efficient device is crucial in our application, when the measurement process may extend beyond the lifetime of a normal battery powered instrument. The energy efficient instrument is very demanding in particular for a remote sensing application.

2.4 The operating procedure

Table 2 contains a series of commands that can be executed during the data acquisition. This set of commands consists of the acquisition initiation, the data transmission, and sending a unit into deep sleep mode. The data acquisition is done by analog values reading from two ADC channels, namely the GPIO 34 and GPIO 35, the voltage difference

of these ports was taken as the measurement data. Using this method we mimic the differential instrumentation measurement with two single ended ADCs. During this process, the LED on the ESP32 board illuminates to inform that measurement is running. Simultaneously, the terminal displays the ongoing acquisition status. Upon completion of the measurement, the LED deactivates, and a confirmation message is given, indicating that the acquired data has been successfully saved on the ESP32 board.

Table 2. Command list along with its corresponding function and description to do acquisition procedure.

Command	Function	Description
beast(sec μsec)go	Blinking LED: Acquisition Start message (LED on); Data saved message (LED on).	Sensor node initiates the data acquisition process, capturing data of a specific length, and then stores it in the internal memory of the ESP32 board Confirmation messages to the main unit will be sent at the start and completion of the acquisition Sec and μsec is a number
sendto(node list)fetch	Data message is being sent Data saved on local PC	Data is transmitted as a header and divided into smaller packets within the body to ensure the complete transmission of the data Node list is the sequence number for a particular node
beast dsleep	Deep sleep start and stop message	Timer starts for a specific duration, with Wi-Fi and ESP NOW connectivity disabled When the maximum duration is reached deep sleep is halted
list	Node sensor list message	Checking the stored list of sensor nodes and displaying it in the terminal with columns for sequence, ID. and MAC Address
del (node list)	Sensor node id message deleted	Removing a specific sensor node from the stored list Node list is the sequence number for a particular node
bcast checking	Message status of the sensor node connection with the main unit	Sending a dummy message and waiting for a 5-second response from the sensor node If the sensor node responds with a message, it is considered connected
bcast ping	Blinking LED	Blinking the LED three times
sendto(node list)info	Send node sensor id message	Checking the sensor node's ID and then sending it to the main unit Node list is the sequence number for a particular node

Some commands are used to perform the acquisition procedure. These commands consist of a command to fetch the information on the number of nodes connected to the main unit. The number is stored in the memory of the main unit to be used to do further communication identity. This ID number is used for processes such as node removal, node availability checking.

The measurement data sent by a sensor node was divided into two structures: the header and the main body. The header encompasses essential information such as the sensor node's identity, timestamp, and data length. The main body comprises the actual

sensor data. Due to data length limitation on the ESP NOW protocol, i.e. 250 kilobytes (kb) per transfer, a chunking method is required. This method involves splitting the file size on the client into several blocks and sending it one part at a time. The data chunks are recombined on the main unit. The similar method was used by [20]. The chunk mechanism must be accompanied by the data integrity check to avoid transmission error.

The deep sleep mode is done by disabling certain features such as WiFi and ESP NOW. The deactivation of WiFi includes turning off ESP NOW functionality and removal of peer nodes from the connection list. The duration of the deep sleep mode defined by the timer configuration. When the timer signal is accepted, the mode is deactivated, and the WiFi and ESP NOW are reenabled. Upon restarting the device, the node sensor promptly provides confirmation of its operational status on the TUI.

3 Result and Discussion

The aim of this work is to obtain a cost-effective and efficient wireless seismic measurement system. Figure 4 shows the housing of the system. Physically, all components within the node sensor casing are securely arranged, with each 3D-printed part fitting precisely as depicted in Figure 4, outcome exceeded our expectations, as the casing boasts a robustness reminiscent of molded structures. Entire 3D printing process for these components took approximately 1 day and 5 hours, employing a printing resolution of 0.16mm.

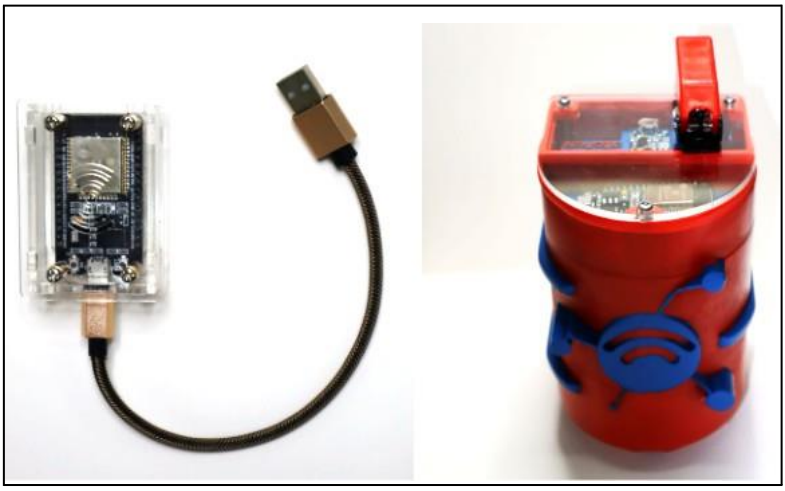

Fig 4. It illustrates the main unit (left) linked to the PC's USB port, functioning alongside the TUI to establish communication with the node sensor. The completed node sensor product (right) showcases the integration of all electrical components.

Sensor readings have yielded the desired results. To assess the response of the WSN seismic sensor, we conducted three measurement attempts. The instrument's expected

behavior includes transmitting an acquisition status confirmation message, activating the LED during acquisition, deactivating the LED, storing sensor data, and confirming data storage. Subsequently, data can be viewed on the PC using the "fetch" command.

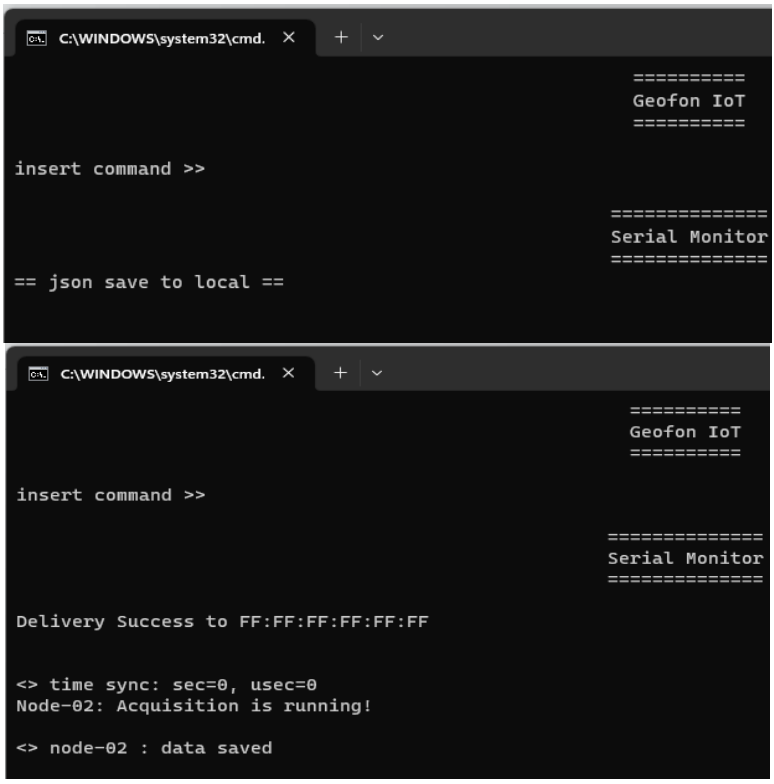

Fig 5. Following the initiation of the acquisition process, the serial monitor displays an initial message directed to the node sensor (top). Terminal User Interface (TUI) provides a confirmation if the data has been successfully stored locally (bottom).

As a default behavior, acquisition command is broadcasted, as indicated by the "FF:FF:.." message displayed on the monitor, as shown in Figure 5. This broadcasting mechanism signifies that the signal emitted from the main unit is disseminated to all deployed node sensors in proximity. In response, the node sensor transmits a confirmation message indicating that the acquisition is in progress. Upon completion of the acquisition, the node sensor sends another message confirming the successful storage of data in JSON format onto the ESP32 board. In the specific context of this test, a sole node sensor was employed. LittleFS extends its capabilities beyond mere storage, encompassing more intricate tasks such as writing, reading, editing, and data removal. This versatility contributes to prolonging the life of the flash memory. In our experimentation, we observed that LittleFS exhibits an adept algorithm for data management, efficiently executing writing, reading, and data removal tasks. This efficiency

translates to a reduction in the amount of memory consumed by program code. Moreover, the adoption of LittleFS opens doors for the implementation of advanced algorithms tailored specifically for storing sensor data.

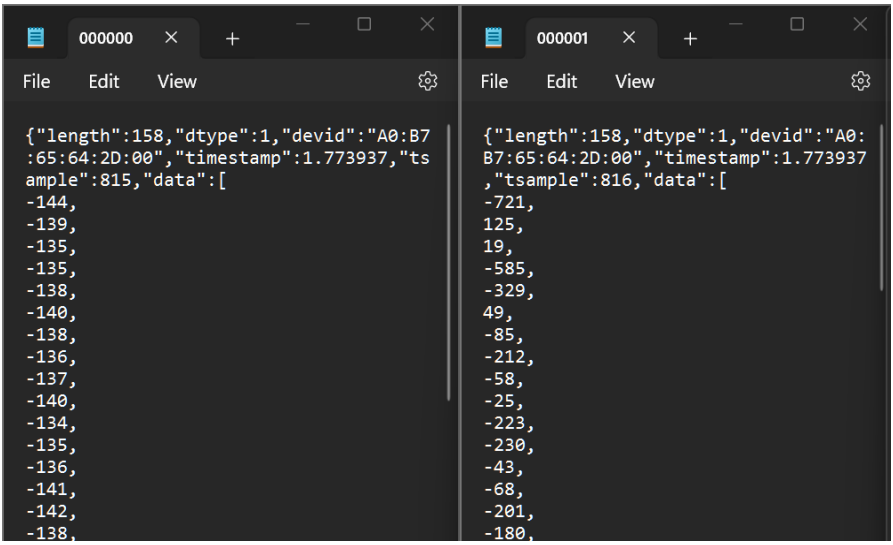

Fig 6. Cropped snapshots of the initial attempt (000000) and the second (00001) JSON data include sensor values.

As depicted in Figure 6, The data is segmented into a header and body. The header is positioned at the top of the dataset, comprising length, data type, device ID, timestamp, and time sample. Meanwhile, the body encompasses the remaining content, housing the actual sensor data. Initial JSON data is labeled as "000000," and the subsequent data is designated as "000001." These datasets are both stored within the same partition as the TUI program. Upon the completion of the acquisition process, we can conserve node sensor power by transitioning into deep sleep mode. In this mode, the node sensor deactivates WiFi and ESP NOW functionalities and commences a countdown. This countdown continues until the set time is reached, at which point the ESP32 board can become operational once again. To initiate the deep sleep mode, the operator inputs the deep sleep command. The Terminal User Interface (TUI) validates the successful delivery of the command, after which the node sensor responds by specifying the nodes that have entered deep sleep mode. Upon the completion of the designated deep sleep duration, the node sensor promptly sends a confirmation message signifying the deactivation of deep sleep mode, request to pairing (Figure 7).

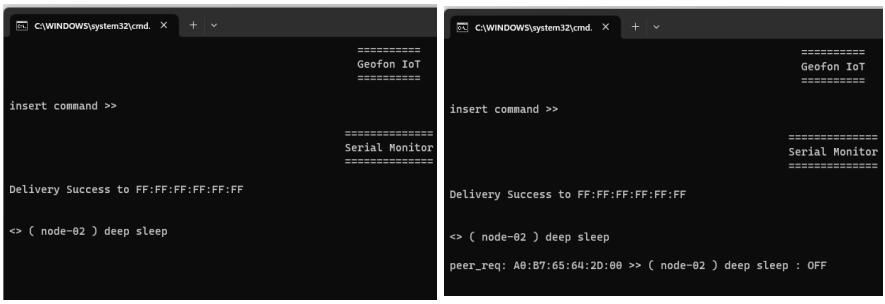

Fig 7. In the initial attempt (left) to activate the node sensor's deep sleep mode, the process begins with the operator issuing a command for a broadcast. After deep sleep mode time out, the corresponding node sensor (peer) initiates a request for re-pairing (right).

Communication was tested in three distinct areas, each representing a different characteristic environment. This approach aimed to simulate various situations encompassing a range of obstacles that could potentially impede the propagation of electromagnetic signals. Testing encompassed open field conditions, a densely vegetated site, and a built-up urban space. Results revealed that the WSN ESP32 seismic sensor managed to achieve a maximum communication distance of 95.3 meters. Within this range, commands were successfully transmitted to the node sensor, which responded by illuminating the LED. However, the main unit encountered difficulties in receiving the expected callback from the node sensor (Figure 8). Notably, at this specific distance, ESP NOW exhibited limitations in its functionality, as it encountered difficulties transmitting the desired information effectively. Despite this setback, the node sensor continued to respond by illuminating the LED. Our experimentation demonstrated that both the orientation and the direction of radiation of the ESP32 board significantly impacted the connection quality. These findings highlight the importance of considering the board's physical positioning and the direction of its radiation for optimal communication performance.

This study was limited to employing a single node sensor and configuring the board to perform fundamental functions, including basic communication, sensor data reading, storage, and power-saving mechanisms. Each capability of this instrument has the potential for further development, allowing for enhanced functionality. Communication range can be expanded by integrating advanced external antennas onto the ESP32 board. Additionally, the ESP32 platform offers a variety of board types and models, providing options for selecting boards that are better suited for optimizing power efficiency and overall performance.

Fig 8. The connectivity range test revealed that the maximum connection distance achievable between the main unit and the node sensor is 95.5 meters.

4 Conclusion

This study has demonstrated the effectiveness of the ESP32 board in realizing a seismic acquisition system. The ESP32 board effectively interfaces with the SM4 seismic sensor, enabling seamless data reading and direct storage within the ESP32 memory. Implementation of a Wireless Sensor Network (WSN) was achieved with the support of the ESP NOW protocol, facilitating bidirectional transmission and reception of information and data. The system was able to establish connections up to a maximum distance of 95.3 meters. Notably, the ESP32 module can be configured to enter the deep sleep mode to reduce power consumption. This study has confirmed the feasibility of applying ESP32 microcontroller units for the implementation of a WSN seismic sensor. As a part of future enhancements, the communication range could be expanded by incorporating better external antennas. Sensor reading precision can be improved through the utilization of higher-resolution ADC and can be amplified and filtered for increased accuracy. Moreover, power efficiency can be further optimized by selecting a more energy efficient ESP32 board type, such as one with a low dropout voltage regulator circuitry.

References

1. Ziemann, V. A Hands-On course in sensors using the Arduino and Raspberry Pi. CRC Press eBooks. (2018)
2. Karplus, M. S., and Schmandt, B. Seismological Research Letters, 89(5), 1597–1600 (2018)

3. Foti, S., Lai, C. G., Rix, G. J., and Strobbia, C. Surface Wave Methods for Near-Surface Site Characterization. CRC Press eBooks. (2014)
4. Deekshit, V. N., Ramesh, M., Indukala, P. K., and Nair, G. J. In: International Conference on Communication and Signal Processing (ICCSP), Adhiparasakthi Engineering College, Melmaruvathur, Tamilnadu, India. IEEE Explorer (2016)
5. Reddy, V. A., Stuber, G., and Al-Dharrab, S. In: IEEE 88th Vehicular Technology Conference (VTC-Fall), IEEE Explorer. (2018)
6. Attia, H., Gaya, S., Alamoudi, A. O., Alshehri, F., Al-Suhaimi, A., Alsulaim, N., . . . Al-Dirini, F. IEEE Access, **8**, 81116–81128. (2020)
7. Linggarjati, J. In: IOP Conference Series, **998**(1), 012042. (2022)
8. Babiuch, M., Foltynek, P., and Smutny, P. In: 20th International Carpathian Control Conference (ICCC), IEEE Explorer. (2019)
9. Crice, D., Flood, P., and Walthinsen, E. In: Symposium on the Application of Geophysics to Engineering and Environmental Problems 2015, pp. 465-468, Society of Exploration Geophysicists and Environment and Engineering Geophysical Society. (2015)
10. Szabo, R., and Gontean, A. In 2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 175-179. IEEE, (2015)
11. Hofstede, C., Eisen, O., Diez, A., Jansen, D., Kristoffersen, Y., Lambrecht, A., and Mayer, C. Annals of Glaciology, **54**(64), 189–200. (2013)
12. Le, Ngoc. Degree Thesis. Retrieved from https://www.theseus.fi/handle/10024/157180 (2018)
13. Dhore, G., Jagtap, R., Bhakad, S., Yadav, P., Sutar, A. B., and Sawrate, S. International Journal of Advance Scientific Research and Engineering Trends, **5**(12), (2021)
14. Brischetto, S., and Torre, R. Journal of Composites Science, **4**(3), 140 (2020)
15. Cameron, N. R. Electronics Projects with the ESP8266 and ESP32. Apress eBooks (2021)
16. Nguyen, Q. Mastering concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming. Packt Publishing Ltd. (2018)
17. Yukhimets, D., Sych, A., and Sakhnenko, A. In: 2020 International Russian Automation Conference (RusAutoCon), IEEE Explorer, Rusia (2020)
18. Pasic, R., Kuzmanov, I., and Atanasovski, K. Izzivi Prihodnosti, **6**(1). (2021)
19. Gatial, E., Balogh, Z., and Hluchy, L. In: IEEE 24th International Conference on Intelligent Engineering Systems (INES), IEEE Explorer (2020)
20. Mallafi, H. Indonesian Journal on Computing, **1**(1). (2016)