IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/47ICT/2026/4725326/
Vol. 2026 (13), Article ID 4725326

Design of a Resilient Android-Based Edge Client for ECG Telemetry Using Bluetooth Low Energy*

Fryderyk GAJDZIK, Michal KIMBAR and Pawel KROWICKI

Wrocław University of Science and Technology, Faculty of Mechanical Engineering, Wroclaw, Poland,

Correspondence should be addressed to: Fryderyk GAJDZIK, fryderykgajdzik@gmail.com

*Presented at the 47th IBIMA International Conference, 29-30 June 2026, Madrid, Spain

Abstract

Mobile biomedical telemetry systems require more than a user-facing application for displaying sensor values.
In practical acquisition scenarios, the mobile device must also act as an edge client that mediates between a constrained Bluetooth Low Energy sensor and a backend service while preserving data continuity under intermittent network conditions. This paper presents the design of a prototype Android-based edge client for session-oriented ECG telemetry acquisition. The aim is to organize the mobile layer as a resilient acquisition component responsible for BLE connectivity, acquisition control, packet parsing, real-time preview, local buffering, session registration, and deferred synchronization with a FastAPI edge backend. The proposed architecture combines a binary BLE frame format with sequence numbering, an offline-first local persistence strategy, CSV storage for raw session samples, and a Room database used as a session metadata registry and synchronization queue. The main contribution is an engineering design that separates the visualization path from the raw-data persistence path and treats the smartphone as a local edge buffer rather than only a display terminal. The work is limited to software architecture and functional verification of the prototype data flow. It does not include clinical validation, diagnostic performance evaluation, or certification as a medical device.

Keywords: mobile edge computing, Bluetooth Low Energy, ECG telemetry, biomedical data acquisition, Android, Room, FastAPI, offline-first synchronization, telemetry data integrity

Introduction

Portable biomedical data acquisition systems increasingly rely on sensor platforms connected to smartphones, which function both as local user interfaces and communication gateways. In such systems, the engineering challenge extends beyond displaying an ECG waveform. The mobile application must manage the acquisition session, receive data over Bluetooth Low Energy (BLE), detect transmission interruptions, persist data locally, and forward completed sessions for further processing. Previous studies on ECG monitoring systems emphasize data-flow architecture, transmission reliability, local processing, security, and measurement reproducibility (Serhani et al., 2020; Jain and Tiwari, 2014).

The Android application presented in this work is treated as a mobile edge client for session-oriented telemetry acquisition. It mediates between the BLE acquisition device and the edge backend while maintaining local data-

Cite this Article as: Fryderyk GAJDZIK, Michal KIMBAR and Pawel KROWICKI Vol. 2026 (13) "Design of a Resilient Android-Based Edge Client for ECG Telemetry Using Bluetooth Low Energy " Communications of International Proceedings, Vol. 2026 (13), Article ID 4725326, https://doi.org/10.5171/2026.4725326

flow stability. This approach is relevant in research prototypes, where electronics, firmware, and backend infrastructure may evolve iteratively, while acquisition should remain independent of continuous Wi-Fi, LTE, or server availability.

BLE is suitable for low-power biomedical sensing, but the BLE layer alone does not guarantee session integrity. Notifications may be lost, the device may disconnect, and the backend may be temporarily unavailable. Therefore, the application requires mechanisms for continuity control, buffering, and deferred synchronization. In the proposed prototype, these functions are implemented through a binary packet format with sequential numbering, CSV storage of completed sessions, and a local metadata registry based on Room. Room is an Android persistence framework that simplifies access to a local SQLite database and is used here to store session metadata, synchronization states, and upload-retry information.

The article has a technical and engineering-oriented character. It does not present clinical validation, diagnostic performance evaluation, or certification as a medical device. The scope is limited to the design and functional verification of the mobile and communication layers.

Technological background

Mobile ECG monitoring systems are usually designed as multilayer architectures consisting of a sensing device, a wireless communication layer, a mobile application, and a server-side or cloud infrastructure. Serhani et al. indicate that such systems must address not only acquisition but also processing, transmission, storage, security, and data-quality assessment (Serhani et al., 2020). Similar conclusions appear in reviews of cardiac-monitoring systems, where reliable data flow is treated as a prerequisite for further analysis (Jain and Tiwari, 2014).

BLE is frequently used in ECG sensor prototypes because it enables low-power operation and provides sufficient throughput for physiological signals sampled at moderate frequencies. However, packet loss and synchronization remain separate engineering problems that cannot be solved solely by increasing link throughput (Tipparaju et al., 2021).

The edge-computing concept supports shifting part of system responsibility closer to the data source. In mobile and IoT architectures, this reduces dependence on centralized infrastructure and improves resilience to temporary network failures (Mao et al., 2017; Dauda et al., 2024). In the analyzed prototype, the smartphone does not perform diagnostic inference, but acts as an edge layer by aggregating data, persisting sessions, maintaining a local registry, and synchronizing results after connectivity is restored.

On the Android side, Room provides a typed abstraction over SQLite for persistent session metadata (Android Developers, n.d.-a), while WorkManager supports background tasks constrained by network availability and retry policies (Android Developers, n.d.-b). Since ECG data may be sensitive, future versions should also consider mobile security standards such as OWASP MASVS, although the current implementation remains a technical prototype (OWASP Foundation, n.d.).

Aim and contributions

The aim of this work was to design and describe a prototype Android client acting as a resilient intermediary layer for session-oriented ECG telemetry acquisition. The client receives data blocks from a BLE device, controls acquisition start and stop, provides real-time visualization, stores complete sessions locally, and synchronizes them with a FastAPI backend.

The main contribution is the treatment of the mobile application as an edge client rather than a simple signal viewer. The proposed acquisition model is session-oriented: the primary unit of storage and synchronization is a completed measurement session. The architecture follows an offline-first strategy, in which local persistence is required for reliable data flow, whereas backend upload is a retryable stage. The design also separates the visualization path from the raw-data persistence path, preventing display-side filtering from modifying samples intended for later analysis.

A lightweight binary BLE protocol with sequential frame numbering enables detection of missing transmission blocks. The Room database serves simultaneously as a session registry, metadata store, and synchronization queue.

System architecture

The system consists of three layers: an ESP32-WROOM-32E acquisition device equipped with the MAX30003CTI+ module, an Android application, and a FastAPI edge backend (Fig 1). The acquisition hardware samples the signal, groups samples into binary frames, and exposes them through BLE notifications. The Android application receives frames, manages the session lifecycle, and persists data locally. The backend receives uploaded session files and registers associated metadata.

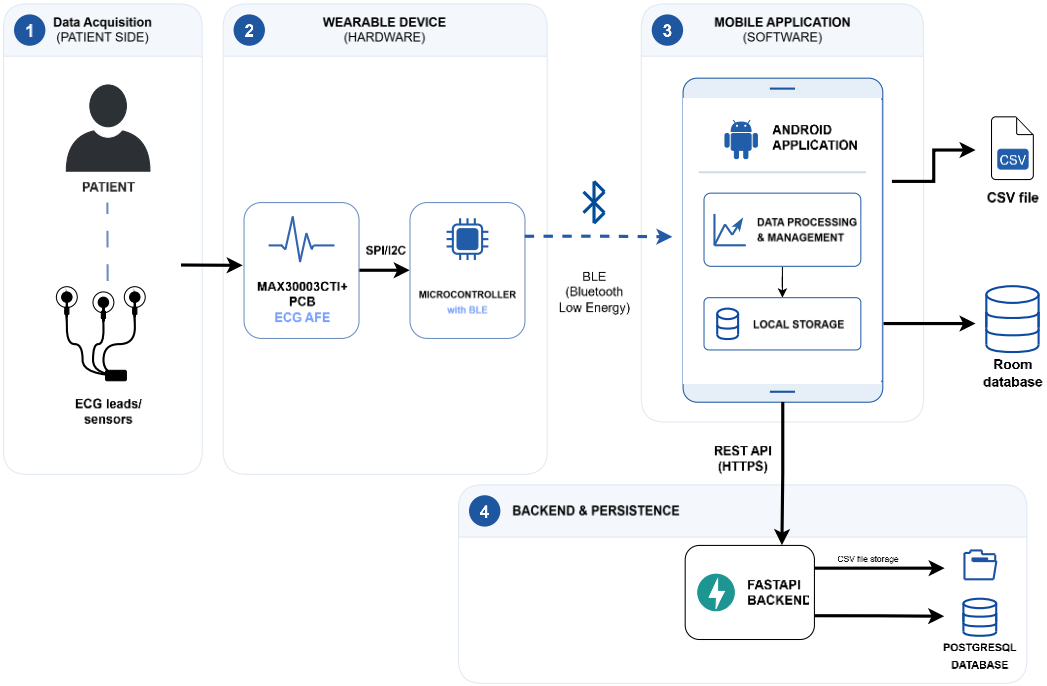


Fig 1. Data flow in the prototype system: BLE device, Android mobile client as the edge layer, and FastAPI backend.

The application architecture includes several functional responsibilities. The BLE connection manager handles service-filtered scanning, GATT connection, MTU negotiation, notification activation, high-priority connection mode, and RSSI monitoring. The acquisition controller sends control commands and resets counters and buffers at the beginning of each session.

Received data are processed by the packet parser, which validates packet length, interprets little-endian fields, extracts sequence numbers, and separates samples from the MAX30003CTI+ and auxiliary channels. The real-time visualization module uses Jetpack Compose and Canvas to display the waveform from a limited visualization buffer. Filtering may be applied only for display purposes and does not affect samples stored to file.

Persistent storage is handled by a local layer that creates a CSV file in the private application directory after session completion. In parallel, Room stores session metadata, including timestamps, sampling frequency, data source, sample count, lost-block count, file name, local path, synchronization status, and server-side identifier.

Synchronization is performed by a WorkManager-based worker that retries uploads when network connectivity is available. The FastAPI interface creates server-side sessions, uploads files, and finalizes transfers. This organization shifts the critical acquisition responsibility to the smartphone, so the backend does not need to remain continuously available during measurement.

BLE communication

Communication with the acquisition hardware is based on a custom GATT service organized similarly to the Nordic UART profile. A control characteristic is used for acquisition commands, while a streaming characteristic sends telemetry notifications.

The application scans for devices advertising the expected service UUID and prioritizes devices whose names begin with the EKG prefix. This does not provide full authentication, but reduces the risk of selecting an incompatible device during testing. After connection, the application discovers services, enables notifications, requests MTU 247, and sets a high BLE connection priority.

The firmware groups 50 samples into one frame, corresponding to approximately 100 ms of signal per notification at the assumed sampling configuration. This frame size reduces notification overhead while maintaining responsive visualization. Smaller frames would increase BLE transaction frequency, whereas larger frames would increase the impact of a single lost block.

Sequential numbering is essential for stream-integrity control. Correct packet length confirms only local format consistency, not whether complete blocks were lost between packets. Therefore, the application calculates the expected next sequence number and increments the lost-block counter when a discontinuity is detected. Packets with lengths other than the expected 206 bytes are rejected and counted as invalid.

Local persistence and session model

The fundamental acquisition unit is the session. A session begins with a start command and ends with a stop command. During acquisition, the application maintains two independent paths: a real-time visualization path with a limited buffer and a persistence path accumulating the complete dataset. Display filtering and scaling must not alter data intended for later analysis.

After acquisition stops, the application generates a CSV file in private storage. CSV is suitable at the prototype stage because it is readable and can easily be imported into analytical tools. However, it is not optimal for long recordings or advanced temporal metadata. Future versions should consider timestamps, binary storage, or a more structured format.

Session metadata are stored in Room as database records. The SessionEntity includes start and end times, sampling frequency, data source, sample count, lost-block count, CSV file name, local path, synchronization state, server-side identifier, remote path, and optional error information. This separation is intentional: CSV stores the sample sequence, while Room stores session state and synchronization information.

Room therefore acts both as a local session registry visible in the history interface and as a synchronization queue for sessions with locally available CSV files.

Synchronization with edge backend

Synchronization is performed after session completion rather than through continuous sample streaming. This separates BLE acquisition stability from IP-network availability. If the backend is reachable, the file can be uploaded immediately. If the upload fails, the CSV remains stored locally and the Room record is marked for retry.

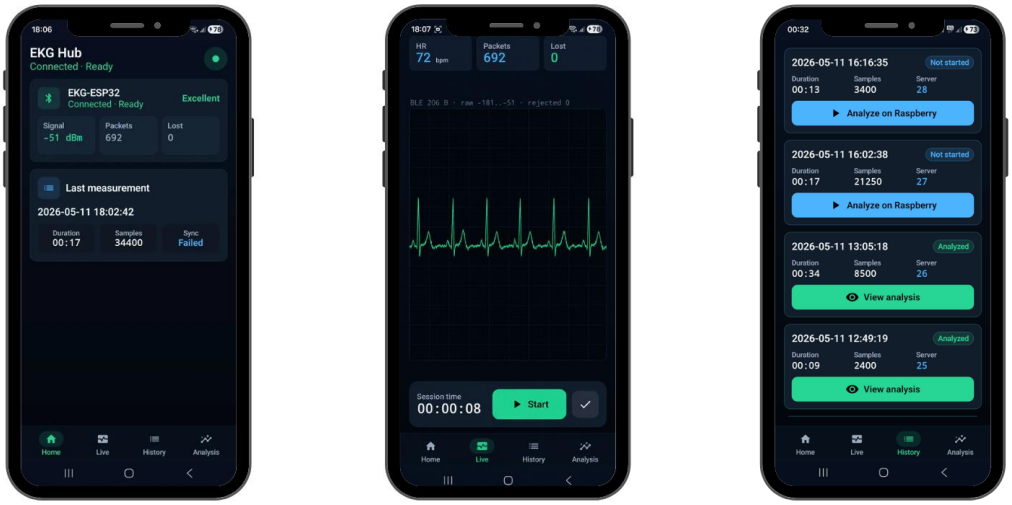
The FastAPI workflow begins with server-side session creation, followed by CSV upload with metadata such as file format, sample count, and duration. After transfer completion, the upload is finalized, and the session status may be queried.

API endpoints require an API-key header. The backend address and API key are provided through build configuration rather than hardcoded in the source code. HTTP is allowed only in local or test configurations; production deployment should require HTTPS.

Background synchronization is implemented through MeasurementSyncWorker. The task requires network availability and uses exponential backoff. After a successful upload, the application stores the server-side

identifier, remote path, and local synchronization status, such as UPLOADED or READY FOR ANALYSIS. If the UPLOAD fails, the status changes to UPLOAD FAILED, and the error is saved in syncError.

The worker can also recover orphaned CSV files by scanning local storage and recreating missing Room records. This allows locally persisted measurements to re-enter the synchronization queue after a partial application failure. The graphical user interface of the mobile application is presented in Fig 2.



(a) BLE connection (b) Signal preview (c) Session history

Fig 2. Selected views of the prototype mobile client: BLE connection, telemetry preview, and session registry.

Functional verification

Functional verification focused on data-flow correctness, synchronization resilience, and consistency with the adopted session-oriented model. The tests were not interpreted as clinical validation or diagnostic-performance assessment. Their purpose was to confirm that the mobile client correctly establishes BLE communication, controls acquisition, receives and interprets frames, persists sessions locally, and synchronizes them with the backend.

Verification was based on approximately 45 measurement sessions conducted in online and offline scenarios. The online scenario represented the standard workflow, in which the application uploaded the recorded file to the FastAPI backend after acquisition. The offline scenario evaluated behavior during network or backend unavailability, followed by a synchronization retry after connectivity restoration. In all sessions, the application behaved according to expectations.

During testing, the application detected the ECG BLE service, established GATT communication, discovered services, and enabled notifications. After the start command, the device transmitted data frames, while the application initialized a new session context, reset counters, and prepared buffers. After the stop command, transmission ended, and the application persisted the session.

Valid frames were parsed and forwarded simultaneously to the visualization path and the session-storage buffer. Invalid-length frames were rejected and counted. Sequential numbering enabled the detection of missing transmission blocks, confirming that continuity control was not limited to received-byte counting.

Local persistence tests confirmed that the application created a CSV file and a corresponding Room metadata record after acquisition. The record contained information required for later restoration and synchronization, including timestamps, source, sample count, lost-block count, file name, path, synchronization status, server-side identifier, and possible error message.

In online scenarios, a successful upload resulted in server-side file acceptance, assignment of a server session identifier, and a synchronization-state update in the application. In offline scenarios, the CSV file remained

stored locally, while the session was marked as pending or failed. After connectivity was restored, WorkManager retried the upload and updated the session status after a successful transfer.

These results confirm the correctness of the main prototype workflow: BLE communication, frame reception, continuity control, local persistence, and retryable synchronization with the edge backend. The verification of approximately 45 sessions indicates stable behavior in basic prototype scenarios but should be interpreted only as functional verification, not clinical validation or proof of medical readiness.

Discussion and limitations

The key architectural decision is the session-oriented acquisition model. Compared with continuous server streaming, this model better fits telemetry prototypes because it separates BLE-link data loss from temporary backend unavailability. Measurement continuity depends primarily on the local sensor–smartphone connection, while server synchronization can be retried later.

CSV is reasonable for a prototype because it enables rapid inspection and easy import into analytical tools. Its limitations include textual overhead and limited metadata representation. In the current implementation, the lack of explicit timestamps means that the file preserves sample order but not individual acquisition times. Longer technical studies should consider timestamps, binary storage, or column-oriented formats.

The separation between Room and CSV is justified by the division of responsibilities. CSV stores potentially large sample sequences, while Room stores compact, structured records such as session states, synchronization statuses, and file paths. Thus, the database serves as a registry and synchronization queue rather than a sample repository.

A separate visualization path reduces methodological risk. The displayed waveform may be filtered or scaled for readability, but the data written to CSV should remain faithful to the samples received from the device. Otherwise, later assessment of acquisition quality could be affected by hidden interface-level processing.

Conclusions

This work presented the design of a prototype Android mobile client for session-oriented ECG telemetry acquisition using BLE and an edge backend. The main conceptual shift is treating the application as a resilient intermediary layer that stabilizes data flow between a constrained BLE acquisition device and a backend server, rather than merely as a simple waveform viewer.

The architecture combines a binary frame format, sequential numbering, local CSV persistence, a Room-based metadata registry, and retryable synchronization through WorkManager. These mechanisms support an offline-first acquisition model in which completed sessions can be stored locally and synchronized when network infrastructure becomes available.

The work remains a technical description of architecture and functional prototype verification. It does not provide clinical results, validate diagnostic utility, or classify the system as a medical device. Future work should include timestamp handling, streaming-oriented persistence, long-term reliability tests, stronger security mechanisms, and formally documented functional procedures.

References

- Android Developers (n.d.-a), Accessing data using Room DAOs. [Online]. Android documentation. [Accessed 2026-05-13]. Available at: https://developer.android.com/training/data-storage/room/accessing-data.
- Android Developers (n.d.-b), Define work requests. [Online]. Android WorkManager documentation. [Accessed 2026-05-13]. Available at: https://developer.android.com/develop/background-work/background-tasks/persistent/getting-started/define-work.
- Dauda, A., Flauzac, O. and Nolot, F. (2024), 'A Survey on IoT Application Architectures', Sensors, vol. 24, no. 16, p. 5320, doi: 10.3390/s24165320.
- Jain, P. K. and Tiwari, A. K. (2014), 'Heart monitoring systems - A review', Computers in Biology and Medicine, vol. 54, pp. 1-13, doi: 10.1016/j.compbiomed.2014.08.014.

- Mao, Y., You, C., Zhang, J., Huang, K. and Letaief, K. B. (2017), 'A Survey on Mobile Edge Computing: The Communication Perspective', IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, doi: 10.1109/COMST.2017.2745201.
- OWASP Foundation (n.d.), Mobile Application Security Verification Standard. [Online]. OWASP MASVS documentation. [Accessed 2026-05-13]. Available at: https://mas.owasp.org/MASVS/.
- Serhani, M. A., El Kassabi, H. T., Ismail, H. and Nujum Navaz, A. (2020), 'ECG monitoring systems: Review, architecture, processes, and key challenges', Sensors, vol. 20, no. 6, p. 1796, doi: 10.3390/s20061796.
- Tipparaju, V. V., Mallires, K. R., Wang, D., Tsow, F. and Xian, X. (2021), 'Mitigation of Data Packet Loss in Bluetooth Low Energy-Based Wearable Healthcare Ecosystem', Biosensors, vol. 11, no. 10, p. 350, doi: 10.3390/bios11100350.