IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 9, 2019, accepted October 31, 2019, date of publication November 7, 2019, date of current version November 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952221

Energy-Efficient Indoor Localization WiFi-Fingerprint System: An Experimental Study

JOSE L. SALAZAR GONZÁLEZ[1], LUIS MIGUEL SORIA MORILLO[1], JUAN A. ÁLVAREZ-GARCÍA[1], FERNANDO ENRÍQUEZ DE SALAMANCA ROS[1], AND ANTONIO R. JIMÉNEZ RUIZ[2]

[1]Departamento de Lenguajes y Sistemas Informáticos, University of Seville, 41012 Sevilla, Spain
[2]Centre for Automation and Robotics (CAR), CSIC-UPM, 28500 Madrid, Spain

Corresponding author: Jose L. Salazar González (jsalazar@us.es)

This work was supported in part by the Spanish Ministry of Economy and Competitiveness through the VICTORY Project under Grant TIN2017-82113-C2-1-R MINECO/FEDER R&D, UE, and in part by the Spanish Ministry of Science, Innovation, and Universities through the MICROCEBUS Project under Grant RTI2018-095168-B-C55 MCIU/AEI/FEDER,UE.

ABSTRACT In order to apply indoor localization systems in real environments it is necessary to provide an accurate location without implying a high impact on the user's normal behaviour. To achieve this goal, in this paper, a combination of battery saving techniques with a system based on WiFi fingerprinting is proposed. This is done by transferring the location calculation workload to the server, leaving user's mobile devices the only responsibility of making periodic WiFi network scans at dynamic intervals based on user activity, through an application running on background. There are not many studies analyzing energy consumption of existing localization systems, even though it is an important factor in real applications. In this paper, both energy consumption and accuracy are analyzed, having an energy consumption of only 0.8 Wh (having a 3.7 V battery) during a 24-hour cycle and an average localization error of 4.51 meters. Worth to mention that as computation is done on server side the system can be expanded to multiple buildings and floors. Finally, the dataset used in this paper has been published making possible to test new algorithms in the same environment.

INDEX TERMS Indoor localization, WiFi fingerprinting, RSSI, battery life, KNN, naive Bayes, dataset.

I. INTRODUCTION

The energy-efficient indoor localization proposed in this study is framed in the context of a security project,[1] in which the main objective is to reduce the impact of a possible threat inside a building. For this goal, once the threat is identified, the system guides users in a dynamic and personalized way, using their own mobile phones, providing the actions needed to keep them safe (e.g. finding the most secure exit route, not necessarily the closest one, which could be blocked). That is why it is necessary to know at all times not only where the threat is, but also the people in the building, so that they can receive the appropriate dynamic indications. Thanks to the application described in this paper, focused on the part of indoor localization, it is possible to locate each user individually inside the building and at the same time do not significantly affect phone's battery drain.

Due to the context in which this study is framed, there are a number of restrictions necessary for the execution of the presented project, namely: 1) the server must know the location of each user, in order to provide a customized escape path; 2) the system must allow to add new facilities in real time to localize users in new areas; 3) the system must be energy efficient in the user's device, not being the responsibility of the user the resolution of the location. To this end, the study will focus on inferring locations from the server side, leaving the client side only to efficiently send information to the server.

The main goal is to implement an energy efficient and scalable localization system. Therefore in this paper it is proposed and presented a fingerprint-based localization system with, on one side, novel energy-efficient techniques that adapt the scanning frequency to the user's physical activity to save

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Omer Farooq.

[1]VICTORY: VIsion and Crowdsensing Technology for an Optimal Response in physical-securitY. http://madeirasic.us.es/victory/

energy, and on the other, a server-based localization that takes all the data processing, leaving the client with only the requirement to perform and send WiFi scans to the server, allowing also to update the localization algorithms in real time and without energy cost in the user devices.

In order to get the most energy-efficient localization system given our targeted scenario, this work has taken into account the energy consumption as well as the accuracy of the system. Both are aspects that have already been studied as mentioned in the state of the art in section II, but whose join combination (energy minimization and location accuracy maximization) has not been studied before as far as we know.

It should also be noted that there are two restrictions in order to deploy our system in the real environment. Our first constraint is that no additional infrastructure can be installed in the building, which in our case is a university building, hence all infrastructure-based technologies must be discarded, such as deploying customized RF-beacons, RFID, infrared, ultrasound, Bluetooth, short-range FM transmitters, lights, or magnetic signal modulators [1]. As there are enough WiFi access points (APs) available to provide internet coverage throughout the building, existing WiFi APs will be used in their current distribution, which cannot be modified either. The second restriction is that the router software can neither be accessed or modified, it is only allowed to be connected like any other user, so the location will be done by the client, passively obtaining the WiFi received signal strength indication (RSSI) that arrives from each AP during client scanning process through the application of the system, and communicating it to the server to predict the position of the user. It also does not support round trip time (RTT), so only RSSI should be used as a reference of the location. Due to these constraints, we do not intend to compare ourselves with the best existing indoor localization system in terms of accuracy, but rather to achieve a scalable and energy efficient system that can be successfully implemented in a real environment, in which will take advantage of existing APs RSSI.

Since localization algorithms are run on the server side, the different localization algorithms can be used without affecting user's devices. In this work, regression KNN, classification KNN and Naive Bayes localization algorithms are tested and compared, showing the results using our collected dataset. This set of experiments included in our dataset has been published so that other researchers can reproduce our experiments and test their solutions on it.

In this paper, four innovations are proposed, namely: 1) the development of an indoor localization system in a real case, this being within an university building, deployed and prepared to give locations to multiple users simultaneously and add additional buildings or floors; 2) an extensive comparison between regression KNN, classification KNN and Naive Bayes applied to indoor localization using WiFi fingerprints; 3) the implementation and study of a combination of state-of-the-art techniques to significantly reduce the battery consumption caused by the localization system; 4) the release of a manually collected WiFi fingerprint dataset at the university described in this study.

The rest of the paper is organized as follows. In Section II the related works in fingerprinting and energy saving are presented. Section III describes the proposed energy-efficient localization system. Section IV describes the experimental set-up. Section V shows the results related to localization performance, comparing KNN and Naive Bayes approaches, and presents the energy results with the proposed saving techniques. Finally, Section VI concludes this study.

II. RELATED WORKS IN FINGERPRINTING AND ENERGY SAVING

Over the last few years several studies have been published with the purpose of finding the most suitable solution to provide an accurate localization inside buildings, also reducing the deployment and maintenance costs. Below some of the most relevant are shown, first from the point of view of fingerprinting and then those related to energy consumption.

A. FINGERPRINTING STATE OF THE ART

Different techniques have been studied in order to provide an accurate position at indoor areas. One of them, is fingerprint-based technique, that first measures and stores all the APs RSSI received at different points on the map. Afterwards, the possible locations are predicted by comparing the RSSI values of the unknown location with those already measured.

There are different research-level and commercial solutions for indoor localization that have been reviewed in several papers [2]–[4]. From the multiple approaches (RF beaconing, inertial, ultrasound, light, etc.) the fingerprinting approach is one of the most extended. For example, Lymberopoulos and Liu [5] compare the ones presented to Microsoft Indoor Localization Competitions from 2014 to 2017, which include many techniques based on Fingerprint.

Two of the most widely used techniques with WiFi fingerprint are the following [6]: techniques for comparing captured fingerprints with new scans, either by calculating the similarity between them or with techniques such as K-Nearest Neighbor (KNN); and techniques that use captured fingerprints to resolve a Log-distance path loss (LDPL) model with physical restrictions and wireless propagation, hence reducing the number of fingerprints required.

The solutions based on comparison of fingerprints are the most frequently used until now because of the cost of implementation and the high accuracy of them. Niu et al. [7] and Hossain et al. [8], among others, use KNN or Naive Bayes technique to determine unknown locations. They propose to augment the fingerprints, achieving an improvement of up to 4 meters approximately using Naive Bayes for datasets with a low number of real points, however it does not improve as much in larger datasets, as it also increases the inference time, being this crucial to provide locations in real time.

On the other hand, there exist solutions that base their systems on fingerprint with LDPL models. Even though it is common to use LDPL models without the use of fingerprints,

studies such as Chintalapudi *et al.* [9], Lim *et al.* [10] or Ji et al. [11], since LDPL models are theoretical, do not always obtain the best results in real environments, yet it reduces the number of fingerprints required.

Even though in recent years we have found more applications of Deep Learning to location systems, these systems require a time-consuming fingerprint capture process, as these models require a high number of fingerprints in order to accurately infer the result. Nevertheless, very precise results can be achieved, such as the solution proposed of Abbas *et al.* [12] who filter the fingerprints and add noise to improve accuracy, or the study of Chen *et al.* [13] who propose a local feature-based deep long short-term memory (LF-DLSTM) approach for reduce the signal noise.

Our proposal makes use a solution based on comparison of fingerprints, exploring classification KNN, regression KNN and Naive Bayes, as it achieves high accuracy when it uses real fingerprints, and allows updating the dataset and facilities in real time without the requirement to recalculate models.

B. ENERGY SAVING STATE OF THE ART

Implementing a localization system in a mobile phone involves the use of its internal sensors, which is an action demanding significant energy from the battery. Therefore, it is an important goal to solve the problem of high energy consumption to avoid the rejection of installing battery-draining App by users. For this reason, in most real systems, the energy saving is as significant as its accuracy.

There are some solutions to solve this energy-efficient problem in continuous user localization. Wang *et al.* [14] perform a study of energy saving techniques in existing mobile crowdsensing, from raw sensing gathering, passing through inference to data aggregation. A recent survey, in the context of fingerprinting localization, includes a review in energy saving methods [6]. He et al. categorizes three main approaches to save energy: reducing the scanning frequency [15], [16], reducing the number of AP's used [17], and replacing WiFi scanning by other RF technologies such as Zigbee. The use of server-side indoor implementations to deleverage the load on the mobile-phone is proposed by Gao *et al.* [18].

A common solution to save battery, that propose, among others, Constandache *et al.* [19] and Lin *et al.* [20], is to change the precision of the system according to the need in each moment, such as, if the user is outdoors and does not need a high precision, localization based on GSM can be used, however if he/she needs a higher precision, localization based on GPS should be used, and if the user is indoors WiFi, Bluetooth or camera according to the required precision can be used. For this reason, this solution is useful for systems that combine indoor and outdoor localization, where precision is not always a determining factor, and therefore, it is possible to use one sensor or another depending on the moment and location. This achieves energy savings of up to 45% according to Lin *et al.* [20].

The energy consumption in the device is given by the use of the sensor, as it has just been seen, but it is also influenced by the scanning frequency. As a consequence, it is possible to reduce the scans made by the sensors in order to obtain information from them only when necessary. A number of solutions already studied are based on: using the accelerometry sensor to adapt the scanning frequency of this sensor to the activity being carried out by the user, as proposed by Viet *et al.* [21], or the solution proposed by King and Kjærgaard [22] based on switch between lightweight monitoring and invasive active scanning to allow positioning and to minimize the impact on the data flow; adapt frequencies from other sensors, such as WiFi, by exploiting user movement information and analysing the concentration of APs, as proposed by Kim *et al.* [23]; or predict routes using existing user data to perform fewer scans, combining GPS with WiFi, as proposed by Thiagarajan *et al.* [24].

In addition, another important energy consumption in mobile devices is the use of the CPU, if there are several mobile applications that make use of sensors, can lead to a desynchronization, that causes the device to be continuously using the sensors or waking up the CPU when it is not in use. For this purpose, there are studies, such as Lane et al. [25], that propose a technique called Piggyback that consists of exploiting the use made by other mobile applications of the sensors, the CPU or data transmission, in order to exploit opportunities of usage, or as Moamen and Jamali [26] propose to group the queries made to the sensors for handling on a single application.

Our proposal to reduce energy consumption combines the idea of Viet *et al.* [21] and Kim *et al.* [23] to adapt scans to the user's movements, as well as the approaches of Constandache *et al.* [19] and Lin *et al.* [20], except that in this case we do not change the precision of the system, since our system only locates indoors. Nevertheless, if the system detects that the user is not in the building, it stops the localization service and reduces the search for APs by lowering the sample to scan once every 10 minutes.

III. PROPOSED ENERGY-EFFICIENT LOCALIZATION APPROACH

This section will describe the architecture of the proposed system, the strategies followed to save energy on the device, the communication between client and server, and how the localization on the server is obtained.

A. SERVER/PHONE LOCATION ARCHITECTURE

The system consists of two elements, namely: 1) a mobile application, to capture the training and testing datasets and perform passive scans by the user; and 2) a server with an API deployed with Flask [27], where locations are inferred by the localization algorithm, saved in a MongoDB [28] database and transmitted to the user. The figure 1 shows the flow between the mobile application and the server to communicate the required information and infer the location.

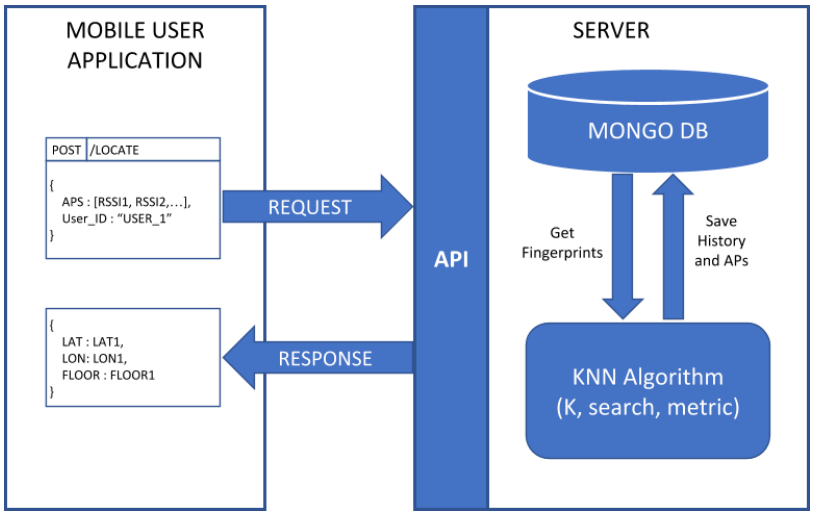

FIGURE 1. Server/phone architecture-architecture diagram.

First, the user's mobile device performs passive scans of WiFi networks when the energy-saving algorithm considers it opportune. Afterwards, the application performs a secure POST request to the server in JSON format with the user details and the scan result. This request is received by the server, which transforms the result of the WiFi scan to be passed later by the localization algorithm. Finally, the user's details, the result of the WiFi scan and the location provided by the localization algorithm are stored in a non-relational database, to show the user a history of their locations and subsequently carry out any necessary analysis, and the user is answered with the location obtained by the localization algorithm in JSON format, indicating both the geographic coordinates and the floor plan of the building.

The user can retrieve, at any time, his last location and a history of it from the mobile application since, as mentioned above, it is stored on the server.

In addition, the system is prepared to detect and report falls. First capturing the fall through the mobile application, analyzing the accelerometer signal, sending it later to the server and notifying through sockets by Socket.IO [29] to the web dashboard of security staff, or other mobile devices through notifications by Firebase Cloud Messaging [30] to issue alerts or request scans of a specific area in case of threat.

B. FINGERPRINT BASED LOCALIZATION PHASES

In order to prepare the algorithm for accurate localization, the system must go through five learning and tuning phases:

- *Survey*. Experimentation to register the training dataset and upload to server,
- *Training*. Off-line algorithm training and tuning for performance improvement,
- *Deployment*. Implementation of on-line localization algorithm on server side,
- *Test*. Test the localization system generating a positioning error map, and
- *Validation*. Validate the localization system with multiple users along several days.

First of all, a survey is made of the area in which the localization system will be implemented by manually selecting every 3 meters on average the current location on the mobile device and performing a number of scans that will be sent to the server to generate the training dataset, through an Android application developed for this purpose. Figure 2 and 3 show the manual selection on the map and WiFi scanning process respectively.

Subsequently, these data are off-line processed and the algorithm is trained on the server along with the stored data in the previous step, performing cross validation to decide the best parameters for the adjustment of the algorithm as shown in subsection III-E.

Afterwards, the algorithm trained from the previous step is deployed on the server with the Flask API, mentioned in subsection III-A.

Once the model has been trained and deployed, a manual test of the system is carried out. In this test, the area selected for the localization system is covered and every three meters on average the location is indicated in the application, the only difference between survey and testing phase is that a single scan is carried out with an initial waiting time of five seconds instead of performing multiple scans. For each scan at this stage, a WiFi scan and location request are first made as if it were a real scenario, only this time the manual location entered is included. Afterwards, the server calculates the error, saves the result and returns it to the app to display the average error on the screen, as it can be seen in Figure 4. When all the area is tested, a testing dataset useful to compare and adjust the algorithm with new data, is obtained. Moreover a map with the error at each point is generated to find areas with less precision.

Finally, a validation process is performed between different groups in order to evaluate the system in a real environment

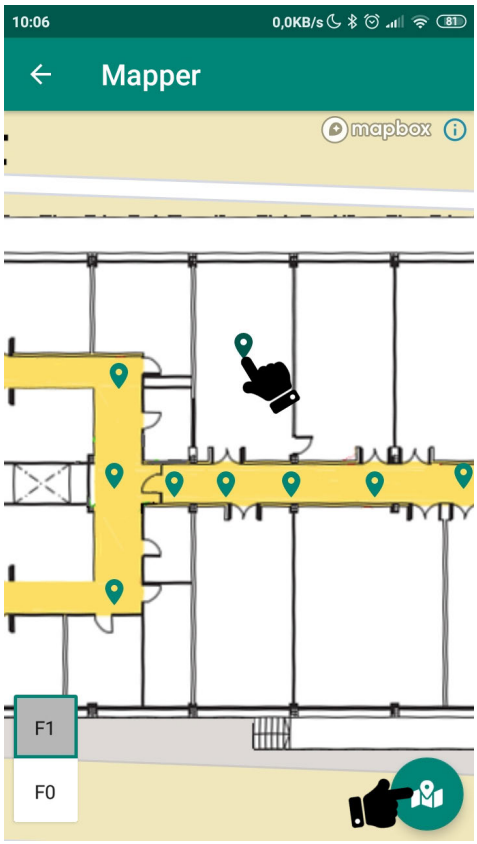

FIGURE 2. Survey phase-selection of the actual position.

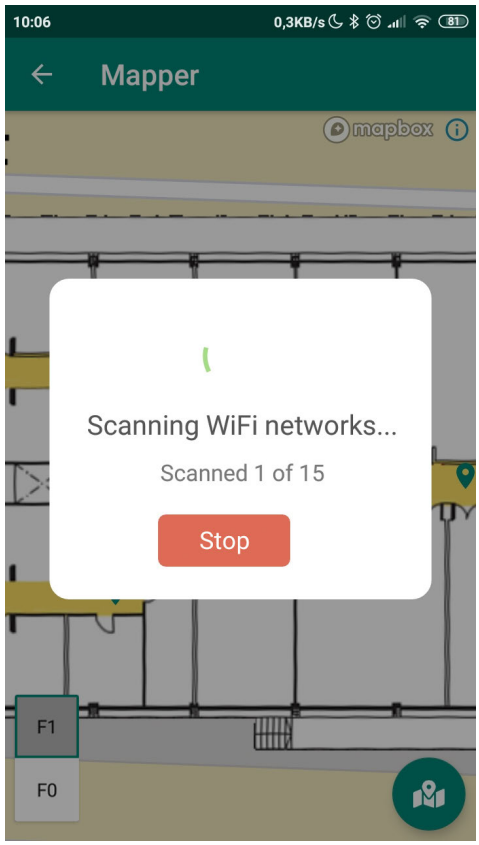

FIGURE 3. Survey phase-WiFi scanning to get WiFi fingerprints.

for a certain number of days with different users. This process will be described in more detail in the following section.

When these phases are completed, the system is ready for a real deployment, inferring locations of all users through the mobile application doing scans with dynamic energy-efficient intervals of maximum 15 seconds, this scanning strategy will be described in the following subsection.

C. SCANNING STRATEGIES FOR ENERGY SAVING IN A PHONE

The last generation of mobile devices usually make intensive use of sensors, processors, wireless connections and screen, which leads to high battery consumption. Many manufacturers choose to introduce customization layers on the operating system itself. These layers of customization, which are discussed in more detail in the results section, are quite aggressive with those applications that record high battery consumption.

While it is true that certain applications are part of the white list of manufacturers (eg. WhatsApp, Facebook, Instagram or the suite of Google applications), it is not the case of the background-executed application described in this article. Therefore, it is essential to reduce as much as possible the consumption of the application with the objective that the user perceives the least possible impact and, on the other hand, that the 'application killers' do not kill the process in which the positioning algorithm is executed.

To avoid that our application can be killed, four variants have been implemented that allow reducing this energy impact based on the detection of activity and the probability that the user is inside a place where the positioning system is available.

To evaluate more accurately the differential consumption between the non battery-optimized method presented in this paper and the different battery saving techniques optimisations, a simulator that recreates the execution environment of the positioning algorithm has been implemented. This will allow to isolate certain undesired operating system processes that can be executed in the background and, in this way, prevent other processes from interfering with the study of our application and the different optimization.

Simulation allows to analyze the behaviour of the application in the long term, avoiding interference with other processes and analyzing this consumption in an accelerated manner. This simulation made it possible to identify the consumption over several weeks in a totally realistic manner, taking into account only the process of localization and the consumption of the sensors needed to do so.

For the simulation implementation, a process has been developed in which the necessary connections to the different external services, geolocation, search and processing of WiFi

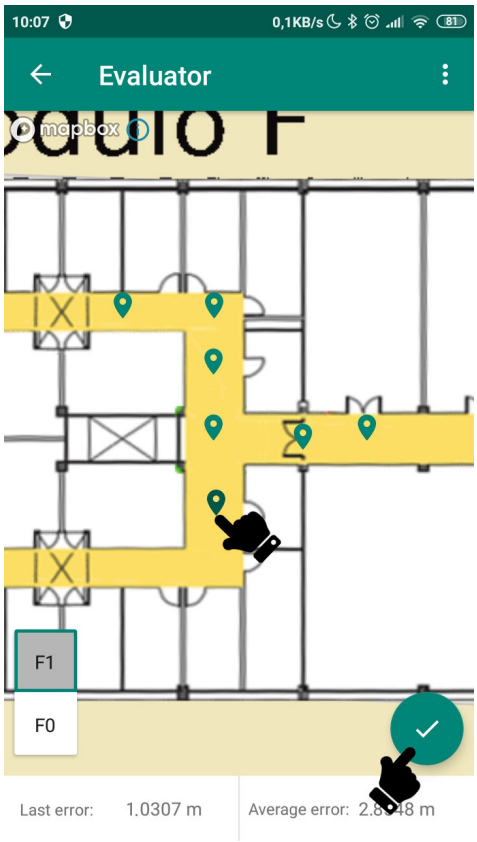

FIGURE 4. Test phase-selection of the actual position.

signals will be simulated. These tasks are really important when evaluating the expected application consumption.

In addition to these simulations, the consumption measured on real devices during the testing process were also taken into account when analyzing the energy cost of the localization process. Thus, these simulations have been calibrated and simulated consumption is accurate with real values.

In any case, the simulator has been developed with the aim of maintaining pessimistic behavior. This ensures that the energy consumed by the location process in the simulator is greater than that required in a real device. In this way, it is possible to obtain energy consumption results that show an accurate enough baseline to affirm that the battery usage time in a real device will be, at least, that shown in the results of this comparison. For this purpose, a series of random consumption variables have been introduced, simulating the behavior of the processes that the operating system performs in the background. These variables follow a normal distribution, similar to those analyzed in the background system tasks [31].

The energy saving techniques implemented in this work are described below. The results of all of them and their impact on both accuracy and consumption will be presented later in the results section.

1) NO BATTERY SAVING (ALWAYS RUNNING)

This method is the one described above in the proposed solution section. In it, the system is scanning every 5 minutes wireless networks in the device current position. This allows to determine if there are identified networks in the labelling process and, in this way, to execute the localization algorithm. The accuracy of this method is high, as is its response rate, although its execution has a slight impact on battery consumption and, more importantly, it is always running.

2) ENERGY SAVING TECHNIQUE BASED ON AP LOSSES

Starting from the previous algorithm, a saving method based on the detection of nearby access points has been implemented. In case there are no access points compatible with the labelled places, a duty cycle technique is carried out. This technique is based on the dilation of the scanning periods according to the circumstances. In this way, if at any given time the algorithm, when conducting a scan of nearby APs, determines that there is no recognized APs in labelled place, the time elapsed until the next scan increase in 3 minutes (duty-cycle concept). This allows to reduce the number of scanning processes when the user is in an unlabelled place until reaching a minimum of one scan every 15 minutes. At the moment in which a known network is found, the period will be reduced to the initial scanning time. This technique has been previously used in the context of outdoor positioning with the GPS system. In this case, the duty-cycle occurs when the device loses connection with satellites. This usually indicates that the user is indoor and, therefore, has no visibility with the GPS positioning system. To avoid unnecessary battery costs, applications or operating systems directly introduce these rest cycles. In this study, an adaptation of this procedure is carried out considering the WiFi signals as beacons to determine whether or not the user is inside an area with the system deployed.

3) ENERGY SAVING TECHNIQUE BASED ON DISPLACEMENTS

Finally, as a basic method of saving energy, the displacement-based energy saving is used. In this case, the system will only scan WiFi networks when the device is moving. This displacement will be determined thanks to the accelerometry sensors installed in the device itself. In our development, having used Android as operating system, the implementation relies on the step counter virtual sensor to carry out this task. Specifically, the system defines ranges of 60 seconds in which is checked whether or not there has been a displacement during this interval. In this way, if during the last minute there has not been any displacement, a trigger is programmed that will stop the WiFi scanning after 10 minutes. If during those 10 minutes at least one step is detected, the trigger is removed and the system continues with normal operation. This technique, in its original version, is used in certain monitoring wristbands (for example with the Xiaomi miBand 4) to reduce the consumption of these wearables. In the basic version, the non-detection of steps means entering a mode of reduced consumption that saves battery power by disconnecting different software and hardware modules from the device. This work applies this concept to the WiFi networks

searching environment for subsequent trilateration. This technique allows reducing consumption by assuming that when there is no displacement, it is not possible to change the position (just as a wearable assume that without displacement there is no activity).

4) COMBINED ENERGY SAVING TECHNIQUE

Finally, the two previous optimization techniques were applied in parallel. Thus, the application enters the low consumption mode when it does not detect displacements, in addition to adding the duty cycle when there are no access points recognised in the surroundings.

D. COMMUNICATION PHONE/SERVER

As mentioned in the subsection III-A, a communication is established between the user's mobile application and the server to transmit the necessary data needed to infer user's location. Therefore, it is required to carry out a data capture process previous to the first location request, collecting a certain number of samples of the available APs signal intensities for each location. Each of these signal samples, captured at a particular location, will be called a WiFi fingerprint.

A WiFi fingerprint, denoted as F at the equation 1 with the position j at database, is composed by a set of received signal strength indicators (RSSI) for every access points (APs). The RSSI indicator for the AP at the position i is denoted as s_i, the range of i is from 1 to m, being m the number of APs at database, if the AP at the position i is not found on the scan, the value of s_i will be zero.

$$F_j = (s_1, s_2, \ldots, s_m) \tag{1}$$

RSSI is measured in decibels, and according to Friis path-loss equation for free-space radio signal propagation the lower the value of the RSSI, the further away the AP is from the user. Despite the fact that the range can vary between manufacturers, and that fading interference appear, if the RSSI values of the APs available in that location are captured, a fingerprint can be obtained. Fingerprints at a given location, combined with much more scans in different locations in a given space, create the database needed to infer the desired unknown locations in the on-line operation. The fingerprint is denoted as F_j, where the range of j is from 1 to n, the total number of fingerprints. This set of n fingerprints, forms our WiFi fingerprint database as represented in equation 2:

$$F = (F_1, F_2, \ldots, F_n). \tag{2}$$

E. LOCALIZATION APPROACHES ON SERVER SIDE

As mentioned in the subsection II-A, there are different types of algorithms to solve this problem. Our solution will make use of KNN algorithm, since, as can be seen afterwards, better results and performance were obtained with the data collected during the experimentation. In addition, it allows to apply changes in the dataset without the need for expensive training process, since KNN is a Lazy Algorithm, it only needs to store the feature vectors and labels of the classes.

In order for the algorithm to obtain better results, a preprocessing was applied to the data, scaling and moving each RSSI individually to a range between zero and one, normalizing every RSSI value, taking into account the maximum and minimum value presented at each AP over the training dataset, since this range can vary. A device scaling was not undertaken due to single device training, although it could be scaled per device as more data is obtained from other devices. When loss of accuracy with low RSSI has been found, it was tried to discard RSSI signals below a certain threshold, as indicated by Gansemer *et al.* [32], yet it made the accuracy lower in more distant areas from the APs. However, the solution that did improve these results was, instead of eliminating signals in the WiFi fingerprint according to a threshold, eliminate APs with a standard deviation of less than 5dbm, since they worsen the accuracy and do not provide relevant information.

The KNN algorithm can be used for both classification and regression. In both cases, first the distances between the scan characteristics of the unknown location and the stored fingerprints are calculated. These distances can be calculated using a number of metrics, such as Euclidean, Manhattan and Mahalanobis distances.

Torres-Sospedra *et al.* [33] made a comparative between 51 metrics, and they obtained the minimum error with the Sorensen metric, given by the equation 3, where: $d(x, F_j)$ denotes the distance between the WiFi scan result x (RSS value) and the WiFi fingerprint F_j at the position j; m is the number of all APs at the database; x_i is the RSSI value with AP i after the WiFi scan; and F_{ij} denotes the RSSI value within the WiFi fingerprint for AP i at the fingerprint position j.

$$d(x, F_j) = \frac{\sum_{i=1}^{m} |x_i - F_{ij}|}{\sum_{i=1}^{m} (x_i + F_{ij})} \tag{3}$$

In our case this metric (equation 3) has a high accuracy but, as shown in the next section, it does not get the best result and it is more time consuming than others. Therefore, after comparing the metrics within our test dataset, Manhattan distance was chosen experimentally since it presented more accurate results.

Besides choosing the metric to calculate distances between characteristics, it is necessary to choose a search algorithm to find the nearest K-Neighbors efficiently. To this end, there are different algorithms, such as: Brute Force, which performs a brute-force calculation between all point pairs in the dataset; Kd-Tree, based on the data structure with the same name, which organizes points in a K-dimensional space by partitioning data on the Cartesian axis; and Ball-Tree, based also on the data structure with the same name, which like Kd-Tree organizes points in a multi-dimensional space, only that Ball-Tree partitions the data into a series of nesting hyper-spheres.

After finding the nearest K points, understanding nearest as the lowest distance using the metric chosen over the characteristics, the result will depends on the type of KNN used: if a classification algorithm is desired, the predicted location

is given by the location of the most repeated one among the selected k-fingerprints; on the contrary, if a regression algorithm is preferred, the location will be the average value of the selected k-fingerprints. These k-fingerprints may also have different weights, such as uniform weights or distance-based weights, applied during the selection process.

Choosing between classification and regression is an important factor in this process, and opting between them depends mainly on the pre-training phase and the domain of our problem. If scans are carried out at close distances, a classification model could provide locations for almost the entire building, since sufficient scans have been carried out for this purpose, guaranteeing that the location will be located in controlled and accessible areas, since only captured locations will be used. On the other hand, if the scans do not cover the majority of the map or are made with higher distances, the implementation of a classification model may lead to a decrease in accuracy as there are many uncovered locations, however, if a regression model is used, the location will be given by the average of the closest k-locations, resulting in non-considered locations, which may be closer to the real position.

As Torres-Sospedra *et al.* [33] mention, most studies propose a 1-NN approach. The choice of k-value is another important factor in this process, since a low value may lead to inaccurate and unstable positions, a high value could consider too many locations and not accurate either. Using a 1-NN will be useful if one WiFi scan per location is compared, at classification KNN or, as Yim [34] proposes, if the location is represented as the average of the scans performed on it. Therefore, on the one hand, reducing the k-value and the training set has the advantage of smaller data size, lower processing requirements and therefore higher speed. On the other hand, not considering all the possible RSSI values that can be obtained for that location may become a problem if it fluctuates enough. In our case, the k-value was empirically chosen by testing different values on the test set for the optimum ratio of precision and performance.

IV. EXPERIMENTAL SET-UP

This section will describe the building used to train and test the system, the structure followed for the datasets, the validation process and control groups.

A. DESCRIPTION OF THE TESTING BUILDING

The experiments were carried out on the first and ground floors of the Escuela Técnica Superior de Ingeniería Informática, in Seville, Spain. The facility has 24.000 m^2 approximately, although only accessible areas were studied.

The facility has 72 Cisco Aironet access points distributed throughout the different modules and exterior corridors, of which 48 are between the first and ground floors, the rest are in upper floors or basement, which will not be represented in the maps as the locating system does not extend to these areas. The APs, marked black, are distributed by the different modules, some within classrooms and others in corridors, the location of these can be seen in the figures 5, 6, 7, 8, 22 and 23.

The distribution of access points on the facility has the goal of giving WiFi coverage to all areas to provide Internet access. Hence, they are not specifically designed for this experiment, nor for any other use of the network other than simple access to the network, meaning a decrease in accuracy. As the number of these was not enough, unknown APs whose range reached the facility were also used. This represents a total of 460 APs, of which only 160 had a standard deviation of at least 5 dBm, over all the survey phase, this value was chosen for this dataset, although it could differ in another environment. Using this filter, it is possible to ignore distant APs that do not provide relevant information to the fingerprints, since, as Moc and Retscher [35] indicate, the relationship between RSSI and distance may be expressed mathematically as the logarithmic function of distance, thus the further we move away from the AP the less the RSSI fluctuates and therefore the less information is given. In addition, by ignoring these APs it is also possible to improve inference times by avoiding unnecessary computations.

As mentioned in the subsection III-A, in order to be able to predict locations, a pre-training process has been carried out, in which the facility is covered and every certain number of meters a number of scans are obtained. For the purpose of this study, 15 scans were gathered every 3 or 5 meters, these values were selected according to a trade-off of accuracy and cost due to their dimensions. These scans were performed using an application installed on a BQ Aquaris E5 4G, which transmitted all captured RSSIs along with a location marked manually on the facility map to the server which stored these data in a database for further processing and implementation.

B. DATASET DESCRIPTION

The training dataset consists of 7175 fingerprints collected from 489 different locations (almost 15 fingerprints per location). Each fingerprint is stored as a JSON object corresponding to an unique scan with the following values:

- _id: contains an unique identifier for the fingerprint, uses to differentiate one fingerprint from another.
- avgMagneticMagnitude: average magnetic magnitude during scanning with the mobile phone sensor, although this value is not used is provided in case it was useful.
- location: object with the coordinates of the real world in which the sample was captured.
  - floor: number indicating the floor in which the sample was captured.
  - lat: latitude as part of the coordinate at which the sample was captured.
  - lon: longitude as part of the coordinate at which the sample was captured.
- timestamp: UNIX timestamp in which the sample was captured.

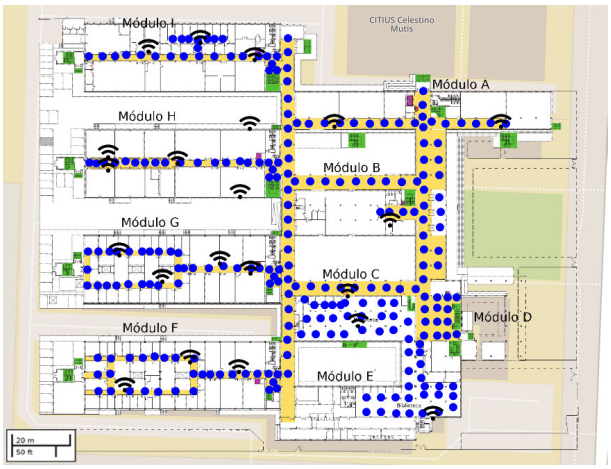

FIGURE 5. Infrastructure-APs and fingerprints at floor 0.

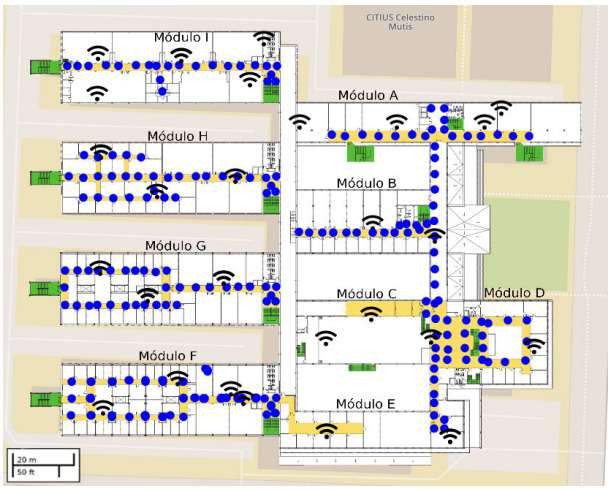

FIGURE 6. Infrastructure - APs and fingerprints at floor 1.

- userId: identifier of the user who captured the sample, this value will be anonymized so that it is not directly identifiable but remains unique.
- WiFiDevices: list of APs appearing in the sample.
  - bssid: unique AP identifier, this value will be anonymized so that it is not directly identifiable but remains unique.
  - frequency: AP WiFi frequency.
  - level: AP WiFi signal strength (RSSI).
  - ssid: AP name, this value will be anonymized so that it is not directly identifiable but can be used to compare APs with the same name.

The training dataset was compiled by taking samples at every 3 meters on average with 15 samples per location. The time at each location was approximately 40 seconds performing consecutive scans with a bq Aquaris E5 4G device using Android stock 6.0.1 without making any movements during the process.

The maps shown in the figures 5 and 6 represent the samples, marked blue, that were taken for the training phase, the samples were taken in accessible areas, excluding private offices and classrooms, as they are usually occupied or closed.

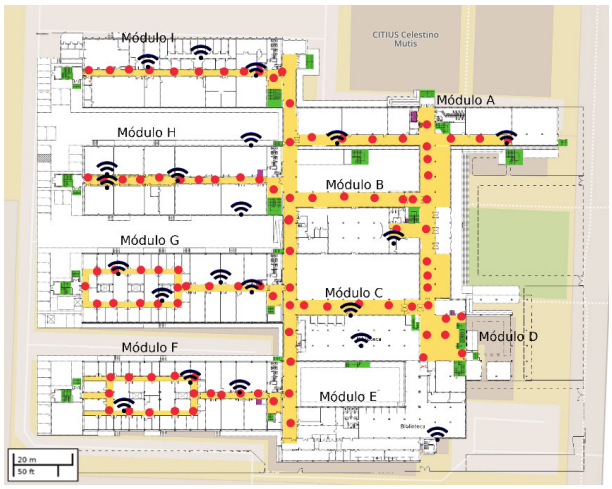

FIGURE 7. Dataset-APs and test locations at floor 0. Corridors are marked in yellow and stairs in green.

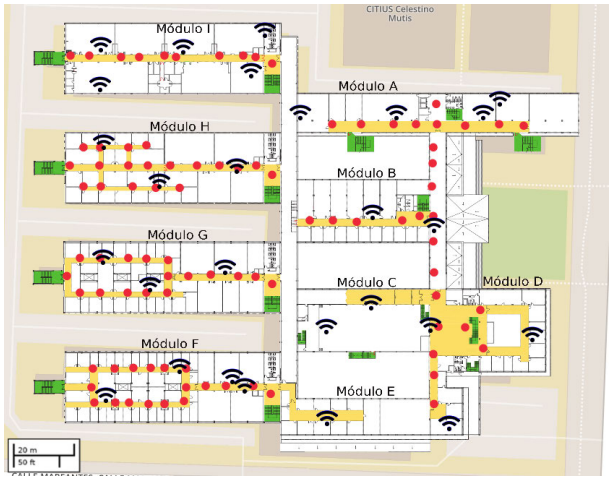

FIGURE 8. Dataset-APs and test locations at floor 1. Corridors are marked in yellow and stairs in green.

In order to test new algorithms, the dataset collected for this study has been published. This dataset can be accessed by following this URL [36].

C. TESTING LOCATIONS AND CONTROL GROUPS

The testing dataset consists of two tests with a total of 390 samples in random locations yet in areas captured by the training dataset and with different devices, the locations of the test samples, marked red, can be seen in the figures 7 and 8. This dataset is grouped by tests and within it are the captured samples, so both the individual error and the average error can be obtained, besides recalculating this error to test different algorithms.

The testing dataset was compiled two days after the training phase by taking samples at random locations with an average of 3 meters, performing a single scan per location.

TABLE 1. Relationship between users and devices.

User Id	Device	Android Version
USER-0	Xiaomi Redmi 4X	Android 7.1.2 with MIUI 10 Global 9.5.16
USER-1	BQ Aquaris E5 4G	Android stock 6.0.1

The samples were taken with two devices, which represent each of the tests individually, a BQ Aquaris E5 4G device using Android stock 6.0.1 and a Xiaomi Redmi 4X using Android 7.1.2 with MIUI 10 Global 9.5.16. Before taking the sample, 5 seconds were waited without making any movements.

In order to provide more information about the device used in each fingerprint of the dataset, the following relationship between users and devices is given:

As mentioned in the subsection III-B, in addition to testing phase, a validation phase is also performed. For this purpose, an experimental group made up of 4 teachers and 6 researchers, who kept the application active for a week in the background and manually confirmed its location when the application requested it.

Afterwards, the application was distributed to a second group made up of 25 students, who left the application active for 2 or 3 days in the background and, as in the previous group, manually confirmed its location. The validations of this group had more variety since they changed classrooms more frequently as students, although they also collaborated less as they had no close link with the project.

Finally, a feedback group made up of 4 teachers checked that the system included the fixes reported by the other two groups and that everything was working correctly.

The evaluation requests for these groups was automatically generated by the application using an activity detection algorithm. This algorithm searched for location changes with a range of 15 meters when at least 15 minutes have been spent in that range, therefore, when this happened, a notification was sent to the user asking for an evaluation of that location, which confirmed or adjusted the location on a map according to the real location. The location shown for evaluation was the average location based on the time spent there.

V. RESULTS & DISCUSSION

In this section the results obtained will be analyzed comparing the different algorithms and parameters used, and the chosen option will be discussed.

A. LOCALIZATION RESULTS

With the collected test dataset mentioned in the subsection III-A, several tests have been carried out to select the algorithm and its configuration that best suits our environment. For this purpose, the use of Naive Bayes, regression KNN and classification KNN has been compared.

In the figure 9 a CDF (Cumulative Distribution Function) is shown comparing the results obtained using Naive Bayes, KNN classification and KNN regression. It can be observed how regression KNN outperforms classification KNN and Bayes. At the 3rd quantile (75% in CDF), the regression KNN error is under 5.90 meters, while classification KNN and Naive Bayes obtain 6.90 and 7.30 meters of error, respectively.

TABLE 2. Classification using Naive Bayes.

Model	Mean Error (m)	Std Error (m)	Milliseconds per location
GaussianNB	5.458296	5.334227	12.198611
ComplementNB	8.713757	7.335323	0.763380
MultinomialNB	5.736507	5.525865	0.731129

Since better results are achieved using regression and classification KNN, we have focused on analyzing and displaying the results of KNN by combining different characteristics, named in subsection III-E:

- Search algorithm: Ball-Tree, Brute and Kd-Tree.
- Distance metric: correlation, Euclidean, Manhattan and Sorensen.
- Weights: uniform or distance-based.
- Number of neighbors: from 1 to 30.

In addition, its CDF, inference time, mean and standard deviation of the error will be analyzed in order to choose the optimal combination for this environment. Although the focus is on KNN, the results obtained using Naive Bayes are also shown.

1) CLASSIFICATION USING NAIVE BAYES

Supervised learning algorithms based on applying Bayes theorem for WiFi fingerprint classification have been implemented. While Naive Bayes is not within the scope of this paper, it has been decided to present the best results obtained with these algorithms in order to compare them with classification KNN results.

Once different models and parameters have been tested, the results of the table 2 are obtained. According to the results, the best average error is 5.45 meters with Gaussian Naive Bayes, followed by Multinomial Naive Bayes with 5.73 meters and Complement Naive Bayes with 8.71 meters. Even though the best average error is obtained with Gaussian Naive Bayes, it is slower, averaging around 12.20 milliseconds per location, while Multinomial Naive Bayes and Complement Naive Bayes spend 0.73 ms and 0.76 ms respectively.

2) METRIC COMPARISON ON REGRESSION KNN

In the figure 10 and in the table 3 can be seen that the algorithm Kd-Tree is the algorithm with the lowest execution time, secondly Ball-Tree with more than twice the execution time of Kd-Tree and finally the Brute algorithm, which is approximately 4 times slower than Kd-Tree.

Afterwards, the average error and the standard deviation of the error will be analyzed according to four types of metrics: Correlation, Euclidean, Manhattan and Sorensen. As shown in the figure 11 and in the table 4, both the mean error

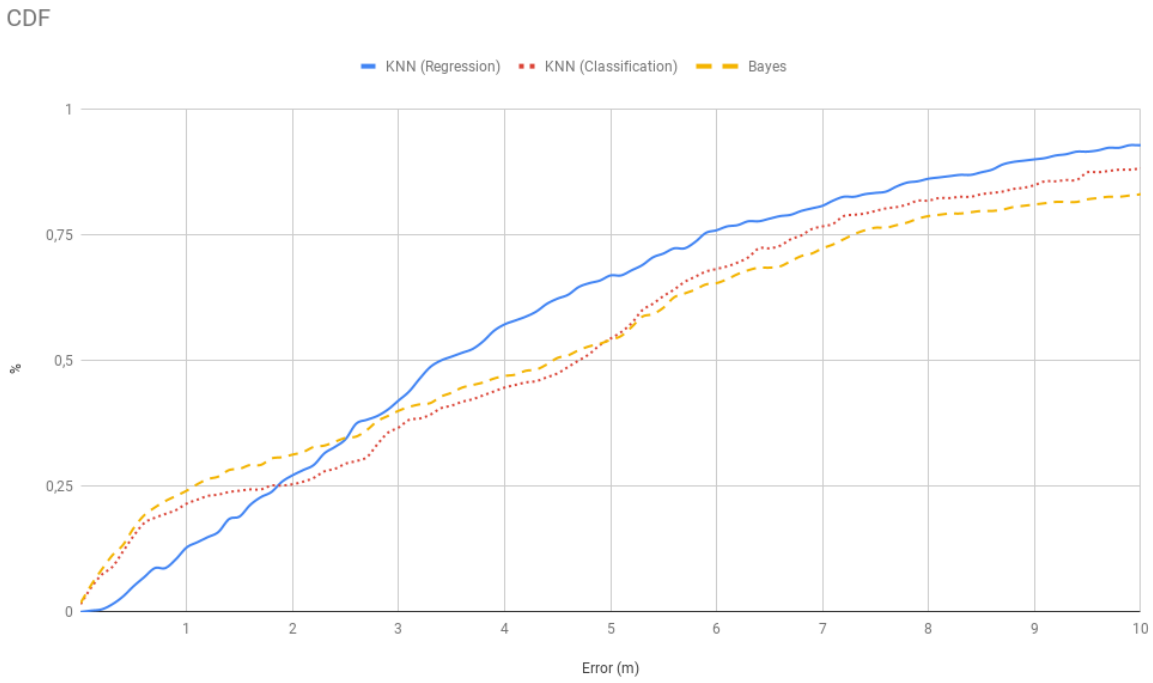

FIGURE 9. Localization results-KNN and Bayes CDF.

TABLE 3. Localization results-algorithm analysis on regression KNN.

Algorithm	Milliseconds per location
Ball-Tree	1.260509
Brute	2.698597
Kd-Tree	0.670763

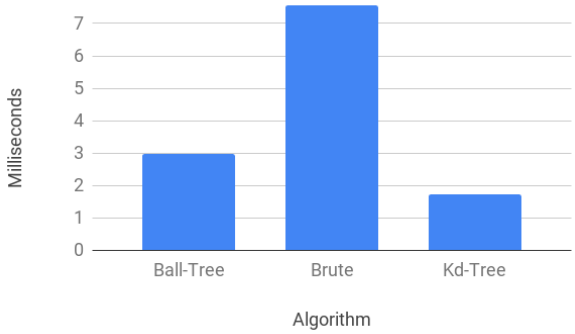

FIGURE 10. Localization results-algorithm time analysis on regression KNN.

TABLE 4. Localization results-metric analysis on regression KNN.

Metric	Mean Error (m)	Std Error (m)	Milliseconds per location
Correlation	5.234330	4.316891	4.486935
Euclidean	5.410298	4.645709	1.248498
Manhattan	4.668422	3.883914	1.241968
Sorensen	4.693356	3.934799	22.052527

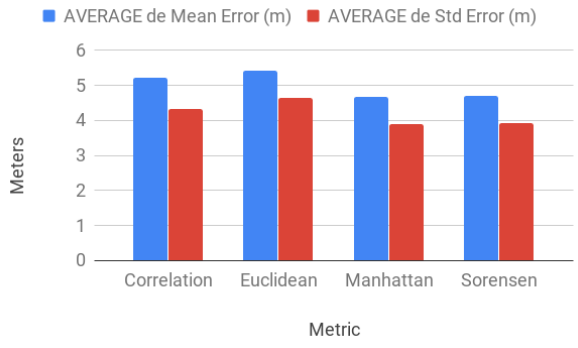

FIGURE 11. Localization results-metric error analysis on regression KNN.

TABLE 5. Localization results-weights analysis on regression KNN.

Metric	Mean Error (m)	Std Error (m)	Milliseconds per location
Distance	5.051915	5.051915	4.585251
Uniform	5.082511	5.082511	3.917465

and the standard deviation are better using Manhattan as a metric, followed by Sorensen with similar results, and finally correlation and Euclidean with more than five meters of error. On the other hand, in the figure 12 and in the table 4 it can be seen that both the Euclidean and Manhattan metrics have the best times, less than 2 milliseconds per execution, whereas Correlation and especially Sorensen have higher execution times.

As the table 5 shows, by comparing the two types of weights it can be seen that both are quite similar, slightly more accurate using distance-based weights although faster using uniform weights.

The figure 13 shows that the average error starts being the highest with $K = 1$, goes down to an optimal point at

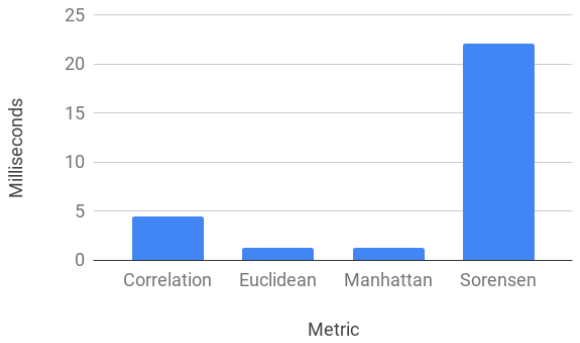

FIGURE 12. Localization results - Metric time analysis on regression KNN.

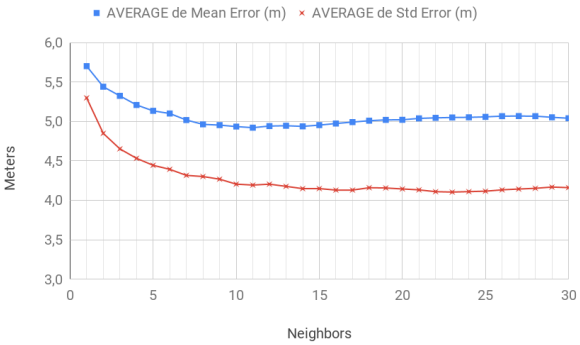

FIGURE 13. Localization results-neighbors error analysis on regression KNN.

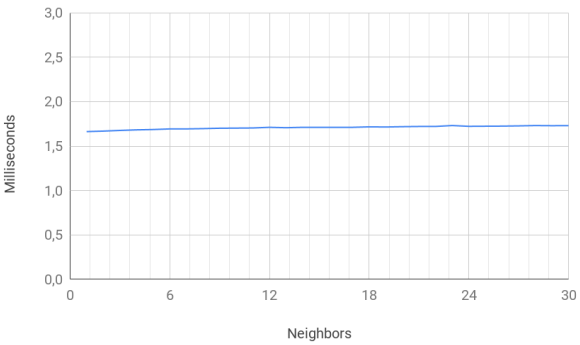

FIGURE 14. Localization results - Neighbors time analysis on regression KNN.

k = 11, rises smoothly, up to approximately 5 meters of error and then stabilizes. On the other hand, has a similar behaviour, except that there is no clear optimal point, the lowest standard deviation is found at K = 23 and then starts to increase smoothly. This does not mean that the minimum error has to be at K = 11, since we are analyzing the mean error between all algorithms, metrics and distances, which means that a combination of these could get the minimum error at a different K. On the other hand, the figure 14 shows that the execution time starts with its minimum value at K = 1 and then rises smoothly.

Finally, the CDF of the metrics used is analyzed with its best number of neighbours, weights and algorithm. The

TABLE 6. Localization results-algorithm analysis on classification KNN.

Algorithm	Milliseconds per location
Ball-Tree	3.754613
Brute	9.946018
Kd-Tree	2.146538

TABLE 7. Localisation results-metric analysis on classification KNN.

Metric	Mean Error (m)	Std Error (m)	Milliseconds per location
Correlation	5.947414	5.510202	11.697531
Euclidean	6.310769	6.108703	3.152675
Manhattan	5.397005	5.058266	3.649058
Sorensen	5.404272	5.095440	43.807494

TABLE 8. Localization results-weights analysis on classification KNN.

Weights	Mean Error (m)	Std Error (m)	Milliseconds per location
Distance	5.801738	5.475009	6.283881
Uniform	5.932758	5.671022	6.275345

figure 15 shows the CDF comparing: Manhattan with K11, brute algorithm and weight based on distance; correlation with K12, brute algorithm and weight based on distance; Euclidean with K11, brute algorithm and weight based on distance; and Sorensen with K14, ball-tree algorithm and weight based on distance.

The CDF at the figure 15 shows two similar groups, on the one hand with lower error probabilities we find Manhattan and Sorensen, followed by correlation and Euclidean, being the latter the worst error, even being similar to the correlation. The lowest error is achieved by Manhattan, which obtains an error of less than 5.90 meters in 75% of the cases (CDF lower than 0.75), followed by Sorensen with an error of less than 6.20 meters, and finally correlation and Euclidean with an error of less than 7 and 7.10 meters respectively.

3) METRIC COMPARISON ON CLASSIFICATION KNN

As in the previous analysis, the Ball-Tree, Brute and Kd-Tree algorithms will be compared, except that this time the classification KNN will be used. The table 6 and the figure 17 show that Kd-Tree gets the best execution time, followed by Ball-Tree and finally Brute algorithm.

The figure 16 and in the table 7 show that Manhattan obtain the best result followed by Sorensen with similar results, Correlation and finally Euclidean distance. Analysing execution time, the table 7 and the figure 18 show that Euclidean get the best execution time, followed by Manhattan, Soresen, and finally Correlation.

As the table 8 shows, again as in regression KNN, by comparing the two types of weights it can be seen that both are quite similar, slightly more accurate using distance-based weights.

The figure 19 shows that both the mean error and the standard deviation behave in almost the same way, with K = 1 it find the best average of the mean error and the best average

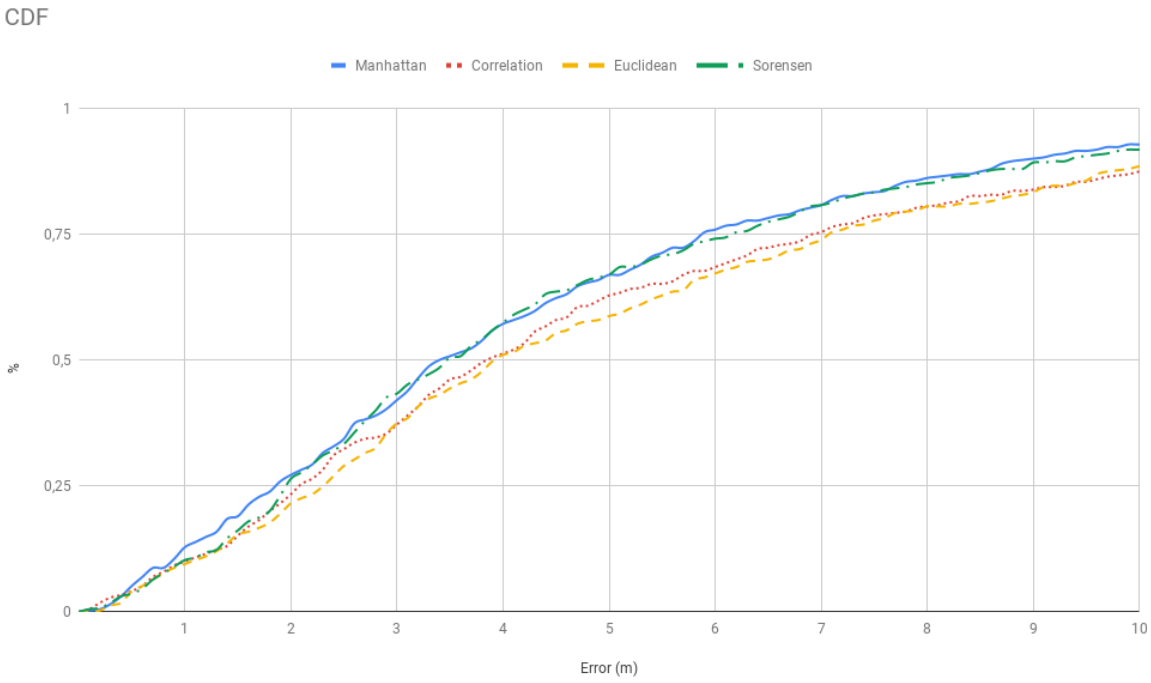

FIGURE 15. Localization results-regression KNN CDF.

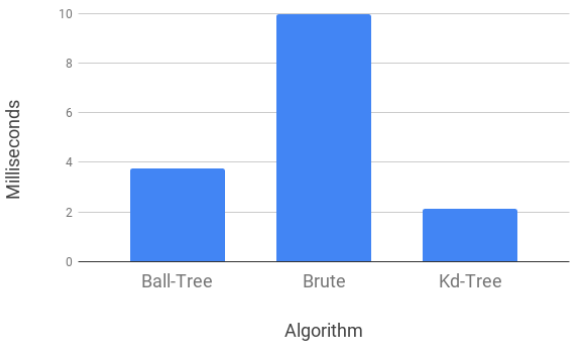

FIGURE 16. Localisation results-metric error analysis on classification KNN.

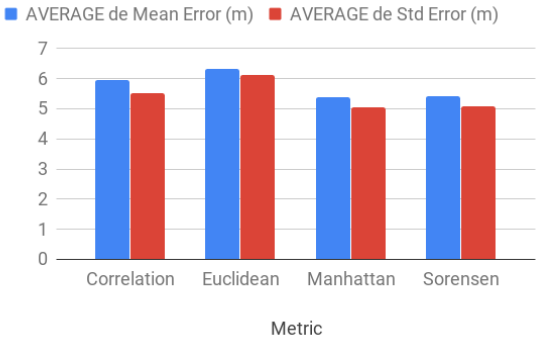

FIGURE 17. Localization results-algorithm time analysis on classification KNN.

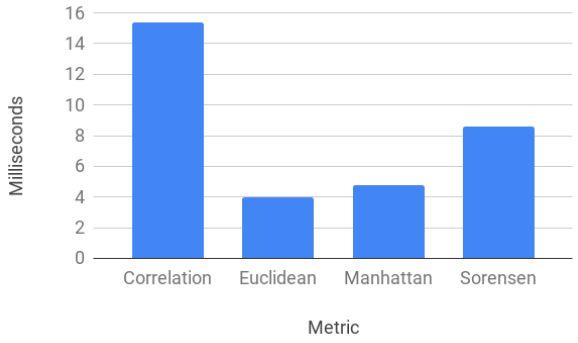

FIGURE 18. Localization results-metric time analysis on classification KNN.

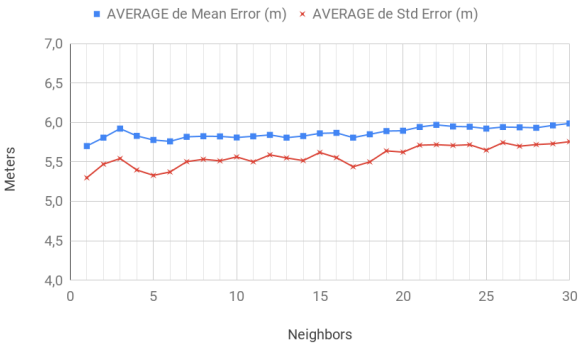

FIGURE 19. Localization results-neighbors error analysis on classification KNN.

of standard deviation, then it continues to rise. From the other side, the figure 20 shows that the mean execution time behaves in the same way, with K = 1 the lowest average of the execution time is found, then it goes up slightly as the neighbors increase.

Finally, the CDF of the metrics used is analyzed again, yet in this case using KNN classification, with its best number

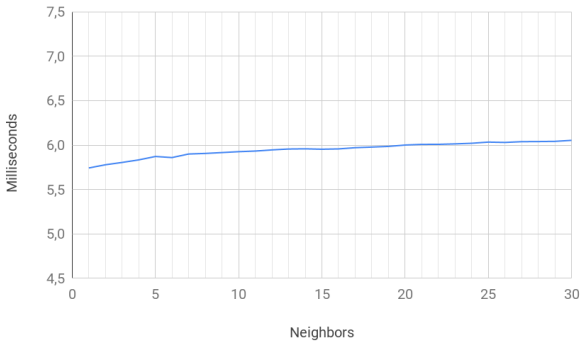

FIGURE 20. Localization results - Neighbors time analysis on classification KNN.

of neighbours, weights and algorithm. The figure 21 shows the CDF comparing: Manhattan with K5, brute algorithm and weight based on distance; correlation with K14, brute algorithm and uniform weight; Euclidean with K1, brute algorithm and weight based on distance; and Sorensen with K6, ball-tree algorithm and weight based on distance.

The CDF at the figure 21 shows again two similar groups, except the metrics within the groups have more similar values here, on the one hand with lower error probabilities we find Manhattan and Sorensen, followed by correlation and Euclidean. The lowest error is achieved by Manhattan and Sorensen, both with an error of less than 6.90 meters in 75% of the cases (CDF lower than 0.75), and lastly correlation and Euclidean with an error of less than 7.50 and 7.70 meters respectively. Note in CDF that some of the tested point locations exactly matches with the estimated positions, since zero error is found in about 3-4% of the cases.

4) COMPARISON OF THE RESULTS

As a conclusion, the regression KNN gives better results than the classification KNN and Naive Bayes in this case. The best regression result is obtained by $K = 11$ Manhattan and weights based on distance, with a mean of 4.51 m and a standard deviation of 3.90 m in the three types of algorithms, being Kd-tree the fastest for this choice with 0.66 ms. On the other hand, with classification KNN the best results are given with a low number of neighbors, being the best average error with $K = 1$, of 5.15 m and 4.75 m of standard deviation, Manhattan as metric and distance-based weights in the three types of algorithms, being also Kd-tree the fastest for this choice with 1.95 ms, to this time the delay caused by the network traffic has to be added, which in the performed tests were 6 ms on average from the internal network and 37 ms on average from an external network. The results of classification KNN, regression KNN and Naive Bayes have been obtained using scikit-learn [37] library.

These tests have been used to apply them on the map as a heat map, to obtain the accuracy at each point scanned, so that it is possible to know where the location is less accurate and where it gets better results. In the figures 22 and 23 the maps can be seen coloured as follows: green indicates an error of less than 5 meters, orange between 5 and 15 meters and red an error of more than 15 meters. The majority of places with little accuracy are more distant from the APs, the zones that are surrounded by APs obtain greater accuracy.

B. ENERGY SAVING RESULTS

This paper develops an indoor localization system based on WiFi networks that, in addition to accuracy, has taken into account the energy efficiency of the solution. Since in this work the user's mobile device performs an active task in the trilateration process, reducing the consumption of the solution is an essential part of achieving high user acceptance. In the context of the work several optimization systems have been carried out.

In the successive battery saving methods, it will also be necessary to determine certain user patterns, such as their location (inside or outside the tagged area) or their activity (periods of physical activity such as walking). Since the simulator runs on a server, these two patterns can not be faithfully recreated. Instead, a pseudo-random algorithm that allows generating these periods of activity has been developed, as well as the entry and exit of the tagged place. To consider this simulation we have based on the most repeated behaviour in the place where the experimentation has been carried out. This is, punctual displacements throughout the morning and early afternoon, entry to the place labelled early in the morning, with punctual departures throughout the day, and a final departure from the early afternoon.

In parallel, in order to evaluate the realism of the developed simulation, the original algorithm has been executed in a Samsung Galaxy Note 8 (having a 3.7 V battery), with an Android operating system version 9 (updated on May 1, 2019) and without any other application running in background. During these tests execution, all battery optimization mechanisms were disabled, as well as adjusting the brightness of the screen to automatic. The device was used on a regular basis only to send and receive calls. E-mail synchronization and other applications using push notifications were also deactivated. Once the test has been carried out over 30 days, consumption results between the real device and the simulation were compared. Both environments present different consumption, mainly due to the fact that in the real device there are hundreds of auxiliary processes that can only be controlled by the operative system. However, in Figure 24 can be seen that trends are quite similar. By hence, it can be verified that the simulator satisfies the necessary requirements to carry out the comparative study of the different saving techniques: a consumption proportional to that of a real device, possibility of executing several simulations in parallel and excluding external tasks consumption.

1) ENERGY SAVING TECHNIQUE BASED ON AP LOSSES

In the first optimization, based on the loss of recognized WiFi access points, the number of recharges required throughout

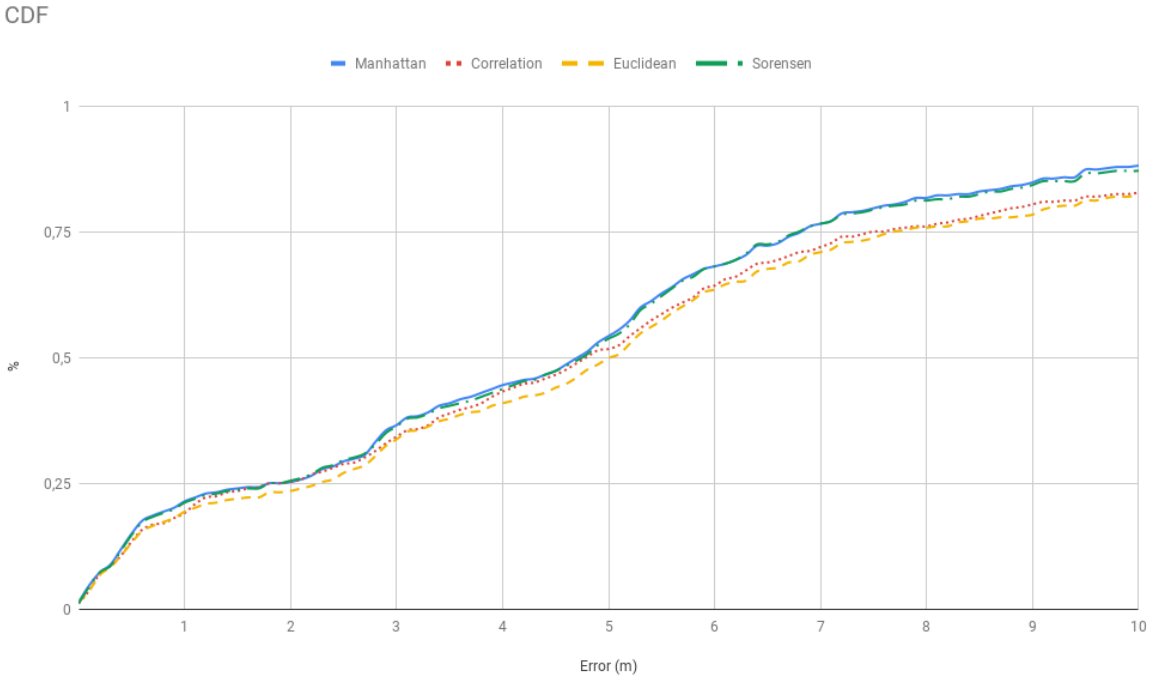

FIGURE 21. Localization results-classification KNN CDF.

FIGURE 22. Localization results-heatmap floor 0 using regression KNN.

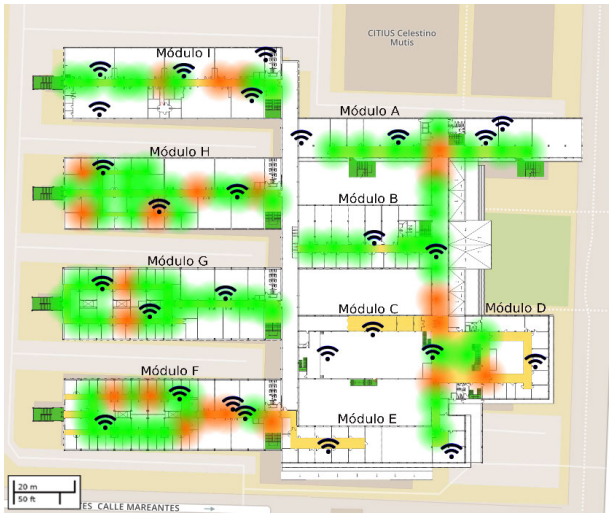

FIGURE 23. Localization results-heatmap floor 1 using regression KNN.

the 30 days of the process in the simulator was 63. The average daily battery consumption using this optimization technique was 1.5 Wh. These recharges occurred every 11 hours approximately, so even without making use of any optimization technique, with the use given to the device, a user could have the mobile for a full day without recharging. However, the continued use of the device as well as the reception of notifications and interactions with the screen will reduce this time drastically.

2) ENERGY SAVING TECHNIQUE BASED ON DISPLACEMENTS

The second optimization method is based on the pause of WiFi network scans during periods of user inactivity. The impact of this method will depend on the level of activity of the user during the day, which will be based largely on the type of work that the user performs. In our case, the simulator has been used to emulate this behaviour in a random way. The displacements are made continuously for 5 minutes with

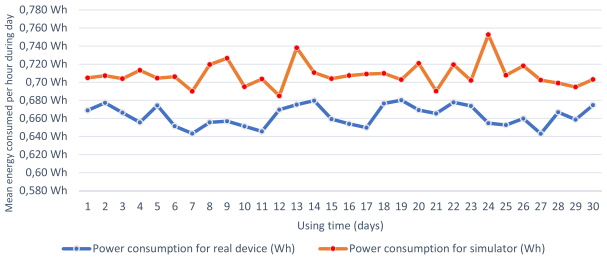

FIGURE 24. Energy consumption comparison between real Samsung Galaxy Note 8 device and the Samsung Galaxy Note 8 based simulator.

a 20% probability of still walking after these 5 minutes. The probability of starting a new displacement is 5%. In this case, the average consumption over the 30 days of evaluation was 1.6 Wh. That is, in this case, the consumption was slightly higher than that obtained using the technique of optimization based on the loss of access points. This is due to the fact that the user, outside the location under location control, generally performs physical activity such as walking. This would reactivate the localization system, even if the user is at home or at any place of recreation.

3) COMBINED ENERGY SAVING TECHNIQUE

Finally, the technique of optimization of consumption based on the combination of the previous two is developed. The parallel execution of the two consumption reduction techniques achieves good results in the conditions under which the executions were tested. This is, in a work environment, with regular trips throughout the day, with a working day of 7.5 hours and with loss of recognized access points in the rest of the day. In this case, as can be seen in figure 25, the reduction in consumption is drastic. During the evaluation month, average daily consumption went from having approximately 1.6 Wh, to 0.8 Wh. This is a 50% reduction with respect to the localization algorithm without any energy optimization.

In addition, the application of this technique does not reduce the accuracy or increase the delay in obtaining the location, so it is concluded that it is the best mechanism to be implemented in the final solution. The context of the problem where the localization algorithm is developed must be kept in mind. The identification of a threat will be the trigger to determine the location of all users of the building and propose a safe escape path. This requires having the location of all users in real time, so the service must run on the devices of those users for the whole time they are in the building. A reduced impact on consumption ensures that the user is not affected by the service and, therefore, does not imply a handicap when implementing the final system. The final solution with the savings techniques presented ensures the viability of this solution, reducing the impact of this service to 8% of the total device energy consumption obtaining during the simulation process. That is, considering that the normal use time of a smartphone with a single recharge is approximately 20 hours, the use of the application presented in this work would reduce it only to 18.5 hours.

TABLE 9. Localization results-load test with multiple users.

	700 users/sec	1000 users/sec	1200 users/sec
Min	3 ms	3 ms	3 ms
50th percentile	5 ms	12 ms	148 ms
75th percentile	8 ms	60 ms	331 ms
95th percentile	66 ms	324 ms	658 ms
99th percentile	920 ms	1438 ms	1643 ms
Max	3002 ms	3584 ms	3771 ms
Mean	34 ms	85 ms	243 ms
Std deviation	203 ms	277 ms	355 ms

C. PERFORMANCE RESULTS

One of the objectives of this system is achieving scalability, therefore, as well as being capable of adding new areas or buildings, it must be able to support multiple requests per second. Consequently, we have performed a load test to check the performance of our server in extreme cases.

The server is installed inside the same building in which the location is made, thus, the tests were made in the same one. The most relevant characteristics of the server for this study are the following:

- RAM: 64 Gb.
- GPU: GeForce GTX 1070 (8 GB) and TITAN Xp (12 GB)
- CPU: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz

The load tests have been performed with the tool Gatling [38], which allows to simulate multiple user requests following configurable patterns, although in our case we have used constant users per second to measure performance in extreme conditions.

In the table 9 can be seen the response time over 700, 1000 and 1200 users per second. Using more than 1200 users some requests started failing, so the limit with this server reaches 1200 users per second. It can be seen that as users increase, the response time increases, this is because the bottleneck is in the CPU, and with 1200 concurrent users reaches 100% of CPU usage. If we had another CPU or balanced the load to other servers could increase the number of users. These results were obtained using a KNN with 11 neighbors, Manhattan as a metric, Kd-Tree as a search algorithm and weights based on distance, if another configuration was used, the results could be improved or worsened, however, with the configuration presented we obtain the best accuracy over our test set.

In a real case, the users are not constant, as the proposed energy saving algorithms prevent many requests from being made, so the results presented are on the most unfavourable case. For this reason, the proposed system supports more than 1200 active users with, in the worst case, an average response time of 243 ms, as mentioned in subsection V-A

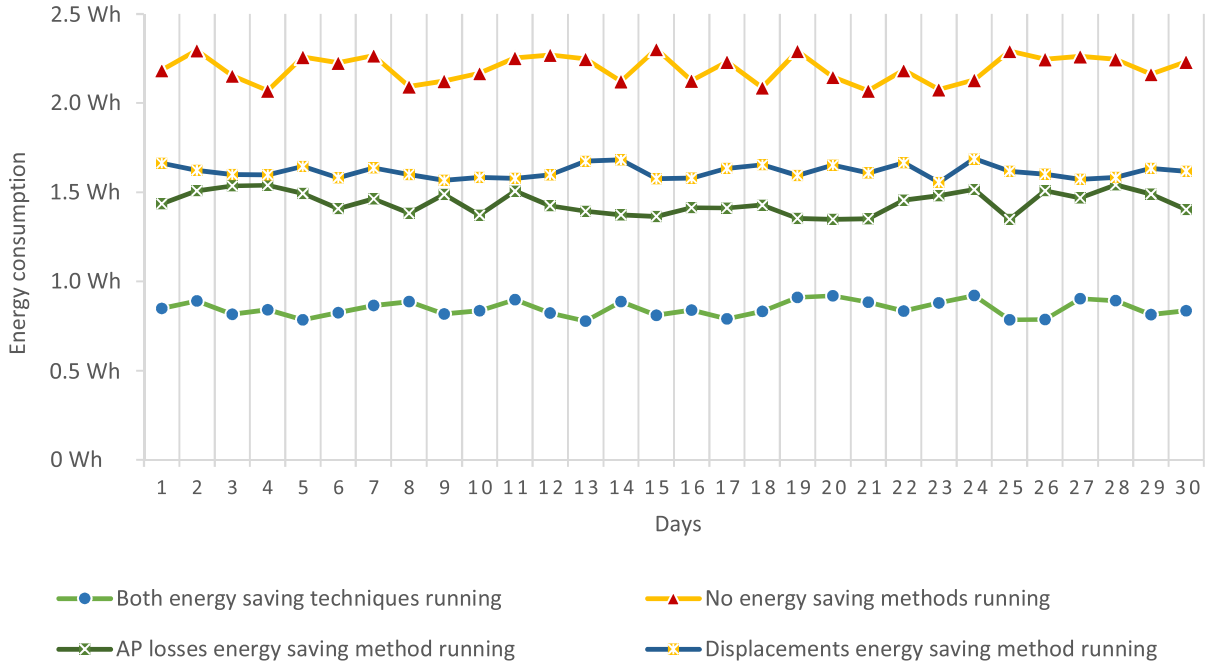

FIGURE 25. Comparison of energy consumption in 24-hour periods for the different battery saving methods developed running on the simulator during 30 days.

to this response time the delay caused by the network traffic has to be added, which in the performed tests were 6 ms on average from the internal network and 37 ms on average from an external network.

VI. CONCLUSIONS & FUTURE WORK

Offices, shopping malls, schools or universities are just some of the facilities in which controlling emergency situations efficiently is critical. To this end, this study has presented a number of innovations: a real case localization system, prepared to provide multiple localization responses simultaneously; an exhaustive comparison of Naive Bayes, classification KNN and regression KNN, using different metrics, search algorithms, number of neighbours and weights; a combination of energy-efficient techniques applied to an indoor localization service based on WiFi fingerprint using Android devices able to run all day long; the publication and description of the dataset compiled in this study in order to replicate and compare the experiments performed.

The resulting location system in this study is implemented in a real environment, with a response capacity of 1200 simultaneous users in 243 ms and an average error of 4.51 meters, with a robust client/server architecture that carries the entire location load to the server, allowing to completely modify the localization algorithm or extending the dataset in real time without affecting the user's device, since the client only performs and sends WiFi scans.

Finally, thanks to the energy saving policies implemented on the original indoor location system, this work reduces the energy impact to only 8% (0.73 Wh without the service vs 0.8 Wh with the service running). In this way, the user will be minimally affected by the use of the location application, thus reducing the possibility of it being uninstalled due to the high battery consumption. This is a critical risk that, thanks to work in terms of energy saving, has been successfully mitigated. Therefore, the additional consumption that our system would produce in a device with a common 3200 mAh battery would be 256 mAh per recharge cycle, which would give a battery time enough for a daily use.

From the experimentation carried out, we have extracted the applicability of each method from a database of WiFi fingerprints collected in a real environment. Furthermore, the saving energy analysis has also been tested under real and simulated situations. Simulated situations was used in order to check the robustness of the system under a huge number of requests.

Since the positioning technique presented in this paper performs network detection on the user's own mobile device, the broadcasting of fake SSIDs from a fraudulent device present security problems. The exposed methodology is not individually based on a single point, but the whole scanned WiFi fingerprints are processed. However, different SSIDs could be falsely broadcasted with a much different intensity from the original RSSI, so that this will present problems for the system. Even so, this is considered as a future work taking into account the security context in which this system is applied.

Finally, it should be taken into account that this work addresses the problem of indoor positioning using an infrastructure where the data from the scanned WiFi networks on the device are sent to the server, where they are processed. This is done in order to monitor the position of each user

in emergency case. However, there are approaches to the same problem where the data is processed entirely in the device, which increases user privacy but also increases system response time and the risk of no valid positions for a user in emergency cases.

Future work will include an analysis of other localization techniques and experimentation of applicability in serverless systems, thus verifying the advantages and disadvantages of each.

REFERENCES

[1] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen, "A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned," in *Proc. 14th Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, New York, NY, USA, 2015, pp. 178–189. [Online]. Available: http://doi.acm.org/10.1145/2737095.2737726

[2] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1281–1293, 3rd Quart., 2013.

[3] J. Xiao, Z. Zhou, Y. Yi, and L. M. Ni, "A survey on wireless indoor localization from the device perspective," *ACM Comput. Surv.*, vol. 49, no. 2, 2016, Art. no. 25.

[4] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 13–32, Mar. 2009.

[5] D. Lymberopoulos and J. Liu, "The microsoft indoor localization competition: Experiences and lessons learned," *IEEE Signal Process. Mag.*, vol. 34, no. 5, pp. 125–140, Sep. 2017.

[6] S. He and S.-H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 466–490, Jan. 2015.

[7] J. Niu, B. Wang, L. Cheng, and J. J. P. C. Rodrigues, "WicLoc: An indoor localization system based on WiFi fingerprints and crowdsourcing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3008–3013.

[8] A. K. M. M. Hossain, H. N. Van, Y. Jin, and W.-S. Soh, "Indoor localization using multiple wireless technologies," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, Oct. 2007, pp. 1–8.

[9] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, New York, NY, USA, 2010, pp. 173–184. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860016

[10] H. Lim, L. Kung, J.-C. Hou, and H. Luo, "Zero-configuration indoor localization over IEEE 802.11 wireless infrastructure," *Wireless Netw.*, vol. 16, no. 2, pp. 405–420, Feb. 2010, doi: 10.1007/s11276-008-0140-3.

[11] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, "ARIADNE: A dynamic indoor signal map construction and localization system," in *Proc. MobiSys*, 2006, pp. 151–164.

[12] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "WiDeep: WiFi-based accurate and robust indoor localization system using deep learning," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Mar. 2019, pp. 1–10.

[13] Z. Chen, H. Zou, J. Yang, H. Jiang, and L. Xie, "WiFi fingerprinting indoor localization using local feature-based deep LSTM," *IEEE Syst. J.*, to be published.

[14] J. Wang, Y. Wang, D. Zhang, and S. Helal, "Energy saving techniques in mobile crowd sensing: Current state and future opportunities," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 164–169, May 2018.

[15] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, Jun. 2010, pp. 315–330. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1814433.1814464

[16] M. B. Kjaergaard, "Location-based services on mobile phones: Minimizing power consumption," *IEEE Pervasive Comput.*, vol. 11, no. 1, pp. 67–73, Jan./Mar. 2012.

[17] N. Brouwers, M. Zuniga, and K. Langendoen, "Incremental Wi-Fi scanning for energy-efficient localization," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2014, pp. 156–162.

[18] Y. Gao, J. Niu, R. Zhou, and G. Xing, "ZiFind: Exploiting cross-technology interference signatures for energy-efficient indoor localization," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2940–2948.

[19] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "EnLoc: Energy-efficient localization for mobile phones," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 2716–2720.

[20] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2010, pp. 285–298. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814462

[21] V. Q. Viet, H. M. Thang, and D. Choi, "Adaptive energy-saving strategy for activity recognition on mobile phone," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2012, pp. 95–100.

[22] T. King and M. B. Kjærgaard, "Composcan: Adaptive scanning for efficient concurrent communications and positioning with 802.11," in *Proc. 6th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2008, pp. 67–80.

[23] K.-H. Kim, A. W. Min, D. Gupta, P. Mohapatra, and J. P. Singh, "Improving energy efficiency of Wi-Fi sensing on smartphones," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2930–2938.

[24] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2009, pp. 85–98. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644048

[25] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback CrowdSensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2013, pp. 7:1–7:14. [Online]. Available: http://doi.acm.org/10.1145/2517351.2517372

[26] A. A. Moamen and N. Jamali, "Share sens: An approach to optimizing energy consumption of continuous mobile sensing workloads," in *Proc. IEEE Int. Conf. Mobile Services*, Jun./Jul. 2015, pp. 89–96.

[27] *Flask—Flask Documentation (1.1.x)*. Accessed: Sep. 1, 2019. [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/

[28] *MongoDB Documentation*. Accessed: Sep. 1, 2019. [Online]. Available: https://docs.mongodb.com/

[29] *Socket.IO—Docs | Socket.IO*. Accessed: Sep. 1, 2019. [Online]. Available: https://socket.io/docs/

[30] *Firebase Cloud Messaging | Firebase*. Accessed: Sep. 1, 2019. [Online]. Available: https://firebase.google.com/docs/cloud-messaging

[31] M. Martins, J. Cappos, and R. Fonseca, "Selectively taming background Android apps to improve battery lifetime," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2015, pp. 563–575.

[32] S. Gansemer, S. Pueschel, R. Frackowiak, S. Hakobyan, and U. Grossmann, "Improved RSSI-based Euclidean distance positioning algorithm for large and dynamic WLAN environments," *Int. J. Comput.*, vol. 9, no. 1, pp. 37–44, 2010. [Online]. Available: http://www.computingonline.net/computing/article/view/696

[33] J. Torres-Sospedra, R. Montoliu, S. Trilles, O. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9263–9278, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417415005527

[34] J. Yim, "Introducing a decision tree-based indoor positioning technique," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1296–1302, Feb. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417406004179

[35] E. Mok and G. Retscher, "Location determination using WiFi fingerprinting versus WiFi trilateration," *J. Location Based Services*, vol. 1, no. 2, pp. 145–159, Feb. 2008, doi: 10.1080/17489720701781905.

[36] J. L. S. González, L. M. S. Morillo, J. A. Álvarez-García, F. E. De Salamanca Ros, and A. R. J. Ruiz, "Energy-efficient indoor localization WiFi-fingerprint dataset," 2019, doi: 10.21227/49yg-5d21.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[38] *Gatling Open-Source Load Testing—For DevOps and CI/CD*. Accessed: Sep. 1, 2019. [Online]. Available: https://gatling.io/

JOSE L. SALAZAR GONZÁLEZ received the degree in computer engineering from the University of Seville, Spain, in September 2018.

He has been a Researcher with the University of Seville, since October 2018. He was with a company developing chat bots that used natural language processing to solve customer queries, from December 2017 to September 2018. He is currently involved in research projects on artificial intelligence in collaboration with the University of Seville.

LUIS MIGUEL SORIA MORILLO received the degree in computer engineering and the Ph.D. degree in computer engineering from the University of Seville, Spain, in 2009 and 2011, respectively. He is an Assistant Professor with the Department of Languages and Computer Systems, University of Seville. His research interests include activity recognition and ubiquitous computing.

JUAN A. ÁLVAREZ-GARCÍA received the degree in computer engineering from the University of Seville, Spain. He is an Associate Professor with the Department of Languages and Computer Systems, University of Seville. His research interests include human behavior analysis and activity recognition using wearable/mobile devices or computer vision from CCTV cameras.

FERNANDO ENRÍQUEZ DE SALAMANCA ROS received the degree in computer engineering from the University of Seville, Spain.

He is an Associate Professor with the University of Seville, since February 2012. He started his career, in 2003, collaborating in research projects on natural language processing. In 2011, he presented the Ph.D. degree dissertation on the combination of classifiers for the improvement of different tasks, such as morphological analysis or named entity recognition. Since then he has participated in many research initiatives related to intelligent information processing and machine learning techniques.

ANTONIO R. JIMÉNEZ RUIZ was born in Santander, Spain, in 1968. He received the degree in physics and computer science and the Ph.D. degree in physics from the Universidad Complutense de Madrid, Madrid, Spain, in 1991 and 1998, respectively. Since 1993, he has been with the Center for Automation and Robotics, Spanish Council for Scientific Research, Madrid, where he holds a research position. He has authored more than 100 articles in journals and conference proceedings. His current research interests include local positioning solutions for indoor/GPS-denied localization and navigation of persons and robots, signal processing, Bayesian estimation, and inertial-ultrasonic-RFID sensor fusion. He is a Reviewer for many international journals and projects in his research field.