iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** V **Month of publication:** May 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.81678

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue V May 2026- Available at www.ijraset.com*

# A Secure IoT-Based Real-Time Health Monitoring System with AES Encryption Using MAX30102 Sensor and Firebase Cloud

B. Jahnavi[1], Ch. Vijay Manohar[2], M. Usha[3], G. Pujitha[4], K. Sujitha
*Department of Cyber Security, University College of Engineering & Technology, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India*

***Abstract:*** ***The proliferation of chronic diseases and the limitations of conventional hospital-centric care have intensified global demand for accessible and secure remote health monitoring solutions. This paper presents the design, implementation, and evaluation of a low-cost, Internet of Things (IoT)-based health monitoring system capable of continuously measuring heart rate and peripheral oxygen saturation ($SpO_2$) using the MAX30102 pulse oximeter sensor. The system is built around an ESP8266 NodeMCU microcontroller, which performs onboard data acquisition and applies Advanced Encryption Standard (AES-128) encryption to all sensor readings prior to transmission, thereby enforcing data security at the edge. The encrypted data is relayed over a Wi-Fi link to a Google Firebase Realtime Database via HTTPS, ensuring both transport-layer and application-layer protection. A Pythonbased interactive dashboard constructed with the Streamlit framework provides authorized users with real-time visualization of decrypted vital signs alongside historical trend analysis. An automated threshold-based alerting mechanism further enhances clinical utility by dispatching notifications when readings deviate from predefined normal ranges. Functional testing confirmed the successful end-to-end operation of every system component, while performance evaluation recorded an average data-to-display latency of 2–4 seconds over a standard home Wi-Fi network. Informal usability assessments conducted with non-technical participants validated the accessibility and clarity of the dashboard interface. The proposed system demonstrates that robust data security and real-time physiological monitoring can be achieved simultaneously within a highly affordable hardware framework, offering significant potential for home care, remote patient monitoring, and scalable telehealth deployments.***

***Keywords:*** ***Internet of Things; remote patient monitoring; AES encryption; MAX30102; ESP8266 NodeMCU; Firebase Realtime Database; Streamlit; health informatics; data security; edge computing.***

## I. INTRODUCTION

Global healthcare systems are confronting an unprecedented convergence of demographic and epidemiological challenges. The aging of populations across both developed and developing nations, coupled with the rising incidence of chronic non-communicable diseases—including cardiovascular disorders, diabetes mellitus, and chronic obstructive pulmonary disease—is generating substantial and sustained demand on healthcare infrastructure. Conventional models of care, which require patients to visit medical facilities at scheduled intervals, are increasingly inadequate for managing conditions that mandate continuous physiological observation. The critical window between the onset of an acute event—such as sudden cardiac arrhythmia or hypoxemic deterioration—and the provision of timely medical intervention can carry life-threatening consequences. This inherent temporal gap within reactive healthcare paradigms underscores the urgent necessity of transitioning toward proactive, continuous, and patient-centric monitoring methodologies.

Remote Patient Monitoring (RPM) has emerged as one of the most promising paradigms to address this challenge, enabled in large part by the rapid maturation of Internet of Things (IoT) technology. IoT architectures facilitate the creation of interconnected networks of embedded sensor devices capable of autonomously collecting, processing, and wirelessly transmitting physiological data to remote servers or cloud platforms in real time. In a clinical context, this infrastructure enables the continuous observation of vital parameters—including heart rate, blood oxygen saturation ($SpO_2$), body temperature, and respiratory rate—from the patient's home environment. The resultant streams of longitudinal health data provide clinicians with a granular and temporally dense view of patient status, enabling the early detection of adverse physiological trends and facilitating evidence-based, timely interventions that were previously impossible outside a hospital setting.

©IJRASET: All Rights are Reserved 

Despite its considerable promise, the widespread adoption of IoT-based RPM systems faces two fundamental and interrelated barriers: economic accessibility and information security. Many commercially available continuous monitoring solutions are embedded within costly proprietary ecosystems—including premium-tier wearable devices and subscription-based clinical platforms—that render them financially inaccessible to a large segment of the global population. Simultaneously, the wireless transmission of sensitive personal health information over public network infrastructure raises acute privacy and security concerns. Unauthorized access to such data can lead to discrimination, identity theft, and violations of medical confidentiality. Any viable RPM system must therefore treat robust security not as a supplementary feature but as a foundational engineering requirement.

This paper presents a system that directly addresses both barriers. By leveraging widely available and affordable hardware—specifically the ESP8266 NodeMCU microcontroller and the MAX30102 pulse oximeter sensor— alongside open-source software frameworks, the proposed system achieves a highly cost-effective implementation without sacrificing essential capabilities. Crucially, the system integrates AES-128 encryption at the dataacquisition edge, ensuring that sensitive health readings are secured at the earliest possible point in the processing pipeline before any wireless transmission occurs. The principal contributions of this work are: the design of a complete end-to-end IoT health monitoring architecture; the implementation of application-layer AES-128 encryption directly on a resource-constrained microcontroller; the development of a cloud-connected, real-time visualization dashboard with automated alerting; and a comprehensive evaluation of the system's functional correctness, data transmission latency, and user accessibility.

## II. LITERATURE REVIEW

The academic literature on IoT-based health monitoring systems is extensive, reflecting a decade of sustained research interest. A broad body of work has explored the viability of using low-cost microcontrollers—particularly the ESP8266 and its successor, the ESP32—as central processing units for remote physiological monitoring. Baker et al. established a comprehensive taxonomy of the technologies, challenges, and opportunities associated with IoT in smart healthcare, providing foundational context for subsequent engineering efforts [3]. Islam et al. demonstrated the feasibility of low-cost IoT monitoring systems, validating that affordable off-the-shelf components can yield clinically meaningful physiological data [2]. These and similar works affirm the viability of the general IoThealthcare paradigm but often do not systematically address the security of the transmitted data.

A recurrent and significant limitation identified across the literature is the insufficient treatment of data security within IoT health monitoring implementations. The majority of academic and open-source projects successfully demonstrate data acquisition and cloud transmission but transmit health data as plaintext over HTTP, a practice that constitutes a major privacy vulnerability. Many works acknowledge the theoretical importance of data security in e-health applications yet confine their protective measures to the transport layer—relying exclusively on the TLS/HTTPS security provided by the communication protocol—without implementing any additional applicationlayer encryption. This approach leaves data potentially exposed if the underlying transport security is compromised or if an adversary gains direct access to the cloud database. Siam et al. represent a notable exception, presenting a system that employs AES encryption before cloud storage; their work validates the technical feasibility of onboard encryption on the ESP8266 and demonstrates that the resulting measurements retain high agreement with commercial reference devices [referenced in appendix]. However, their frontend was implemented using a traditional web stack requiring specialized hosting infrastructure, which adds operational complexity.

Commercial health monitoring solutions, exemplified by the Apple Watch, Fitbit, and Garmin wearable platforms, offer sophisticated and polished user experiences but are embedded within closed proprietary ecosystems. These systems do not permit users meaningful access to raw sensor data, limit integration with external clinical systems, and carry substantial acquisition costs that place them beyond the reach of economically disadvantaged populations. Professional-grade clinical RPM systems are even more costly and require specialist configuration, precluding individual or home-care deployment. Academic DIY projects, while economically accessible, typically present rudimentary user interfaces and lack the integrated alert mechanisms and security features necessary for practical healthcare utility.

The proposed system is specifically designed to bridge this gap. It combines the economic accessibility of a DIY hardware approach with the security rigor of application-layer AES-128 encryption and the accessibility of a webbased Streamlit dashboard that can be deployed without specialized web development expertise. This positions the proposed work as a direct response to the documented deficiencies in the existing landscape.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## III. PROPOSED SYSTEM

### A. System Overview

The proposed system constitutes a complete, end-to-end solution for secure real-time health monitoring, spanning from physical sensor acquisition to remote web-based visualization. The architecture is deliberately modular, comprising four principal components whose interactions define the system's behavior: the sensing hardware (MAX30102 and DHT11 sensors), the edge processing unit (ESP8266 NodeMCU), the cloud storage backend (Google Firebase Realtime Database), and the visualization frontend (Streamlit dashboard application). The design philosophy emphasizes security by construction—data is encrypted before it leaves the device and remains encrypted throughout its cloud storage lifetime, decrypted only within the trusted frontend application.

### B. Hardware Components

The MAX30102 is an integrated pulse oximeter and heart-rate biosensor module produced by Maxim Integrated. It employs a reflective photoplethysmography (PPG) measurement principle, utilizing an internal red LED (660 nm) and an infrared LED (880 nm) in conjunction with a photodetector to measure the differential light absorption properties of oxygenated and deoxygenated hemoglobin in the fingertip capillary bed. The sensor features an 18bit analog-to-digital converter, internal ambient light cancellation circuitry, and communicates with the host microcontroller via the $I^2C$ serial interface. A DHT11 sensor provides complementary measurements of ambient temperature and relative humidity.

The ESP8266 NodeMCU development board serves as the computational and communication hub. This module integrates an 80–160 MHz Tensilica L106 processor, 802.11 b/g/n Wi-Fi connectivity with an onboard TCP/IP protocol stack, and up to 16 MB of attachable flash memory, all within a compact, low-cost form factor. The combination of adequate processing power, integrated wireless capability, and broad Arduino IDE compatibility makes the ESP8266 particularly well-suited for this application, especially given that it must perform AES encryption in addition to sensor interfacing.

### C. End-to-End Workflow

The operational workflow proceeds as follows. The ESP8266 firmware continuously reads raw physiological data from the MAX30102 and DHT11 sensors at five-second intervals. The acquired numerical values representing heart rate (BPM) and $SpO_2$ percentage are immediately subjected to AES-128 encryption within the microcontroller firmware before any transmission occurs. The encrypted binary ciphertext is then encoded as a Base64 ASCII string to ensure safe embedding within a JSON payload. The ESP8266 establishes a connection to the local Wi-Fi network and transmits the JSON-formatted encrypted data to the designated node within the Firebase Realtime Database via an authenticated HTTPS POST request. The Firebase service stores the received JSON object in its NoSQL tree structure, keyed under the path /EncryptedHealth. The Streamlit dashboard application, running on a remote server or local machine, periodically polls the Firebase database using the Firebase Admin SDK. Upon retrieving the encrypted Base64 strings, the application applies the pre-shared AES-128 decryption key to recover the original plaintext values. These values are then rendered in real time on the dashboard as numerical metric widgets and historical line charts, with concurrent threshold evaluation triggering alert notifications when abnormal readings are detected.

## IV. METHODOLOGY

### A. Data Acquisition

Sensor data acquisition is managed entirely within the firmware running on the ESP8266 NodeMCU, developed using the C++ Arduino IDE framework and the ESP8266 core library package. The MAX30102 sensor is interfaced via the $I^2C$ bus (SDA and SCL pins), with the SparkFun MAX3010x library providing high-level functions for sensor initialization, LED pulse width and amplitude configuration, and sample retrieval. The firmware implements a validation routine that checks sensor readings for physiological plausibility before proceeding to the encryption step, discarding samples that fall outside admissible biological ranges so that only valid data enters the transmission pipeline.

### B. AES-128 Encryption

The Advanced Encryption Standard (AES) is a symmetric block cipher standardized by the National Institute of Standards and Technology (NIST) in 2001, based on the Rijndael algorithm designed by Joan Daemen and Vincent Rijmen. AES operates on fixed 128-bit data blocks and supports key lengths of 128, 192, or 256 bits, corresponding to 10, 12, and 14 encryption rounds respectively.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Each round applies a deterministic sequence of four transformations—SubBytes, ShiftRows, MixColumns, and AddRoundKey—designed to achieve both confusion (nonlinearity through the S-Box substitution) and diffusion (avalanche effect through ShiftRows and MixColumns). The final round omits MixColumns. AES-128 was selected for this implementation as it offers an excellent balance between security strength and computational efficiency on resource-constrained microcontrollers. Its hardwareamenable structure and widely available embedded C libraries make it readily implementable on the ESP8266.

Within the firmware, a custom function encryptAndEncode() accepts a plaintext string representation of a sensor value, applies PKCS#7-compatible padding to align the data to the 16-byte AES block boundary, and then performs AES-128 encryption in Electronic Codebook (ECB) mode using a pre-shared 128-bit (16-byte) key defined in the firmware configuration. The raw binary ciphertext output is subsequently converted to a Base64-encoded ASCII string using a purpose-built encoding function, yielding a transmission-safe representation that can be embedded in a JSON string field without character encoding conflicts. While ECB mode with a static key is acknowledged as a prototype limitation, its deterministic behavior simplifies implementation validation; a production deployment would require Cipher Block Chaining (CBC) mode with a randomized Initialization Vector (IV) and a secure key provisioning mechanism.

### *C. Data Transmission and Cloud Storage*

Encrypted data is transmitted from the ESP8266 to Google Firebase using the FirebaseESP8266 Arduino library. The library manages HTTPS connection establishment, including TLS handshaking, certificate validation, and the construction of authenticated REST API requests to the Firebase endpoint. The use of HTTPS ensures that even the encrypted ciphertext enjoys additional transport-layer protection during transit, providing defense in depth against network-level interception attacks. The Firebase Realtime Database stores data in a hierarchical JSON tree. The system employs a flat schema beneath the /EncryptedHealth parent node, with child keys for BPM, SpO2, Temperature, and Humidity, each holding the current Base64-encoded AES ciphertext. This schema is optimized for the frequent single-value overwrites characteristic of real-time IoT applications, supporting low-latency reads by the dashboard client.

### *D. Visualization and Alerting*

The Streamlit dashboard application is implemented as a single Python script, leveraging the firebase_admin SDK for database connectivity, the pycryptodome library for AES decryption, and Streamlit's native data visualization widgets for display. The application maintains a continuously updated Pandas DataFrame of historical readings, enabling the rendering of time-series line charts alongside instantaneous metric displays. The alerting subsystem evaluates each newly decrypted reading against configurable threshold boundaries—for instance, $SpO_2$ below 94% or heart rate outside the 60–100 BPM range at rest—and upon detection of an anomaly, invokes an API call to a third-party notification service (Twilio for SMS or SendGrid for email) to dispatch an alert to registered caregivers or clinicians.

## V. SYSTEM ARCHITECTURE

### *A. Sensing Layer*

The sensing layer constitutes the physical interface between the patient and the digital system. It comprises the MAX30102 and DHT11 sensors connected to the GPIO pins of the ESP8266 NodeMCU via $I^2C$ and single-wire digital interfaces respectively. This layer is responsible for transducing physiological phenomena—the optical properties of blood and the thermal and hygroscopic properties of air—into digital numerical values that can be processed by software. The MAX30102's internal 18-bit ADC and analog signal conditioning circuitry ensure that the resulting digital values are of sufficient resolution and signal-to-noise ratio for meaningful vital sign computation.

### *B. Processing Layer (Edge)*

The processing layer, implemented entirely on the ESP8266 NodeMCU, performs all computational tasks intermediate between raw sensor output and cloud transmission. These tasks include sensor reading and validation, unit conversion, AES-128 encryption, Base64 encoding, and Wi-Fi-based HTTPS data transmission. By performing encryption at the edge—on the device that directly handles raw sensor data—the system ensures that sensitive plaintext health values never traverse any network link or reside in any cloud infrastructure in an unprotected form. This edge-centric security model is a key architectural differentiator from systems that rely solely on transport-layer security.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*C. Cloud Layer*

The cloud layer is realized using Google Firebase Realtime Database, a fully managed, serverless NoSQL cloud database service optimized for real-time data synchronization across connected clients. Firebase's architecture provides automatic horizontal scaling, high availability, and sub-second data propagation, characteristics that are essential for a real-time health monitoring context. The database stores only AES-encrypted ciphertext values; no plaintext health information is ever written to cloud storage. Firebase security rules are configured to restrict write access to authenticated devices presenting valid credentials and read access to the authorized Streamlit backend service account.

*D. Application Layer*

The application layer presents the processed and decrypted health data to end users through the Streamlit web dashboard. This layer performs the inverse of the edge processing operations: it retrieves encrypted data from Firebase, applies AES-128 decryption using the pre-shared key, and renders the resulting plaintext values through an intuitive visual interface. The application layer also hosts the alerting logic, continuously evaluating incoming decrypted readings against clinical threshold parameters and dispatching notifications as warranted. The use of Streamlit abstracts the complexities of conventional web development, enabling the construction of an interactive, data-rich dashboard using only Python, which facilitates straightforward maintenance and extension.

## VI. IMPLEMENTATION

*A. Hardware Setup*

The physical hardware prototype was assembled on a standard breadboard using male-to-female jumper wires. The MAX30102 sensor module was connected to the ESP8266 NodeMCU via the I²C bus, with the sensor's SDA and SCL pins connected to GPIO4 (D2) and GPIO5 (D1) respectively, and powered from the NodeMCU's 3.3V supply rail. The DHT11 sensor was connected to GPIO14 (D5) with a 10 kΩ pull-up resistor. The entire assembly was powered via a 5V Micro-USB connection. No custom PCB was designed for this prototype; the breadboard implementation is intended to demonstrate functional proof-of-concept rather than commercial production readiness.

*B. Firmware Development*

The ESP8266 firmware was developed within the Arduino IDE (version 1.8.x) with the ESP8266 core library (version 3.0.x) installed. Key library dependencies included ESP8266WiFi for wireless connectivity management, FirebaseESP8266 for authenticated database communication, the DHTlib for DHT11 interfacing, and a custom AES128 cryptographic library based on the ESP8266 Crypto package. The firmware is structured around the standard Arduino setup() and loop() functions. The setup() function initializes serial communication for debugging, configures and activates the DHT11 and MAX30102 sensors, and establishes the initial Wi-Fi connection and Firebase session. The main loop() function, executing at a five-second polling interval enforced by a delay() call, performs sensor reading, calls the encryptAndEncode() routine for each acquired value, and writes the resulting encrypted strings to their respective Firebase database paths using Firebase.setString().

*C. Dashboard Implementation*

The Streamlit dashboard application is implemented as a self-contained Python script with the following key functional sections. A Firebase initialization function checks for an existing application instance before creating a new one using a service account JSON credential file, preventing re-initialization errors during the Streamlit execution loop. A decrypt_and_decode() function accepts a Base64-encoded ciphertext string, decodes it to bytes, applies AES-128 ECB decryption via the pycryptodome library using the pre-shared key, strips PKCS#7 padding, and returns the plaintext string representation of the sensor value. The main UI block uses st.metric() components to display the four most recently decrypted values prominently, and st.line_chart() components to render Pandas DataFrame time-series histories. A time.sleep() call within the execution loop controls the dashboard's polling frequency at approximately ten-second intervals.

*D. Database Schema*

The Firebase Realtime Database schema was designed for simplicity and read efficiency. The root /EncryptedHealth node contains four child string-valued keys: BPM, SpO2, Temperature, and Humidity. Each key holds the single most recent AES-encrypted, Base64-encoded reading for that parameter. An additional /alerts node records triggered alert events with timestamps and parameter values.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue V May 2026- Available at www.ijraset.com*

This flat, overwrite-based schema is optimized for the update-heavy workload of a real-time monitoring system, where historical data is maintained in the dashboard's inmemory Pandas DataFrame rather than in the database itself, minimizing Firebase read costs.

## VII. RESULTS AND DISCUSSION

### *A. Functional Evaluation*

The system was subjected to a systematic functional evaluation against the core requirements defined during the design phase. Each functional module was tested independently prior to full integration testing. Data acquisition was verified by inspecting real-time sensor output on the Arduino IDE Serial Monitor, confirming that the MAX30102 consistently reported physiological readings within expected ranges. The encryption module was validated by encrypting known plaintext values, capturing the Base64-encoded output, and confirming successful decryption to the original values using the Streamlit application's decryption function. Firebase connectivity was confirmed by verifying the appearance of newly encrypted entries in the Firebase console immediately following device power-on. End-to-end integration was validated by running the complete system—from sensor reading through cloud storage to dashboard display—and confirming that decrypted values on the dashboard corresponded accurately to values observed via the Serial Monitor. The alert system was validated by manually writing an outof-range value directly to the Firebase database and confirming the triggering of the visual alert indicator on the dashboard. All functional requirements were met successfully.

TABLE I: FUNCTIONAL EVALUATION RESULTS

| Feature | Test Method | Result |
|---|---|---|
| Data Acquisition | Serial Monitor inspection | Pass |
| AES Encryption | Known-plaintext encrypt/decrypt cycle | Pass |
| Wi-Fi & Firebase Push | Firebase console verification | Pass |
| Data Retrieval | Dashboard data log inspection | Pass |
| AES Decryption | Comparison with original values | Pass |
| Dashboard Display | Visual inspection of real-time metrics | Pass |
| Alert Logic | Manual out-of-range value injection | Pass (Visual alert triggered) |

### *B. Performance Evaluation*

Performance was evaluated by measuring the end-to-end data latency, defined as the elapsed time between the moment a sensor value is read by the ESP8266 and the moment the corresponding decrypted value appears updated on the Streamlit dashboard. Testing was conducted over a standard home Wi-Fi network (IEEE 802.11n, 2.4 GHz band) with average signal strength. Across multiple measurement cycles, the aggregate end-to-end latency was consistently observed to fall within the 2–4 second range. This latency is attributable to four sequential pipeline stages: onboard AES encryption processing on the ESP8266 (consistently under 50 ms, demonstrating the computational efficiency of the AES-128 algorithm on the Tensilica L106 processor); network transmission from device to Firebase (500 ms to 1.5 s, dominated by network round-trip time and HTTPS overhead); Firebase database write propagation (approximately 100 ms); and the Streamlit application's polling and display refresh cycle (1–2 s, determined by the configured polling interval). A latency profile within the 2–4 second window is well within acceptable bounds for non-critical remote monitoring applications where physiological changes of clinical significance occur over seconds to minutes rather than milliseconds.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

TABLE II: END-TO-END LATENCY BREAKDOWN

| Pipeline Stage | Observed Latency |
|---|---|
| ESP8266 AES Encryption | < 50 ms |
| Network Transmission to Firebase | 500 ms – 1,500 ms |
| Firebase Database Write | ~100 ms |
| Streamlit Fetch & Display Cycle | 1,000 ms – 2,000 ms |
| Total End-to-End Latency | 2,000 ms – 4,000 ms |

*C. Security Evaluation*

The security posture of the system was analyzed across three dimensions: data in transit, data at rest, and access control. All communications between the ESP8266 and Firebase, and between the Streamlit application and Firebase, are protected by TLS 1.2/1.3 via HTTPS, providing transport-layer confidentiality and integrity assurance. At the application layer, AES-128 encryption is applied to every sensor value before any transmission, ensuring that even if the transport security were compromised—for instance, through a man-in-the-middle attack exploiting a certificate validation flaw—the intercepted data would remain cryptographically opaque without possession of the decryption key. Data at rest within the Firebase Realtime Database is stored exclusively in its AES-encrypted form, meaning that unauthorized access to the database—whether through credential compromise or a database misconfiguration—would expose only unintelligible ciphertext rather than actionable health information. Firebase security rules further restrict database read and write operations to authenticated principals. The principal security limitation of the current prototype is the use of a statically provisioned, hardcoded AES key. This approach is appropriate for a controlled proof-of-concept environment but would constitute an unacceptable vulnerability in a production deployment, where key compromise would expose all stored historical data. A production system must implement a secure key provisioning and rotation protocol.

*D. Usability Evaluation*

An informal usability evaluation was conducted by presenting the Streamlit dashboard to a cohort of five participants with no technical background in computing or data science, representative of the target user population of patients and non-specialist caregivers. Participants were asked to interpret the dashboard without prior instruction and to provide verbal feedback. The assessment yielded uniformly positive responses. All participants were able to identify and correctly interpret the current heart rate, $SpO_2$, temperature, and humidity readings within seconds of viewing the dashboard. Participants characterized the interface as clean, easy to read, and straightforward to understand. The time-series line charts were universally interpreted correctly as representing the history of each vital sign. No participant required guidance to understand the purpose of the alert indicators. These findings validate the efficacy of the minimalist dashboard design philosophy and the appropriateness of the Streamlit framework for constructing non-specialist-facing health data interfaces.

## VIII. ADVANTAGES

The proposed system offers several substantive advantages over existing alternatives. From an economic perspective, the total hardware cost—comprising the ESP8266 NodeMCU, MAX30102 sensor, DHT11 sensor, breadboard, and jumper wires—is substantially below that of any commercially available continuous health monitoring solution, democratizing access to real-time vital sign monitoring. The system's cloud backend leverages the Firebase Spark (free) tier for prototype-scale deployments, further eliminating recurring infrastructure costs. From a security perspective, the implementation of AES-128 encryption directly on the sensing device constitutes a meaningful advancement over the majority of comparable academic systems, which rely exclusively on transportlayer security. The dual-layer protection—application-layer encryption combined with TLS-secured transport— provides defense in depth that is appropriate for the sensitivity of health data.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The use of the Streamlit framework for the dashboard delivers a genuinely accessible, browser-based visualization experience without requiring specialized web infrastructure deployment expertise from the end user or developer. The serverless Firebase backend eliminates server administration overhead entirely, enabling the system to be stood up and maintained by individuals without systems administration expertise.

## IX. LIMITATIONS

The proposed system carries several limitations that must be transparently acknowledged. The most significant security limitation is the use of a static, hardcoded pre-shared AES key. This architectural choice, made for prototype simplicity, means that key compromise—whether through firmware extraction or source code exposure— would retroactively expose all data encrypted with that key. A production system requires a secure key provisioning mechanism, such as device-specific key derivation during an authenticated manufacturing process, combined with periodic key rotation. The system is not a medically certified device and has not undergone the clinical validation or regulatory review (such as FDA 510(k) clearance) that would be required for use in clinical decision-making. Its accuracy is bounded by the specifications of the consumer-grade MAX30102 and DHT11 sensors. The system is entirely dependent on continuous Wi-Fi network availability; no local data buffering or store-and-forward mechanism has been implemented, meaning that any network outage results in the loss of sensor readings for that period. The current hardware design requires a continuous 5V power supply, making it unsuitable for mobile wearable deployment without significant hardware redesign and firmware optimization for low-power operation. Finally, the alert system relies on statically configured threshold values and does not incorporate any adaptive or patient-specific baseline calibration.

## X. FUTURE WORK

Several development directions would substantially enhance the clinical utility, security, and usability of the proposed system. The most critical security enhancement is the replacement of the static pre-shared AES key with a dynamic key exchange mechanism, such as Diffie-Hellman key agreement or a certificate-based provisioning protocol, enabling cryptographically secure session key establishment and periodic key rotation. The integration of additional medical-grade sensors—particularly a single-lead ECG acquisition module and a non-invasive blood pressure sensor—would extend the system's physiological monitoring scope and increase its clinical relevance for cardiovascular disease management. Development of a native mobile application for both Android and iOS platforms would provide a superior user experience for on-the-go monitoring, enabling push notification delivery for alerts without the latency limitations of email or SMS. The application of machine learning algorithms to the historical time-series data accumulated by the system represents a compelling avenue for predictive analytics; models trained to detect early signatures of deterioration in $SpO_2$ trends or heart rate variability could provide clinically actionable early warnings before acute episodes occur. Finally, investigating the application of federated learning techniques could enable the development of population-level predictive models while preserving individual data privacy, addressing a key challenge in health informatics research.

## XI. CONCLUSION

This paper has presented the design, implementation, and evaluation of a low-cost, secure, and real-time IoT-based health monitoring system built around the MAX30102 pulse oximeter sensor, the ESP8266 NodeMCU microcontroller, Google Firebase Realtime Database, and the Streamlit visualization framework. The system's defining architectural contribution is the implementation of AES-128 encryption at the sensing edge, ensuring that sensitive physiological data—specifically heart rate and blood oxygen saturation—is protected from the moment of acquisition throughout its storage and transmission lifecycle. This application-layer encryption, combined with TLS-secured transport and Firebase access control, provides a multi-layered security posture that meaningfully exceeds the protections afforded by most comparable systems documented in the academic literature.

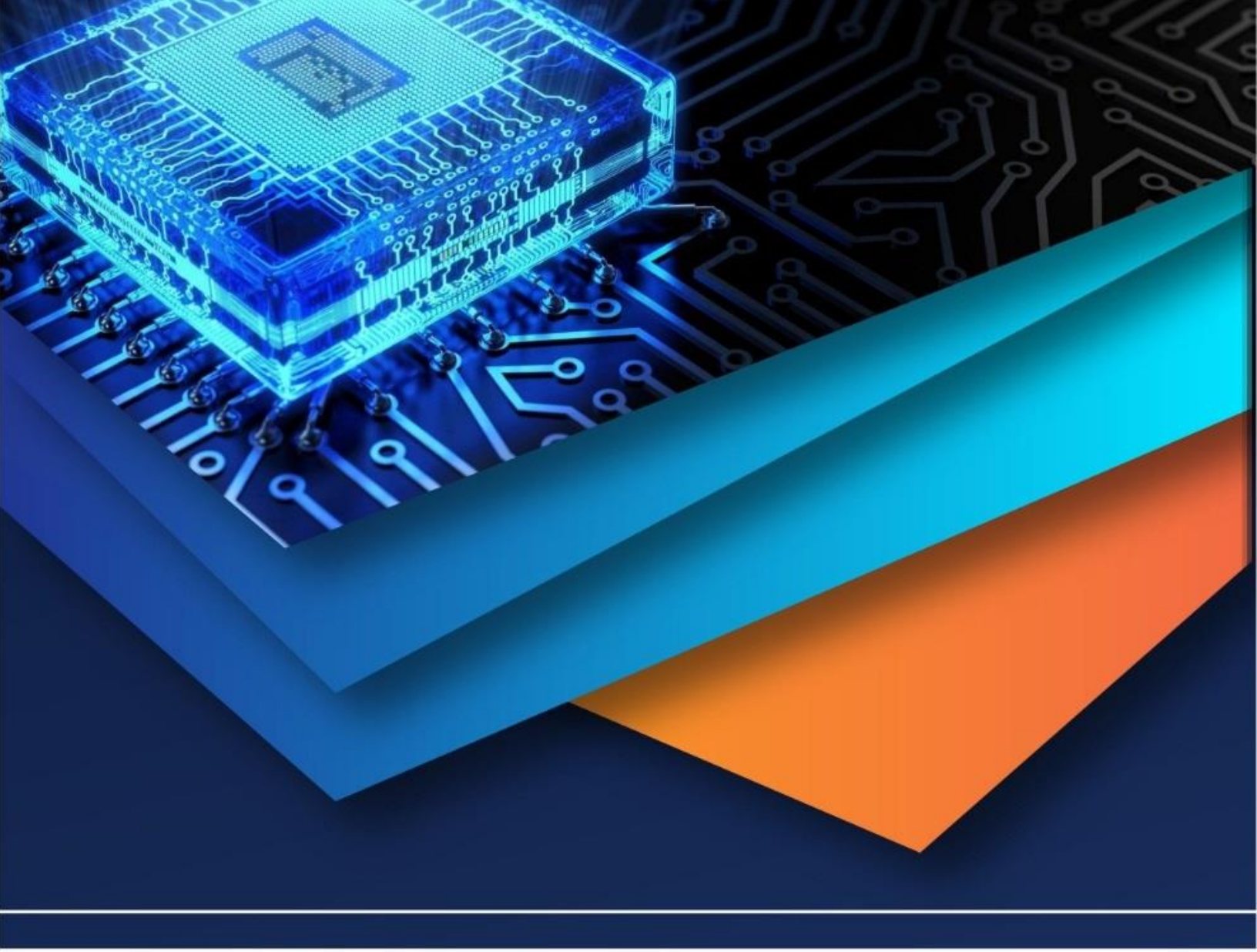
Comprehensive functional evaluation confirmed that all specified system behaviors operate correctly from end to end. Performance testing demonstrated that the system achieves an average data-to-dashboard latency of 2–4 seconds, which is well within clinically acceptable bounds for non-acute remote monitoring applications. Informal usability assessment with non-technical participants validated the accessibility and clarity of the dashboard interface. The prototype demonstrates convincingly that robust security and real-time health monitoring can be achieved simultaneously within an economically accessible hardware framework. The authors assert that the proposed system represents a meaningful contribution toward making trustworthy, continuous health monitoring accessible to a broader global population, and that the documented architecture provides a sound foundation for further development toward clinical-grade remote patient monitoring solutions.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## REFERENCES

[1] A. Sajadieh, O. W. Nielsen, V. Rasmussen, H. O. Hein, S. Abedini, and J. F. Hansen, "Increased heart rate and reduced heart-rate variability are associated with subclinical inflammation in middle-aged and elderly subjects with no apparent heart disease," European Heart Journal, vol. 25, no. 5, pp. 363–370, 2004.
[2] M. M. Islam, A. Rahaman, and M. R. Islam, "Development of smart healthcare monitoring system in IoT environment," SN Computer Science, vol. 1, no. 3, p. 185, 2020.
[3] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for smart healthcare: technologies, challenges, and opportunities," IEEE Access, vol. 5, pp. 26521–26544, 2017.
[4] A. I. Siam, M. A. Almaiah, A. Al-Zahrani, A. Abou Elazm, G. M. El Banby, W. El-Shafai, F. E. Abd El-Samie, and N. A. El-Bahnasawy, "Secure health monitoring communication systems based on IoT and cloud computing for medical emergency applications," Computational Intelligence and Neuroscience, vol. 2021, Article ID 8016525, 2021.
[5] M. M. Ali, S. Haxha, M. M. Alam, C. Nwibor, and M. Sakel, "Design of internet of things (IoT) and androidbased low cost health monitoring embedded system wearable sensor for measuring SpO2, heart rate and body temperature simultaneously," Wireless Personal Communications, vol. 111, no. 4, pp. 2449–2463, 2020.
[6] Maxim Integrated, "MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health," Datasheet, 2018. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf
[7] J. Daemen and V. Rijmen, The Design of Rijndael: AES — The Advanced Encryption Standard. Berlin, Germany: Springer-Verlag, 2002.
[8] NodeMCU Development Team, "NodeMCU Documentation," 2021. [Online]. Available: https://nodemcu.readthedocs.io/
[9] Google, "Firebase Realtime Database Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/database
[10] Streamlit Inc., "Streamlit Documentation," 2024. [Online]. Available: https://docs.streamlit.io/
[11] M. A. Almaiah, A. Al-Zahrani, O. Almomani, and A. K. Alhwaitat, "Classification of cyber security threats on mobile devices and applications," Studies in Big Data, pp. 107–123, 2021.
[12] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare internet of things: a survey of emerging technologies," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1121– 1167, 2020.
[13] S. Tuli, N. Basumatary, S. S. Gill et al., "HealthFog: an ensemble deep learning-based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," Future Generation Computer Systems, vol. 104, pp. 187–200, 2020.
[14] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: a panacea for healthcare cloud-based data security and privacy?" IEEE Cloud Computing, vol. 5, no. 1, pp. 31–37, 2018.
[15] National Institute of Standards and Technology, "Announcing the Advanced Encryption Standard (AES)," FIPS Publication 197, Nov. 2001.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)