PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

SmartClassroom

Ricardo Bastos Leta Vieira

Projeto Final de Graduação

Centro Técnico Científico - CTC

Departamento de Informática

Curso de Graduação em Ciência da Computação

Rio de Janeiro, Dezembro de 2025

Ricardo Bastos Leta Vieira

SmartClassroom

Relatório de Projeto Final, apresentado ao programa de Graduação do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Markus Endler

Coorientador: Alexandre Malheiros Meslin

Rio de Janeiro, Dezembro de 2025

Abstract

Bastos Leta Vieira, Ricardo. Endler, Markus. SmartClassroom. Rio de Janeiro, 2025. 34 pages. Undergraduate Thesis II. Pontifical Catholic University of Rio de Janeiro.

SmartClassroom is a modular system that automates student attendance tracking using the ContextNet architecture. The solution combines a mobile app, developed in Flutter/Dart, with ESP32 devices installed in the classroom. The ESP32 devices, powered by photovoltaic cells, transmit the room's corresponding ID via BLE. Students' phones receive this location, acting as mobile nodes, and send the data to a processing node that records attendance. The devices are housed in custom cases designed for production via 3D printing.

Keywords: BLE (Bluetooth Low Energy), ESP32, ContextNet, Mobile application, Photovoltaic power, IoT in education, 3D printing

Resumo

Bastos Leta Vieira, Ricardo. Endler, Markus. SmartClassroom. Rio de Janeiro, 2025. 34 páginas. Trabalho de conclusão de curso, projeto final II. Pontifícia Universidade Católica do Rio de Janeiro.

O SmartClassroom é um sistema modular que automatiza o registro de presença de alunos, utilizando a arquitetura ContextNet. A solução combina um aplicativo mobile, desenvolvido em Flutter/Dart, com dispositivos ESP32 instalados em sala de aula. Os dispositivos ESP32, alimentados por células fotovoltaicas, transmitem o ID correspondente à sala que estão via BLE, os celulares dos alunos recebem essa localização, atuando como nós móveis e enviam os dados a um nó de processamento, que registra a presença. Os dispositivos são acomodados em cases personalizados, planejados para produção via impressão 3D.

Palavras-chave: BLE (Bluetooth Low Energy), ESP32, ContextNet, Aplicativo mobile, Energia fotovoltaica, IoT na educação, Impressão 3D

Contents

1 Introduction

During my exchange program in Japan, I realized that classroom attendance control was carried out through a system in which students had to access a course page and enter a code provided by the professor each class. While this method was functional, it still required manual action by the students and an administrative step within class time, which, in many cases, led to delays, forgetfulness, or even fraud. This experience led me to reflect on how technology could be used to make this process more automated, reliable, and fluid, benefiting both professors and students.

Later, while taking the course INF1304 — Distribution and Concurrency — I was introduced to ContextNet [1], a scalable middleware system aimed at distributed sensor networks and efficient communication between mobile and fixed nodes. With one of the course assignments, I quickly saw a concrete opportunity to apply this knowledge to a real problem I had personally experienced: classroom presence registration. This gave rise to the idea of developing a ContextNet-based system for the course that would automate this process in an intelligent and transparent way and refine it to make it work in a real environment, specifically in PUC classrooms.

The central problem addressed by this work of mine is the current dependence on traditional and semi-automated attendance control methods, which require class time, repetitive actions from professors and students, and are often subject to inconsistencies or fraud. Even in institutions with greater technological infrastructure, it is common for the frequency control process to still depend on human actions, which makes it vulnerable to distraction, error, and bad faith.

I consider this problem relevant precisely because of the absence of solutions for the educational environment that are truly automatic, accessible, and easily integrable with other technologies. This also presented an opportunity for a practical implementation of ContextNet on a smaller scale. The proposal of this work is to develop a modular and energetically autonomous system using ESP32s with communication via Bluetooth Low Energy (BLE), powered by photovoltaic cells, and integrated with a cross-platform mobile application. This is a solution designed for the reality of Brazilian institutions but has the potential for adaptation to different educational contexts.

2 Current Situation

2.1 Glossary

This project utilizes the concepts and terminology used by the ContextNet system, based on the following terms:

- **Mobile Node (MN):** A possibly mobile node that sends sensor data and measurement context information (e.g., its location) to the ContextNet's Group Definer.
- **Processing Node (PN):** A central processing node (i.e., running in a backend) that can communicate with Mobile Nodes using their specific identifiers or with groups of MNs, as established by the Group Definer service.
- **Group Definer (GD):** The central ContextNet service that defines context-based groups of nodes and thus determines the groups to which each Mobile Node belongs based on the context received from the respective MN.

2.2 Existing Solutions

The predominant methods for attendance control in educational institutions suffer from significant limitations that compromise their efficiency and reliability. The traditional manual roll call, for example, consumes a considerable amount of class time that could be dedicated to teaching, in addition to being inherently prone to registration errors. Even approaches seeking some modernization, such as using unique codes per class, scanning QR codes, or using ID cards (RFID/NFC), do not eliminate the need for manual intervention. These solutions require students to perform a specific action (type a code, scan an image, tap a card) and for the professor to administer the process, which leaves the door open to forgetfulness, delays, technical difficulties, and, crucially, fraud, such as one student registering the presence of an absent colleague. The dependence on human action makes these systems vulnerable to distraction, error, and bad faith.

While the exploration of technologies like BLE and the use of smartphones as mobile hubs (such as M-Hub) represent a conceptual advance for data collection in IoT, their implementations often run into significant platform limitations that hinder their practical application in heterogeneous environments. Middleware platforms like M-Hub2, developed natively for Android (using Kotlin), demonstrate technical feasibility within a single ecosystem. SmartClassroom aims to expand this approach by integrating M-Hub2 as the native Android implementation within a Flutter package, using method channels. This strategy not only consolidates the existing Android solution but also paves the way for future native iOS implementations within the same Flutter package, ensuring that the client application functions transparently and accessibly on the vast majority of mobile devices used by students, regardless of the operating system.

Still, while the investigation of sensor-based solutions advances in presence detection, many of these approaches are limited to the aggregate count of individuals, a functionality that, in isolation, proves inadequate for the requirements of nominal attendance control, as required in educational institutions. Illustratively, Perra et al. [2] developed a system to monitor indoor occupancy using an array of low-cost Infrared (IR) sensors strategically positioned at classroom entrances. The methodology employs a real-time pattern recognition algorithm to analyze thermal data, detect the direction of passage, and subsequently transmit this information via a Z-Wave network, enabling people counting in the location. However, while effective for quantification, such a system suffers from the intrinsic limitation of not offering individual identification, a capability that is critical in the academic context. In contrast, SmartClassroom seeks to overcome precisely this deficiency by focusing on mechanisms capable of nominally identifying the individuals present in the room, thereby generating data with the granularity indispensable for the specific verification of each student's presence.

Other solutions explore BLE for identification, such as the "Mobile Attendance System" which uses the NRF51822 module for contactless attendance marking. In this system, students use a mobile application to detect BLE signals and actively register their presence, with data integrity ensured by linking to the mobile device's unique ID. This approach, however, differs substantially from SmartClassroom, where the fixed ESP32 devices in the classrooms transmit the room ID via BLE, and the students' cell phones, acting as Mobile Nodes, receive this location information. This information is then sent to a processing node, eliminating the need for active student intervention and centralizing the control logic. Similarly, the project by Kaczmarek et al [3]. employs BLE for indoor tracking, focusing on analyzing human traffic in tourist facilities through BLE beacons and Raspberry Pi 4B receivers. While it uses a "Winner Takes All" principle to mitigate signal overlap problems in dense environments, its application and architecture diverge from SmartClassroom, which is specifically designed for the automated registration of student attendance with ESP32 nodes

integrated into ContextNet. The choice of the ESP32 for SmartClassroom is also distinguished from hardware alternatives like the Raspberry Pi and the NRF51822, as the ESP32 is lower in cost than both while offering greater processing capacity compared to the NRF51822.
As mentioned, the core technology proposed in this work involves the use of ESP32 microcontrollers, programmed to act as BLE location transmitters, and a mobile application developed in Flutter/Dart that uses M-Hub to receive this data. The idea is for the ESP32s, installed in the rooms and powered by photovoltaic energy, to transmit their location IDs via BLE. These ESP32 were prototyped by a colleague, Lucas [4]. The students' cell phones, upon receiving these signals, will act as Mobile Nodes, sending the location data to a centralized processing node in ContextNet for automatic attendance registration. This approach aims to overcome the limitations of current methods by offering automation, using accessible technologies, and proposing energetic autonomy.

3 Project Objectives

3.1 Description of the Proposed Solution

This work proposes the development of SmartClassroom, a Mobile Internet of Things application based on the ContextNet middleware and a project developed by my group for the course INF1304 - DISTRIBUTION AND CONCURRENCY, to automate the registration of student attendance in the classroom. The solution is comprised of two main components: software and hardware, with a focus on modularity and integration with other systems.
For the software component, a cross-platform mobile application has been developed in Flutter/Dart. This application, through a package whose Android implementation will utilize M-Hub2 [5, 6] modules (specifically for BLE discovery and connection, and communication with ContextNet's Group Definer and Processing Node), will allow students' cell phones to receive the room ID via Bluetooth Low Energy (BLE) from the ESP32 devices permanently installed in the classrooms. Students' cell phones will act as Mobile Nodes of the ContextNet network, using the context update and message sending functionalities to report their location (the room ID received from the ESP32) to the Group Definer.
The Group Definer will be responsible for defining the groups to which each Mobile

Node (student's cell phone) belongs, based on the received location information. Subsequently, the Processing Node will be responsible for processing this group information, identifying student entry and exit, and automatically registering their attendance.
In the hardware component, the ESP32s will be designed to operate with their own power supply, preferably by means of photovoltaic cells that capture ambient light. They will also be housed in physical cases produced by 3D printing, ensuring security and ease of installation in a school environment.

3.2 Specific Objectives

Based on the problem definition and the state-of-the-art in attendance control systems, the specific objectives of the project include:

- Refactor the project developed in INF1304 to allow communication between modules via BLE, eliminating the dependency on log files.
- Develop a Flutter/Dart package for cell phones that, in its Android implementation, uses the methods related to BLE (device discovery and, possibly, connection) and the methods related to ContextNet (connection to the PN and the Group Definer, context update, message receiving and sending) from M-Hub2 to periodically receive location identifiers (room IDs) via BLE from the ESP32s and transmit this information to the ContextNet as a Mobile Node.
- Adapt the behavior of the ESP32 devices to act as fixed transmitters of location identifiers (room IDs) via BLE.
- Implement the logic for identification and attendance registration in the Processing Node, based on the received information.
- Integrate an autonomous power solution with a photovoltaic cell into the ESP32.
- Design and manufacture physical cases for the devices using 3D printing.

Desired System Scope

The project encompasses the complete development of a distributed system for automated attendance control, including:

- A functional mobile application that receives BLE data from the ESP32s and acts as a Mobile Node, transmitting the student's location to ContextNet.
- Fixed ESP32 devices capable of transmitting location identifiers via BLE and operating with energetic autonomy.

- A backend system (Processing Node) capable of processing, interpreting, and registering attendance data.
- Physical components ready for installation and use in school environments.

Users/Developers and Scenarios to be Supported

The SmartClassroom system is intended for:

- **Educational institutions** that wish to automate attendance control with minimal manual intervention.
- **Professors and coordinators** who need reliable and organized data on student frequency.
- **Students**, whose presence will be registered discreetly and automatically.
- **Researchers and developers** interested in expanding ContextNet and integrating sensors and BLE networks with educational solutions.

Differences or Advancements Compared to Existing Solutions

Compared to traditional methods (such as roll calls, QR codes, RFID cards), SmartClassroom offers significant advancements:

- **Total automation** of the process, without the need for manual action from the student or professor.
- Use of **modern and accessible technologies** (Flutter, ESP32, BLE) with low cost and high scalability.
- **Autonomous power** via photovoltaic cell, eliminating external power sources.
- **Modular architecture** compatible with IoT networks based on ContextNet.
- Ability to monitor multiple students with a single device, reducing the amount of hardware needed.

Items to be Implemented

- Functional **Flutter/Dart** package for mobile devices, which utilizes specific **M-Hub2** modules (for BLE discovery/connection and communication with ContextNet) in the Android implementation, for receiving location identifiers from the ESP32s and transmitting student location data to ContextNet.
- Updated source code for the ContextNet modules (*group definer*, *processing node*, *mobile node*).

- **ESP32** devices configured as fixed transmitters of location identifiers via **BLE**, with integrated **solar power**.
- Physical device cases produced via **3D printer**.
- Documented tests in a laboratory and a real environment (classroom).

4 Study and Development

This section details the trajectory of the project during the course of the final development. It covers the preliminary knowledge possessed prior to the start of the work, the theoretical and technical studies conducted, the prototypes developed, and the methodology applied throughout the research. Additionally, it presents a detailed comparison between the planned and realised schedules, highlighting the impact of architectural changes on the timeline.

4.1 Preliminary Studies

The development of *SmartClassroom* required a combination of distributed systems concepts and mobile development skills. Prior to the start of this work, the experience with the target technological environment was mixed. A solid grasp of the foundational concepts of the *ContextNet* middleware architecture had already been established during the *Distribution and Concurrency* (INF1304) course. This prior contact was essential for identifying the opportunity to automate attendance tracking using the Group Definer and Processing Node structures.
However, a significant gap existed regarding the mobile development framework chosen for the project, specifically *Flutter* and the *Dart* language. Consequently, a portion of the initial effort was dedicated to understanding the framework's state management, widget lifecycle, and its integration with native platform features (Android and iOS) to access Bluetooth Low Energy (BLE) hardware.
Additionally, a comprehensive study of the original *Mobile Hub* application was conducted. Although initially excluded from the plan due to its native Android-only nature, which limited cross-platform accessibility, the subsequent architectural shift proved its utility. The decision to integrate it as the native Android layer behind a Flutter interface not only satisfied the immediate technical requirements but also leveraged the Flutter framework to open possibilities for future iOS integration, thereby broadening the potential scope of the system.

4.2 Conceptual and Technology Studies

To bridge the knowledge gap and validate architectural decisions, several specific studies were conducted. The primary focus was on cross-platform development using Flutter and Dart, with a specific emphasis on implementing BLE broadcasting and scanning. This involved understanding the mechanism of bridging Dart code with native Android layers through Method Channels to effectively integrate with the existing M-Hub implementation.

Concurrently, an investigation into the capabilities of the ESP32 microcontroller was necessary to ensure it could handle multiple simultaneous BLE identifiers. A critical component of this research involved determining the feasibility of running a Java Virtual Machine (JVM) directly on the ESP32. The initial proposal envisaged the ESP32 acting as a full *Mobile Node*, which would require it to run Java code to integrate seamlessly with ContextNet.

Although the hardware implementation was the focus of a collaborative partner project, done by Lucas Manoel Martins de Souza [4], it was also necessary to study the constraints of the ESP32 to ensure that the software architecture, particularly related to BLE, remained compatible with the ESP32 firmware and the energy budget of a solar-powered indoor system.

4.3 Tests and Prototypes for Learning and Demonstration

To consolidate the concepts studied and demonstrate technical feasibility, prototypes and refactoring activities were carried out. The legacy logic from the INF1304 course project underwent significant refactoring. The *Group Definer* was modified to cease writing static files for group assignments, while the *Processing Node* was altered to utilize message passing for presence verification instead of relying on log files.

The role of the *Mobile Node* simulator proved critical throughout the development phases, though its application differed significantly between architectures. In the initial design, where the ESP32 was tasked with collecting student IDs, the simulator was configured to imitate this static device behavior, passing multiple student identifiers to the backend. This allowed for the validation of the *Processing Node* and *Group Definer* logic prior to hardware availability. Conversely, in the revised architecture, the simulator was repurposed to represent a single student acting as a *MobileHub*. This configuration was instrumental in testing the delivery of groupcast

messages to the actual smartphones running the final Flutter application, ensuring the communication loop was functional.
In addition to software simulations, two comprehensive bench tests were conducted to validate the integrated system. The first test focused on the hardware-software interface, specifically testing the BLE scanning functionalities with the ESP32 devices acting as beacons. The second bench test targeted the distributed communication aspects, validating the successful transmission and reception of groupcast messages involving the ESP32 nodes within the ContextNet environment.
Furthermore, a proof-of-concept application was implemented in Flutter/Dart to test the broadcasting of BLE signals. This prototype served as the foundation for the final student application, validating the core communication mechanisms required for the system.

4.4 Method

The project followed an iterative development method, characterised by a significant architectural pivot driven by technical findings. Initially, the architecture proposed that the ESP32 devices would act as *ContextNet Mobile Nodes*. This design required the ESP32 to run a JVM to communicate directly with the gateway. However, technical studies revealed that running a robust JVM on an ESP32 is an uncommon and complex task, with available solutions for BLE in Java being either commercially licensed or unstable.
This finding necessitated a fundamental re-evaluation of the architecture. The decision was made to invert the roles, transforming the ESP32 into a simple BLE beacon that broadcasts the Room ID, thus allowing the use of lightweight, standard firmware. Consequently, the students' smartphones assumed the role of the Mobile Node, receiving the Room ID via BLE and reporting this context to the Group Definer. This change simplified the hardware requirements and shifted the complexity to the smartphone, where computational resources and development tools are more abundant. To realise this new role effectively, the methodology incorporated the adaptation of the existing native Android *Mobile-Hub* middleware into a Flutter plugin. This strategic choice allowed the system to leverage the proven robustness of the native implementation while utilising Flutter's capabilities for future cross-platform expansion.

4.5 Schedule

The following subsections present the schedule as originally planned in the proposal and the schedule that was actually realised.

4.5.1 Schedule in Final Project 1

The following shows in order the activities done in the previous semester

1. Refactoring of attendance tracking logic (07/04/2025 – 28/04/2025).
   (a) Modify Group Definer to cease writing file-based group logs.
   (b) Modify Processing Node to verify presence via message passing rather than file logs.
   (c) Modify Mobile Node to simulate a static ESP32 handling multiple IDs.
2. Drafting of the Project Proposal (28/04/2025 – 05/05/2025).
3. Study of Flutter/Dart technologies for BLE broadcasting (05/05/2025 – 12/05/2025).
4. Editing of the final Project Proposal (05/05/2025 – 12/05/2025).
5. Implementation of Flutter/Dart app for BLE ID broadcasting (12/05/2025 – 26/05/2025).
6. Study and implementation of BLE student ID acquisition on the Mobile Node (26/05/2025 – 02/06/2025).
7. Study and implementation of the Mobile Node on an ESP32 (02/05/2025 – 09/06/2025).
8. Drafting of the Final Project 1 Report (09/06/2025 – 16/06/2025).
9. Study of photovoltaic cell power supply for ESP32 (09/06/2025 – 16/06/2025).
10. Editing and submission of the Final Project 1 Report (16/06/2025 – 27/06/2025).
11. Refactoring of attendance tracking logic (07/04/2025 – 28/04/2025).
    (a) Modify Group Definer to cease writing file-based group logs.
    (b) Modify Processing Node to verify presence via message passing rather than file logs.
    (c) Modify Mobile Node to simulate a static ESP32.
12. Drafting of the Project Proposal (28/04/2025 – 05/05/2025).
13. Study of Flutter/Dart technologies for BLE broadcasting (05/05/2025 – 12/05/2025).
14. Implementation of Flutter/Dart app for BLE ID broadcasting (12/05/2025 – 02/06/2025).
15. Editing of the final Project Proposal (26/05/2025 – 30/05/2025).
16. Study of technologies for acquiring student IDs via BLE on the Mobile Node

(02/06/2025 – 09/06/2025).

17. Study of technologies to implement the Mobile Node on an ESP32 (02/05/2025 – 09/06/2025).
18. Re-evaluation of project architecture due to difficulties in implementing Mobile Node on ESP32 (09/05/2025 – 13/06/2025).
19. Drafting of the Final Project 1 Report (13/06/2025 – 23/06/2025).
20. Editing and submission of the Final Project 1 Report (23/06/2025 – 27/06/2025).

4.5.2 Planned schedule of pending tasks

1. **Implementation of Method Channels (06/08/2025 – 20/08/2025):** Development of the method channels in Flutter to enable communication between the Dart interface and the native *Mobile Hub* Android code.
2. **Application Implementation (20/08/2025 – 03/09/2025):** Implementation of the full Flutter/Dart application using the created method channels to handle BLE scanning and context reporting.
3. **ESP32 Firmware Implementation (03/09/2025 – 10/09/2025):** Finalising the code for the ESP32 to act as a BLE beacon transmitting the room ID.
4. **Case Design and Fabrication (10/09/2025 – 17/09/2025):** Design and 3D printing of the physical case for the ESP32
5. **Hardware Assembly (17/09/2025 – 24/09/2025):** Soldering the ESP32 to the photovoltaic cell and final assembly of the device.
6. **Bench Testing (24/09/2025 – 01/10/2025):** Execution of comprehensive bench tests to validate the integration between the ESP32 beacon, the mobile app, and the backend.
7. **Deployment and Field Testing (01/10/2025 – 08/10/2025):** Deployment of the system in a real classroom environment and execution of field tests to verify robustness.
8. **Final Report and Presentation (08/10/2025 – 29/10/2025):** Writing the Final Project II Report and preparing the final presentation.

4.5.3 Realised schedule of Final Project II

1. **Implementation of Method Channels and Mobile Hub Integration (06/08/2025 – 15/10/2025):** This activity consumed the majority of the project's timeline. Developing the Flutter plugin to communicate with the native *Mobile Hub* proved

significantly more complex than anticipated. Specifically, correctly initialising the `MobileHubService` and establishing a reliable context transmission stream from the Android layer to the Dart environment required extensive debugging and refactoring, causing this task to extend well beyond the original plan.

2. **Hardware and Firmware Development (06/08/2025 – 15/10/2025):** Executed in parallel by the collaborative partner. While the software integration faced delays, the ESP32 firmware was finalised, and the physical cases were designed, printed, and assembled with the photovoltaic cells, ensuring the hardware was ready for integration once the software hurdles were overcome.
3. **Application Implementation (15/10/2025 – 29/10/2025):** Once the Method Channels were functional, the development of the UI and the remaining application logic proceeded rapidly. The delay in the previous step required this phase to be compressed, but the core functionalities of BLE scanning and context reporting were successfully implemented.
4. **Bench Testing (29/10/2025 – 12/11/2025):** With both software and hardware ready, the integration tests were conducted intensively over a shorter period. The system was successfully deployed in the test environment, validating the architecture. .

4.5.4 Deviations and Current Status

The execution of the project presented deviations from the initial plan, primarily driven by the architectural pivot previously described and the technical challenges of cross-platform integration. The infeasibility of running a robust JVM on the ESP32 necessitated a re-evaluation period in May and June, shifting the focus towards a smartphone-centric Mobile Node architecture.

A significant portion of the development effort following the conclusion of Final Project I was dedicated to the conversion of the *Mobile-Hub* into a Flutter plugin. This process was critical to enable the new architecture but proved to be technically demanding. The vast majority of the time between the end of Final Project I and the present was consumed by the complexities of integrating the native Android service with the Flutter environment. Specifically, substantial effort was required to successfully initialise the `MobileHubService` and ensure the mobile device could reliably transmit its context information to the gateway. These integration challenges were the primary factor in the schedule adjustments for the latter phases of the

project.
Consequently, the planned deployment of the system in real classroom environments for extensive field testing could not be realised within the available timeframe. While bench testing confirmed the system's functionality and architectural viability, the time lost to resolving the plugin integration issues precluded the opportunity to gather usage data from live academic settings as originally scheduled.

5 System design

This section presents the architectural and structural design of the *SmartClassroom* system. It details the evolution of the system's architecture, the software components developed to enable the mobile integration with ContextNet, the physical deployment of the nodes, and the technological stack selected for implementation.

5.1 Architecture

The architecture of the *SmartClassroom* system is built upon the ContextNet middleware, leveraging its hierarchical structure of mobile and processing nodes to manage distributed context data. Throughout the development process, the system architecture underwent a significant transformation to address technical constraints related to the embedded hardware capabilities.

5.1.1 Initial Architecture

The preliminary design, illustrated in Figure 1, envisaged a topology where the ESP32 microcontrollers acted directly as the ContextNet Mobile Nodes. in this configuration, the ESP32 was responsible for scanning the environment for students' Bluetooth Low Energy (BLE) signals. Upon detection, the device would process this data and transmit the context information, specifically the presence of student IDs within the room, directly to the ContextNet Gateway via an Internet connection.
This approach placed the burden of the *Mobile Hub* logic, including the MRUDP [7] communication stack and the local processing of ContextNet messages, entirely on the microcontroller. As discussed in the previous section, this architecture required a Java Virtual Machine (or a C++ version of the Mobile Node Simulator) to run on the ESP32, which proved infeasible for the project's scope and stability requirements.

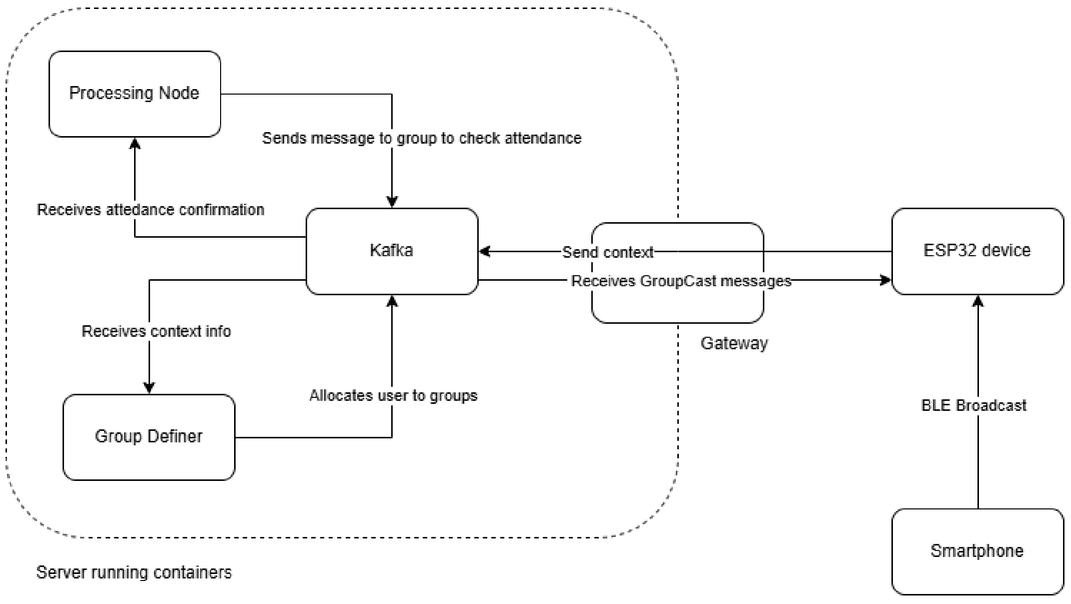


Figure 1: Diagram of the initial architecture of SmartClassroom

5.1.2 Final Architecture

The final architecture, presented in Figure 2, inverts the roles of the edge devices to leverage the superior processing power and connectivity of modern smartphones. In this design, the ESP32 functions as a static BLE beacon. Its sole responsibility is to continuously broadcast a unique Universal Unique Identifier (UUID), which the group definer is able to map to the corresponding, specific classroom the ESP32 inhabits.

The students' smartphones assume the role of the ContextNet Mobile Nodes. Each student's device runs the *SmartClassroom* mobile application, which utilizes the *Mobile Hub* plugin to scan for surrounding BLE beacons. When the application detects a beacon from an ESP32, it extracts the room identifier (the ESP32's UUID) and encapsulates this information into a context message, using ContextNet's API 3.0. This message is then transmitted via a UDP/MRUDP connection to the ContextNet Gateway. The Gateway forwards this data through the GroupReportTopic kafka topic to the *Group Definer* and *Processing Node*, which are responsible for aggregating the data, determining the student's location based on the strongest signal received, and logging the attendance record.

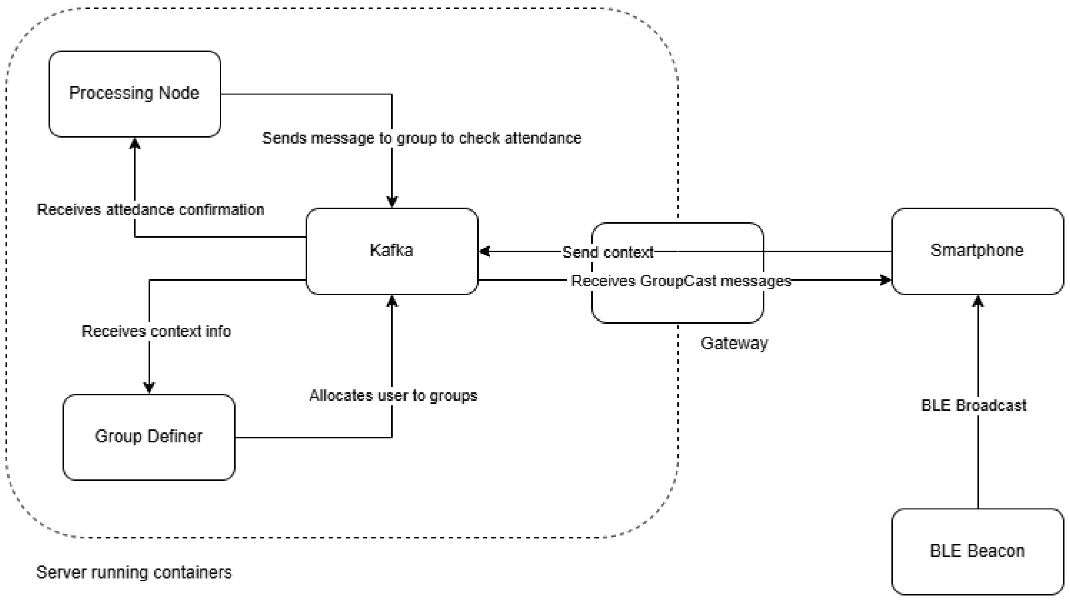


Figure 2: Diagram of the final architecture of SmartClassroom

5.1.3 Final Architecture

The final architecture, presented in Figure 2, inverts the roles of the edge devices to leverage the superior processing power and connectivity of modern smartphones. In this validated design, the ESP32 functions as a static BLE beacon.

Beacon Data Generation and Transmission The ESP32 firmware is configured to operate in a generic broadcaster mode. The device continuously transmits advertising packets containing a specific payload. The critical piece of information transmitted is a 128-bit Universally Unique Identifier (UUID). This UUID is pre-configured to uniquely map to the specific physical classroom where the device is installed. The beacon does not require a paired connection; it simply broadcasts its identity to any listening device in range.
The students' smartphones assume the role of the ContextNet Mobile Nodes. Each student's device runs the *SmartClassroom* mobile application, which utilizes the *Mobile Hub* plugin to passively scan for these surrounding BLE beacons. When the application detects a beacon, it captures the UUID along with the Received Signal Strength Indicator (RSSI).

Handling Multiple Beacons In real-world scenarios, such as a university corridor or adjacent classrooms with thin walls, a Mobile Node might simultaneously receive signals from beacons in multiple rooms. To handle this, the mobile application does not filter these signals locally based on a simple threshold, which could be unreliable. Instead, it encapsulates the data of all detected beacon UUIDs and their corresponding signal strengths into the context message. This aggregate message is then transmitted via the smartphone's internet connection (Wi-Fi or 4G/5G) to the ContextNet Gateway. The Gateway forwards this data to the *Processing Node*. It is the Processing Node's responsibility to execute a ”Winner Takes All” logic: by comparing the RSSI values of all reported beacons, it identifies the strongest signal to accurately resolve the student's location to the specific room they are physically occupying, effectively filtering out noise from neighboring classes.

5.2 Software Design and Component Structure

The core software innovation in this project lies in the integration of the native Android *Mobile Hub* functionality into a cross-platform Flutter environment. This was achieved through the implementation of a layered plugin architecture that bridges the Dart application layer with the native Kotlin service layer.

5.2.1 Platform Channels and Native Interoperability

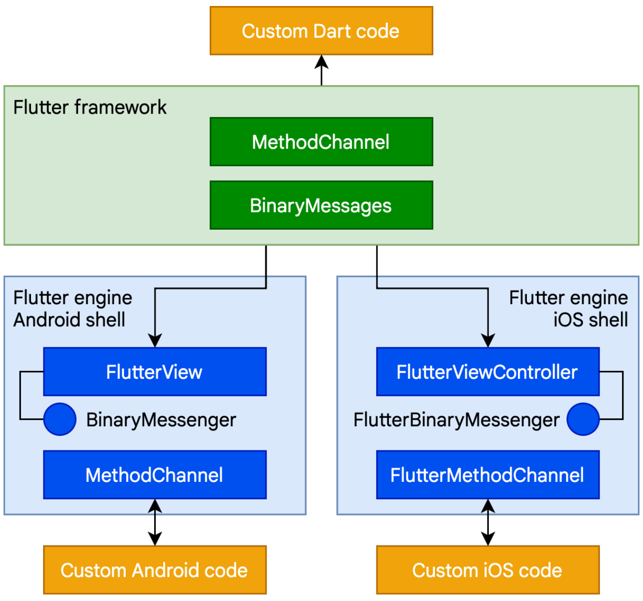


Figure 3: Architecture of Platform Channels and the communication between Dart and Native code. Source: Adapted from Flutter Documentation [8].

Flutter applications are built using the Dart programming language, which compiles to native machine code but operates within its own execution environment. To access platform-specific APIs and hardware features that are not exposed directly to Dart—such as Bluetooth Low Energy stacks, background services, or specific OS sensors—Flutter relies on a flexible message-passing mechanism known as *Platform Channels*.

A Platform Channel acts as a bridge that creates a persistent communication line between the Dart client (the Flutter UI) and the host platform (Android or iOS). It uses a standard method call interface where the Dart side sends a message containing the method name and arguments. This message is serialized into a binary format, transferred to the host platform asynchronously, and then deserialized. The host platform's native code (Kotlin or Java for Android, Swift or Objective-C for iOS) receives the call, executes the requested native function, and returns the result back to Dart through the same channel. This architecture allows Flutter to leverage existing, robust native libraries without the need to rewrite complex system-level logic in Dart.

5.2.2 The Flutter-Native Bridge

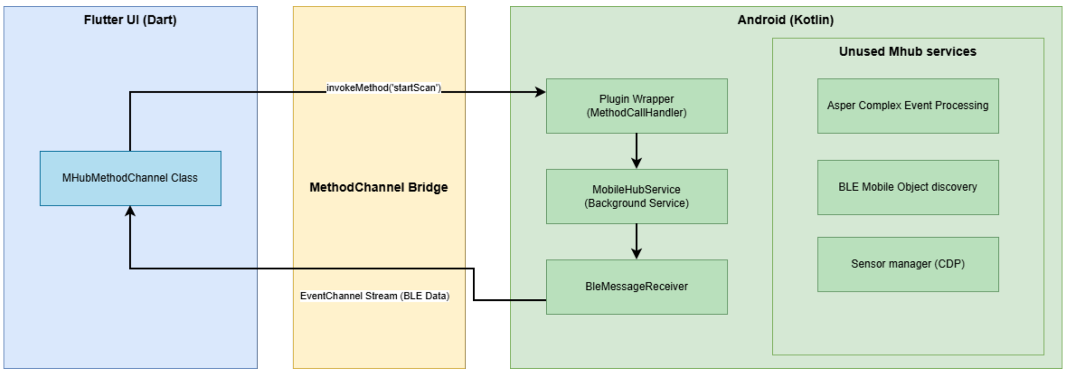

Flutter UI (Dart)
MHubMethodChannel Class
invokeMethod('startScan')
MethodChannel Bridge
EventChannel Stream (BLE Data)
Android (Kotlin)
Plugin Wrapper (MethodCallHandler)
MobileHubService (Background Service)
BleMessageReceiver
Unused Mhub services
Asper Complex Event Processing
BLE Mobile Object discovery
Sensor manager (CDP)

Figure 4: Architecture of the Flutter-Native bridge. The `MHubMethodChannel` (Dart) invokes methods on the native Android layer via a MethodChannel, while the `MobileHubService` streams BLE data back to the UI via an EventChannel.

The client-side software is structured around the `MHub` Flutter package, which utilizes the Platform Channel pattern described above. On the Dart side, the MHub-MethodChannel class implements the platform interface, creating a specific communication channel named `mhub_package`. This class is responsible for serialising commands—such as initialising the service or sending a context message—and invoking the corresponding methods on the native platform. Conversely, it sets up event handlers to listen for incoming data streams from the native side, allowing the

Flutter UI to react to real-time events like sensor readings or processing results.
On the Android side, the architecture is anchored by the `MobileHubService`, a background service written in Kotlin. This service manages the lifecycle of the *Mobile Hub* instance, ensuring that context monitoring continues even when the application is not in the foreground. The `Plugin` class acts as the intermediary, implementing the `MethodCallHandler` interface to intercept incoming Dart calls and route them to the appropriate service methods.

5.2.3 Context and Sensor Management

Within the native Android layer, the `BleMessageReceiver` class is responsible for handling the scanning of device UUIDs and passing them on to the onBleDataReceived event stream. This differs from the original *Mobile Hub* application, where the `BleWPAN` class was tasked with discovering and connecting to Mobile Objects, although both implementations utilize the same `RxBleClient` for their respective tasks. Regarding the core logic, the functioning of the `MobileHub` is largely preserved, ensuring that most native features remain operational. However, method channels were not implemented for all existing features due to the specific scope of this project, leaving these integrations as interesting subjects for future work. The communication with the external ContextNet Gateway is managed by the `MrudpWLAN` component, which implements the Mobile Reliable UDP protocol, ensuring efficient data delivery over the wireless network.

5.3 Deployment View

The physical deployment of the *SmartClassroom* system involves three distinct environments. The first is the **Cloud/Server Environment**, where the ContextNet infrastructure resides. This includes the Gateway, which acts as the entry point for all mobile nodes, the Group Definer service which manages logical groupings of users based on context, and the Processing Node which executes the business logic for attendance tracking. These components typically run in containerised environments (e.g., Docker) for scalability.
The second environment is the **Classroom Environment**, which contains the static hardware nodes. The ESP32 devices are physically mounted in the classrooms, encased in 3D-printed protective shells. A critical aspect of this deployment is the power supply; the devices are designed to be powered by photovoltaic cells to en-

sure energy autonomy, eliminating the need for complex wiring. This practical implementation of energy harvesting is not merely theoretical but has been realized in a parallel work by Lucas Manuel. His work successfully implemented and deployed a functional prototype of this solar-powered enclosure, demonstrating the real-world viability of this power solution for the *SmartClassroom* project.

The third environment is the **Mobile Environment**. This consists of the students' personal smartphones. The deployment here relies on the user installing the *SmartClassroom* application. The application operates as a background service on the device, creating a dynamic mobile grid that interacts with the fixed infrastructure of the classroom as students move through the campus.

5.4 Technologies Used

The implementation of the system relied on a specific set of technologies chosen for their interoperability and performance. **ContextNet** served as the foundational middleware, providing the protocols (MRUDP) and architectural pattern (SDDL) necessary for large-scale mobile context management.

For the mobile application, **Flutter** and **Dart** were selected to provide a modern, reactive user interface and to facilitate potential future deployment on iOS devices. The native Android integration required the use of **Kotlin**, which provided the necessary access to low-level system services such as background processing and Bluetooth hardware management. The communication between these layers relied on Flutter's **Platform Channels**.

Finally, the hardware logic for the beacons was implemented using **C++** within the ESP-IDF framework, chosen for its efficiency and fine-grained control over the ESP32's power management and BLE stack, essential for the energy-harvesting requirements of the physical device.

This section describes the implementation process of the *SmartClassroom* system, detailing the technical challenges encountered during the development of the mobile application and the testing strategies employed to validate the system's functionality.

5.5 Implementation Challenges

The translation of the system design into a functional application presented several significant technical hurdles, particularly regarding the integration of the legacy native Android code with the new Flutter environment.

5.5.1 Plugin Architecture and Module Unification

A primary challenge in the early stages of development was the structural adaptation of the original *Mobile Hub* source code. The legacy project was composed of multiple distinct Android modules, a structure that proved incompatible with the standard Flutter plugin architecture, which typically expects a unified Android library module. To enable the *Method Channel* communication required for the Flutter bridge, it was necessary to unify these separate modules into a singular Kotlin module. This process involved complex conflict resolution regarding dependencies and the merging of multiple `AndroidManifest.xml` files to ensure all services and permissions were correctly declared and accessible to the Flutter host application.

5.5.2 ContextNet Compatibility and Context Updates

A critical issue arose during the implementation of the context update mechanism. The initial integration utilised the main branch of the *Mobile Hub* repository, which utilizes version 2.0 of the ContextNet API. However, the Processing Node, Group Definer and Gateway used in this project use the newer API version, 3.0, introducing fundamental incompatibilities with MobileHub due to way messages are structured in the two versions. In 2.0 messages use the Record class whereas 3.0 uses the custom SwapData class, meaning the gateway forbid any Record messages from arriving at their destination due to wrong formatting. This impasse was resolved through consultation with a previous developer, Tatiana Reimer, who identified that the necessary compatibility with the current Gateway infrastructure was maintained in a separate development branch. Migrating the project to use this specific branch resolved the communication discrepancies.
Furthermore, the documentation regarding context publication within the ContextNet ecosystem proved to be ambiguous. Previous works suggested that publishing context messages required strictly targeting the `GroupReportTopic` Kafka topic, which the *Group Definer* listens to. However, practical implementation revealed that the defining characteristic of a context message is the invocation of the `setContext` method during the creation of the message object. While failing to set the topic explicitly triggers a warning, it does not prevent the message from being treated as context data, provided the subclass is correctly instantiated. Clarifying this be-

haviour was essential for establishing reliable communication between the mobile node and the gateway.

5.6 Testing Strategy

The verification of the system was conducted through a combination of formal bench tests and ad-hoc development testing. Given the complexity of the legacy codebase and the learning curve associated with the Flutter/Dart environment, the testing strategy prioritised functional validation over comprehensive unit test coverage for the new components.

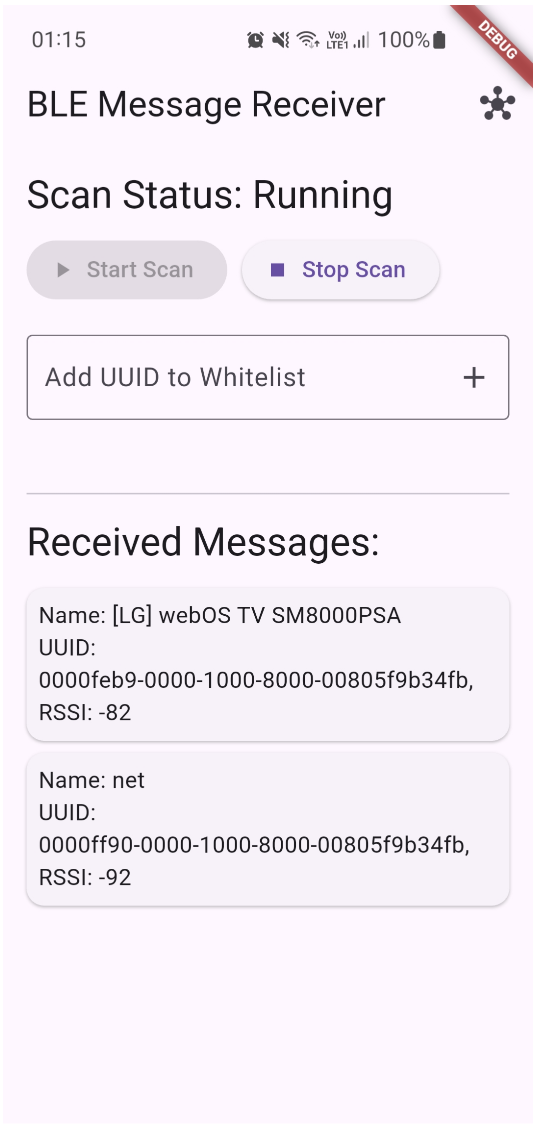
5.6.1 Development Testing

During the active development phase, testing was primarily manual and ad-hoc. New unit tests were introduced where feasible, but the focus remained on ensuring that the integrated components could successfully initialize and communicate. This approach was partly justified by the fact that the legacy native Android code already possessed a suite of unit tests. These tests were automatically executed by the Gradle wrapper during the Flutter build process, ensuring that the underlying core logic remained robust and regression-free. Consequently, the ad-hoc testing efforts were concentrated on the new Flutter integration layer and the bridging logic, where unit test coverage was naturally lacking. The unfamiliarity with the hybrid development environment (Flutter interacting with deep native Android services) meant that rapid prototyping and manual verification were more effective in identifying integration issues than writing extensive test suites for the legacy logic.

5.6.2 Bench Tests

To formally validate the system's core requirements, three distinct bench tests were executed:

1. **BLE Scanning Verification:** This test focused on the hardware-software interface. It involved configuring the ESP32 devices to broadcast specific UUIDs and verifying that the *SmartClassroom* application could correctly scan, filter, and identify these beacons amidst other Bluetooth signals.



BLE Message Receiver

Scan Status: Running

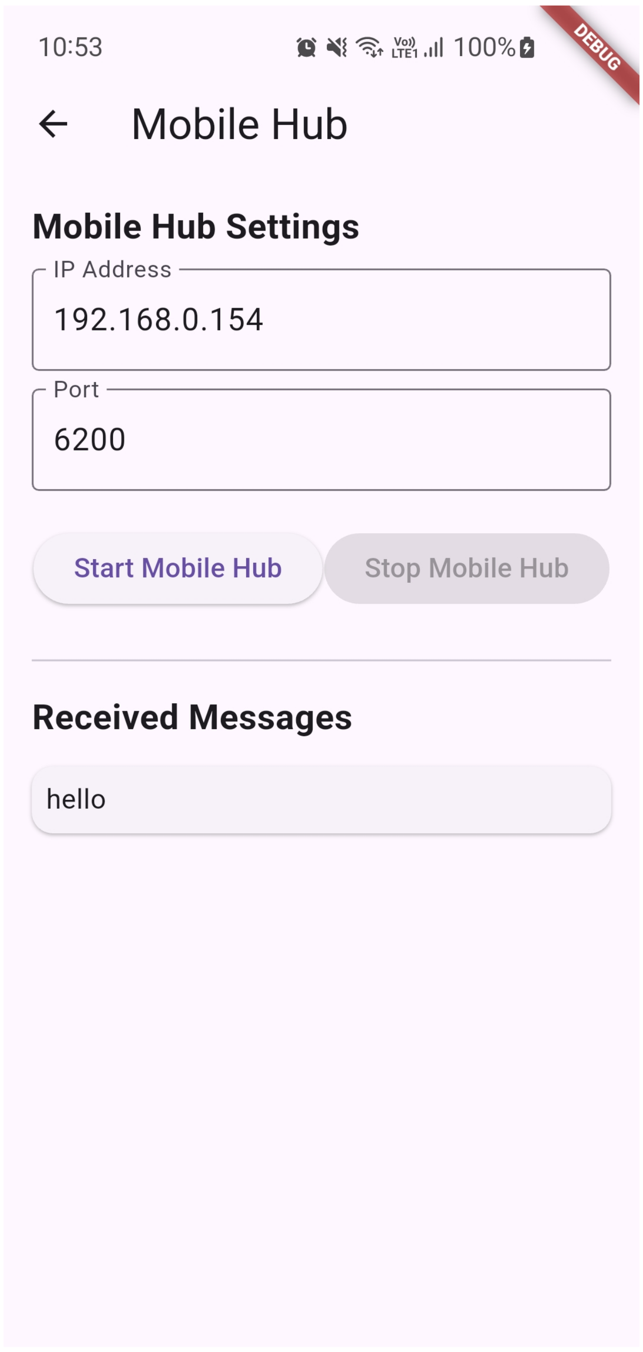
Received Messages:

Name: [LG] webOS TV SM8000PSA
UUID:
0000feb9-0000-1000-8000-00805f9b34fb,
RSSI: -82

Name: net
UUID:
0000ff90-0000-1000-8000-00805f9b34fb,
RSSI: -92

Figure 5: Screenshot of the SmartClassroom application performing a BLE scan, displaying detected UUIDs and RSSI signal strength.

2. **Groupcast Messaging Test:** This test validated the distributed communication capabilities. It verified that the system could successfully send groupcast messages from the gateway to the correct subset of smartphones (simulated as Mobile Nodes) based on the context information (Room ID) received from the ESP32 beacons.


10:53 100%

DEBUG

Mobile Hub

Mobile Hub Settings

IP Address: 192.168.0.154

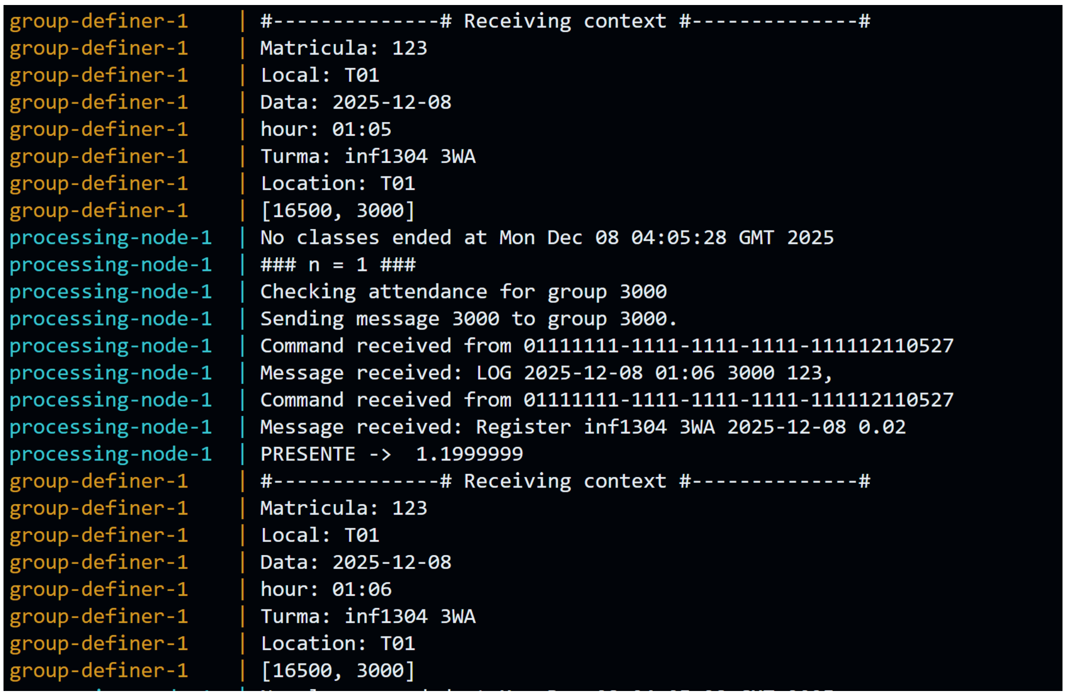
Port: 6200

Received Messages

hello

Figure 6: Configuration screen of the Mobile Hub plugin within the application, showing the connection parameters and the message reception interface.

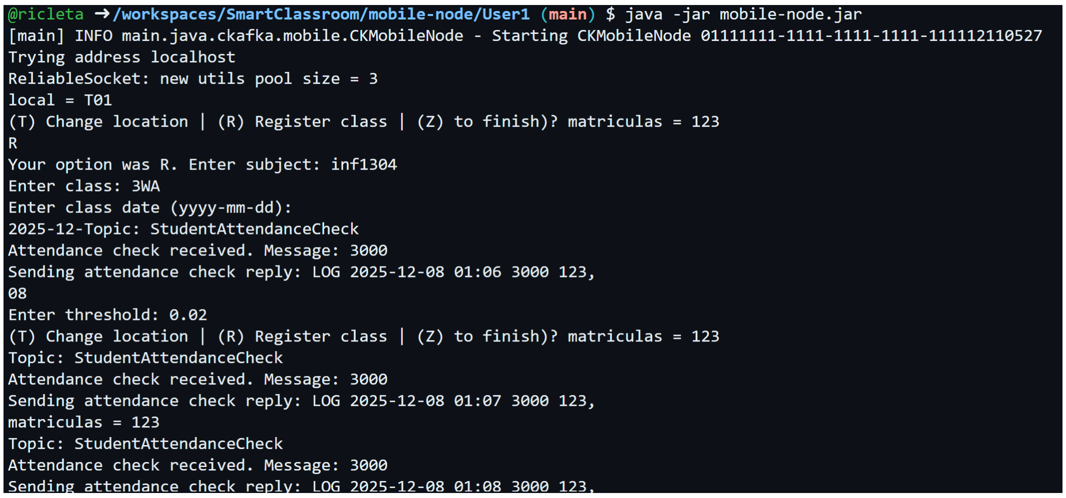
3. **Attendance Logic Validation:** A specific bench test was conducted using the Mobile Node simulator to verify the backend attendance logic. In this scenario, the simulator was configured to emulate a student's device entering and leaving a geofenced area to simulate a classroom environment. As shown in Figure 8, the simulator logs demonstrate the successful execution of the attendance check protocol without the need for physical hardware.



```
group-definer-1     | #--------------# Receiving context #--------------#
group-definer-1     | Matricula: 123
group-definer-1     | Local: T01
group-definer-1     | Data: 2025-12-08
group-definer-1     | hour: 01:05
group-definer-1     | Turma: inf1304 3WA
group-definer-1     | Location: T01
group-definer-1     | [16500, 3000]
processing-node-1   | No classes ended at Mon Dec 08 04:05:28 GMT 2025
processing-node-1   | ### n = 1 ###
processing-node-1   | Checking attendance for group 3000
processing-node-1   | Sending message 3000 to group 3000.
processing-node-1   | Command received from 01111111-1111-1111-1111-111112110527
processing-node-1   | Message received: LOG 2025-12-08 01:06 3000 123,
processing-node-1   | Command received from 01111111-1111-1111-1111-111112110527
processing-node-1   | Message received: Register inf1304 3WA 2025-12-08 0.02
processing-node-1   | PRESENTE ->  1.1999999
group-definer-1     | #--------------# Receiving context #--------------#
group-definer-1     | Matricula: 123
group-definer-1     | Local: T01
group-definer-1     | Data: 2025-12-08
group-definer-1     | hour: 01:06
group-definer-1     | Turma: inf1304 3WA
group-definer-1     | Location: T01
group-definer-1     | [16500, 3000]
```

Figure 7: Terminal output showing the interaction between the Group Definer (top) receiving context data and the Processing Node (bottom) executing attendance logic.

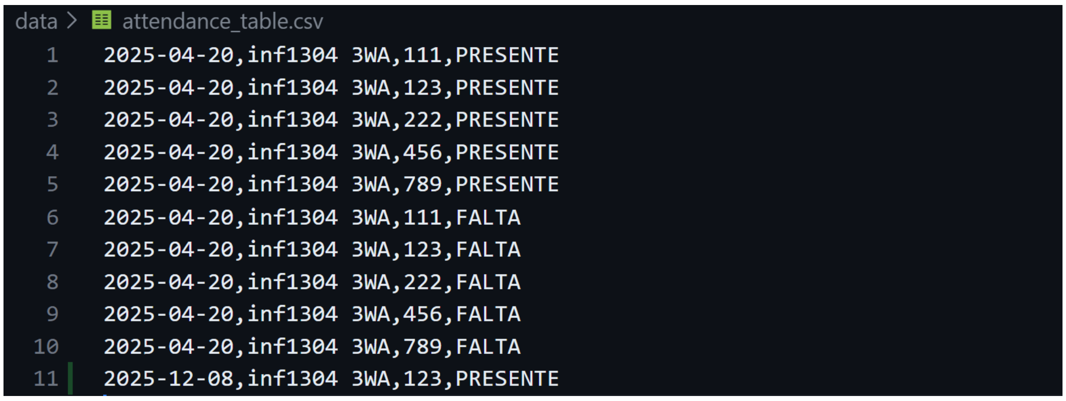
The actions performed by the simulator triggered immediate responses from the backend infrastructure. The Group Definer received the context data to locate the user, while the Processing Node executed the logic to calculate the duration of the stay. Figure 7 illustrates this interaction, where the top section shows the Group Definer receiving context updates and the bottom section displays the Processing Node verifying the attendance rules.



```
@ricleta →/workspaces/SmartClassroom/mobile-node/User1 (main) $ java -jar mobile-node.jar
[main] INFO main.java.ckafka.mobile.CKMobileNode - Starting CKMobileNode 01111111-1111-1111-1111-111112110527
Trying address localhost
ReliableSocket: new utils pool size = 3
local = T01
(T) Change location | (R) Register class | (Z) to finish)? matriculas = 123
R
Your option was R. Enter subject: inf1304
Enter class: 3WA
Enter class date (yyyy-mm-dd):
2025-12-Topic: StudentAttendanceCheck
Attendance check received. Message: 3000
Sending attendance check reply: LOG 2025-12-08 01:06 3000 123,
08
Enter threshold: 0.02
(T) Change location | (R) Register class | (Z) to finish)? matriculas = 123
Topic: StudentAttendanceCheck
Attendance check received. Message: 3000
Sending attendance check reply: LOG 2025-12-08 01:07 3000 123,
matriculas = 123
Topic: StudentAttendanceCheck
Attendance check received. Message: 3000
Sending attendance check reply: LOG 2025-12-08 01:08 3000 123,
```

Figure 8: Execution log of the Mobile Node simulator used to validate the attendance check protocol without physical hardware.

This process culminated in the generation of a persistent record of the student's activity. The system determined whether the student met the presence threshold and logged the result accordingly. Figure 9 presents the final output file generated by the Processing Node, which lists the date, the specific subject, and the final status of either Presence or Absence for the monitored student IDs.



```
data > attendance_table.csv
2025-04-20,inf1304 3WA,111,PRESENTE
2025-04-20,inf1304 3WA,123,PRESENTE
2025-04-20,inf1304 3WA,222,PRESENTE
2025-04-20,inf1304 3WA,456,PRESENTE
2025-04-20,inf1304 3WA,789,PRESENTE
2025-04-20,inf1304 3WA,111,FALTA
2025-04-20,inf1304 3WA,123,FALTA
2025-04-20,inf1304 3WA,222,FALTA
2025-04-20,inf1304 3WA,456,FALTA
2025-04-20,inf1304 3WA,789,FALTA
2025-12-08,inf1304 3WA,123,PRESENTE
```

Figure 9: The final CSV log generated by the Processing Node, recording the date, subject, and attendance status (Presence/Absence) for each student.

These tests confirmed the successful operation of the critical path: detecting a location via BLE, reporting it to the ContextNet infrastructure, and correctly processing attendance data.

6 Conclusion

This work presented the design and implementation of *SmartClassroom*, a distributed system aimed at automating student attendance tracking within university environments using the ContextNet middleware. The project addressed the limitations of traditional manual attendance methods by proposing a seamless, technology-driven solution that leverages Bluetooth Low Energy (BLE) beacons and mobile edge computing.

The development process was characterised by a significant architectural evolution. Initial investigations revealed the practical infeasibility of deploying complex Mobile Node logic, specifically a Java Virtual Machine, directly onto ESP32 microcontrollers. This technical constraint necessitated a strategic pivot to shifting the Mobile Node role to the students' smartphones. This decision not only simplified the hardware requirements for the classroom infrastructure but also unlocked superior processing capabilities and facilitated a more scalable and maintainable system architecture.

The primary contribution of this project is the successful integration of the native Android *Mobile Hub* service into a cross-platform Flutter environment. By encapsulating the robust, native communication and sensing logic within a Flutter plugin, the system achieves a high degree of interoperability. This architecture allows the application to benefit from the extensive ecosystem of the ContextNet middleware (such as the MRUDP protocol for reliable mobile communication) while offering a modern, reactive user interface that is prepared for future expansion to other mobile operating systems.

Despite the technical hurdles encountered, particularly regarding the unification of legacy Android modules and the alignment of ContextNet API versions, a functional prototype was successfully developed and validated. Bench tests confirmed the system's ability to reliably detect classroom beacons via BLE, manage context updates, and execute groupcast messaging logic. The collaborative effort to develop autonomous, solar-powered hardware further strengthens the proposal's viability for real-world deployment, minimising maintenance overhead for educational institutions.

Reflecting on the development journey, a different approach would be adopted if the project were to start today. The initial phase dedicated to forcing the ESP32 to act as a full Mobile Node consumed valuable time and resources on a path that ultimately proved unviable due to hardware and software limitations. With the benefit of hindsight, the most effective strategy would have been to bypass this experimental phase entirely and proceed directly to the conversion of the *Mobile Hub* application into a Flutter plugin. Prioritising this integration from the outset would have allowed for a more extensive period of testing and refinement of the mobile application, potentially enabling the field tests that could not be realised within the current timeframe.

6.1 Limitations

While the core functionality of the system has been verified in controlled environments, certain limitations remain. The primary constraint was the inability to conduct extensive field testing in a live classroom setting with a large volume of concurrent users. Consequently, the system's behavior under high-density network conditions (typical of a crowded lecture hall) remains a theoretical projection based on bench tests. Additionally, the current implementation of the Flutter plugin, while functional for the scope of this project, does not yet expose the full suite of *Mobile Hub* features, such as dynamic service reconfiguration and complex event processing rule management directly from the Dart layer.

6.2 Future Work

Several avenues for future research and development have been identified to further enhance the *SmartClassroom* system:

- **iOS Integration:** The adoption of Flutter provides a solid foundation for cross-platform support. Future work should focus on implementing the iOS-native counterpart of the *Mobile Hub* services, enabling the application to serve the entire student body regardless of their device ecosystem.
- **Security and Privacy Enhancements:** As the system handles sensitive student data, implementing robust encryption for BLE broadcasts and securing the communication channels between the mobile application and the gateway is paramount. Investigating methods to prevent ”attendance spoofing” (e.g., students checking in for absent friends) would also be valuable.
- **Large-Scale Field Pilots:** Conducting a semester-long pilot program in multiple classrooms would provide critical data on system reliability, battery consumption on student devices, and the long-term durability of the solar-powered

beacons.

- **Advanced Analytics:** Integrating an analytics dashboard for professors and administrators could provide insights beyond simple attendance, such as student punctuality patterns and classroom occupancy rates over time.

In conclusion, *SmartClassroom* demonstrates the potential of combining IoT hardware with modern mobile frameworks to modernise educational administrative processes. The system lays a robust groundwork for a scalable, automated attendance solution that is both technologically sound and practically adaptable.

References

[1] M. Endler and F. S. e Silva, "Past, present and future of the contextnet iomt middleware," *Open Journal of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 7–23, 2018. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil.

[2] C. Perra, A. Kumar, M. Losito, P. Pirino, M. Moradpour, and G. Gatto, "Monitoring indoor people presence in buildings using low-cost infrared sensor array in doorways," *Sensors*, vol. 21, no. 12, 2021.

[3] M. Kaczmarek, J. Ruminski, and A. Bujnowski, "Accuracy analysis of the rssi ble sensortag signal for indoor localization purposes," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1413–1416, 2016.

[4] EnQyMo, "Beacon: Esp32 ble beacon firmware." `https://github.com/AirQyMo/Beacon`, 2025. Accessed: November 18, 2025.

[5] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e. Silva, "The mobile hub concept: Enabling applications for the internet of mobile things," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 123–128, 2015.

[6] T. Reimer, L. E. T. Ríos, M. Cavalcanti, L. Lucchesi, and M. Endler, "Design and implementation of a flexible mobile edge middleware for multi-protocol wireless connectivity," in *2024 IEEE Conference on Pervasive and Intelligent Computing (PICom)*, pp. 40–46, 2024.

[7] L. D. Silva, M. Endler, and M. Roriz, "MR-UDP: Yet another Reliable User Datagram Protocol, now for Mobile Nodes," Tech. Rep. 06/2013, Departamento

de Informática, PUC-Rio, 2013.

[8] Google, “Architectural overview,” 2025. Accessed: 2025-12-08.

[9] G. Thiagarajan, M. Saran, K. Sneha, and J. Janani, “A smart attendance management system using nrf51822 ble module and mobile application,” *International Research Journal on Advanced Engineering Hub (IRJAEH)*, vol. 3, pp. 1343–1347, 04 2025.

[10] R. Belka, R. S. Deniziak, G. Łukawski, and P. Pięta, “Ble-based indoor tracking system with overlapping-resistant iot solution for tourism applications,” *Sensors*, vol. 21, no. 2, 2021.

[11] R. B. L. Vieira, “Smartclassroom, an contextnet application.” `https://github.com/ricleta/SmartClassroom`, 2025. Acessed: December 02, 2025.

[12] P. Spachos and A. Mackey, “Energy efficiency and accuracy of solar powered ble beacons,” *Computer Communications*, vol. 119, pp. 94–100, 2018.

[13] M. Ghaemifar *et al.*, “Bluetooth low energy for indoor positioning: Challenges, algorithms and datasets,” *Automation in Construction*, vol. 177, p. 106316, 2025.

[14] Y. Ranjan *et al.*, “Radar-iot: An open-source, interoperable, and extensible iot gateway framework for health research,” *Sensors*, vol. 24, no. 14, p. 4614, 2024.