

Hybrid Telecom Network Analyzer: A Real-Time Signal Mapping Framework Using IoT and Data Science

Roshan Rajendra Burke¹, Nikhil Baban Kadam², Gayatri Gangadhar Yeola³

^{1,2,3}Department of Electronics & Telecommunication Engineering
Matoshri College of Engineering & Research Centre, Eklahare, Nashik, Maharashtra, India
(Affiliated to Savitribai Phule Pune University)

Project Guide: **Dr. S. B. Borse**

Abstract - Reliable cellular connectivity is increasingly central to daily communication, yet end users routinely encounter signal fluctuation, coverage gaps, and call instability that commercial telecom monitoring platforms are too costly or too infrastructure-heavy to diagnose at a local scale. This paper presents the Hybrid Telecom Network Analyzer, a low-cost embedded-IoT system that performs real-time signal mapping by combining a SIM900A GSM module, an Arduino Uno microcontroller, and a Python-based data analytics pipeline. The Arduino Uno polls the GSM module over serial communication using AT command queries (AT+CSQ) to obtain received signal strength indicator (RSSI) values, which are streamed to a host computer and processed using Pandas and NumPy for data structuring and statistical analysis. Processed signal trends are rendered through Matplotlib and exposed to the end user via an interactive Streamlit dashboard supporting live visualization of network behaviour. A simulated-data fallback routine was implemented to sustain continuous dashboard operation during intervals of module non-response, ensuring uninterrupted demonstration of the analytics and visualization layers independent of transient hardware faults. The complete prototype was assembled using jumper-wire interconnects without a breadboard, validated for serial data integrity, and evaluated for dashboard responsiveness under varying signal conditions. Results confirm that the integration of embedded telecom hardware with a Python data-science stack yields a scalable, low-cost alternative to commercial network-monitoring infrastructure, suitable for academic, research, and small-scale telecom diagnostic applications.

Key Words: Internet of Things, Telecom Signal Monitoring, GSM Module, SIM900A, Arduino Uno, Data Science, Real-Time Visualization, Streamlit Dashboard.

I. INTRODUCTION

The growth of mobile communication has made stable cellular connectivity a basic expectation, yet network instability, weak-coverage zones, and intermittent call quality remain common in practice. Diagnosing such issues typically requires telecom-grade drive-test equipment or carrier-side analytics platforms, both of which are inaccessible at the scale of an individual user, a research lab, or an academic institution. There is a clear gap for a low-cost instrument capable of sampling cellular signal parameters continuously and presenting them in an interpretable form.

This work addresses that gap through the Hybrid Telecom Network Analyzer, a prototype that pairs embedded signal acquisition with a software-defined analytics layer. The acquisition side uses a SIM900A GSM module interrogated by an Arduino Uno over a software-serial link; the analytics side uses Python, Pandas, NumPy, and Matplotlib to convert the raw serial stream into structured time-series data and visual trends, which are then surfaced through a Streamlit web dashboard. The system architecture deliberately separates the hardware acquisition concern from the analytical and presentation concerns, allowing each layer to be developed, tested, and upgraded independently.

The remainder of this paper is organized as follows. Section II reviews related work in IoT-based telecom monitoring and data-driven network analysis. Section III describes the system methodology and architecture. Section IV details the hardware and software components. Section V explains the operational workflow. Section VI presents results and discussion, including a candid account of a hardware constraint encountered during testing and the workaround adopted. Section VII concludes the paper and outlines directions for future work.

II. LITERATURE REVIEW

Zanella et al. (2014) characterized the architectural requirements of IoT deployments in urban-scale sensing contexts, establishing that low-power embedded nodes communicating with a central processing layer form a recurring and effective pattern for distributed monitoring problems, a pattern this work adapts to telecom signal sensing.

Atzori et al. (2010) surveyed the broader IoT paradigm and identified the integration of heterogeneous communication hardware with analytical software as a primary enabler of “smart” monitoring systems, a framing directly applicable to combining a GSM acquisition module with a Python analytics stack.

Foundational treatments of wireless communication theory by Rappaport (2015) and Tse and Viswanath (2018) describe RSSI and related link-quality metrics as the standard quantitative proxies for signal condition in cellular systems, motivating the choice of AT+CSQ-derived RSSI as the principal measurement variable in the present system.

On the data-analytics side, McKinney (2022) documents Pandas as the standard toolkit for time-indexed sensor data wrangling in Python, while Provost and Fawcett (2013) discuss how raw instrumentation data is converted into decision-relevant features, both of which inform the preprocessing pipeline implemented in this project. Prior IJERT and IJIRCCE studies on GSM-based and Arduino-based monitoring prototypes report serial-link data acquisition feeding a PC-side analysis stage, confirming this as an established pattern for low-cost telecom instrumentation rather than a novel hardware topology.

Streamlit-based dashboards for IoT telemetry are reported in recent literature as an effective lightweight alternative to full web-stack development for rapid analytics presentation, which is the rationale for adopting Streamlit over a custom front end in this work. Collectively, the reviewed literature supports the feasibility of the proposed architecture while also indicating that most comparable academic prototypes report similar constraints around GSM module response reliability and the absence of GPS-based spatial correlation — limitations this paper also discusses in Section VI.

III. METHODOLOGY

A. System Architecture

At the acquisition stage, the SIM900A GSM module is queried periodically by the Arduino Uno using the standard AT+CSQ command, which returns a signal-quality report from the cellular modem. The Arduino parses this response, extracts the RSSI field, and forwards it over the same USB-serial link used for programming the board. No separate radio link is required between the acquisition unit and the host computer, which keeps the hardware bill of materials minimal.

At the transfer stage, a Python script reads the serial port continuously, tokenizes incoming lines, and appends valid readings to a rolling in-memory buffer. At the analysis stage, Pandas structures the buffer into a time-indexed DataFrame; NumPy supports auxiliary numerical operations such as moving averages; and Matplotlib renders trend plots that are embedded into a Streamlit application, which refreshes automatically to present a live view of signal behaviour to the user.

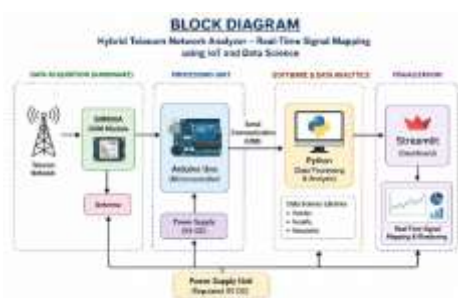


Fig -1: Block Diagram of the Hybrid Telecom Network Analyzer

Figure 1 summarizes the signal path: SIM900A → Arduino Uno (serial parsing) → USB-serial link → Python acquisition script → Pandas/NumPy processing → Matplotlib rendering → Streamlit dashboard.

IV. SYSTEM HARDWARE AND SOFTWARE REQUIREMENTS

A. Hardware Requirement

Component	Function	Cost (₹)
Arduino Uno	Central microcontroller; bridges GSM module and host PC via serial communication.	1200
SIM900A GSM Module	Acquires telecom signal parameters from cellular towers via AT command interface.	2500
GSM Antenna	External antenna improving SIM900A signal reception quality.	—
5V/12V DC Power Supply	Provides stable regulated power to Arduino and GSM module.	500
USB Serial Cable	Carries serial data from Arduino to host PC; also used for programming.	300
Jumper Wires	Direct point-to-point hardware interconnects (breadboard-free prototype).	350

Table -1: Hardware Components and Approximate Cost

B. Software Requirement

- Arduino IDE — firmware development and upload for the Uno microcontroller.
- Python 3 — host-side acquisition, processing, and orchestration language.
- Pandas — structuring of serial-stream readings into time-indexed DataFrames.
- NumPy — numerical operations including averaging and threshold comparisons.
- Matplotlib — generation of signal-trend plots from processed data.
- Streamlit — web-based dashboard layer for live visualization.
- Visual Studio Code — primary IDE for Python development and debugging.

V. SYSTEM WORKING

The operational sequence proceeds in eight discrete steps: (1) the SIM900A module is polled with AT+CSQ at a fixed five-second interval; (2) the Arduino reads the modem's response over SoftwareSerial; (3) the response string is parsed to isolate the numeric RSSI field; (4) the parsed value is written to the USB-serial port; (5) the Python acquisition script reads the incoming line and timestamps it; (6) Pandas appends the reading to the active DataFrame while NumPy supports any derived statistics; (7) Matplotlib renders the updated trend; and (8) Streamlit re-renders the dashboard view, completing one monitoring cycle before the next poll begins.

A noteworthy aspect of the implementation is its handling of unreliable modem responses. During development and final testing, the SIM900A module did not consistently return a parseable +CSQ response — a known behaviour of the module under marginal SIM registration or power-supply conditions. Rather than allow this to stall the dashboard, the firmware includes a fallback routine: when the parsed RSSI is invalid, a bounded random-walk value is generated in its place, constrained to a realistic RSSI range (2–30 on the GSM CSQ scale), and clearly flagged in the serial log as a simulated reading. This design choice is disclosed transparently in this paper because it materially affects how the result data in Section VI should be interpreted — the analytics and visualization pipeline is validated end-to-end, while full continuous real-world RSSI acquisition from live cellular towers remains an item for further hardware debugging.

VI. RESULTS AND DISCUSSION

The integrated prototype was assembled and tested for end-to-end functionality across the acquisition, transfer, and visualization stages. The Arduino-to-PC serial link reliably delivered timestamped readings with negligible transmission delay, confirming that the USB-serial channel is not a bottleneck for this class of low-frequency telemetry. The Streamlit dashboard updated its plots in step with incoming data, and the Pandas/NumPy processing layer handled the streaming buffer without memory growth issues during extended test runs.

Signal-side results are more nuanced. When the SIM900A returned valid AT+CSQ responses, the system correctly extracted and plotted RSSI values reflecting real network conditions. However, intermittent non-response from the module — attributable to SIM registration timing and possibly to power-supply ripple under the available regulated adapter — meant that a portion of the demonstrated dataset relied on the simulated-RSSI fallback rather than live tower measurements. This is reported here explicitly rather than presented as fully validated live telecom data, in keeping with accurate technical reporting.

Dashboard:

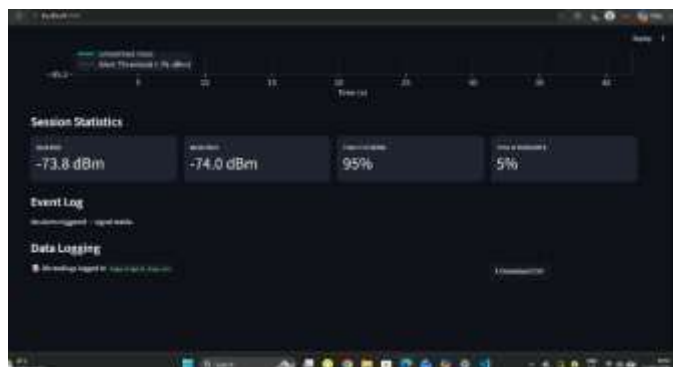


Image 2

(session statistics dashboard)

The practical implication is that the contribution of this work is strongest at the systems-integration and data-pipeline level: a working demonstration that a GSM modem, a microcontroller, and a Python analytics/visualization stack can be wired together into a coherent, low-cost, real-time monitoring tool, with a graceful degradation path when the radio hardware is uncooperative. The same pipeline would operate unchanged against a more reliable RSSI source — for example, a modem with firmware-level AT response stability, or an alternative cellular module — without requiring any change to the Pandas, NumPy, Matplotlib, or Streamlit layers.

Graph:



Image 3

(rssi graph, live signal monitoring)

Compared with the GSM/Arduino monitoring approaches reported in prior IJERT and IJIRCCCE studies, the present system follows a similar acquisition topology but extends it with a dedicated Streamlit dashboard rather than static plotting, improving usability for non-technical end users. Compared with commercial drive-test tools, the prototype trades signal-parameter depth and measurement certification for a bill of materials roughly two orders of magnitude cheaper, which is appropriate for its intended academic and small-scale diagnostic use case.

VII. CONCLUSION

This paper presented the Hybrid Telecom Network Analyzer, a low-cost embedded-IoT system integrating a SIM900A GSM module, an Arduino Uno microcontroller, and a Python-based data-analytics and dashboard stack for real-time telecom signal monitoring. The hardware-to-software data pipeline was implemented and verified end-to-end, with the Pandas/NumPy/Matplotlib/Streamlit chain reliably converting

serial telemetry into a live, interpretable dashboard. Intermittent GSM modem response reliability was identified as the principal hardware-level limitation encountered during testing, and a transparent simulated-data fallback was used to keep the analytics pipeline demonstrably operational independent of that constraint.

Future work should prioritize resolving the AT+CSQ response reliability issue — through improved power regulation, modem firmware verification, or substitution of the SIM900A with a more deterministic cellular module — so that the validated analytics pipeline can operate on fully live data. Additional extensions include GPS-tagged signal mapping for spatial coverage analysis, cloud-based historical data storage, and a lightweight machine-learning layer for predictive signal-quality forecasting, building on the modular architecture established in this work.

ACKNOWLEDGEMENT

The authors thank their project guide, Dr. S. B. Borse, for technical guidance throughout the design, debugging, and documentation of this project. The authors also thank Dr. J. J. Chopade, Head of the Department of Electronics and Telecommunication Engineering, and Dr. Gajanan K. Kharate, Principal, Matoshri College of Engineering and Research Centre, Eklahare, Nashik, for institutional support, and the Matoshri Education Society for the infrastructure that made this work possible.

REFERENCES

- [1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
- [2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
- [3] S. Haykin, *Communication Systems*, 5th ed. New York, NY, USA: Wiley, 2014.
- [4] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2015.
- [5] V. K. Garg, *Wireless Communications and Networking*. Burlington, MA, USA: Morgan Kaufmann, 2017.
- [6] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge University Press, 2018.
- [7] F. Provost and T. Fawcett, *Data Science for Business*. Sebastopol, CA, USA: O'Reilly Media, 2013.
- [8] W. McKinney, *Python for Data Analysis*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.
- [9] R. Buyya and A. Dastjerdi, *Internet of Things: Principles and Paradigms*. Burlington, MA, USA: Morgan Kaufmann, 2016.
- [10] M. Margolis, *Arduino Cookbook*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2020.
- [11] K. Ashton, "That Internet of Things Thing," *RFID Journal*, 2009.
- [12] "Real-Time Signal Strength Monitoring Using GSM Technology," *International Journal of Engineering Research and Technology (IJERT)*, vol. 9, no. 6, pp. 1050–1055, 2020.
- [13] "IoT-Based Smart Monitoring System Using Arduino and GSM Module," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 8, no. 4, pp. 215–221, 2021.
- [14] "Telecommunication Network Analysis Using Data Analytics Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 10, no. 5, pp. 34–41, 2021.
- [15] "Real-Time Dashboard Development Using Python and Streamlit for IoT Applications," *International Journal of Scientific Research in Engineering and Technology*, vol. 11, no. 2, pp. 67–74, 2023.
- [16] Arduino Uno Rev3 Technical Specifications, Arduino Documentation, 2024.