

TraceMeNow: an Open-Source Software Framework for Indoor Localization Applications

Mauro Borrazzo¹, Giuseppe D'Ambrosio¹, Vittorio Scarano¹ and Carmine Spagnuolo¹

¹Dipartimento di Informatica, Università degli Studi di Salerno, Salerno, Italy

Abstract

The increased number of smartphone-connected devices and the pervasive presence of sensors enabled the application of localization and tracking technologies in multiple contexts. Health, customer care, traveling, crowd management are just some of the possible fields where an Indoor Positioning System could be useful. Despite the effort of the research community, the integration of an Indoor Positioning System within an application is still difficult since it requires expertise in diverse fields like communication technologies, localization techniques, and hardware. This paper presents TraceMeNow, an open-source framework for developing applications comprising an Indoor Positioning System based on Bluetooth Low Energy and low-cost hardware. TraceMeNow is designed to improve developers' experience without requiring specific knowledge to be used, thus aiming to be a simple and valuable tool suitable for different situations and accessible to any developer. TraceMeNow adopts a modular architecture enhancing the interoperability between components and supporting the developer throughout all the implementation phases. We aim to show the flexibility and ease of use of our framework by presenting an application to address a real-world use case. TraceMeNow aims to reduce the cost and the effort needed to create an application comprising an Indoor Positioning System, providing the basis for all the components and relying on mainstream technologies for hardware and communication. Moreover, TraceMeNow allows developers exploiting cloud computing when facing large scenarios with specific requirements such as high scalability and reliability, maintaining the same ease of use since the interaction with the provider is entirely abstracted.

Keywords

Indoor Localization, Indoor Positioning System, Bluetooth Low Energy

1. Introduction


Nowadays, Indoor Positioning System (IPS) is a growing research area due to the rising necessity to track people and objects in a variety of contexts [1] exploiting indoor localization. This field involves several technologies to obtain information associated with a particular position within buildings. The global market size of indoor localization is expected to increase from USD 6.1 billion in 2020 to USD 17.0 billion by 2025 [2]. The major growth factors are the increased number of smartphone-connected devices and location-aware technologies, as well as the pervasive presence of sensors and smart objects, enabling the integration of indoor localization in countless applications. Most IPSs are used in different vertical sectors, such as health, to help in hospital

IPIN 2021 WiP Proceedings, November 29 – December 2, 2021, Lloret de Mar, Spain

✉ m.borrazzo@studenti.unisa.it (M. Borrazzo); gdambrosio@unisa.it (G. D'Ambrosio); vitsca@unisa.it (V. Scarano); cspagnuolo@unisa.it (C. Spagnuolo)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

tracking [3] [4] and improve epidemic control measures [5], workspace-management [6], and tourism [7], but there are many possible use cases. Spacing from customer care and traveling, from crowd management and surveillance, many fields could benefit from adopting an IPS improving people's experience, services, security, and organization [8].

Despite the great interest of the scientific community, indoor localization is still far from mature and has many open challenges [9]. Developing an application exploiting an IPS requires knowledge in diverse fields due to the number of involved technologies, techniques, and algorithms. Programmers need to choose the communication technology, the localization techniques, the hardware involved besides the software to use, and the application design.

Although some attempts have been made to ease the development of an IPS, it is still an open research area. In general, prior studies are limited to a subset of issues regarding indoor localization, namely identifying the most suitable communication technology or improving the position estimation through signal processing and filtering. Moreover, no other works in the literature focus the attention on the developers' experience. The existing solutions are usually related to a specific context or hard to use for non-experts and rarely provide code access.

The aim of this work is to propose a tool specifically designed to support programmers for the development and the integration of an IPS within an application without requiring deep knowledge in signal processing, communication technologies, or hardware implementation.

In this work, we introduce TraceMeNow, an open-source framework for developing applications exploiting IPS, based on Bluetooth Low Energy (BLE). TraceMeNow allows programmers to build software thanks to a modular architecture simplifying the implementation of tracking and localization modules within an indoor environment using low-cost devices. Each component of the framework has a dedicated library with default functionalities designed to support the developer in several situations. Additionally, every method can be easily customized, and new features can be introduced if needed. Furthermore, the framework includes the possibility to exploit cloud computing resources to gain scalability and manage large scenarios. Complete code, full documentation and other material about TraceMeNow is available on GitHub¹.

We present the flexibility and ease of use of TraceMeNow by describing an application designed to address a real-world scenario. We propose the Attendance Management System (AMS) helping to manage the presence and the punctuality of employees within companies, offices, and factories. In detail, AMS turns employees' smartphones into an access badge automatically detected and tracked when located within the workplace where the tracking nodes are installed. The devices sense the employees' movements and report the information to a central server monitoring the building. The application complete code is available on GitHub².

The paper is structured as follows. Section 2 describes the related work. Section 3 presents TraceMeNow, its architecture, and its components. Section 4 introduces the AMS design. Section 5 includes the conclusion and future work.

¹TraceMeNow official GitHub repository - <https://github.com/isislab-unisa/trace-me-now>

²TraceMeNow: Attendance Management System - <https://bit.ly/attendance-management-system>

2. Related Work

Several studies in the literature regard techniques to improve the accuracy of position estimation, using algorithms and filtering techniques concerning signal processing. Jiang et al. [10] described an algorithm based on trilateration and a dynamic-circle-expanding mechanism, while Yan et al. [11] proposed a multi-dimensional scaling-based approach. The work of Zhuang et al. [12] exploited a more sophisticated technique based on a combination of channel-separate polynomial regression model, channel-separate fingerprinting, outlier detection, and extended Kalman filtering to increment the accuracy in smartphone-based indoor localization. The same problem is addressed in Satan et al. [13] and Ciabattini et al. [14], proposing an Android application and an architecture for IPS, based on smartphones, respectively.

Another area of research in indoor localization concerns systems to address specific situations or tied to particular hardware. SmartITS [15] is a human identification, monitoring, and location tracking system based on GPS, Wi-Fi, and BLE. However, the system needs devices with certain characteristics, and the authors do not provide an open-source implementation. Similar issues are found in the IoT-based and BLE-based IPS presented in Mekki et al. [16] since it requires specific devices. Regarding solutions for vertical sectors, the work of Nguyen et al. [4] proposes an asset tracking and real-time location system for hospitals using RSSI-based estimation and BLE. Belka et al. [7] presents an IoT solution for BLE-based indoor tracking systems for smart tourism exploiting IoT sensors, while Li et al. [17] introduces a framework for indoor localization with dedicated algorithms for emergency systems.

Other works in the literature report several solutions aiming to improve the development process of an IPS. Anyplace 4.0 IoT [18] is an open-source architecture for localization using the physical environment data stored in a database for estimation. Teran et al. [19] employed a similar approach introducing a modular system able to compute the position after a training and calibration phase. A framework for IPS development is present in Jimenez et al. [20] with an open-source multi-sensor framework for indoor smartphone positioning providing a series of tools to collect and interpret data. However, the system requires knowledge in managing and exploiting the information received from the different sensors.

The scientific community exhibited a growing interest in indoor localization, proposing various studies addressing different aspects of the topic. However, although some attempts have been made to ease the development and the integration of an IPS within an application, it is still an open problem. As described before, the existing solutions are complex to use or focus only on acquiring better positioning accuracy giving little attention to code and developer experience. A practical IPS should bear the properties like easy implementation, acceptable localization accuracy, feasible system cost, scalability and robustness, and reasonable computational complexity [9]. Moreover, an application needs to be accurately designed to exploit an IPS with these characteristics. Programmers have to deal with the different hardware and software technologies included, besides writing the code.

The objective of this work is to propose a simple and valuable tool suitable for any situation and accessible for developers with low experience in indoor localization and related technologies. We introduce TraceMeNow, an open-source framework for developing IPS compatible with standard and cheap devices, suitable for any scenario. The flexibility of TraceMeNow unties it from a specific usage, as it happens in existing works dedicated to specific contexts, such as

hospitals [4], smart tourism [7] or emergency management [17]. TraceMeNow does not require a strong knowledge of signal processing, communication techniques, or hardware since it offers a series of libraries comprising of default implementations designed to be helpful in most cases, as well as methods to customize existing functionalities and introduce new ones.

3. TraceMeNow

TraceMeNow supports programmers in developing applications comprising localization and tracking capabilities providing interoperability between components thanks to its modular architecture. The IPS implemented with the framework consists of mobile nodes, whose position is unknown, and several localization nodes, with known locations, placed within the indoor environment communicating with an application module acting as a server.

The position estimation and the communication between the nodes rely on BLE, a protocol that already demonstrated its value in IPS [21] [22]. BLE is designed for short-range wireless transmission of small amounts of data with low energy and cost, granting a convenient trade-off between power consumption and accuracy [23]. TraceMeNow exploits the one-to-many communication provided by BLE called broadcasting, enabling data sent to any listening device in range. Specifically, the mobile nodes transmit BLE packets in broadcast, allowing the nearby anchor node to detect and localize them. Another reason for using BLE is the high market penetration of the protocol [22] that increases the hardware compatible with our framework since nowadays, almost every smartphone, tablet, and smart device supports BLE or can be equipped with cheap external modules.

TraceMeNow is designed to guide the developer through the implementation and integration of the IPS within any application, thanks to the libraries available for each system component. The framework proposes several default implementations useful in most scenarios as well as methods to modify the existing features and/or introduce new ones. For instance, other localization techniques can be used, or additional components can be included.

The modular architecture of TraceMeNow is illustrated in Figure 1 where each element corresponds to a specific component of the IPS. The system includes mobile nodes, tracking nodes, the communication protocol, and the application module. The mobile node can be any device with a BLE and Wi-Fi interface on which an application built with TraceMeNow is installed. This enables the node to continuously broadcast a message with some defined information via BLE while the Wi-Fi interface is used to interact with the application module using a specific communication protocol. A group of tracking nodes is responsible for the localization process and consists of embedded systems supporting the two wireless technologies used. TraceMeNow provides the localization system exploiting the BLE interface and the communication protocol using Wi-Fi. The application module handles the whole IPS and can be hosted on a Cloud or on-premise architecture. It contains a lightweight server engine, a database to store the collected data, maintaining the global status of the system, and the TraceMeNow communication protocol based on the Event/Notification system.

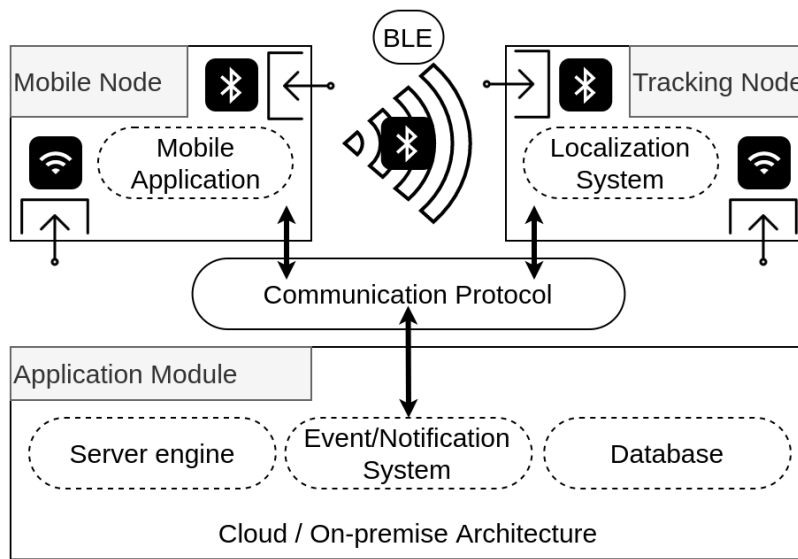


Figure 1: TraceMeNow general architecture.

The workflow of an IPS based on TraceMeNow is the following:

1. the mobile node continuously sends BLE packets in broadcast;
2. the nearby tracking node detects the packets, collects the information about the mobile node, computes the distance, and sends all data through the Event/Notification system;
3. the Event/Notification system receives the message causing the generation of an event triggering a notification;
4. the application module receives the notification about the new event, collects the attached information, and executes the correspondent functions, if needed.

The modular architecture adopted in TraceMeNow allows developers implementing a fully functional IPS easily and quickly using the libraries available for each component. In particular, a series of default implementations provides the basic tracking and localization functionalities as well as methods for communication and data storage. Moreover, the developer is supported in customizing and extending the system functionalities. For instance, different position estimation techniques can be integrated within the localization system, and the Event/Notification system can handle additional events. Furthermore, the server engine exposes its functionalities using a series of RESTful APIs allowing developers to easily integrate the application module with other components such as web app or visualization engines.

3.1. Mobile node

TraceMeNow provides a library to implement a mobile application that will turn any BLE device into a BLE beacon, i.e., a node that continuously sends BLE packets in broadcast. These packets are based on the iBeacon protocol, firstly developed by Apple but currently compatible with

a wide selection of hardware and operating systems. The iBeacon protocol defines a specific data structure included in the BLE packet containing a Universal Unique Identifier (UUID) and additional information [24].

The library is based on Android and comprehends all the dependencies and requirements needed to build a ready-to-use application. The methods available help the developer to set up, start and stop the broadcast transmission making the mobile node visible and detectable by the tracking nodes. Moreover, functions exist to communicate with the application module using the Event/Notification system. In detail, the mobile node can send a message to a topic or subscribe to it to receive notifications when an event occurs. If the notifications are no longer required, the node can unsubscribe from the topic. The developer can implement a function triggered when a message is sent on a specific topic, defining the behavior of the node in response to the notification.

3.2. Tracking node

The tracking node library is based on Javascript and runs over a NodeJS engine³ requiring a device with a compatible operating system in addition to the BLE and Wi-Fi interfaces. The methods included allow the developer to start and stop the scan of the room and communicate with the application module using the Event/Notification system. Once the scanning is started, the tracking node will automatically localize and track the mobile nodes within the range, sending messages according to the events recorded. It is important to note that each tracking node needs to be configured with the room number where it is placed to perform the localization since the position must be known. The library provides specific methods supporting the addition of new functionalities, such as introducing new actions in response to events. The default configuration of TraceMeNow involves a series of Raspberry Pi boards as tracking nodes. However, any hardware with Wi-Fi and BLE, compatible with NodeJS, can be used. Raspberry Pi is a portable computer board with a Linux-based operating system particularly suitable for indoor localization thanks to its low cost, low power consumption, and high availability of built-in hardware like Wi-Fi and BLE module. The use of Raspberry Pi for indoor localization is not a novelty in literature. Several works exist including these boards within their IPS architecture [7] [18] [25].

Localization system The localization system of TraceMeNow is included in the tracking node and the default implementation uses the Received Signal Strength Indicator (RSSI) to estimate the position.

The RSSI is one of the most used methods for localization in IPS [26] since it does not require additional hardware or complex software [9]. The RSSI value indicates the power signal strength of a received radio signal. The higher the value, the smaller the distance between the sender and the receiver. The default implementation present in the library uses the most commonly adopted formula, described in Shang et al. [27], to estimate the distance between the mobile node and the tracking node.

³NodeJS - <https://nodejs.org>

The formula employing the RSSI is described below:

$$d = 10^{\frac{(P_{tx} - RSSI)}{10 \cdot p_{loss}}}$$

- d : distance;
- P_{tx} : transmit power of collector device;
- $RSSI$: Received Signal Strength Indicator;
- p_{loss} : path loss exponent.

3.3. Communication protocol: Event/Notification system

The Event/Notification system realizes the communication protocol of TraceMeNow exploiting the MQTT protocol. MQTT is a lightweight publish/subscribe messaging protocol ideal for connecting remote devices with a small code footprint and minimal network bandwidth⁴.

The system includes two MQTT topics for each event, with some exceptions that use only one topic. The first one is the event topic, where the tracking node publishes the message indicating the detection of an event. The second one is the notification topic, where the function triggered by the event sends the processed data in the form of notification and where the components interested in a particular event will subscribe. When a change in the IPS occurs, a message with attached information is sent on the event topic, triggering a function. This function performs some action and then publishes a response on the notification topic. The use of two or more topics for each event type improves communication management granting more control to the developer. For instance, the interested device can subscribe only to the notification topic where the function will publish the elaborated data.

The framework provides a default implementation of the Event/Notification system, including the most commonly occurring events that may happen within an IPS. However, custom events can be easily implemented using TraceMeNow libraries, defining the related functions for each of them.

The topics of the default Event/Notification system are shown in Table 1 and detailed below:

- *device/new* - contains the events related to the detection of a new mobile node. The correspondent *notify/new* notification topic notifies all the subscribers every time a device enters into the system;
- *device/delete* - contains the events related to the exit of a mobile node. The correspondent *notify/delete* notification topic notifies all the subscribers every time a device leaves the system;
- *device/location* - the position requests of a specific device. Publishing a message with the *uuid* of the device of interest on this topic allows to receive the device position on the correspondent *notify/location/uuid* notification topic, where the *uuid* will be the one specified;
- *notify/position/uuid* - notifies all the subscribers every time the device position with the specified *uuid* changes;

⁴MQTT - <https://mqtt.org>

Table 1

TraceMeNow default Event/Notification system topics.

Event	Notification
device/new	notify/new
device/delete	notify/delete
device/location	notify/location/uuid notify/position/uuid notify/change/uuid
device/custom	notify/custom

- *notify/change/uuid* - notifies all the subscribers every time the tracing node of the device with the specified *uuid* changes, i.e., moves under the range of a node different from the previous one.

All the messages exchanged through the MQTT protocol respect the default JSON format depicted in Listing 1. However, following the customizable principle of TraceMeNow, the developer can customize the message format adding additional values, modifying or removing the existing ones. The default values are the following:

- `uuid`: unique identifier of the mobile node;
- `lastPosition`: distance in meters between the mobile node and the tracking node under which it is located;
- `lastSeen`: time of the last interaction with the last seen tracking node;
- `trackingNodeId`: identifier of the last seen tracking node;
- `roomNumber`: identifier of the room where the device is located; the room corresponds to the localization area of a tracking node.

```
{
  "uuid": "XXXXXX-XXXX-XXXX-XXXX-XXXXXXXX",
  "lastPosition": "1.11",
  "lastSeen": "11:11",
  "trackingNodeId": "XXXX-XXXX-XXXX-XXXX-XXXXXXXX",
  "roomNumber": "1"
}
```

Listing 1: MQTT Message format representing a device in the system.

3.4. Application module

The application module manages the whole IPS and includes the server engine, the database, and the Event/Notification system. The server engine represents the core of the application module connecting all the different components. It maintains the global status of the system

within a NoSQL database, storing all the information of the devices seen so far. Moreover, the server engine handles all the application functionalities implementing the functions executed in response to the events received through the Event/Notification system. For example, when a message is published on the *notify/location/uuid* topic, the server executes a query on the database to get the device position with the specified *uuid*.

TraceMeNow provides two different architectures for hosting the application module based on the application requirements. The on-premise architecture is suited for small systems with few devices involved and grants low cost if not null. The low requirements of the application module enable the use of a regular computer for the execution. On the opposite side, when facing scenarios with many devices involved, the cloud architecture grants the scalability and the performance needed, keeping the costs low thanks to the appropriate use of the cloud computing service models. TraceMeNow includes a library for each architecture handling all the needed tasks automatically, from the connection with the database to the communication with other components. However, the developer can customize each process and implement additional functionalities, such as custom events and functions.

On-premise architecture The on-premise architecture implements the application module using a series of Docker containers, ensuring high compatibility and making the installation as quick and simple as possible. In detail, TraceMeNow provides a Docker Compose file which includes all the dependencies and the services needed to execute the application module exploiting the TraceMeNow image available on DockerHub⁵. The server engine is realized through several Python modules while the Event/Notifications system is based on Eclipse Mosquitto⁶, an open-source message broker based on the MQTT protocol. The database exploits MongoDB⁷, a cross-platform NoSQL database with high scalability and flexibility. Finally, the security of the architecture relies on the Transport Layer Security protocol.

Developers can add new functionalities following the standard workflow of TraceMeNow. For instance, a new event can be introduced, creating one topic where it is generated, one where the notification is provided, and defining the function triggered in response. The function can be written in Python language, whose input parameters consist of the body of the message sent on the event topic and the returned values are the response sent on the notification topic. Both follow the JSON format defining a standard to follow simplifying the implementation process.

Cloud architecture The cloud architecture exploits cloud computing to realize the application module. TraceMeNow handles all the processes regarding the cloud services, from their creation to the integration within the IPS, abstracting the developer from the interaction with the cloud provider.

The use of cloud in indoor localization is a growing trend thanks to the advantages provided, such as easy maintenance, cost-effectiveness, agility, flexibility, scalability, and effective management [2]. The integration of cloud computing within an IPS is based on the computation offloading process, a standard solution used in the Internet of Things (IoT) field to overcome the

⁵TraceMeNow Image - <https://hub.docker.com/r/isislab/trace-me-now>

⁶Eclipse Mosquitto - <https://mosquitto.org>

⁷MongoDB - <https://www.mongodb.com>

limits in computational power and memory of the devices. Computational offloading transfers intensive computational tasks from the local to an external environment, such as a cloud server, granting the high scalability needed to manage thousands of devices or large scenarios.

The application module for the cloud architecture employs Amazon Web Services (AWS)⁸. The server engine is realized using AWS Lambda functions⁹, the serverless service of AWS. A serverless function is a block of code that can be run on cloud with zero administration since the provider automatically allocates the resources needed for the execution. Moreover, the serverless paradigm follows the event-driven model meaning that a function is triggered in response to a defined event, making it particularly suited for IPS. TraceMeNow uses Lambda functions to perform the actions in response to the events and manage interaction with the other components. The Event/Notification system is based on AWS IoT Core¹⁰ acting as the MQTT broker. The service enables the integration of IoT devices with other AWS resources like AWS Lambda. The database is implemented through Amazon DynamoDB¹¹, a serverless key-value and document database with high performance and high scalability. The interaction with the application module relies on Amazon API Gateway¹², a service for the creation and management of APIs. In detail, the APIs represent the trigger of the Lambda functions, working as entry points. For instance, any database operation is performed using the endpoint provided by API Gateway. Finally, the system security is based on AWS Identity and Access Management (IAM)¹³ that enables the access management to the AWS services using specific roles and policies enhancing the privacy protection within applications. Moreover, all the tracking nodes need to have an AWS PEM certificate to access the environment.

All the services required in the cloud architecture can be deployed using dedicated scripts provided by TraceMeNow. After their execution, the resulting environment will be up and running with the following resources:

- a group of Lambda functions related to the default events;
- an IoT Core instance configured to trigger the default Lambda functions;
- a group of API Gateway APIs acting as the entry points for the default Lambda functions;
- a DynamoDB database;
- a new IAM role and a new IAM policy.

The developer can add new events and functions without using the AWS console, thanks to specific scripts abstracting the interaction with the provider. In detail, the new Lambda function can be written in Python code whose input parameters and returned values follow the same rules of the on-premise architecture. A new API Gateway represents the entry point, while the IoT core provides the implementation of the trigger event generated when a message comes from a specific MQTT topic. The function deployment is done automatically through the scripts making the integration phase as simple as possible.

⁸ Amazon Web Services (AWS) - <https://aws.amazon.com>

⁹ AWS Lambda - <https://aws.amazon.com/lambda>

¹⁰ AWS IoT Core - <https://aws.amazon.com/iot-core>

¹¹ Amazon DynamoDB - <https://aws.amazon.com/dynamodb>

¹² Amazon API Gateway - <https://aws.amazon.com/api-gateway>

¹³ AWS IAM - <https://aws.amazon.com/iam>

4. Use Case: Attendance Management System

This section describes an application developed with TraceMeNow designed for a real-world scenario to demonstrate the flexibility and ease of use of our framework. We developed the Attendance Management System (AMS) that helps to track employees' attendance, monitoring their presence and punctuality by exploiting indoor localization. The application complete code is available on GitHub¹⁴.

Employee attendance tracking is a priority for many employers, making it easier to schedule work shifts and identify any attendance issues. Nowadays, offices and factories provide access badges to their employees to report the start, the end of the work shift, and the breaks, checking the respect of the working time. However, this kind of system is not safe and can be easily tricked, for example, by clocking in and then leaving the workplace or by giving the badge to a colleague who clocks in for multiple employees not present. Therefore, access badges are not reliable and, in addition, the infrastructure needed to realize the system is usually expensive since it requires proprietary hardware and software and frequent maintenance.

An architecture composed of an IPS and an application specifically designed can overcome the problems of traditional access card systems, introducing innovative solutions. The AMS developed with TraceMeNow aims to be one of these solutions, providing an effective and low-cost system easy to install and maintain. The idea is to turn employees' smartphones into badges using an Android application. A group of devices installed within the workplace senses the employees' movements and reports the information to a central server monitoring the building. The application provides a registration phase to gain the information and the working time of the employee. Based on that data, the smartphone shows a notification if the user leaves the building during the work shift, asking for a reason. All the information acquired and the movements detected are registered within a database accessible through a web interface. The use of TraceMeNow reduces the cost and the effort needed to create AMS since the default implementations included in the libraries already provide the basis for all the components. In addition, the installation of the IPS is easy since the Raspberry Pi boards can be used as the tracking node and placed everywhere within the building, thanks to their tiny size. Finally, the upfront investment is small considering that smartphones with BLE are widely used, Raspberry Pi boards are cheap, and the server does not require any specific requirements.

4.1. Android application

The Android application¹⁵ developed using the mobile node library turns the smartphone into an access badge automatically detected and tracked when located within the workplace. On the first start, a registration form is shown where the employee inserts name and working time to sign-up to the system. When activated, the badge constantly sends BLE packets in broadcast with the employee data allowing the tracking nodes to identify and track them. Using this mechanism, the badge is automatically registered as soon as it goes in or out of the building. AMS can notice if the employee leaves the workplace during the shift by sending a notification on the smartphone. The application does not allow other operations until a reason for the

¹⁴TraceMeNow: Attendance Management System - <https://bit.ly/attendance-management-system>

¹⁵TraceMeNow: AMS Android Application - <https://bit.ly/ams-android-application>

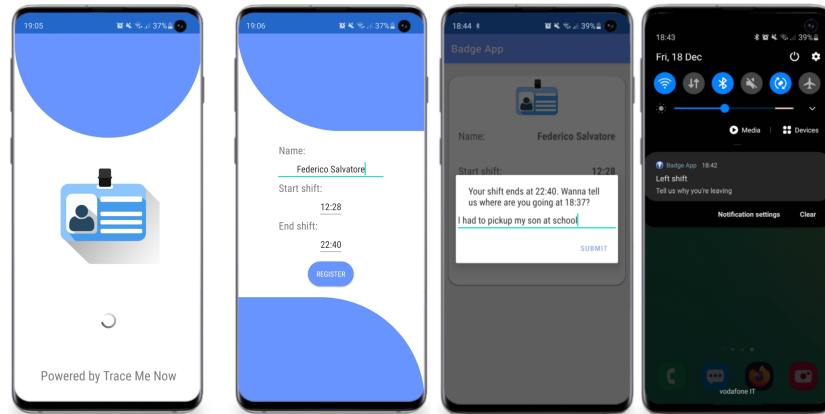


Figure 2: The AMS Android application.

movement is provided. Finally, the badge can be disabled when needed, such as when the employee is not at work or whenever the localization should not be performed. The overall interface of the Android application is shown in Figure 2.

4.2. Detector

The tracking node library transforms the Raspberry Pi boards into detectors¹⁶, allowing them to scan the building and detect the employees' smartphones. The boards communicate with the central server reporting the information collected through the Event/Notification system. The default implementation of TraceMeNow already provides the mobile node detection, position estimation, and data report without requiring additional coding, simplifying the development of the tracking node application.

4.3. Central server

The central server¹⁷ handling AMS uses the application module library hosted on the on-premise architecture. The default implementation realizes the data collection and the communication with the boards and the smartphones. However, some additional events and functions are needed to realize AMS. In detail, the system needs to get the information about all the employees, those within the workplace or who left it at least once, in addition to the registration phase. The Event/Notification system allows the central server managing all the events detected within the building. Whenever a new device is detected, the database is updated and accessed to check if the employee is late. On the other hand, when a badge is no longer within the workplace, the server controls if the work shift is finished and, if not, sends a notification to the smartphone asking for a reason.

¹⁶TraceMeNow: AMS Detector - <https://bit.ly/ams-detector>

¹⁷TraceMeNow: AMS Central Server - <https://bit.ly/ams-central-server>

UUID	Name	Start shift	End shift	On-site
2083ae30-3b7f-49a7-ae5a-ac69764d18ad	Mauro Borrazzo	9:30	18:30	Yes
49f98e7-3910-4338-bc9f-86a0373b8d3	Giorgio Tucano	8:30	19:30	Yes
c1168464-a0e0-4d67-ab88-d895d32be50d	Pippo Baudo	7:0	19:0	No
3b4c602a-6373-4b2d-87d2-1c94e6a95933	Lapo Elkann	20:30	6:30	No
2c7d065f-6309-4789-9722-277382787206	Umberto Miletto	20:30	23:59	No
28de4cce-1bbc-4d8a-bc92-4346529585c0	Ornella Vanoni	5:25	13:15	No
4b023985-4948-43d4-bd36-c1a14d5cec17	Lucia Miranda	16:30	23:30	No
a5043354-26d2-4a30-b70c-cdb63d73b1c3	Michele Sallicandio	23:0	6:50	No
e78e79f8-713e-4ba1-8e63-68d343395cbd	Nunzia Ferrante	11:30	19:30	No
60aaec45-3683-445c-a171-311477a6532e	Andrea Nunziata	11:26	23:27	No
e1b2b63a-288d-4c0d-b214-30a62a1c137b	Ugo Foscolo	12:7	20:8	No

Figure 3: The AMS web interface displaying all the employees in the building.

UUID	Name	Start shift	End shift	Arrived	In late
2083ae30-3b7f-49a7-ae5a-ac69764d18ad	Mauro Borrazzo	9:30	18:30	9:31	No
49f98e7-3910-4338-bc9f-86a0373b8d3	Giorgio Tucano	8:30	19:30	9:01	Yes

Figure 4: The AMS web interface displaying the arrival of notifications.

4.4. Web interface

A web interface¹⁸ built from scratch using React¹⁹ provides access to all the data of the system, including employees' information and notifications responses. Even though TraceMeNow does not provide a specific library for a web client, its design helps the programmer developing the interface effortlessly using the application module libraries. In detail, the Event/Notification system allows the update of the visualized data in real-time, triggering a function each time an event of interest occurs. Figure 3 shows the admin dashboard displaying all the employees within the building and Figure 4 depicts the arrival of notifications.

¹⁸TraceMeNow: AMS Web Interface - <https://bit.ly/ams-web-interface>

¹⁹React - <https://it.reactjs.org>

5. Conclusion

In this work, we presented TraceMeNow, an open-source framework for developing applications exploiting a BLE-based IPS using low-cost hardware and without requiring specific knowledge. TraceMeNow aims to provide a simple and valuable tool suitable for different situations and accessible for any developers with low experience in localization systems, signal processing, and communication technologies. The modular architecture of our framework grants interoperability between components providing dedicated libraries for each one. Developers are supported in all phases of the implementation, with default functionalities useful in most situations. In addition, TraceMeNow includes methods to customize every aspect of the components or add additional features. The IPS based on TraceMeNow consists of mobile nodes, whose position is unknown, localized by a group of tracking nodes with known locations placed within the indoor environment. An application module manages the whole system realizing the communication protocol and the data storage. It can be hosted either on a cloud architecture or an on-premise one allowing the developer to choose according to the requirements while still maintaining the same ease of use. Full documentation and other material is available on GitHub²⁰. TraceMeNow is still under active development. Future work includes refining the existing methods, introducing additional default implementations, and extending the compatibility with hardware and cloud providers. We plan to introduce the support to other operative systems, like iOS, and further BLE modules. The NodeJS library²¹ used is compatible only with specific models, even if the most common ones. Although the developer can easily change the library, a more generalized approach is preferred. Moreover, we intend to support other cloud providers besides AWS, such as Microsoft Azure²² and Google Cloud Platform²³. Finally, different techniques for localization can be introduced as default implementation, allowing the developer to choose between a more accurate or a more rough position based on the requirements of the IPS.

References

- [1] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, E. Aboutanios, Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications, *IEEE Communications Surveys and Tutorials* 19 (2017) 1327–1346. doi:10.1109/COMST.2016.2632427.
- [2] MarketsandMarkets™, Indoor Location Market by Component, Deployment Mode, Organization Size, Technology, Application, Vertical, and Region - Global Forecast to 2025, 2020. <https://www.marketsandmarkets.com/Market-Reports/indoor-location-market-989.html>.
- [3] C. Yang, H. Shao, Wifi-based indoor positioning, *IEEE Communications Magazine* 53 (2015) 150–157. doi:10.1109/MCOM.2015.7060497.
- [4] Q. H. Nguyen, P. Johnson, T. T. Nguyen, M. Randles, A novel architecture using iBeacons

²⁰TraceMeNow official GitHub repository - <https://github.com/isislab-unisa/trace-me-now>

²¹Noble - <https://github.com/noble>

²²Microsoft Azure - <https://azure.microsoft.com>

²³Google Cloud Platform - <https://cloud.google.com>

- for localization and tracking of people within healthcare environment, Global IoT Summit, GIoT 2019 - Proceedings (2019).
- [5] K. A. Nguyen, Z. Luo, C. Watkins, Epidemic contact tracing with smartphone sensors, *Journal of Location Based Services* 14 (2020) 92–128. doi:10.1080/17489725.2020.1805521.
 - [6] B. Doherty, N. Gardner, A. Ray, B. Higgs, I. Varshney, The shakedown: developing an indoor-localization system for quantifying toilet usage in offices, *Architectural Science Review* 63 (2020) 325–338. doi:10.1080/00038628.2020.1748869.
 - [7] R. Belka, R. S. Deniziak, G. Łukawski, P. Pięta, Ble-based indoor tracking system with overlapping-resistant iot solution for tourism applications, *Sensors (Switzerland)* 21 (2021) 1–21. doi:10.3390/s21020329.
 - [8] R. Melamed, Indoor localization: Challenges and opportunities, *Proceedings of the International Conference on Mobile Software Engineering and Systems* (2016) 1–2. doi:10.1145/2897073.2897074.
 - [9] S. Subedi, J. Y. Pyun, A survey of smartphone-based indoor positioning system using RF-based wireless technologies, *Sensors (Switzerland)* 20 (2020) 1–32. doi:10.3390/s20247230.
 - [10] J. A. Jiang, X. Y. Zheng, Y. F. Chen, C. H. Wang, P. T. Chen, C. L. Chuang, C. P. Chen, A distributed rss-based localization using a dynamic circle expanding mechanism, *IEEE Sensors Journal* 13 (2013) 3754–3766.
 - [11] S. Yan, H. Luo, F. Zhao, W. Shao, Z. Li, A. Crivello, Wi-Fi RTT based indoor positioning with dynamic weighted multidimensional scaling, *2019 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2019* (2019) 12–15. doi:10.1109/IPIN.2019.8911783.
 - [12] Y. Zhuang, J. Yang, Y. Li, L. Qi, N. El-Sheimy, Smartphone-based indoor localization with bluetooth low energy beacons, *Sensors (Switzerland)* 16 (2016) 1–20. doi:10.3390/s16050596.
 - [13] A. Satan, Z. Toth, Development of bluetooth based indoor positioning application, *2018 IEEE International Conference on Future IoT Technologies, Future IoT 2018* 2018-Janua (2018) 1–6. doi:10.1109/FIOT.2018.8325586.
 - [14] L. Ciabattini, G. Foresi, A. Monteriù, L. Pepa, D. Proietti, P. Luca, S. Federica, Real time indoor localization integrating a model based pedestrian dead reckoning on smartphone and BLE beacons, *Journal of Ambient Intelligence and Humanized Computing* 10 (2019) 1–12.
 - [15] T. Kulshrestha, D. Saxena, R. Niyogi, V. Raychoudhury, M. Misra, SmartITS: Smartphone-based identification and tracking using seamless indoor-outdoor localization, *Journal of Network and Computer Applications* 98 (2017) 97–113. doi:10.1016/j.jnca.2017.09.003.
 - [16] K. Mekki, E. Bajic, F. Meyer, Indoor positioning system for iot device based on ble technology and mqtt protocol, *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings* (2019) 787–792.
 - [17] N. Li, B. Becerik-Gerber, L. Soibelman, B. Krishnamachari, Comparative assessment of an indoor localization framework for building emergency response, *Automation in Construction* 57 (2015) 42–54.

- [18] P. Mpeis, T. Roussel, M. Kumar, C. Costa, C. Laoudiasdenis, D. Capot-Ray, D. Zeinalipour-Yazti, The Anyplace 4.0 IoT Localization Architecture, Proceedings - IEEE International Conference on Mobile Data Management 2020-June (2020) 218–225. doi:10.1109/MDM48529.2020.00045.
- [19] M. Terán, J. Aranda, H. Carrillo, D. Mendez, C. Parra, Iot-based system for indoor location using bluetooth low energy, 2017 IEEE Colombian Conference on Communications and Computing (COLCOM) (2017) 1–6. doi:10.1109/ColComCon.2017.8088211.
- [20] A. R. Jimenez, F. Seco, J. Torres-Sospedra, Tools for smartphone multi-sensor data registration and GT mapping for positioning applications, 2019 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2019 (2019) 1–9. doi:10.1109/IPIN.2019.8911784.
- [21] Z. Jianyong, L. Haiyong, C. Zili, L. Zhaohui, Rssi based bluetooth low energy indoor positioning, 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (2014) 526–533. doi:10.1109/IPIN.2014.7275525.
- [22] D. Giovanelli, E. Farella, D. Fontanelli, D. MacLi, Bluetooth-Based Indoor Positioning Through ToF and RSSI Data Fusion, IPIN 2018 - 9th International Conference on Indoor Positioning and Indoor Navigation (2018) 24–27. doi:10.1109/IPIN.2018.8533853.
- [23] R. Khullar, Z. Dong, Indoor localization framework with WiFi fingerprinting, 2017 26th Wireless and Optical Communication Conference, WOCC 2017 (2017). doi:10.1109/WOCC.2017.7928970.
- [24] M. Estel, L. Fischer, Feasibility of bluetooth ibeacons for indoor localization, Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) 244 (2015) 97–108.
- [25] B. Bonné, A. Barzan, P. Quax, W. Lamotte, Wifipi: Involuntary tracking of visitors at mass events, 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (2013) 1–6. doi:10.1109/WoWMoM.2013.6583443.
- [26] F. Haouari, R. Faraj, J. M. AlJa'am, Fog computing potentials, applications, and challenges, 2018 International Conference on Computer and Applications (ICCA) (2018) 399–406. doi:10.1109/COMAPP.2018.8460182.
- [27] F. Shang, W. Su, Q. Wang, H. Gao, Q. Fu, A location estimation algorithm based on rssi vector similarity degree, International Journal of Distributed Sensor Networks 10 (2014). doi:10.1155/2014/371350.