POLITECNICO DI MILANO

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni
Dipartimento di Elettronica, Informazione e Bioingegneria DEIB

Occupancy Estimation through Bluetooth Classic and Low Energy packet sniffing applying Machine Learning

AntLAB
Advanced Network Technologies LABoratory
del Politecnico di Milano

Relatore: Ing. Alessandro Enrico C. Redondi
Correlatore: Ing. Edoardo Longo

Tesi di Laurea di:
Tomas La Carrubba, matricola 864107

Anno Accademico 2018-2019

Abstract

Information regarding occupancy is a key component to facilitate people's lives. Indeed, occupants presence have significant impact on space around us. It is interesting to know how many people are present in different environments.

Generally speaking, information about allocation and reservation of spaces make people feel more comfortable, allowing them to save their time using public transportation, to better manage energy consumption of indoor environments, etc.

The main goal of this thesis project is to perform space occupancy estimation using packet sniffing and Machine Learning algorithms. On the one hand, packet sniffing is carried out through Bluetooth Low Energy and Classic Bluetooth advertising packet detection and count. Packets have been captured using a sniffing hardware named Ubertooth One and the bluetooth Linux stack Bluez HCItool. On the other hand, machine learning techniques such as Linear Regression, Decision Tree Regression and Support Vector Regression were used to estimate occupancy. Despite we used low-cost wireless sniffers, it is shown that the task of occupancy estimation can be performed with a good level of accuracy.

Sniffing and Data Acquisition processes are discussed, also explaining how the features sets are built to best fit the occupancy predicting models.

Experimental results, expressed in terms of RMSE and MAPE indicators demonstrate the validity of the proposed system in both indoor and outdoor uncontrolled scenarios. Finally, we spend a few words on the reason why we believe in the potential of this non-intrusive methodology.

Sommario

L'informazione riguardo l'occupazione di un luogo è una componente chiave per facilitare la quotidianità delle persone. Infatti, la presenza di occupanti ha un impatto significativo sullo spazio intorno a noi. È interessante sapere quante persone sono presenti in diversi ambienti. In generale, conoscere le informazioni sull'assegnazione e la prenotazione degli spazi fanno sentire le persone più a loro agio, consentendo loro di risparmiare tempo utilizzando i mezzi pubblici, per gestire meglio il consumo energetico degli ambienti interni, ecc.

L'obiettivo principale di questo progetto di tesi è quello di eseguire la stima di occupazione dello spazio usando tecniche di "Sniffing" e "Machine Learning". Da un lato, il processo di Sniffing viene eseguito tramite la rilevazione e il conteggio di pacchetti pubblicitari "Bluetooth Classico" e "Bluetooth Low Energy". I pacchetti vengono catturati usando un dispositivo di sniffing, chiamato Ubertooth One, e lo stack Bluetooth di Linux, Bluez hcitool. Dall'altro lato, tecniche di Machine Learning, come Regressione lineare, Regressione a supporto vettoriale e alberi di decisione, sono state utilizzate per stimare l'occupazione. Nonostante abbiamo utilizzato sniffer wireless a basso costo, abbiamo dimostrato che l'obiettivo della stima di occupazione può essere eseguito con un ottimo livello di accuratezza. Vengono discussi i processi di sniffing e acquisizione dati, spiegando come vengono costruiti i gruppi di descrittori per adattarsi al meglio ai modelli di previsione di occupazione. I risultati sperimentali, espressi in termini di indicatori RMSE e MAPE, dimostrano la validità del sistema proposto in scenari sia interni che esterni incontrollati. Infine, dedichiamo alcune parole sul motivo per cui crediamo nel potenziale di questo metodo non intrusivo.

Ringraziamenti

Il primo ringraziamento vorrei dedicarlo al mio relatore Prof. Alessandro E. C. Redondi per avermi concesso la possibilità di conseguire il seguente lavoro di tesi, lo ringrazio per la sua costante disponibilità nei miei confronti e i suoi preziosi consigli.

Ringrazio il mio co-relatore Edoardo Longo, che ha visto questo lavoro nascere, crescere e svilupparsi, rappresentando per me un punto di riferimento per qualsiasi dubbio, approfondimento e suggerimento.

Ringrazio Andrea, che prima di essere stato il mio "secondo co-relatore" è uno dei miei più cari amici, che ha dato l'anima nei momenti più difficili, con i suoi consigli, con le sue preziosissime ore spese a darmi una mano.

Ringrazio i miei amici, quelli nuovi, e quelli che ci sono sempre stati. È anche grazie a loro che ho affrontato questi anni con allegria.

Ringrazio mio fratello Dario, che rappresenta un pilastro portante della mia vita, nati e cresciuti insieme, inseparabili. Ringrazio il suo ottimismo e la sua determinazione, la forza che mi ha trasmesso senza la quale avrei superato soltanto la metà delle difficoltà.

Ringrazio Noemi, lei che mi ha sempre ascoltato, compreso e sopportato, e che continua a farlo senza alcun segno di cedimento. Ringrazio le emozioni che mi regala, la serenità che mi trasmette con cui ho affrontato questi anni, i migliori di sempre.

Infine dedico questo lavoro ai complici di tutti i miei successi, alle due persone che mi hanno educato e sostenuto in questi anni, con tanti sacrifici, con tanto coraggio. A loro che hanno sempre creduto in me rendendo possibile tutto questo.
A voi Mamma e Papà.

Contents

Chapter 1

Introduction

Imagine we could estimate the number of people in a library, or a study hall, making people aware of the sits still available before leaving home. Also, we could have information on public spaces occupancy level such as a bus stop or a volleyball field. Information of this type can be easily collected leveraging wireless packets sniffing coming from smartphones or less complex personal devices. Indeed, the presence of personal devices with wireless communication capabilities other than smartphones is exponentially growing. It is estimated that being equipped with such devices will soon be a primary requirement.

Recent CISCO reports estimate that by 2021 there will be around 11.6 billions active mobile-connected devices, exceeding the forecasted global population of 7.8 billion, that is around 1.5 mobile devices per capita. Most of these devices will be smartphones or equivalent handheld personal ready devices; however, it is expected that a fraction as large as 30% will be constituted by wearable or M2M devices (smart watches, health and fitness trackers, etc.) that will communicate to the network either directly through embedded cellular connectivity or most likely through a smartphone with other wireless communication technologies such as WiFi, Bluetooth or Bluetooth Low Energy (BLE). The growth of the presence of these devices shows no signs of slowing down and it steadily becomes the cause of a new pervasive paradigm in computing and communications.

Internet of Things (IoT) is a recent communication paradigm that envisions a near future, in which the objects of everyday life will be equipped with microcontrollers, transceivers for digital communication, and suitable protocol

stacks that will make them able to communicate with one another and with the users, becoming an integral part of the Internet. The IoT concept, hence, aims at making the Internet even more immersive and pervasive. Furthermore, by enabling easy access and interaction with a wide variety of devices such as, for instance, home appliances, surveillance cameras, monitoring sensors, actuators, displays, vehicles, and so on, the IoT will foster the development of a number of applications that make use of the potentially enormous amount and variety of data generated by such objects to provide new services to citizens, companies, and public administrations. This paradigm indeed finds application in many different domains, such as home automation, industrial automation, medical aids, mobile healthcare, elderly assistance, intelligent energy management and smart grids, automotive, traffic management, and many others. The adoption of the IoT paradigm in urban context is of particular interest, as it responds to the strong push of many national governments to adopt ICT solutions in the management of public affairs, thus realizing the so-called Smart City concept.

The Smart City market is estimated at hundreds of billion dollars by 2020, with an annual spending reaching nearly 16 billions. This market springs from the synergic interconnection of key industry and service sectors, such as Smart Governance, Smart Mobility, Smart Utilities, Smart Buildings, and Smart Environment. These sectors have also been considered in the European Smart Cities project (http://www.smart-cities.eu) to define a ranking criterion that can be used to assess the level of "smartness" of European cities. The IoT vision can become the building block to realize a unified urban-scale ICT platform, thus unleashing the potential of the Smart City vision.

With the proliferation of Internet of Things (IoT) devices such as smartphones, sensors, cameras, and RFIDs, it is possible to collect massive amount of data for counting people. Indeed, both Bluetooth and Bluetooth Low Energy (BLE) are wireless communication protocols whose management frames can be easily captured with minimal hardware equipments and processed to perform user localization and tracking, behaviour estimation, device classification and de-anonymisation, market analysis and many others [1]. In this way, we can have real-time access to occupancy counts in different zones of the building and outdoor environments. Also, it is possible to locate most of the users carrying a wireless device. This real-time occupancy status information can be used in a variety of applications. For example, the smart building systems

of the future can adjust their energy consumption by intelligently controlling the HVAC, an accurate information of the number of occupants may allow energy savings ranging from 30% up to 80%, according to several studies [2][3][4], and also make the system respond promptly to any potential issues that can put the building off its track to carbon neutrality [5][6]. In addition to energy issues, real-time occupancy counting may also help rescuing survivors in case of emergency response applications. Finally, this information may also be used to improve building surveillance and security, and help in better deploying the wireless communication infrastructure for fulfilling ubiquitous throughput guarantees throughout the buildings.

Numerous methods have been developed to study occupancy in both indoor and outdoor environments. These methods can be categorized as "direct approach" and "indirect approach": the former is a method based on direct detection of occupants using positioning technologies such as passive infrared (PIR) motion detector, video camera and radio-frequency identification (RFID); the latter is a method based on indirect sensing management frames coming from wireless devices, CO2 concentration, temperature data in the near environment to estimate occupancy through the use of machine learning techniques.

The direct approach technologies rely on occupancy status analysis from data directly collected in the surrounding environment. While environmental sensors are a cheap solution although generally not very accurate, camera-based systems are much more precise but generally costly to deploy. Moreover, despite their effectiveness in detecting occupancy levels, privacy can be a major concern for deploying these technologies in the real world. For these reasons, our approaches is based on non-intrusive "indirect" occupant measurement methods exploiting wireless sniffing and machine learning algorithms. We show that the task of occupancy estimation can be performed accurately using low-cost Bluetooth sniffers, which capture management frames transmitted by Bluetooth personal devices even without a proper connection to a network nor user data to deliver. Our work provides an analysis of the existing approaches and help address the aforementioned issue by promoting the use of multi-modal data fusion that will be collected from the existing IoT network. Fusing data from Classic and Low Energy Bluetooth could improve the accuracy of occupancy detection while maintaining a low intrusiveness. By exploiting the synergy among the available data, information fusion techniques can filter noisy measurements

coming from IoT devices, and make predictions and inferences about occupancy status.

Chapter 2

State of the art

Occupancy detection is one of the most stimulating study that interests many researchers developing new methods.

Many approaches have been proposed in the literature by considering the use of different devices, assumptions, and goals. These approaches have certain drawbacks with respect to accuracy, cost, intrusiveness, and privacy. Accuracy, cost and intrusiveness are inter-related in the sense that with the increased cost, you can deploy additional devices (such as various sensors, RFIDS, cameras) and increase the accuracy of the system while at the same time increase the intrusiveness. Therefore, a wise method to reduce costs is to rely on the existing devices as much as possible. This automatically addresses the intrusiveness issue since there will be no need to deploy additional devices inside the rooms, and additional applications on the users' devices. Nonetheless, this raises the question of accuracy which may be severely affected.

In the chapter 1 we mentioned the division of direct and indirect approaches. In this chapter we focus on indirect approaches, dividing the chapter in two sections, we first propose a small summary on related Occupancy estimation direct-approach-based works, emphasising the pros and cons; second, we analyse indirect-approaches-based works distinguishing between Occupancy estimation through wireless technologies and Occupancy estimation through machine learning techniques such as Classification and Regression, emphasising similarities and results with respect to our work.

2.1 Direct approaches Video-based Occupancy estimation works

The direct approach consists in using video cameras for estimating the number of people occupying a space through image processing techniques [7]. The main works in this area are summarised in Table 2.1: the occupancy estimation problem is generally approached as a multi-class classification problem and the main performance figure used in such works is the classification accuracy. Cameras installed in public spaces generally raise privacy concerns: therefore they are either installed in such a way such that faces are not revealed, e.g. attached to the ceilings, or they are operated with very low image resolutions [8], however it does not solve the problem because the perception of the people present remains that of being continually observed. Other approaches ensuring privacy use cameras in the non-visible domain, such as passive infrared (PIR) sensors [9][10] or depth cameras [11]. Compared to indirect approaches solutions, video-based approach allows for higher estimation accuracy, but does not solve the aforementioned privacy issue and are also more costly to setup and maintain as well as being sensible to lighting conditions. For this reasons, some works propose hybrid data-fusion systems in which cameras are coupled with CO2 and other environmental sensors [12], in order to improve the estimation accuracy in poor lighting conditions.

Reference	**Sensor**	**Location**	**MAX #People**	**Accuracy**
Benezeth et al., [7]	RGB camera (QVGA)	Indoor	2	93%
Amin et al., [8]	RGB + thermal	Indoor	10	97%
Wahl et al., [9]	PIR	Indoor	3	99%
Munir et al., [11]	Depth camera	Indoor	12	100%
Wang et al., [12]	CO2 RGB camera	Indoor	12	91%

Table 2.1: Summary of video-based occupancy estimation works

2.2 Indirect approaches-based Occupancy estimation works

2.2.1 Environmental sensing Occupancy estimation using Machine Learning

Many works leverage environmental data to perform either presence detection (i.e., detecting if someone is present in an area or not) or occupation estimation (i.e., counting how many people are present). For what concerns indoor spaces, the gold standard input measurement used in this kind of works is the CO2 level, which is a good indicator of the number of person in a room and at the same time is able to preserve the privacy of the occupants [13]. Occupancy information is generally retrieved by analyzing the gradient of the CO2 level or by solving the air mass balance equation [14]. However, both methods assume that the indoor CO2 concentration is uniform. Therefore, in real applications, these methods suffer from common issues such as unpredictable opening of doors and windows and uncertainties involved with the CO2 concentration level or its gradient, which lead to poor estimation accuracy. Presence detection and occupancy estimation are generally treated as classification problems and solved with supervised machine learning tools. While presence detection is approached as a binary classification problem (absence/presence of people, [15]), occupancy estimation is solved with multi-class approaches and ground truth information is generally quantised in occupancy levels (e.g. low, medium, high, [16]) or directly used as class label when the maximum number of person to estimate is very low (e.g. below 10). The performance obtained with such methods are generally excellent for what concern presence detection, with classification accuracy close to 100%, whereas for occupancy estimation the results obtained are mediocre. To improve the performance, some works take into account other environmental sources of information such as light, temperature, humidity or even energy consumption of appliances [17]. Seung Ho Ryu and Hyeun Jun Moon tackle these issues, proposing a data-driven model for occupancy prediction using machine learning techniques. The experiments was conducted in a controlled space to collect the ground truth occupancy profiles, indoor CO2 concentrations and electricity consumption of lighting systems and appliances. The results show that using the decision tree and hidden Markov model (HMM) algorithms are well suited to account for occupancy detection at the current state and

Reference	Sensor	Location	MAX #People	Models used	Accuracy
Candanedo et al.,[15]	CO2 Light Temperature Humidity	Indoor	Absence/Presence	CART Binary Classifier	99%
Chen et al., [17]	CO2 Light Temperature Pressure	Indoor	10	SVM ANN KNN	71.6%
Ryu et al., [18]	CO2 Energy	Indoor	5	HMM	93.2%

Table 2.2: Related works using environmental data using Machine learning

occupancy prediction at the future state, respectively [18]. Another similar work was proposed by Sunil Mamidi and Yu-Han Chang, they implemented a multi-modal sensor agent that combines information such as motion detection, CO2 reading, sound level, ambient light, and door state sensing. Results show that these sensor agents can be used to accurately estimate the number of occupants in each room using machine learning technique such as linear regression, logistic regression, multi-layer perception, and support vector machines (SVM) [19]. These types of works have usually a very low level of intrusiveness, thanks to the fact that people do not perceive the presence of sensors due to their reduced size, have a low cost, but as we can see from the results in table 2.2, the level of accuracy of occupancy counting is low, and in those cases where the accuracy is high the experiments were conducted in "controlled" indoor environments, where the number of occupants is quite low, from zero to ten.

2.2.2 Wireless technologies Radio-based Occupancy estimation

Very recently, attention has been given to occupancy estimation systems which are based on wireless radio measurements rather than on traditional sensors. Beside being very cheap, such systems have the benefit of being able to work both in indoor and outdoor spaces. Many works approach the problem in a device-free fashion, without relying on the personal devices carried by people. Such works analyse the propagation characteristics of probing radio signals and correlate features derived by the Received Signal Strength (RSS) to the number of people occupying a certain space. Representative works include the one in [20], where an approach for estimating the total number of people based

on only WiFi power measurement was proposed, a couple of transmitter and receiver devices is used to assess the impact of a certain number of people on the signal strength indicator at the receiver or on line of sight (LOS) blocking effects. Authors develop a model for the probability distribution of the received signal amplitude as a function of the total number of occupants and use that as estimation methods. The work in [21] approaches the same problem using 13 radio transmitters and 9 receivers. Lars Mikkelsen, in [22], provides an estimation of public transport occupancy collecting WiFi probes emitted by WiFi enabled devices of the passengers, specifically estimating bus passenger load. Filippoupolitis proposed a new approach using Bluetooth BLE instead WiFi, this work is able to provide the location of a user using information from BLE beacons installed in a building. Three machine learning approaches (k-nearestneighbours, logistic regression and support vector machines) were used to determine the presence of occupants in indoor environment [23]. In [24], the authors present a system based on iBeacons for detecting the occupancy of a building. They have evaluated their system by predicting whether a user was inside or outside of a single room. Another approach presented in [25] aims at estimating a building's occupancy using Arduino hardware beacons that implement Apple's iBeacon protocol and Apple mobile phones. The evaluation only addresses the users presence inside or outside a room and does not give details on the individual room occupancy accuracy. The systems that use iBeacons need applications to be installed on user's devices to work properly, this raises up the intrusiveness of the system. Authors in [26] propose an occupancy estimation system based on the capture of both Wi-Fi and Bluetooth or Bluetooth Low Energy management frames transmitted from discoverable devices. The system leverages a supervised learning model to adapt to different spaces. This work is similar to our work, with the crucial difference that the system is not able to capture Bluetooth packets coming from Non-visible devices. Research studies on WiFi and Bluetooth/BLE frame sniffing and analysis targeting user tracking, device classification, social analysis and privacy issues can be found in [27][28][29][30].

A general comment on these works is that the occupancy estimation problem is again treated as a classification problem by the most for indoor environments only. The experimental evaluation of these works is therefore generally limited to less than 10 people to keep the problem tractable, and it is not clear if such

approaches scale to higher number of occupants. Moreover, the intrusiveness of the systems aforementioned could be increased. Our work tends to avoid these cons above cited: we treat the occupancy estimation using Regression instead Classification, which works well for counting people presence; we capture RSSI information from the packets of the devices to better adapt the model in the different spaces, indoor and outdoor; our method is able to capture Bluetooth packets from non-visible devices also; our method does not need any application to be installed on the user's devices. Therefore we propose a method that guarantees accuracy, non-intrusiveness and low cost.

Chapter 3

Theoretical fundamentals

This chapter is intended as a review of the theoretical fundamentals used in this thesis. The aim is to better comprehend the working principles of the technologies used in this work. In order to do this, first this chapter introduces the standard 802.15.1, spending a few words on the main features and the reasons why this technology has a strong impact in Telecommunications. Secondly, we give a review of Classic and Low Energy version of Bluetooth, showing some details of the architecture, connectivity, and others details, emphasising the discovery and advertising processes, differences and similarities.

3.1 Introduction on IEEE 802.15.1 Standard

IEEE 802.15.1, better known as Bluetooth, is the name given to a technology standard using short-range radio links, intended to replace the cable(s) connecting portable and/or fixed electronic devices. The standard defines a uniform structure for a wide range of devices to communicate with each other, with minimal user effort. Its key features are robustness, low complexity, low power and low cost. The technology also offers wireless access to LANs, PSTN, the mobile phone network and the Internet for a host of home appliances and portable handheld interfaces. The immediate need for Bluetooth came from the desire to connect peripherals and devices without cables. Bluetooth integration is further fueled by the demand for mobile and wireless access to LANs, Internet over mobile and other existing networks, where the backbone is wired but the interface is free to move. This not only makes the network easier to use but also

extends its reach. The advantages and rapid proliferation of LANs suggest that setting up personal area networks, that is, connections among devices in the proximity of the user, will have many beneficial uses. Bluetooth could also be used in home networking applications. With increasing numbers of homes having multiple PCs, the need for networks that are simple to install and maintain, is growing. There is also the commercial need to provide "information push" capabilities, which is important for handheld and other such mobile devices and this has been partially incorporated in Bluetooth. Bluetooth's main strength is its ability to simultaneously handle both data and voice transmissions, allowing such innovative solutions as a mobile hands-free headset for voice calls, print to fax capability, and automatically synchronizing PDA, laptop, and cell phone address book applications. These uses suggest that a technology like Bluetooth is extremely useful and will have a significant effect on the way information is accessed and used.

3.2 Classic Bluetooth (BT)

Bluetooth is managed by the Bluetooth Special Interest Group (SIG). Bluetooth is a wireless technology built to Personal Area Networks (PANs). Since Bluetooth operates in the unlicensed ISM band that is also used by other devices such as 802.11 networks, baby monitors, garage door openers, microwave ovens etc, there is possibility of interference, Bluetooth uses Frequency Hop Spread Spectrum (FHSS) to avoid any interference. A Bluetooth channel is divided into time slots each 625 μs in length. The devices hop through these timeslots making 1600 hops per second, with adaptive frequency-hopping (AFH) enabled. This trades bandwidth efficiency for reliability, integrity and security.

3.2.1 Device classes

Devices equipped with Bluetooth are divided into 3 transmission power classes:

- Class 1;
- Class 2;
- Class 3;

Class	Max Trasmission Power	Range
Class 1	100 mW (20dBm)	100m
Class 2	2.5 mW (4dBm)	10m
Class 3	1 mW (0dBm)	1m

Figure 3.1: Classes of Bluetooth

These Classes determine the maximum trasmission power (ERP) in radio frequency, including the increase due to the gain in transmission of the antenna of the device and the maximum range of communication, without obstacles, within which the connection between BT devices can occur.

3.2.2 Timing and clock

Bluetooth technology plans to synchronize most operations with a clock signal in real time. It serves, for example, to synchronize data exchanges between devices, to distinguish between retransmitted or lost packets, to generate a predictable and reproducible pseudo-random sequence. The Bluetooth clock is made with a 28-bit counter that is set to 0 when the device is switched on and immediately continues, increasing every 312.5 μs (half slot then). The counter cycle covers approximately one day's duration.

Each Bluetooth device has its native clock (CLKN) that controls the timing of that device. In addition to this value, just for each device, Bluetooth defines two other clocks: CLK: this is the clock of the piconet, coincides with the CLKN of the master unit of the piconet. All units active in the piconet must synchronize their CLKN with the CLK. The synchronization is done by adding an offset to the CLKN of the slave to make it coincide with the pictonet CLK. CLKE: also this clock is derived by an offset from the CLKN and is used by the master in the specific case of creating a connection to a slave, and before that slave has synchronized with the master (ie when it is a new slave) . The first 2 bits of the counter are used directly to delimit the slots and the so-called "half slots", for the transmission and reception of the packets; they also serve to establish Tx (transmission) or Rx (reception) slots over time, depending on whether the device in question is operating as master or slave. A transmission by the master will always start when CLK [1: 0] = 00 (even index slot), while a slave transmission will always start when CLK [1: 0] = 10 (odd index slot).

3.2.3 Connectivity

Bluetooth is a packet-based protocol with a master/slave architecture. Two or more devices connected together form a piconet. Devices within a piconet can be of two types: master or slave. The master is the device that within a piconet takes care of everything related to the synchronization of the clock of other devices (slaves) and the sequence of frequency jumps. Slaves are units of the piconet synchronized to the master's clock and to the frequency channel. Slaves can belong to several piconets simultaneously, while the master of a piconet can at most be the slave of another. The connections that can be established between the different devices are of two types: Connection Oriented and Connectionless. A Connection-Oriented connection requires establishing a connection between the devices before sending the data; while a link without a connection requires no connection before sending the packets. The transmitter can at any time start sending its own packets as long as it knows the address of the recipient.

The Bluetooth technology defines two types of connections to support voice and data transfer applications: a Asynchronous ConnectionLess Service (ACL) and a Synchronous Connection-Oriented Service (SCO).

ACL supports data type traffic and is based on a best-effort service. The information conveyed can be of the user or control type. SCO, on the other hand, is a link that supports connections with real-time and multimedia traffic. The ACL connection supports packet-switched connections, point-to-multipoint connections, and symmetric or asymmetric connections. In the case of symmetrical connections, the maximum data rate is 433.9 kbit/s in both directions; while, for asymmetrical connections, 723.2 kbit/s in one direction and 57.6 kbit/s in the opposite one are reached. A slave can only transmit if it had received a packet from the master in the previous slot. In these types of connections, packet retransmission is typically applied.

The SCO connection provides circuit-switched connections, point-to-point connections and symmetrical connections. This type of connection is generally used for the transport of the voice in 64 kbit/s channels. The master can support up to three SCO connections to the same slave or to different slaves belonging to the same piconet. A slave, on the other hand, can support up to three SCO connections to the same master, or two if the links have been created by several masters. Because of the delayed sensitivity of these packets (they

carry real-time data), no retransmission is expected in case of error or loss.

Each device, when connecting to a Bluetooth network, identifies the other devices via a 24-bit code (COD), activating the correct services.

3.2.4 Architecture

As with the OSI architecture shown in figure 3.2, Bluetooth specifies a tiered approach in its protocol structure. Different protocols are used for different applications. Regardless of the type of application, however, the Bluetooth protocol stack always leads to the use of data-link and physical levels. Not all applications use all the protocols of the Bluetooth stack, in fact, it is represented on multiple vertical levels, above which there is a specific application.

Going down a little more in detail, it is possible to identify the main functions performed by the most important protocols of the Bluetooth stack:

- Bluetooth Radio: defines the requirements of the radio frequency part. This is where the radio signals are processed.

- Baseband: enables the physical connection between devices within a piconet. This level is based on inquiry and paging procedures for synchronization and connection of bluetooth devices. It allows to establish the two types of connection (ACL and SCO) mentioned above.

- LMP: is responsible for organizing the connection, controlling between bluetooth devices and controlling and negotiating packet size. It is also used for security: authentication and encryption, generation, exchange and key control. It also controls the various power management modes (park, sniff, hold) and the connection status of a device within the piconet. LMP messages are filtered and interpreted by the link manager at reception, as a result they will never be transmitted to the higher levels. These messages take priority over packets that carry user data.

- L2CAP: performs the multiplexing of the higher level protocols, the segmentation and reassembly of the packets and the transport of information related to the QoS (Quality of Service) or it is possible to request a certain QoS to be reserved for a given link. L2CAP allows higher level protocols and applications to transmit and receive data packets larger than 64 kb. It defines only a connectionless connection. Audio channels are usually

run on SCO connections; to overcome this problem, audio data can be sent in packets of protocols that run on L2CAP.

- RFCOMM: emulates a serial port (RS-232) on the L2CAP protocol. This level is necessary because there are applications (such as OBEX) that use a serial transmission mechanism.
- TCS BIN: operates at the bit level and defines the control signals for voice and data calls between Bluetooth devices and the procedures for managing groups of TCS devices.
- SDP: is an important element within Bluetooth technology, as it allows applications to have information about the devices, the services offered and the features of the services available. After identifying the device that implements a certain service, it is possible to establish a connection.
- AUDIO: the function of this layer is to encode the audio signal. Two techniques can be adopted: log PCM and CVSD; both provide a 64 kbit/s bit stream. The PCM (Pulse Code Modulation) coding consists of an uneven 8-bit quantization. In the CVSD (Continuous Variable Slope Delta Modulation) encoding the output bit indicates whether the predicted value is greater or less than the value of the incoming waveform, consisting of a PCM signal with uniform quantization. The pitch is determined by the slope of the waveform.

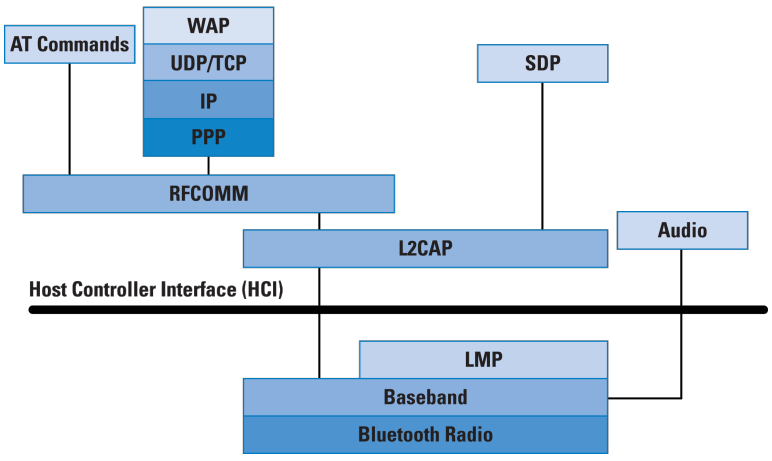

Figure 3.2: Bluetooth Architecture

3.2.5 Inquiry and Discovery Processes for Bluetooth devices

Bluetooth devices have two major states, connection and standby, and seven substates. Connection is used for communication whereas standby is the power-save mode in which no transmissions occur. The substates are used for joining as a slave in a piconet. The page substate is used by the master for adding slaves. For this paging procedure, the clock counter (28-bit, CLK) and the MAC address of the devices must be used. In the inquiry procedure this information is exchanged in order to set up a lasting connection. As the master sends its address and clock value, the slave can construct the correct hopping sequence of the piconet by that information. The master also provides the slave with a 3-bit identification number. This limits the number of slaves in a piconet to seven. In order to exchange this kind of information between devices, a process must take place to actually find each other (discover).

During the inquiry (discovery) process, the master enters the inquiry substate, whereas the slaves enter the inquiry scan substate. During the inquiry process the master trasmits inquiry packets on different frequencies, and the slave, called also inquiry scanner, scans those frequencies. An inquirer transmits two inquiry packets on two different frequencies during one regular transmission timeslot. 625 μs later, the inquirer listens on the same frequency. The inquiry scanners, in scan mode, change the frequency on which they listen every 1.28 seconds. In those 1.28 seconds they scan for 11.25 ms only. After receiving an inquiry packet, the inquiry scanner replies with an FHS (Frequency Hopping Synchronization) packet 625 μs later, and enters a backoff period between 0 and 1024 timeslots (0-640 ms). This FHS packet contains the device's address, its clock offset and a CRC code. Using this information a link can be established.

Moreover, we can distinguish two cases in order to begin a Bluetooth connection between two devices, the target device can be:

- "Non-Discoverable": if not all devices are able to see it;
- "Discoverable": if every device is able to see it;

In the first case only devices which know the device address of the target device can reach it and initiate the pairing process. In the second case, the target device is visible to everybody and the pairing process can be easily started. This is not an information to neglect, we will see later how much crucial this information is and how it impacts on our purpose.

Basically, the device which wish to set up a connection sends inquiry packets as explained before, and all the visible devices, in scan mode, can answer. Then a inquiry scan process starts, and zero or more devices can be discovered.

The inquiry scan provides some information, we list only those of our interest down below:

- Unique MAC address: a physical address assigned uniquely to each device;
- Received Signal Strength Indicator (RSSI): a measure of the power present in a received radio signal;

The reason why we paid more attention to these two fields will be clarified in chapter 4.

3.2.6 Topology Options

To best meet the wireless connectivity needs of different applications, Bluetooth technology supports multiple topology options. The Bluetooth specifications provide 3 types of topologies are Point-to-Point, Point-to-Multipoint and Mesh, several piconets can be connected to each other in a topology called scatternet:

- Point-to-Point: it is a network topology used for establishing one-to-one (1:1) device communications. The point-to-point topology available on Bluetooth Basic Rate/Enhanced Data Rate(BR/EDR) is optimized for audio streaming and is ideally suited for a wide range of wireless devices, such as speakers, headsets, and hands-free car kits. The point-to-point topology available on Bluetooth Low Energy (LE) is optimized for data transfer and is well suited for connected device products, such as fitness trackers, health monitors, and PC peripherals and accessories.
- Broadcast (Point-to-Multipoint): it is a network topology used for establishing one-to-many (1:m) device communications. The broadcast topology only available on Bluetooth LE is optimized for localized information sharing and is ideal for location services such as retail point-of-interest information, indoor navigation and wayfinding, as well as item and asset tracking.
- Mesh: it is a network topology used for establishing many-to-many (m:m) device communications. The mesh topology only available on Bluetooth

LE enables the creation of large-scale device networks and is ideally suited for control, monitoring, and automation systems where tens, hundreds, or thousands of devices need to reliably and securely communicate with one another.

3.3 BLuetooth Low Energy (BLE)

Bluetooth Low Energy (BLE), sometimes referred to as "Bluetooth Smart", is a light-weight subset of Classic Bluetooth and it was introduced as part of the Bluetooth 4.0 core specification. While there is some overlap with BT, BLE actually has a completely different lineage and was started by Nokia as an in-house project called 'Wibree' before being adopted by the Bluetooth SIG. Compared to Classic Bluetooth, Bluetooth Low Energy is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. Bluetooth Low Energy is a Wireless Personal Area Network technology (WPAN) aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Bluetooth Low Energy is expected to be incorporated into billions of devices in the next few years. The number of devices integrating BTLE is expected to grow by 2.9 billion devices per year by 2016. BLE represents a trade-off between energy consumption, latency, piconet size, and throughput. According to theoretical results, the lifetime of a BLE device powered by a coin cell battery ranges between 2 days and 14 years. The number of simultaneous slaves per master ranges between 2 and 5,917. The minimum latency for a master to obtain a sensor reading is 676 µs.

3.3.1 Radio Channels

BLE radio uses the 2.4 GHz ISM (Industrial, Scientific, and Medical) band to communicate and divides this band into 40 channels on 2 MHz spacing from 2.4000 GHz to 2.4835 GHz, starting at 2402 MHz.
As we can see in the figure 3.3 below, the 40 channels are divided into 3 Advertising Channels (Ch. 37, 38, 39), and 37 Data Channels (Ch. 0-36). Advertising Channel are used for:

- Device Discovery;
- Connection Establishment;

- Broadcast Transmissions;

Data Channel instead for:

- Bidirectional communication between connected devices;
- Adaptive frequency hopping used for subsequent connection events;

When transmitting data, the BLE radio transmits at 1 Mbps, with 1 bit per symbol. The radio is optimized for sending small chunks of data quickly. The BLE radio uses Gaussian frequency-shift keying (GFSK), whereby the data pulses are filtered with a Gaussian filter before being applied to alter the carrier frequency, in order to make the frequency transitions smoother. Since the advertising channels form the basis for how BLE operates, they have been assigned center frequencies that minimize overlapping with the most common 802.11 channels. In figure 3.3 we illustrate a picture representing the channels and the frequency hopping.

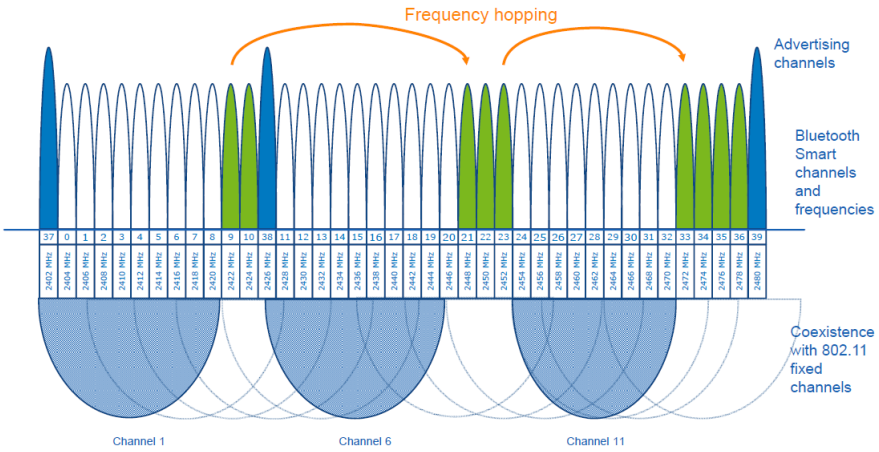

Figure 3.3: Radio channels and frequency hopping

3.3.2 Architecture

The following diagram in figure 3.2, depicts the architecture (major layers) of the BLE protocol stack divided in: Controller Layer, Host Layer and Application Layer. The Bluetooth Low Energy (BLE) Controller Layer is divided in:

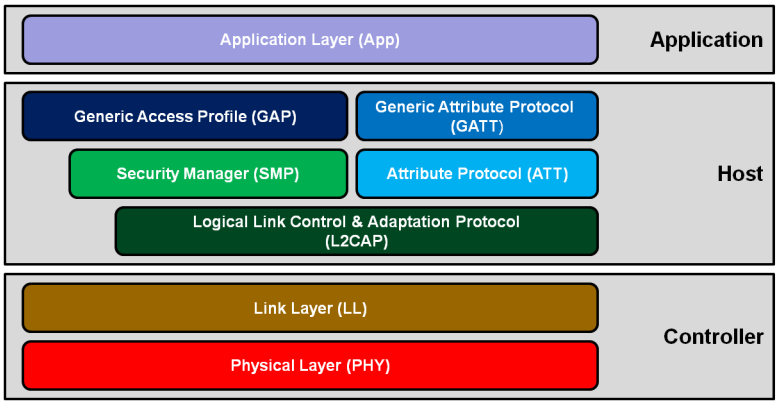

Figure 3.4: BLE Architecture

- Physical Layer (PHY): The BLE physical layer contains the analog communications circuitry responsible for translation of digital symbols over the air. It is the lowest layer of the protocol stack, and provides its services to the Link Layer.

- Link Layer (LL): The Link Layer is the part that directly interfaces to the PHY. It is responsible for advertising, scanning, and creating/maintaining connections.

The Host Layer is divided in:

- Generic Access Profile (GAP): it controls connections and advertising process. GAP is what makes your device visible to the outside world, and determines how two devices can (or can't) interact with each other.

- Generic Attribute Protocol (GATT): it comes into play once a dedicated connection is established between two devices, meaning that you have already gone through the advertising process governed by GAP. GATT defines the way that two Bluetooth Low Energy devices transfer data back and forth using concepts called "Services" and "Characteristics". It makes use of a generic data protocol called the "Attribute Protocol" (ATT), which is used to store Services, Characteristics and related data in a simple lookup table using 16-bit IDs for each entry in the table.

- Security Manager (SMP): The SMP (Security Manager Protocol) offers applications running over a Bluetooth Low Energy stack access to the

types of services like Device Authentication and Authorization, Data Integrity and Privacy, what services and keys are used for the communication between two devices are established during the SMP Pairing procedure which is performed by the SMP and set up by the Application according to its needs.

- Logical Link Control Adaptation Protocol (L2CAP): The L2CAP layer transfers data between the upper layers of the host (GAP, GATT, application) and the lower layer protocol stack. This layer is responsible for protocol multiplexing capability, segmentation, and reassembly operation for data exchanged between the host and the protocol stack.

It is worth spending a few more words on GAP protocol because, for our purpose, the most important thing is how the Advertising and Scan Response processes take place.

3.3.3 Advertising Process with GAP

Before two Bluetooth low energy devices can connect and get it on (technical term meaning "interact and exploit each other's services") they need to find each other and decide if one is interested. GAP is responsible for helping devices find each other and connect. We call this the discovery process.

The discovery process involves devices wishing to be discovered doing something known as 'advertising'. An advertising device emits small packets containing data deemed useful to scanning devices. How often advertising packets are emitted is a parameter which may vary considerably according to the device type or use case. Devices wishing to find other devices to connect to engage in a process called 'scanning', and receive and process advertising packets from other devices.
To better explain the Advertising process it is necessary to distinguish the device roles:

- Peripheral device: is that device that wants another device to connect to it. They are usually small, low power and such a device that can connect to a much more powerful central device. Peripheral devices are things like a heart rate monitor, a BLE enabled proximity tag, etc.

- Broadcaster devices: are often Bluetooth beacons, these devices advertises themselves but are not willing to accept connections. For scanning devices, sometimes the content of the advertising packet is the only thing needed, there being no intention to connect at all and again. When this is the case, the scanning device is termed a Observer.

- Central devices: are usually mobile phone or table, devices that have more processing power and memory and are interested in connecting to the other device.

As we have seen in the radio channels paragraph there are three channels which can be used for advertising one, two or all three may be used. In contrast, data packets are subject to adaptive frequency hopping across the other 37 channels. When advertising is using more than one of the three channels (and using all three is quite typical) then each time an advertising packet is transmitted, the same packet is transmitted on each of those designated channels.

It is important what advertising devices put in those advertising packets. The data part of an advertising packet consists of fields called AD Types. Examples of AD Types include Local Name, Service UUID and Manufacturer Specific Data to name but a few of the available types. Developers choose what to include in advertising packets from the list of AD Types but are constrained by the amount of available space and so they must make choices. If the device is a GAP Broadcaster then the AD types they intend to use to contain the data they want their device to broadcast are chosen. If it is a GAP Peripheral, they consider how best to indicate to a scanner this is a device it should be interested in connecting to. A common approach is to use the Service UUID AD Type to include a list of the UUIDs of the most important GATT services which the device has. Scanning devices should be able to deduce that the device is of interest from this list. If there is not a defined AD Type for the kind of data the device wishes to advertise, the Manufacturer Specific Data field can be used. This field is designed to contain anything you like. The only rule is that the first two bytes must contain the Company Identifier for the company whose data format is being used within the Manufacturer Specific Data field. There are four different types of advertising which are possible, some more commonly used than others:

- General Advertising (Connectable): this is the most basic type of adver-

tising and one which acts as a general invitation to some other device.

- Directed Advertising: this is a special-purpose type of advertising, designed to invite a specific peer device to connect as quickly as possible. It contains the address of both the advertising device and the device being invited to connect. On receipt of this advertising packet, the receiving device (known now as the "initiator") will immediately and automatically send a connect request. In direct advertising mode, advertising packets are sent very frequently, every 3.75 milliseconds, but for no more than 1.28 seconds. This is so that advertising channels do not get congested.

- Non-connectable Advertising: GAP Broadcasters use non-connectable advertising. How this is indicated in advertising packets is explained below in the section on "Flags". Devices whose sole purpose it to perform non-connectable advertising only need to be equipped with a transmitter. They do not need a receiver in their controller.

- Discoverable Advertising (Non-connectable): this mode can be used for what might be thought of as "extended broadcasting". Devices performing discoverable advertising do not accept connections but they can respond to Scan Requests and therefore in contrast to non-connectable advertising, they do need a receiver as well as a transmitter.

The Bluetooth Low Energy Link Layer has only one packet format used for both advertising channel packets and data channel packets.

There are seven advertising channel PDU types, each having a different payload

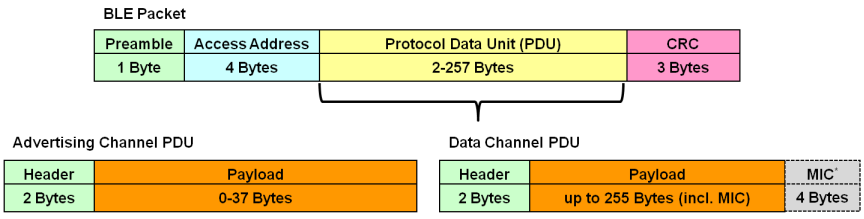

Figure 3.5: BLE packet format

format and function:

- Advertising PDUs: ADV_IND, ADV_DIRECT_IND, ADV_NONCONN_IND, ADV_SCAN_IND.

- Scanning PDUs: SCAN_REQ, SCAN_RSP.
- Initiating PDUs: CONNECT_REQ.

3.3.4 Discovery Modes

The discovery modes are crucial to our project because, they are concerned with how a peripheral advertises its presence and what centrals can/should do with that information. The following table matches several peripheral "discovery" modes with applicable central "discovery" procedures:

Discovery Modes	**Applicable Roles**	**Applicable Peer Procedure**
Non-Discoverable	Peripheral	N/A
Limited-Discoverable	Peripheral	Limited and General-Discoverable
General Discoverable	Peripheral	General-Discoverable

Table 3.1: Discovery Modes

As we can see in the table 3.1, the discovery modes are: Non-discoverable, Limited-discoverable and General-discoverable:

- Non Discoverable Mode: this state indicates that the peripheral does not desire to be discovered.
- Limited Discoverable Mode: this state indicates the peripheral's desire to reconnect to a specific peer. It is identified by the transmission of ADV_DIRECT_IND advertising packets in short bursts. These packets contain the MAC address of the desired Central device.
- General Discoverable Mode: this state indicates the peripheral's desire to be discovered by peers for connection establishment. It is identified by the transmission of ADV_IND advertising packets at regular intervals. It represents the initial "factory default" state for the peripheral.

3.3.5 Connection Establishment Modes

These are primarily concerned with how a Central selects which Peripheral to interact with. The table 3.2 matches several peripheral "connection" modes

with applicable central "connection" procedures:

Connection Modes	Applicable Roles	Applicable Peer Procedure
Non-Connectable	Peripheral	N/A
Undirected-Connectable	Peripheral	General Connection Establishment
Directed-Connectable	Peripheral	Directed Connection Establishment

Table 3.2: Connection Modes

- Non Connectable Mode: the peripheral does not need to exchange any data information, usually this type of devices are sensors that do not need to connect to any device. They store the information they want to broadcast in ADV_NONCONN_IND advertising packets and send them periodically.
- Undirected Connectable Mode: a peripheral is automatically in this "connection mode" when operating in "General-Discoverable" mode as discussed above. It is sending ADV_IND packets (which are promiscuous), and looking for a connection with any peer ("Undirected").
- Directed Connectable Mode: in this case the peripheral device is in "connection mode" but only with a specific device.
  It sends ADV_DIRECT_IND advertising packets to reach the device.

3.3.6 Discovery and Connection process

We summarize the Discovery and Connection Process as follows: first discussing the the Unicast Connection (Peer-to-Peer) and then the Broadcast one.

Unicast (Peer-Peer) Connection

The diagram below, in fig 3.7, depicts two BLE hosts, initially in a Standby (unconnected) state. They enter a Discovery state whereby the device wishing to be discovered becomes the Advertiser and the host wishing to connect becomes a Scanner.

There are four types of advertising that can be performed: *General*, *Directed*, *Non-Connectable*, and *Discoverable*. Each time a device advertises it transmits the same packet in each of the three advertising channels. This sequence of events is called an advertising event. The Advertiser sends advertising packets containing basic information about the host. All Scanners receive these packets. There are two ways to send advertising out with GAP. The Advertising Data payload and the Scan Response payload. Both payloads are identical and can contain up to 31 bytes of data, but only the advertising data payload is mandatory, since this is the payload that will be constantly transmitted out from the device to let central devices in range know that it exists, and this is crucial to our purpose. The scan response payload is an optional secondary payload that central devices can request, and allows device designers to fit a bit more information in the advertising payload such a strings for a device name, etc.

A peripheral device will set a specific advertising interval, and every time this interval passes, it will retransmit its main advertising packet, as shown in figure 3.6. A longer delays saves power but feels less responsive if the device only advertises itself once every 2 seconds instead of every 20ms. If a listening device is interested in the scan response payload (and it is available on the peripheral) it can optionally request the scan response payload, and the peripheral will respond with the additional data.

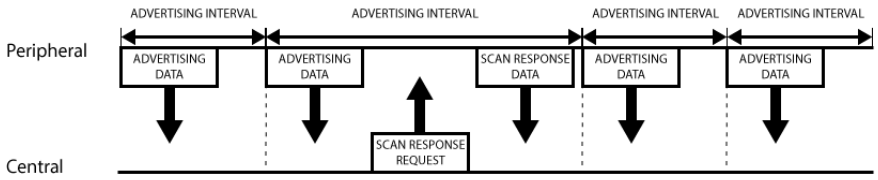

Figure 3.6: Advertising intervals

At some point, the Scanner (after filtering/analyzing information contained in the advertising packets) becomes an Initiator and decides to initiate a connection with a specific advertiser. There are two types of Scanners:

- Passive Scanners: the scanner simply listens for advertising packets. The advertiser is never aware that packets were received.
- Active Scanners: is typically used when the potential central device would

like more information than can be provided in an ADV_IND packet, before making a decision to connect to it. In an advertising interval, the scanner issues a SCAN_REQ packet. The advertiser responds with more information in a SCAN_RSP packet.

Once a Scanner has acquired enough information to decide which Advertiser to connect to (including its MAC address, RSSI, etc.), it becomes an Initiator, initiating a connection process. This is known as the Connecting phase and is highlighted by the Initiator sending a CONNECT_REQ advertising packet to the Advertiser, as we can see in the picture.

Once the CONNECT_REQ packet is received, the devices are connected and data packets can be exchanged. The Initiator becomes the Link Layer Master, while the Advertiser becomes the Link Layer Slave. This is known as the Connected phase.

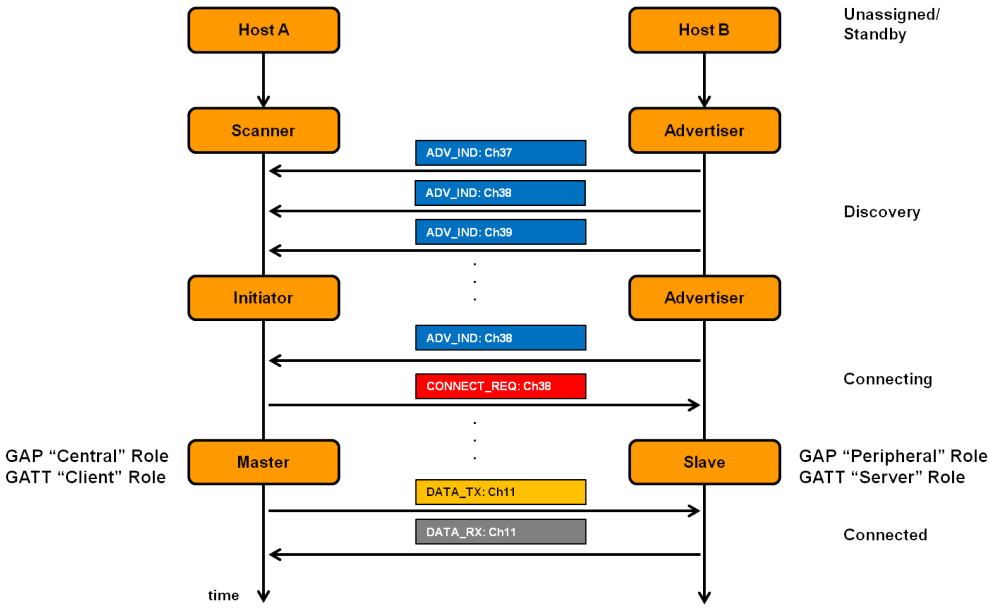

Figure 3.7: Discovery and Connection process

Broadcast Connection

While most peripherals advertise themselves so that a connection can be established and GATT services and characteristics can be used (which allows for much more data to be exchanged and in both directions), there are situations where devices only want to advertise data. The main use case here is where you want a peripheral to send data to more than one device at a time. This is only possible using the advertising packet since data sent and received in connected mode can only be seen by those two connected devices. By including a small amount of custom data in the 31 byte advertising or scan response payloads, you can use a low cost Bluetooth Low Energy peripheral to sent data one-way to any devices in listening range. This is known as Broadcasting in Bluetooth Low Energy, , as shown in the illustration below, figure 3.8, the Broadcast Network Topology.

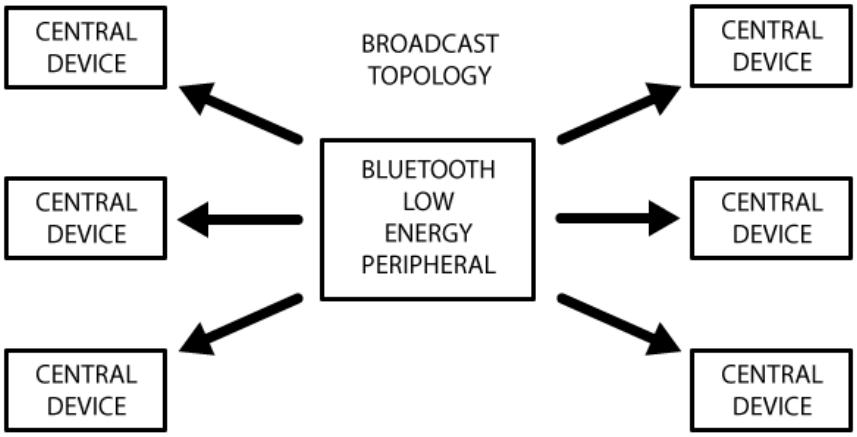

Figure 3.8: Broadcast Topology

For broadcast connections, the link layer roles do not change. The defined roles are Broadcaster (the host sending the packets) and Observer. Messages can be: one-way, one-to-many and ADV_IND is one of three advertising packet types that can be used by Broadcasters to broadcast data to Observers. See Figure 3.9.

Once you establish a connection between your peripheral and a central device, the advertising process will generally stop and peripheral will typically no longer be able to send advertising packets out anymore, and you will use

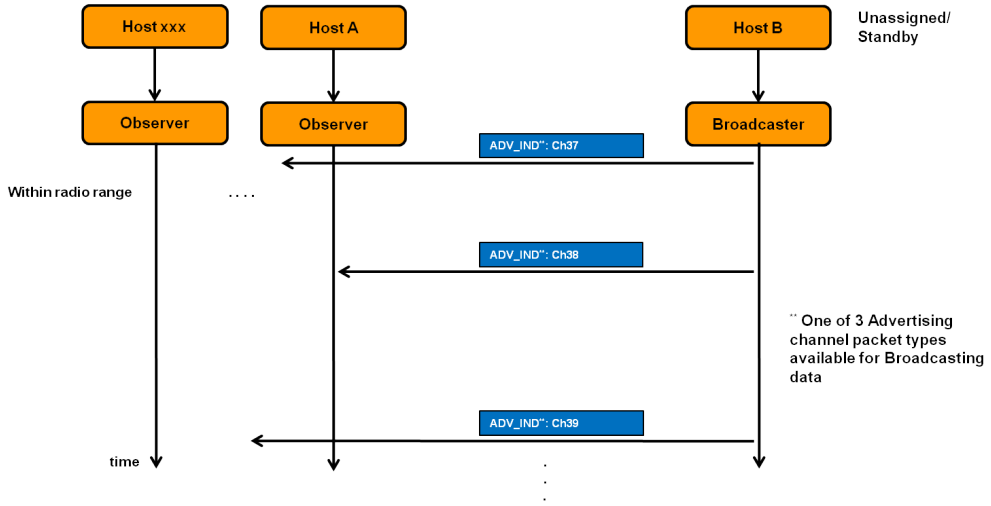

Figure 3.9: Example of a Broadcaster sending packets

GATT services and characteristics to communicate in both directions. GATT comes into play once a dedicated connection is established between two devices, meaning that you have already gone through the advertising process governed by GAP.

The most important thing to keep in mind with GATT and connections is that connections are exclusive. What is meant by that is that a BLE peripheral can only be connected to one central device (a mobile phone, etc.) at a time. As soon as a peripheral connects to a central device, it will stop advertising itself and other devices will no longer be able to see it or connect to it until the existing connection is broken.

3.4 Differences and similitudes between Bluetooth radio versions: BT and BLE

In this section we summarise the differences and similitudes of the two radio versions.

3.4.1 Differences

The main differences between Classic and BLE Bluetooth protocols can be summarised as:

- Both Bluetooth versions operate in the unlicensed ISM band at 2.4 GHz, with the difference that the Classic one uses 79 channels between 2.402 GHz to 2.480 GHz, and the BLE one 40 channels where 3 used for advertising process and the rest for data trasmission.

- Classic Bluetooth radio version includes multiple PHY options that support data rates from 1 Mb/s to 3 Mb/s, and supports multiple power levels, from 1 mW to 100 mW, as well as multiple security options. The Bluetooth LE radio provides developers a tremendous amount of flexibility, including multiple PHY options that support data rates from 125 Kb/s to 2 Mb/s, multiple power levels, from 1 mW to 100 mW, as well as multiple security options up to government grade.

- Moreover, Classic Bluetooth supports a point-to-point network topology that is optimized for audio streaming. Instead, Bluetooth LE also supports multiple network topologies, including a point-to-point option used for data transfer, a broadcast option used for location services and a mesh option used for creating large-scale device networks.

3.4.2 Similitudes

The two radio versions have two main similitudes, RSSI and MAC device address. We will see in the next chapter how crucial these similitudes are for our purpose.

Received Signal Strength Indicator (RSSI)

One of the most important concept used in the thesis is the RSS indicator. The Received Signal Strength Indicator (RSSI) is a measurement of the power present in a received radio signal after the antenna of the receiver device. Usually, the RSSI is represented in decibels referenced to one milliwatt called "dBm". Thus, when an RSSI value is represented in a negative form (e.g. -100), the closer the value is to 0, the stronger the received signal has been, therefore, the higher the RSSI number, the stronger the signal. The two versions work with the same range of power levels and because of that the RSSI provides more or less the same metric evaluation of distance.

Device MAC Address

MAC address is the acronym of Media Access Control Address. It is an unique identifier of a IEEE 802 network interface. Some examples of IEEE 802 standards are: ethernet, WiFi, ZigBee, FDDI (Fiber Distributed Data Interface) and Bluetooth. In our case MAC address is a fundamental information because it identifies uniquely a particular network interface of the device. The structure is a 12 digits (48 bits or 6 bytes) address for both radio versions, usually written in the following format:

- MM:MM:MM:SS:SS:SS

The leftmost 6 digits (24 bits) called prefix is associated with the adapter manufacturer, called OUI (Organizationally Unique Identifier). Each vendor registers and obtains MAC prefixes as assigned by the IEEE. Vendors often possess many prefix numbers associated with their different products. Discover on the web the vendor from the prefix is quite easy. The rightmost digits of a MAC address represent an identification number for the specific device. It is called Network Interface Controller (NIC). Among all devices manufactured with the same vendor prefix, each is given its own unique 24 bits number. To safeguard user privacy, manufacturers can make use of a Bluetooth Smart feature known as "LE Privacy".

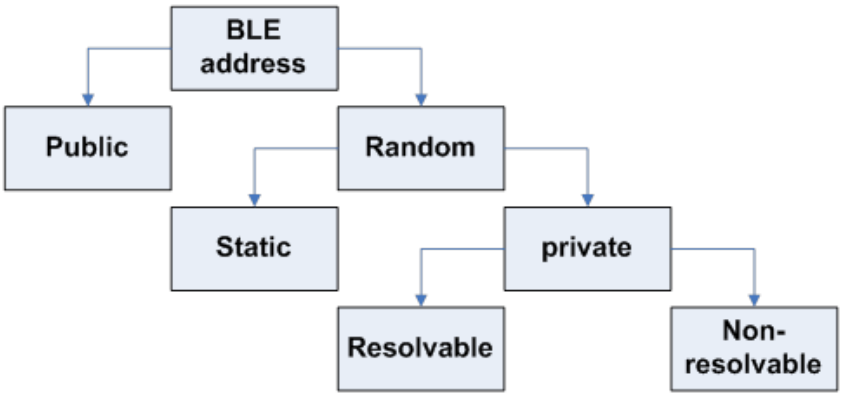

Figure 3.10: MAC address Randomization

As shown in figure 3.10, there are two macro types of device addresses:

- **Public Device Address**: this is the standard, IEEE-assigned 48-bit

universal LAN MAC address which must be obtained from the IEEE Registration Authority. It is divided into two fields:

  - IEEE-assigned company ID held in the 24 most-significant bits
  - Company-assigned device ID held in the 24 least significant bits

- **Random Device Address**: since all BLE packets include a Device Address, it's possible to track the BLE device as it's moving and communicating, unless it changes its address periodically. BLE adds the ability to periodically change the address. Two Random Address Types are provided:

  - Static Address: A 48-bit randomly generated address. A new value is generated after each power cycle.
  - Private Address: When a device wants to remain private, it uses private addresses. These are addresses that can be periodically changed so that the device can not be tracked. These may be resolvable or not:

    * Resolvable Private Address: This is an address that can be resolved through a pre-shared hash key. Only the trusted entities that have your pre-shared key can identify you. For all other entities, the address seems to be randomly changing and untrackable.
    * Non-Resolvable Private Address: This is an address that is random and can not be "expected". A possible use case: a device that already communicated a non-resolvable address to a peer for a reconnection.

Chapter 4

Proposed System

Traditional systems for occupancy estimation that rely either on environmental sensors (CO2,temperature, humidity) or video cameras have drawbacks of privacy, intrusiveness and also cost. Our goal is to perform occupancy detection to avoid the traditional systems drawbacks. Due to the capillary spread of personal devices equipped with wireless communication capabilities we proposed a cheap and accurate occupancy estimation system based on the capture of Classic Bluetooth and BLE management frames transmitted from users' devices. This is a non-intrusive indirect approach, and so it is less problematic for what concerns privacy issues. The system is implemented on low-cost hardware and leverages a supervised learning model to adapt to different spaces.

In every project there are theoretical basis and a practical application of them. In the chapter 3 the theoretical fundamentals were discussed while this chapter is intended as an explanation of the system's implementation.
The chapter is divided in two main parts:

- **Tools**: in this part we discuss about the used materials, *BlueZ* and the *Ubertooth One* sniffer. After a brief introduction on the characteristics of them, we discuss the main functionalities and properties.

- **Code**: this part covers the explanation of the python-coded programs used to handle the management frames transmitted from usersâ devices and processed by the sniffer.

4.1 Tools

In this section we discuss the tools we needed to build the Sniffing process, giving an overview on the main features used.

4.1.1 Bluez

In the Linux kernel-based family operating system, the Bluetooth stack is managed by Bluez. The most useful command of Bluez is *hcitool*. Hcitool (Host Controller Interface Tool) is used to configure Bluetooth connections and send some special command to the Bluetooth devices. The main functionalities are to discover (inquire a remote device), add, and manage devices on the piconet; to configure controller properties; to set up, manage and release logical transports and links. In particular, hcitool provide access to the device MAC address and RSSI of a connected device, this is a fundamental information.

4.1.2 Ubertooth One

Ubertooth One is the hardware platform of Project Ubertooth. Ubertooth is a platform for Bluetooth experimentation. It is able to sniff BLE, discover undiscoverable classic Bluetooth devices, and perform sniffing of classic Bluetooth devices. Ubertooth is a USB dongle with an RF frontend, CC2400 radio chip, and LPC microcontroller. The CC2400 has a reconfigurable narrowband radio transceiver that can monitor a single BTLE channel at any given moment. The CC2400 converts RF into a bitstream, which is then processed entirely on the LPC. When a BTLE device transmits a packet on a particular channel it generates a small amount of RF energy. At the lowest level this modulated RF is what we aim to sniff. Ubertooth converts this RF energy into something we can work with: bits. Ubertooth transmits and receives at the frequency of 2.4 GHz, transmitting power and receiving sensitivity are comparable to a Class 1 Bluetooth device.

Ubertooth One provides a lot of interesting sniffing tools, usable typing preformatted line codes. We tested many tools and finally used the main that best matches our purpose, that are:

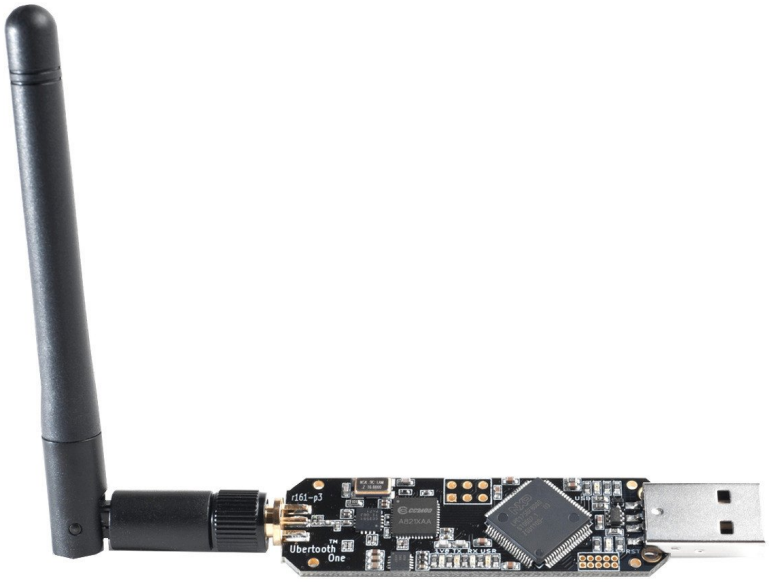

Figure 4.1: Ubertooth One and its radio antenna

- **ubertooth-btle**: sniffing connections is the most robust feature supported by ubertooth-btle. It has two primary modes of operation: *follow mode*, *no-follow mode* and *promiscuous mode*.
  - The follow mode can be used typing the pre-formatted line code *"ubertooth-btle -f"*. Ubertooth will listen on one of three advertising channels waiting for a BLE connection to be established. When a connection is established, Ubertooth will hop along the data channels, passively capturing the data sent between the central and peripheral. After the connection terminates, Ubertooth will return to the advertising channel and wait for another connection.
  - No-follow mode is similar to follow mode, we use it typing *"ubertooth-btle -n"*. In this case Ubertooth only logs advertising packets and will not follow connections as they are established.
  - Promiscuous mode is a mode for sniffing connections after they have already been established. the pre-formatted line code is *"ubertooth-btle -p"*. This mode can be used to sniff long-lived connections.

  When sniffing, Ubertooth can only operate in either follow mode or promis-

cuous mode, but not both at the same time.

- **ubertooth-rx**: It is the primary interface into Classic Bluetooth (BT) functionality provided by Ubertooth. It has two main modes of operation: *piconet following mode* and *survey mode*. In either mode, ubertooth-rx is able to discover undiscoverable devices. As we said in the previous chapter, Classic Bluetooth devices iterate each others in piconets. Classic Bluetooth piconets are defined by the Lower Address Part (LAP) and Upper Address Part (UAP) of the master device. These are elements of the master device's Bluetooth Address (BD ADDR) that *ubertooth-rx* tool try to find.

  - In piconet following mode, the tool will follow the first piconet it fully identifies. Piconet following is the main mode entered when no arguments are passed to the command or a LAP and optionally a UAP are provided. If no arguments are passed, the tool will attempt to calculate the UAP for any observed LAPs. If a LAP is passed, the UAP will be calculated for that specific LAP. The tool will recover LAP values from the air and attempt to calculate the UAP from those. Once a LAP and UAP have been recovered, the tool will attempt to recover the clock value, and if that succeeds it will follow that piconet.
  - Survey mode instead will record all LAPs and attempt to calculate the UAPs for any observed LAPs. In survey mode the device will attempt to identify all piconets in a given area and display them after either a timeout or manual interruption.

  Example: Consider the following BD ADDR: 22:44:98:88:AA:BB. The lower address part (LAP) is the lower 24 bits, so 88:AA:BB. The upper address part is the next 8 bits, so 98. The 22:44 is called the Non-significant Address Part (NAP).

  To convert LAP+UAP pairs back into Bluetooth addresses, the tool do the reverse of the above. For example, if the tool recovers a LAP of 88:AA:BB and a UAP of 98, the associated Bluetooth address is "??:??:98:88:AA:BB". Any value can be substituted into the "??" slots and most Bluetooth tools will still work. For example, hcitool name

00:00:98:88:AA:BB will establish a connection to the device and return its name.

4.2 Code

The software is written in Python (version 2.7) through PyCharm, an integrated development environment (IDE) used in computer programming because it makes writing the code much easier thanks to the large amount of libraries available on it.
To better explain the python program we can distinguish two phases:

- **Sniffing Packets Process**: explaining the various aspects of the sniffing process and how we organised it.
- **Data Acquisition Process**: an explanation of the way we acquired the data coming from the sniffing process and how we organised those data.

4.2.1 Sniffing Process

As we said in the previous chapters, we are interested in capturing Bluetooth and Bluetooth Low Energy packets transmitted from users' devices in the surrounding environment. In this section we explain how we capture packets and which type of them we are interested to. First of all, we plug in the Ubertooth One USB dongle to the port of our Laptop, we will see that three LEDs will light up: 1V8, RST and USB. In this case Ubertooth is working properly.
The entire Sniffing process was organized as follows:

- T1;
- T2;
- T3;
- T = T1 + T2 + T3;

respectively, T1 is the Sniffing time dedicated for BLE device packets sniffing tool using the "ubertooth-btle -n", we have chosen T1 equal to 30 seconds; T2 the time dedicated for the Non-discoverable device packets using "ubertooth-rx" function, setting T2 equal to 30 seconds; T3 is a variable amount of time

needed to complete the inquiry process to discover the Discoverable devices using the Bluez hcitool tool. Finally, we define T as the sum of the above Sniffing times.

Capturing BLE device packets (T1)

Choosing the right ubertooth function requires to understand how BLE devices interact each other first. As for devices working according to the Bluetooth Low Energy standard, they periodically transmit advertising packets to establish a connection with other devices or simply to broadcast information. Such advertisement packets are transmitted in-the-clear over 3 of the 40 available Bluetooth frequency channels and contain, among other fields, the advertiser's address. Listening periodically on the three advertising channels with a BLE receiver is therefore enough to capture such packets. We discussed about the function *ubertooth-btle* in chapter 3. Studying entirely the interaction between the BLE devices, we have chosen to use the "*ubertooth-btle -n*" function instead the "*ubertooth-btle -f*" because we do not need to capture data packets because the information carried by the advertising packets provides all we need to recognize the presence of the devices. Examples of capture of BLE Connectable and Non-Connectable are shown in figures 4.2 and 4.3 respectively.

```
systime=1552578670 freq=2402 addr=8e89bed6 delta_t=41.965 ms rssi=-87
40 14 c5 b5 81 76 b1 60 02 01 1a 0a ef 4c 00 10 05 03 1c d7 29 45 54 04 c5
Advertising / AA 8e89bed6 (valid)/ 20 bytes
    Channel Index: 37
    Type:  ADV_IND
    AdvA:  60:b1:76:81:b5:c5 (random)
    AdvData: 02 01 1a 0a ef 4c 00 10 05 03 1c d7 29 45
        Type 01 (Flags)
           00011010
               LE General Discoverable Mode
               Simultaneous LE and BR/EDR to Same Device Capable (Controller)
               Simultaneous LE and BR/EDR to Same Device Capable (Host)

        Type ef
            4c 00 10 05 03 1c d7 29 45

    Data:  c5 b5 81 76 b1 60 02 01 1a 0a ef 4c 00 10 05 03 1c d7 29 45
    CRC:   54 04 c5
```

Figure 4.2: Example of BLE Connectable packet sniffed by "ubertooth-btle -n"

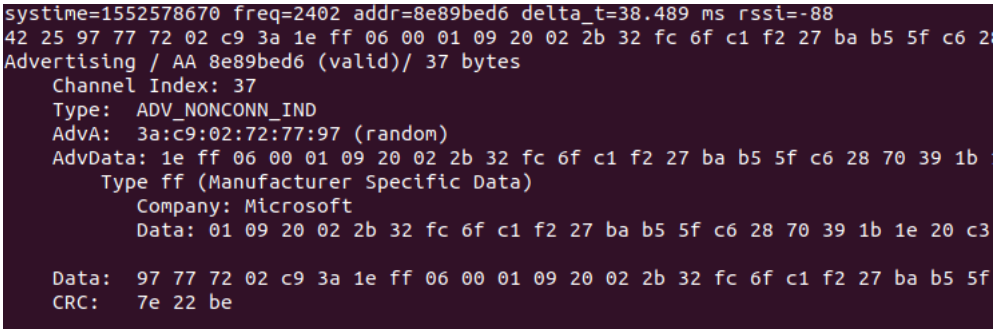

Figure 4.3: Example of BLE Non-Connectable packet sniffed by "ubertooth-btle -n"

Capturing Non-Discoverable BT Device packets (T2)

There are some Classic Bluetooth devices that are configured as "Non-Discoverable Mode", and the common Bluetooth Discovery Tools are not be able to discover this kind of devices. To solve this problem we used again Ubertooth, choosing the "*ubertooth-rx*" function. According to the Ubertooth procedure, the Bluetooth MAC address will be built finding the LAP and UAP of Non-Discoverable devices. Ubertooth is not always able to recover the UAP, if it occurs the address is not complete. Nevertheless we can easily count the number of different LAPs to recognise the presence of the surrounding Non-Discoverable devices, even if ubertooth does not recover the UAP. Below we explain how it is possible to count them anyway using an real example.

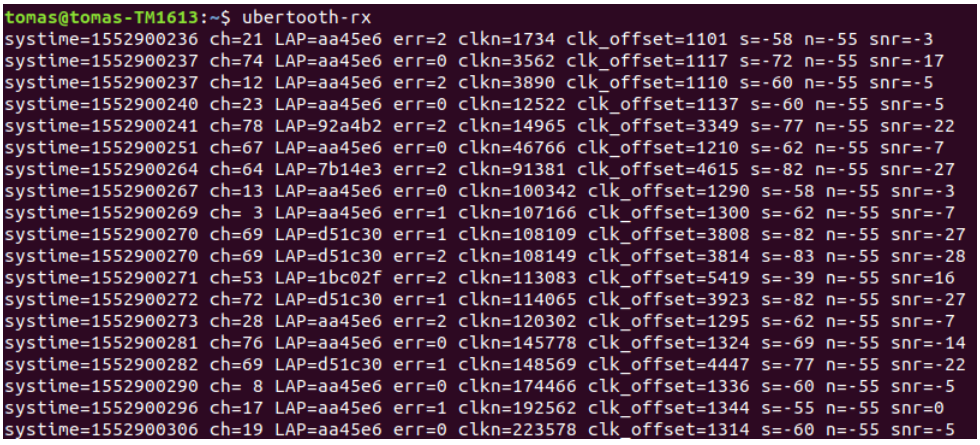

Figure 4.4: Example of the output of the tool "ubertooth-rx"

A real example of Bluetooth MAC addresses are:

- F4:E3:FB:85:53:1D
- F4:E3:FB:A5:66:D8

We know that the first three bytes are useful to identify the Vendor of the device, the remaining part (LAP) is useful to identify that particular device. In the first address we can see that UAP is 'FB' and LAP is '85:53:1D'. In the second one the UAP is the same of the previous address and LAP is different, 'A5:66:D8'. What we can surely say is that they are two distinct devices because of the different LAPs, and they belong to the same manufacturer. Therefore, if Ubertooth is not able to recover the UAP byte we simply do not know the Manufacturer, but we can distinguish two distinct devices anyway, knowing the LAP only. Thus, we can recognise devices with different LAP as different Non-Discoverable devices and count them anyway. In figure 4.4 we can see a common output of ubertooth-rx tool.

Capturing Discoverable BT Device packets (T3)

The Discoverable devices are the most easiest to find. The inquiry process regulates the discovery of available devices. In details, a master device transmits inquiry packets on different frequencies and a discoverable device receiving one of such inquiries must reply with a particular Inquiry Response packet, carrying the unit's parameter including its MAC address. Inquiry Response packets may be easily captured. It is therefore enough to mimic such an inquiry process and use a Bluetooth transmitter to broadcast inquiry packets in order to retrieve information on the available surrounding Bluetooth Classic devices. We used a python library called "bluepy" that performs the inquiry process using Bluez hcitool and gets from the responding Bluetooth devices the information we need to count, the Bluetooth MAC address and its RSSI value. In figure 4.5 is shown an example of the output of a single inquiry process.

4.2.2 Data Acquisition

The Data Acquisition process consists of collecting all the useful information from the capture of the different packets mentioned above. We explain individually the various steps of data acquisition for each category, respectively

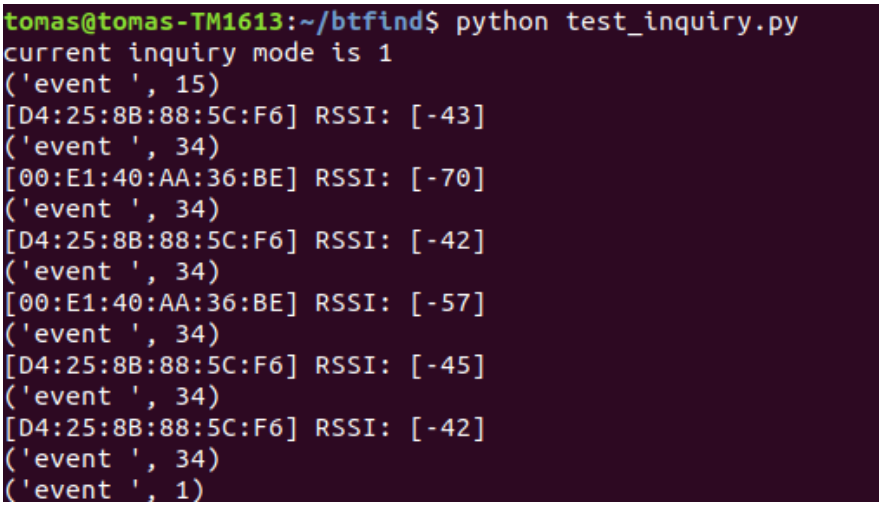

Figure 4.5: Example of bluepy code output.

BLE, Non-Discoverable and Discoverable packets coming from the output of the sniffing process.

BLE devices Data Acquisition

As we have seen, everytime Ubertooth capture a BLE packet, it provides a series of information fields such as RSSI, Type, Channel, AdvA, Data, CRC, etc., it is clearly shown in figure 4.2. In this stage we have chosen to consider only a part of these fields, the most useful to count devices, that are:

- RSSI: this information is very important for our purpose because it allows us to have an idea on how far the device is from our receiver, and then to understand whether it should be considered in our count or not. It is counted if the value does not exceed the threshold -100dBm, otherwise we do not take it into account because the device could be too far away from the receiver and probably be outside our range of interest.

- Type: the Type field tells us which type of Connection Mode the device is set to. In particular we distinguish two main types of Connection Modes, ADV_IND (CONNECTABLE) and ADV_NONCONN_IND (NON-CONNECTABLE). These types are the most common. Sniffing many packets we have noticed that only a very small percentage of devices use different Types.

- AdvA: the AdvA is the device address, 48 bits useful for identifying the device that sent the packet. As we have seen in the theoretical chapter, a Device Address can be "Public" or "Random". A device is uniquely identified if its address always remains the same, and this only happens if the address is "Public". In the other case the address will be random, so it will certainly be a different address with respect to the real one.

We consider two different Status of the address, "Random" and "Non-Random". In the case of public address a Non-Random device address will be considered and we have no problem to identify and count it, differently concerning the random ones. The problem is that a random device address could change continuously with a variable frequency depending on the factory manufacturer, so it could be extremely difficult to recognise if packets with different addresses come from the same device or not. As we can see in the figure 4.2, Ubertooth provides the status of the device address between the brackets on the same field of the address itself, thus we can get a direct mapping between the address and its status. To solve the aforementioned problem of recognising devices with random address, we have considered the following algorithm to divide the packets:

1. the type they belong:
    - ADV_IND (Connectable);
    - ADV_NONCONN_IND (Non-Connectable);
2. the Status of the address:
    - Non-Random Address;
    - Random Address;
3. its RSSI:
    - from -100dBm to -30dBm;

Certainly it is not possible to completely solve the problem of address randomness, but we proposed the following solution. We merge the informations defining a series of features that will be given at the input of a predicting model, here we list them:

- Connectable Non-Random: we consider Connectable Non-Random a device that has ADV_IND as AD Type and maintains its address over time. This type of device sends many packets with the same address and thus we easily identify them counting those packets throughout a packet sniffing period, then we count that address as a single device.
- Connectable Random: Connectable Random are all the devices that have ADV_IND as AD Type and its address appears only once throughout the packet sniffing period. This type of device can send hundreds of packets changing address everytime, so we decided to divide them from Non-Random ones because the predicting model must give a different weight to reduce the error made by the fact that we could count hundreds device instead of one.
- Non-Connectable Non-Random: Non-Connectable Non-Random are the devices that have ADV_NONCONN_IND in the AD Type field and have Non-Random device address.
- Non-Connectable Random: Non-Connectable Random are the Non-Connectable devices that have Random device address.

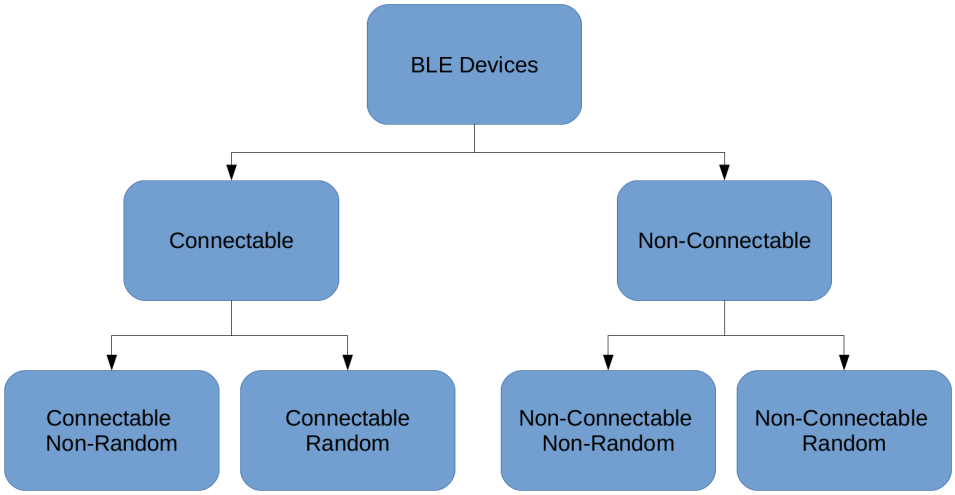

Figure 4.6: In this figure we represent the division above mentioned.

In figure 4.6 is shown the division. In this way we can easily count the number of surrounding device addresses within a certain RSSI threshould, for each of

the four feature above mentioned. In the next chapter we will explain how the way we count them.

Non-Discoverable devices Data Acquisition

As explained above, the sniffing process of the Non-discoverable device packets made by the Ubertooth function "ubertooth-rx" provides us the following useful informations. What we do in this stage is to save the two main fields of each packet:

- LAP+UAP (Address);
- RSSI;

In this way we have all the Non-Discoverable device address and its corresponding RSSI value. We can easily have the direct mapping between the number of Non-Discoverable device addresses within a RSSI threshould as well as the previous cases.

Discoverable devices Data Acquisition

In the case of Discoverable devices, the data acquisition is the most easier because we do not need any processing of the output of the inquiry process Bluez hcitool, the tool already provides us exactly the direct mapping between the MAC address and RSSI. We only count the number of Discoverable devices present within the RSSI threshould.

Chapter 5

Experiments and Results

5.1 Experiments

In this section we describe the experiments setup. A description of the different environments where the experiments took place is provided, highlighting their main characteristics. Then, we discuss the way we build the sets of the ML features to give as input of the estimation models. Finally, we give a brief review of the estimation models and how we optimised them.

5.1.1 Environments

In order to evaluate the performance of the proposed system, we carried out measurements in different environments. These environments are used for working, studying and leisure activities and are characterised by different degrees of occupants mobility. All measurements are taken using the same procedure for every environment, which is:

1. Place the laptop, with the Ubertooth One sniffer plugged in, in the center of the space to uniformly acquire packets distributed throughout the space;

2. Run the developed python script to launch the sniffing process. The process takes a time T to make a single observation, as mentioned in the previous chapter;

3. Count the people present in the surrounding environment and insert the count in the python script to collect the ground truth;

4. Repeat the last two phases (2. and 3.) as many times as the number of the observations we are interested in taking.

In particular, we divide the considered environments in indoor, outdoor and hybrid (i.e. an overlapping indoor and outdoor area) as follows:

- **Indoor**:
  - **Indoor Space 1 (IS1)**: the first indoor experiment is conducted in a university laboratory, the Advanced Network Technologies LABoratory (ANTLab). The environment has an area of about 80 sqm and is characterized by about 30 fixed seats and a daily number of occupants average of about 16, with a minimum of 4 to a maximum of 26 people present. Since the laboratory focus is on wireless communication technologies, with daily activities on wireless device testing and experimentation, a great amount of wireless devices are present. The measurement campaign lasted for about 8 hours distributed in two days, and a total of 318 ground truth observation were collected.
  - **Indoor Space 2 (IS2)**: the second indoor experiment is run in a secondary Laboratory where usually students study and work for their thesis. The room measures 21 sqm, has 13 fixed seats and is occupied by an average of 7 people, with a minimum of 2 and a maximum of 13 people. The measurement campaign lasted for about 4 hours distributed in two days, with a total of almost 200 observations. Due to the fact that the environment has many adjacent rooms, our expectation was a great amount of background noise due to the missing count of the people present in the adjacent rooms.
  - **Indoor Space 3 (IS3)**: the third indoor experiment is made in a room of the Central Library of Politecnico di Milano. The room is about 70 sqm with 46 fixed seats and an occupancy average of 27 people, with a minimum of 3 people present and full environment state. In this Library room we took 200 observations distributed in almost 5 hours in two days. The environment is quite static, the

filling and emptying of the space is gradual and varies slowly. This environment is mostly frequented by students and a large percentage of them used laptops, smartphones and headphones.

- **Outdoor**:
  - **Outdoor Space 1 (OS1)**: this experiment is executed in an outdoor recreational area where people have lunch or take a coffee break. The space is about 80 sqm and has 54 fixed seats. The average number of people present in this space is 21, with a minimum of 1 person and a maximum of 52 people present. Even in this environment were taken 200 observation with the same procedure. In this case, the environment was very dynamic, due to the fact that many students have lunch or take a coffee break in group, and when they finish they leave together, so we observe a very variable dataset.
  - **Outdoor Space 2 (OS2-m)**: the experiment OS2-m is conducted in a public transport vehicle in Milan, to be precise on the tram line 33. This vehicle is the iconic old school Milan tram series 1500 that can accommodate 29 seats and up to 100 standing places. The environment is occupied by an average of 13 people, with a minimum of 2 and a maximum of 38 people. The 225 observations were taken by sitting in the center of the moving vehicle and counting all those traveling inside. Our expectations were that this environment could be noisy due to the fact that the transport vehicle moves to the side of the road where there are pedestrians and travelers driving cars and motorcycle in the road immediately adjacent to the tram line.
- **Hybrid**:
  - **Hybrid Space (HS)**: the last experiment is run in a recreational space used even for studying, we consider it an hybrid environment because it looks like a cross between an outdoor and an indoor environment. The space is outdoor but is totally enclosed by a removable awning. It is very similar to the OS1 in terms of size and seating capacity, with the difference that this one is less dynamic. The measurement campaign lasted for about 4 hours in two days, and a total of 200 ground truth observation were collected. The average number

of people present was 19, with a minimum of 7 to a maximum of 35 people.

The details of the previously described scenarios are summarised in the following tables 5.1 and 5.2.
The environments have different characteristics from one another. We can distinguish four:

- **Dispersion**: the indoor environments are clearly less dispersive in terms of crowding of occupants. In the outdoor environments people move without following a precise path often also at the boundary of the space while in the indoor ones people have a precise location.

- **Average occupancy**: the environments differ each other for the average number of occupants in the space, as we can see in the table 5.1.

- **Dynamicity**: all the space differ also respect to how rapidly the occupants presence changes in time. In the indoor spaces people are usually standing, while in the indoor ones people are sitting.

- **Type of people**: we can distinguish different type of people for each space, IS1, IS2 and IS3 are spaces are frequented by engineer students, that more likely carried many Bluetooth wireless devices with respect to a common user present in the other spaces. In particular, IS1 and IS2 are used for wireless studies and experiments, adding a great noise to the environment.

Testbed	**Min #People**	**Max #People**	**AVG #People**
IS1	4	26	15.89
IS2	2	13	6.64
IS3	3	46	26.99
OS1	1	52	21.24
OS2-m	2	38	13.42
HS	7	35	19.44

Table 5.1: A description on the minimum and maximum number of people and its average

Testbed	Duration time (h:m)	# dev. addr.	# BLE addr.	# BT Non-Disc.	# BT Disc.	RSSI Min\Max (dBm)
IS1	8:06	183553	178429	1947	3177	-99/-31
IS2	3:52	46862	45860	686	316	-99/-30
IS3	4:52	110524	106467	1188	2869	-99/-30
OS1	4:57	256286	251980	2731	1575	-99/-30
OS2-m	4:54	241707	236903	3411	1393	-99/-30
HS	4:23	85207	83380	769	1058	-99/-35

Table 5.2: Here it is shown respectively the duration in time of the Sniffing process, the number of the device addresses captured, BLE only and BT Discoverable and Non-Discoverable, the minimum and maximum of RSSI per environment.

5.1.2 Features Implementation

In this paragraph we discuss how we implemented the set of features to be provided at the input of the model. As we said in the previous chapter, the cascade of sniffing and data acquisition processes provided us a direct mapping between six different categories of devices (i.e. BLE CONN Non-Rand, BLE CONN Rand, BLE NON-CONN Non-Rand, BLE NON-CONN Rand, BT Non-Discoverable and BT Discoverable) and the corresponding RSSI values.

Figure 5.1 represents an example situation, where the centrally positioned Ubertooth One sniffer is surrounded by a group of devices, belonging to a given Bluetooth category. Figure 5.2 gives a schematic representation of the same situation. Circles at fixed RSSI correspond to a fixed distance from the sniffer, equal to the radius of the circles themselves. Considering this representation, for a given Bluetooth category *cat*, we can easily count the devices staying within a given circle, i.e. all those devices which RSSI is lower or equal than the circle's RSSI threshold, which we refer to as θ. With this we introduce ${DevAddr^{cat}}_{\theta}$ as a feature. Ranging θ from -60 dBm to -100 dBm with pace of 5 dBm, we define 6 sets (i.e. one per category) of 9 features each. Table 5.3 shows an example of a dataset built considering all the devices for a given space. The first column is the ground truth, i.e. the real number of people in the considered space. The other columns correspond to the features for the different values of θ. Later we will describe how we merged the different features sets to perform prediction.

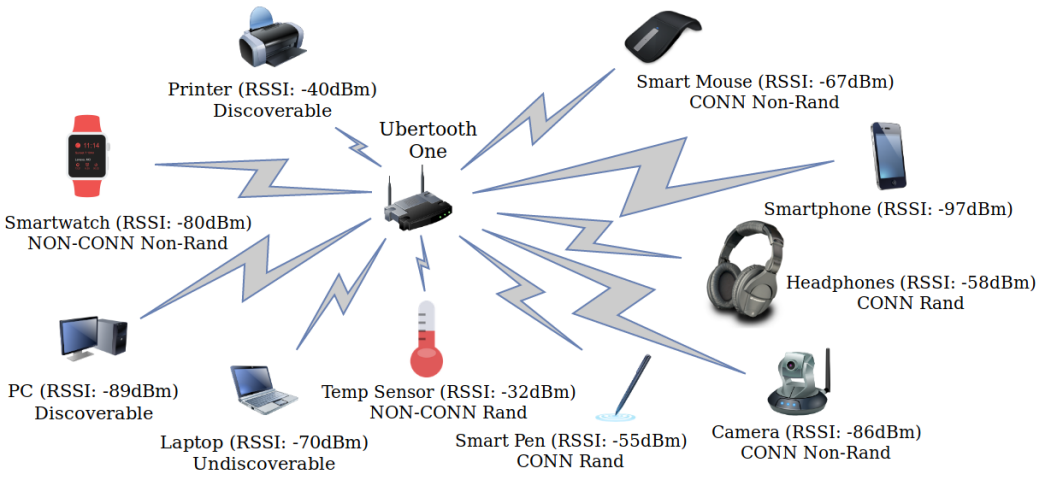

Figure 5.1: Example of Sniffing of different device packets with RSSI

5.1.3 The Estimation Models

The estimation models we used are the following ones:

- Lasso Regression
- Support Vector Regression (SVR)
- Decision Tree

Lasso Regression

In literature, in the vast majority of cases of occupancy estimation, the linear regression model is used. It is used to determine the extent to which there is a linear relationship between a dependent variable and one or more independent variables. In particular, the purpose of linear regression is to "predict" the value of the dependent variable (*regressand*) based upon the values of one or more independent variables (*predictors*). It matches perfectly to our purpose. The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line, and in our case, is the difference between the predicted number of people present and the ground truth occupancy information.

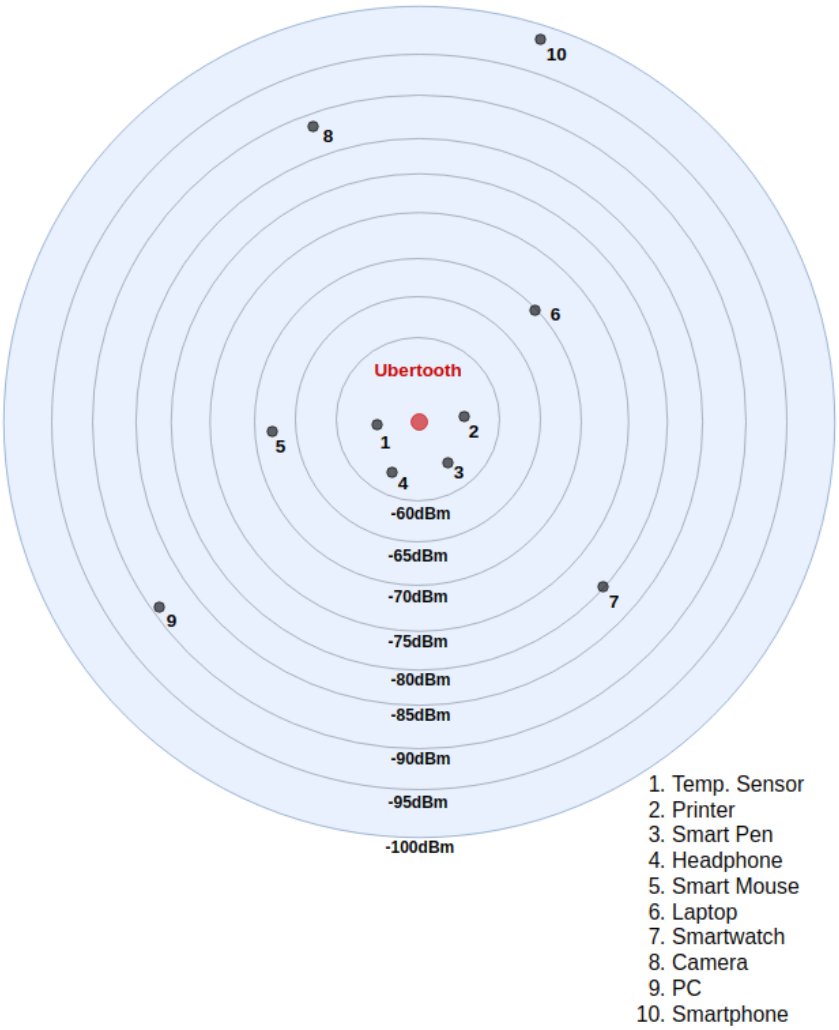

Figure 5.2: Concentric Circles and Bluetooth Devices

Having more features may seem like a perfect way for improving the accuracy of our trained model (reducing the loss), because the model that will be trained will be more flexible and will take into account more parameters. On the other hand, we need to be extremely careful about overfitting the data. As we know, every dataset has noisy samples. The inaccuracies can lead to a low-quality model if not trained carefully. The model might end up memorizing the noise instead of learning the trend of the data. If not filtered and explored up front, some features can be more destructive than helpful, repeat information that already expressed by other features and add high noise to the dataset.

# people	BLE CONN Non-Rand (-100dBm)	BLE CONN Non-Rand (-95dBm)	...	BLE CONN Non-Rand (-60dBm)	BLE CONN Rand (-100dBm)	BLE CONN Rand (-95dBm)	...	BT Disc (-100dBm)	...	BT Disc (-60dBm)
23	67	63	...	0	350	342	...	14	...	3
23	61	60	...	1	379	366	...	14	...	2
23	73	69	...	1	423	412	...	16	...	1
20	68	68	...	1	317	302	...	15	...	1
16	62	60	...	2	364	345	...	14	...	3
17	63	59	...	0	284	245	...	14	...	2
18	68	64	...	1	307	304	...	12	...	2
18	72	69	...	0	428	412	...	14	...	2
18	70	67	...	0	350	344	...	12	...	1
19	65	65	...	0	265	254	...	13	...	0
20	59	58	...	0	352	322	...	13	...	3
21	59	54	...	0	301	293	...	10	...	2
22	57	56	...	1	321	311	...	13	...	3
22	67	62	...	1	402	397	...	13	...	1
19	69	62	...	1	356	323	...	10	...	1
19	70	65	...	3	355	343	...	15	...	1
19	63	62	...	1	401	387	...	13	...	2
19	64	61	...	1	349	333	...	12	...	2
19	55	53	...	1	290	288	...	11	...	2
19	55	54	...	1	360	346	...	11	...	2

Table 5.3: Example Complete sets of features, we can see all the features of all the categories per θ

Because overfit is an extremely common issue in many machine learning problems, there are different approaches to solving it. The main concept behind avoiding overfit is simplifying the models as much as possible. Simple models do not usually overfit. On the other hand, we need to pay attention the to gentle trade-off between overfitting and underfitting a model. One of the most common mechanisms for avoiding overfit is regularization. Regularized machine learning model, is a model that its loss function contains another element that should be minimized as well.
Regularized Linear Regression Models, have a loss function that includes two elements:

$$L = \sum_{i=1}^{T} (\widehat{y_i} - y_i)^2 + \lambda \sum_{i=1}^{T} \beta_i^2 \tag{5.1}$$

where the first term is the sum of distances between each prediction and its ground truth and the second one is the regularization term.

There is a gentle trade-off between fitting the model, but not overfitting it. This approach is called Ridge regression. Ridge regression is a regularized linear regression model where the λ parameter is a scalar that help to minimize the loss function for high values of the coefficient beta. The λ parameter should be learned as well, using a method called cross validation that will be discussed later. An important fact we need to notice about Ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero, it will not get rid of irrelevant features but rather minimize their impact on the trained model. Lasso Regression is another extension built on regularized linear regression, the only difference from Ridge regression is that Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the β coefficients but actually setting them to zero if they are not relevant. Therefore, you might end up with fewer features included in the model than you started with, which is a huge advantage for us. This difference has a huge impact on the trade-off we have discussed before. For this reason we have chosen the Lasso Regression Model implemented in python by scikit-learn.

Lasso allows us to set many parameters, the most important is the λ parameter. As we said, the λ parameter must be learned using a method called cross validation. We divided each dataset of each environment as follows:

1. The dataset is divided according to k-fold cross-validation with k=5, this means that we divide the entire dataset in two part, Training set (80%) and Test set (20%) five times. In this way we have five pairs of Training set and Test set.

2. Per each of the these five pairs of sets we divide once more the Training set (80%) in another internal Training_in set (80% of the Training set) and Validation set (20% of the Training set) using another k-fold cross-validation (k=5) five times.

The cross-validation process was made as follows:

1. First, we fit the Lasso model with the Training_in set using a vector of λ values from 0.05 to 5 with step 0.01, and we predict on the Validation set. We calculate the corresponding RMSE per each λ value, retrieving the best, the one that provide the minimum RMSE. We iterate this process

as many times as the number of the folds, thus five times, getting five Best λ values. Finally we calculate the average of the five Best values. In this way we avoid the overfitting due to the Best λ value choice.

2. Second, we fit the Lasso model with the Training set using the Best average λ value found in the previous point. Then we predict on the Test set and calculate the corresponding RMSE. We iterate again the second process five times, once per each fold, getting five RMSE values and we take the average.

The same procedure was executed per each environment dataset. Here below some examples on the Lasso Regression.
In the graphs below, we can see examples of Lasso regression using respectively: BLE sets of features only in figure 5.3, BT DISC set only in figure 5.4 and BT NON-DISC set only in figure 5.5. Some point are marked with a bigger marker. The size of the marker will grow up as much as the number of observations dropped at that point of the graph.

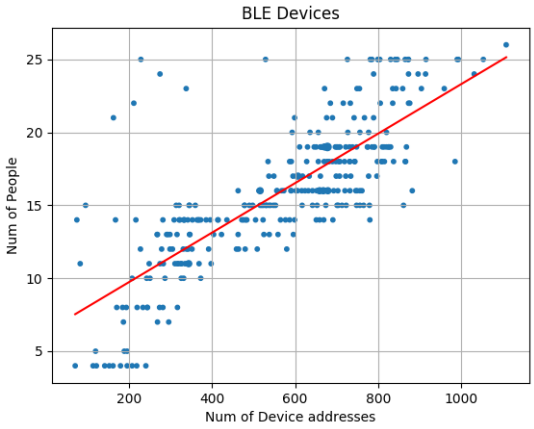

Figure 5.3: IS1 Lasso Regression result using BLE only

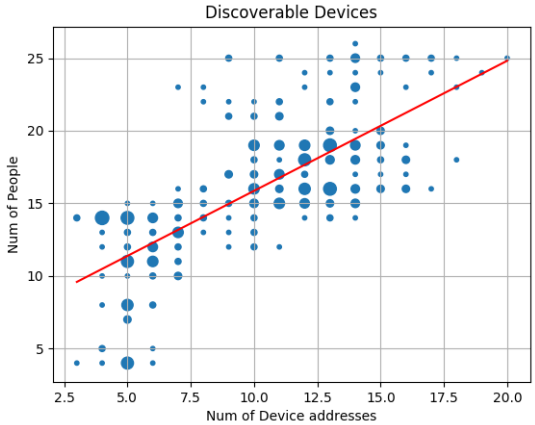

Figure 5.4: IS1 Lasso Regression result using BT Discoverable only

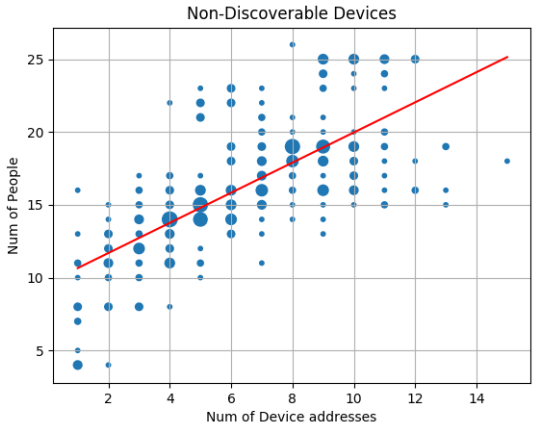

Figure 5.5: IS1 Lasso Regression result using BT Non-Discoverable only

Support Vector Regression (SVR)

Support vector machines (SVMs) are a set of supervised learning methods. The basic idea of this regression model is to find a linear function f(x) that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this. It is possible that no such function f(x) exists to satisfy

these constraints for all points. To deal with otherwise infeasible constraints, introduce slack variables ξ_i and ξ_i' for each point. This approach is similar to the "soft margin" concept in SVM classification, because the slack variables allow regression errors to exist up to the value of ξ_i and ξ_i', yet still satisfy the required conditions. Including slack variables leads to the objective function, also known as the primal formula

$$L = \frac{1}{2}\beta'\beta + C\sum_{i=1}^{T}(\xi_i + \xi_i') \tag{5.2}$$

The constant C is the box constraint, a positive numeric value that controls the penalty imposed on observations that lie outside the ε margin and helps to prevent overfitting (regularization). This value determines the trade-off between the flatness of f(x) and the amount up to which deviations larger than ε are tolerated. There are different implementations of Support Vector Regression, we have chosen the faster version of SVR, the one with the linear kernel, implemented by scikit-learn in python. The regularization, in this case, was done by finding the optimal value of the C parameter through the aforementioned cross-validation process. Thus:

1. First, we fit the linear kernel SVR model with the Training_in set using various values of C to understand first which is the range of the best values. Trying many times we understood that the best value is a number between 0.001 and 0.01, thus we fitted the model using a vector of C values from 0.001 to 0.01 with step 0.001, and we predict on the Validation set. We calculate the corresponding RMSE per each value of C and we take the best, the one that provide the minimum RMSE. We iterate this process as many times as the number of the folds, thus five times, getting five Best C value. Finally we calculate the average of the five Best values.

2. Second, we fit the SVR model with the Training set using the Best average C value found in the previous point. Then we predict on the Test set and calculate the corresponding RMSE. We iterate again the second process five times, once per each fold, getting five RMSE values and we take the average.

As the case of Lasso Regression we avoided to overfit the choice of C parameter by cross-validation.

Decision Tree Model

Decision Tree is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The decision Tree builds regression or classification models in the form of a tree structure, greedily from top to bottom. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. Each split is selected to maximize information gain (IG), the difference between the Error before the split and the Error after the split. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches, each representing values for attribute tested and a Leaf node represents a decision tree on the numerical target. Finally, the topmost decision node in a tree which corresponds to the best predictor is called Root node.

The goal of the decision tree is to find a function f(x) such that minimize the error at the training set:

$$min \sum_{i=1}^{T} (f(x_i) - y_i)^2 \tag{5.3}$$

Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit. Setting the maximum depth of the tree is so necessary to avoid this problem.

For our purpose, the scikit-learn DecisionTreeRegressor model was adopted to perform the regression of Decision tree. It provides many parameters to set in order to optimize at best. For simplicity, we have chosen to leave all the default parameters unchanged except the *maxDepth* parameter. Initially we tried to fit the model starting from the *maxDepth* parameter value of 3, as reported in literature, and continuing to increase it we found that the best value fluctuated between 3 and 20. Therefore we have defined, as in the previous cases, a vector of *maxDepth* values from 2 to 20 with step 2. The regularization was done following the aforementioned procedure, which we again summarize:

1. We fit the *DecisionTreeRegressor* model with the Training_in set using the

vector of *maxDepth* we discussed above, and we predict on the Validation set. We calculate the corresponding RMSE per each value and we take the best, the one that provide the minimum RMSE. We iterate this process as many times as the number of the folds, thus five times, getting five Best *maxDepth* values. Then we calculate the average of the five Best values.

2. Then we fit the model with the Training set using the Best average *maxDepth* value found in the previous point. Then we predict on the Test set and calculate the corresponding RMSE. We iterate again the second process five times, once per each fold, getting five RMSE values and we take the average.

5.2 Results

In this paragraph we first discuss about the two indicators used, Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), to evaluate the performance of the three models seen for each dataset of the different environments described above, spending a few words on how we split the datasets with which we fit the models. Furthermore we discuss the performances obtained by the various combinations of set of features in terms of RMSE and MAPE.

Evaluation Metrics

Here we give a briefly description of two commonly used evaluation metrics of machine learning algorithms performances, which are the Root Mean Square Error (RMSE) and the Mean Absolute Percentage Error (MAPE):

- **RMSE**: Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are. In other words, it tells you how concentrated the data is around the "line of best fit", and thus is a measure of how spread out these residuals are. RMSE is always non-negative value. In our case the lower is the value the better the model works. However, comparisons across different types of data would be invalid because

the measure is dependent on the scale of the numbers used. Therefore RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, but not between datasets, because it is scale-dependent.

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(\widehat{y_i} - y_i)^2} \tag{5.4}$$

- **MAPE**: Mean Absolute Percentage Error is a measure of difference between two continuous variables. As the name suggests, the MAPE is an average of the absolute errors between the prediction and the true value divided by the true value. It expresses accuracy as a percentage. Compared to RMSE, MAPE is not scale-dependent and is often useful for forecast evaluation, therefore can be used to make comparisons between datasets using different scales.

$$MAPE = \frac{100\%}{T}\sum_{i=1}^{T}\left|\frac{\widehat{y_i} - y_i}{y_i}\right| \tag{5.5}$$

Performances

It is interesting to see the performances in terms of RMSE and MAPE of the models we explained above. We have distinguished several features set combinations to understand which of them carries more useful information and which leads the model to make more mistakes. Below we provide a description of how we merged the Features sets and then the results of the chosen combinations of sets per space for each classifier used in the tables 5.4, 5.5, 5.6, 5.7, 5.8, 5.9:

- ALL: we used the entire features sets to fit the models (BLE CONN Non-Rand, BLE CONN Rand, BLE NONCONN Non-Rand, BLE NONCONN Rand, Non-Discoverable and Discoverable);
- BLE: in this case we considered all the features sets that belong to BLE only (BLE CONN Non-Rand, BLE CONN Rand, BLE NONCONN Non-

Rand, BLE NONCONN Rand). Within this group, we distinguish the following case studies:

  - CONN: we used the Connectable ones only (BLE CONN Non-Rand, BLE CONN Rand),
  - NON-CONN: the Non-connectable only (BLE NONCONN Non-Rand, BLE NONCONN Rand),
  - RAND: we used BLE Connectable and Non-connectable with randomized addresses only (BLE CONN Rand, BLE NONCONN Rand),
  - NON-RAND: BLE Connectable and Non-connectable with non-randomized addresses only (BLE CONN Non-Rand, BLE NONCONN Non-Rand),
  - BLE-ALL: all BLE features;

- BT-CLASSIC: in this case, we considered the set of features belonging to Bluetooth Classic only (Discoverable and Non-Discoverable). We distinguish three sub-cases:

  - DISC: the Discoverable ones only,
  - NON-DISC: the Non-Discoverable ones only,
  - BT-ALL: all BT features.

RMSE	**ALL**	**BLE**					**BT**		
Testbed		**BLE-ALL**	**BLE CONN**	**BLE NON-CONN**	**BLE RAND**	**BLE NON-RAND**	**BT DISC**	**BT NON-DISC**	**BT-ALL**
IS1	**2.002**	2.324	2.587	2.802	2.992	2.395	3.052	3.202	2.774
IS2	**1.635**	1.651	2.705	1.679	1.983	1.776	2.863	2.283	2.207
IS3	**3.673**	3.743	5.738	4.486	5.890	3.874	8.683	9.209	7.537
OS1	8.546	**8.325**	9.079	12.737	11.168	8.752	13.127	13.798	13.715
OS2-m	**6.631**	6.695	7.028	7.840	7.510	6.741	7.944	7.328	7.364
HS	**4.493**	4.883	4.974	7.170	6.295	5.052	5.332	7.101	5.265

Table 5.4: Lasso Regression RMSE results

MAPE	ALL	BLE					BT		
Testbed		**BLE-ALL**	**BLE CONN**	**BLE NON-CONN**	**BLE RAND**	**BLE NON-RAND**	**BT DISC**	**BT NON-DISC**	**BT-ALL**
IS1	**13.50%**	15.64%	16.93%	19.47%	18.29%	16.41%	21.45%	20.33%	18.52%
IS2	**26.76%**	26.98%	46.13%	27.17%	34.49%	28.78%	51.76%	38.96%	38.90%
IS3	**15.76%**	16.05%	30.83%	18.43%	32.16%	15.93%	50.20%	52.85%	38.88%
OS1	73.33%	**72.53%**	77.50%	112.52%	99.15%	73.67%	113.89%	124.21	118.67%
OS2-m	**67.11%**	68.42%	73.10%	82.31%	79.98%	67.85%	87.37%	75.74%	75.56%
HS	**24.15%**	25.73%	27.35%	40.83%	34.95%	28.09%	29.47%	40.42%	28.26%

Table 5.5: Lasso Regression MAPE results

RMSE	ALL	BLE					BT		
Testbed		**BLE-ALL**	**BLE CONN**	**BLE NON-CONN**	**BLE RAND**	**BLE NON-RAND**	**BT DISC**	**BT NON-DISC**	**BT-ALL**
IS1	**2.119**	2.300	2.709	2.835	2.976	2.324	3.133	3.253	2.808
IS2	**1.692**	1.765	2.761	1.796	1.994	2.140	2.875	2.364	2.401
IS3	**3.794**	3.821	5.471	4.667	5.907	4.288	8.560	8.890	7.655
OS1	8.084	**8.369**	8.880	11.540	9.732	9.968	14.098	15.080	14.320
OS2-m	**7.032**	7.230	7.628	8.126	7.988	7.249	8.189	7.560	7.587
HS	**4.318**	5.046	4.966	7.258	6.276	5.115	5.257	7.155	5.279

Table 5.6: Support Vector Regression RMSE results

As we can see, we proposed two tables per classifier, the reason is that we need both to compare the results of the various environment and classifier used. In particular, as mentioned above, we cannot compare results coming from different datasets in terms of RMSE because of the scale-dependency. Therefore we use RMSE to evaluate the performances of each classifier fixing the environment (i.e. compare results fixing the row) and MAPE to evaluate the performances comparing the different datasets (i.e. comparing results in columns).

Let's consider the results of Lasso Regression in terms of RMSE: As we can see from the table 5.4, the best performances are given by using the entire sets of features for almost all the environment datasets except by the OS1 space, the Library, where it is worth to not consider the information coming from BT packets. It means that for the vast majority of cases both BT and BLE information gives a contribution to the model, but BLE dominates the comparison, indeed if we only consider BLE packets the performances are almost the same. In terms of MAPE, the results provided by table 5.5 say that the performances

MAPE	ALL	BLE					BT		
Testbed		**BLE-ALL**	**BLE CONN**	**BLE NON-CONN**	**BLE RAND**	**BLE NON-RAND**	**BT DISC**	**BT NON-DISC**	**BT-ALL**
IS1	**14.45%**	15.69%	17.24%	19.37%	17.63%	16.26%	20.92%	20.13%	18.45%
IS2	**25.85%**	26.31%	46.01%	28.13%	32.72%	34.19%	47.00%	41.70%	42.19%
IS3	**15.66%**	16.73%	30.73%	18.78%	28.72%	17.94%	49.35%	52.24%	39.77%
OS1	61.67%	**59.46%**	61.21%	75.58%	65.40%	85.97%	98.98%	108.21%	97.62%
OS2-m	**54.57%**	55.49%	65.51%	77.00%	68.02%	54.58%	82.37%	67.05%	66.96%
HS	**23.24%**	28.04%	29.33%	39.53%	34.42%	29.21%	29.09%	40.85%	29.35%

Table 5.7: Support Vector Regresssion MAPE results

RMSE	ALL	BLE					BT		
Testbed		**BLE-ALL**	**BLE CONN**	**BLE NON-CONN**	**BLE RAND**	**BLE NON-RAND**	**BT DISC**	**BT NON-DISC**	**BT-ALL**
IS1	**1.988**	2.438	2.943	2.679	3.548	2.052	3.419	3.164	2.570
IS2	1.905	1.811	2.971	**1.737**	2.373	1.804	2.982	2.333	2.466
IS3	4.967	4.636	5.552	5.985	6.104	**4.308**	8.398	8.859	8.397
OS1	10.365	**10.234**	11.707	16.124	13.474	10.765	14.741	15.090	13.382
OS2-m	**7.404**	7.596	7.679	7.737	8.203	8.079	8.295	7.964	7.783
HS	4.347	**4.156**	5.311	7.492	6.398	4.466	5.406	7.264	5.795

Table 5.8: Decision Tree RMSE results

depends on the characteristics of the spaces mentioned in the environments description, thus the performances depend on:

- Dispersion: Lasso regression provides better results where the surrounding environment forces people to stay in a limited space. Indeed, indoor spaces provide better performances with respect to the outdoors and hybrid.

- Average occupancy: the model works better if the average number of occupants is higher, this is confirmed by the fact that the performances of the IS2 are a bit lower with respect to the others with higher average number of occupants, although it is an Indoor space. The table 5.1 shows the differences of environments in terms of average number of people.

- Dynamicity: there is clearly a relationship between the results of the dynamic spaces and the non-dynamic ones. Performances of the Library and Laboratories, are higher respect to the ones provided by the spaces

MAPE	ALL	BLE					BT		
Testbed		BLE-ALL	BLE CONN	BLE NON-CONN	BLE RAND	BLE NON-RAND	BT DISC	BT NON-DISC	BT-ALL
IS1	**10.45%**	13.16%	14.89%	16.25%	18.02%	11.17%	22.05%	18.34%	15.74%
IS2	26.22%	26.54%	36.69%	**23.64%**	37.65%	25.46%	52.76%	35.50%	39.25%
IS3	16.77%	17.89%	20.98%	21.55%	21.63%	**15.02%**	33.15%	37.96%	32.23%
OS1	77.11%	**76.52%**	87.79%	123.57%	110.31%	77.77%	112.44%	120.29%	102.92%
OS2-m	**72.14%**	74.87%	74.19%	79.00%	80.17%	77.71%	88.62%	75.59%	80.23%
HS	17.66%	**15.87%**	27.14%	41.06%	33.55%	18.39%	26.37%	39.14%	29.99%

Table 5.9: Decision Tree MAPE results

where people pass quickly through or nearby without stopping, as in the cases of the HS (Bar) and OS2-m (Tram).

- Type of people: the results give us the idea that performances are as better as the number of wireless devices present per occupant. In the case of IS1 the space was narrow and fairly crowded, with a minimum of 4 people and a maximum of 26, furthermore people were PhD students in a Network technology Laboratory that surely use wireless technology more than the average usage of normal people. This surely impacts on our measurements because we have more probability to acquire a lot of bluetooth packets. We can say the same for the IS3 (Library), where many occupants were students and had headphones, smartphones, smartwatches and laptops.

In the case of Support Vector Regression in tables 5.6 and 5.7, we are not surprised about almost the same performances of the Lasso regression because they are both linear classifier and work similarly each other. The fact that they provide the same results validates the correctness of the performance achieved. Therefore, also in this case we can say that the most useful information is carried out by BLE packets, but the system works better if we take into account BT packets also. The performances are strongly dependent on the spaces characteristics aforementioned.

Looking at the results given by Tables 5.8 and 5.9, the decision tree seems to work better using different sets of features compared to the two Linear classifiers. In particular, the MAPE results highlight that decision tree works a bit better of the two linear classifiers in Indoor environments IS1, IS2, IS3 and in HS, worse in OS1 and OS2-m. We would say that the reason is the dynamicity of the last two and the non-linearity that linear regression does not consider.

Chapter 6

Conclusion and Future works

The thesis proposes a system for estimating the occupancy of spaces based on the capture of Bluetooth/BLE inquiry and advertising packets. The system is built using *Ubertooth One* sniffer and *BlueZ Linux* Bluetooth stack able to process received packets and create different sets of features capturing different device behaviours. Such sets of features are merged in different combinations and later fed into three supervised Machine Learning models, Regularized Linear Regression (Lasso), Support Vector Regression and Decision Tree for performing precise occupancy estimation. We tested our prototype in several uncontrolled scenarios: three indoor, two outdoor, an hybrid one and we validate them through extensive experiments. The results obtained confirm the validity of the proposed solution, which is able to perform accurate occupancy estimation in both indoor and outdoor environments, also characterised by different levels of crowding (from few people to more than fifty) and dynamicity. In particular the performances highlight that *Decision Tree* algorithm works a slightly better in indoor environments but presents lower accuracy in the cases of outdoor and more dynamic spaces with respect to Lasso Regression and Support Vector Regression.

Compared to other works in the literature, the proposed solution comes at a very low cost for what concerns both the hardware design and the overhead for supervising the learning algorithms, it is accurate as much as the related works that use more than one IEEE standard and environmental data sensing, reaching the accuracy of 85%. Our work is an indirect approach and so is non-intrusive.

Furthermore, with the proliferation of the Internet of Things, the Smart City project will soon be a reality, and thanks to this more and more people will carry one or even more wireless device with them. This is a key component for our system, because it will work better and better, reaching even higher levels of accuracy. We strongly believe on the potential of this indirect approach occupancy estimation and future works can be conducted merging other wireless technologies including WiFi, Wireless Personal Access Networks, such as ZigBee and 6LoWPAN, adding more useful information to be processed by supervised learning models.
Future works can be based on the use of different predicting occupancy models such as Artificial Neural Networks (ANN) that is more complicated but can reach better performances. Another way to improve this system can be a better analysis on the surrounding environment features, evaluating the geometry of the spaces and their obstacles, that can impact negatively to the accuracy of RSSI values because of multipath and scattering effects and consequently affect performances.

Bibliography

[1] Alessandro E.C. Redondi and Matteo Cesana. Building up knowledge through passive wifi probes. *Computer Communications*, 117:1 – 12, 2018.

[2] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 1–6, New York, NY, USA, 2010. ACM.

[3] Varick L. Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E. Cerpa, Michael D. Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pages 19–24, New York, NY, USA, 2009. ACM.

[4] Zheng Yang, Nan Li, Burcin Becerik-Gerber, and Michael Orosz. A systematic approach to occupancy modeling in ambient sensor-rich buildings. *SIMULATION*, 90(8):960–977, 2014.

[5] Alessandra De Paola, Marco Ortolani, Giuseppe Lo Re, Giuseppe Anastasi, and Sajal K. Das. Intelligent management systems for energy efficiency in buildings: A survey. *ACM Comput. Surv.*, 47(1):13:1–13:38, June 2014.

[6] Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings*, 56:244 – 257, 2013.

[7] Y. Benezeth, H. Laurent, B. Emile, and C. Rosenberger. Towards a sensor for detecting human presence and characterizing activity. *Energy and Buildings*, 43(2):305 – 314, 2011.

[8] I.J. Amin, A.J. Taylor, F. Junejo, A. Al-Habaibeh, and R.M. Parkin. Automated people-counting by using low-resolution infrared and visual cameras. *Measurement*, 41(6):589 – 599, 2008.

[9] F. Wahl, M. Milenkovic, and O. Amft. A distributed pir-based approach for estimating people count in office environments. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 640–647, Dec 2012.

[10] F. Wahl, M. Milenkovic, and O. Amft. A green autonomous self-sustaining sensor node for counting people in office environments. In *2012 5th European DSP Education and Research Conference (EDERC)*, pages 203–207, Sep. 2012.

[11] S. Munir, R. S. Arora, C. Hesling, J. Li, J. Francis, C. Shelton, C. Martin, A. Rowe, and M. Berges. Real-time fine grained occupancy estimation using depth sensors on arm embedded platforms. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 295–306, April 2017.

[12] Fulin Wang, Qingqing Feng, Zheliang Chen, Qianchuan Zhao, Zhijing Cheng, Jianhong Zou, Yufeng Zhang, Jinbo Mai, Yun Li, and Hayden Reeve. Predictive control of indoor environment using occupant number detected by video data and co2 concentration. *Energy and Buildings*, 145:155 – 162, 2017.

[13] Mattias Gruber, Anders TrÃ¼schel, and Jan-Olof DalenbÃck. Co2 sensors for occupancy estimations: Potential in building automation applications. *Energy and Buildings*, 84:548 – 556, 2014.

[14] Davide Cali', Peter Matthes, Kristian Huchtemann, Rita Streblow, and Dirk Muller. Co2 based occupancy detection algorithm: Experimental analysis and validation for office and residential buildings. *Building and Environment*, 86:39 – 49, 2015.

[15] Luis M. Candanedo and VÃ©ronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28 – 39, 2016.

[16] L. Zimmermann, R. Weigel, and G. Fischer. Fusion of nonintrusive environmental sensors for occupancy detection in smart homes. *IEEE Internet of Things Journal*, 5(4):2343–2352, Aug 2018.

[17] Zhenghua Chen, Mustafa K. Masood, and Yeng Chai Soh. A fusion framework for occupancy estimation in office buildings based on environmental sensor data. *Energy and Buildings*, 133:790 – 798, 2016.

[18] Seung Ho Ryu and Hyeun Jun Moon. Development of an occupancy prediction model using indoor environmental data based on machine learning techniques. *Building and Environment*, 107:1 – 9, 2016.

[19] Sunil Mamidi, Yu-Han Chang, and Rajiv Maheswaran. Improving building energy efficiency with a network of sensing, learning and prediction agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 45–52, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[20] S. Depatla, A. Muralidharan, and Y. Mostofi. Occupancy estimation using only wifi power measurements. *IEEE Journal on Selected Areas in Communications*, 33(7):1381–1393, July 2015.

[21] Chenren Xu, Bernhard Firner, Robert S. Moore, Yanyong Zhang, Wade Trappe, Richard Howard, Feixiong Zhang, and Ning An. Scpl: Indoor device-free multi-subject counting and localization using radio signal strength. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 79–90, New York, NY, USA, 2013. ACM.

[22] L. Mikkelsen, R. Buchakchiev, T. Madsen, and H. P. Schwefel. Public transport occupancy estimation using wlan probing. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 302–308, Sep. 2016.

[23] A. Filippoupolitis, W. Oliff, and G. Loukas. Bluetooth low energy based occupancy detection for emergency management. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 In-*

ternational Symposium on Cyberspace and Security (IUCC-CSS), pages 31–38, Dec 2016.

[24] A. Corna, L. Fontana, A. A. Nacci, and D. Sciuto. Occupancy detection via ibeacon on android devices for smart building management. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 629–632, San Jose, CA, USA, 2015. EDA Consortium.

[25] Giorgio Conte, Massimo De Marchi, Alessandro A. Nacci, Vincenzo Rana, and Donatella Sciuto. Bluesentinel: A first approach using ibeacon for an energy efficient occupancy detection system. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pages 11–19, New York, NY, USA, 2014. ACM.

[26] Matteo Cesana Edoardo Longo, Alessandro E. C. Redondi. Accurate occupancy estimation with wifi and bluetooth/ble packet capture. Date '18, Milano, MI, Italy, 2018.

[27] M. Cunche, , and R. Boreli. I know who you will meet this evening! linking wireless devices using wi-fi probe requests. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, June 2012.

[28] A. J. Ruiz-Ruiz, H. Blunck, T. S. Prentow, A. Stisen, and M. B. KjÃrgaard. Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 130–138, March 2014.

[29] Marco V. Barbera, Alessandro Epasto, Alessandro Mei, Vasile C. Perta, and Julinda Stefa. Signals from the crowd: Uncovering social relationships through smartphone probes. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 265–276, New York, NY, USA, 2013. ACM.

[30] Dieter Oosterlinck, Dries F. Benoit, Philippe Baecke, and Nico Van de Weghe. Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits. *Applied Geography*, 78:55 – 65, 2017.