Occupancy Detection via iBeacon on Android Devices for Smart Building Management

Omitted for blind review

Abstract— **Building heating, ventilation, and air conditioning (HVAC) systems are considered to be the main target for energy reduction due to their significant contribution to commercial buildings' energy consumption. Knowing a building's occupancy plays a crucial role in implementing demand-response HVAC.**
In this paper we propose a new solution based on the iBeacon technology. This solution is different from the previous ones because it leverages on the Bluetooth Low Energy standard, which provides lower power consumption. Moreover, the iBeacon protocol can be used both on iOs systems and Android ones, making this new approach portable. Differently from our previous work based on iOS devices, in this paper we focus on an Android based solution with the aim of increasing the accuracy of the location and the energy efficiency of the entire system. We increased the accuracy by 10% and the energy efficiency by 15%.

Keywords—smart buildings, indoor location, iBeacon, energy efficiency

I. INTRODUCTION

Smart buildings, places where sensors and actuators make the location more intelligent, are becoming more and more relevant and they are the natural evolution of today's constructions. The goal of this new trend is to use the information collected to adapt some building parameters to achieve better energy efficiency, without reducing the user comfort perception. Heating, ventilation and air conditioning (HVAC) systems represent almost half of the energy consumption of a commercial building and lighting is another important component. In the US it is estimated that buildings are the major consumers of energy, accounting for a 40% of the total energy consumption [16]. Within this context, it is possible to increase the energy efficiency and the level of comfort of a building by exploiting information on who is in a specific room in order to tailor the behavior of the building could be easily tailored to the position of the occupants inside the building. In this way, it is possible to avoid energy wastes using the HVAC system only when needed. Another possible use case that benefits of the occupancy information is the efficient management of the lighting system; within a smart building that is aware of the user position, it is possible to turn on and off the lights according to the actual needs, increasing the building efficiency.

Many different works have tackled the problem of deriving the building occupancy status: as it will be presented in Section 2, generally the different approaches use different types of sensors and algorithms. However, despite numerous research works have been conducted to find a cheap, simple, power-efficient and reliable solution to this issue, the problem is still open and an optimal solution, satisfying all three constraints, has still to be found.
With this work, we propose the Apple iBeacon technology [1] as a possible solution to detect the number of users in a room, and how it can be used to gather information about their movements (thus identifying and tracking them) inside the building even if it has not been developed to solve the occupancy detection problem but to enable the design of indoor proximity systems. In a previous work [17], we made a similar study using Apple mobile devices (iPhone and iPad). Differently, here we want to port and improve the same methodology on Android devices, since they represent a huge part of the smartphones market. As it will shown in the next section, the porting of such technology on Android devices is challenging due to some restrictions of the underlying operating system. Moreover, with respect to the previous work, we increased the accuracy of the classification algorithm we use for the occupancy information from 80% to 90%.

This paper is structured as follows: Section II introduces the state of the art showing the different techniques defined for the occupancy detection. In Section III we provide an overview of the iBeacon technology while Section IV introduces the proposed architectural solution for Android-based devices. Section V, VI and VII focus on the critical aspects of the proposed system, and solutions are proposed and validated. Section VIII shows the limitations of the current approach and proposes possible solutions to be investigated as future work..

II. STATE OF THE ART

Previous studies [10] have developed different algorithms in order to estimate the position of a user through a set of information acquired using different types and combinations of ambient sensors. The state of the art methods exploited many different technologies to estimate occupancy, like, for instance, infrared, RIFD, ultrasound-pulses, GSM and Wi-Fi. However, there is still a lack of low cost but accurate solutions. In one of the first works addressing the occupancy detection problem [18] a solution based on ***infrared sensors*** has been proposed: with this solution users must wear an active badge that broadcasts a unique identifier, while, on the building side, a quite huge number of infrared sensors must be placed all around the target building making the system expensive. Despite the high installation cost, the accuracy of this solution in crowded rooms is low due to the high number of collisions.
Vice versa, FastSlam [19] and Landmarc [20] advocate the use of RFID, requiring the placement of antennas in the space to be

monitored. Furthermore, a tag has to be assigned to each building occupant, which must be carried so that they can be correctly identified when they are close to an antenna. In this case, the main drawback is due to the high number of antennas since the coverage area of the RFID signal is quite limited (6 meters), making the installation cost of this solution quite high. Other works propose the use of ***ultrasound pulses*** [21] or rely on the GSM network [22] to retrieve the occupancy data of a building. Unfortunately, both these approaches are quite inaccurate. On the contrary, very good results in terms of accuracy have been obtained by the use ***Wi-Fi networks*** [23,24,25,26,27], reaching an accuracy of 99.84% without requiring any training phase (unsupervised learning). The big drawback of these methods is the power consumption of the mobile devices (smartphones) used as collectors of the Wi-Fi signals. Also ***Bluetooth*** has been proposed as a possible solution [28,29] since it allows low power communications with respect to Wi-Fi in case of low data rate (usually small bursts) [30]. Unfortunately, the reached accuracy with the standard Bluetooth protocol is worse than the accuracy obtained by exploiting Wi-Fi networks.

Differently from the cited works, as we will show in the next sections, with this paper we want to exploit the use of the iBeacon technology on Android devices in order to evaluate its effectiveness for the occupancy detection problem.

III. iBeacon Technology

iBeacon technology [1] is based over the Bluetooth Low Energy (BLE) [31], a new standard introduced by the 4th version of the protocol, designed to provide significantly lower power consumption with the same efficiency as previous versions. More in detail, iBeacon is a particular implementation of the GATT [2] protocol, which allows both the advertisement of a particular service and the connection between two devices that can exchange data. Differently from the complete GATT implementation, iBeacon only implements the first feature. Even if iBeacon is a partial implementation of GATT, it fits perfectly with our needs since our main goal is to know if a person (associated with a device) is inside a particular room; in fact, within this context, we need to uniquely identify the signal that a room is transmitting (i.e., the advertising signal).

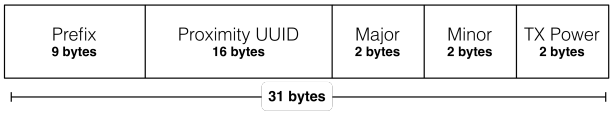

Figure 1. iBeacon Packet Structure

The iBeacon protocol has two main components: a transmitter and a receiver. The transmitter broadcasts packets that uniquely identify it creating an iBeacon region, a set of beacons identified by the same proximity *Universally Unique Identifier,* UUID. The receiver periodically scans signals in the air in order to detect particular iBeacon packets. Inside an iBeacon packet (Figure 1) we can identify 5 different fields: the iBeacon prefix (9 bytes) is a constant field to identify the iBeacon protocol, the proximity UUID (16 bytes) that identifies beacons belonging to a certain organization, major value (2 bytes) that characterizes a group of related beacons, minor value (2 bytes) that is used to distinguish beacons with the same UUID and the TX power (2 bytes) that indicates the signal's strength measured at 1 meter from the device.

The iBeacon protocol allows the implementation of two main functionalities: region monitoring and ranging. The ***monitoring*** notifies a listener application every time we enter/exit a specific iBeacon region. We can define more than one region to be monitored and this functionality can work in background. The ***ranging*** provides an approximation of the distance from the iBeacon transmitter using the information of the TX Power field. As the strength of the signal decreases predictably as we get further, knowing the RSSI (received signal strength indicator) at 1 meter, and the current RSSI, it is possible to calculate the difference. The iBeacon protocol has been developed with the aim of detecting the proximity to a particular object. An example of its use described by Apple is the possibility of creating a smart museum: as soon as you approach to a painting, the smartphone will show you the most interesting information and some interactive experience related to it. For our purpose, we try to exploit the proximity information provided by iBeacon in order to infer the occupancy of a particular room.

IV. The Proposed Occupancy Detection System

The philosophy behind this system is quite straightforward: we envision the users with their smartphones (or smart things in general) within a smart building that is instrumented with low cost Bluetooth 4.0 antennas (Wi-Fi access points can easily integrate this feature; moreover, each computer inside the building can be used as an antenna). When a user enter a room that is iBeacon enabled (i.e., it has an antenna), the room advertise itself to the user; consequently, the user smart-device detects the advertisement and sends this information to the Building Management System (BMS) [32].

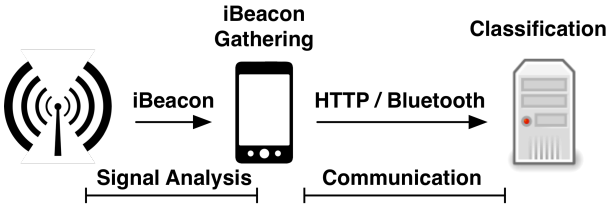

Figure 2. Main aspects of the proposed solution

To implement such functionalities, we have created an architecture composed of three main components: the (1) ***beacon transmitters***, devices within the rooms sending uniquely identified iBeacon packets to a (2) ***client mobile application*** installed on the occupants smartphones; this app is able to detect beacons produced by the building and sends this information to a (3) ***building remote server*** (the BMS) through an HTTP request or a Bluetooth connection. On this server, some classification algorithms are in charge of extrapolating the occupancy data from the detected packet information. Figure 2 shows the aforementioned architecture that highlights two aspects: the need for an accurate signal analysis and the need for an energy efficient communication between the devices and the remote server. For both aspects we evaluate different techniques that will be shown in the next sections.

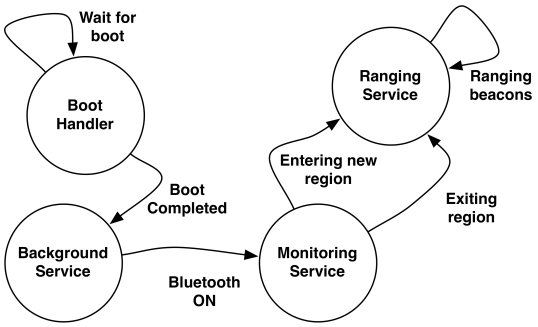

Figure 3. Application behavior

A. Beacon Transmitter Implementation

The transmitter is implemented on a *Raspberry PI board (version b)* [4] that runs the *Raspbian Operating System*. In order to provide the BLE functionalities to the board, we used a Bluetooth 4.0 USB dongle (Inateck BTA-CSR4B5) [5]. The software stack to generate iBeacon packets is provided the *bluez* [6] kernel module and its related tools. In order to make the transmitter work properly it is necessary to calibrate the TX power field. This can be done by putting the device one meter away from the transmitter and through Radius Networks iBeacon Locate [7] app, changing the TX power field until the detected distance by the device is about one meter.

B. Server Implementation

The server has to collect all information sent by the user smart and to insert them in a database the association between the device and the room where it is located. These information are then used by a classification algorithm (Section VI) in order to get the occupancy information. We realized our prototype server with another Raspberry Pi; since the server has to be able to receive HTTP requests, we implemented a RESTFul interface using Flask micro-framework [3]. In order to handle a large number of concurrent requests we have chosen the Standalone WSGI Container Tornado, which, thanks to his non-blocking approach, suits very well our needs.

C. Smartphones app

The smartphone app retrieves the iBeacon packets from the antennas and sends them to the building server. As said in the introduction, differently from our previous work [17] where we used iOS based devices, in this work we faced the challenges of implementing the system using Android. Testing the possibility to implement such occupancy detection technique on Android devices is important since they represent the 85% of the today market [33]. Unfortunately, since there has been little support from Google to the iBeacon technology, we had to face some implementation problems. In fact, Android does not provide any software stack for iBeacon. For this reason, we used an open source library by Radius Networks [8]. As said, the application has to detect the presence of beacons in the air continuously; to implement such functionality, we created a background service: Figure 3 shows the behavior of the application.

The ***Boot Handler*** listens to the boot complete event raised by the Android OS at the end of the boot phase and launches the ***Background Service***. This service will take care of turning on the Bluetooth and creating the ***Monitoring Service***. This last service implements the *iBeacon Monitoring Feature* (Section III) and detects if the device is entered in a new *iBeacon Region*. Accordingly to the iBeacon protocol, the app has to be aware about the region code that has to be monitored: as a consequence, the app and the transmitter has to be configured on the same *Region UUID*. After this configuration phase (to be done once at the system setup), the app is notified whenever a new iBeacon packet is detected. However, the monitoring service does not provide the information of the received beacon packet (UUID, major, minor, TX power); to obtain such information, it is necessary to execute the ***Ranging Service*** as soon as the device entered in a region. This service identifies the beacon received and provides the approximate distance from the beacon transmitter. This information is processed and sent to the server.

V. Signal Analysis

Similar to other high frequency signals that are transmitted through air, Bluetooth is affected by humidity, presence of other signals and many other environmental factors [9]. Therefore, different tests have been performed to evaluate the fluctuation of the signal received. Tests consisted in positioning the device at a given distance D from the transmitter, after a suitable calibration, and registering the detected signals. Figure 5 shows the recorded values detected with $D = 2\ mt$ with a Samsung S3 mini. It can be observed that there is a large variability of the estimated distance between the transmitter and the Android based receiver.

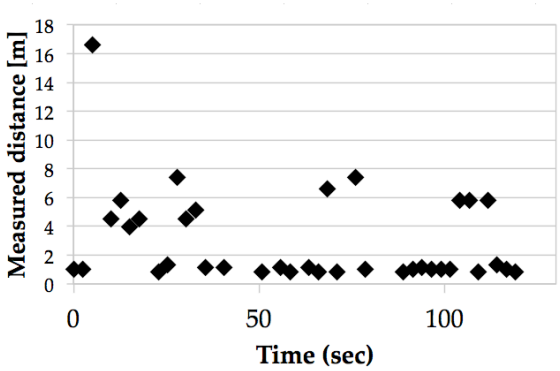

Figure 4. Signals collected with 2 seconds scan period

This lack of accuracy is caused also by a limit of the Android operating system since its BLE APIs allows only a single signal strength measurement per scan[1], differently from iOS where it is possible to get many measurements for each broadcast advertisement by the transmitter. To understand this concept, let us consider an example: having a scan period of two seconds and an iBeacon generator that transmits thirty times per second, an Android device that scans for ten seconds gets only five samples (despite the rate of the transmitter the device will get one sample per scan and so ten divided by two samples). On the contrary, an iOS device receives three hundred samples because inside each scan it can collect more

1. The scan period is the time used to collect samples for estimating the distance

than one sample. As a consequence, the iOS distance measurements result to be more accurate, since it is allowed to work on a higher number of recorded data. Another important problem we faced during the implementation of our prototype system is that the adapter sometimes looses some samples due to bugs in the software stack. In order to cope with the aforementioned problems, we increased the scan period to collect more sample obtaining more accurate distance estimations (Figure 6).

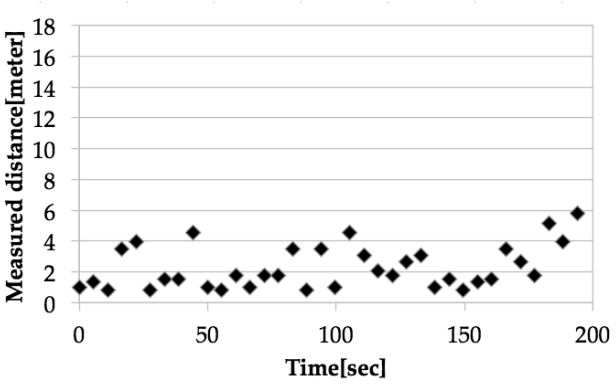

Figure 6. Signal Evaluation with 5 seconds scan period

Unfortunately, increasing the scan period, the estimation phase takes a longer time, causing the application to be less reactive to distance changes by the user. In order to obtain a low latency but good distance estimation at the same time, we implemented a custom distance estimation algorithm. With the proposed algorithm[2], we can solve the problem of beacons' losses since we remove the beacon information only after the second consecutive loss, otherwise its value is maintained. Moreover, we solve also the problem of the fluctuation of the signal since we consider that if at a certain time T the device is in a position P, at time T+1 the position will be P+ΔP, and ΔP depends on the speed and on the time interval. With these assumptions, the older position has a role in determining which will be the current one that can be estimated as follows:

$$p_i = p_{i-1} * coefficient + (1 - coefficient) * v_i$$

where p_i is the result of the computation of the value related to a single beacon, p_{i-1} the value of the signal history and v_i the new measurement. So the older position will influence the current one with a given probability, the next one with a lower probability and so on. Increasing the coefficient makes the signal more stable and less affected by peaks but on the other hand it becomes less responsive to movements. To determine the best trade-off for this coefficient some dynamic tests have been performed by moving the device from one transmitter to another at a speed of 1 - 1.5 m/s and registering the responsiveness to the fluctuation of the signal. After some parameters tuning we found that 0.65 is a good trade off between stability and responsiveness as shown in Figure 7 and Figure 8.

VI. Algorithms for Indoor Occupancy

Given the information provided by the transmitter after the signal analysis, it is necessary to determine the position of the user. However, it is not easy to model the radio propagation in indoor places because of the different factors that can affect the signal [9]. Previous studies [10] have developed different algorithms to estimate the position of a user through a set of information; they can be classified in 3 main categories: *Triangulation*, *Proximity* and *Scene Analysis*. Triangulation has been discarded because it requires very stable and accurate input data [10] and due to the signal fluctuation we decided to not use this technique. In our previous work [17] we used the *Proximity Technique*; this technique uses the strongest signal received from a grid of transmitters, each of which associated with a particular location, in order to determine the position of the user. The results of our first work were encouraging (we reached an accuracy of the 84%) but in this paper we try to increase also the classification accuracy.

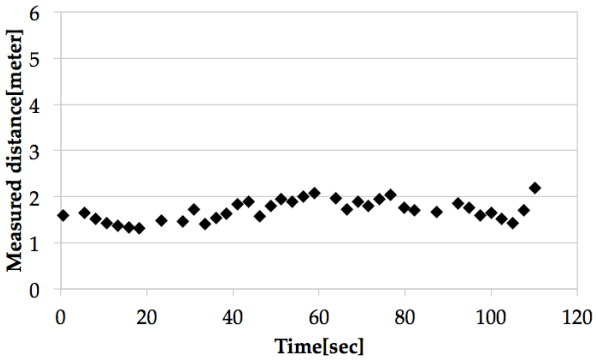

Figure 5. Signal static evaluation (Coeff = 0.65)

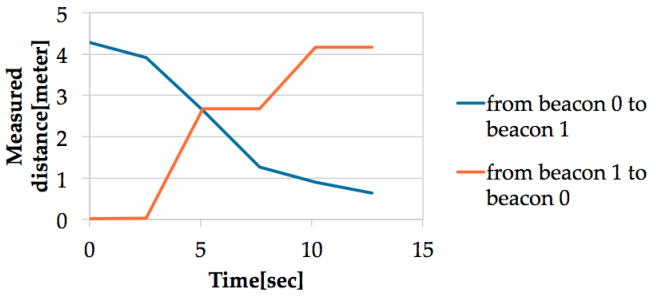

Figure 8. Signal dynamic evaluation (Coeff = 0.65)

For this reason, we propose the adoption of the ***Scene Analysis*** technique [10]: it is a pattern recognition method that uses the characteristics of the location to make a classification. More in detail, the approach compares the observed characteristics to pre-stored characteristics for each pattern to determine a match. In our implementation, the relevant feature considered is the detected distance from the different iBeacon transmitters inside the room. First, a data collection phase is needed, requiring an operator that walks around the building collecting samples (beacon identifiers and their detected distances). These samples are then associated with the specific room and sent to the server that stores them in the database.

2. The code of the presented system is open source and can be downloaded from http://xxxxxxx-omitted-for-blind-review

After this phase the server creates a supervised machine-learning model based on all the samples. When a user enters the building the application will send to the server the list of all the beacons detected at a certain instant and their respective distances. The server using the pre-computed model can estimate the user's location. Our implementation used Support Vector Machines (SVM) [13] with the Radial Basis Function kernel, as suggested by [12]. To test the accuracy of this solution we have created a testing application and we asked a user to move within a house and to indicate its actual location.

Number of samples	561
Correct classified samples	529
Incorrect classified samlpes	32
Accuracy score	0.9428
Min absolute errore score	0.6576
Min squared error	0.6576

(a)

True Positive Rate	84.1%
False Positive Rate	20%
Precision	94%
Recall	94%
Fi-score	94%
Support	561

(b)

Outside	Inside
227	19
13	292

(c)

Figure 9. Experimental results

Part of the collected data was then used to build the aforementioned SVM model (*training set*), while another part was used to test its behaviors (*testing set*). As result we have obtained an accuracy of about the 94% (Figure 9), increasing the accuracy of about 10% from previous work. From the confusion matrix (Figure 9.c) the number of false positive, detection of the user inside the room while he was outside is slightly higher than the number of false negative, detection of the user outside the room while he was inside, is about the same. This result is good since it is better to have false positive than a false negative because false negatives are a problem in terms of user comfort and safety.

VII. Mobile Device Energy Consumption and Communication Infrastructure

In Section IV, we anticipated that we envisioned two different ways to send the beacons received by the smartphones to the building server: a first one based on Wi-Fi and a second one based on Bluetooth. In our previous work with iOS devices, we discovered that an architecture based on the Wi-Fi protocol is very expensive from the energy consumption point of view [17]. As known, having energy efficient applications is crucial on mobile devices since the battery is a very limited resource [34]. For this reason, we focused our attention also on the measurement of the energy consumption caused by our app on the Android device. In this work, we performed the measurements with the Wi-Fi communication channel (the same used on iOS) and also with an alternative channel based on Bluetooth. In this last case, a Bluetooth connection is established between the smart device and the beacon transmitter when a beacon is received. To develop this second solution we have created a Bluetooth server in the iBeacon transmitter (that is thought to be not-battery based) that retransmits the information received to the central server using HTTP requests. Implementing this new Bluetooth based solution, we discovered that both implementations have pros and cons: the Wi-Fi is more reliable and stable but forces to keep on the wireless adapter that has a high power consumption. On the other hand, the Bluetooth one is more energy, but it's less stable than the Wi-Fi solution due to bugs in the BLE Android API.

In order to understand the energy consumption of our system, we measured the energy consumption of our app using the ***VeryNiceBlindApp*** application [34] we have developed. This application is basically is background service that logs the battery status is a very energy efficient way in order influence the least possible the battery behavior and it is able to model the energy profile of a device. Figure 10 shows the average of 10 measurements performed on a Samsung Galaxy S3 Mini with Android 4.1.

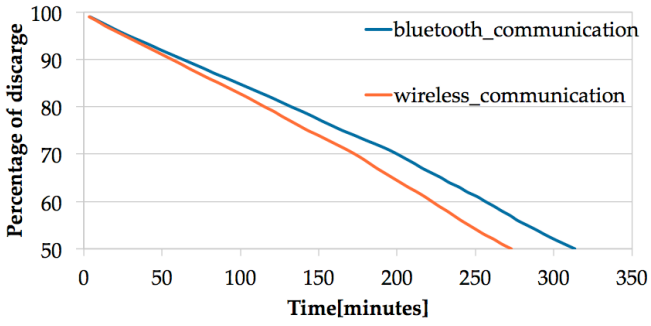

Figure 10. Consumption with http communication

As expected, the Wi-Fi solution is more expensive in terms of energy consumption compared to the second one. Using the Bluetooth based architecture we obtained an energy saving of the 15%. As a drawback, in order to support the Bluetooth architecture, a more complicated antenna board is required. As last consideration, with our app installed, the battery lifetime of the mobile device is around 10 hours.

VIII. Limitations

With this work we further investigated the possibility of using iBeacon as underlying technology for the occupancy detection in a building. Some issues are still opened and are left for future investigations. As described in Section V, we underline that the strength of the signal received from an iBeacon antenna, considering the same transmitter and the same distance, changes significantly between different devices. Figure 11 shows an example of two smartphones, a Nexus 5 and S3 mini, positioned at the same distance.

A possible solution to this problem might be to collect experimental information on the power strength received by different devices and using them to tune the information that is provided to the server during the setup phase. Another problem that is still open is the overall energy efficiency of the smartphone app. By switching to the Bluetooth solution we increase the energy efficiency by 15% but the total duration of the battery is still limited to 10 hours. A possible solution to this problem, since a relevant part of the power consumption comes from the communication modules, is to use the accelerometer to detect if the user is moving to enable the iBeacon sensing and transmitting (if the user has not changed position, it means that there is no useful information about the occupancy).

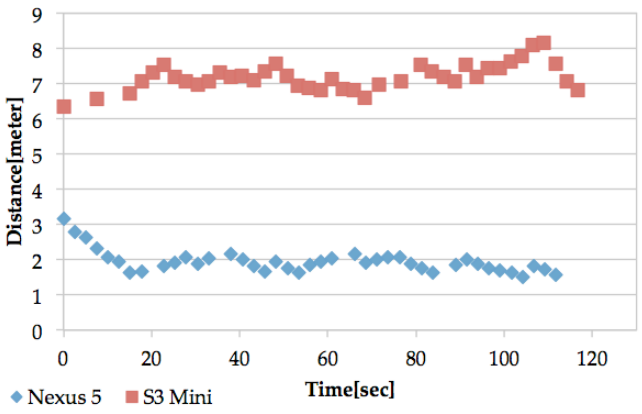

Figure 11. Differences in the received signal strengths

IX. Concluding Remarks and Future Developments

With this work we aimed at evaluating the possibility of using iBeacon on Android devices as a suitable technology for the occupancy detection in a smart building. The paper has shown the major challenges in using such technology on the proposed architecture: a big effort has been put in the signal stabilization, on the classification algorithms and on the energy efficiency on the mobile device used to sense the environment. On the classification algorithms side, we have increased the accuracy from 84% to 94%. On the application energy efficiency, proposing an alternative communication pattern via Bluetooth, we obtained a 15% improvement. We believe this is a good starting point for further developments on the different components of the proposed solution. In particular, signal accuracy is variable, but this would require a modification to the Android kernel to provide more samples and achieve the same level of accuracy of the iOS devices. Google announced the release of Android L OS by the end of September, that promises to correct some of the bugs related to Bluetooth present in Android 4.4 and permits to generate beacon packets from the device [14]. With this new support more solutions become possible, with an improvement of the information provided by the devices. The source code of the described system can be found online at: http://xxxxxxx-omitted-for-blind-review.

References

[1] Apple: iBeacon for developers: https://developer.apple.com/ibeacon/

[2] Bluetooth GATT Specification: https://developer.bluetooth.org/gatt/Pages/default.aspx

[3] Ronacher, A.: Flask web development, one drop at a time

[4] Raspberry PI - www.raspberrypi.org

[5] Inateck: Bta csr4b5 bluetooth usb 4.0 adapter - http://www.inateck.com/inateck-bta-csr4b5-usb-bluetooth-4-0-adapter/

[6] Bluez, official linux bluetooth protocol stack - www.bluez.org

[7] Radius Networks: iBeacon locate application - http://developer.radiusnetworks.com/2013/11/13/ibeacon-monitoring-in-the-background-and-foreground.html

[8] Radius Networks: Welcome developers - http://developer.radiusnetworks.com

[9] Enterprises, L.: Make your wi-fi hi-fi: The "truth" about wireless signal interference

[10] Liu, H.L.H.D.P.B.J.: Survey of wireless indoor positioning techniques and systems. Technical report, IEEE (2007)

[11] Scikit-learn developers: scikit learn

[12] Bolliger, P.: Redpin adaptive, zero configuration in- door localization through user collaboration. Tech- nical report, Institute for Pervasive Computing ETH Zurich, Switzerland (2008)

[13] Laura Auria, R.A.M.: Support vector machines (svm) as a technique for solvency analysis. Technical report, Deutsche Bundesbank, Hannover; German Institute for Economic Research, Berlin. (2007)

[14] Networks, R.: Making an iBeacon with android l

[15] Foundation, T.A.S.: ab - apache http server bench- marking tool

[16] Buildings energy data book. Technical report, US Department of En- ergy, August 2012.

[17] OMITTED FOR BLIND REVIEW

[18] WANT, Roy, et al. The active badge location system. ACM Transactions on Information Systems (TOIS), 1992, 10.1: 91-102.

[19] HAHNEL, Dirk, et al. Mapping and localization with RFID technology. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. IEEE, 2004. p. 1015-1020.

[20] NI, Lionel M., et al. LANDMARC: indoor location sensing using active RFID. Wireless networks, 2004, 10.6: 701-710.

[21] WARD, Andy; JONES, Alan; HOPPER, Andy. A new location technique for the active office. Personal Communications, IEEE, 1997, 4.5: 42-47.

[22] OTSASON, Veljo, et al. Accurate GSM indoor localization. In: UbiComp 2005: Ubiquitous Computing. Springer Berlin Heidelberg, 2005. p. 141-158.

[23] JIANG, Yifei, et al. Ariel: Automatic wi-fi based room fingerprinting for indoor localization. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. ACM, 2012. p. 441-450.

[24] BOLLIGER, Philipp. Redpin-adaptive, zero-configuration indoor localization through user collaboration. In: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments. ACM, 2008. p. 55-60.

[25] MELFI, Ryan, et al. Measuring building occupancy using existing network infrastructure. In: Green Computing Conference and Workshops (IGCC), 2011 International. IEEE, 2011. p. 1-8.

[26] BALAJI, Bharathan, et al. Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. ACM, 2013. p. 17.

[27] NGUYEN, Nam Tuan; ZHENG, Rong; HAN, Zhu. UMLI: An unsupervised mobile locations extraction approach with incomplete data. In: Wireless Communications and Networking Conference (WCNC), 2013 IEEE. IEEE, 2013. p. 2119-2124.

[28] PEI, Ling, et al. Inquiry-based bluetooth indoor positioning via rssi probability distributions. In: Advances in Satellite and Space Communications (SPACOMM), 2010 Second International Conference on. IEEE, 2010. p. 151-156.

[29] ANASTASI, Giuseppe, et al. Experimenting an indoor bluetooth-based positioning service. In: Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on. IEEE, 2003.

[30] Jin-Shyan Lee; Yu-Wei Su; Chung-Chou Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE , vol., no., pp.46,51, 5-8 Nov. 2007

[31] Bluetooth Smart - bluetooth.com/Pages/Bluetooth-Smart.aspx

[32] OMITTED FOR BLIND REVIEW

[33] Android market share - http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/

[34] OMITTED FOR BLIND REVIEW