POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master of Science in Electronic Engineering

Master Thesis

Design and Implementation of an UWB-based Indoor Localization System for Large Scale Scenarios

Supervisor:
Prof. Roberto Garello
Eng. Orlando Tovar Ordonez

Candidate:
Jorge Luis Delos Campos Nieto

September 2018

Acknowledgements

First of all, I would like to thank to professor Roberto Garello for giving me the opportunity of developing my thesis in a company such as the Istituto Superiore Mario Boella. I feel the imperious need of thanking Eng. Orlando Tovar, Eng. Michele Ligios, Eng. Luigi Coriasco, Eng. Francesco Sottile, and everybody in the Pert area of ISMB, as well as the interns that had collaborated in one way or another to my thesis: Constantin Stamatiadis, Davide Maria Bruno and Luca Favario.

I also have to express my infinite gratitude to my family and friends in Peru. They have been and they always will be with me to everywhere I go.
I want to thank also to all my friends that have been along my side during my stay in Italy.

Abstract

Given the need to cover localization requirements, that systems such as the global position system (GPS) were not able to, there were conceived the Indoor Position Systems (IPS), that are able to provide accurate location estimates for indoors.
Differently to GPS, that uses satellites to get localization measurements, IPS are based on a Wireless Sensor Network (WSN), been able in this way to set up a Real Time Locating System (RTLS). RTLS provide repeatedly the real time position of a moving node (or tags), with respect to fix nodes (or anchors).
This localization estimation of a real time position is obtained in two phases: ranging and positioning. The first one estimates the distance between mobile and anchor nodes, while the second one uses these distance measurements to localize the mobile node in the system.

The implementation of an IPS for this thesis is based on the Ultra- wideband (UWB) technology, more specifically the standard IEEE802.15.4a.

This thesis has been done in collaboration with the Istituto Superiore Mario Boella (ISMB), starting from an existing and perfectly working localization system, and will continue using its technology for the localization task, but with the aim of enlarge the operation field of the network, by means of creating more unit areas, or clusters, that will be part of one same system. This will be done by means of creating a central location engine, or Super Gateway (SGW), that has the task to manage the data information and parameters of each of the unit areas, as well as taking decisions on how to assign parameters to the mobile units to be localized inside the system, with the help of location engines, or Gateways (GW) that will be placed in each of the area units, The communication between SGW and GWs is based in the middleware MQTT. The GWs transmit, through a Bluetooth Low Energy (BLE) connection, all the information needed by the tags to be able to carry on with the localization task.
Moreover, it is provided an overview of RTLS regarding radio-based ranging techniques and the most common localization algorithms.

According with the standard ISO/IEC/IEEE 42010:2011(E) the implementation of the system is based on a proposed solution that is formed by a functional and a deployment architecture, as well as an information flow that give the guideline to test the system under stress condition such as a tag switching from one cluster to another in a successful way, a tag getting parameters from a wrong cluster, or even a tag that is enable to do ranging after receiving the cluster parameters. All of this to verify the correct operation of the system.

Keywords: IPS, WSN, RTLS, CLE, BLE, UWB.

Acronyms

ACL Asynchronous Connection-Less
AOA Angle of Arrival
ATT Attribute Protocol
BLE Bluetooth Low Energy
CLE Central Location Engine
CRLB Cramér-Rao Lower Bound
DAA Detect and Avoid
DS-SS Direct-Sequence Spread-Spectrum
ED Energy Detection
EKF Extended Kalman Filter
FW Firmware
GAP Generic Access Profile
GATT Generic Attribute Profile
GPS Global Positioning System
GUI Graphic User Interface
GW Gateway
IPS Indoor Position Systems
ISMB Istituto Superiore Mario Boella
LDC Low Duty Cycle
LL Link Layer
LLS Linear Least Square
LOS Line-Of Sight
LQI Link Quality Indication
LS Least-Squares
MDS Multidimensional Scaling
MITM Man-In-The-Middle
MPC Multipath Component
MTU Maximum Transmission UNIT
NFC Near Field Communication
NLOS No Line-Of-Sight
PDP Power Delay Profile
PERT Pervasive Technologies
PHR PHY Header
PPDU PHY Protocol Data Unit
PPM Pulse Position Modulation
PRF Pulse Repetition Frequency
PV Position-Velocity
QAM Quadrature Amplitude Modulation
RF Radio Frequency
RSS Received Signal Strength
RSSI Received Signal Strength Indication
RTLS Real Time Locating System
SCC Simple Crosscorrelator
SFD Start-Of-Frame Delimiter
SGW Super Gateway
SHR Synchronization Header
SNR Signal To-Nois Ratio

SS Signal Strength
TDMA Time Division Multiple Access
TDOA Time Difference of Arrival
TOA Time of Arrival
UUID Universal Unique Identifier
UWB Ultrawide-Band
WLS Weighted LS
WSN Wireless Sensor Network

Summary

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Indoor position systems (IPS) were developed to cover requirements were other technologies such as the global positioning system (GPS) were not able to provide reliable data, or even data that would be useful (like in particular cases where it is needed a high precision localization esteem) due to its limitations in indoors usage. So IPS would be able to deliver localization of the devices that belong to the system with high position esteem accuracy.

While GPS uses satellites for its localization measurements, an IPS is based on a Wireless Sensor Network (WSN) that is an ideal technology for the deployment of a Real Time Locating System (RTLS). A WSN is design to locate, in specific places of the network, sensors able to read different types of data, depending on the type of system, such as pressure, temperature, motion, etc. RTLS combine hardware and software to periodically provide the real time position of a moving capable asset using mobile nodes, called “Tags”. These tags will communicate with fix sensors with known position, estimated in coordinates, called “Anchors”.

This localization estimation of a real time position is obtained in two phases: ranging and positioning. The first one estimates the distance between mobile and anchor nodes, while the second one uses these distance measurements to localize the mobile node in the system based in coordinates.
There are several practical cases where having an IPS could be of great utility, such as animal tracking, logistics, safety security, etc.

In terms of animal tracking, if we talk about biological research, knowing how animals behave and interact over time and over very wide range, with each other, in loneliness or with some other species, and getting to know the inter-animal distances, could be meaningful.

In terms of logistics, automatic localization would increase automation of processes that has been traditionally wired, like monitoring and control of machinery. Making the manufacturing floor more dynamic. Having a network that can localize mobile equipment becomes handy when it needs to be found or get lost, and even contact security if it is about to leave the premises of where it was intended to be geographically.

The radio technology in which base the implementation of an IPS for this thesis was the Ultra-wideband (UWB) technology, more specifically the standard IEEE802.15.4a.

1.2 Thesis motivation and objectives

This thesis was born from the need of designing a system capable of using the already existing Real Time Localization system, in the Istituto Superiore Mario Boella (ISMB), in a large-scale system. Meaning that the previously mentioned system was designed to work in a single unit are (Cluster), in terms of all the parameters that conform it, such as frequency and channel, anchors coordinates, etc. Creating the need of a new plan to make possible the interaction between two or more clusters, solving problems such as the decision-making process to define the parameters to be used trying to avoid interferences between adjacent clusters. And figuring how to establish a central location engine (CLE), where all assets will be stored ad managed.

To do so it was proposed an area distribution plan for the clusters deployment. As well as the idea of having a location engine, also called Gateway (GW), on top of each cluster, that then will have to communicate with the CLE, also called Super Gateway (SGW), that will manage the interaction of the tags within the cluster and also in the case of changing its position from one cluster to another.
For this mean it had to be chosen a technology to stablish a communication channel between the Super Gateway and the Gateways, as well as one between the Gateways and the tags.

For the link between Super Gateway and Gateway a TCP/IP connection will be stablished, on top of it the middleware MQTT would be the one in charge of handling the data flow. MQTT is a Client Server publish/subscribe messaging transport protocol. It is very light weight given that it has a minimal packet overhead. There is an agent called "Broker" that is located in between publisher and subscriber and it will be the one that will filter the messages to be distributed to the right place, so there is no need that the publisher and subscriber have information about the other one. MQTT employs a space decoupling technique in which publisher and subscriber will only need to know hostname/ip and port of the broker to be able to publish messages or to subscribe to them.

For the link between Gateway and tags it is presented in the deployment architecture a Bluetooth Low Energy (BLE) connection. BLE maintains a very similar communication range to the classic Bluetooth technology, but it provides reduced power consumption as well as a lower cost.
For this case the already existing tag firmware designed by ISMB will be modify to add the BLE services and characteristics required to get the cluster parameters from the GW.

The previously mentioned tag firmware has the characteristic of being static, meaning that the tags after being programmed with such firmware are not able in any way to change their own parameters, that are the ones that allow it to range with the anchors of a specific cluster.
This static behavior of the tags will be changed for a dynamic one. In which the firmware will be subject of constant parameters changes, that will arrive from the GW, to make possible the continue working of the tag even when changing its location to a new cluster.

1.2 Thesis overview

In chapter 2 it will be described an overview of Indoor Localization in which it is going to be detailed the properties of the ranging techniques, such as real-time localization algorithms and the two main types of localization approaches.

In Chapter 3 it will be an overview on Standards and Protocols. Which is going to be divided in three important subjects: UWB standard for localization, seeing in detail its characteristics and features to demonstrate why IEEE802.15.4a is the most suitable standard for indoor positioning. Bluetooth Low Energy, describing the standard most important layers and protocols. And MQTT as a messaging transport protocol for the exchange of information between GW and SGW.

In Chapter 4 it will be proposed a solution as a design of the Large-Scale Localization system, by describing the architecture that will define the functionality and the deployment of the project. And it will be also defined the use cases in which the system will be tested.

In Chapter 5 it will be explained the implementation of such proposed design. It will address the firmware modifications on the tags, as well as the creation of the algorithms for the GW and SGW. And it will show the result of the testing and validation of the use cases suggested in the previous chapter.

Finally, the conclusions and remarks of the thesis will be presented.

Chapter 2

Overview of Indoor Real Time Locating Systems

2.1 Introduction

Real-time locating systems (RTLS) are a combination of hardware and software used to determine and provide real-time information about the location of assets and resources, such as objects, people, animals or anything equipped with devices designed to work with the system.
The most pervasive example of an RTLS is GPS. It is a well-known satellite-based technology that is used, assuming the availability of the appropriate hardware and software, to find a GPS-equipped device geographically worldwide. This is a truly remarkable system that has irrevocably changed the face of commercial and personal navigation.

However, some problems appear when talking about indoors. The technology that GPS provides has non-irrelevant limitations to work indoors, so all the tracking and location functionality that GPS provides get lost. So, the need to find some technology suitable for indoor location capability got increased.
Basically, an indoor locating system is based in some localization hardware that is formed by a small number of sensors, called reference nodes, with fix known coordinates (either via GPS or from a system administrator during start-up) and the unknown-location nodes or tags (facing the power supply problem that they have limited energy to work with). Which are associated to a location engine and an end-user application.
RTLS are based in many types of communication technologies, such us radio frequency (RF) or optical systems (infrared), or acoustic (ultrasound).

Future RTLS systems are envisioned to be based on low power electronic tags used to track and/or monitor assets, people or anything of value with very high accuracy and mobility.
Nevertheless, we can identify two main drawbacks when talking about RTLS tags: The first one being wideband based tags provide accuracy but with limited range, while relatively narrowband based tags do not provide the accuracy some applications require. The second one is that the tags can be large and power hungry. The goal of RTLS design is to eliminate this complicated trade-off between accuracy, power consumption and range [1].

2.2 Ranging Techniques

To figure out the location of a tagged object, in 2D or 3D space, it is important to determine how far away it is from various well-known points. With this learning and with some generally basic calculations it is possible to figure the position of the gadget.

There are various techniques for implementing RTLS utilizing wireless schemes however they are divided basically into three types of scheme:

1) Radio Signal Strength based: normally referred as Received Signal Strength Indication or RSSI schemes.

2) Angle of Arrival (AOA)

3) Time of Arrival (TOA)

2.2.1 Receive Signal Strength

Taking into account a path-loss model, the separation between two nodes can be figured by estimating the energy of the received signal at one node. This distance-based system requires no less than three reference nodes to locate the 2D position of a given tag.

RSS is the voltage estimated by a receiver's received signal strength indicator (RSSI) circuit. Frequently, RSS is reported as estimated power, i.e., the squared magnitude of the signal strength. Wireless sensors talk with neighbouring sensors, so the RSS of RF signs can be estimated by every beneficiary amongst ordinary information communication without introducing extra bandwidth or energy requirements. RSS estimations are generally modest and easy to execute in hardware equipment. However, RSS estimations are very unpredictable.

2.2.1.1 Major sources of error

In free space, signal power rots relatively to d^{-2}, where d is the separation between the transmitter and receiver. In real-world channels, multipath signals and shadowing are two noteworthy sources of environment dependence in the estimated RSS. Numerous signals arrive at the receiver with various amplitudes and phases, and these signals sum valuably or damagingly as a component of the frequency, causing frequency-selective fading.

The impact of this type of fading can be reduced by utilizing a spread-spectrum technique (e.g., direct-sequence or frequency hopping) that averages the received power over an extensive variety of frequencies. Spread-spectrum receivers are a worthy arrangement since these techniques additionally reduce interference in the unlicensed bands in which wireless sensors commonly work. The estimated received power utilizing a wideband strategy is equivalent to estimating the sum of the powers of each multipath signal [3]. Assuming that frequency-selective effects become smaller, environment-dependent errors in RSS estimations are caused by shadowing, for example, the attenuation of a signal because of impediments (furniture, walls, trees, structures, etc.) that a signal must go through or diffract around on the way between the transmitter and receiver. these shadowing effects are modelled as random (as a component of the environment in which the system is set).

2.2.1.2 Statistical model

Usually, the received power in a real- world obstructed channel gets diminished by d^{-np}, where np is the path-loss exponent, commonly between two and four. The power at a distance d is modelled as:

$$\bar{\mathrm{P}}(d) = P_0 - 10n_p \log \frac{d}{d_0} \quad (1)$$

where P0 is the received power (dBm) at a short reference distance d_0. The contrast between a measured received power and its average, because of the shadowing randomness, is demonstrated as log-normal (i.e., Gaussian if it is in decibels). The log-normal model depends on a wide variety of measurement results [4] and scientific confirmation [5]. The standard deviation of the received power, σ_{dB}, is constant with distance. The received power (dBm) at sensor *i* transmitted by *j*, $P_{i,j}$ is as follows:

$$f\ P_{i,j} = p \mid \theta\ = \mathcal{N}\ p; \bar{P}(d_{i,j}), \sigma_{dB}^2 \quad (2)$$

where $\mathcal{N}$ (x; y, z) is the notation for the value at x of a Gaussian probability density function (pdf) with mean y and variance z, θ is the coordinate parameter vector $\theta = [\theta_x, \theta_y]$

$$\theta_x = [x_1, ..., x_n],\ \theta_y = [y_1, ..., y_n] \quad (3)$$

and the transmitter-receiver distance $d_{i,j}$ is:

$$d_{i,j} = \sqrt{x_i - x_j\ ^2 +\ y_i - y_j\ ^2} \quad (4)$$

The result of the log-normal model is that RSS-based range estimates have variance proportional to their actual range. This isn't an inconsistency of the prior explanation that σ_{dB} is constant with range. Constant standard deviation implies that the multiplicative components are consistent with range; this clarifies the proportionality. RSS is most profitable in high-density sensor systems.

2.2.1.3 Calibration and synchronization

In addition to the path loss, estimated RSS is likewise a component of the calibration of both the transmitter and receiver. Depending on the cost of the assembling procedure, RSSI circuits and transmit powers will vary from gadget to gadget. Transmit powers can change as batteries drain. Sensors may be intended to quantify and report their own particular calibration information to their neighbors. On the other hand, every sensor's transmitted power can be viewed as an unknown parameter to be evaluated. This implies the unknown vector θ is enlarged to include the actual transmitted power of every sensor along with its coordinates. Or, we can consider just the difference between RSS estimated at sets of receivers [6]. The RSS contrast between two sensors shows data about their relative separation from the transmitter and removes the dependency on the transmit power.

2.2.1.4 RSS for high accuracy

These schemes include estimating the signal strength of the arriving signal at the receiver. Knowing the power at which it was transmitted, the propagation characteristics of the signal and some information of the environment it is possible to compute around where the transmission began in light of how attenuated it is at the receiver.

These schemes are sufficient in specific conditions, but where high precision is required they should be helped with some other technologies (choke points, infra-red, ultrasound, etc.) to give the required level of precision [7].

2.2.2 Angle of arrival

The second ranging technique utilizes the angle of arrival to measure the orientation of the transmitter with respect to the receiver. By estimating the difference in arrival times of a signal on the components of an antenna cluster, the course of arrival can be evaluated as in Figure 1. This can be seen as the reverse of beamforming.

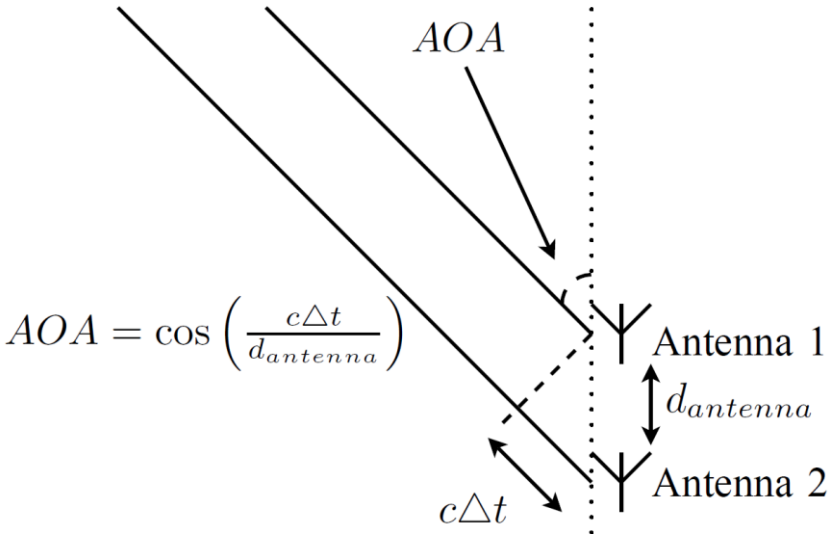

Figure 1. An angle of arrival system using two antennas. By utilizing the difference in the time of an arrival of a signal at the antenna pair (△t) along with the antenna spacing dantenna, the arrival direction can be estimated [1].

There are two ways that sensors measure Angle of Arrivel (AOA) (Figure 2) [8]. The most widely used method is to utilize a sensor cluster and utilize signal processing strategies at the sensor nodes. For this situation, every sensor node is formed by at least two individual sensors (such as, microphones for acoustic signals or antennas for RF) whose position with respect to a known node are constant.

A four-component Y array like in Figure 2(a). The AOA is assessed from the differences in arrival times for a transmitted signal at every one of the components of the array. The estimation is like time-delay estimation, however, summed up to for an array of more than two elements. At the point when the impinging signal is narrowband (its bandwidth is substantially less than its center frequency), a time delay τ have a relationship with a phase delay φ by $\varphi = 2\pi f_c \tau$ where f_c is the center frequency.

A second technique to deal with AOA estimation utilizes the RSS proportion between (at least two) directional antennas situated on the sensor [Figure 2(b)]. Two directional antennas pointed in different ways, with the end goal that their primary beams overlap, can be used to assess the AOA from the proportion of their individual RSS esteems.

Both AOA approaches require antennas arrays, which can add to the sensor gadget cost and size. However, acoustic sensor clusters may be required in gadgets for some environmental observing and security applications, in which the motivation behind the sensor network is to distinguish and find acoustic sources [9].

RF antennas clusters infer large gadget size unless the central frequencies are very high. However, accessible bandwidth and diminishing assembling costs at millimeter-wave frequencies may make them good for sensor network applications. For instance, at 60 GHz, higher attenuation because of oxygen absorption mitigates multipath and precise indoor AOA estimations have been shown [10].

2

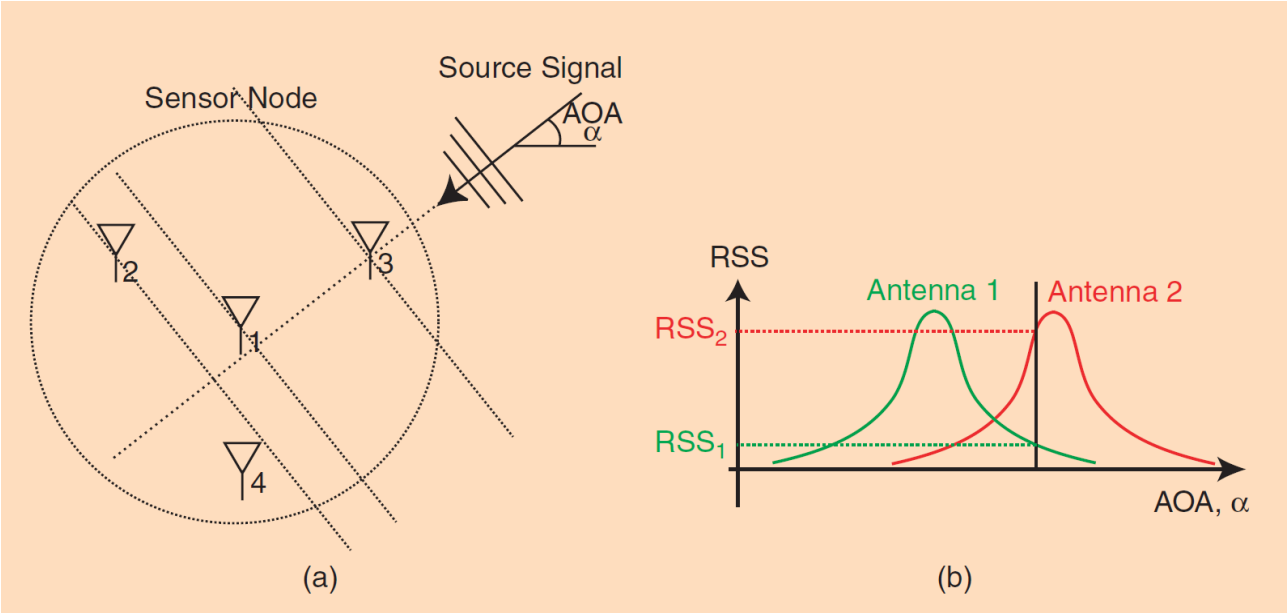

Figure 2. AOA estimation methods. (a) AOA is estimated from the TOA differences among sensor elements embedded in the node; a four-element Y-shaped array is shown. (b) AOA can also be estimated from the RSS ratio RSS1/RSS2 between directional antennas [8].

Angle of Arrival schemes do not deal especially well with multipath propagation between the transmitter and the receiver antenna array as are most appropriate to Line of Sight situations.

2.2.3 Time of arrival

Time of arrival (TOA) is the estimated time at which a signal (RF, acoustic, or so on) first gets at a receiver. The estimated TOA is the time of transmission in addition to a propagation-induced time delay. This time delay, $T_{i,j}$, between transmission at sensor i and reception at sensor j, is equivalent to the transmitter-receiver distance, $d_{i,j}$, divided by the propagation velocity, v_p. This speed for RF is roughly 106 times as quick as the speed of sound; For acoustic propagation, 1 ms means 1 ft (0.3 m), while for RF, 1 ns means 1 ft. The foundation of time-based procedures is the receiver's capacity to precisely measure the arrival time of the line-of sight (LOS) signal. This estimation is hampered both by additive noise and multipath signals [8].

2.2.3.1 Major Sources of Error: Additive Noise

Even without multipath signals, the exactness of the arrival time is constrained by additive noise. Estimation of time delay in additive noise is an already developed field [11]. Ordinarily, the TOA estimate is the time that maximizes the cross-correlation between the received signals and the known transmitted signal [8]. This estimator is known as a simple crosscorrelator (SCC). The generalized cross-correlator (GCC) inferred by Knapp and Carter [12] (the maximum likelihood estimator (MLE) for the TOA) broadens the SCC by applying prefilters to increase spectral parts of the signal that have little noise and attenuate elements with high noise. All things considered, the GCC requires information (or assessments) of the signal and noise power spectra.

For a given bandwidth and signal to-nois ratio (SNR), our time-delay measure can just accomplish a limited precision. The CRB gives a lower bound on the variance of the TOA estimate in a multipath-free channel. For a signal with bandwidth B in (hertz), when B is much lower than the center frequency, F_c (Hz), and signal and noise powers are constant for the bandwidth used

$$\text{var}(TOA) \geq \frac{1}{8\pi^2 B T_s F_c^2 SNR} \tag{5}$$

where T_s is the signal length in seconds. By designing the system to accomplish adequately high SNR, the bound anticipated by the CRB in (5) can be accomplished in multipath-free channels. Subsequently (5) gives an approach about how signal parameters like length, bandwidth, and power influence our capacity to precisely estimate the TOA. For instance, multiplying either the transmission power or the bandwidth will slice ranging variance in half. This CRB on TOA fluctuation is complementary to the bound that will be exhibited for location variance in light of the fact that it requires, as an input, the variation of the TOA estimates.

2.2.3.2 Major Sources of Error: Multipath

TOA-based range errors in multipath channels affect more the performance of the system than those caused by additive noise alone. Basically, all late-arriving multipath elements are self-interference that adequately diminish the SNR of the desired LOS signal. Instead of finding the highest peak of the cross-correlation, in the multipath channel, the receiver must locate the first arriving peak in light of the fact that there is no assurance that the LOS signal will be the strongest of the arriving signals. This should be possible by estimating the time that the cross-correlation first crosses a threshold. Then again, in template-matching, the main edge of the cross-correlation is coordinated in a least-squares (LS) sense to the main edge of the auto-correlation (the correlation of the transmitted signal with itself) to accomplish subsampling time resolutions.

Errors, in TOA estimation, are caused by two main issues:

1) *Early-arriving multipath*. Numerous multipath signals arrive not long after the LOS signal, and their contributions to the cross-correlation darken the location of the peak from the LOS signal.
2) *Attenuated LOS*. The LOS signal can be seriously attenuate in comparison with the late-arriving multipath elements, making it to be "lost in the noise" and missed totally; this prompt to big positive errors in the TOA estimate.

In dense sensor systems, in which any couple of sensors can estimate TOA, we have the advantage position of having the capacity to measure TOA between close-by neighbors. As the path length diminishes, the LOS signal power (with respect to the power in the multipath elements) increments. In this way, the seriously attenuated LOS issue is just extreme in systems with large inter-sensor separations.

While early-arriving multipath elements cause little errors, they are exceptionally hard to battle. For the most part, more extensive signal bandwidths are important to get more temporal resolution. The peak width of the autocorrelation function is inversely proportional to the signal bandwidth. A thin autocorrelation peak improves the capacity to pinpoint the arrival time of a signal and aides in isolating the LOS signal cross-correlation contribution from the early-arriving multipath signals. Wideband direct-sequence spread-spectrum (DS-SS) or Ultra-wideband (UWB) signals are well

known procedures for high-bandwidth TOA estimations. In any case, more extensive bandwidth requires higher speed signal processing, higher gadget costs, and higher energy costs. Standards proposed to the IEEE 802.15 Alternative PHY Task Group 3a quote receiver power consumption on the order of 200 mW. Furthermore, even if high-speed regularly circuitry implies higher energy consumption, the additional bandwidth can be utilized to bring down the time average power consumption. Moving data in less time implies investing more time in standby mode.

At last, take note of that time delays in the transmitter and receiver equipment and program add to the estimated TOA. While the nominal delays are ordinarily known, variance in component details and reaction times can be an extra source of TOA variance.

2.2.3.3 Statistical Model

Estimations have demonstrated that for short-range estimations, estimated time delay can be generally modeled as Gaussian

$$f(T_{i,j} = t \mid \theta) = \mathcal{N}(t; d_{i,j|}/v_p + \mu T, \sigma_T^2) \tag{6}$$

where μT and σ_T^2 are the mean and variance of the time delay error, θ is defined in (3), $d_{\mathrm{i,j}}$ is given in (4), and v_{p} is the propagation velocity. Wideband DS-SS estimations detailed in [4] support the Gaussian error model and indicated μT = 10.9 ns and σ_T = 6.1 ns. UWB estimations directed on an empty Motorola industrial facility floor demonstrated μT = 0.3 ns and σ_T = 1.9 ns. This mean mistake μT can be estimated by the localization algorithm with the goal that it can be subtracted out.

In any case, the presence of big errors can become the Gaussian model more complicated. These errors make the tails of the distribution of estimated TOA heavier than Gaussian and have been modeled utilizing a hybrid distribution. With a little likelihood, the TOA estimation results of an alternate, higher-variance distribution, as portrayed in [31] also by Gustafsson and Gunnarsson. Localization systems ought to be intended to be robust to these expansive mistakes, called NLOS errors.

For TOA estimations mode over time in a changing channel, the TOAs that have excess delays can be recognized and ignored [31]. Indeed, even in static channels, if the quantity of range estimations for a gadget is bigger than the minimum required, the redundancy can be utilized to recognize likely NLOS mistakes.

2.2.3.4 Calibration and Synchronization

In the event that wireless sensors have clocks that are precisely synchronized, at that point the time delay is dictated by subtracting the known transmit time from the estimated TOA. Sensor network clock synchronization calculations have precisions on the range of 10μs [21]. In view of the difference in propagation speed, such clock precisions are sufficient for acoustic signals [22] however not for RF signals.

For TOA in asynchronous sensor systems, a typical practice is to utilize two-way (or round-trip) TOA estimations. In this technique, one sensor transmits a signal to another sensor, which instantly answers with its own particular signal. At the first sensor, the estimated delay between its transmission and its reception of the answer is double the propagation delay in addition to an inner answer delay of the second sensor. This inner delay is either known or estimated and sent to the first sensor to be

subtracted. For the most part, each couple of sensors measures round-trip TOA independently in time. However, if the main sensor has the signal handling capacity, more the one sensor can answer in the meantime, and two-way TOAs can be measured at the same time utilizing multiuser interference cancelation.

2.3 Real-time localization algorithms

To estimate the position when working with wireless positioning systems there are two main stages. In the initial stage is utilized a ranging technique, as TOA or RSS the most widely used, to measure the distance between the tag and the anchors, at that point in the second stage, this data is utilized as a part of the location algorithms to estimate the tag's position. The most widely recognized localization algorithms utilized are Linear Least Square (LLS) and Extended Kalman Filter (EKF).

2.3.1 Linear Least Square

Linear Least Square is a localization algorithm of low computational complexity. LLS will deliver a reasonable positioning accuracy for applications that do not need high computational complexity. LLS is optimal to start a location process that will not be of high precision, but that could be used as a base to initialize a high accuracy localization algorithm, such as EKF. Doing this the computational complexity and the final location error get decreased.

Having a wireless network with N reference nodes, with the i_{th} node being located at $l_i = [x_i, y_i]^T$ for $i = 1,....,N$.

The scope is to estimate the location of the tag, meant by $l = [x, y]^T$, in view of N TOA estimations between the tag and the anchors. Z_{i} shows to the separation measured from the i_{th} TOA estimation:

$$z_i = c\tau_i = f_i(x, y) + n_i \text{ , for } i = 1,..., \text{N} \tag{7}$$

where τ_i represents the TOA measure for the i_{th} signal, c is the speed of light, n_i is the noise in the i_{th} measurement, and $fi(x, y)$ is the distance between the tag and the i_{th} anchor, given by

$$f_i(\text{x}, \text{y}) = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{8}$$

In an LLS strategy, from the estimations in (7) another estimation set is gotten.

The LLS approach begins with the following set of equations

$$z_i^2 = (x - x_i)^2 + (\text{y} - y_i)^2 \text{, for } i = 1,..., \text{N} \tag{9}$$

Each circle of uncertain region is defined by a distance measurement. Then, one of the equations in (6), say the r_{th} one, is fixed and subtracted from all of the other equations. After some manipulation, the following linear relation can be obtained [12]:

$$Al = \text{P} \tag{10}$$

Where $l = xy^T$,

$$A = 2\begin{pmatrix} x_1 - x_r & y_1 - y_r \\ \vdots & \vdots \\ x_{r-1} - x_r & y_{r-1} - y_r \\ x_{r+1} - y_r & y_{r+1} - y_r \\ \vdots & \vdots \\ x_N - x_r & y_N - y_r \end{pmatrix} \tag{11}$$

And

$$\mathrm{P} = \begin{pmatrix} z_r^2 - z_1^2 - k_r + k_1 \\ \vdots \\ z_r^2 - z_{r-1}^2 - k_r + k_{r-1} \\ z_r^2 - z_{r+1}^2 - k_r + k_{r+1} \\ \vdots \\ z_r^2 - z_N^2 - k_r + k_N \end{pmatrix} \tag{12}$$

With

$$k_i = x_i^2 + y_i^2 \text{, for } i = 1, 2, \ldots, N \tag{13}$$

r being the selected reference anchor index that is used to obtain linear relations. A is an (N − 1) × 2 matrix, and p is an (N -1) sized vector, since the r_{th} measurement is used as a reference for the other measurements.

From (10), we obtained the LS solution as

$$\hat{l} = (A^T A)^{-1} A^T \mathrm{P} \tag{14}$$

This linear LS (LLS) estimator has low computational complexity. However, it is not optimal.

2.3.2 Extended Kalman filter

The Kalman Filter algorithm, offers an elegant, effective and ideal solution for location issues when the network is linear and irregular measurements errors are gotten after a Gaussian distribution. Since these conditions are not generally met, some linearization and approximations are expected to change the KF into the Extended Kalman Filter (EKF), appropriate for non-linear systems [13].

The condition of a dynamic system can be measured recursively by the discrete EKF modeled by discrete-time state equation:

$$x_k = f(x_{k-1}) + w_k \text{,} \tag{15}$$
$$w_k \sim \mathcal{N}(0, Q_k)$$

where x_k is the state vector at time k, f is the *state transition function* which is in function of the previous state x_{k-1}.

The *process noise vector* w_k takes into account the non-linearities and perturbations on the system, modeled by a vector of random noise normally distributed with zero mean and covariance matrix $\mathbf{Q}_k$. The following measurement equation allows us to observed the system through it:

$$z_k = h(x_k) + v_k, \qquad (16)$$
$$v_k \sim \mathcal{N}(0, R_k),$$

where $\boldsymbol{z}_k$ is the *measurement vector* at time k, h is the *observation function* which estimates the expected measurements at the true state $\boldsymbol{x}_k$, and $\boldsymbol{v}_k$ is the *observation noise vector* assumed as a vector of random variables normally distributed with zero mean and covariance matrix $\mathbf{R}_k$.

Given both models, the EKF algorithm iteratively tracks the state evolution in two phases of the filter: *predict* and *update* [14].

2.3.2.1 Predict phase

The EKF predicts the *a priori* state vector estimation $\hat{\mathrm{x}}_{k|k-1}$ at time k based on the previous *a posteriori* state estimate $\hat{\mathrm{x}}_{k-1|k-1}$:

$$\hat{\mathrm{x}}_{k|k-1} = f(\hat{\mathrm{x}}_{k-1|k-1}) \qquad (17)$$

The covariance matrix $\mathrm{P}_{k|k-1}$ associated to the predicted state vector $\hat{\mathrm{x}}_{k|k-1}$ is also evaluated from the previous estimate $\mathrm{P}_{k-1|k-1}$ and process noise covariance matrix Q_k:

$$\mathrm{P}_{k|k-1} = F_k \mathrm{P}_{k-1|k-1} F_k^T + Q_k, \qquad (18)$$

Where $F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{\mathrm{x}}_{k-1|k-1}}$ is the Jacobian matrix of the state transition function f computed around the previous estimates [31].

2.3.2.2 Update phase

This phase occurs when an observation $\mathbf{z}_k$ becomes available, at time k. During this phase, the *innovation vector* $\tilde{y}_k$ is calculated as the difference between $\mathbf{z}_k$ and the expected measurement $h(\hat{\mathrm{x}}_{k|k-1})$:

$$\tilde{y}_k = z_k - h(\hat{\mathrm{x}}_{k|k-1}) \qquad (19)$$

The *covariance matrix* $\boldsymbol{S}_k$ is calculated, with the innovation vector, as the expected measurement estimation error given an *a priori* state error covariance in addition to the measurement covariance matrix $\boldsymbol{R}_k$:

$$S_k = H_k \mathrm{P}_{k|k-1} H_k^T + R_k \tag{20}$$

Where $H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{\mathrm{x}}_{k-1|k-1}}$ is the Jacobian matrix of the observation function evaluated around the *a priori* state estimate.

Finally, the algorithm calculates the last state estimate $\hat{\mathrm{x}}_{k|k}$ and corresponding covariance matrix $\mathrm{P}_{k|k}$ by correcting the earlier state estimate $\hat{\mathrm{x}}_{k|k-1}$ and $\mathrm{P}_{k|k-1}$:

$$\hat{\mathrm{x}}_{k|k} = \hat{\mathrm{x}}_{k|k-1} + K_k \tilde{y}_k \tag{21}$$

$$\mathrm{P}_{k|k} = (I_n - K_k H_k)\mathrm{P}_{k|k-1} \tag{22}$$

Where K_k is the optimal Kalman gain:

$$K_k = \mathrm{P}_{k|k-1} H_k^T S_k^{-1}. \tag{23}$$

2.4 Localization Approaches

2.4.1 Centralized Approach

In the Centralized approach all the ranging measurements between tags and anchors are sent to a central computer, through a gateway, that implements the localization algorithms. This is the most common approach adopted for localization matters. An advantage of this approach is that it has a good performance when the localization algorithms are complex and the device's processor is not powerful. Instead, a disadvantage is that a system that has long paths has a lower accuracy of the estimations.

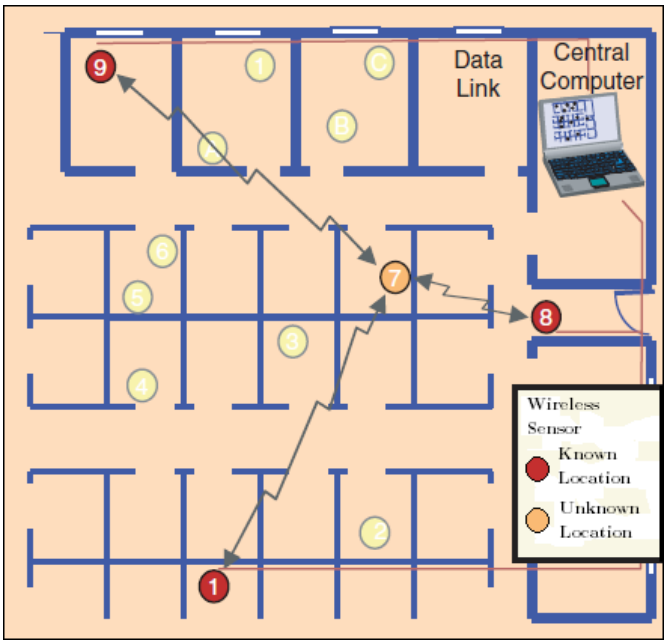

Figure 3. Localization Centralized Approach [8].

In Figure 3 is shown how in a centralized approach after the tag and anchors get a ranging measurement they will be transmitted to the central computer through a data link, where the localization algorithm is running and the position of the tag is going to be resolved.

2.4.2 Distributed Approach

In a Distributed approach the tags are the ones to compute their own position, through ranging with the anchors. This is achieved by using a localization algorithm that performance in the tag itself. For this approach it is convenient to use the Linear Least Square (LLS) algorithm, which is a low complexity one. There is also the possibility that the tags shared their estimated positions between them to get better precision. This is called a cooperative approach.

There are three main reasons to apply a distributed approach to a system:
When a high quantity of sensors needs to send their ranging measurements to a central computer a bottleneck is created in the communication channel.
There is not always possible to have a central computer that will fit the technological requirements, to be able to handle of the calculations for some applications.
As said in section 2.4.1, when there are long paths in big environments, the accuracy of the estimation is affected.

In Figure 4 is shown a Distributed approach applied to a network. As usual, the ranging between tags and anchors is performed. This data, as well as an estimated position from other tags, are sent to the central computer.

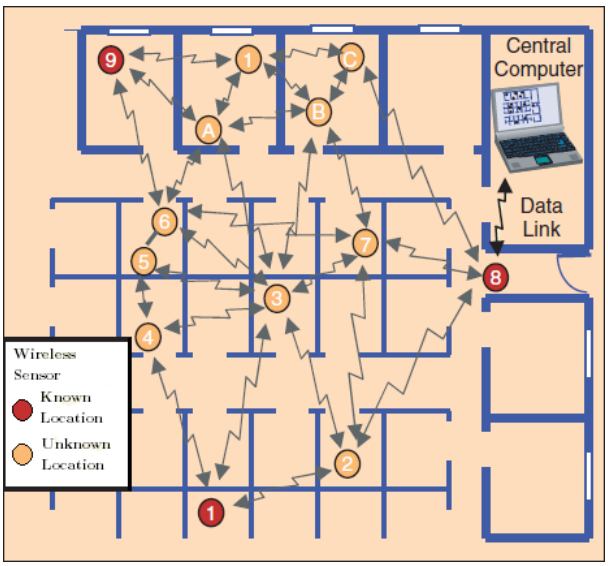

Figure 4. Localization Distributed Approach [8].

2.4.3 Comparison

These approaches, Centralized and Distributed algorithms, have both relatively high costs of communication.

When talking about the Centralized approach each sensor's estimation has to pass through many hops to get to a central processor in a large network, for this reason it is likely to consume a lot of power,

that's reflected in energy costs. So, for the Distributed approach each sensor's estimation is sent just one hop away, and this will be reflected in a saving of the communication energy costs.
To reduce the energy consumption, beyond the independent costs for each approach, it is possible to have a combination of these algorithms that mix centralized and distributed features. For instance, in a large network we could divide the area of deployment into smaller clusters where we could have a processor for each of these clusters, then each one of them could utilize a distributed algorithm to merge and optimize the local measurements [16].

2.5 Large Scale Localization System

When the network, to be covered by the RTLS, is going to be deployed in a building, or wider campus, where the range of the system is not going to be able to cover all of the area, then additional anchors are going to be needed. Now, we are going to learn how to expand the system to have many anchors for a network ramification.

There is a limitation for the air-occupancy for the working tags of the system, however in large scale we would aim to reuse the same air-time for different tags that are located far enough from each other in the system, so there would not be any interference between them.

So, two main questions arise for the design of the system:

1) *System architecture:* Where are the ranging results gathered, solved, and used?

2) *System scheduling and control:* How does the tag know what anchor units are in the vicinity and which of those anchors it should range to, and when?

The approaches to the possible solutions for each of these questions are discussed in the next section.

2.5.1 System architecture for large area deployment

The localization approach used could be part of a PC (GUI) application [32] and this can operate in two ways:

1) The application gets the range measure for all tags talking to a main Anchor, when the PC is connected to this same anchor. It also receives the estimates from all the other anchors. In this way it can solve and display the result of the localization of the tag in the system. In this way all the information is gathered, monitored and reported into a central location engine (CLE) in a RTLS system (tracking) mode.

2) When the PC is connected to a mobile tag, it sends the distance estimated, after ranging with all the anchors in its range, to the application, that afterwards, with this data, solves the position localization of the mobile tag. This is the so-called navigation mode, where the tag and PC solver are the mobile system that find its own location as it moves around the environment.

It is also possible, for navigation application, to calculate its location when the distance to the anchors is estimated, by incorporating a solver. It is not mandatory that the anchors themselves know their own position, but it would be useful if they do, so they could pass this data in the ranging response message. The solver has to know the anchor locations to solve the trilateration and locate the mobile

tag. Depending on the capabilities of the mobile device the anchor location could be distributed by any sort of system or communication technology.

For a mobile tag tracking RTLS it is normal that the CLE gets the tag ranges reports from the anchors, to then be solve. For this aim, a *backhaul* is need for transporting this information from anchors, or tags to the CLE. Many options exist for the scope, such us wired Ethernet, Wi-Fi, etc.

However, navigation asset tracking use cases are anyway supported if the solver would be in a CLE on the system or in the mobile tag unit. The RTLS solution can be transmitted by the CLE to the tag if needed for navigation, or, the information could be sent to the central monitoring station for reporting and logging as required. One of the main issues is to select the right anchors to range to for a tag that awakes as new in the system, and how this is going to be managed.

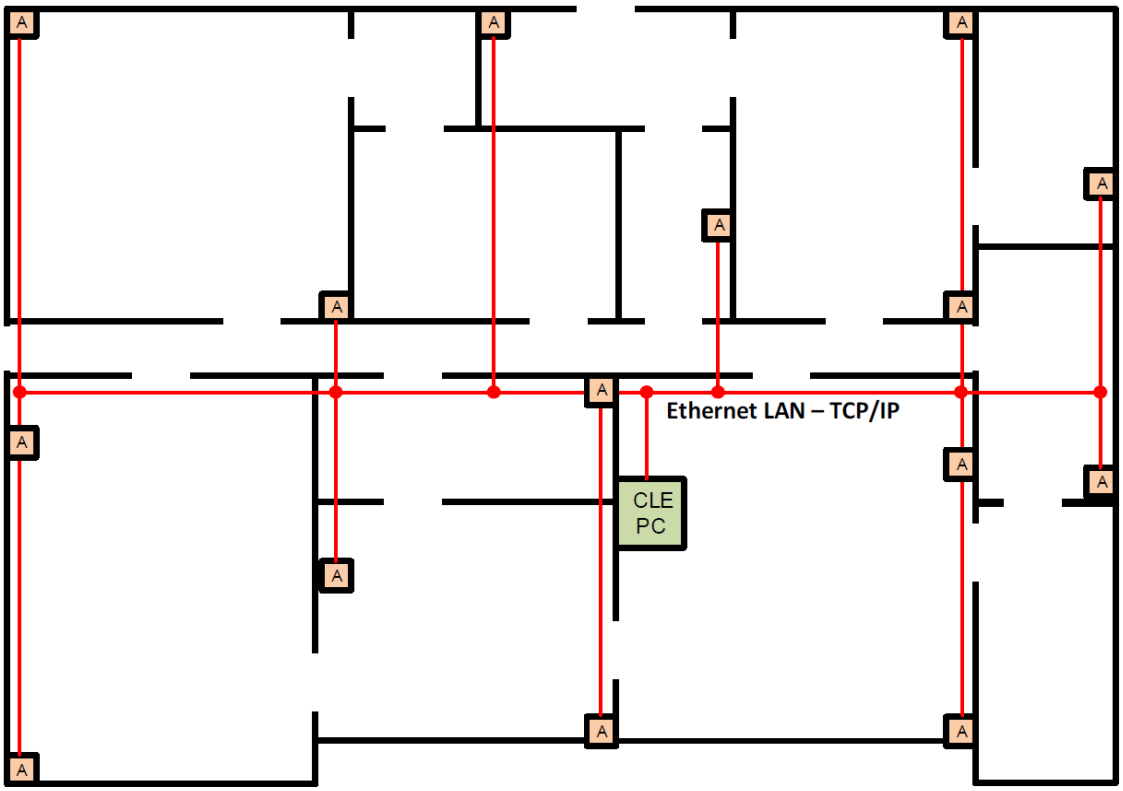

Figure 5. Typical expanded RTLS system with central location engine (CLE) [32].

2.6 UWB for precision locating

As presented in section 2.2, there are many ranging techniques for a RTLS implementation. Now, is presented a discussion for which combine techniques could improve the precision for the tag localization using UWB technology.

RSSI techniques are not going to be consider any further, since time-based schemes utilizing Ultra-Wideband can accomplish a significantly more exact outcome.

The AOA approach is not suited for the utilization of UWB technologies because, in the first place, utilization of antennas arrays expands the network cost, cancelling the primary advantage of UWB radio equipped with low-cost transceivers, and for carrier frequencies under 10GHz, where most

wireless networks exist, the antenna size can be very substantial. Having various antennas increment the shape of the receiver. So, it becomes unattractive.

Given the huge bandwidth of a UWB signal, the quantity of paths might be extensive, particularly in indoor situations. Thusly, precise angle estimation turns out to be difficult to get because of diffusing from objects that are part of the context. Besides, time-based methodologies can give exceptionally exact location measures, and subsequently they are better for UWB over the costlier AOA-based techniques.
The first one being Time of Arrival (discussed in section 2.2.3). In this approach the time of flight is measured in the well-known references nodes. Then this measure of time between the tag and the nodes will be denoted as $\hat{\tau}_n$, (x_n,y_n) the coordinates of the *n* references nodes, which are known to the CLE and lastly, (x_t,y_t) the unknown coordinates of the tag, to be estimated.

Estimating the location of the tag is equivalent to finding the intersection of three circles, as in figure 6.

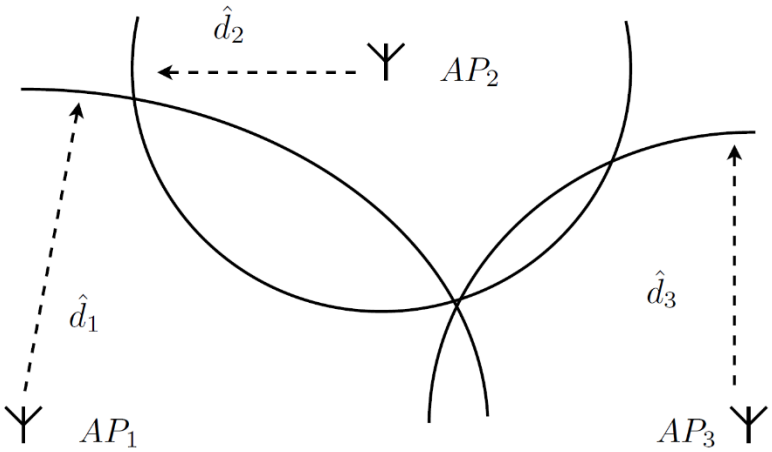

Figure 6. Example of using 2D trilateration for locating a tag. By estimating the distance between the tag and each of the three application points (TOF/RSS), the location of the tag can be estimated by finding the intersection of the three circles [1].

These circles are given by the following set of equations

$$(\hat{\tau}_1 c)^2 = (x_1 - x_t)^2 + (y_1 - y_t)^2 \tag{24}$$

$$(\hat{\tau}_2 c)^2 = (x_2 - x_t)^2 + (y_2 - y_t)^2 \tag{25}$$

$$(\hat{\tau}_3 c)^2 = (x_3 - x_t)^2 + (y_3 - y_t)^2 \tag{26}$$

This is called trilateration. In this method by using the geometry of triangles it is possible to estimate the relative position of objects, kind of how it is done in triangulation. The difference with triangulation, in which angles are measured along at least one known distance to a node, is that trilateration uses the known position of at least two nodes and the measured distance between the tag and every node to calculate the tags location.

Nevertheless, there is a limitation for this algorithm due that it need that all the tags and nodes share the same clock, which is something difficult to achieve. It is possible to synchronize the clock for every device in the system but the efficiency of the network would be compromised in an environment with high density of tags.

There is the algorithm called *time difference of arrival* TDOA based in TOA algorithm with the requirement that all clocks are synchronized. The tag will transmit a packet at time t_0 that will be received at time t_n for each one of the *n* reference nodes. Therefore, the times of flight are:

$$\tau_1 = t_1 - t_0 \tag{27}$$

$$\tau_2 = t_2 - t_0 \tag{28}$$

$$\tau_3 = t_3 - t_0 \tag{29}$$

Trilateration algorithm can be used to estimate the location (x_t,y_t) because the CLE knows all of these times, and they are all based in the same reference clock

Multilateration, also called hyperbolic positioning, is a method with which using the TDOA that arrive to three or more receivers, it is possible to compute the location of the emitter object. So, this time is observed at each of the reference nodes that have known locations. These nodes should have the same clock reference. The time difference of arrival between node number 1 and node number 2 is defined as $\Delta\tau_{12}$

$$\begin{aligned}\Delta\tau_{12} &= \tau_1 - \tau_2 \\ &= t_1 - t_0 - t_2 + t_0 \\ &= t_1 - t_2\end{aligned} \tag{30}$$

One of the nodes is taken as the system origin. Assuming node number 3 is located at (0, 0). Using the relationships defined in equations 24 to 30, we can write:

$$\begin{aligned}\Delta\tau_{13} &= \frac{1}{c}\sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2} - \frac{1}{c}\sqrt{(x_3 - x_t)^2 + (y_3 - y_t)^2} \\ &= \frac{1}{c}\left[\sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2} - \sqrt{x_t^2 + y_t^2}\right]\end{aligned} \tag{31}$$

$$\begin{aligned}\Delta\tau_{23} &= \frac{1}{c}\sqrt{(x_2 - x_t)^2 + (y_2 - y_t)^2} - \frac{1}{c}\sqrt{(x_3 - x_t)^2 + (y_3 - y_t)^2} \\ &= \frac{1}{c}\left[\sqrt{(x_2 - x_t)^2 + (y_2 - y_t)^2} - \sqrt{x_t^2 + y_t^2}\right]\end{aligned} \tag{32}$$

The estimation of the tag location (x_t,y_t) is defined by the intersection of the hyperbolas obtained from the two las equations, 31 and 32.

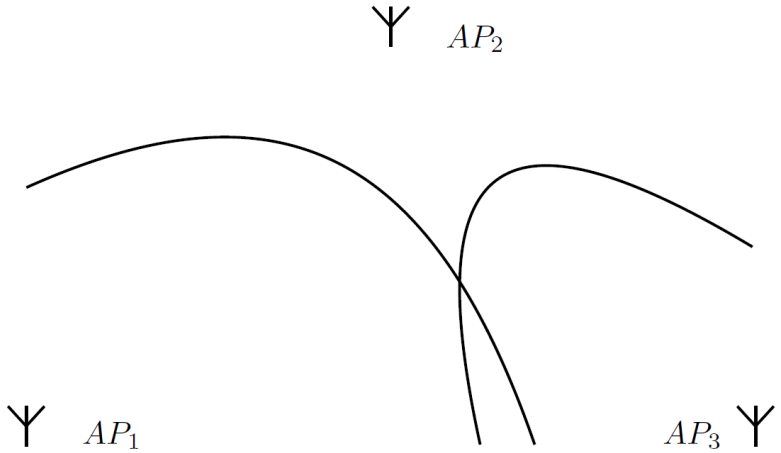

Figure 7. Example of using 2D multilateration for locating a tag. Application point two is taken as the system origin and two hyperbola traced using the time difference of arrival values and the intersection point of these two is the location of the tag. [1].

The accuracy of the arrival time estimates will define the accuracy of these two algorithms. Noise, harsh NLOS channels or interference between tags can be some causes for errors. So, to be able to achieve sub-meter precision it will be necessary to have UWB pulses. Almost every multipath can be resolved by its wide bandwidth, this would allow to resolve the direct path and with a very fine time resolution. Either way, the UWB signal has to be received coherently to achieve this potential precision. Which requires high sampling rates.

Chapter 3

Protocols and Standards overview

3.1 Introduction

In this section it will be given an overview of the standards and protocols that represent the foundations of the system that was developed for this thesis.

In section 3.2 the Ultra-wideband (UWB) standard IEEE 802.15.4.a is discussed. It is the technology adopted for the indoor localization system. In section 3.3 the standard Bluetooth Low Energy is presented, given that in a large-scale implementation it would be needed some a backhaul to have the possibility of send and receive data over it, as part of the management of the localization system. And Finally, in section 3.4 the protocol MQTT is summarized. It plays an important role as middleware between the clients and what it would be the CLE.

3.2 Ultra-wideband Standard IEEE 802.15.4.a for Localization

3.2.1 Introduction

The standard IEEE802.15.4a. has a typical very large bandwidth of about 500 MHz, in which it spreads the transmitted signal. Given that this spreading factor is large, because of the big bandwidth, it presents a good robustness against interference and fading.

IEEE802.15.4a. presents considerable advantages for geolocation, due that the precision of a ranging measurements depends on the bandwidth that can be employed, so in this case, having a large bandwidth influences directly on the capability of having high precision ranging (at least 1m accuracy)

3.2.2 Principles of UWB

3.2.2.1 Frequency regulation

The absolute bandwidth of ultrawideband signals is larger than 500 MHz [19]. A system based in this technology is less probable to interfere with other existing systems given the signal spreading over a large bandwidth. Hence, the unlicensed operation of UWB systems is (or will be) issuing rulings by worldwide frequency regulators that allows its operation, regardless weather UWB spectrum overlaps with another already existing system spectrums. There has to be, though, a limit on the power spectral

density, that represents a guaranteed adherence to a frequency mask. In the USA, emissions between 3.1 and 10.6 GHz are allowed.
In Europe it was defined the allowed operation for UWB devices in the frequency range between 4.2 and 4.8 GHz and the frequency range between 6 and 8.5 GHz. The power spectral density for the operating frequency band has to be under 41.3 dBm/MHz.
The spreading factor of low rate data transmission (on the order of 1000), for UWB transmission, is very large given its large absolute bandwidth. The transmission propagates over reasonable distances (10 – 50m), despite the restrictions on the power spectral density, thanks to the spreading previously mentioned. It also improves the robustness from narrowband interferers (jammers) and/or other UWB devices. Moreover, we get a more accurate ranging, due to the large absolute bandwidth, since the accuracy is proportional to the bandwidth of the emitted signal.

Furthermore, another advantage of UWB signals having a large relative bandwidth is that some frequency components have a better chance to propagate through, or around, obstacles. Meaning that there is diversity of propagation paths provided by the wide relative bandwidth, that leads to a higher robustness of the transmitted signal. Nevertheless, this effect is predominantly present in the frequencies below 1 GHz.

3.2.2.2 Transmission schemes for UWB

It is possible in many ways to spread signals to large bandwidths. From a signal processing point of view, UWB is just spread-spectrum with a very large spreading bandwidth. Consequently, it is possible to use any of the well-known spread-spectrum approaches. There are a couple of approaches that are currently use referring to high data-rate UWB systems:

1) *Direct sequence code division multiple access* [20] Is used in the high data-rate UWB system of the UWB Forum
2) A combination of frequency hopping, error-correction coding and repetition coding together with OFDM modulation is used in the ECMA268 standard for high data-rate UWB systems [21].

It offers the best tradeoff between complexity and performance when talking about low data rates, such as time-hopping impulse radio (TH-IR). TH-IR was proposed and investigated based on the principle that each data symbol is represented by a sequence of pulses with pseudorandom delays. Then, it is applied to the whole pulse sequence a type of modulation (either pulse position modulation PPM or quadrature amplitude modulation QAM). The sequence varies differently for each user. This permits the receiver to differentiate between multiple users. The transmitted spectrum is defined by the duration of the pulses. The 802.15.4a standard is based in this principle but includes some other features as well.

3.2.3 The UWB PHY

The UWB PHY waveform is based on an impulse radio signaling scheme using band-limited data pulses [22]. The UWB PHY operates in three different independent bands: the sub-gigahertz band, that includes only one channel (channel 0) and goes from 249.6 to 749.6 MHz in the spectrum. The low band, that includes four channels (from channel 1 to channel 4) and goes from 3.1 to 4.8 GHz in the spectrum. And the high band, that includes eleven channels (from channel 5 to channel 15) and

goes from 5.8 to 10.6 GHz in the spectrum. A device that operates with the UWB PHY must support at least one of the mandatory channels (channel 0 for sub-gigahertz band, channel 3 for low band, and channel 9 for high band). The channel 4, 7, 11 and 15 are optional and they have a particular larger bandwidth (bigger than 500 MHz). Thanks to this large bandwidth the devices can transmit at a higher power and get a longer communication range. Moreover, given that these pulses have a larger bandwidth the multipath resistance sees itself improved and this lead to a more precise range estimation.
For the network to success it will be important the ability to relocate within the spectrum given that in the deployment area it is more than possible to find multiple types of wireless networks operating in the same frequency bands. It is allowed by the standard the dynamic channel selection within the operating band. Aiming to achieve this, the PHY contains several lower-level functions, such as receiver energy detection (ED), link quality indication (LQI), and channel switching. Thanks to these functions it is enable the channel assessment and frequency agility.

3.2.3.1 The UWB Frame Format

The packet format utilized for the communication between devices for IEEE 802.15.4a networks is as follows:

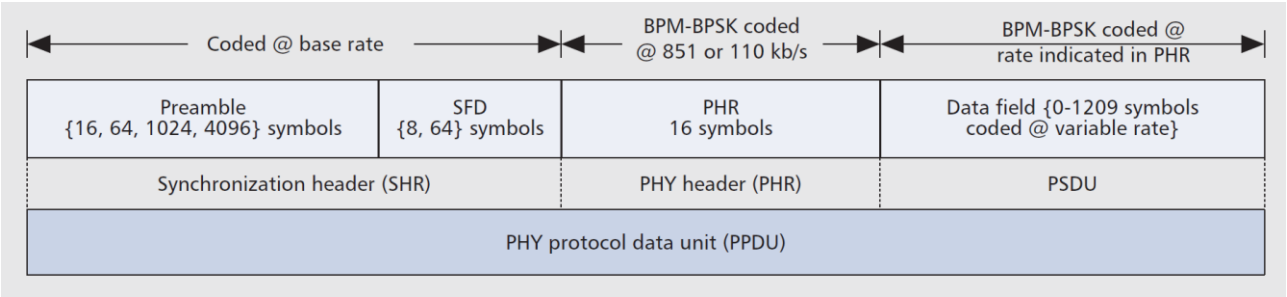

Figure 8. UWB PHY frame structure [21].

Each packet, or PHY protocol data unit (PPDU), contains a synchronization header (SHR) preamble, a PHY header (PHR), and a data field, or PHY service data unit (PSDU). The SHR preamble is composed of a preamble and a start-of-frame delimiter (SFD). The SFD delimits the end of the preamble and the beginning of the PHY header. It is usual to define the frame timing and its detection is important for accurate ranging counting. The UWB PHY supports a mandatory short SFD (8 symbols) for default and medium data rates, and an optional long SFD (64 symbols) for the nominal low data rate of 110 kb/s. [21].

Each application defines, by its requirements, the number of symbols in the preamble. There can be 16, 64, 1024, or 4096 symbols in the preamble, yielding different time durations for the SHR of the UWB frame. The last two ones with the biggest values, 1024 and 4096, are the best one for noncoherent receivers in terms of signal to noise ratio (SNR) via processing gain. Each symbol of preamble uses a preamble code, of two different lengths of 31 or 127. Every preamble code is defined by a ternary alphabet {-1,0,1}, they have perfect periodic autocorrelation properties and because of those they are selected to be used in the UWB PHY. PHY can support two different preamble codes. But it needs to use mandatorily just one preamble symbol length.

The frame is attached to the PHR, after the SHR has been created, with a length of 16 symbols. The PHR contains information useful for the decoding of the packet at the receiver side. Typical packets size are in the order of several bytes, for non-demanding applications. Instead for others more demanding ones may require larger packets size.

3.2.3.2 Management of PHY options

UWB PHY addresses a broad spectrum of applications and service conditions due to a very rigid framework of option management rules. UWB PHY has numerous varieties of optional modes configurations. The modes to be implemented allowed by the UWB PHY are the following ones:

A very rigid framework of option management rules governs the UWB PHY in a way that addresses a broad spectrum of applications and service conditions. Toward this, the specification of the UWB PHY includes a rich set of optional modes and operational configurations. These modes result from the list of available variables the UWB PHY allows to be implemented, including [21]:

• Center frequencies
• Occupied bandwidth
• Mean PRFs
• Chip rates
• Data rates
• Preamble codes
• Preamble symbol lengths
• Forward error correction (FEC) options (i.e., no FEC, Reed-Solomon (RS) only, convolutional only, or RS with convolution)
• Waveforms (one mandatory and four optional pulse types)
• Optional use of clear channel assessment (CCA)
• Optional ranging (private or not)

Just a few ones of the combinations of capabilities are mandatory modes, ensuring in this way a low-cost objective for the standard.

3.2.3.3 UWB PHY modulation

Coherent and non-coherent receivers are supported by using a common signaling scheme of burst position modulation (BPM) and binary phase shift keying (BPSK). The symbols are composed of an active burst of UWB pulses, that are modulated by the combined BPM-BPSK. Each symbol is able to carry two bits of information: the first bit is used to get the position of a burst of pulses, and the second one to modulate the phase of that same burst.

3.2.3.4 General Radio Specifications

3.2.3.4.1 Sensitivity and range

The standard IEEE 802.15.4a specifies receiver sensitivities of at least –85 dBm for the 1 Mb/s case, and at least –91 dBm for the 250 kb/s case of the CSS PHY. Of course, manufacturing tolerances are covered given that these values have a margin of tolerance, to allow low-cost implementations.

It depends on the receiver sensitivity and transmission power the range that is going to be obtained. The standard does not stablish a minimum transmission power for UWB devices. However, they need to transmit at low power, as possible, to not interfere with other systems. Instead, for maximum transmission power it depends on the local regulatory entities.

3.2.3.4.2 UWB Band Coexistence

Low-rate wireless personal area network (LR-WPAN) devices share the spectrum with other PHYs that also operate in the unlicensed bands. For this reason, it is important that they have a good coexistence performance with other systems. There are many features of the UWB PHY that ensure the coexistence, such as los power spectral density (PSD), according to the regulation for UWB in different parts of the world. Multiple bands and operating frequencies within each band to avoid the use of bands that are already being utilized, and optional modes that deliver shorter symbol timing to minimize the channel occupancy.

3.2.3.4.3 Band Plan

The UWB signals bandwidth must be selected, it is possible to increase the transmitted power by increasing the signal bandwidth and the higher it gets the higher degree of delay diversity is obtained.

The optimal bandwidth for UWB channels lies between 100 MHz and 2 GHz. The clock-speed and the speed of the receiver electronics will depend on the bandwidth of the system. As well as the cost of the system, the lower the bandwidth de lower the cost.

freq. band	Fc (MHz)	BW(MHz)	Admissible region
0	339.36	499.2	USA
1	3494.4	499.2	USA, Europe
2	3993.6	499.2	USA, Europe, Japan
3	4492.8	499.2	USA, Europe, Japan
4	3993.6	1331.1	USA, Europe, Japan
5	6489.6	499.2	USA, Europe
6	6988.8	499.2	USA, Europe
7	6489.6	1081.6	USA, Europe
8	7488.0	499.2	USA, Europe, Japan
9	7987.2	499.2	USA, Europe, Japan
10	8486.4	499.2	USA, Japan
11	7887.2	1331.2	USA, Japan
12	8985.6	499.2	USA, Japan
13	9484.8	499.2	USA, Japan
14	9484.0	499.2	USA, Japan
15	9484.8	1354.9	USA, Japan

Table 1. IEEE 802.15.4a UWB frequency bands. fc: center frequency [23].

3.2.4 UWB Geo-location

3.2.4.1 Ranging

Thanks to the support of some specific PHY capabilities, as well as some MAC behaviors and protocols, ranging is possible to achieve. In Figure 9 is shown the mandatory ranging protocol two-way ranging, as mentioned in section 2.5.2, it does not need a common time reference for the ranging estimates.

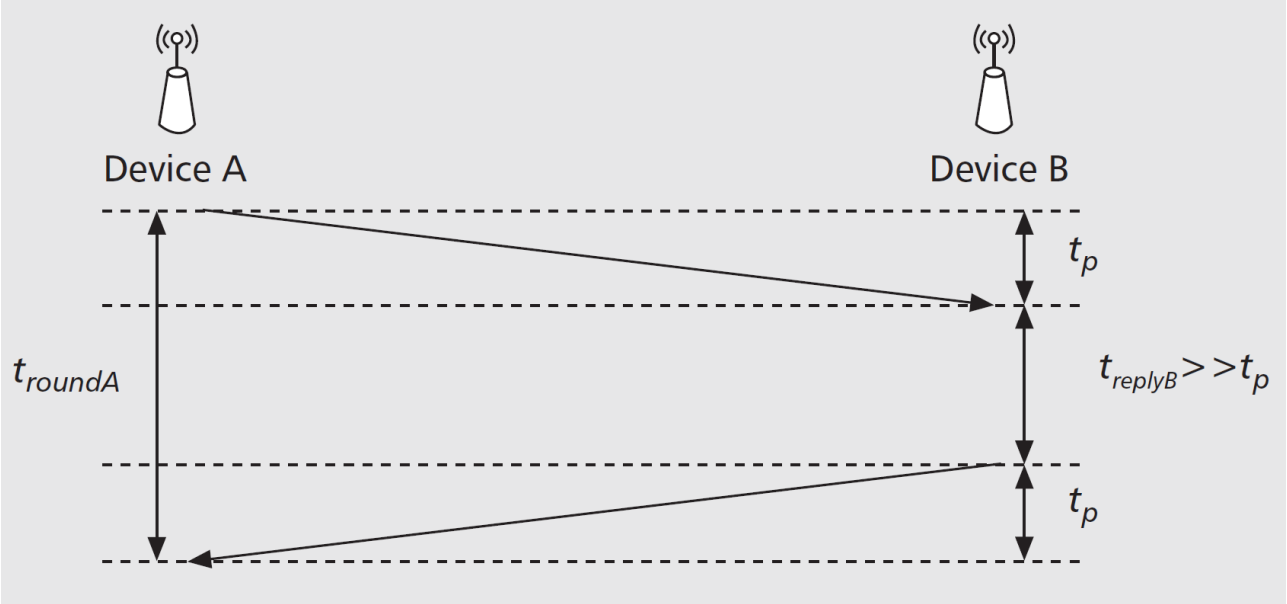

Figure 9. Exchange of message in two-way ranging [21].

In this scheme, the device A is capable of doing ranging so it begins the session by sending a ranging request packet to device B, afterwards device B waits for a given time "t_{replyB}", that is known by both devices, to send a request back to device A. Then device A will be able to measure the round-trip time $t_{roundA}=2tp+t_{replyB}$ to then calculate the one-way time-of-fligh "tp" with respect to its own reference time. The large bandwidth of the UWB signals allows the system to have high precision, having ranging estimations of less than 3ns, which delivers an uncertainty of less than 1 meter.

The transmitting PHY, for frames used in ranging, sets a bit value in the PHR of the packet, called the ranging bit, and serves to let know to the receiver that this particular frame is intended for ranging. This UWB frame is called ranging frame (RFRAME). The ranging bit is the only thing that makes the frame unique. The RFRAME is able to transport data, and it does not necessarily require and acknowledgment.

3.2.4.2 Detection in LOS and NLOS

Th roundtrip time of the packet has to be estimated by the receiver, this is a main task for itself, to do so need the identification of the first arriving MPC. In channels where the MPC is strong this task is yielded relatively easy, like in LOS scenarios. But in NLOS scenarios it is much more difficult, because there is a delay of several tens of nanoseconds between the first and strongest components, leading to a lack of precision of the range estimate.

The same preamble that is used for synchronization and channel estimation is also utilized for doing ranging. The received signal is a periodic repetition of the channel impulse response and because of this it is possible to estimate the different MPCs from the preamble. However, the receiver has a noisy estimate of the impulse response as well as the arrival time of the strongest MPC. From this point, the receiver is able to "search back" to find any possible earlier MPCs, thanks to the previously mentioned arrival time.

There are two main issues with the "search back":

1) The high-level noise of the received signal. There is a threshold that must be exceeded to count that determinate component. Then it must be understood if that component is an MPC or not.
2) The received signal arrives in clusters. It is possible that the first arriving cluster does not contain the strongest MPC.

3.3 Bluetooth Low Energy

3.3.1 Introduction

Bluetooth has become the most popular way to share voice, data, music, etc. between two paired devices. Lately, it has been introduced the version 4.0 with low energy features to stablish the communication between a new generation of low power devices.

Bluetooth low energy (BLE) allows the devices to function on a single charged battery for a long period of time (several months). This is ideal for a large-scale localization system, where the tags will need to communicate with the CLE using as a backhaul the BLE connection.

3.3.2 The BLE Stack

In this section the BLE stack, as shown in figure 10, will be described, partying from the bottom of the stack, the physical layer, to the GATT profile. Being this information portrayed as in the Bluetooth Core Specifications [24]

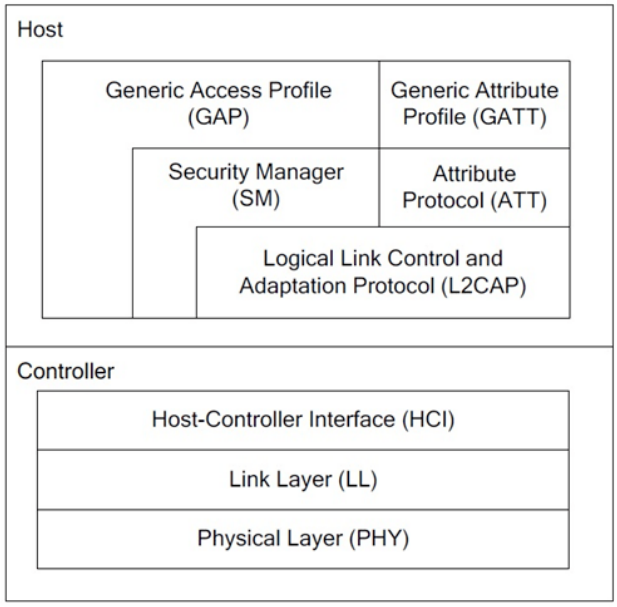

Figure 10. The BLE protocol Stack [25].

3.3.2.1 Physical Layer

Bluetooth operates in the unlicensed 2.4 GHz ISM band, it is composed by 79 channels in the range between 2.4 GHz and 2.4835 GHz with a 1 MHz spacing from one to another. However, even if BLE operates in the same band it only uses 40 channels with a 2 MHz spacing, so it has a bigger bandwidth.

As for the BLE specifications it has a 1 Mbit/s maximum bit rate, a power feed to the antenna between -20 dBm and 10 dBm, and it uses Gaussian Frequency Shift Keying (GFSK).

In BLE there are specific channels for sending the advertisement messages, that includes data as their own existence and rudimentary information of the device. Channels 37 at 2402 MHz, 38 at 2426 MHz and 39 at 80MHz are dedicated to send advertisement data. The rest of the 37 channels can be used for sending data payload. In figure 11 is shown an overview of the BLE channels.

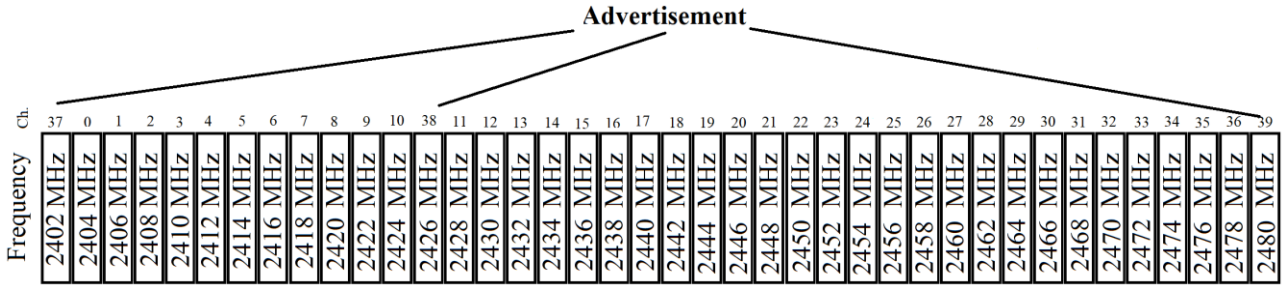

Figure 11. The BLE channel built-up.

The advertisement channels are located outside the most common WIFI channels, having WIFI Channel 1 overlapping with BLE channels 0 to 8, WIFI Channel 6 overlapping with BLE channels 11 to 20 and WIFI channel 11 overlapping with BLE channels 24 to 32. In this way the interference between both standards is reduced. BLE uses Frequency Division Multiple Access (FDMA) so the device can only get one channel assigned, so when it synchronizes that channel in frequency and timing then it will be connected.

During advertising a device broadcasts information on one of the advertisement channels. So, another BLE device, that is located in the vicinities, can receive this data without even establish a connection. Single channels can be deactivated to prevent collisions. As said previously, this broadcast can contain general information. For example, a device that measures temperature can broadcast this measure, without being necessary that other devices connect to it to read the data. If needed, these devices can send connection request to get more information from the transmitter device. In figure 12 is shown this procedure with its time measures.

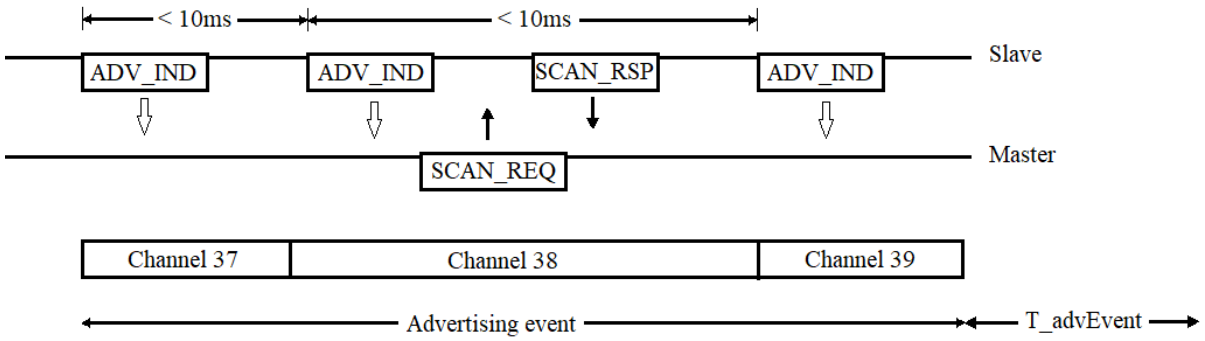

Figure 12. BLE advertisement event with scan request [24].

After a connection request is sent, two, or more, devices make a piconet. The connection procedure is shown in figure 13.

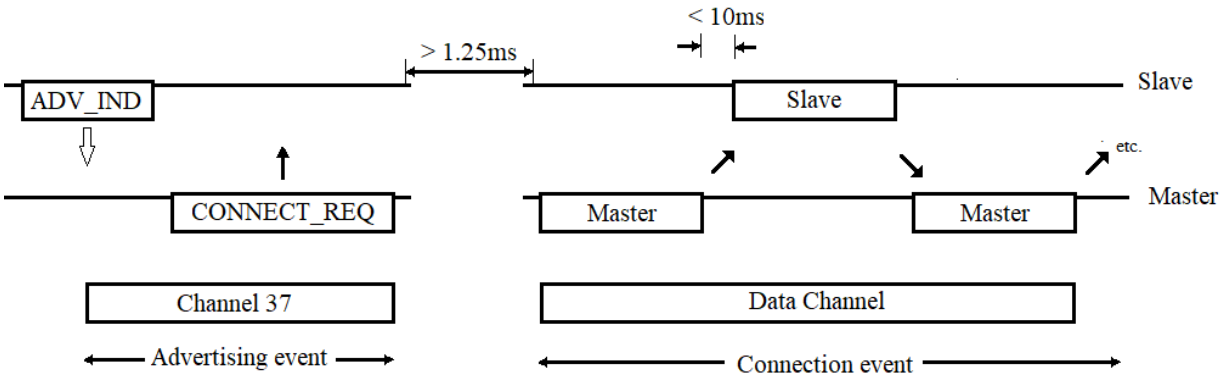

Figure 13. Illustration of a BLE connection event [24].

A piconet is defined by a channel map, pseudo random sets of physical channels, where the device can connect to. They also set the timings and indexes for the physical channel to be used for the connection request, the timeslot in which the device can transmit data is predetermined, as well. So, the second scheme used in BLE is time division multiple access (TDMA). So, in a piconet there is only one master (device that has the control of the net and the access of the channel) and any number of devices that can communicate at the same time. But the communication is between the master and the salves (slaves cannot communicate between them). However, these devices can belong to more than one piconet, and in some cases a master of a piconet could be a slave in a different piconet.

Transmissions begin with the master sending a data packet. Events are used to send a data packet between the devices in a piconet. A slave can reply to the master using the same physical channel but at a specific time. After this event both devices change to next frequency. Even if it is established at the setup, the channel map can vary to avoid interferences.

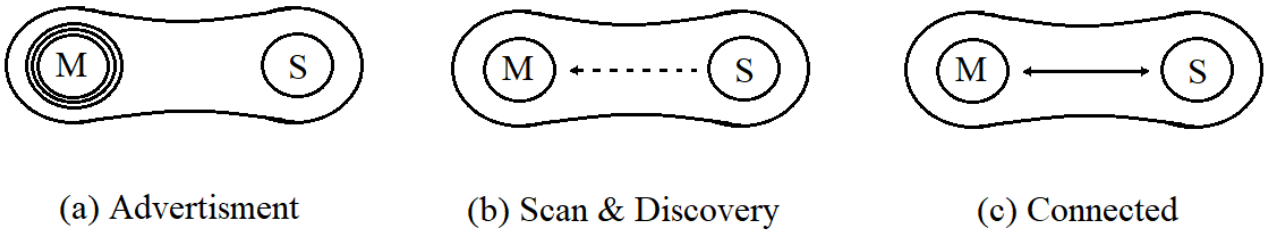

Figure 14. The BLE operation modes.

There are different modes supported by the BLE devices, as shown in figure 14. To illustrate in a better way the possible BLE topologies figure 15 is presented with two examples.

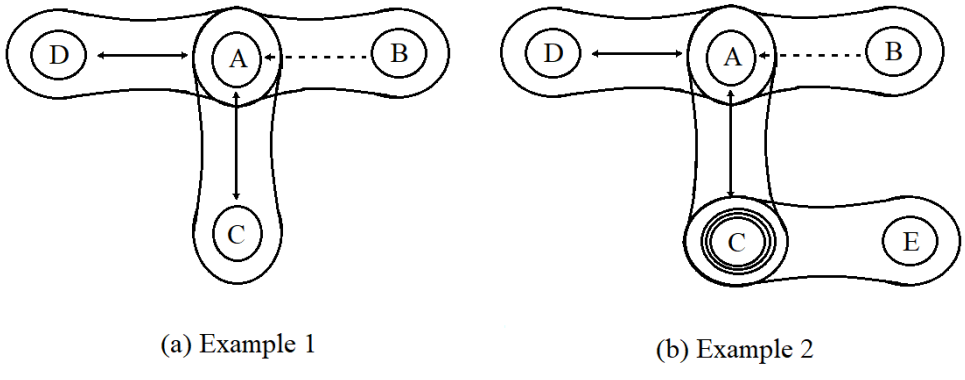

Figure 15. Example topologies for BLE Piconets.

In figure 15(a) device A is a slave in two different piconets, having as master devices C and D for each one of them, however device A is an advertiser so device B tries to connect to it. Then, in figure 15(b) is presented a case where device A is the master of a piconet having as slaves devices C and D and device B is, again, sending a connection request to A. However, device E is in range of device C so it can listen to it, but without attempting any connection.

If a master and salve are in a point-to-point connection then the BLE stack implements physical links. There is an Access address that has the function of identifier for these links, that is exchanged using a Link-Layer packet.

3.3.2.2 Link Layer

The Link Layer (LL) is formed by two types of logical transports. The first one is the asynchronous connection-less (ACL) logical transport, it carries control signals and user data. This is a connection established between the master and a slave in a piconet and each one has an identification access channel number. They then are removed when the devices disconnect.
The second one is the advertising broadcast logical transport, it sends control and user data to all listening devices in the area. There is not an acknowledgement expected from the devices, so data is retransmitted a few times to get a higher level of reliability. The majority of this data is transmitted in just one direction, from the master to the slaves. However, slaves can send requests to the master, asking for more information, or an ACL connection.

There are four core protocols that can be used in the logical transport links for receiving or sending data. These links are:

- Logical link identifier (LLID): It is part of the payload header.
- Control logical link (LE-C): It sends control data over the ACL logical transport. Used for configuration and connection built up.
- User asynchronous logical link (LE-U): Used to communicate user payload data. Each packet is identified an LLID, to define if the packet contains a start or a continuation of a higher-level frame. This makes the reassembly, for the higher level, easier.
- Advertising broadcast control (ADVB-C): It handles scanning and connection requests.
- User data logical links (ADVB-U): It sends broadcast payloads without a connection.

In figure 16 it is shown the headers for physical and logical layers. It is shown that there is a CRC area in the packet, it is 24-bits long and calculated for already encrypted packages.

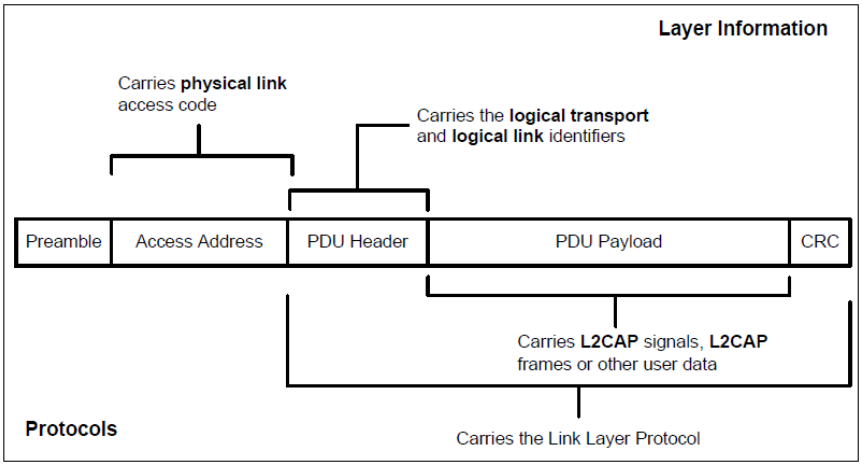

Figure 16. Entire BLE packet structure [24].

3.3.2.3 Link Control (L2CAP)

The logical link control and application layer protocol (L2CAP) provides logical channels to higher layers, to send data with or without connection. It has the possibility to interact directly with the link layer in the same stack, or it can connect to a Host-Controller Interface (HCI), as shown in the BLE protocol stack in figure 10.

The L2CAP has the following tasks:

- Protocol and channel multiplexing
- Segmentation and reassembly
- Flow control
- Error control
- Streaming
- Fragmentation
- Quality of service.

The architectural structure of the L2CAP layer is shown in figure 17. The channel manager communicates with control signals from the upper and lower layers, it controls the L2CAP layers functionalities, by using a set of control messages over dedicated signalling channels, for running a state machine. The recourse manager coordinates the data packages, it provides a frame service to the channel manager to have a continuous connection while managing frames from different services.

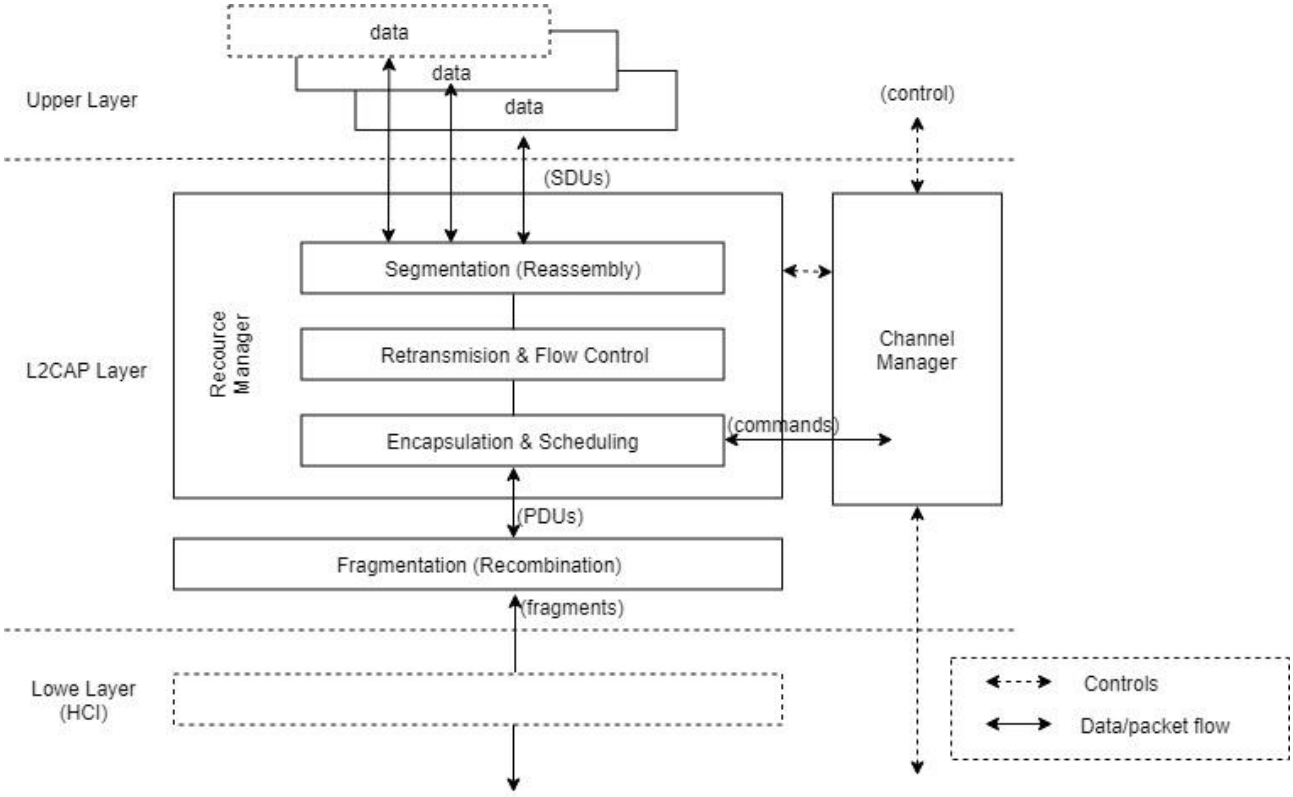

Figure 17. Illustration of the BLE L2CAP layer architecture [24].

L2CAP uses channels that have an id, so-called channel identifier (CID), to set the configuration for the communication. Some of these channels have particular and specific functions, such as signalling and configuration, some others can be dynamically allocated for general purpose data

communication. There are a set of parameters is every channel that can be modified by the L2CAP configuration mechanism.

Packets that arrive from higher levels can be divided to allow flow control for connection-oriented data. The SDU length field, that is part of the first LE information fame (LE-frame) defines the size of the L2CAP SDU entirely. LE-frames are formed by a L2CAP header, the SDU length and the payload, as shown in figure 18. The L2CAP header yields the frames size and the channel ID (CID), that determines the destination of the packet.

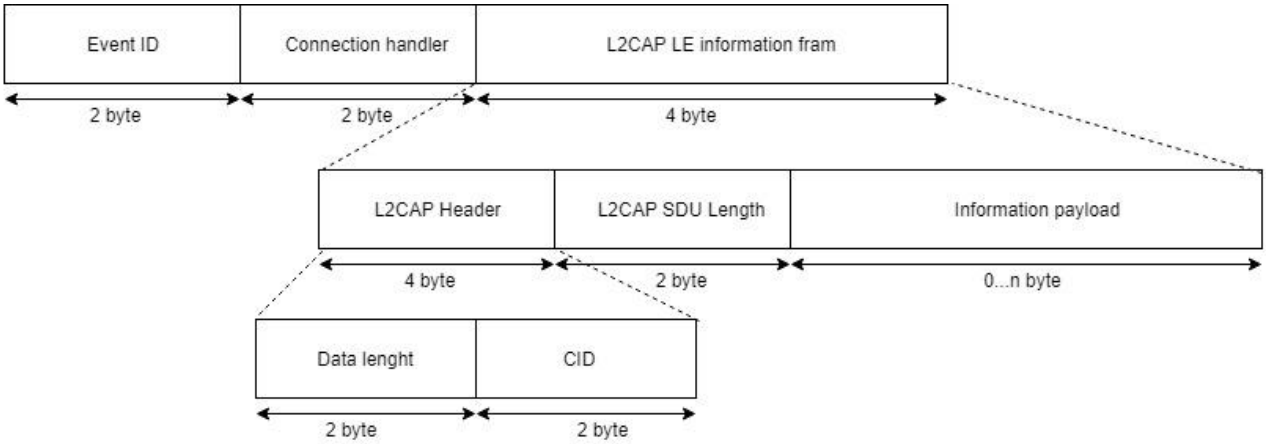

Figure 18. Structure of the L2CAP packet [24].

A number of credits is given to every channel, in the Credit based flow control mode, along the connection establishment. The device is going to decrement this value with each LE-frame that is sent. If the credit counter gets to 0 then LE-frame is not going to be sent anymore. When a device receives a LE flow control credit packet with a value of zero then the connection is terminated. In the case that a connectionless data channel would be need, for sending advertising information, it would be transmitted a Group Frame (G-Frame). The channel id is flagged as “0x0002”, that is the CID for connectionless traffic. Additionally, there is a protocol/service multiplexer (PSM) field added to the frame.

Multiplexing is the process of mapping all the traffic to the LL channel, as well as indicate the route the data should take to get to the upper layer entity. So, L2CAP is able to multiplex channels and protocols from higher layers, so the communication over on logical link is feasible.

Some features where implemented to improve the reliability of L2CAP, a main characteristic is the acknowledgment that will determined if the packet arrived to the receiver and retransmit it if it did not arrive to its destination. There are defined two bits for the transmission process, the RetransmissionDisablebit (R) and the RetransmissionTimer. There is a timer that gets activated when a frame is sent, and if the timer finishes its count down without receiving an acknowledge then it will retransmit the frame. The last received R needs to be set as zero to allow the information frames to be transmitted. There is the possibility to send new frames even if the earlier ones have not been acknowledged at the moment.

Recovery actions will take place if deeper layer error or L2CAP procedural errors are presented during any exception. If any unexpected behaviour is obtained by checking the transmit and receive sequence number then an error will be held, e.g. if a packet is received twice by checking the sequence number, it will be obvious that there is a duplicate and one of them will be discarded.

The L2CAP has also the possibility of fragmenting data into smaller pieces before sending it to the lower layers, to then be recombined. The lower layer controllers are designed to try to deliver packaged in sequence without error. Nonetheless L2CAP will put back together the PDUs and SDUs and then check if they were correctly received. The controller can send one packet frame at a time to the L2CAP on arrival or it can wait before to have a group of packets and then send them all together. If there are any received packet with a length different from the amount of received data, they will be discarded, unless it was explicitly specified not to be discarded.

3.3.2.4 Attribute Protocol (ATT)

The Attribute Protocol (ATT) defines devices into two different types. The first one being server devices, that can provide a set of attributes, and the second one being the client devices than can access to the previously mentioned attributes. For example, a piconet client can be used as an ATT server, creating another piconet master with information in it. BLE devices support both, server and client, roles. The kinds of attribute used are *discovering, reading* and *writing*. The server can, as well, indicate and notify about specific attributes to the clients. Attributes are a set of data defined by a particular type, defined by a universal unique identifier (UUID), an attribute handler and a set of permissions that were previously assigned by higher layer specifications.

The universal unique identifier (UUID) is and identifier assigned to the attribute type and can be of 16 bit, 32 bit or 128 bit. Even if, between these options there are a couple that are short and are supported, they are translated into 128 bit UUIDs by using the Bluetooth Base UUID. A certain set of 16 bit and 32 bit UUID are reserved for a certain type of attributes for BLE. All the other 128 bit UUIDs are assigned to custom attributes. In figure 19 is shown a couple of examples, the first one depicts an specified attribute with 16 bit UUID and the second one a custom attribute.

Handle	UUID	Permission	Value
0x0027	0X0180D (Heart Rate)	R/W	bpm
0x0032	0x6FDDFFA55F04F7B2D545AAE60001555A	R/W/AUTH	"random value"

Figure 19. Example for ATT data [24].

The 16-bit attribute handle is the one in charge for the discovering of available attributes in an ATT server. These ones are different from the attribute identifiers and should not be confused. The range for the attribute handles goes from 0x0001 to 0xFFFF, but in praxis it is reduced by the actual number of implemented attributes. It was mentioned before that a device can work as a client or a server, but it can run just run just one server at a time. Devices working as clients can scan a specific handle range, using a Find Information Request, to discover services and attributes. If these services or attributes are available then the server would reply with a Find Information Response that will contain the handles and UUIDs of everything that was requested.

A client will be able to read the value of a certain attribute after sending a Read Request to the server with the respective handle of the attribute. The server will send back the Read Response If read permission is given, otherwise it would not send it if the encryption requirements are not fulfilled or if the handle is not valid. The maximus packet size for ATT layer connections is set by the default

size of the Maximum Transmission UNIT (MTU) which is 23 B. This value does not define the maximum length of attributes, though. They can be longer if needed so. For this last case, the Read Blob Requests and Read Blob Responses will read the whole attribute value in MTU sized segments.

The ATT layer has multiple operations to write on attributes. The server can receive a Write request sent by the client, together with the handler and a value to change in given attribute. This value will be changed if the permission is granted and the encryption requirements are the right ones. The attribute could be shortened if the new values are shorter than the ones before. Afterwards there is going to be sent an acknowledgment to the client, called Write Response. It is possible to write just certain segment of an attribute by sending a Prepare Write Request, which is a request for that matter. It is followed by the Prepare Write Response. Then an Execute Write Request will write these segments after being accumulated. Table 2 shows the different options presented for the interaction between an ATT server and the clients.

Attribute PDU Name	Opcode	Parameters
Error Response	0x01	Request Opcode in Error, Attribute Handle In Error, Error Code
Exchange MTU Request	0x02	Client Rx MTU
Exchange MTU Response	0x03	Server Rx MTU
Find Information Request	0x04	Starting Handle, Ending Handle, UUID
Find Information Response	0x05	Format, Information Data
Find By Type Value Request	0x06	Starting Handle, Ending Handle, Attribute Type, Attribute Value
Find By Type Value Response	0x07	Handles Information List
Read By Type Request	0x08	Starting Handle, Ending Handle, UUID
Read By Type Response	0x09	Length, Attribute Data List
Read Request	0x0A	Attribute Handle
Read Response	0x0B	Attribute Value
Read Blob Request	0x0C	Attribute Handle, Value Offset
Read Blob Response	0x0D	Part Attribute Value
Read Multiple Request	0x0E	Handle Set
Read Multiple Response	0x0F	Value Set
Read by Group Type Request	0x10	Start Handle, Ending Handle, UUID
Read by Group Type Response	0x11	Length, Attribute Data List
Write Request	0x12	Attribute Handle, Attribute Value
Write Response	0x13	-
Write Command	0x52	Attribute Handle, Attribute Value
Prepare Write Request	0x16	Attribute Handle, Value Offset, Part Attribute Value
Prepare Write Response	0x17	Attribute Handle, Value Offset, Part Attribute Value
Execute Write Request	0x18	Flags
Execute Write Response	0x19	-
Handle Value Notification	0x1B	Attribute Handle, Attribute Value
Handle Value Indication	0x1D	Attribute Handle, Attribute Value
Handle Value Confirmation	0x1E	-
Signed Write Command	0xD2	Attribute Handle, Attribute Value, Authentication Signature

Table 2. Summary of ATT operations with opcodes and the respective parameters [24].

The notification and indication features are two very useful for BLE communications. They are both mainly used to let a client know when a change takes place in one of the server's attributes. So, a

client can be able to get updates of specific attribute values without polling them, reducing, in this way, the read requests and responses. So, the attribute is configured just to notify or identify a client. In one hand, notifications do not need to be acknowledged by the client, keeping the traffic low in this case. On the other hand, indications need a Handle Value Confirmation needs to be sent to the client. These two features would be useful for sensor systems where new measured values could be sent regularly.

Some kinds of read and write operations need authorization or even an encrypted connection. This security measures need to be fulfilled, otherwise the ATT layer will send error messages back to the requesting device. These requirements are higher layer ones so they could change accordingly to the device. Clients would need to check their permissions on an attribute, though, given that they might have read permissions without a proper connection, but, instead, they might need some authenticated connection for the writing operation. The attribute permissions are reading, writing, encryption, authentication and authorization.

3.3.2.5 Generic Access Profile (GAP)

An important feature of Bluetooth is that it can provide inter device communication, for universal functions on all kinds of devices, despite the manufacturer. Because of that devices can are discoverable and connectable even for units with different functionalities. So, the GAP profile defines the framework and communication schemes for BLE devices to have a standardized communication between devices.

There are four established GAP roles for BLE devices: Broadcaster, Observer, Peripheral and Central. The Broadcaster role will transmit advertisement messages for letting everybody know of the existence of a device, or even to send some small amount of data to a high number of units without needing to meet a connection status.
Then, a device operating with the Observer role, would get these messages. A device operating with this role will listening to all the messages being broadcasted from all the three BLE advertisement channels.
Instead, if a connection takes place then a device would assume the Peripheral role, that would be in a physical link with a device assuming a Central role. So, the Peripheral would be a link-layer Slave, while the Central would be a link-layer Master. A device could perform more than one of these roles, just if supported by the lower layers.

Apart from these four roles, a set of modes and procedures are supported by GAP for the inter device communication. They can take place simultaneously to allow the access to basic BLE communication functions for the users. These modes and procedures can be categorized into four groups:

- Broadcast mode and observation procedures
- Discovery modes and procedures
- Connection modes and procedures
- Boning modes and procedures

In the Broadcasting mode devices are going to send unidirectional connectionless advertisement packets. These packets could be scannable advertising data o non-connectible data. Then, a Peripheral device would use the observation procedure to get connectionless data from an advertiser.
For the Discoverable mode, every active BLE unit support three subcategories: Non-Discoverable mode, Limited Discoverable mode and General Discoverable mode. Even if the device is in Non-

Discoverable mode it can advertise to other BLE units. While in Limited mode the devices are going to be scannable just for a limited amount of time. Afterwards the state will go back to the non-discoverable one, rather it stablished a connection or not. This behavior based on time is not implemented on the General mode. If the device is in the Central GAP role it will be able to run the Discovery procedures, such as receive addresses, advertising and scan response data. The devices that are in General mode can anyway receive data from devices in any other mode.

As its name implies, the connection modes and procedures aim to establish a connection between devices to then send connection-oriented data. It presents three modes. The first one being the non-connectable mode, in which Peripheral, either Broadcasters or Observers, do not allow any connection establishment. The second one being the Direct connection mode in which only known devices are allowed. Instead, the third one, Undirect connection mode, supports unknown devices. Just devices in the Peripheral role can used these last two modes. Furthermore, a set of connection procedures is implemented:

- **Auto:** The host would automatically connect to a certain number of devices, after receiving an advertisement packet saved in a common white list.

- **General:** All the advertisements sent by the Broadcasters are accepted, afterwards there will be held a scanning of the parameters to then compare it to the connectible devices list.

- **Selective:** There is a selection made by the host to define the connection white list of allowed devices. Just the advertisements sent by these devices would be accepted.

- **Direct:** It will accept just one device, directly, ignoring all the other ones that are part of the white list.

Furthermore, we can find the connection parameter update procedure that will modify the parameter of the link layer, of a Peripheral or a Central, while connected. These parameters will be changed by routines based on the link layer or the L2CAP. When the host would want to terminate a connection, it will use the terminate connection procedure.

It was introduced the Bluetooth concept of trusted relationships, thanks to the development of device bonding. It would mean that two devices after exchanging and storing the connection and security information for the first time would be able to share data without needing to redo it no more in the future. Another important procedure would be the bonding procedure which when a device gets a request it could optionally implement the bondable mode that would allow the host to two units, saving this connection for future works, and this procedure would be used by the Peripheral or the Central.

3.3.2.6 The Generic Attribute Profile (GATT)

The Generic Attribute Profile (GATT) defines a set of service protocols and formats, aiming to establish communication over the attribute protocol. The GATT attributes can be read, written, discovered, notified or indicated by using specific procedures. It is also used for the broadcast configurations of certain characteristics. The Bluetooth Special Interest Group (SIG) defines GATT profiles that are used for general purpose applications, e.g. temperature sensors, cycling computers, etc. These profiles have to be in accordance to the specification to be assure their interoperability.
However, profiles can also be defined by the programmer to suit the specifications of a particular purpose.

Two roles are defined for the GATT profile:

- **Server:** The device functioning as a server will offer attributes than can be accessed.
- **Client:** The device functioning as a client will access to the data on a server.

It was mentioned before, in the ATT section, that these roles have nothing to do with the master and salve roles of piconets. So, a device that already is working as a master in a piconet can also implement a GATT client and vice versa. It is also possible for a device to implement both roles at the same time, or change them depending on the needs of the procedure. E.g. in an BLE temperature sensor in which the sensor itself would be a server and it would be accessed by a device, such as a smartphone, working as a client that would go and read a specific attribute were the temperature measured is saved.

GATT packets are composed by a set of ATT data, they are formed by a handle, an attribute type, a value and a field associated to this last one, for the permissions. Then, they are encapsulated in an attribute protocol PDU with also an opcode and an optional signature. This data structure, with its respective byte length, is shown in figure 20. The previously mentioned opcode will contain the operation code of one of the following requests: Response, indication, notification or confirmation. Or it will contain a flag for authentication. Furthermore, in terms of security it could be added an optional authentication signature.

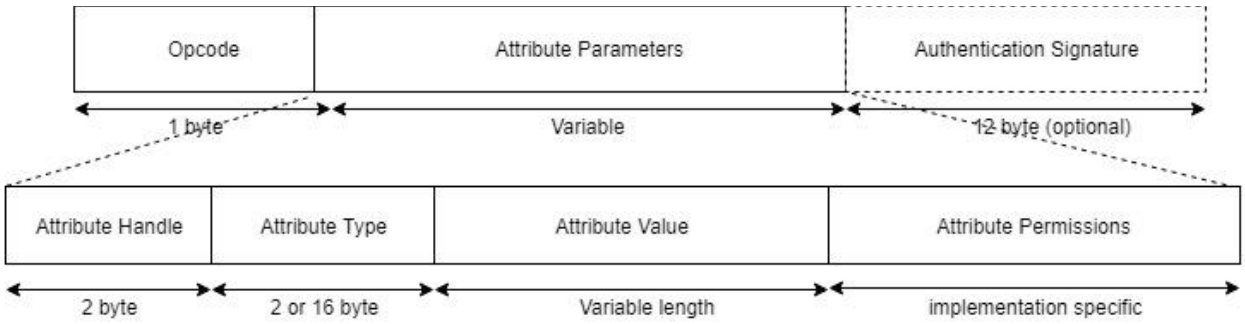

Figure 20. Example for GATT PDU structure [24].

All the access permissions of the client are specified in a field that every attribute has. The decision about who will be able to read or write on an attribute will be taken by the server and it could even need some type of encryption for this task. Anyhow, there is taken into account the difference between a no-authorized, an unauthorized and authorized encryption. In the case of the unauthorized mode there is, anyway, need to have an encryption but there is no need to authenticate the key.

When a specific function or feature is needed then the GAT service will combine sets of attributes that will define them. There are two types of services, primary or secondary and they will be defined by the type of a field of an attribute. The difference between the primary service and the secondary ones is that the first group will describe the main function of a device, while the second group will define some additional features. They are identified by two UUIDs, the first one is 0x2800 and the second one 0x2801. The value field will contain a 16-bit UUID for standard services or a 128-bit UUID for custom implementation. The service declaration attributes are only readable on do not need any authentication or authorization.

Once a service is defined characteristics can be add to it starting with a Characteristic attribute. It functions as a sort of header to a set of attributes. Its value field includes properties, the value attribute

handle and an UUID. The permission should be read only and without access restrictions. The value attribute handle in the value field refers to the position of the attribute containing the actual value itself. Usually that is the next attribute. Every characteristic has a set of properties defining its usage. The ones of interest to this work are explained in table 3.
A characteristic attribute would be the first thing to add when adding characteristics to a service that has been define. It will be a kind of header for a set of attributes, its value field would contain properties, the attribute handle value and an UUID. It will have read only permissions without any access restraints. The lastly mentioned attribute handle value contains the position of the attribute that contains the real value. The most used and important GATT characteristics are shown in table 3.

Properties	Value	Description
Broadcast	`0x01`	If set, permits broadcasting the Characteristic Value using Server Characteristic Configuration Descriptor. If set, the *Server Characteristic Configuration Descriptor* should exist.
Read	`0x02`	If set, permits reading the Characteristic Value
Write	`0x08`	If set, permits writing the Characteristic Value with response
Notify	`0x10`	If set, permits notifications of a Characteristic Value without acknowledgement. If set, the *Client Characteristic Configuration Descriptor* should exist.
Indicate	`0x20`	If set, permits indications of a Characteristic Value with acknowledgement. If set, the *Client Characteristic Configuration Descriptor* should exist.

Table 3. Some of the GATT characteristic properties [24].

It is possible to add other attribute to a characteristic to have more information with respect to it. For instance, it will be possible to establish a format attribute to describe the type of the data of the value. It will be formed by a format, size of the exponent, unit, name and description. The standard defines the format types as well as most of the common data types, such as integers and floats of various sizes. However, if general purpose data has to be transmitted it is not mandatory to utilize these attributes. The Characteristic User Description is another attribute that would be useful for custom profiles, it contains a string describing the characteristic. Instead, the Client and Server Characteristic Configuration Descriptor are mandatory, if the characteristic has some specific properties. In table 4 it is shown an example service.

Handle	Type	Value	Permission
`0x0022`	Primary Service	Thermometer Humidity Service	read
`0x0023`	Characteristic	{`0x02`, `0x0024`, "Temperature"}	read
`0x0024`	Temperature	`0x028A`	read
`0x0025`	Characteristic Format	{`0x0E`, `0xFE`, "degrees Celsius", `0x01`, "Outside"}	read
`0x0026`	Characteristic User Description	"Outside Temperature"	read
`0x0027`	Characteristic	{`0x02`, `0x0028`, "Relative Humidity"}	read
`0x0028`	Relative Humidity	`0x27`	read
`0x0029`	Characteristic Format	{`0x04`, `0x00`, "Percent", "Bluetooth SIG", "Outside"}	read
`0x0030`	Characteristic User Description	"Outside Relative Humidity"	read

Table 4. Example for GATT service [24].

It is important that a GATT implementation uses a set of procedures and features for the communication between server and client. Table 6 shows and overview of them. In addition, the table also shows if the implementation of the function is either mandatory (M) or optional (O), whether for the client or server side. There is a different complexity between each of the procedures.

Feature	Procedure	Client	Server
Server Configuration	Exchange MTU	O	O
Primary Service Discovery	Discover All Primary Services	O	M
	Discover Primary Services By Service UUID	O	M
Relationship Discovery	Find Included Services	O	M
Characteristic Discovery	Discover All Characteristic of a Service	O	M
	Discover Characteristic by UUID	O	M
Characteristic Descriptor	Discover All Characteristic Descriptors	O	M
Characteristic Value Read	Read Characteristic Value	O	M
	Read Using Characteristic UUID	O	M
	Read Long Characteristic Values	O	O
	Read Multiple Characteristic Values	O	O
Characteristic Value Write	Write Without Response	O	C[7]
	Signed Write Without Response	O	O
	Write Characteristic Value	O	C[8]
	Write Long Characteristic Values	O	O
	Characteristic Value Reliable Writes	O	O
Characteristic Value Notification	Notifications	O	O
Characteristic Value Indication	Indications	M	C3[9]
Characteristic Descriptor	Read Characteristic Descriptors	O	O
	Read Long Characteristic Descriptors	O	O
Characteristic Descriptor Value Write	Write Characteristic Descriptors	O	O
	Write Long Characteristic Descriptors	O	O

Table 5. Overview of procedures needed to access GATT attributes [24].

Via the fixed L2CAP logical link channel, already established, the GATT attributes would be exchanged between server and client. It is important that the minimum size of the ATT MTU, for server and clients, would be of 23 B to assure the interoperability between multiple devices. Every packet of the attribute protocol will know the beforehand fixed ID of the L2CAP logical link. It is as well important that there is best effort QoS method implemented by the ATT, as it forbids flushing of PDUs. A GAP procedure would be use to establish the link, and both devices, already connected, can cut out the link at any time.

3.3.2.7 Security

As shown in figure 10, Bluetooth Low Energy implements a Security Manager (SM) that is situated between the L2CAP and the GAP layers and it provides key distribution and pairing features to the devices through different methods. In a similar way to the GATT profile it sets a fixed link over L2CAP of 23B MTU, it also implements a best effort QoS and assures packet delivery. The Security Manager Protocol defines a set of instructions to take into account over this link, that should be used to access to the different features.

For Bluetooth to establish a trusted connection it has to pass through a process called Paring. The devices involved in this process have to exchange keys in order to allow an encrypted connection. These exchanged keys are going to maintain the security settings for the devices for future connections, without needing to pair again. To achieve pairing between devices there are three steps that can be performed in different ways that would provide different levels of security. Of course, it will depend on the BLE device, its capability for inter device interaction and its software to choose the methods to do so.

The first step on the paring process is based on the exchanged of the pairing features, between the devices interested in established a secure connection. It would start by the server asking for security to the client before allowing it to access to one of its attributes. However, it could be also started by the client that would send a paring request before getting any data communication. This paring request and then the reply that it would obtain are filled with the security features of the devices involved in this process. In the second step of the paring process both the devices will agreed on a method, according to their capabilities, to exchange over it their secrets. Is in this step that paring is going to be achieved using the paring protocol and one method between the LE legacy or LE Secure connection. At this point the data would be encrypted with the keys generated during this second step. Finally, the third step is just used in particular situations when additional keys need to be exchanged and distributed. In Figure 21 it is shown this paring steps on a schematic representation.

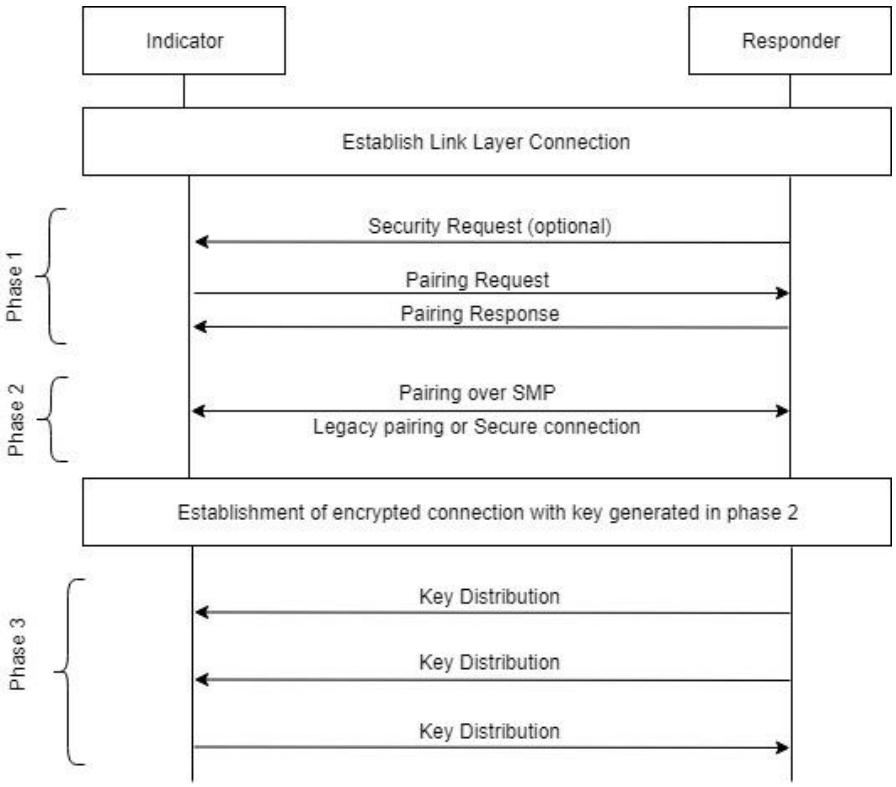

Figure 21. The BLE pairing phases [24].

In the original BLE standard there are three authentication methods that are part of the Secure Simple Paring (SSP) model, from where one of them should be chosen to be implemented on the device. This would be the previously mentioned LE Legacy paring. The devices interested on paring will define the method depending on their input/output capabilities. The methods available are the following ones:

Just Works: Used in devices without user interfaces (e.g. display or keyboard), so user interaction is not available. However, it supports a prompt to let the user known that the paring has taken place. There is not any kind of protection for man-in-the-middle (MITM).

Passkey Entry: This mode is used between a device provided of display and another one provided of keyboard, so, it allows the exchange of a 6-digit numeric code. For two devices that allow the use of the keyboard, this method allows both of them to share the same code. Given that this method has user input it gives MITM protection.

Out of Band (OOB): In this case it is possible to exchange information, to be used in the pairing process, by external means of communication (e.g. Near Field Communication (NFC)). It gives a certain level of MITM, depending on the OOB to be used.

Furthermore, the standard supports a fourth paring method given by the LE Secure Connections. This would further improve the security.

Numeric Comparison: In this method, both devices will be prompted with the same 6-digit key, but, at least one of them should have a button to be pressed if the keys match. If this process success, after the user has compared both keys and confirmed them in both devices, it will guarantee MITM protection.

To exchange these keys, this paring model allows the use of the Elliptical Curve Hellman-Diffie protocol. This protocol gives the possibility to exchange secrets over a non-secure channel, which could be used to generate a new key, or either to encrypt the communication.

Each of the hosts will generate the keys to be used in the BLE encryption. So, the algorithm that would create the key could be updated in every device separately. These generated keys are 128-bit long and their duration is variable. BLE needs to exchange the following secrets to establish a secure connection:

IRK: Identity Resolving Key, resolves the addresses of other devices, it consists in 128 bits. It is randomly generated at the manufacturing process.

CSRK: Connection Signature Resolving Key, signs the data and verifies the signatures in a connection, it consists in 128 bits. It can be generated randomly or set at the manufacturing process.

LTK: Long Term Key, it will generate a session key to have an encrypted connection, it consists in 128 bits. It is going to be created every time it will be sent.

EDIV: Encrypted Diversifier, while paring this value will verify the LTK, it consists in 16 bits. It's going to be generated each time an LTK is sent.

Rand: Random Number, it consists in 64 bits and it is going to be generated, as well, each time an LTK is sent, to identify it.

To generate a session secrete all keys between master and slave, of the LE Legacy paring, should be shared. Instead, only IRK and CSRK should be exchanged for LE Secure Connections. These keys are going to encrypt the data to be transmitted. It is going to be generated 128.bit encrypted information from the 128-bit plain text and the 128-bit key.

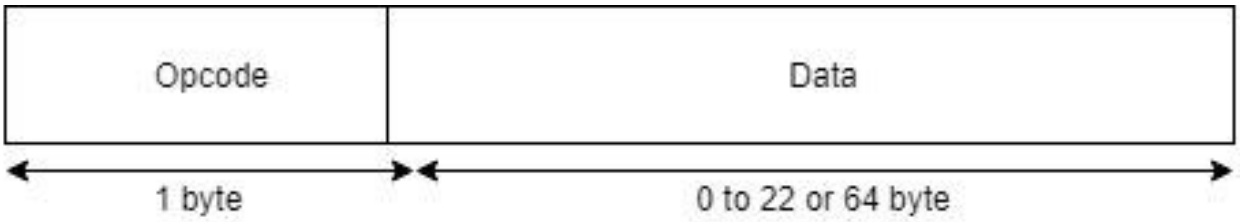

Figure 22. The BLE pairing phases [24].

Figure 22 shows a data packet sent by the Security Manager over the L2CAP channel. There are a group of commands that can be used in this protocol, and they are abbreviated by a code, table 6 shows a list of codes. There exist some Security Manager commands that will send not just the code but data, after the code. E.g. a Paring Request Packet includes information regarding the IO capability, an OOB data flag, the requested level of security, the size of the encryption key and two bytes that indicate the used key by the indicator and the responder.

Code	Description	Supported Logical Links
0x01	Pairing Request	LE-U, ACL-U
0x02	Pairing Response	LE-U, ACL-U
0x03	Pairing Confirm	LE-U
0x04	Pairing Random	LE-U
0x05	Pairing Failed	LE-U, ACL-U
0x06	Encryption Information	LE-U
0x07	Master Identification	LE-U
0x08	Identity Information	LE-U, ACL-U
0x09	Identity Address Information	LE-U, ACL-U
0x0A	Signing Information	LE-U, ACL-U
0x0B	Security Request	LE-U
0x0C	Pairing Public Key	LE-U
0x0D	Pairing DHKey Check	LE-U
0x0E	Pairing Keypress Notification	LE-U

Table 6. Overview of Security Manager Command code [24].

3.4 Middleware-based MQTT

3.4.1 Introduction

MQTT is a Client Server publish/subscribe messaging transport protocol [26]. It is a binary protocol and is very light weight, as it is simple and very easy to implement on the client side. Characteristics that excels when it transfers data over wire, in comparison to other protocols such as HTTP, given that its packet overhead is minimal. All of these are going to make it very productive in constrained contexts and devices with limited resources. Being all of the information concerning to MQTT portrayed as in the MQTT Version 3.1.1 OASIS Standard [26]

3.4.2 The publish/subscribe pattern

So alternatively, to the client-server model where a client communicates with and endpoint, it is presented a publish/subscribe (pub/sub) pattern. So, it is added the concept of Broker, an entity which is in the middle between publisher and subscriber, so they both don't need to know about the existence of the other one. After a client sends a message (called publisher) to another client (called subscriber) the broker will filter the messages to then distribute them in the right way.

The main feature of MQTT is the decoupling of publisher and subscriber, which could be achieved with different methods:

- **Space decoupling:** The address of the clients (e.g. ip address and port) can be unknown. So, Publisher and subscriber would not know each other directly.
- **Time decoupling:** It is not mandatory that clients, publisher and subscriber, run during the same period of time.
- **Synchronization decoupling:** Clients do not stop their current operations while publishing or receiving.

In short, the decoupling has three dimensions. Space, Time and Synchronization. A publish/subscribe pattern will decouple from a message the publisher and the subscriber. It is possible to filter the messages so only some particular clients would receive some particular messages.

3.4.2.1 Message Filtering

Message filtering deals with how is it going to be determined what published messages are going to arrive to each subscriber. There are mainly three options:

Subject-based filtering: Each message would have a subject or topic, and it is going to be the one to decide where the message should go. The receiver client should subscribe to the broker to the topics of its interest, so later on it would get all the messages published related to that specific topic.
Topics are a hierarchical structure based in general string.

Content-based filtering: As its name implies, the broker will filter the messages based on the content of the messages, by using a content filter-language. So, clients will subscribe to queries of messages

of their interest. A disadvantage of this method is that the content should be known in advance, so it could not be encrypted.
Type-based filtering: It is possible to filter the messages based on its type/class when using an object-oriented language. So, a client would listen to every message from an specific type.

3.4.3 MQTT publish/subscribe

MQTT embodies all of the mentioned aspects, depending on what you want to achieve with it. MQTT decouples the space of publisher and subscriber. So, they just have to know hostname/ip and port of the broker in order to publish/subscribe to messages. It also decouples from time, but often this is just a fall-back behaviour, because the use case mostly is to delivery message in near-realtime.

Previously it was explained how the publish/subscribe pattern works in general. Now it is going to be addressed particularly for MQTT.
MQTT use space decoupling, so for it needs the hostname/ip and port of the broker to be able to publish or subscribe to messages. It can also utilize time decoupling and synchronization decoupling, given that most of the client libraries are based on call-back and are asynchronous, so tasks would not be block while waiting to receive a message or to publish one. Given that it is possible to have synchronization and in some cases is useful to do so, some libraries should have synchronous APIs in order to wait for a particular message. However, the flow is mostly asynchronous.

On regard of the type of filtering used by MQTT it adopts the subject-based message filtering. So, the broker will use the topic of the messages to decide whether it should pass the message to a subscribing client or not.

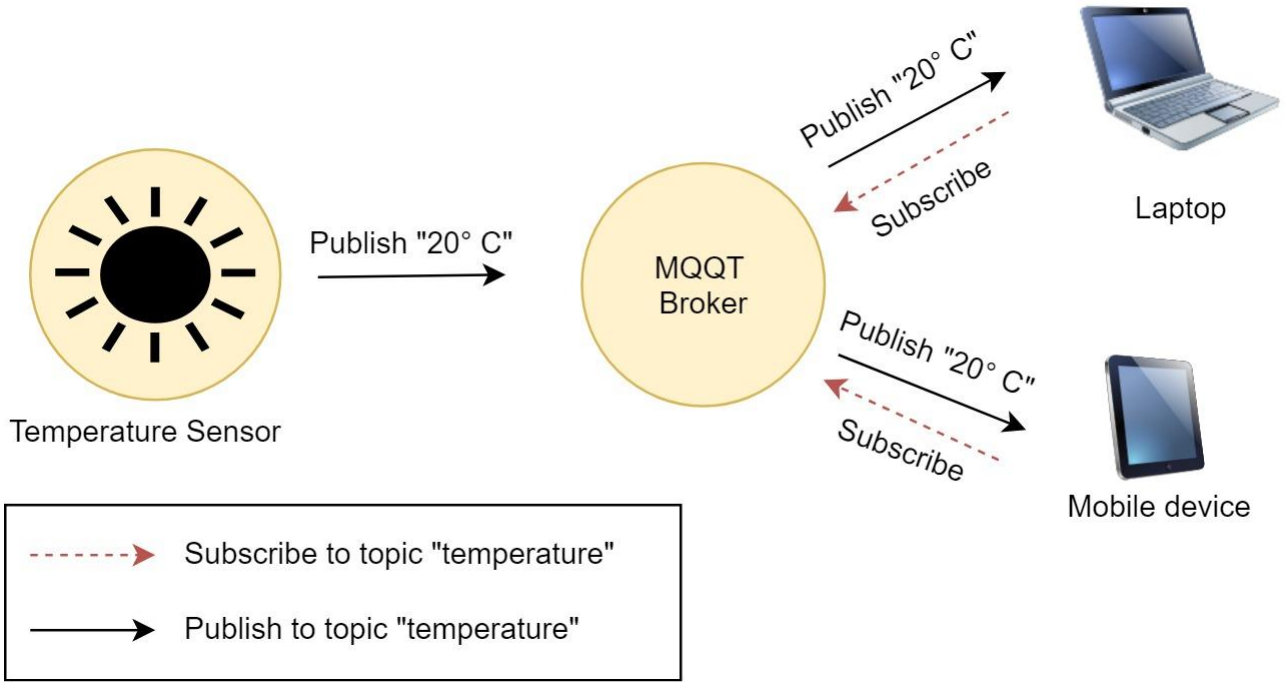

Figure 23. MQTT pub/sub example.

3.4.4 Client, Broker and Connection Establishment

3.4.4.1 Client

In MQTT there are two types of clients, as mentioned before, denominated publishers and subscribers, depending on if they will send messages or receive them. It is possible, that one device could perform both roles, though. For a MQTT client be defined as so it has to be any kind of device from a micro controller up to a server, but, that has a MQTT library running on it and it has to be connected to an MQTT broker located in the same network. For example, it could be a very small device with just a poor library with the minimum requirement or a full-equipped computer running an MQTT client, essentially any device that supports the TCP/IP stack and runs MQTT on it. MQTT is suitable for small devices given that the implementation of the protocol on a client is very essential. And it is possible to find the MQTT client libraries for most of the programming languages, such as C, C++, C#, Java, JavaScript, .NET, etc.

3.4.4.2 Broker

The MQTT broker is the centre of the publish/subscribe protocol system, and it is able to manage a very large number of connected MQTT clients. As mentioned before, the broker is the one running the filtering of the messages after receiving them from the publishers on its network, then it should decide which ones are the subscribers interested in these messages to afterwards send them to all the clients that it has found. It will hold the session of all persisted clients, even subscriptions and missed messages. The Broker is also responsible for the authentication and authorization of its clients. It is also possible to integrate custom authentications and authorizations into backend systems. The Broker is usually the component of the system that is going to be exposed in the internet, so integration is an important issue to be address to handle the large quantity of clients that it should and be able to pass the messages along to downstream analysing and processing systems. So, summarizing, The Broker is the central hub from where every message will pass through, so it should be highly scalable, integratable into backend systems, easy to monitor and failure-resistant.

3.4.4.3 Connection

The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack.

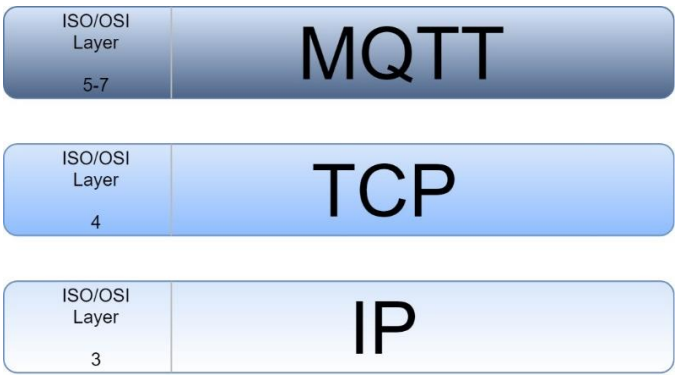

Figure 24. MQTT stack.

The MQTT connection will be stablished between the clients and the broker, this means that there is not any connection between clients. To begin the connection, process the client will send a CONNECT message to the broker, to which the broker will reply with an CONNACK and a status code. From there the connection will be established and to maintain it so the broker will keep it open until the client will send a disconnect command or will lose the connection.

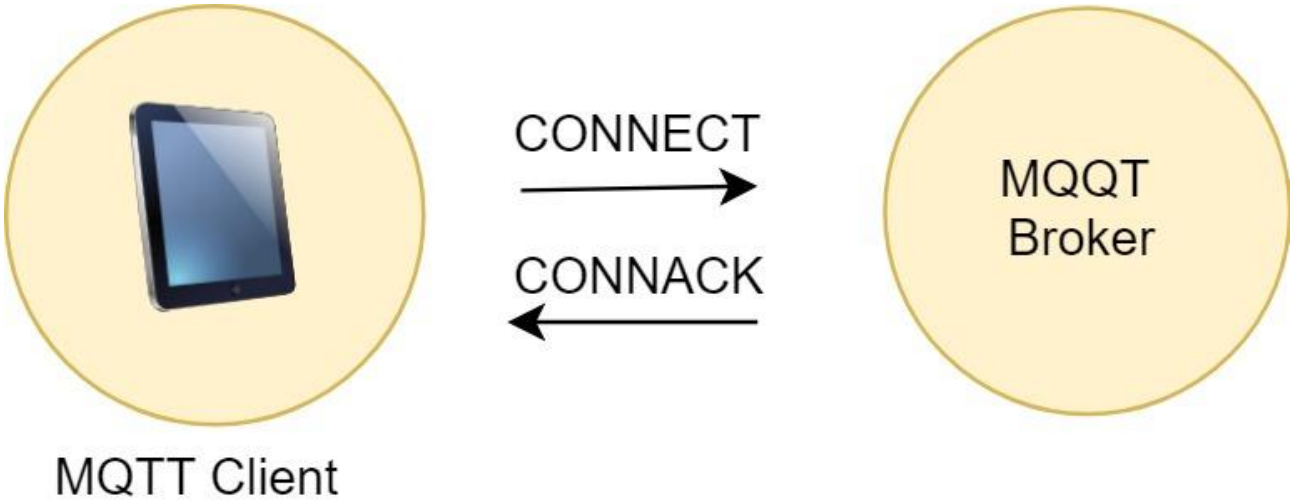

Figure 25. MQTT Connection.

3.4.4.4 Topics

As mentioned previously a topic is a string, a UTF-8 one, that will allow the broker to filter the messages for each client connected to its network. A topic could be formed by one or more level, and each one of these will be separate by a forward slash, called topic level separator.

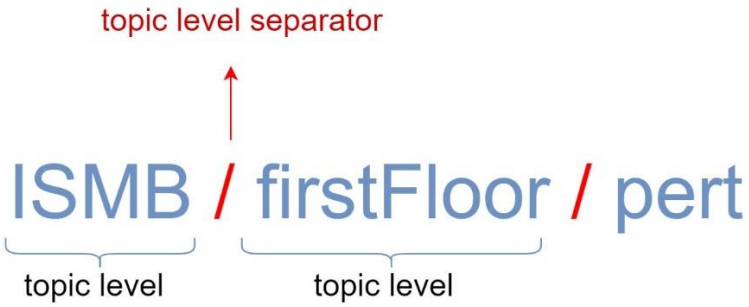

Figure 26. Example of MQTT topics.

The broker will accept any valid topic without any kind of prior initialization, so the client does not need to create the topic before publishing or subscribing to it. This makes the topic method very lightweight.

Chapter 4

Design of a Large Scale UWB-based Localization system

4.1 The Initial system and the proposed Solution

4.1.1 Initial Short-Scale Localization System

This thesis was meant to be designed and implemented on top of an already working Indoor Localization System, based on the UWB IEEE802.15.4a DecaWave[1] platform for RTLS.

Basically, the Short-Scale Localization system will be based on an independent indoor area in which four anchors are disposed, to range with up to four tags that are programmed to work with the parameters of that defined area. An estimation of each position is sent, through BLE, to a gateway in order to monitor in real time this application. This position estimation is going to be publish through an MQTT message.

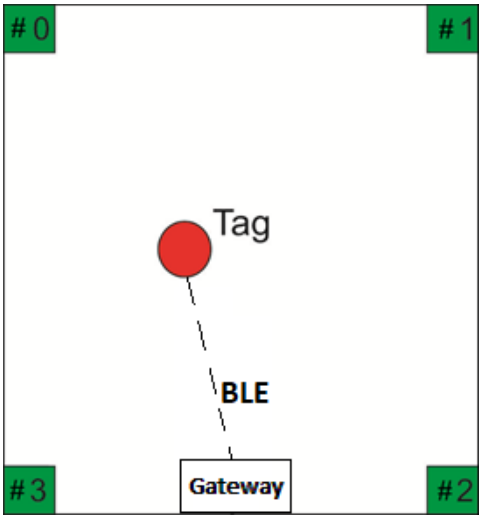

Figure 27. Short-Scale Localization System.

The system is based in a distributed approach, where it will be the tags to estimate their own position. The sequence for ranging is performed as usual, just that as soon as the tag finishes doing ranging with the anchors it will stop the ranging process to start the localization algorithm to estimate its position. At the same time this position is sent to the gateway.

The system was deployed twice, first for an indoor office environment of around 13 x 10 m^2 (Figure 28, red office), and second for another indoor office environment of around 12 x 10 m^2 (Figure 28, blue office) in which we can identify two totally independent systems. Each one consists in four anchors, positioned as shown in Figure 29.

[1] http://www.decawave.com/

These anchors were designed to work with the DW1000 DecaWave chip[2], that uses the IEEE802.15.4a standard, as reviewed in section 3.2 it has some important features, such us high accuracy, low power consumption, etc.

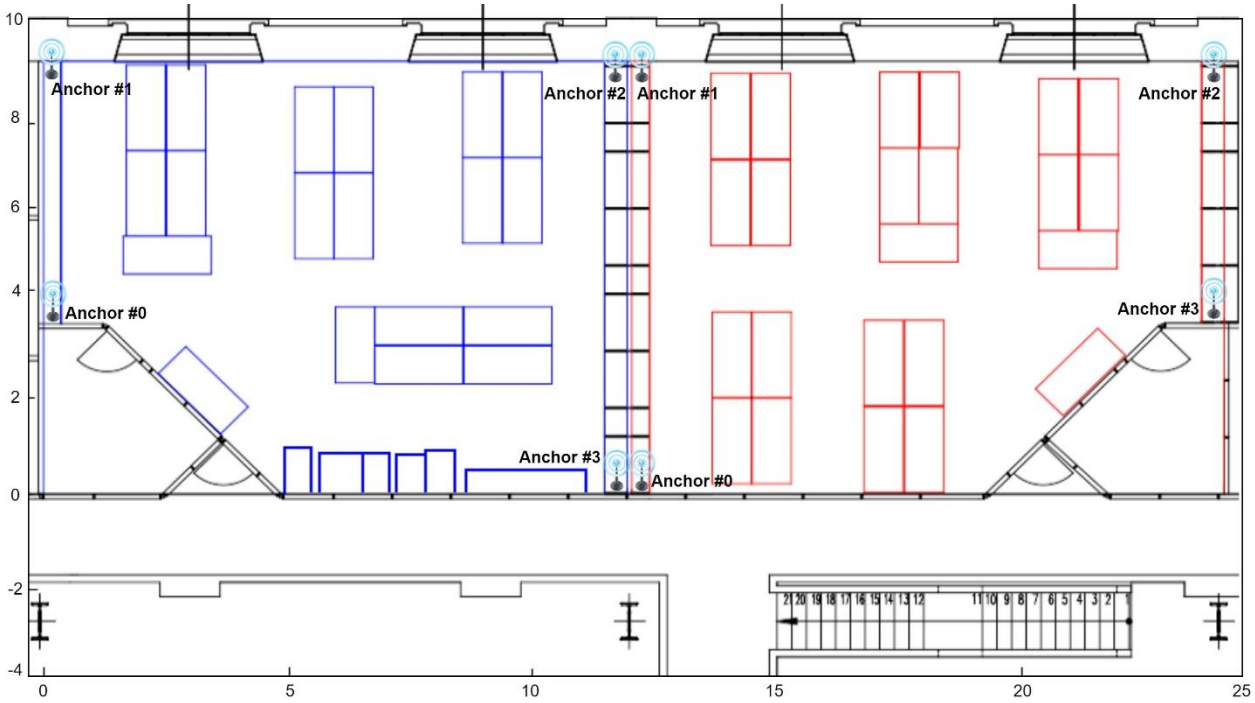

Figure 28. UWB initial systems.

As for the ranging process, the application will send and receive timestamps, with a method based in three different messages (poll, response and final) to make a two round trip measurement and calculate the range.

The system will define the Anchors as devices that will be always listening and waiting for messages coming from the Tag. It all begins with the tag sending a poll message from where the anchor will send a response to the tag, afterwards it will send back to the anchor a final message with the transmit and receive timestamps for the poll. Here, the anchor will use these values with the ones of its own to calculate the two round trip times, that have to be divided by two to get an average, so it is obtained the TOA. Later on, this value should be multiplied by the speed of light to lead to the estimation of the distance between the devices.
The anchor replies with a report message, in which it will transmit the measured distance to the tag, so in this way both devices would share this information that could be processed depending on the different applications. Shown in Figure 29.

[2] http://www.decawave.com/products/dwm1000-module

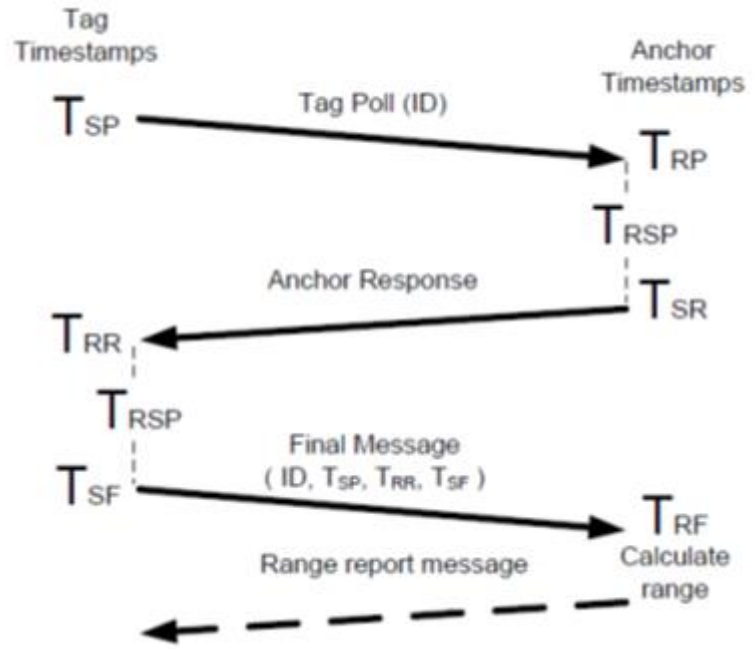

Figure 29. Range Calculation [28].

From Figure 29 we have T_{SP} the Send Poll timestamp, T_{RP} the Received Poll timestamp, T_{SR} the Send Response timestamp, T_{RR} the Received Response timestamp, T_{SF} the Send Final timestamp, and T_{RF} the Received Final timestamp.

But this is when a tag performs TWR individually with each anchor, instead it can broadcast a single Poll message that will be received by all the anchors, afterwards each one of them will send their own Response message, and finally, the tag will complete the ranging exchange by sending a single Final message that will be received by all the anchors. Each device will timestamp the transmission and reception times of every message.

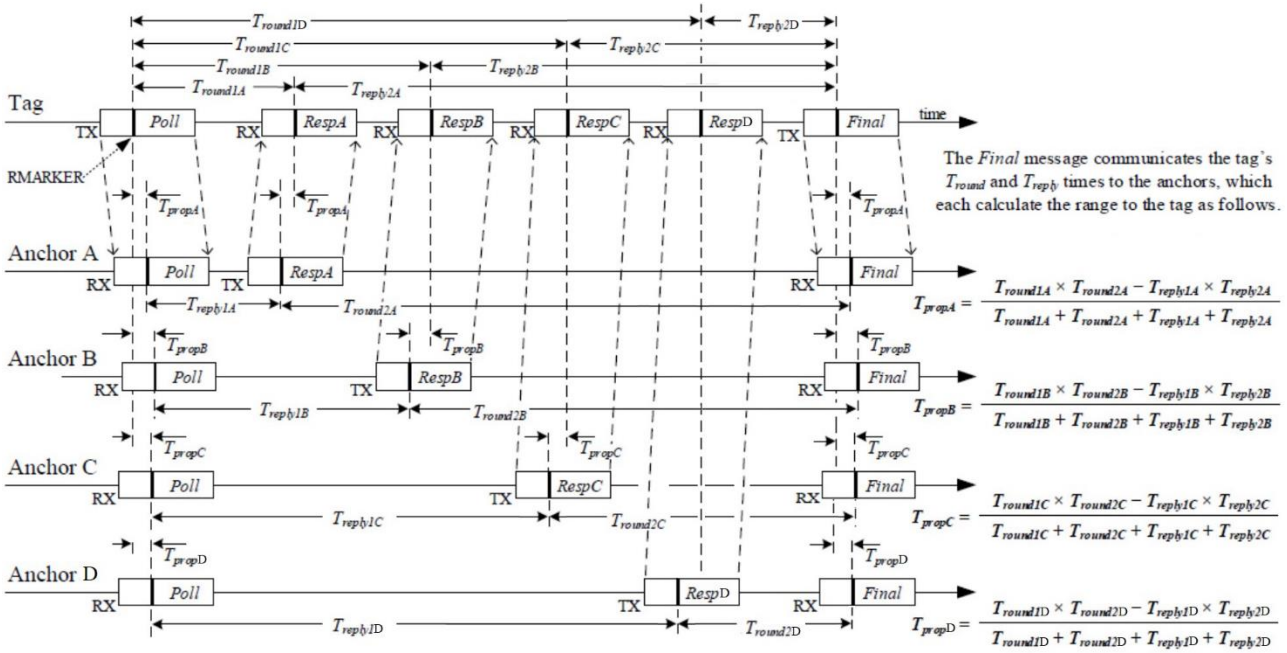

Figure 30. Two-way ranging to four anchors [17].

The configuration for the Anchors was set as follows: Channel 5 (6240 - 6739.2 MHz, centre frequency 6489.6 MHz) and channel 2 (3774 – 4243.2 MHz, centre frequency 3993.6 MHz), data rate of 110 kbps, preamble length of 1024 and clock frequency at 72 MHz.

About the tag network capacity for cluster, it will depend on the packet transmission rate, the length of the packet and the efficiency of the network protocol. By increasing the transmission rate the tag number supported would increase as well. Reducing the length of the packet will also increase it. However, the Short-scale system was designed to work with a maximum number of four tags [30], with ID number from 0 to 3, as shown in Figure 31.

As discussed previously in Section 2.5, these anchors have a coverage limitation, that is why in this 25x10m2 office there were deployed two clusters. Each one working in a different channel for the UWB system. The tags were programmed with a firmware with fix parameters (See Figure 31), such as the data rate mode, channel, unit number (number assigned to the tag for the cluster, from 0 to 3), preamble code (See Figure 32), the pulse repetition frequency (PRF), preamble length, etc.

```
/*
 * UWB Settings
 */
#define UWB_MODE            UWB_MODE_110K   /* UWB_MODE_6M8 or UWB_MODE_110K */
#define UWB_CHANNEL         UWB_CHANNEL_5   /* 2 or 5 */
#define UWB_UNIT            TAG             /* ANCHOR or TAG */
#define UWB_UNIT_NUMBER     0               /* 3-bit value (0-3 for Anchor) (0-7 for Tags) */
#define ROOM                PERT2           // see below for settings description

#define PERT2               0               // PerT 2 main room
#define PERT1               1               // PerT 1 main room
#define ORLANDO             2               // PerT 2 Orlando's desktop
#define CERTH               3               // CERTH / CPERI
#define SUNLIGHT            4               // Sunlight
#define COMAU               5               // Comau
#define COMAUT              6               // Comau test
#define SITI                7               // SiTi

```

Figure 31. Tag Firmware general parameters set on the tag.

```
//Configuration for DecaRangeRTLS TREK Modes (4 default use cases selected by the switch S1 [2,3] on EVB1000, indexed 0 to 3 )
chConfig_t chConfig[4] ={
        //mode 1 - S1: 2 off, 3 off
        {
                2,              // channel
                DWT_PRF_16M,    // prf
                DWT_BR_110K,    // datarate
                3,              // preambleCode
                DWT_PLEN_1024,  // preambleLength
                DWT_PAC32,      // pacSize
                1,      // non-standard SFD
                (1025 + 64 - 32) //SFD timeout
        },
        //mode 2 - S1: 2 on, 3 off
        {
                2,              // channel
                DWT_PRF_16M,    // prf
                DWT_BR_6M8,    // datarate
                4,             // preambleCode
                DWT_PLEN_128,   // preambleLength
                DWT_PAC8,       // pacSize
                0,      // non-standard SFD
                (129 + 8 - 8) //SFD timeout
        },
```

Figure 32. Tag firmware channel configuration.

And even the coordinates of the anchors with which the tag would have to performed ranging were pre-set (Figure 33). This means that the tags with these pre-set parameters would be able to operate just and exclusively in the cluster for which they were programmed, without the possibility of taking

those same tags and localize them in a different cluster, given that all, or most, of the parameters would not match with the ones of that another cluster, and this would make impossible to range with the anchors.

```
52  Coordinates coordinates[MAX_ANCHOR_LIST_SIZE]={ // Anchor coordinates
53
54          // ISMB -> Local {X,Y,Z} coordinates
55          // Pert 2
56          #if (ROOM == PERT2)
57                  {12.41,9.12,2.3},       //Anchor0
58                  {23.69,9.08,2.3},   //Anchor1
59                  {23.68,3.596,2.3},    //Anchor2
60                  {12.41,0.02,2.3}     //Anchor3
61          // Pert 1
62          #elif (ROOM == PERT1)
63                  {0.334,9.08,2.3},       //Anchor0
64                  {11.51,9.08,2.3},   //Anchor1
65                  {11.51,0.345,2.3},    //Anchor2
66                  {0.334,3.603,2.3}     //Anchor3
67
68          // Pert 2 - Orlando's Desktop
69          #elif (ROOM == ORLANDO)
70                  {14.44,1.992,0.83},     //Anchor0
71                  {14.44,0.2491,0.83},   //Anchor1
72                  {13.7,0.2491,0.83},    //Anchor2
73                  {13.7,1.992,0.83}     //Anchor3
74
75          // CERTH / CPERI
76          #elif (ROOM == CERTH)
77                  {32.425,11.1,3.05},     //Anchor0
78                  {32.395,0.15,3.23},   //Anchor1
79                  {20.7050,0,2.9},    //Anchor2
80                  {22.115,11.03,3.23}     //Anchor3
```

Figure 33. Segment of the definition of the anchor's coordinates set on the tag.

4.1.2 The proposed Large-Scale Localization System

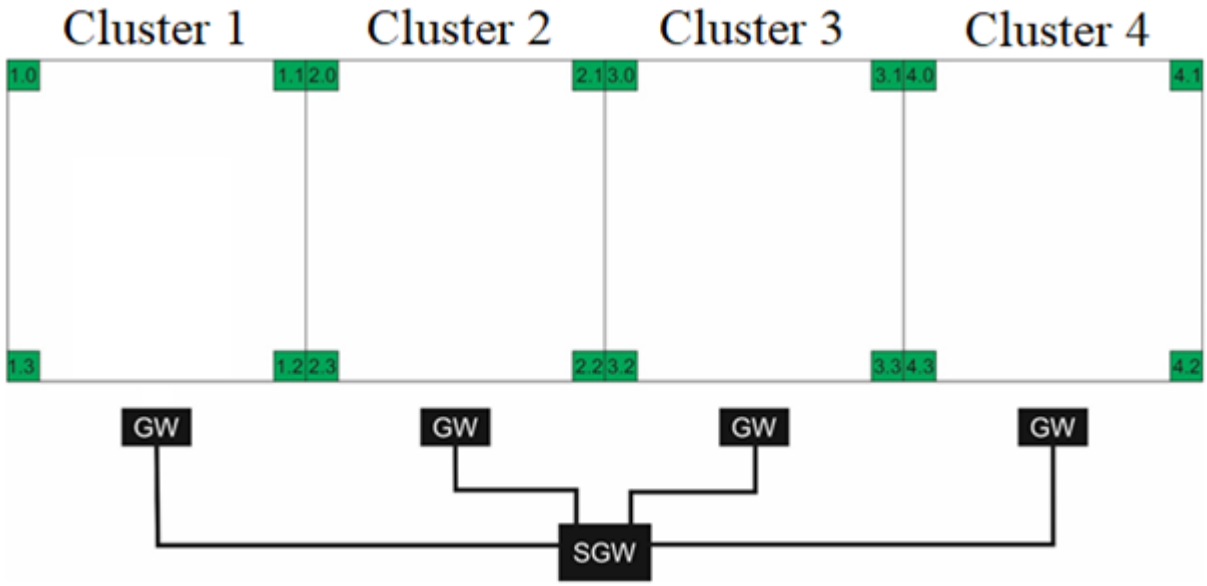

Figure 34. Large-Scale Localization Scenario.

The thesis proposal for the Large-Scale Localization System is based on the UWB system described in section 4.1. In particular, it is proposed a distributed and scalable system following a non-invasive approach with respect to the initial systems. Basically, it is proposed a system formed by many Clusters, as the one for the Short-Scale scenario, each one being configured almost identically but adopting different channel parameters in order to avoid the inter-channel interference. In this way the already existing firmware would be exploited as much as possible.

Figure 34 shows an approximated architecture for the deployment of the Large-Scale Localization System.

The aim is to add an entity that could allow the clusters to be part of a unique system. This means having an intelligence that could figure out when a new tag appears into the network and in which cluster it should perform ranging, it should be smart enough to handle a case in which an already existing and configured tag would change between one cluster to another, without losing it from sight.

According to the previous premises it has been added a Gateway in each of the clusters, able to communicate to the tags using BLE communication technology, to send them the initialization parameters that they need to be capable to perform ranging to the anchors of that specific cluster and be localized. As explained in section 4.1.1, these parameters were fixed in the tag's firmware when first programmed. So as soon as they would be turned on they would start raging with the anchors located nearby, without any kind of check or control of its parameters with the ones of the anchors in the vicinities. This is something that has to be changed. The tag should not perform any ranging before receiving, from the gateway, all the information corresponding to the cluster where it is located.

Afterwards all of these gateways will be able to share information somehow. Is due to this interaction between clusters that it is obtained a Large-Scale system. Aiming for this, it was added a Super Gateway, that would have the function to process all the information regarding to the whole system. Both, gateway and super gateway will be explained with more detail in the following sections.

4.2 Design of the system architecture

4.2.1 Functional Architecture

The first thing that was planned for the design of the system was the Functional Architecture (See Figure 35). It will describe how the system should function, the interaction between the UWB system, the gateways (GW) and the super gateway (SGW), and even more what kind of services or functions would operate in every part of the system.

The UWB-based Localization System is formed by the Anchors and the Tags. As described previously, they will be exchanging ranging messages between them to let the tags estimate their position based in a coordinates system.

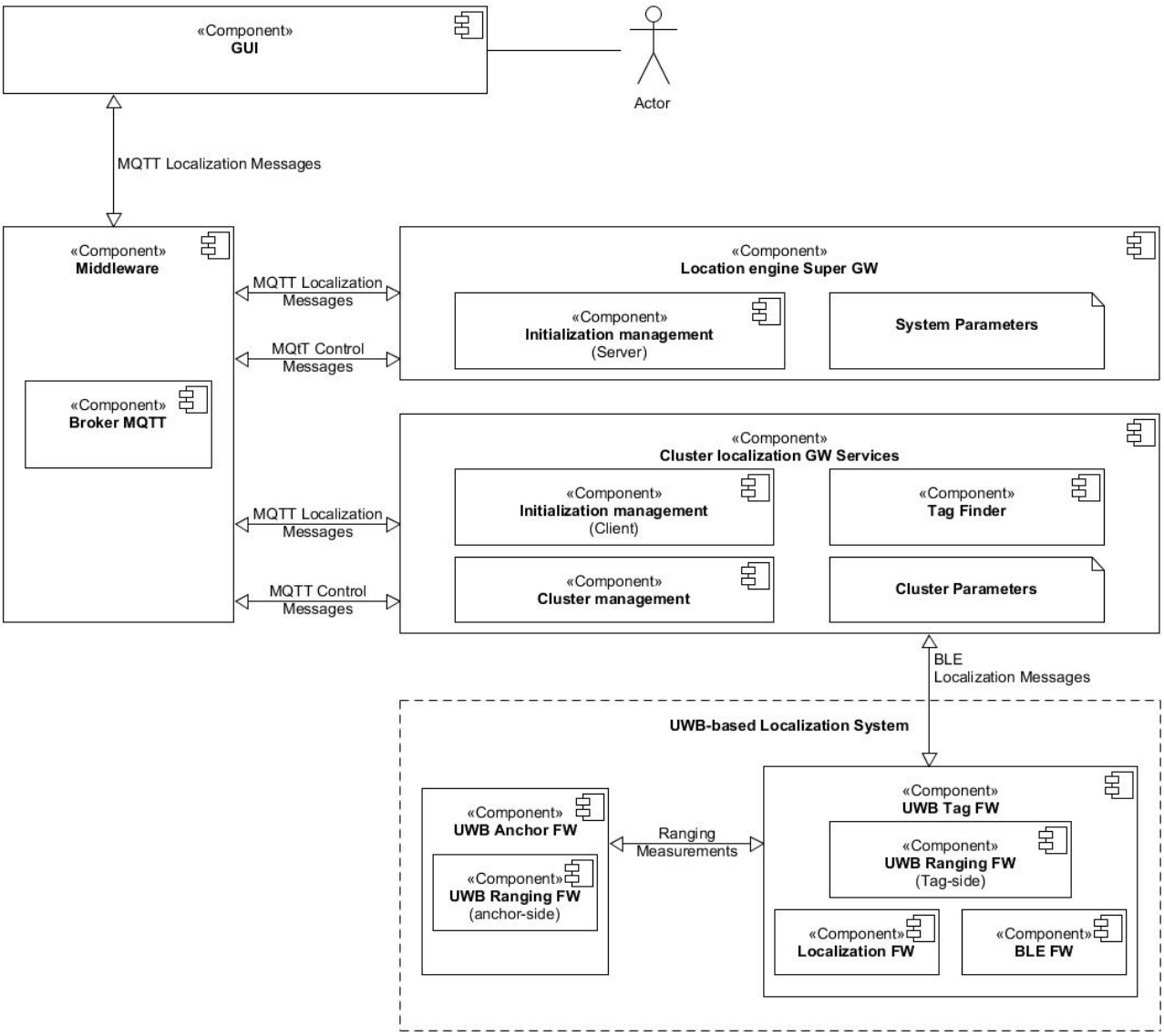

Figure 35. Large-Scale system Functional Architecture.

The Anchors contains the UWB Anchor Firmware (FW) that will not be modified, given that the UWB positioning will be used as it already is. Instead, The UWB Tag FW will contain the Ranging FW and the Localization system as they already were, The Bluetooth Low Energy Firmware (BLE FW) will be modified to fit the requirements of the new network. BLE is the communication technology that was chosen for the communication between Tags and GWs.

Between the Tags and the GW and then the SGW there is going to be a big load flow of messages.

For the case when a new Tag awakes within the system, the flow will start with the *Discovery messages*. The Tag Finder service will get from the tag its mac address as well as an *RSSI measurement*, that will be then pass to the SGW, through the MQTT middleware, to the Initialization management (server) service. If the SGW would get more than one *RSSI measurement*, coming from different GWs, of the same tag mac address, it will compare them to then decide which of these values, coming from what GW, would be the biggest one, so the tag would get the parameters from the GW from where it got the highest *RSSI measurement*. Otherwise, if the SGW would receive just one measurement it will automatically send the parameters of the GW to the new discovered tag.

At this point, the *Initialization Messages* will be sent from the SGW to the GW, containing all the configuration parameters that the tag would need to range to the anchors, such as Channel, Preamble Code and Anchors' coordinates. Afterwards the *Initialization parameters* will be sent from the GW to the Tag.

Afterwards the Tag will send its *Location Data* to the Cluster GW, this last one will check if the estimated position corresponds to a valid location for that cluster area. If it does, the GW will send the *Location Data* to the SGW, letting it know that that tag mac address in inside and belongs to that clusters, so no further actions must be taken. Otherwise, if this *Location Data* would not be part of that cluster area, but of one of the adjacent clusters, the GW would send a *Switching Message* to the SGW, letting it know that the tag is located in a different cluster. With this information the SGW will send the new parameters to the right cluster, which would do the same towards the tag, assuring that it is now part of the right cluster.

In the case of an already existing tag that, by moving from one place to another, would change from one cluster to another then it would be the GW, due to the *Location Data* received from the tag, that will know that the tag is about to get from its area and enter into another one. Then, it would send to the SGW the *Switching messages*, again to let it know from where the tag is getting out and to where the tag is getting in. The SGW will send the right parameter by the *Initialization Messages* to the new GW.

Basically, the switching process it is the same when a tag is initialized for the first time but in a wrong cluster than when an already well-set tag is about to leave a cluster to enter into another one.

4.2.1.1 Cluster Localization GW Services

The Cluster GW will be the body in the middle between the tags and the SGW, so it will have to handle information from both sides, routing it towards the right place depending on the nature of the data. As mentioned, the communication towards the tags will be through BLE messages, but towards the GW it will be through the MQTT middleware, reviewed in section 3.4.

The services to be implemented in the GW are as follows:

- *Tag Finder*: This service will measure the RSSI of a tag, when it gets discovered within the cluster coverage region, and send it to the *Location engine Super GW*, through the MQTT middleware, together with its physical MAC address.

- *Cluster parameter*: JavaScript Object Notation (JSON) based dictionary where the fix coordinates for the cluster anchors will be set from the *Location engine Super GW*, after the initialization of the system, and then recover by the *Initialization management (client)*.

- *Initialization management (client)*: Service that will receive the initialization parameters from the *Initialization management (server)* and then assign them to the new discovered tag.

- *Cluster Management*: It will receive the Location Data from the tag and analyse it to determine if the location is part of the current cluster. If yes, it will be sent throughout the MQTT

middleware to the GUI for consultation. If no, then it will send a switching alert to the *Initialization management (server).* It could also create a switching alert when the tag enters into a switching zone between two different clusters, then wait for new location to identify if the tag has left the cluster.

4.2.1.2 Location engine Super GW Services

The SGW is the central intelligence of the system. It takes decisions such as what parameters (of what precise cluster) pass to a specific tag. This decision is made based on the received RSSI message from the GW about the tag discovering, or when switching between clusters. To do so it will rely in a service and a dictionary, as follows:

- *System parameter*: JSON based dictionary where the fix coordinates for all the system anchors will be set and save. Once the system is started these coordinates will be send to each cluster, through the *Initialization management (server)*.

- *Initialization management (server)*: Service at the Super Gateway side that is in charge of passing to the Gateway the cluster parameters. These parameters are saved in a JSON dictionary. It will, as well, send the initialization message for a new discovered tag, after deciding which is the best cluster to assign to this given tag. This task will be made throughout an analysis and comparison of the RSSI measurements received from the GWs. It will also handle the switching messages, sent by the GWs, to pass them the new parameters when changing from one cluster to another.

4.2.2 Deployment Architecture

As fort the deployment of the system it was designed and architecture (see Figure 36), that would specify the hardware components that will be deployed all along every cluster, to make the network operative.
The architectures is described in the next sub-sections. Starting from the bottom of Figure 36, the UWB-based location system, and continuing all the way up until the Location Engine Super Gateway.

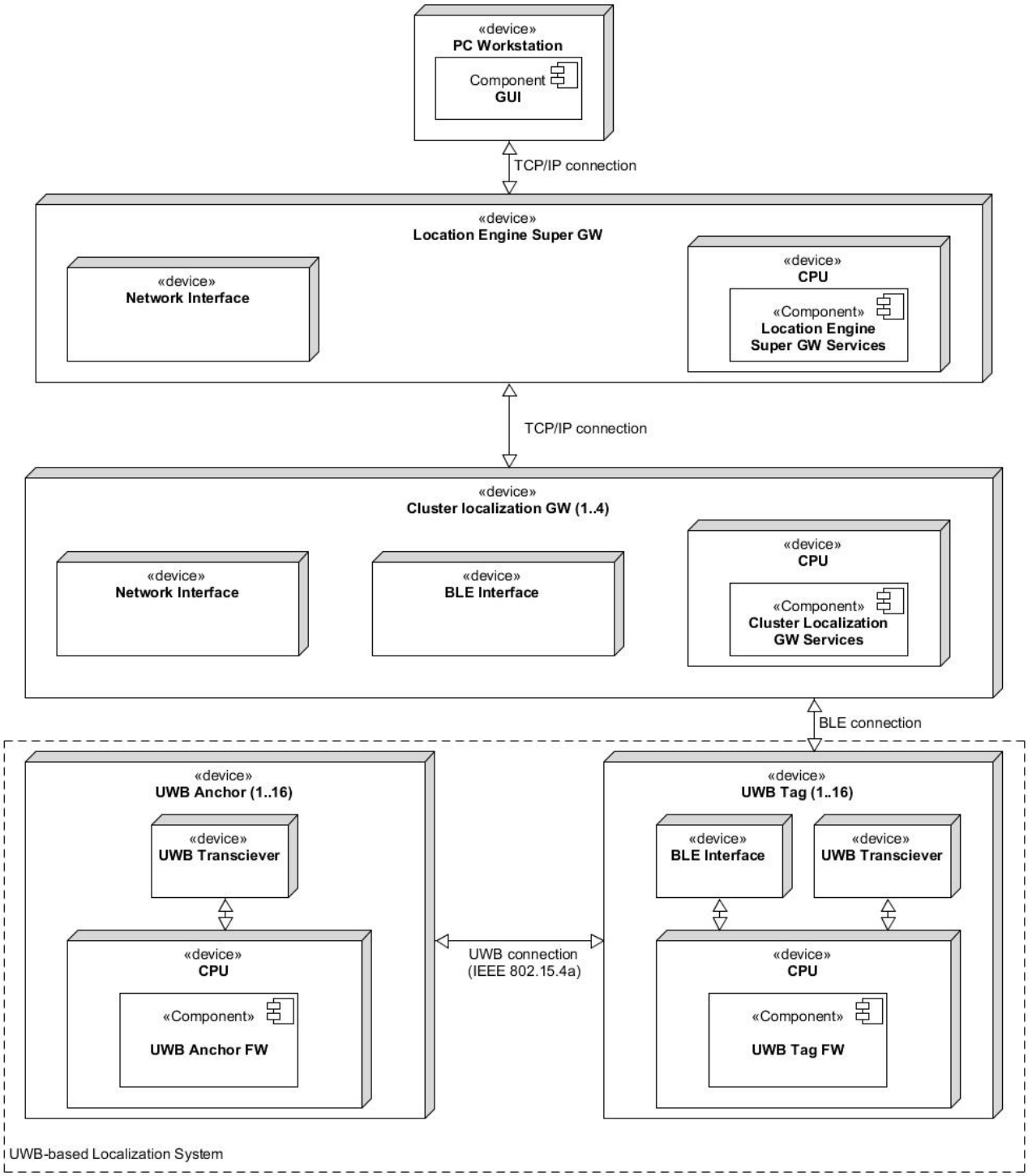

Figure 36. Large-Scale system Deployment Architecture.

4.2.2.1 UWB-based Localization System Deployment

The UWB-based Localization system is based in the anchors (Figure 37) and the tags (Figure 38) designed by the ISMB as prototypes for testing the UWB-based indoor RTLS. They both will be communicating through their UWB transceiver units, operating with the UWB IEEE802.15.4.a standard (See section 3.2).

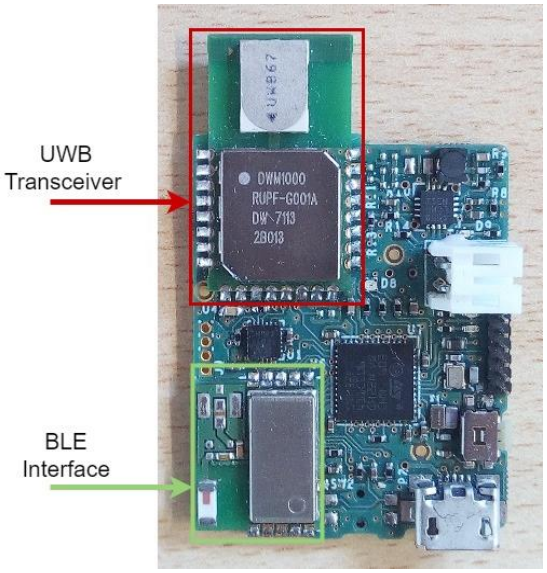

Figure 37. Custom ISMB Tag PCB.

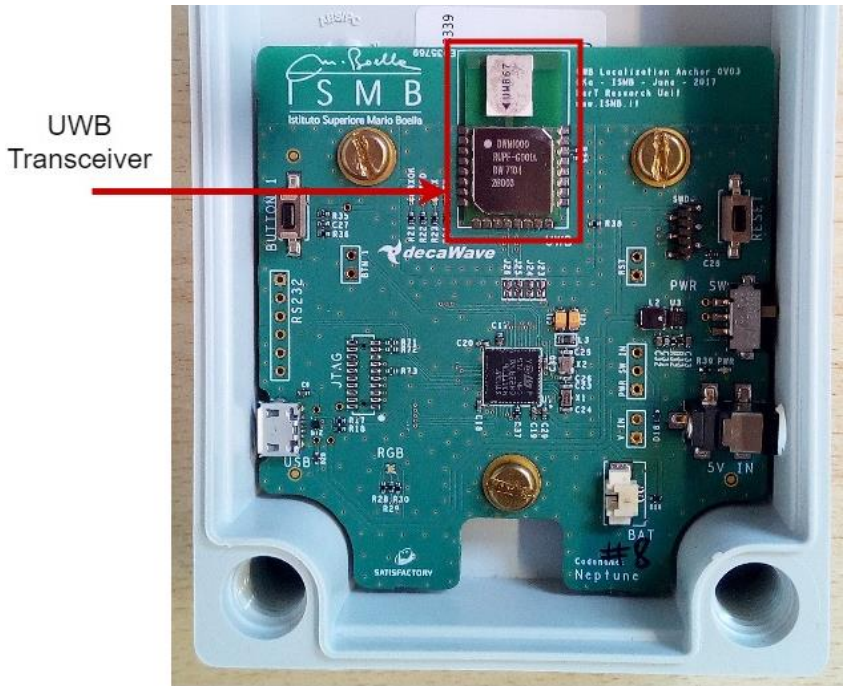

Figure 38. Custom ISMB Anchor PCB

4.2.2.2 Super Gateway and Cluster Gateway

The SGW and GW programs are set into the Raspberry Pi 3 Model B (Figure 39), which is a single-board computer, in which all the services, described in section 4.2.1, will run. The Raspberry that will function as a SGW will only need to communicate with the GWs using the TCP/IP Protocol, more specifically using its WLAN interface. The SGW, as well, generates the WLAN network to which all the other GWs are connected

As for the GW, it will communicate to the SGW, as mentioned before, but also to the tags throughout its BLE interface.

Figure 39. Raspberry Pi 3 Model B.

4.2.2.3 Graphic User Interface

The Graphic User Interface (GUI) will be a Phyton program based in the one created for the Short-Scale case, that is able to show how the tags move around their environment. It will have to be modified for the Lar-Scale case, to be able to show how a tag moves not just around one cluster, but also when it will switch between clusters, signalling appropriately for the Cluster in which it will be found, and its ID number.

Basically, the GUI can run in any PC able to run Python, with an active WIFI connection to the SGW to get the MQTT messages of the localization estimates.

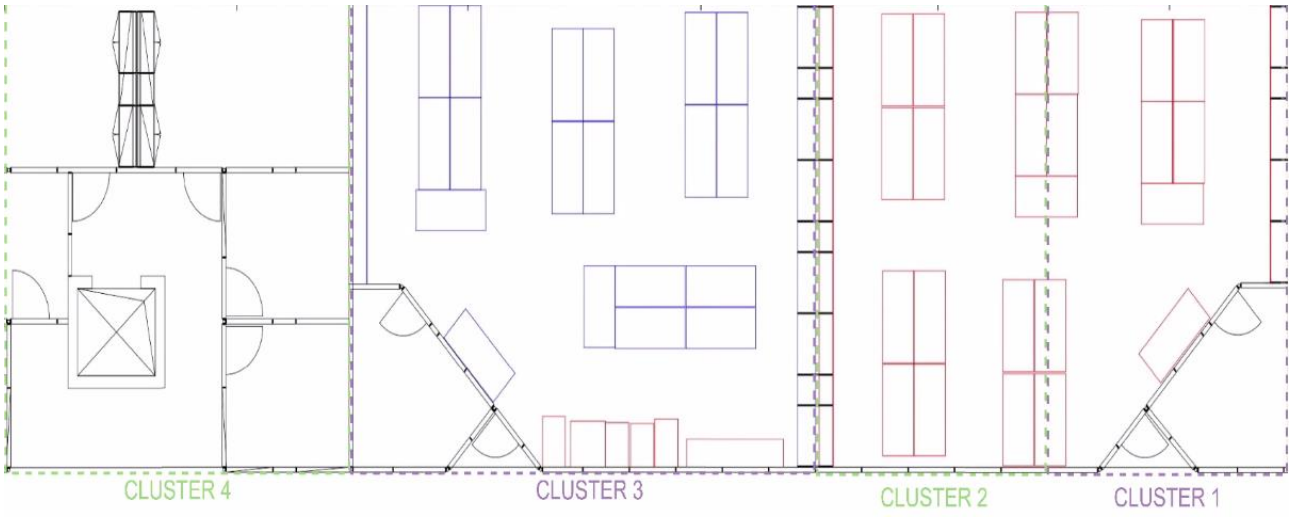

Figure 40. GUI for the RTLS in the ISMB

4.2.3 Information Flow

The Information Flow (Figure 41) shows all the data exchange that is going to be flowing inside the system, between every component that belongs to it.

Between the Anchors and the tags the only messages to be exchange are the *Ranging messages*, already presented in section 4.1.1. That have been already long discussed.

The following table shows the message that flows in the architecture, with their content, as well as a description.

FROM	TO	DATA	TYPE OF MESSAGE	NAME OF MESSAGE	DESCRIPTION
GW	Tag	Channel, Preamble Code, Anchor Coordinates	BLE	Initialization Parameters	Message that includes the parameter to initialize a tag in a given Cluster
Tag	Broadcast	MAC address, RSSI	BLE	Discovery message	Information that the tag broadcasts to be seeing enters into the system
Tag	GW	X position, Y position, EstimFlag, Tag id, MAC address	BLE	Location Data	Estimated position sent by the tag
GW	MQTT Broker	X position, Y position, MAC address, Cluster id	MQTT	Location Data	Sends the position estimated to the Broker
GW	MQTT Broker	MAC address, Cluster id, IN message, OUT message	MQTT	Switching messages	An alert is sent to the broken when a device gets into one zone cluster, or when a device gets out the cluster.
GW	MQTT	MAC address, Cluster ID	MQTT	RSSI measurement	GW sends the received RSSI measurement, during the discovery, to the SGW
MQTT	GW	MAC address, Channel, Preamble code, Tag id	MQTT	Initialization message	Send tag parameters to GW to get a tag set into a Cluster

MQTT	SGW	MAC address, Cluster ID	MQTT	Switching messages	An alert is sent to the SGW when a device gets into one zone cluster, or when a device gets out the zone cluster.
MQTT	SGW	MAC address, Cluster ID	MQTT	RSSI measurement	Broker sends the received RSSI measurement to the SGW
MQTT	SGW	X position, Y position, MAC address, Cluster id	MQTT	Location Data	Sends the position estimated to the SGW
SGW	MQTT	MAC address, Channel, Preamble code, Tag id	MQTT	Initialization message	Send the parameters to set a tag into a cluster to the broker
MQTT	GUI	X position, Y position, MAC address, Cluster id	MQTT	Location Data	Send the position estimated to the GUI

Table 7. Information flow messages.

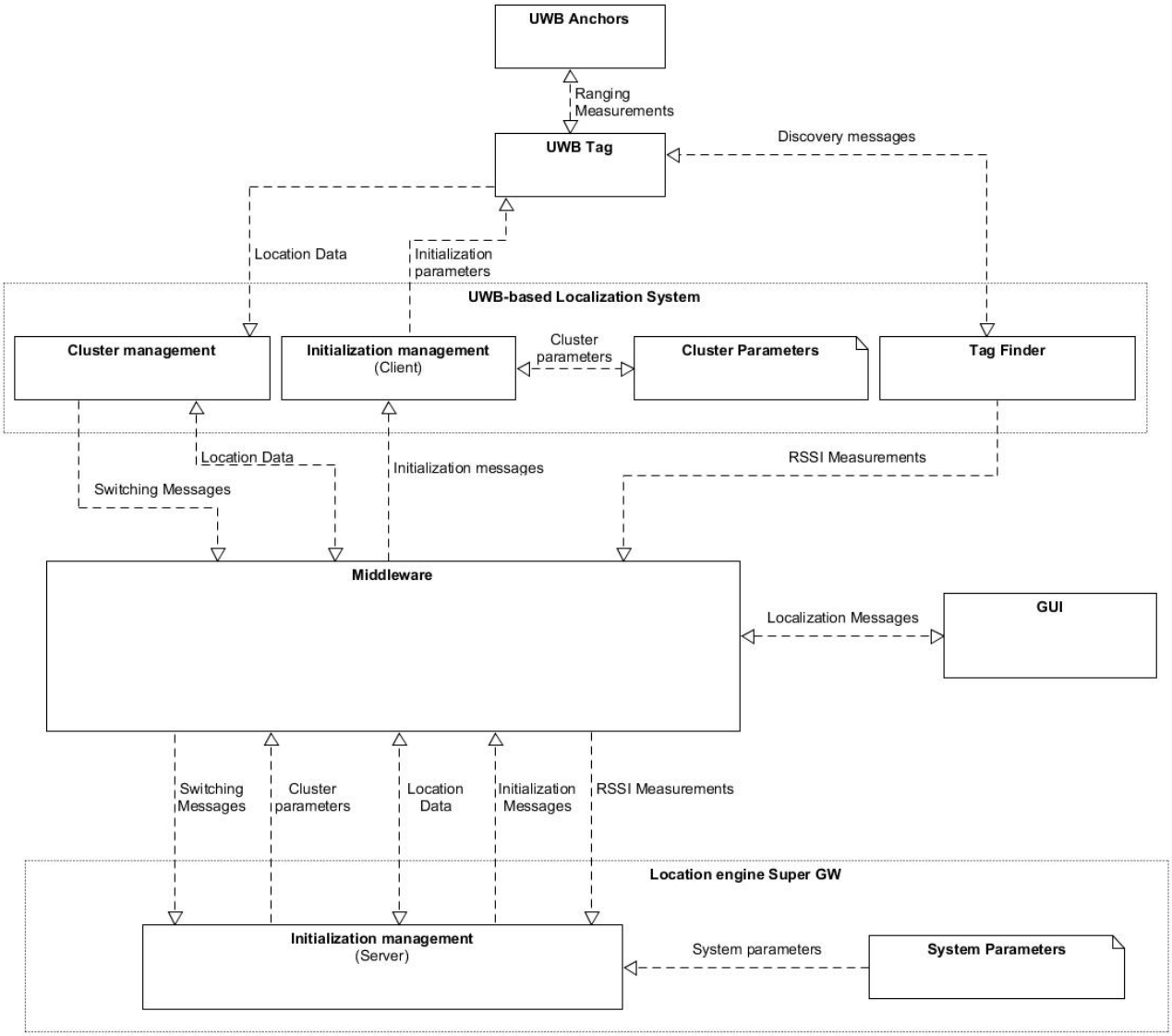

Figure 41. Large-Scale system Information Flow.

4.2.4 Sequence Diagrams

All of these previously mentioned processes are going to be better explained by defining the following four use cases.

4.2.4.1 Use case 1: Successful Initialization

This first use case (see Figure 42) defines the situation in which a new tag gets discovered for the first time inside the network, and how after being discovered it will get the right parameters to be able to range with the anchors of the cluster where it is truly located.

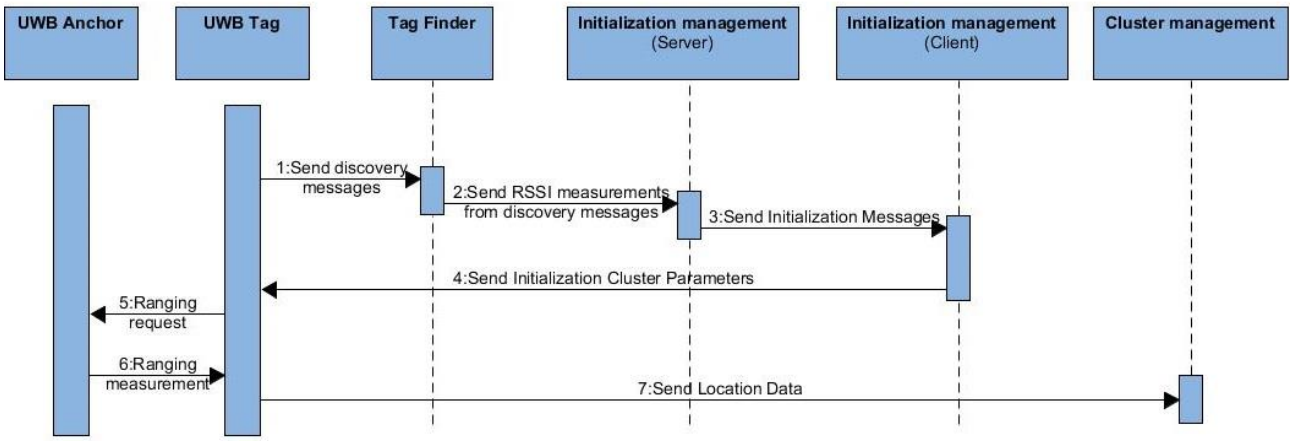

Figure 42. Use case 1: Successful Initialization.

4.2.4.2 Use case 2: Successful Initialization within a wrong cluster

The second use case (see Figure 43) would pick the previous one from the point were the tag gets for the first time a set of parameters, that happen to be the ones of the adjacent cluster and not the ones in which it is located currently. So, the GW will ask the SGW to switch the tag to the right cluster.

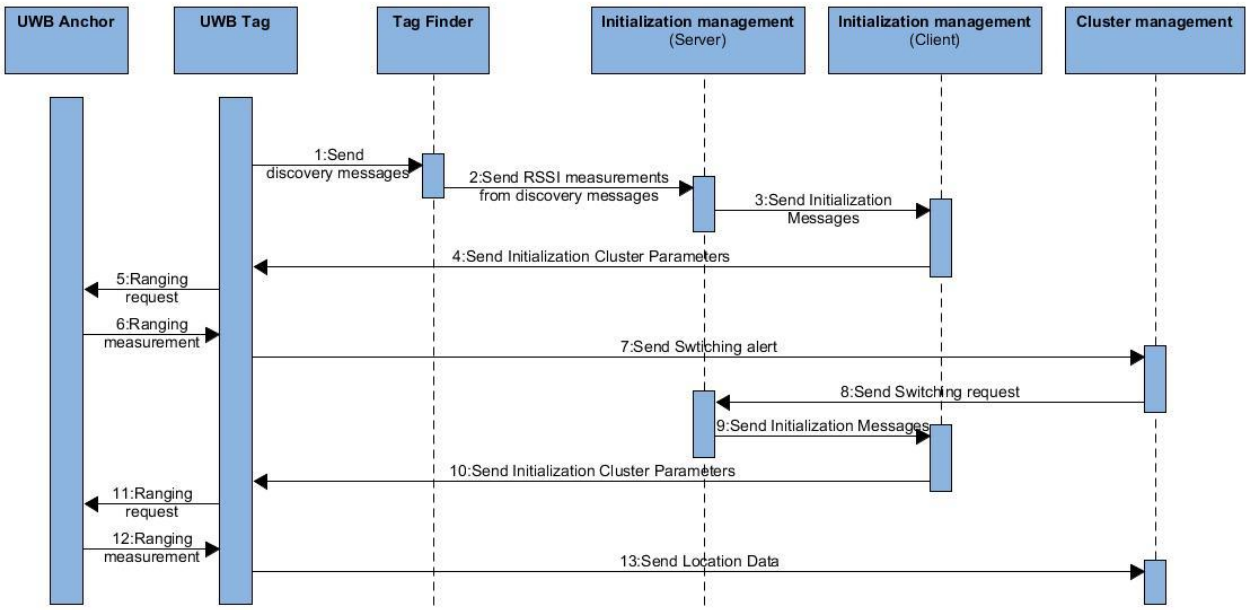

Figure 43. Use case 2: Successful Initialization within a wrong cluster.

4.2.4.3 Use Case 3: Unsuccessful Initialization with switching request

The third case (see Figure 44) shows how after a tag got the set of parameters for a determined cluster it is unable to perform ranging with the anchors, by any possible reason such as interferences, or maybe because it was assigned within the wrong cluster so it is out of the range of the anchors with which it should range. In this case a switching request would take place. The GW would request to the SGW to reassigned the tag, due to its failure at trying to get a localization estimate.

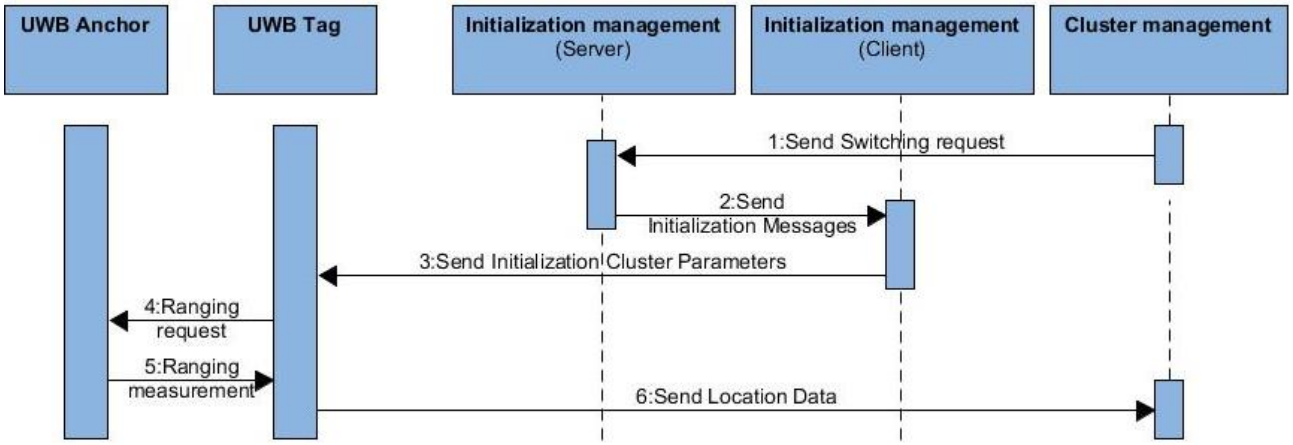

Figure 44. Use Case 3: Unsuccessful Initialization with switching request.

4.2.4.4 Use Case 4: Switching request from an already initialized tag

The fourth, and last, case (see Figure 45), as explained during section 4.2.3 will present a tag that after being working properly inside a cluster, would move towards its limits, to leave from it and enter into another one. In this case all the information about from where it is leaving and to where it is going in would be pass from the GW to the SGW, which then will handle the switching process, and send the new set of parameters for the tag to keep working properly in the new cluster.

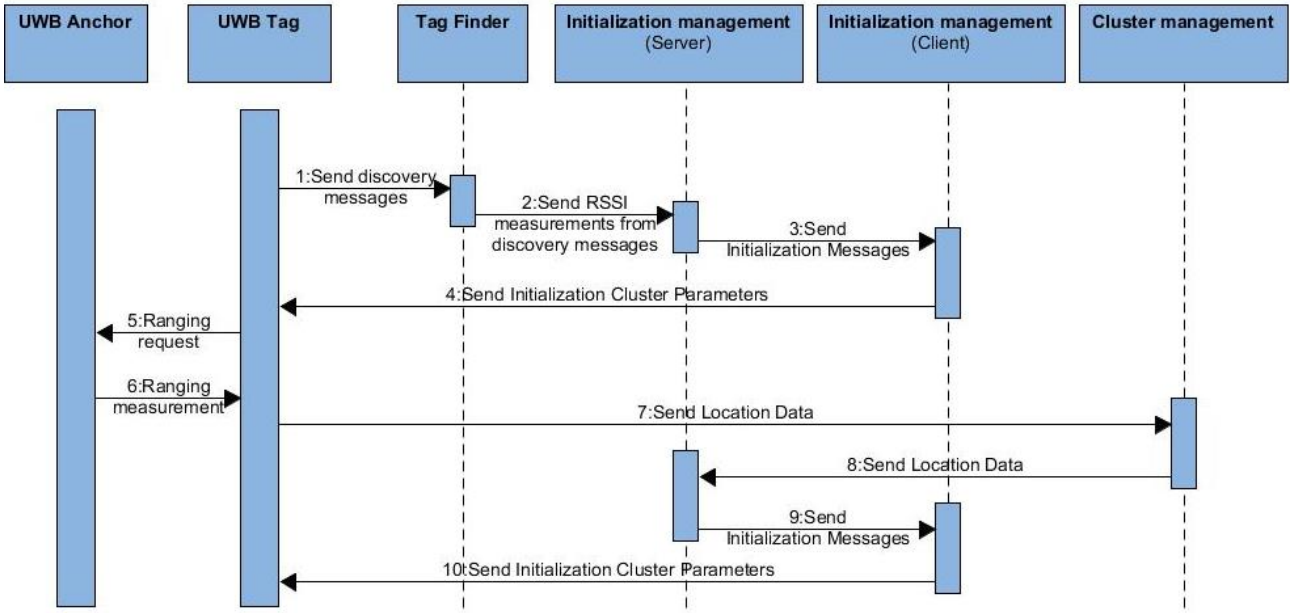

Figure 45. Use Case 4: Switching request from an already initialized tag.

Chapter 5

Implementation of a Large Scale UWB-based Localization system

5.1 Firmware modifications for UWB tag

The firmware (FW) utilized in the tag implementation was provided by ISMB. As explained in Chapter 4 the UWB RLTS was already implemented and working in the PerT offices, but just for single clusters independently.
So, the first part of the implementation for the Large-scale system was to modified the FW to adapt it to the proposed design in section 4.2.1.

5.1.1 Creation of BLE characteristics

The first step was to create inside a BLE service, called "`COPY_UWB_SERVOCE_UUID(uuid_struct)`", the characteristics to which the tag will receive, from the GW, the parameters to be update to set up in the cluster that was defined by the SGW.

There were created four characteristics. One denominated "`COPY_PREAMBLE_UUID(uuid_struct)`", for getting the preamble code and the tag's identifier (id). Then three characteristics in the form "`COPY_DISTANCH0_UUID(uuid_struct)`", for getting the coordinates of the anchors. In which the number after the word "DISTANCH" represents the Anchor id for which the coordinates correspond.

```
//Characteristics UUID
#define COPY_PREAMBLE_UUID(uuid_struct)         COPY_UUID_128(uuid_struct,0x02,0x36,0x6e,0x80, 0xcf,0x3a,
                                                              0x11,0xe1, 0xac,0x36,0x00,0x42,0xa5,0xd5,
                                                              0xc5,0x1b)
#define COPY_DISTANCH0_UUID(uuid_struct)        COPY_UUID_128(uuid_struct,0x02,0x36,0x6e,0x80, 0xcf,0x3a,
                                                              0x11,0xe1, 0xac,0x36,0x00,0x02,0xa5,0xd5,
                                                              0xc5,0x1b)
#define COPY_DISTANCH1_UUID(uuid_struct)        COPY_UUID_128(uuid_struct,0x02,0x36,0x6e,0x80, 0xcf,0x3a,
                                                              0x11,0xe1, 0xac,0x36, 0x00,0x02,0xa5,0xd5,
                                                              0xc5,0x2b)
#define COPY_DISTANCH2_UUID(uuid_struct)        COPY_UUID_128(uuid_struct,0x02,0x36,0x6e,0x80, 0xcf,0x3a,
                                                              0x11,0xe1, 0xac,0x36, 0x00,0x02,0xa5,0xd5,
                                                              0xc5,0x3b)
#define COPY_DISTANCH3_UUID(uuid_struct)        COPY_UUID_128(uuid_struct,0x02,0x36,0x6e,0x80, 0xcf,0x3a,
                                                              0x11,0xe1, 0xac,0x36, 0x00,0x02,0xa5,0xd5,
                                                              0xc5,0x4b)
```

5.1.2 Setting BLE characteristics

Afterwards it was added, through "`aci_gatt_add_serv`", the service "`COPY_UWB_SERVOCE_UUID(uuid)`" to the GATT Server.
Once this is done it was added, through "`aci_gatt_add_char`", the characteristics "`COPY_PREAMBLE_UUID(uuid)`", "`COPY_DISTANCH0_UUID(uuid)`", "`COPY_DISTANCH1_UUID(uuid)`", "`COPY_DISTANCH2_UUID(uuid)`", "`COPY_DISTANCH3_UUID(uuid)`" in which is defined as a property that they are readable and writeable characteristics. This mean that they can get, through a Bluetooth message, data to be written down.

```
#if LARGE_SCALE
tBleStatus Add_UWB_Service(void)
{
        tBleStatus ret;

        uint8_t uuid[16];

        COPY_UWB_SERVICE_UUID(uuid);
        ret = aci_gatt_add_serv(UUID_TYPE_128,  uuid, PRIMARY_SERVICE, 31,
                        &UWBServHandle);
        if (ret != BLE_STATUS_SUCCESS) goto fail;

        COPY_PREAMBLE_UUID(uuid);
        ret =  aci_gatt_add_char(UWBServHandle, UUID_TYPE_128, uuid, 3,
                        CHAR_PROP_NOTIFY|CHAR_PROP_WRITE|CHAR_PROP_READ,
                        ATTR_PERMISSION_NONE,
                        GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP,
                        16, 0, &preambleHandle);
        if (ret != BLE_STATUS_SUCCESS) goto fail;

        COPY_DISTANCH0_UUID(uuid);
        ret =  aci_gatt_add_char(UWBServHandle, UUID_TYPE_128, uuid, 9,
                        CHAR_PROP_NOTIFY|CHAR_PROP_WRITE|CHAR_PROP_READ,
                        ATTR_PERMISSION_NONE,
                        GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP,
                        16, 0, &distAnch0CharHandle);
        if (ret != BLE_STATUS_SUCCESS) goto fail;

        .
        .
        .

        return BLE_STATUS_SUCCESS;

        fail:
        //PRINTF("Error while adding ACC service.\n");
        return BLE_STATUS_ERROR ;
}
#endif
```

5.1.3 BLE update task

Subsequently, it was created a task to update the previously configured parameters on the tag's FW. This task will run first the command "`aci_gatt_read_handle_value`" will read the value that was written in the handle attribute "`preambleHandle`" of the characteristic "`COPY_PREAMBLE_UUID(uuid)`", save the newly received values for the Preamble Code and the tag id into a buffer called "BLEbuff[]" and then run an algorithm that will, first of all, compare the current values of the Preamble Code and the tag's id with the ones received through the BLE connection. The first time that the tag receives a BLE message it will compare the new values with the default ones, noticing that they are different, so it will initialize the ranging task by setting a flag bit called "BLEstart" from 0 to 1.
This flag was created given that the original FW made the tag initialize the raging process with the anchors as soon as it was turned on. This had to be changed for the new design in which the tag has to wait for a BLE message with the parameters to be set to then start doing ranging with the anchors of the cluster. From the second BLE message, and on, the "BLEstart" flag will be always 1 so it will pass directly to set the "BLEupdate" flag to 1, if the data received for the preamble code or the tag id is different from the one already saved.

Then, in a second instance the same command "`aci_gatt_read_handle_value`" will be used to get the data of the anchor's coordinates, through reading the value in the handle attribute "`distAnch0CharHandle`"

(using, again, the number "0" as a reference to the Anchor id from the characteristic "COPY_DISTANCH0_UUID(uuid)", and so on for the other three anchors.

```
#if LARGE_SCALE
void BLEupdateTask(void const * argument)
{
	UNUSED (argument);
	tBleStatus ret;
	osEvent  evt;
	uint16_t size = 3;
	uint16_t sizeAnchor = 9;
	uint8_t i;
	uint32_t cord;
	float32_t f;
	char myString[10];

	while(1)
	{
		if(1
#if SWITCHES
		&& is_group_active(UWB_bit)
#endif
	)
	{

	    while(BLE_STATUS_SUCCESS  !=  aci_gatt_read_handle_value(preambleHandle+  1,  size,  &size,
	        BLEbuff))
		{
			osDelay(10);
			Background();
		}

		if(BLEupdate == 0 && (BLEpc != BLEbuff[1] || BLEtag != BLEbuff[2]))
		{
			if (BLEstart == 0)
			{
				BLEstart = 1;
			}
			BLEupdate = 1;
		}

			while(BLE_STATUS_SUCCESS  !=  aci_gatt_read_handle_value(distAnch0CharHandle  +  1,
			    sizeAnchor, &sizeAnchor, BLEbuff0))
				{
					osDelay(10);
					Background();
				}

				if(BLEbuff0[0] == 0 && BLEstart ==1)
				{
					for(i=1; i<SIZE_ANCHOR; i=i+SIZE_COORDINATES)
					{
						sprintf(myString,"0x%02x%02x%02x%02x",BLEbuff0[i],
						    BLEbuff0[i+1],BLEbuff0[i+2], BLEbuff0[i+3]);
						sscanf(myString, "%x", &cord);
						f = *((float32_t*)&cord);

						if(i < 1+SIZE_COORDINATES)
							coordinates[0].x = f;
						else
							coordinates[0].y = f;
					}
				}

			.
			.
			.

		}
		osDelay(1000);
	}
}
#endif
```

5.2 Super Gateway and Gateway implementation

As described in section 4.2.1, Super Gateway and Gateway are two entities that will work co-dependently as the ones to manage the information flow inside one cluster (GW) as well as the whole Large-Scale system (SGW).
In this section are going to be explained the algorithms used for such purpose.

5.2.1 BLE device discovery

The first algorithm applied to the system is the one focused on the discovering of BLE devices in the proximities of the GW to then get its data to be sent to the SGW to let the initialization management task decide if the device has to get any parameters and of what cluster. As shown in Figure 46.

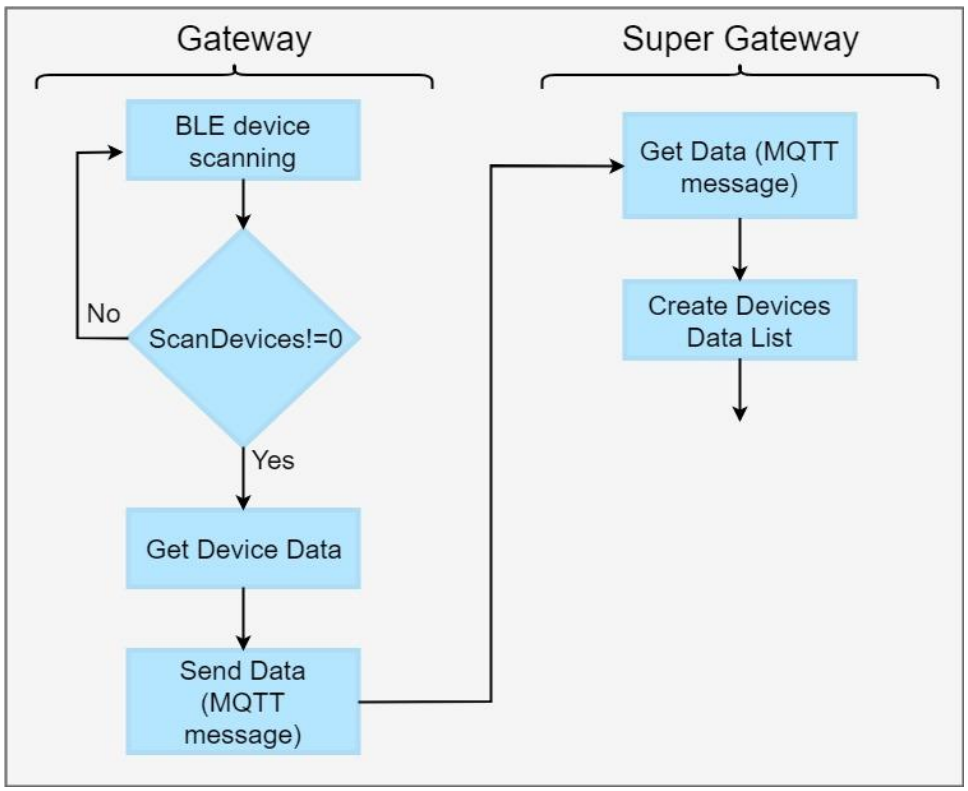

Figure 46. BLE Discovering.

The function "BLE device scanning" runs the command "`scanner.scan(scanningTime)`" that, given a predefined time, will periodically scan for BLE devices around the GW. When devices are found it will put them into a queue.

```
def scanBLE(ThreadName):
    global can_run,scanningTime
    scanner = Scanner().withDelegate(ScanDelegate())

    while can_run:
        try:
            #Scan for available BLE devices
            Ble_sem.acquire(True)
            devices = scanner.scan(scanningTime)
            Ble_sem.release()

            if devices:
                q.put(devices)
```

When the queue gets values in it the thread "`getMacRssi`" will first check, through their MAC address, if the devices are part of the system, in other words if they are DecaWave tags. To then, take from them their MAC address, and RSSI to be sent as a MQTT message to the SGW.

```
#Thread that discovers decawave devices and then sends MAC and RSSI to the SGW
def getMacRssi(ThreadName):

    global can_run

    while can_run:
        try:
            if not q.empty():

                devices = q.get()
                Ble_sem.acquire(True)
                for dev in devices:
                    if dev.addr[6:] == "de:ca:01:00": # UWB-based Wearable device
                        if dev.addr in dict:
                            pass

                        else:
                            dict.append(dev.addr)

                            msg_json = {}
                            msg_json["RSSI"] = dev.rssi
                            msg_json["Address"] = dev.addr
                            msg_string = json.dumps(msg_json)

                            pubData = "%s/data" %(current_config["GW_Name"])

                            mqttc.publish(pubData,msg_string, 0)
```

The SGW as soon as it receives the MQTT message with the tag's information it will put it into a queue.

```
     #Receives MAC and RSSI of discovered tags from the GWs
    if GW_topic == "data":
        print "ON MESSAGE %s:" %GW + GW_topic
        mqtt_data = (topic + ";" + payload)
        q.put(mqtt_data)
```

To then create a list with the data of all the possible MQTT messages sent by the GW's. Given that it is possible than one GW sends more than one message if there were more than one tag awakening in its cluster. Or that the same tag was seen by more than one GW, so the SGW will get two different messages with two possible different RSSI measurements of the same tag.

```
#Thread that gets MACs and RSSIs from the mqtt queue and puts them into a dictonary
def get_mac_rssi(ThreadName):

    global can_run
    global data_Li
    global elapsedtime
    global Tstart
    global thirdThread
    global thirdThread_stop
    global preamble

    data_Li = []

    Tstart = 0
    elapsedtime = 0
    counter = 0

    while can_run:

        try:
```

```
            if q.empty():

                if Tstart != 0:
                    elapsedtime = time.time() - Tstart

            else:

                if counter == 0:
                    Tstart = time.time()

                data_Li.append(q.get())

                counter = counter +1
```

From this list it will be created a dictionary with the form [GW#][MacAddress][RSSImeassure] that will be use in the next algorithm.

```
            if GW_topic not in dict["GW_info"].keys():
             dict["GW_info"][GW_topic] = {}
             dict["GW_info"][GW_topic][str(payload_json["Address"])] = {}
             dict["GW_info"][GW_topic][str(payload_json["Address"])]["RSSI"] = payload_json["RSSI"]
```

5.2.2 Cluster Assignment

Once the SGW has created a dictionary with the information received from one or more GW's it will run an algorithm aimed to define the Cluster in which the tag is located and from which will have to get the parameters to range to those specific anchors.

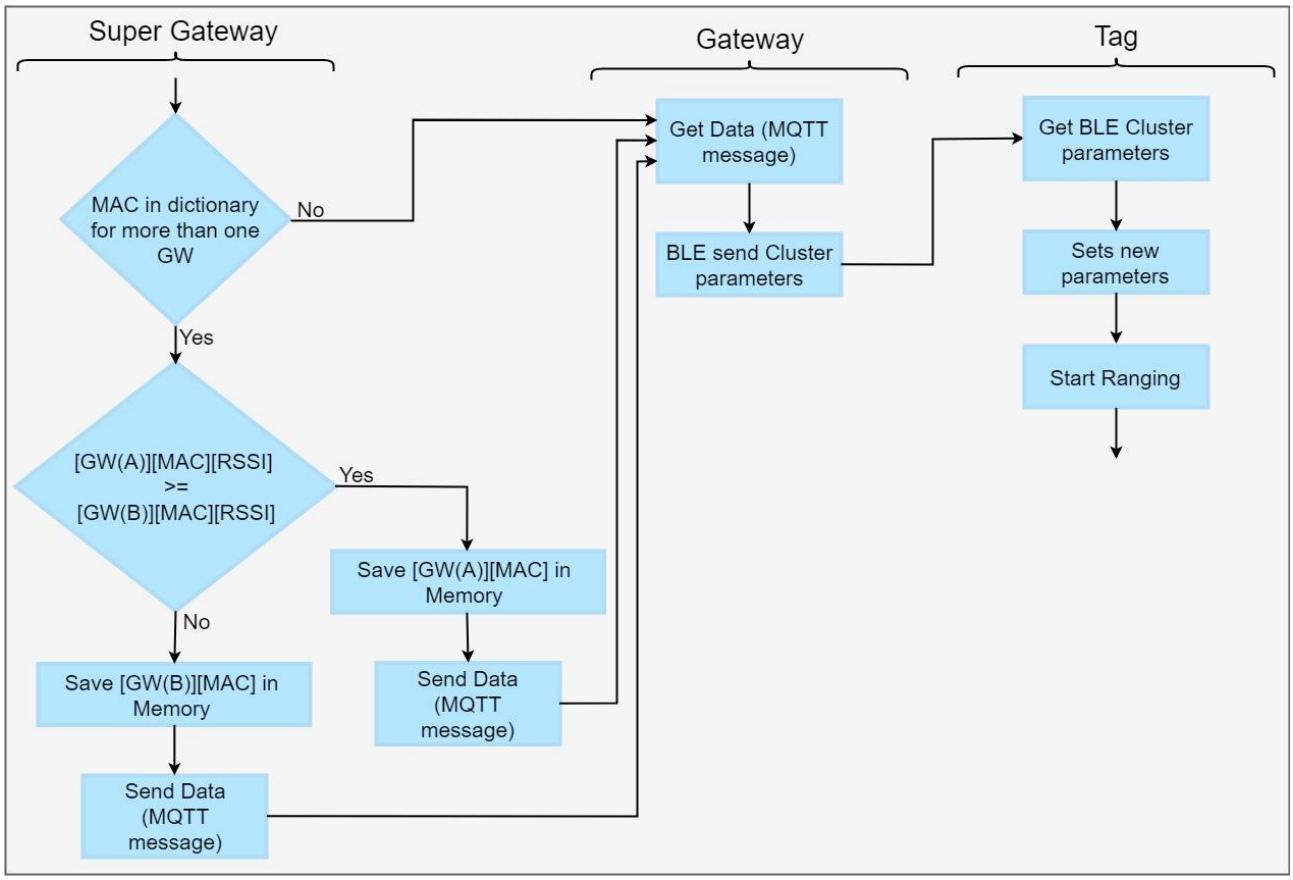

Figure 47. Cluster Assignment algorithm.

The first step is to check if it has received RSSI measurements of one same MAC from more than one GW.
If not, it will, firstly, set the Preamble Cod according to the GW from which the MAC proceeds. Later it will save the MAC address into its memory list, attached to the reference number of its GW.

```
if key2 == "GW2" or key2 == "GW4" or key2 == "GW6":
    preamCode = 03
else:
    preamCode = 04

if key2 in mem["Data"]:
    if macId not in mem["Data"][key2]:
        mem["Data"][key2].append(macId)
```

Otherwise, the algorithm will verify if for one device's Mac were received more than one RSSI measurements, to then compare them. As explained in section 2.2.1, the RSSI is the voltage estimated with which a receiver feels the transmitter device. Meaning that the GW that is closer to the tag will have a higher RSSI measurement, giving in this way a rough idea of in what cluster the tag is located. After the RSSI comparison the algorithm will set the Preamble Cod according to the GW from which the MAC proceeds. Later it will save the MAC address into its memory list, attached to the reference number of its GW.

```
if dict["GW_info"][key2][macId]["RSSI"] >=
   dict["GW_info"][key][macId]["RSSI"]:

    if key2 == "GW2" or key2 == "GW4" or key2 == "GW6":
        preamCode = 03
    else:
        preamCode = 04

    if key2 in mem["Data"]:
        if macId not in mem["Data"][key2]:
            mem["Data"][key2].append(macId)
```

At this point the Cluster in which the tag will be assigned is already defined. In either case, the function "SendData" will be called. In which first is going to be assigned a Tag id for a pool of four number (from 0 to 3), that will be memorized in a dictionary, linked to the MAC address and the Cluster's GW number.

```
for i in range(0,MaxTag):

   if search_list(idMem[valueGW], i) == None:
           idMem[valueGW][valueMac] = i
           idNumber = i
           break
```

After this, the parameters of Preamble Code and Tag id are defined. So, they will be sent to the Cluster's GW through a MQTT message.

```
Init_msg = {}
Init_msg["Mac"] = valueMac
Init_msg["Channel"] = 05
Init_msg["Preamble"] = valuePrCo
Init_msg["TagId"] = idNumber

msg_str = json.dumps(Init_msg)

pubData = "/SGW/%s/data" %(valueGW)
mqttc.publish(pubData, msg_str, 0)
```

When the GW receives the MQTT message with the MAC address of the tag to connect to, the Preamble Code and the Tag id it will called the function “SendParameters”. From where will send the parameters received by the SGW, as well as the anchors’ coordinates that will bi pulled from a json dictionary of its own, through a BLE connection by utilizing the “gatttool” commands.

```
        # Run gatttool interactively.
        print "________________________________________________________________\n"

         # print("Run gatttool...")
        connection_command = "gatttool -I -b %s" %mac
        child = pexpect.spawn(connection_command)


        # Connect to the device.
        print("Connecting to "),
        print("%s \n") % (mac)
        child.sendline("connect {0}".format(mac))
        child.expect("Connection successful", timeout=2)
         # print("Connected!\n")

        # Write Info. Channel, Preamble Code, Tag id, Anchors’ coordinates:
        command = "char-write-req 0x000e 0%s0%s0%s" %(ch,pc,Tid)
        print("Writing new parameters")
        child.sendline(command)
        child.expect("Characteristic value was written successfully", timeout=2)
        # time.sleep(1)

        command = "char-write-req 0x0011 0%s%s%s" %(i0,h1,h2)
        print("Writing new parameters")
        child.sendline(command)
        child.expect("Characteristic value was written successfully", timeout=2)

       .
       .
       .

        # Disconnect from device.
        print("Disconnecting to"),
        print("%s \n") % (mac)
        if child.sendline("disconnect {0}".format(mac)):
            print("Disonnected!")
            mem[mac]=Tid
```

Finally, the Tag will set up the parameters in its FW as explained in section 5.1 and start the ranging and localization tasks.

5.2.3 Definition of the Cluster Area

Before continuing with the GW’s and SGW’s algorithms, in this section is explained how the delimitation of the Cluster area works.
As discussed in section 4.1, the clusters are defined in a x and y coordinates system. So, an algorithm, called “`zone2polygons`” will pick the coordinates, set in the cluster zones configuration file, that will define the four vertexes of the Cluster’s area, that later on will define the active zone for the tags to get localize inside that cluster.

```
def zone2polygons(zones):

    #creates polygons given the zones definitions
    for zone in zones:
        #get the polygon
        polygon_zone = zone['polygon']

        #prepare the polygon points
        polygon_vertices = []

        #iterate over vertices
        for vertex in polygon_zone:
            print vertex
            polygon_vertices.append([vertex['x'],vertex['y']])

        #create the polygon
        polygon_latlon = Polygon(polygon_vertices)

        #store it
        current_zones[zone['id']] = polygon_latlon
        zone_occupation[zone['id']]={}
```

5.2.4 Tag localization zone control

Now that the cluster zones are defined it is possible to check when tags get into a cluster or get out of them, through the information gotten by the localization task fulfilled by the tag itself.

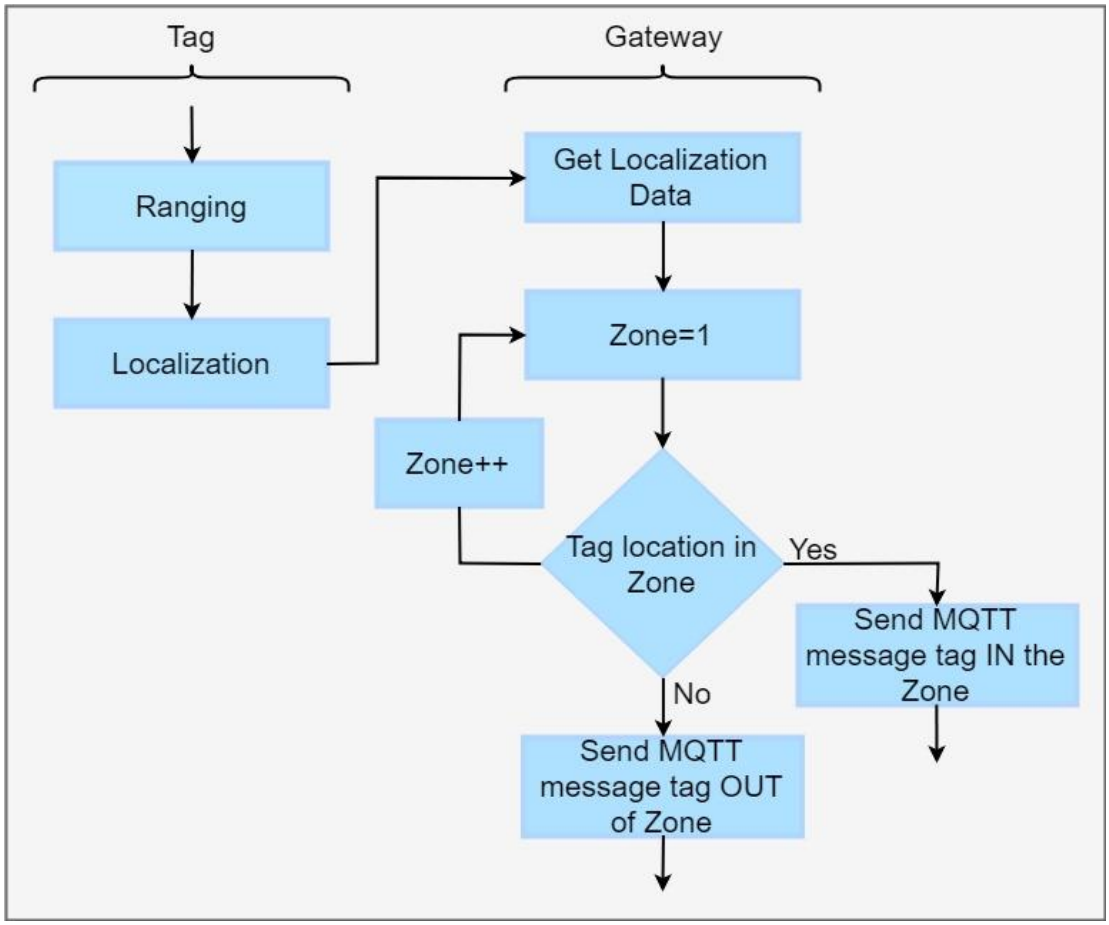

Figure 48. Zone control algorithm.

This location data is collected by the function "`GetDeviceData`" on the GW, that will at the same time publish it through MQTT.

```
                                # Take the Anchor connectivity
                                AncCOn = int(value[16:18],16)
                                # Take the estimated coordinates
                                Xpos = convert(value[24:26] + value[22:24] + value[20:22] + value[18:20])
                                Ypos = convert(value[32:34] + value[30:32] + value[28:30] + value[26:28])
                                Zpos = convert(value[40:42] + value[38:40] + value[36:38] + value[34:36])
                                # Take the estimated flag
                                EstimFlag = int(value[42:44],16)
                                # Estimate the number of measurements
                                Nmeas = get_bin(AncCOn,4).count("1")
                                # Creates the json_data
                                json_data = {}
                                json_data["y"]= Ypos
                                json_data["x"]= Xpos
                                json_data["idtag"]= Idtag
                                json_data["idgui"]= str(Idtag) + 'GW1'
                                json_data["mac"]= key
                                json_data["estimflag"]= EstimFlag
                                json_data["anchcon"]= AncCOn
                                string = json.dumps(json_data)

                                pubData="%s"%(current_config["GW_Name"]+current_config["mqtt_data_topic"])
                                mqttc.publish(pubData,string, 0)
```

Later, the algorithm will check if the location of the tag is in the current zone, or in any of the other ones. Doing this it will determined if the tag gets out of the cluster and gets into another one of the adjacent clusters. This information then will be sent to the SGW to manage the switching zones algorithm.

```
for zone in current_zones:

        if(current_zones[zone].contains(location_point)):
                try:
                        #try to check if the id tag was not yet in the zone
                        if(zone_occupation[zone][id] == False):

                                print "Entered into zone " + str(zone)
                                switch_msg = {}
                                switch_msg["Address"] = dev.addr
                                switch_msg["Zone"] = zone
                                switch_msg["State"] = "IN"
                                msg_string = json.dumps(switch_msg)


                                #publish on mqtt
                                print current_config["GW_Name"]+current_config["mqtt_alert_topic"]
                                pubData = current_config["GW_Name"]+current_config["mqtt_alert_topic"]
                                result = mqttc.publish(pubData,msg_string, 0) # Publish Data as data_topic

                                #update the zone occupation
                                zone_occupation[zone][id] = True

        else:
                try:
                        #try to check if the tag was already in the zone and therefore shall be removed
                        if(zone_occupation[zone][id]==True):

                                print "Out of zone " + str(zone)
                                switch_msg = {}
                                switch_msg["Address"] = dev.addr
                                switch_msg["Zone"] = zone
                                switch_msg["State"] = "OUT"
                                msg_string = json.dumps(switch_msg)


                        #publish on mqtt
                        print current_config["GW_Name"]+current_config["mqtt_alert_topic"]
```

```
                    pubData = current_config["GW_Name"]+current_config["mqtt_alert_topic"]
                    result = mqttc.publish(pubData,msg_string, 0) # Publish Data as data_topic

                    #update the zone occupation
                    zone_occupation[zone][id] = False
```

5.2.5 Switching clusters

At this point the MQTT message, or messages, from the zones control algorithm arrives, or arrive, to the SGW, where an algorithm will resolve if the MAC is going out a cluster and entering into another, or if by any case it received just a message of a MAC getting into a new cluster but without the message of the tag exiting a cluster.
It is important to precise that the first time that a tag gets localize into a cluster it will send a "IN Cluster" message, that will be ignore in the switching algorithm given that the tag is not changing from one algorithm to another, but has just awaken in the cluster for the first time.

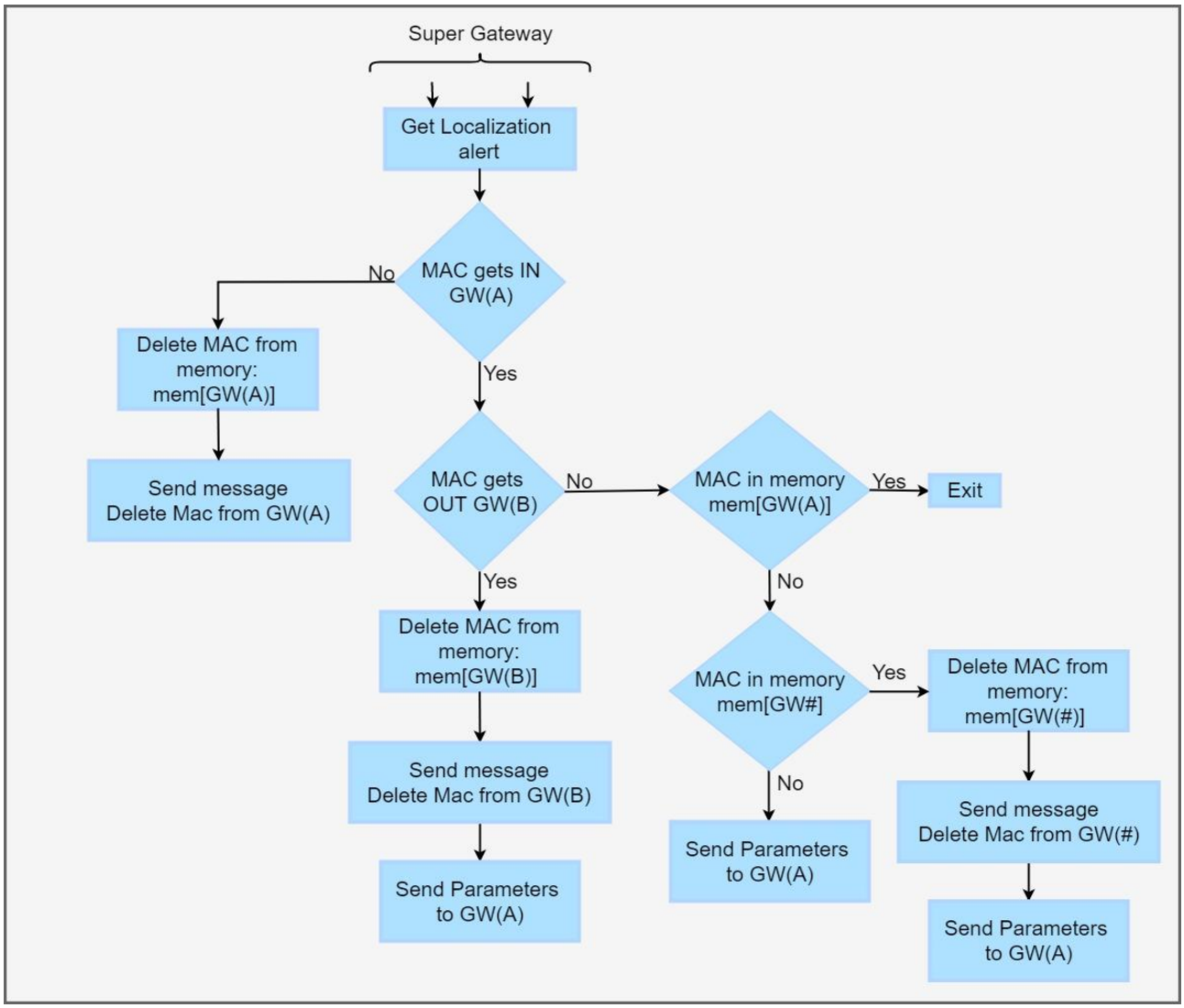

Figure 49. Switching algorithm.

Basically, the algorithm will receive a Localization alert with an "IN" message, then it will check if the MAC address of the tag is already saved in the memory of devices associated to that same cluster and it will rise and exception and exit.

```
                        if macId in mem["Data"][valueGWIn]:
                          raise Found
```

From then on, the localization zone control will send an “OUT” or an “IN” message just when the tag exits its current cluster and it enters into one of the adjacent ones. Once the switching algorithm receive the message it will check if it is an IN or OUT message.
If it is OUT it will continue to delete the MAC address from the memory of the SGW associated to the origin cluster and will send an MQTT message to the GW of given cluster to delete from its memory too the same MAC address.

```
                        ###Send MQTT to GW to delete mac from memory
                        if macId in mem["Data"][valueGWOut]:
                            deleteMac(valueGWOut, macId)
```

If it is IN, then it will check if any other message received contained an OUT for the same MAC:

- If not then it first will check if the MAC was already saved in the memory for that Cluster (this is the previous explained case), if not it will run a little algorithm to check if that MAC address was saved in the memory of any other cluster of the system:
    - If it was not, then the SGW will send the preamble code, and Tag id suitable for the new cluster in which the tag is getting into.
    - If it was, then it will delete the MAC address from the memory associated to that Cluster, proceeding to send an MQTT message to the respective GW to delete from its memory given MAC address. Afterwards it will send the preamble code, and Tag id suitable for the new cluster in which the tag is getting into.

```
                for key in switching["Zones"][GW][macId]:
                    if key == "IN":
                        valueGWIn = "GW%s" %(switching["Zones"][GW][macId]["IN"])

                        if "OUT" not in switching["Zones"][GW][macId]:

                            if valueGWIn not in mem["Data"]:
                                mem["Data"][valueGWIn] = []

                            if macId in mem["Data"][valueGWIn]:
                              raise Found

                            else:

                                if valueGWIn != GW:
                                    for key3 in mem["Data"]:
                                        if key3 != valueGWIn:
                                            if macId in mem["Data"][key3]:
                                                ###Send MQTT to GW to delete mac from memory
                                                deleteMac(GW, macId)

                                    ###Send new GW parameters to tag
                                    if valueGWIn == "GW2" or valueGWIn == "GW4" or valueGWIn == "GW6":
                                        preamCode = 03
                                    else:
                                        preamCode = 04

                                    if valueGWIn in mem["Data"]:
                                        if macId not in mem["Data"][valueGWIn]:
                                            mem["Data"][valueGWIn].append(macId)
                                    else:
                                        mem["Data"][valueGWIn] = []
                                        mem["Data"][valueGWIn].append(macId)

                                    if mem["Data"][valueGWIn].index(macId) < MaxTag:
                                        time.sleep(2)
                                        sendData(valueGWIn, macId, preamCode)
```

- If yes it will delete the MAC address from the memory associated to that Cluster, proceeding to send an MQTT message to the respective GW to delete from its memory given MAC address. Afterwards it will send the preamble code, and Tag id suitable for the new cluster in which the tag is getting into.

```
                    else: #key==OUT
                        valueGWOut = "GW%s" %(switching["Zones"][GW][macId]["OUT"])

                        if valueGWOut not in mem["Data"]:
                            mem["Data"][valueGWOut] = []

                        ###Send MQTT to GW to delete mac from memory
                        if macId in mem["Data"][valueGWOut]:
                            deleteMac(valueGWOut, macId)

                        ###Send new GW parameters to tag
                        if valueGWIn == "GW2" or valueGWIn == "GW4" or valueGWIn == "GW6":
                            preamCode = 03
                        else:
                            preamCode = 04

                        if valueGWIn in mem["Data"]:
                            if macId not in mem["Data"][valueGWIn]:
                                mem["Data"][valueGWIn].append(macId)
                        else:
                            mem["Data"][valueGWIn] = []
                            mem["Data"][valueGWIn].append(macId)

                        if mem["Data"][valueGWIn].index(macId) < MaxTag:
                            time.sleep(2)
                            sendData(valueGWIn, macId, preamCode)
```

5.2.6 Connectivity control

Another algorithm that was implemented in the SGW was the one called "`noPositionTimeout`" that will delete the devices that has not send a valid location data for a period of time defined "`elapsedTime`", define in 15 seconds, for the case of a failure during the BLE connection between the GW and the tag, or the case in which the tag is not able to connect to the minimum quantity of anchors require to estimate successfully its position. So, this algorithm will activate a timer for every tag registered in the SGW memory, that will delete the tag from this memory when the timer expires.
This timer gets activated once the SGW sends to the GW the parameters to be assign to the tag for the cluster in which was found, and this timer will be reset every time the SGW gets a valid position of the tag corresponding to that timer.

This algorithm makes sure that if by any case a tag disappears from the network (e.g. in case that it s turned off, or that the battery died) the MAC address will be deleted from the GW as well as the SGW, giving the possibility to this tag to be localize once again it turns back to be operative.

In case that the position updates stop arriving to the SGW, the timer will expire and it will activate the procedure to delete the Mac address of that tag from the whole system. First by deleting it from the memory of the SGW where the devices, with their respective cluster name, were saved. Afterwards it will send a deletion broadcast message. This means that it will send a request to delete the tag from the memory of every GW connected to the network.

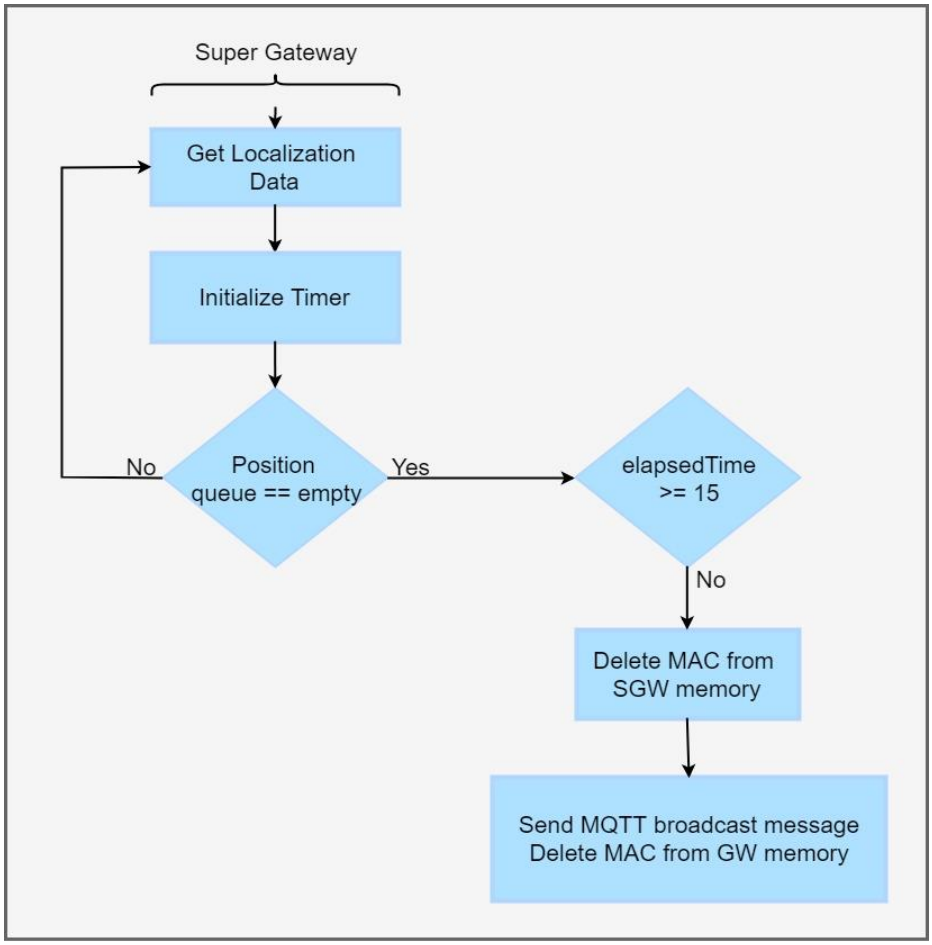

Figure 50. Position connectivity control algorithm.

5.3 Sequence diagrams validation

In this section are explained the results for the use cases proposed in section 4.2.4 for the validation of the proposed solution of the Large-Scale system.

5.3.1 Use case 1: Successful Initialization

For the first case it was proposed that the tag after awakening for the first time, it had to get the parameters for the current cluster in which it is located, just by means of a right measure of its RSSI.

To test the system the tag was located, before being turned on, near the limit between two clusters. Having the chance that the SGW had gotten very similar values of RSSI from two different GWs. As seen in Figure 51 the tag was registered as “0GW2”. Which means that the tag id is the number 0 (for being the first tag to be registered in that empty cluster) and it is located inside the cluster of the GW number 2. This means that the SGW got a higher value of the RSSI from the Cluster 2 than the Cluster 1, assigning then the parameters to the tag via BLE.

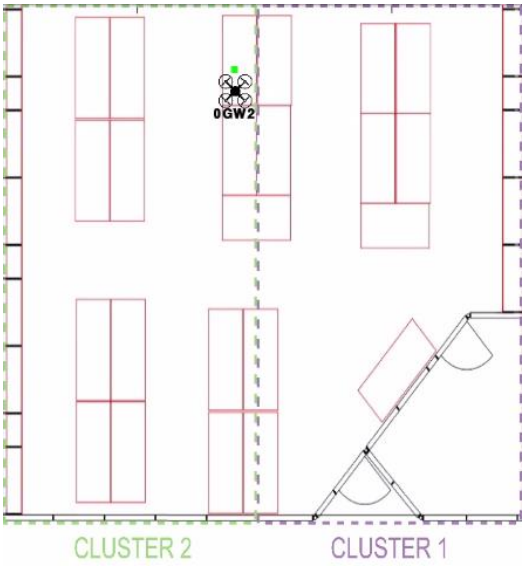

Figure 51. Successful Initialization.

5.3.2 Use case 2: Successful Initialization within a wrong cluster

For this particular case the tag was located, again, very near to the frontier between Cluster 1 and Cluster 2. Creating in this way a situation in which the SGW received a higher RSSI from the Cluster 1 even if the tag was located in the Cluster number 2.
So, as explained in section 5.2.4 the Zone control algorithm detected that the tag was located in a different zone from the one that he got the set-up parameters. This algorithm sent an alert to the SGW, that activated the Switching algorithm, that proceeded to send the set-up parameters of Cluster 2 to the tag.
In Figure 52(a) is shown how, in a first instance, the tag was assigned with the name "0GW1", meaning that it was assigned to the GW of the Cluster 1. In a second instance, Figure 52(b), the tag has the name "0GW2", proving that it has been relocated to the GW of the Cluster 2.

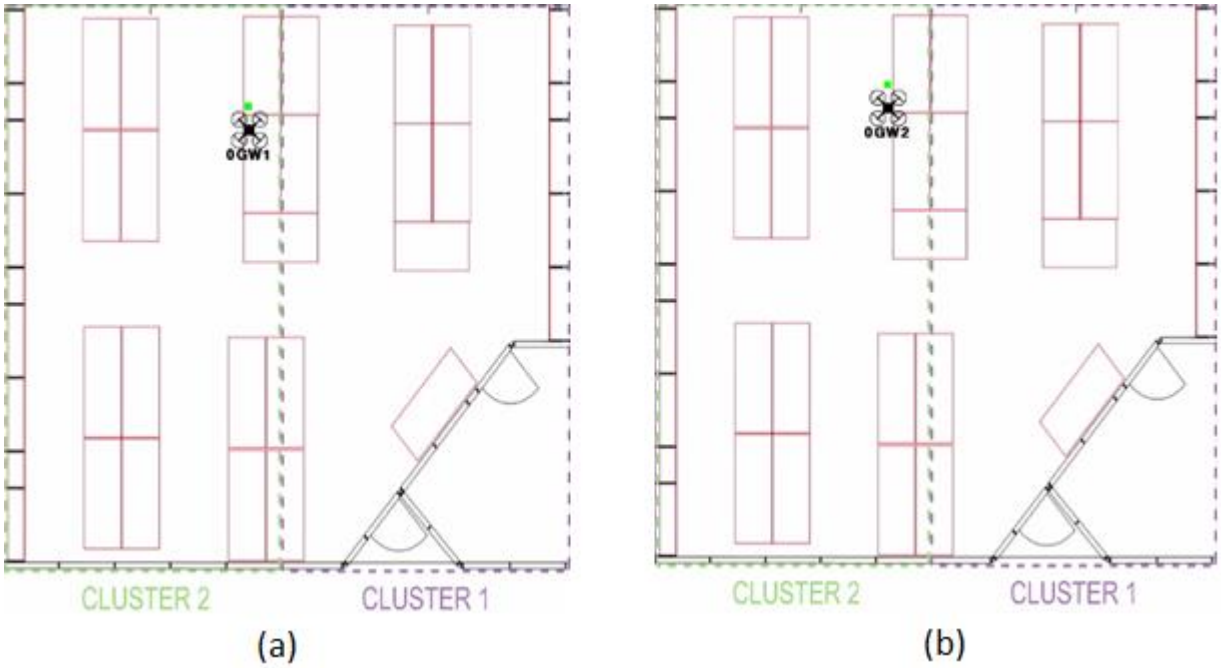

Figure 52. Successful Initialization within a wrong cluster.

5.3.3 Use Case 3: Unsuccessful Initialization with switching request

For this case it was implemented the algorithm explained in section 5.2.6 of Connectivity control. In which the SGW sent the set-up parameters, for the tag, to the GW and in that precise moment a time out countdown of 15 second started for receiving a position estimate of the MAC address of the tag. After this period when no answer was received the SGW erased the MAC address from its register, and sent a MQTT broadcast message to get the tag removed from every GW registered in which it was saved.
All of this is not reflected in the GUI, given that it shows just exclusively when a tag successfully estimates its position.

5.3.4 Use Case 4: Switching request from an already initialized tag

The last case represents the situation in which an already allocated tag, by moving, changes its position from one cluster to another. For this aim the algorithm "Switching clusters", explained in section 5.2.5, was activated in the SGW, after that the "zone control" algorithm, explained in section 5.2.4, sent an alert of the tag going out of the current cluster and getting in the adjacent one.

As seen in Figure 53(a), the tag named "0GW2", with id "0" in Cluster 2, moved to Cluster 1. In figured 53(b) is shown the tag relocated with the name "0GW1", meaning that it is now register on Cluster 1, after a successful zone switching.

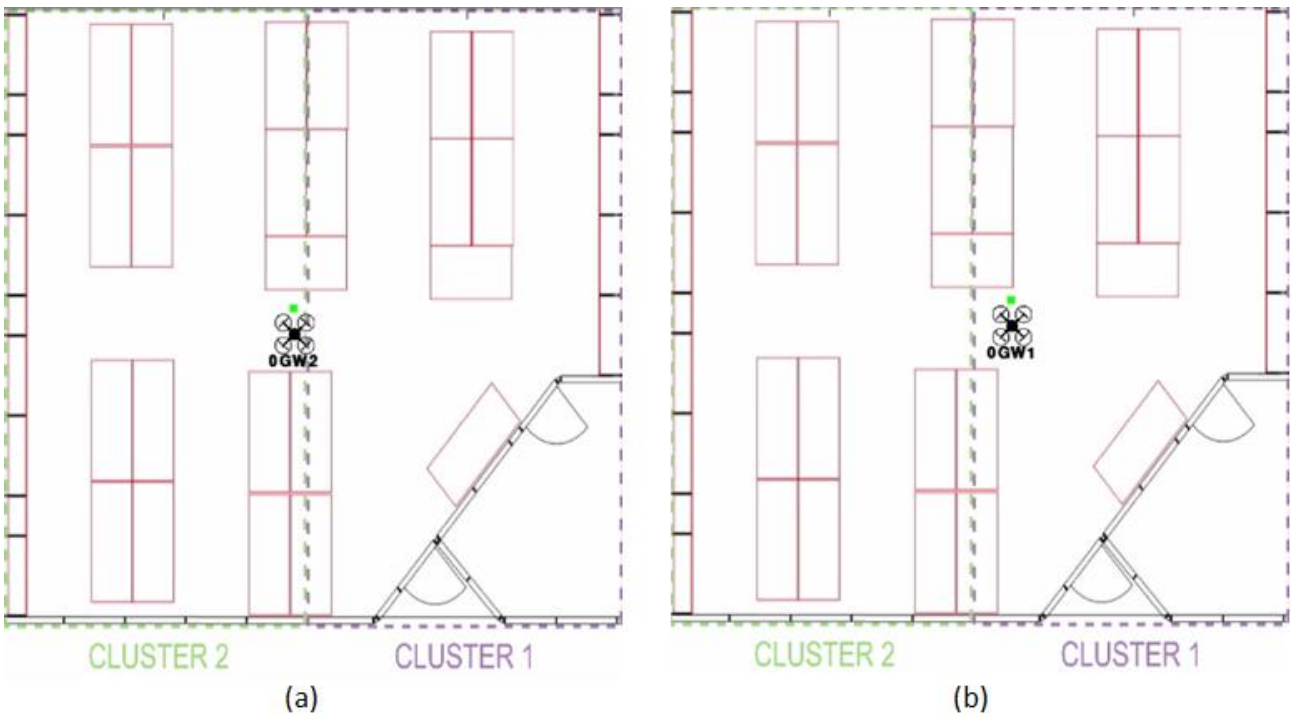

Figure 53. Switching request from an already initialized tag.

Conclusions

This thesis was proposed with the aim to design and implement a scalable UWB-based indoor localization system for large scale scenarios. The design and development were based in a pre-existing localization system, operating just in a small area, defined by the devices connectivity. The aim was to propose, in a non-intrusive way, a system where more than one localization area were deployed and they could interact, thus creating a unique large system.

The design covered the creation of "clusters" as small localization areas, as well as providing smart gateways (GW) to control them by means of the Bluetooth low energy (BLE) technology. The BLE interface was used to communicate and control tag nodes from GWs. Finally, the GWs were connected, through a WLAN interface, to a Super GW (SGW), that had the function of operating as central location engine (CLE), meaning that it managed the data flowing in the overall system.

The design of the large-scale system was based on three architectural views (functional, deployment and information flow) to later define its implementation. The *Functional Architecture* defined all functionalities, services, processes of the system. Starting from the BLE messages exchange between the GW and the UWB tags, as well as the data exchange between the GW and the SGW, through the messaging transport protocol MQTT. The *Deployment Architecture* defined how the system was designed to be implemented in terms of hardware. It was proposed the use of the Raspberry Pi 3 as computer running the software components of GW and SGW. Finally, The *Information Flow* diagram shows what kind of messages and how interact system components among them. This was very important for the, later on, implementation of the system.

Besides, there were defined four sequence diagrams aiming to demonstrate the behaviour of the system under stress situations:

1. **Successful initialization:** This use case had the aim to test and verify the correct configuration of a tag inside a cluster, at the very first time it gets the set-up parameters from the GW. It means that a tag got the parameters, such as Preamble Code and tag's id from the SGW through the GW. These parameters were sent through a BLE connection to the Tag. At this point the tag was able to perform ranging with the anchors and sent to the system its own position.
2. **Successful Initialization within a wrong cluster:** It happened that a tag, at the moment of its awakening, was in a position very near to the limit between two adjacent clusters. Getting a higher RSSI reading from the gateway of the adjacent cluster of its proper location. Meaning that the SGW sent him the parameters of initialization of a wrong cluster. The tag was able to range to the anchors of the wrong cluster being able to calculate its own position. Immediately, it was sent a switching alert to the SGW, asking to send the parameters of the right cluster
3. **Unsuccessful Initialization with switching request:** The tag was not able to perform ranging with the anchors, it could be caused by many different reasons, such as interferences, or the tag was assigned to a wrong cluster from which it is impossible to connect to the anchors, etc. In this situation after 15 seconds the tag timer in the SGW expired, liberating the whole system of the knowledge of the existence of that tag. Subsequently, this tag performed every task from the beginning.

4. **Switching request from an already initialized tag:** When a tag, by moving, changed its position from one cluster to another a message was sent to the SGW to manage the switching of clusters.

In conclusion all of the use cases where validated through the algorithms implemented for the realization of the Large-scale system, for which the UWB-based Indoor Localization System for Large Scale Scenarios was successfully achieved.

References

[1] Dr. Brian Gaffney: “Considerations and challenges in real time locating systems design.”. Decawave Ltd, 2014.

[2] Sinan Gezici, Zhi Tian, Georgios B. Giannakis, Hisashi Kobayashi, Andreas F. Molisch, H. Vincent Poor, and Zafer Sahinoglu: "Localization via Ultra-Wideband Radios: A look at positioning aspects of future sensor networks", Proc. IEEE Signal Processing Magazine, vol. 70, 2005.

[3] N. Patwari, A.O. Hero III, M. Perkins, N.S. Correal, and R.J. O’Dea, “Relative location estimation in wireless sensor networks,’’ IEEE Trans. Signal Processing, vol. 51, no. 8, pp. 2137–2148, Aug. 2003.

[4] DecaWave: “APS003 Application Note: Real Time Location Systems. An Introduction”, DecaWave Ltd, 2014.

[5] Neal Patwari, Joshua N. Ash, Spyros Kyperountas, Alfred O. Hero III, Randolph L. Moses, and Neiyer S. Correal: "Locating the nodes: Cooperative localization in wireless sensor networks", Proc. IEEE Signal Processing Magazine, vol. 54, 2005.

[6] B.B. Peterson, C. Kmiecik, R. Hartnett, P.M. Thompson, J. Mendoza, and H. Nguyen, “Spread spectrum indoor geolocation,’’ *J. Inst. Navigat.*, vol. 45, no. 2, 1998.

[7] K. Pahlavan, P. Krishnamurthy, and J. Beneat, “Wideband radio propagation modeling for indoor geolocation applications,’’ *IEEE Commun. Mag.*, vol. 36, no. 4, Apr. 1998

[8] N.S. Correal, S. Kyperountas, Q. Shi, and M. Welborn, “An ultra wideband relative location system,’’ in *Proc. IEEE Conf. Ultra Wideband Systems and Technologies*, Nov. 2003.

[9] E.G. Larsson, “Cramér-Rao bound analysis of distributed positioning in sensor networks,’’ *IEEE Signal Processing Lett.*, vol. 11, no. 3, Mar. 2004.

[10] Y. Qi, “Wireless geolocation in a non-line-of-sight environment,” Ph.D. Dissertation, Princeton University, Dec. 2004

[11] Sinan Gezici, Ismail Guvenc, and Zafer Sahinoglu: "On the Performance of Linear Least-Squares Estimation in Wireless Positioning Systems", Proc. IEEE International Conferences On communications.

[12] A. H. Sayed, A. Tarighat, and N. Khajehnouri, “Network-based wireless location,” *IEEE Signal Processing Mag.*, vol. 22, no. 4, July 2005.

[13] G. F. Welch and G. Bishop, “An introduction to the kalman filter,” University of North Carolina, Chapel Hill, NC, USA, Tech. Rep., 1995

[14] Mauricio A. Caceres, Francesco Sottile, and Maurizio A. Spirito: "Adaptative Location Tracking by Kalman Filter in Wireless Sensor Networks", Proc. IEEE International Conferences On Wireless and Mobile Computing, Networking and Communications.

[15] C. Savarese, J.M. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks,'' in *Proc. IEEE ICASSP*, May 2001..

[16] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling,'' in *Proc. IEEE INFOCOM*, 2004.

[17] DecaWave: "APS016: Moving from TREk1000 to a product: A discussion of the elements to consider when developing a commercial product based on the TREK1000 Two-Way-Ranging (TWR) RTLS IC Evalution kit", DecaWave Ltd, 2015.

[18] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks (WPANS. Amendment 1: Add Alternate PHYs." *IEEE Computer Society*, Annex D.

[19] Federal Communications Commission, "First report and order 02-48",2002.

[20] A. J. Viterbi, CDMA: Principles of Spread Spectrum Communications. New York: Addison-Wesley, 1995.

[21] ECMA, "Uwb: High rate ultra wideband phy and mac standard," tech. rep., 2005. www.ecma-international.org. Eirini Karapistoli, and Fotini-Niovi Pavlidou: "An Overview of the IEEE 802.15.4a Standard", Proc. IEEE Communications Magazine.

[22] M. Z. Win and R. A. Scholtz, "Ultra-Wide Bandwidth Time-Hopping Spread-Spectrum Impulse Radio for Wireless Multiple-Access Communications," *IEEE Trans. Commun*., vol. 48, no. 4, Apr. 2000.

[23] Andreas F. Molisch, Philip Orlik, Zafer Sahinoglu, and Jin Zhang: "UWB-based sensor networks and the IEEE 802.15.4a standard: a tutorial", Proc. First international conference on Communications and Networking in China,2006.

[24] B. SIG, "Bluetooth Core Specification 4.2," Bluetooth SIG, 5209 Lake Washington Blvd NE, Suite 350, Kirkland, WA 98033, USA, Tech. Rep., 2014

[25] Texas Instruments. CC2540 and CC2541 *Bluetooth*® low energy Software Developer's, Reference Guide, Oct 2010, Revised Sept 2015.

[26] Oasis, MQTT version 3.1.1, Oasis Standar, Oct 2014.

[27] DecaWave: DW1000 Datasheet Version 2.12. Decawave Ltd 2016.

[28] DecaRanging Ranging Demo Application (PC Version) User Guide", DecaWave Ltd, 2013

[29] Orlando Tovar. Thesis: "Design and Implementation of an Indoor Localization System based on IEEE 802.15.4a Devices". Politecnico di Torino. 2014.

[30] Mario Alexander Ruiz Guirales. Thesis: "Design and Implementation of a Cooperative Indoor Localization System based on UWB Devices". Politecnico di Torino. 2016.

[31] J. Borrás, P. Hatrack, and N.B. Mandayam, "Decision theoretic framework for NLOS identification,'' in *Proc. IEEE VTC 1998*, vol. 2, pp. 1583–1587, May 1998.

[32] DecaWave: "APS016: Moving from TREk1000 to a product: A discussion of the elements to consider when developing a commercial product based on the TREK1000 Two-Way-Ranging (TWR) RTLS IC Evalution kit", DecaWave Ltd, 2015.