# Project Proposal: ProximityNet

A Wearable Proximity Sensing System for Social Analytics

Luca Tedeschini
luca.tedeschini3@studio.unibo.it

Luca Lizzi
luca.lizzi3@studio.unibo.it

July 2025

**Abstract**

This document outlines a proposal for an Internet of Things (IoT) project named ProximityNet. ProximityNet is a prototype for a wearable device designed to detect proximity to other compatible devices. Each device pairs with a user's smartphone, which serves as a gateway to relay collected proximity data to a centralized server via its internet connection. The primary goal is to anonymously log interactions between participants at controlled events, such as conferences or social gatherings. This data can then be aggregated and analyzed to generate insights into social dynamics, crowd behavior, and network structures.

## 1 Introduction

In social settings, understanding the dynamics of human interaction is a complex challenge. ProximityNet aims to address this by providing a technological solution for capturing real-world social networks as they form. The system is built around a small, low-power wearable device.

Prior to an event, each participant pairs their ProximityNet device with their smartphone via Bluetooth. During the event, the wearable continuously scans for other ProximityNet devices in the vicinity. When a connection is established, it logs the interaction, including the unique identifier of the other device and the duration of the proximity. This data is periodically transmitted to a central server using the paired smartphone as a relay.

The core objective is to create a rich dataset of anonymized interactions, which can be processed to reveal patterns and structures within the event's population.

## 2 Potential Applications and Use Cases

The data gathered by the ProximityNet system can be leveraged for a variety of powerful applications. Once an event concludes, the aggregated interaction data can be processed to:

- **Social Network Analysis:** Construct a social graph where attendees represent nodes and the cumulative time spent in proximity forms the weighted edges between them. This graph can be used to:
  - Identify influential individuals (centrality analysis).
  - Discover communities or cliques within the group.
  - Visualize the flow of interaction and the overall structure of the social event.
  - Provide personalized post-event analytics to users, highlighting their key interactions.
- **Crowd and Security Monitoring:** By combining the interaction graph with high-level location data (a potential future extension, not included in this prototype), the system

could provide insights for event organizers and security personnel. Analysis could answer questions such as:

  - Where are the highest concentrations of people at any given time?
  - How do crowd dynamics shift when a specific group moves through the venue?
  - Can we predict potential bottlenecks or overcrowded areas?

- **Health and Safety Contact Tracing:** In environments where health is a concern, the system offers an effective and privacy-preserving method for contact tracing. If an individual reports an illness post-event, the system can quickly identify others who spent significant time in close proximity, enabling targeted and efficient communication.

## 3 Proposed Technical Details

This section will outline the core technical components of the ProximityNet prototype.

### 3.1 Hardware Components

The core of the wearable device will be the **ESP32 microcontroller**. This platform is exceptionally well-suited for the project due to its compelling combination of features:

- **Integrated Wireless Connectivity:** The built-in support for both Wi-Fi and Bluetooth Low Energy (BLE) is essential for device-to-device communication and for relaying data through a smartphone.
- **Low Power Consumption:** The ESP32's deep-sleep modes are critical for ensuring the wearable device can operate for the entire duration of an event on a small battery.
- **Sufficient Processing Power:** It possesses ample performance to handle BLE scanning, data buffering, and communication protocols without issue.
- **Cost-Effectiveness and Community Support:** The low cost and extensive ecosystem make it an ideal choice for developing a scalable prototype.

### 3.2 Communication Protocol

The system will employ a connectionless communication strategy using **BLE Advertisement Packets**. This approach is highly efficient and scalable for a scenario with many devices.

The process is as follows:

1. **Broadcasting:** At a randomized interval to minimize packet collisions, each device will broadcast a BLE advertisement packet. This packet's payload will contain a unique identifier (ID) for the device.
2. **Scanning:** Concurrently, all other devices will be in a low-power scanning mode, listening for these advertisement packets from nearby peers.
3. **Logging:** When a device receives an advertisement packet, it will log the detected device's ID along with the **Received Signal Strength Indicator (RSSI)**. The RSSI value serves as a proxy for physical proximity. These interaction logs (Peer ID, RSSI, Timestamp) will be temporarily stored in the device's local memory (RAM or flash).
4. **Data Relay:** Periodically, the device will connect to its paired smartphone to upload the buffered logs. The smartphone application is then responsible for transmitting this data to the central server.

This one-to-many broadcast architecture is inherently resilient; the occasional loss of an advertisement packet due to collisions does not compromise the system's ability to detect presence, as subsequent packets will be captured. This avoids the complexity and overhead of establishing and maintaining direct connections between all nearby devices.

## 3.3 Software Architecture

- **Firmware:** The embedded software on the ESP32 responsible for managing BLE broadcast/scan cycles, data logging, power management, and communication with the mobile app.
- **Mobile Application:** A lightweight app (Android/iOS) for initial device pairing, user authentication, and serving as a data gateway to the backend. This component may not be developed for the prototype, leaving the duty to send the data to a server to the esp32 microcontroller.
- **Backend Server:** A cloud-based service with a database to store and process the interaction data, and an API for the mobile app to report data.