Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 46 No. 5
May 2025

Implementation of self-healing network: An improved real-time environmental data reading and monitoring using eps32 and bme28 sensor empowered by smart contract

Suale Yakubu[1], Agnes Mindila[2], Peter Kihato[3]

[1]Department of Electrical Engineering, Pan Africa University Institute for Basic Sciences, Technology and Innovation, Kenya

[2]Department of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Kenya

[3]Department of Electrical and Electronic Engineering, Jomo Kenyatta University of Agriculture and Technology, Kenya

Abstract

This paper presents a novel approach on the implementation of self-healing network solution leveraging the combined capabilities of ESP32 microcontroller and BME280 sensor. The proposed solution addresses the key limitations of conventional monitoring infrastructure including limited autonomy and fault tolerance. The ESP32 was utilized to achieve the self-healing capabilities due to its embedded wireless fidelity (Wi-Fi) features. A smart contract was deployed to ensure data integrity and automate decision making process which is triggered locally by the ESP32 via a web3 interface to enable unified interactions between the decentralized backend and the ESP32 nodes. This was evaluated with a prototype using ESP32 and BME280 sensors to validate the feasibility of IoT-enabled self-healing network and to demonstrate its effectiveness in maintaining secure data transmission among peers efficiently. The results shows that the system demonstrates robust fault tolerance through its self-healing capabilities, recovering from node or network failure within 0.05 seconds. It archived an average interval of 15s relay of data collected showcasing its potential in timely submission of data, suitable for time sensitive networks application domain. The study provides a secure and reliable solution for modern communication challenges making it suitable for resource-constrained networks with dynamic topologies. These findings have practical implications for industries requiring robust and reliable network infrastructure, offering a transformative approach to decentralized communication systems.

Keywords: IoT, Smart contract, APIs, Self-healing Networks

1. Introduction

The global increase of climate change has raised the concerns of real-time monitoring of environmental parameters for governments, researchers, and industrial players. Accurate data on temperature, humidity or air pressure are important not only to prevent natural disasters, but also to optimize the management of natural resources and strengthen sustainable development policies[1], [2]. The conventional monitoring systems have their associated constraints such as lack of autonomy recovery and limited fault tolerance, data centralization, and high risks of network disruptions[3]. Based on these challenges, the development of self-healing networks offers a promising alternative. These systems are characterized by their ability to automatically detect internal failures, dynamically reorganize, and maintain continuity of services without human intervention [4]. The implementation of such networks is based on distributed, resilient and adaptive architectures, integrating both high- robust communication protocols and performance microcontrollers[5]. This paper presents an innovative approach combining the ESP32 microcontrollers, renowned for its local processing capabilities, built-in Wi-Fi/Bluetooth connectivity, and low power consumption, with the BME280 sensor, which enables multiple reliable measurement of environmental conditions[6]. This technology duo is at the heart of a real-time monitoring system, designed to operate in a variety of environments, including those with intermittent or harsh connectivity[7], [8]. But beyond the hardware aspect, this work draws its strength from the integration of smart contracts, deployed on a lightweight blockchain. These contracts not only automate the management and verification of the data collected, but also guarantee their integrity, immutability and traceability without requiring a trusted third party[9]. This means that every sensor reading is immediately and securely recorded, and can be audited at any time, increasing the transparency and reliability of the system[10], [11]. By combining IoT technologies, network resilience and blockchain, this study explores an innovative path towards the creation of autonomous, intelligent and reliable systems for reading and monitoring environmental data in real time. It opens the door to many applications in fields

such as precision agriculture, climate monitoring, smart cities and environmental risk management.

2. Objectives

This paper aims to design, implement, and evaluate self-healing network using ESP32 and BME280 empowered by smart contracts. The study addresses the limitation of conventional network infrastructure issues in terms of autonomous recovery, data integrity, and fault tolerance. This seeks to provide a secure, robust and real-time platform for environmental data collection and monitoring suitable for dynamic and resource-constrained networks.

3. Methods

The methodology adopted for this paper revolves around the development of a resilient, real-time environmental monitoring system that combines edge computing, wireless communication, and blockchain-based data verification as illustrated in Figure 1. The architecture is composed of four main functional components: a BME280 sensor for data collection, an ESP32 microcontroller for local processing, a Wi-Fi interface for data transmission, and a Raspberry Pi 4 device that handles data verification and storage through smart contracts on a blockchain.

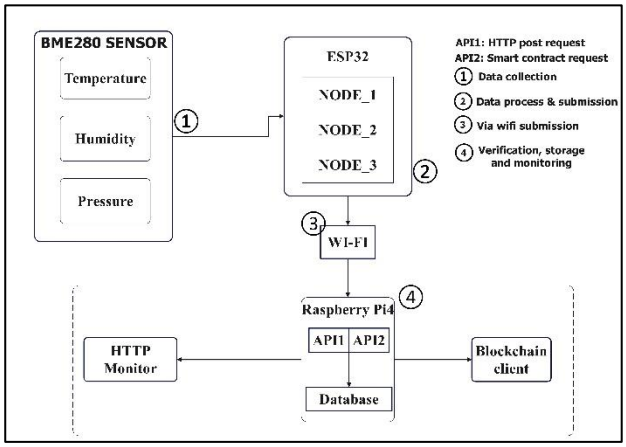

Figure 1: Block diagram of the implementation

The process begins at the sensing level with the BME280 sensor, which captures real-time environmental parameters such as temperature, humidity, and atmospheric pressure. This sensor is selected for its high accuracy, low power consumption, and compact design, making it ideal for embedded applications. The sensor transmits the collected data directly to the ESP32, which serves as the core processing unit at the edge of the network.

Within the ESP32, the incoming data is handled through a modular node-based structure designated as NODE_1, NODE_2, and NODE_3. Each node is responsible for a distinct task such as data filtering, formatting, and temporary storage. This design not only streamlines processing but also introduces fault tolerance into the system. If one node fails or becomes unresponsive, the others can dynamically reassign the workload, ensuring that data continues to flow without interruption a key aspect of the self-healing network principle.

Once the data has been processed and structured, it is transmitted wirelessly via Wi-Fi to a Raspberry Pi 4, which operates as the central data gateway. At this stage, the Raspberry Pi runs two application programming interfaces: API1 and API2. API1 is responsible for handling HTTP POST requests from the ESP32. It verifies the data format and stores it in a local database. API2, on the other hand, handles the smart contract logic. It takes the verified data and submits it to a lightweight blockchain client running on the Raspberry Pi. This interaction with the blockchain ensures that every data point is securely recorded with a timestamp, providing tamper-proof, verifiable storage without the need for a centralized authority.

The integration of smart contracts enhances the transparency and reliability of the system, allowing for real-time monitoring, immutable logging, and historical audits of environmental data. Furthermore, the use of local edge computing (ESP32) combined with decentralized storage (via blockchain) reduces dependency on cloud infrastructure and mitigates risks associated with network downtime or data breaches.

A) Implementation

BME280 sensor was utilized to measure temperature, humidity and barometric pressure which is submitted by the ESP32 to the raspberry pi APIs for validation, verification and storage as shown in Figure 1. The ESP32 establishes a mesh connection with its neighbours via wireless connectivity. It is used to process and submit the collected data from the BME280 sensor. This was configured to retain the APIs of the HTTP and the smart contract post request. When submitted, the raspberry pi hosting the blockchain smart contract verifies and log transactions in blocks whiles the HTTP verifies and store in MYSQL database as shown in Figure 1 and Figure 3. Wi-Fi was chosen as the based communication mode to facilitate automatic recovery from failures [12], [13] such as power failures. Since wireless communication devices can reconfigure themselves back to the network, the optimal operation will be enhanced. When any device fails, due to power or any reason, it's able to reconnect when available and

continuous operations[14], [15]. This minimizes human error and interferences. In the other hand, the raspberry pi hosts the smart contract, HTTP post APIs which receives and verifies the data submitted by ESP32 which is subsequently stored in the database and append a new block in addition to the existing blocks in the blockchain.

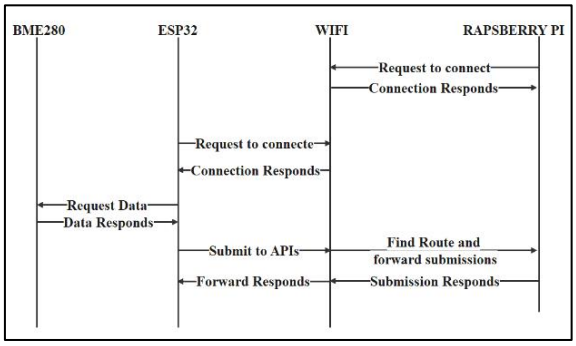

Figure 2: Structure of Communication between the devices

All the responds from the APIs can be monitored over the serial monitor on the Arduino integrated development environment (IDE). The implementation was done by using three ESP32 as communication nodes with individual BME280 sensor for data collection. The data collection was done in every 30s to allow the boards to switched into different mode to minimize the power consumption and heat. The structure of the setup can be seen in Figure 3.

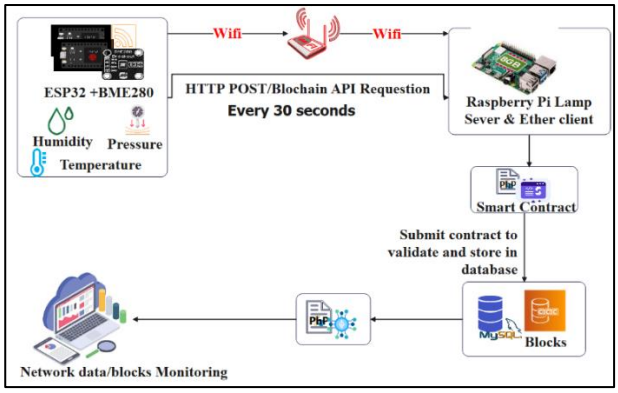

Figure 3: Structure of Setup

Figure 3 shows the setup of the implementation and demonstrate the logics behind it. A smart contract and HTTP post request was created to handle data submission form the ESP32 collected by the BME280 sensors to be verified and stored. These was written in solidity language using remix IDE, monitored over ganache client on the local host machine. This facilitated the verification and monitoring of the nature of data submission and blocks increments when success. As the name implies, smart contract is a self-executable file that facilitate an agreement between participant upon meet certain predefined conditions[16], [17]. With the help of web 3 on node java script (js)[18], the initiation of the smart contract by the ESP32 was made possible and easy to integrating two APIs (http & smart contract APIs) for data (transaction) submission.

4. Results

This paper employs the implementation of innovative approach to enhance self-healing and self-organizing mesh networks. The ESP32 in combination with BME280 was utilized as nodes to initiate transaction in a blockchain by submitting the collected data by the sensor. Raspberry pi4 was used as the local server that host both AIPs of the HTTP and blockchain post request. Table 1 shows the nature of data collection at an interval of 15 seconds.

Table 1: Sample data collected from various stations

Sensor	Location	Temperature	Humidity	Pressure
Node 3	Room3	25.97	44.08	849.39
Node 2	Room2	25.45	47.32	849.95
Node 1	Room1	26.34	43.67	849.43
Node 3	Room3	25.84	47.06	849.99
Node 2	Room2	26.36	45.23	849.43
Node 1	Room1	25.87	47.36	850.05
Node 3	Room3	26.32	44.69	849.41
Node 2	Room2	25.85	47.49	850.00
Node 1	Room1	26.3	44.03	849.45
Node 3	Room3	25.89	46.67	849.99
Node 2	Room2	26.26	43.91	849.45
Node 1	Room1	25.91	46.13	850.01
Node 3	Room3	26.26	45.42	849.42
Node 2	Room2	25.92	47.23	849.99
Node 1	Room1	26.25	45.94	849.44
Node 3	Room3	25.92	48.11	850.01

The results in Table 1 shows that the minimum, average, and maximum temperature red was 25.45^{oC}, 26.04^{oC}, and 26.36^{oC} respectively whiles humidity was 43.67%, 45.90%, and 48.11% similarly for pressure, it

was 849.39, 849.71 and 850.05 Pa. Figure 4-*6* show the graphical representation of the temperature, humidity and pressure readings.

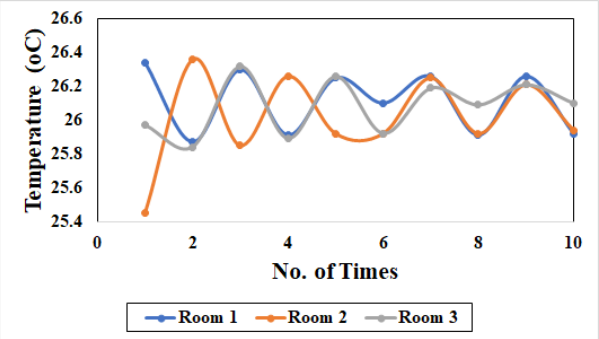

Figure 4: Temperature Readings

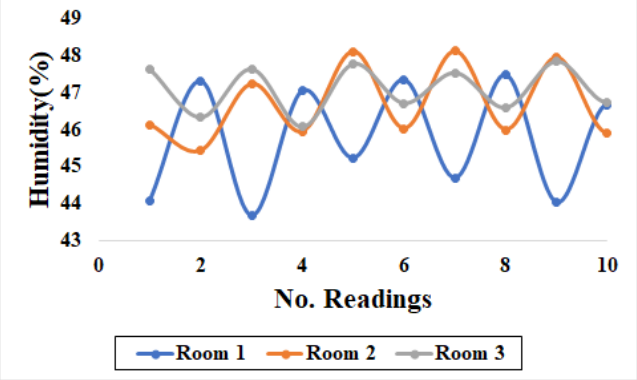

Figure 5: Humidity Readings

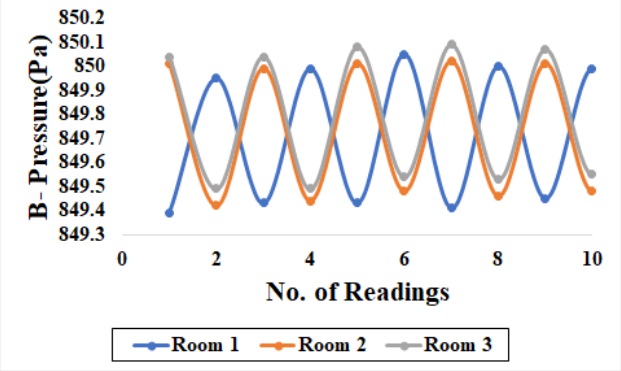

Figure 6: Barometric Pressure Readings

These show variation pattern of the reading of temperature, humidity and pressure by the various bme280 sensor against the number of times it was able to read data in the various rooms.

The results of the robust monitoring system highlight the system's ability to reliably track network traffic in real-time. Leveraging ESP32 microcontrollers, BME280 sensors, and the mesh network, the system effectively addresses the challenges of data collection, secure communication, and accurate visualization of data as shown in Figure 7.

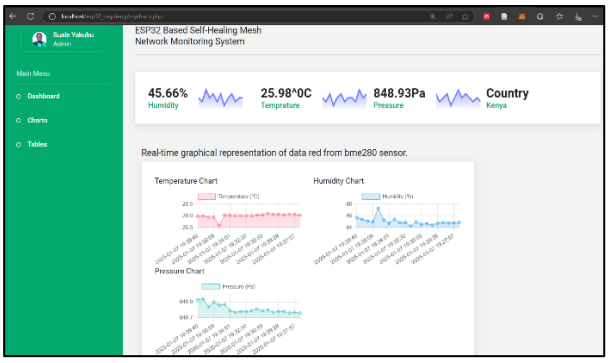

Figure 7: Network Monitor Results

The robust monitoring system demonstrates its effectiveness in tracking network data and secure data management within the network. The responsive dashboard effectively visualizes real-time data, allowing administrators to monitor conditions and promptly receive alerts for threshold breaches. The system's rapid responsiveness to environmental changes and low power consumption enhanced its practicality for continuous operation.

5. Discussion

In comparison to centralized networks for environmental monitoring, the deployment of a self-healing network demonstrated notable gains in network recovery, fault tolerance, and autonomy. The BME280 sensors and ESP32 microcontrollers facilitate effective local data collection while preserving the networks' resilience and fault tolerance. Real-time monitoring in unstable environments requires a robust self-healing capability, which is highlighted by the system's ability to freely recover from node catastrophes in less than 0.05 seconds. One of the main issues in dynamic and fault-prone environments was effectively resolved by the self-healing network's exceptional recovery performance. According to experimental results, the ESP32-based mesh network was able to detect and reconfigure itself autonomously in 0.05 seconds on average when a node or network failed, whether as a result of hardware malfunction, power outage, or disconnection. The ESP32's Wi-Fi mesh networking protocol's built-in features, which enable automatic neighbor discovery and re-association without human assistance, are responsible for this quick recovery. The system's operational continuity was thus preserved even in cases where one or more nodes momentarily disconnected from the network; data transmission resumed nearly immediately upon their re-entry. Furthermore, data integrity was maintained throughout recovery, thanks to the integration of edge computing and decentralized data handling (through blockchain smart contracts). By functioning autonomously, the ESP32 nodes maintained their local processing power, enabling smooth task realignment and transition between active

nodes, reducing transmission delays and data loss. These results demonstrate the system's resilience and robustness, demonstrating that it can continue to monitor the environment in real time without experiencing major disruptions. With this performance level, the suggested system is ideal for deployment in settings like disaster monitoring zones, remote agricultural fields, or smart industrial sites where dependability and low downtime are crucial. However, even though the recovery rate was remarkable, it is acknowledged that extensive deployments in more complicated environmental settings or with more nodes. Without depending on a central authority, the implementation of a lightweight network with smart contracts guarantees unchangeable data records, improving trust and traceability. The dangers of data breaches, tampering, and single points of failure are greatly decreased by this decentralized strategy. Additionally, by concurrently uploading data to a blockchain and a local database, the system maintained redundancy, improving reliability and auditability. The system is highly suitable for time-sensitive applications such as precision agriculture, smart city deployments, and environmental risk assessment since it can successfully collect and transmit environmental data in every 15 seconds. The consistency of the observed temperature, humidity, and pressure readings across the nodes showed how well the BME280 sensors worked and how stable the communication network was. The study recognizes possible issues with scalability, power consumption optimization, and wider cloud integration, even though the prototype demonstrated remarkable efficacy in a controlled setting. Future improvements could include adding more nodes to the network, putting energy harvesting strategies into practice, and incorporating hybrid cloud-edge architectures to accommodate larger deployments and reduce latency even more. Overall, the study effectively confirms that utilizing a blockchain and IoT framework together is feasible.

Acknowledgement

The authors acknowledgement goes to the Pan Africa University Institute for Basic Science, Technology and Innovation for the support in finding this work.

References

[1] U. A. K. Betz *et al.*, “Game changers in science and technology - now and beyond,” *Technol Forecast Soc Change*, vol. 193, p. 122588, Aug. 2023, doi: 10.1016/J.TECHFORE.2023.122588.

[2] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, “Software Defined Networks-Based Smart Grid Communication: A Comprehensive Survey,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2637–2670, Jul. 2019, doi: 10.1109/COMST.2019.2908266.

[3] E. al. Nand Kumar, “Self-Healing Networks AI-Based Approaches for Fault Detection and Recovery,” *Power System Technology*, vol. 47, no. 4, pp. 371–386, Dec. 2023, doi: 10.52783/PST.206.

[4] D. C. Nguyen *et al.*, “6G Internet of Things: A Comprehensive Survey,” *IEEE Internet Things J*, vol. 9, no. 1, pp. 359–383, Jan. 2022, doi: 10.1109/JIOT.2021.3103320.

[5] M. Acevedo-Iles, D. Romero-Quete, and C. A. Cortes, “A Distributed Coordination Approach for Enhancing Protection System Adaptability in Active Distribution Networks,” *Energies 2024, Vol. 17, Page 4338*, vol. 17, no. 17, p. 4338, Aug. 2024, doi: 10.3390/EN17174338.

[6] S. Bosch, “BME280-Data sheet,” 2018.

[7] R. Uddin and I. Koo, “Real-Time Remote Patient Monitoring: A Review of Biosensors Integrated with Multi-Hop IoT Systems via Cloud Connectivity,” Mar. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app14051876.

[8] A. Benharref and M. A. Serhani, “Novel cloud and SOA-based framework for E-health monitoring using Novel cloud and SOA-based framework for E-health monitoring using wireless biosensors wireless biosensors.” [Online]. Available: https://ro.uow.edu.au/dubaipapers/550

[9] S. Tern, “Survey of Smart Contract Technology and Application Based on Blockchain,” *Open Journal of Applied Sciences*, vol. 11, no. 10, pp. 1135–1148, 2021, doi: 10.4236/OJAPPS.2021.1110085.

[10] M. G. Alles, A. Kogan, and M. A. Vasarhelyi, “Restoring auditor credibility: Tertiary monitoring and logging of continuous assurance systems,” *International Journal of Accounting Information Systems*, vol. 5, no. 2, pp. 183–202, Jul. 2004, doi: 10.1016/j.accinf.2004.01.010.

[11] S. Mirzamohammadi, J. A. Chen, A. A. Sani, S. Mehrotra, and G. Tsudik, “Trustworthy Auditing of Sensor Activities in Mobile & IoT Devices,”

Mobile & IoT Devices, vol. 14, no. 17, pp. 1–14, Nov. 2017, doi: 10.1145/3131672.3131688.

[12] T. Brockmann, M. Rethfeldt, B. Beichler, F. Golatowski, and C. Haubelt, "MAC-Filter based Topology Control for WLAN Mesh Networks", Accessed: Sep. 20, 2024. [Online]. Available: https://github.com/o11s/open80211s/wiki/HOWTO

[13] L. Claire, O. K. Shana, and W. Zongjie, "Self-Healing Materials for Bioelectronic Devices - Liu - 2024 - Advanced Materials - Wiley Online Library," Wiley-VCHGmbH. Accessed: Sep. 20, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.202401219

[14] S. Vladov, R. Yakovliev, V. Vysotska, M. Nazarkevych, and V. Lytvyn, "The Method of Restoring Lost Information from Sensors Based on Auto-Associative Neural Networks," *Applied System Innovation 2024, Vol. 7, Page 53*, vol. 7, no. 3, p. 53, Jun. 2024, doi: 10.3390/ASI7030053.

[15] I. Burcea, H. A. Jacobsen, E. De Lara, V. Muthusamy, and M. Petrovic, "Disconnected operation in publish/subscribe middleware," *Proceedings - 2004 IEEE International Conference on Mobile Data Management*, pp. 39–50, 2004, doi: 10.1109/MDM.2004.1263041.

[16] N. Afraz, F. Wilhelmi, H. Ahmadi, and M. Ruffini, "Blockchain and Smart Contracts for Telecommunications: Requirements vs. Cost Analysis," *IEEE Access*, vol. 11, pp. 95653–95666, 2023, doi: 10.1109/ACCESS.2023.3309423.

[17] S. Wang *et al.*, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends; Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," *SYSTEMS*, vol. 49, no. 11, 2019, doi: 10.1109/TSMC.2019.2895123.

[18] O. Takahiro, "Handle smart contract on Ethereum with Arduino or ESP32 | by Takahiro Okada | Medium," Medium. Accessed: Sep. 22, 2024. [Online]. Available: https://medium.com/@takahirookada/handle-smart-contract-on-ethereum-with-arduino-or-esp32-1bb5cbaddbf4